When Threshold Meets Anamorphic Signatures: What is Possible and What is Not!

Hien Chu[2,4], Khue Do[1], Lucjan Hanzlik[1], Sri AravindaKrishnan Thyagarajan[3]

[1] CISPA Helmholtz Center for Information Security, Germany
[2] Friedrich-Alexander Universität Erlangen-Nürnberg, Germany
[3] University of Sydney, Australia
[4] TU Wien, Austria

Abstract. Anamorphic signatures allow covert communication through signatures in environments where encryption is restricted. They enable trusted recipients with a double key to extract hidden messages while the signature remains indistinguishable from a fresh and regular one. However, the traditional notion of anamorphic signatures suffers from vulnerabilities, particularly when a single recipient or sender is compromised, exposing all hidden messages and providing undeniable proof that citizens are part of the anamorphic exchange.

To address these limitations, we explore a threshold-based approach to distribute trust among multiple recipients, preventing adversaries from decrypting anamorphic messages even if some recipients are compromised. Our first contribution is the formalization of the notion of *threshold-recipient anamorphic signatures*, where decryption is possible only through collaboration among a subset of recipients. We then explore a *stronger model* where the dictator controls the key generation process through which it learns all secret keys and how citizens store cryptographic keys. A particular example of this model in the real world is a dictator providing citizens with electronic identity documents (eIDs) and blocking all other usage of cryptography. We demonstrate that anamorphic communication is still possible even in such a scenario. Our construction is secure against post-quantum adversaries and does not rely on any computational assumptions except the random oracle model. Finally, we show an *impossibility result* for encoding anamorphic messages with a threshold-sender model when using many existing threshold signature schemes and the adversary is part of the signing group. Our work outlines both the possibilities and limitations of extending anamorphic signatures with threshold cryptography, offering new insights into improving the security and privacy of individuals under authoritarian regimes.

1 Introduction

Anamorphic signature [28] is a cryptographic primitive designed to embed hidden messages within digital signatures while preserving their authenticity. This mechanism is particularly useful in environments where encryption is restricted, such as under authoritarian regimes. In such scenarios, anamorphic signatures allow a sender to covertly include a hidden message (referred to as the *anamorphic message*) within an otherwise standard signature. Crucially, this signature remains

indistinguishable from one without a hidden message, even to adversaries such as governments that can demand access to cryptographic keys and messages. Only trusted users, possessing a special *double key*, can extract the concealed message.

This cryptographic primitive builds on prior ideas but applies them to signature schemes. The first proposed anamorphic primitive was anamorphic encryption [36], which allowed users to create an additional covert channel inside a regular ciphertext. Therefore, even if the dictator requested access to the decryption key, it would only learn the benign and not the hidden message. A crucial requirement for both anamorphic encryption and signatures is that standard schemes support this mode of operation. The motivation behind this is that even if a regime mandates users to disclose their encryption keys, these policies are ineffective because there is always a way to implement an anamorphic covert communication channel while relying on standardized (e.g., by NIST) cryptographic encryption and signature schemes. Many follow-up works have further studied and refined the notions of anamorphic encryption and signatures [2, 7, 8, 37, 41].

In more detail, anamorphic signatures exploit the randomness inherent in the signing process, allowing a secret message to be embedded without altering the verification of the signature. Two methods of embedding anamorphic messages into signatures were proposed in [28]. The first solution assumes that the signing key is shared between the sender and the recipient. It also relies on a signature scheme that, having access to the secret key and a signature, allows one to extract the random coins used during signature generation. Examples of such signatures include Schnorr signatures and (EC)DSA. For the former, given a signature $s = r + \mathsf{H}(g^r, m) \cdot \mathsf{sk}$ and signing key sk, it is possible to retrieve the randomness r. The sender can then use any symmetric key encryption scheme (e.g., AES), setting the resulting ciphertext as the randomness r. From the signature, the recipient computes r and uses the double key to decrypt the AES ciphertext to obtain the anamorphic message.

The second method is more versatile as it does not require the recipient to know the sender's signing key. This is especially relevant in scenarios where users are provided with hardware components (e.g., electronic identity documents or eIDs) by an authoritarian entity, which stores the key and generates signatures. In this case, the sender can take advantage of the signing process's randomness and employ a technique similar to rejection sampling-sampling multiple signatures for the same message until one satisfies a specific predicate. For example, the predicate could be that the last bits of the signature match the anamorphic message. Unfortunately, this approach allows the encryption of significantly fewer bits than the first method for the same parameters of the signature scheme. To overcome this limitation, the sender can use multiple signatures to encode a single payload, which in this case is a symmetric encryption ciphertext.

A significant weakness of the current anamorphic signature model is its single point of failure. If a dictator compromises either the sender or the recipient, they can decrypt all exchanged anamorphic messages. Worse, if the sender is compromised, the dictator can produce undeniable proof—the anamorphic double key—to incriminate the recipient, as the key is symmetric. Additionally, existing

research on anamorphic cryptosystems has focused almost exclusively on peer-to-peer communication, overlooking scenarios with multiple participants. However, modern internet protocols frequently involve multi-party interactions, such as group messaging, broadcast channels, and threshold cryptography. To ensure security in real-world applications, anamorphic cryptosystems must be designed and analyzed with these multi-party settings in mind.

A promising solution to the aforementioned single point of failure problem is the distribution of "trust" among multiple parties similar to threshold encryption [18, 19] and signature schemes [34, 35]. In threshold encryption, a secret key is shared among several users, and only a subset (or threshold) of them needs to cooperate to decrypt a message, ensuring that no single entity has complete control over the key. Similarly, in threshold signatures, the signing capability is split across a group, where a minimum number of participants must collaborate to produce a valid signature. This paper discusses trust distribution in anamorphic signatures and employs the threshold primitives described above.

We explore a scenario where multiple recipients collaborate to decrypt an anamorphic message, such as through threshold decryption. Unlike standard anamorphic signatures, this approach prevents a dictator from targeting individual recipients, even if the sender is compromised. Additionally, if some recipients are compromised, the dictator still cannot access the message without cooperation from the remaining recipients. A key question arises: how do these parties coordinate, given that previous anamorphic communication models were strictly peer-to-peer? Below, we present two practical use cases that address this challenge.

Multiple Decrypting Devices In a peer-to-peer setting, a recipient can distribute decryption across multiple devices it owns (e.g., smartphone, laptop, hardware token). This eliminates a single point of failure—if one device is compromised, the message remains secure, and the dictator cannot distinguish if the recipient participates in the anamorphic exchange. Moreover, the recipient can involve the devices of trusted individuals (e.g., family members), ensuring confidentiality even if some are compromised. This setup enhances security by requiring multiple devices to collaborate for decryption.

Distributed Whistleblowing Mechanism This model applies when a sender needs to securely communicate with multiple recipients who can organize themselves, such as a whistleblower sharing information with journalists. The sender publishes an anamorphic message as an anamorphic signature, while recipients define their security policies for decryption, e.g., requiring in-person meetings or secure data exchanges. This approach ensures:

- Sender anonymity, upheld by the anamorphic signature scheme.
- Data confidentiality, is enforced through a threshold-based access control policy among recipients.

This framework is particularly relevant for whistleblowing, where sensitive data must be securely shared while protecting sender and recipient identities.

In the second scenario, we tackle the challenge of a dictator using advanced techniques to detect anamorphic signing. For instance, side-channel attacks could expose anamorphic signatures by exploiting the fact that they take longer to

generate than standard signatures. To avoid detection, the sender could use a multi-party protocol like threshold signatures, blending in with non-anamorphic signers. In this setup, the dictator may identify the signers' group but can not pinpoint the specific anamorphic sender. This scenario is especially relevant given the increasing importance of threshold signatures, which IETF and NIST are currently standardizing. Concretely, our paper asks the following question.

Is it possible to distribute trust and improve the security of individuals in anamorphic signatures? What can be done, and what is impossible?

We provide a positive answer to the first part of our research question by introducing an extension of anamorphic signatures called *threshold-recipient anamorphic signatures*, and show how to construct such schemes generically. We then expand this notion to consider a powerful dictator who, for example, controls the generation of all the keys in the system. Even in this extreme case, we show that secure anamorphic message exchange remains possible, with post-quantum security, independent of the keys held by the sender and recipients. However, on the negative side, we show that anamorphic messages cannot be encoded in many existing threshold signature schemes if the dictator is one of the signing parties.

1.1 Anamorphic Threshold Recipient

Formalization and generic construction. We formalize the notion of threshold-recipient anamorphic signatures (in Section 3) as a natural extension of base anamorphic signatures where recipients act as decryptors similar to the threshold encryption scenario. In our model, we consider asymmetric double keys, i.e., the double key held by the sender can be viewed as a public key corresponding to the set of recipients while each recipient holds unique double key used for decryption. The sender creates the signature using its signing and double keys, while the recipient can use the signature and their double key to obtain so-called decryption shares. We leave the actual way of exchanging decryption shares outside the model (similar to threshold encryption) but discuss potential solutions.

In this model, we enhance the adversary's capabilities compared to the standard anamorphic signature setting. Here, the adversary can choose the sender's signing key pair and can compromise the sender's double key as well. Additionally, the adversary has access to a corrupt oracle, from which it can obtain the double keys of users— this models scenario where the dictator captures a recipient and forcibly acquires their keys. Interestingly, unlike standard anamorphic signatures, we observe that the indistinguishability of real and anamorphic signatures (i.e., anamorphic property) does not imply the CPA security of the anamorphic message. We delve deeper into this observation in Section 6. We model the CPA-like security via an experiment in which the adversary with access to the mentioned corrupt oracle specifies two anamorphic messages, sees the anamorphic signature, and needs to decide which message was encoded.

Finally, we show how to instantiate the threshold-recipient model generically, using standard threshold encryption and a basic anamorphic encryption scheme.

Extended real-world setting. We extend the threshold recipient model with an even more powerful adversary (in Section 4). We observe that in the real world, a dictator can force citizens to rely on specialized hardware to create signatures, generate and store cryptographic keys. Moreover, the dictator can also enforce that no other keys outside of those stored inside the specialized hardware can be used by regularly inspecting the device. While this might seem like a scenario limited to authoritarian regimes, it's not. An existing real-world example is electronic identity documents (eIDs), which hold cryptographic keys. A regulatory body could mandate that all digital signatures be created exclusively via these eIDs. Even outside authoritarian contexts, device inspection is not uncommon—e.g., border control officers often have the authority to search personal devices capable of storing data [9, 10]. Such a scenario is becoming increasingly realistic, as a French scientist has been denied entry by border control due to the discovery of oppositional information during personal device checks [29].

To formally capture this model, we assume that the sender and recipients do not hold any other secret except the key pair generated and provided by the adversary. Unfortunately, since no anamorphic double key exists in such a setting, peer-to-peer anamorphic communication is impossible. Therefore, we explore the threshold recipient setting, where multiple parties decrypt the anamorphic message jointly. A real-world example of such a group could be a subset of journalists who receive whistleblowing information. Other examples are private group chats and secret societies.

The anamorphic message can be decrypted as part of an in-person ceremony, using some external secure channel, or by implementing the threshold parties as separate devices of one user. For the latter scenario, one could imagine that the dictator allows multiple devices (e.g., eID, smartphone, laptop), but all are equipped with pre-generated keys.

In the multi-recipient setting, a secret is unknown to the dictator, i.e., which recipient is part of the decryption process, i.e.,
- which citizen is part of the anamorphic group,
- which subset of journalists received the leak,
- who is in the private group chat or secret society.

As it turns out, this information is quite powerful. We construct a scheme that uses any non-deterministic signature scheme and the Diffie-Hellman (DH) key exchange to exchange anamorphic messages in this setting. We assume that the device allowed by the dictator provides a means to sign and create secure channels via a basing DH key exchange. It is worth noting that these are simple and widely supported requirements, mandated globally by ICAO and even met by eID documents, thereby reinforcing the strength of the results due to the non-restrictive nature of the proposed solution.

The security of our extended threshold recipient scheme cannot be based on the security of the underlying signature scheme or Diffie-Hellman since the adversary controls all keys and provides them to both the sender and recipient. In other words, we show that even if the dictator forces citizens to use specialized

and simple devices for cryptography and can inspect other citizens' devices, it is still possible to exchange messages anamorphically. Moreover, since the scheme does not rely on any classical computational assumption except the random oracle model, it is also post-quantum secure.

1.2 Anamorphic Threshold Sender

Impossibility result for threshold sender. We explore anamorphic properties in threshold signatures, a topic gaining traction among standardization bodies. A positive result for this primitive would enhance sender privacy, allowing senders to blend in with non-anamorphic users. To deepen the understanding of anamorphic threshold signatures, we first establish a concrete syntax and formal security definitions within the framework of anamorphic cryptography.

Unfortunately, we show impossibility results for a mainstream class of threshold signatures, particularly the scheme considered by NIST, i.e., FROST [27] and other schemes like MuSig2 [30], Sparkle [14], and TRacoon [38]. In Section 5, we show that if the dictator is part of the signing group, no anamorphic message can be encoded by any other participants. We cover the abovementioned schemes with an encoding that uses the signature as the base (see Figure 1 for classification). In other words, the impossibility results only hold for schemes that use the non-determinism of the signature to create a subliminal channel. The results are detrimental since a dictator can always argue that a 2 out of 2 threshold signing was introduced to provide citizens more security by providing a service storing a share of the signing keys. In fact, such systems were already considered and proposed for practical use (see mediated RSA [4]).

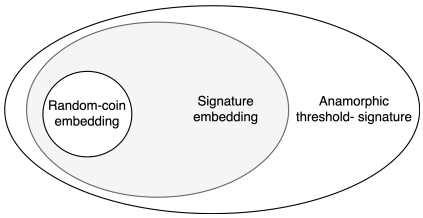

Fig. 1. *Random-coin embedding* is a class of all construction approaches where the covert message is embedded to the random coin of the signature, e.g., $\mathsf{ct} = \mathsf{E.Enc}(k, \mathsf{amsg}), r = \mathsf{ct}$ (with or without rejection sampling). The broader class, labeled *Signature embedding*, encompasses approaches where amsg is embedded in any components of the signatures. The outer region represents alternative approaches to anamorphic threshold signatures that do not belong to the signature embedding class. Our impossibility results are restricted to the signature embedding class.

To obtain the most substantial impossibility result, we consider the most restricted and honest adversarial model. The rationale is that if the impossibility holds even for a minimal and constrained adversary, it extends naturally to stronger adversaries. The adversary is semi-honest, meaning it follows the protocol correctly, ruling out trivial denial-of-service attacks. It can choose one single

corrupted signer at the start but not adaptively. Otherwise, a trivial distinguisher can detect anamorphic keys from a signer running anamorphic signing. The signing quorum's size is strictly controlled: smaller than the threshold, and signing is impossible; larger than the threshold, and honest signers can always form a sub-quorum that excludes the adversary. This approach rules out trivial failures of anamorphism by definition, allowing us to focus on more meaningful limitations arising from properties inherent to standardized constructions.

The intuition is that for all the mentioned schemes, the final signature and public key are those of the underlying standard digital scheme, which means that signers also secretly share the random coins. Therefore, the final signature that is output as part of the signing process looks like a uniformly sampled signature from the space of signatures that are valid for the message. We show that a dictator can always be the party that ensures this property, making it impossible for the anamorphic sender to encode meaningful messages correctly into the final signature. The only strategy left for the sender is to look at the final signature and decide not to publish it, similar to rejection sampling. However, the dictator can easily discover such a sender, making this approach impractical.

It is worth noting that schemes exist that do not fall into the impossibility result, and in fact, it turns out to be hard to identify strict rules that could be used as a simple litmus test for schemes. We leave this as an interesting open problem. An example of a threshold signature that still allows for anamorphism is a naive scheme where a threshold signature consists of t individual signatures of the threshold parties. Since the signatures are independent, randomness is also independent, and standard anamorphic signature techniques can be used even if the dictator is responsible for one of the other signatures.

Concrete classification and analysis. Our work involved systematically classifying the types of (threshold) signature schemes that allow or prevent embedding anamorphic messages as well as the embedding techniques. Previous works have focused on individual schemes, showing that certain specific constructions are anamorphic by, for instance, replacing the randomness with a one-time pad of the covert message. In contrast, we present (im)possibility results for entire classes of schemes rather than specific cases. In particular, the results in Section 3 and Section 4 are generic to any (randomized) signature scheme, while the former is generic to any threshold encryption. Moreover, this classification clarifies the scope of our impossibility results for threshold senders in two key ways:

- It categorizes anamorphic embedding approaches, as shown in Figure 1, demonstrating that our results in Section 5 exclude a broader class than traditional random-embedding methods.
- It provides an implicit structural classification of threshold signature schemes. Notably, most standardized candidates fall within this class.

It is important to note that, in contrast to the anamorphic user's goal of ensuring that anamorphism is supported by as many schemes as possible, the dictator's objective is the opposite: it suffices for him to identify a single scheme that prohibits anamorphic communication and enforce its mass adoption in public. Hence, it is

sufficiently interesting to show the impossibility of a state-of-the-art scheme such as FROST, especially given the dictator's incentives to make it standard. However, our results go further, revealing that it is not the algebraic specifics of a scheme but rather the signature-preserving property and distributed randomness, common to standardizing threshold signature candidates, that block anamorphism. This approach reduces the complexity of determining whether a scheme supports anamorphism by restricting the testing to its signature-preserving property rather than conducting a full security analysis. This shift allows us to avoid the intricate and often ad hoc analysis required for each individual scheme, which arises due to variations in their algebraic structure and syntactic differences, such as the number of rounds. Consequently, any scheme satisfying these structural properties is inherently unlikely to support anamorphism. Moreover, our classification facilitates a more systematic identification of potential approaches to overcome such limitations. As discussed later (in Appendix A.1), the only viable solution to overcome this impossibility is to embed the anamorphic message directly within the regular message, particularly in the use case of *stealth addresses* [13].

This paper is structured as follows. Section 2 provides the syntax and security definitions of related primitives. The core of the paper is divided into two parts: anamorphic signatures for threshold recipients (Section 3 and Section 4) and the anamorphic threshold signature for senders (Section 5). Section 3 defines the syntax and security model for threshold recipient anamorphic signatures and proposes a generic construction meeting these security requirements. Section 4 strengthens these security models and presents a concrete scheme achieving the proposed security guarantees. Finally, in Section 5, we establish the impossibility of incorporating the anamorphic property within a threshold signature scheme.

2 Preliminaries

Let $\lambda \in \mathbb{N}$ be the security parameter, and for integer n we define $[n] := \{1, \ldots, n\}$. We use uppercase letters $\mathcal{A}, \mathcal{B}$ to denote algorithms, and $y \leftarrow \mathcal{A}(x)$ denotes the output of $\mathcal{A}$ on input x. For a randomized algorithm $\mathcal{A}$, we use $y \leftarrow\!\!\$\, \mathcal{A}(x)$. To derandomize the algorithm $\mathcal{A}$, we write it explicitly as $y \leftarrow \mathcal{A}(x; \mathsf{st})$. We write $\mathcal{A}^{\mathcal{B}}$ to denote that $\mathcal{A}$ has oracle access to $\mathcal{B}$. We say a function is negligible and denote it as $\mathsf{negl}(\lambda)$ if it vanishes faster than any polynomial.

2.1 Digital Signature Scheme

Definition 1. *A digital signature scheme* $\mathsf{S} = (\mathsf{S.KGen}, \mathsf{S.Sign}, \mathsf{S.Vf})$ *consists of the following p.p.t. algorithms:*

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{S.KGen}(1^\lambda)$: *A key generation algorithm that on input security parameter* 1^λ*, outputs a public key* pk *and a secret key* sk*.*

$\sigma \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; \mathsf{st})$: *A signing algorithm that on input the secret key* sk*, the message* msg *and a random coin* st*, outputs the signature* σ*.*

$\{0, 1\} \leftarrow \mathsf{S.Vf}(\mathsf{pk}, \mathsf{msg}, \sigma)$: *A signature verification algorithm that on input the public key* pk*, the message* msg *and the signature* σ*, outputs* 1 *if* σ *is the valid signature and* 0 *otherwise.*

The definitions for correctness and unforgeability are provided in Appendix B.2.

We also consider the extractor functionality of a digital signature scheme S*, which consists of the following p.p.t algorithms:*

$\{\mathsf{st}, \perp\} \leftarrow \mathsf{RanExt}(\mathsf{sk}, \mathsf{msg}, \sigma)$: *The random coin extract algorithm that on input the secret key* sk*, the message* msg *and the signature* σ*, outputs the corresponding random coin* st *and* $\perp$ *otherwise.*

$\{\mathsf{sk}, \perp\} \leftarrow \mathsf{SKExt}(\mathsf{st}, \mathsf{msg}, \sigma)$: *The secret key extract algorithm that on input the random coin* st*, the message* msg *and the signature* σ*, outputs the corresponding secret key* sk *and* $\perp$ *otherwise.*

We say that S *is random coin extractable if, for all public parameters* 1^λ*, all messages* msg *in the associated message space, the following event holds:*

$$\Pr\left[\begin{array}{c} \mathsf{st} = \mathsf{st}' \\ \& \ \mathsf{sk} = \mathsf{sk}' \end{array} \middle| \begin{array}{c} \sigma \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; \mathsf{st});\ \mathsf{st}' \leftarrow \mathsf{RanExt}(\mathsf{sk}, \\ \mathsf{msg}, \sigma);\ \mathsf{sk}' \leftarrow \mathsf{SKExt}(\mathsf{st}, \mathsf{msg}, \sigma) \end{array} \right] = 1,$$

where the probability is over the random variable $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{S.KGen}(1^\lambda)$ *and the random coins of* $\mathsf{S.Sign}$.

One can easily verify that certain standard signature schemes like [22, 32, 40] or post-quantum candidates like [16, 21] are random coin extractable.

2.2 Threshold Cryptosystems

Definition 2 (Threshold Encryption). *A threshold encryption scheme* $\mathsf{TE} = (\mathsf{TE.KGen}, \mathsf{TE.Enc}, \mathsf{TE.Combine}, \mathsf{TE.Dec})$ *consists of the following p.p.t. algorithms:*

$(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TE.KGen}(1^\lambda, n, t)$: *A key generation algorithm that on input security parameter* 1^λ*, a number of participant* n *and a threshold* t*, outputs a public encryption key* pk *and* n *secret decryption keys* $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$.

$\mathsf{ct} \leftarrow \mathsf{TE.Enc}(\mathsf{pk}, \mathsf{pt})$: *An encryption algorithm that on input a public encryption key* pk *and a plaintext* pt*, outputs a ciphertext* ct.

$\mathsf{pdec}_i \leftarrow \mathsf{TE.Dec}(\mathsf{sk}_i, \mathsf{ct})$: *A partial decryption algorithm that on input a secret decryption key* sk_i *and a ciphertext* ct*, outputs a partial decryption* pdec_i.

$\{\mathsf{pt}, \perp\} \leftarrow \mathsf{TE.Combine}(\mathsf{pdec}_{i_1}, \ldots, \mathsf{pdec}_{i_t})$: *A partial decryption combination algorithm that on input a set of partial decryption* $(\mathsf{pdec}_{i_1}, \ldots, \mathsf{pdec}_{i_t})$*, outputs either a plaintext* pt *or* $\perp$.

The definitions for the correctness, semantic security, and pseudorandom ciphertext property of a threshold encryption scheme are provided in Appendix B.4. Multiple instantiations [3, 5, 6, 20] are thresholdized versions of Cramer-Shoup, LWE, or class group cryptosystems.

Definition 3 (Threshold Signature). *A 2-round threshold signature* $\mathsf{TS} = (\mathsf{TS.KGen}, \mathsf{TS.Sign}, \mathsf{TS.Combine}, \mathsf{TS.Vf})$ *consists of the following p.p.t. algorithms:*

$(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TS.KGen}(1^\lambda, n, t)$: *A key generation algorithm that on input security parameter* 1^λ*, a number of participant* n *and a threshold* t*, outputs a public key* pk *and* n *secret signing keys* $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$.

$(\mathsf{msg}_{i,1}, \mathsf{St}_{i,1}) \leftarrow \mathsf{TS.Sign}_1(\mathsf{sk}_i, \mathsf{msg}, \mathcal{J})$: *a randomized algorithm on input a secret key* sk_i*, a message* msg*, and a subset* $\mathcal{J}$ *of indices, returns a state* $\mathsf{St}_{i,1}$ *and a first message* $\mathsf{msg}_{i,1}$ *to be sent during round 1 of the protocol to all signers in* $\mathcal{J} \setminus \{i\}$.

$s_i \leftarrow \mathsf{TS.Sign}_2(\mathsf{St}_{i,1}, \{\mathsf{msg}_{j,1}\}_{j\in\mathcal{J}\setminus\{i\}})$: *a deterministic algorithm on input a state* $\mathsf{St}_{i,1}$ *and incoming messages* $\{\mathsf{msg}_{j,1}\}_{j\in\mathcal{J}\setminus\{i\}}$*, outputs a share* s_i*.*

$\{\sigma, \perp\} \leftarrow \mathsf{TS.Combine}(\mathsf{pk}, s_{i_1}, \ldots, s_{i_t})$: *A share combination algorithm that on input a public key* pk *and a set of signature shares* $(s_{i_1}, \ldots, s_{i_t})$*, outputs either a signature* σ *or* $\perp$*.*

$\{0, 1\} \leftarrow \mathsf{TS.Vf}(\mathsf{pk}, \mathsf{msg}, \sigma)$: *A signature verification algorithm that on input a public key* pk*, a message* msg *and a signature* σ*, outputs* 1 *if* σ *is the valid signature for* msg *and* 0 *otherwise.*

The definitions for the correctness and unforgeability of a threshold signature scheme are provided in Appendix B.5. We also consider the additional algorithms of secret-key aggregation and state aggregation of a TS scheme .

$\mathsf{sk} \leftarrow \mathsf{TS.SkAgg}(\{\mathsf{sk}_i\}_{i\in I})$: A secret-key aggregation algorithm that aggregates any t-subset of secret keys $\{\mathsf{sk}_i\}_{i\in I}$ to an aggregated secret key sk.

$\mathsf{st} \leftarrow \mathsf{TS.StAgg}(\{\mathsf{St}_{i,1}\}_{i\in I})$: A state aggregation algorithm that aggregates any t-subset of states $\{\mathsf{St}_{i,1}\}_{i\in I}$ to an aggregated state st.

Definition 4 (Signature-preserving). *We say that a* TS *scheme is signature-preserving w.r.t. a digital signature scheme* S *if it shares a common message, key, and state spaces with* S *and for all message* msg *and keys generated from* $\mathsf{TS.KGen}$*, the following holds:*

$$\begin{aligned}&\mathsf{TS.Combine}(\mathsf{pkc}, \{\mathsf{TS.Sign}_2(\{\mathsf{TS.Sign}_1(\mathsf{sk}_j, \mathsf{msg}, \mathsf{amsg}, \mathcal{J})\}_{j\in\mathcal{J}\setminus\{i\}})\}_{i\in\mathcal{J}}, \mathsf{St}_{i,1})\\ &\quad = \mathsf{Sign}(\mathsf{TS.SkAgg}(\{\mathsf{sk}_i\}_{i\in\mathcal{J}}), \mathsf{msg}; \mathsf{TS.StAgg}(\{\mathsf{St}_{i,1}\}_{i\in\mathcal{J}})) \qquad (1)\end{aligned}$$

All threshold signature schemes we consider in Section 5 are signature-preserving. We will provide corresponding secret-key and sate aggregation algorithms then for easy verification of Equation (1). Clearly, a signature-preserving threshold signature TS is w.r.t. a digital signature scheme S can always share the same verification algorithm with S.

2.3 Anamorphic Signature Scheme

Motivated by [2], we modify the syntax of the anamorphic signature scheme from [28] to decouple double key generation from signature key-pair generation. This allows on-the-fly double-key setup for an already deployed public key, supports multiple double keys for distinct covert channels, and strengthens the security model by ensuring that the double key and the signing key pair are uncorrelated. In this work, we focus on symmetric anamorphic signatures, where the anamorphic decryption algorithm additionally requires the signing key as input. As we mainly focus on the anamorphic communication application, where both sender and recipient are non-malicious, we omit unforgeability. In this setting, although the dictator holds the sender's signing key and can produce signatures, they cannot embed anamorphic messages, as the double key, uncorrelated with the signing key pair, is unknown to the dictator.

Definition 5 (Anamorphic Signature Scheme [28]). *An anamorphic signature scheme* $\mathsf{AS} = (\mathsf{AS.KGen}, \mathsf{AS.Sign}, \mathsf{AS.Dec})$ *associated with a digital signature scheme* $\mathsf{S} = (\mathsf{S.KGen}, \mathsf{S.Sign}, \mathsf{S.Vf})$*, a pair of sender's signing and verification keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{S.KGen}(1^\lambda)$ *consists of the following p.p.t. algorithm:*

$(\mathsf{adk}) \leftarrow \mathsf{AS.KGen}(1^\lambda)$: *The anamorphic key generation algorithm on input security parameter* 1^λ *outputs the double key* adk.

$\sigma^{\mathsf{a}} \leftarrow \mathsf{AS.Sign}(\mathsf{sk}, \mathsf{adk}, \mathsf{msg}, \mathsf{amsg})$: *The anamorphic signing algorithm on input a signing key* sk*, a double key* adk*, a regular (benign) message* msg*, and an anamorphic message* amsg*, outputs an anamorphic signature* σ^{a}.

$\mathsf{amsg} \leftarrow \mathsf{AS.Dec}(\mathsf{sk}, \mathsf{adk}, \sigma^{\mathsf{a}})$: *The anamorphic decryption algorithm on input a signing key* sk*, a double key* adk *and an anamorphic signature* σ^{a}*, outputs an anamorphic message* amsg.

The definitions for the correctness, anamorphism ($\mathsf{ExpAnam}-$ which ensures indistinguishability of the anamorphic from the standard signature), and semantic security ($\mathsf{ExpA\text{-}CPA}-$ which ensures protection for the embedded anamorphic message) of an AS scheme can be found in [28]. Concrete descriptions are also provided in Appendix B.6. For the unforgeability property, we refer to [28].

Theorem 1 (Theorem 10, [28]). *Any random coin extractable signature scheme (see Definition 1) is anamorphic.*

3 Threshold-Recipient Anamorphic Signatures

In this section, we introduce the notion of recipient-threshold anamorphic signatures. As mentioned in the introduction, one of the main bottlenecks of standard anamorphic signature is that once the dictator learns the receiver's double key adk, the anamorphic message amsg leaks. In the case of multiple receivers, the dictator can identify that someone is part of the anamorphic message exchange and decrypt the message by compromising a single recipient. To overcome this problem, we consider a multi-recipient scenario but assume that decryption is now a thresholded process where a subset of recipients must cooperate to decrypt amsg. We describe how such a system can work in more detail below.

We assume that all members of the anamorphic system went through a setup phase that generates their double keys, i.e., we consider the function $\mathsf{TRAS.KGen}(1^\lambda, n, t)$ that defines the threshold t out of n and outputs n double keys adk_i and a unique double key for the sender adk^S. In our generic construction of recipient-threshold anamorphic signatures, the sender's double key will be the public key for a threshold encryption scheme. Similar to the standard anamorphic signature, we assume that the signer can compute a signature encoding the anamorphic message amsg using $\sigma^{\mathsf{a}} \leftarrow \mathsf{TRAS.Sign}(\mathsf{sk}^S, \mathsf{adk}^S, \mathsf{msg}, \mathsf{amsg})$, where sk^S is the sender's signing key generated by $(\mathsf{pk}^S, \mathsf{sk}^S) \leftarrow \mathsf{S.KGen}(1^\lambda)$, and msg is an actual message. Once the anamorphic signature is provided to recipients, they can, using the sender's signing key sk^S, compute their decryption shares as $\mathsf{pdec}_i \leftarrow \mathsf{TRAS.Dec}(\mathsf{sk}^S, \mathsf{adk}_i, \sigma^{\mathsf{a}})$. Note that we do not assume here how the partial decryptions are exchanged. One strategy would be for users to join offline or use other means of communication. Another one, as we will see in the next section, would be to use standard anamorphic signatures that work without a double key to broadcast the pseudorandom partial decryption pdec_i to others. Given a quorum set $\mathcal{J}$ of partial decryption that fulfills the threshold bound, the user can compute the anamorphic message using $\{\mathsf{amsg}, \perp\} \leftarrow \mathsf{TRAS.Combine}(\{\mathsf{pdec}_i\}_{i \in \mathcal{J}}$.

For security, we consider two properties: anamorphism and semantic security (IND-CPA). The former property ensures that recipient-threshold anamorphic signatures are indistinguishable from the underlying standard digital signatures, i.e., the dictator is oblivious if an anamorphic message is sent. The latter property ensures that an adversary cannot distinguish the anamorphic message from a random one if it holds less than the threshold double keys, i.e., the dictator learns that some users are exchanging anamorphic messages but still cannot decrypt them. Contrary to anamorphic signature, IND-CPA security of TRAS does not follow from anamorphism. The reason is that in our notion, we provide the adversary with an additional corruption oracle that outputs the double key adk_i of a given recipient i. With this oracle, we want to model the adversary learning some double keys while still being below the threshold. We provide the formal model for recipient-threshold anamorphic signatures below.

3.1 Syntax and Security

Definition 6 (Threshold-recipient Anamorphic Signature Scheme.). *A threshold-recipient anamorphic signature* $\mathsf{TRAS} := (\mathsf{TRAS.KGen}, \mathsf{TRAS.Sign}, \mathsf{TRAS.Dec}, \mathsf{TRAS.Combine})$ *associated with a digital signature scheme* $\mathsf{S} = (\mathsf{S.KGen}, \mathsf{S.Sign}, \mathsf{S.Vf})$ *and a pair of sender's signing and verification keys* $(\mathsf{pk}^S, \mathsf{sk}^S) \leftarrow \mathsf{S.KGen}(1^\lambda)$*, consists of the following p.p.t. algorithm:*

$(\mathsf{adk}^S, \mathsf{adk}_1, \ldots, \mathsf{adk}_n) \leftarrow \mathsf{TRAS.KGen}(1^\lambda, n, t)$: *A anamorphic key generation algorithm on input security parameter* 1^λ*, the number of participants* n *and a threshold* t *outputs the sender double key* adk^S *and* n *recipient double keys* $(\mathsf{adk}_1, \ldots, \mathsf{adk}_n)$. [5]

$\sigma^{\mathsf{a}} \leftarrow \mathsf{TRAS.Sign}(\mathsf{sk}^S, \mathsf{adk}^S, \mathsf{msg}, \mathsf{amsg})$: *A anamorphic signing algorithm on input an signing key* sk^S*, a sender double key* adk^S*, a regular message* msg*, and an anamorphic message* amsg*, outputs an anamorphic signature* σ^{a}*.*

$\mathsf{pdec}_i \leftarrow \mathsf{TRAS.Dec}(\mathsf{sk}^S, \mathsf{adk}_i, \sigma^{\mathsf{a}})$: *A partial anamorphic decryption algorithm that, on input a secret key* sk^S*, a double key* adk_i*, and an anamorphic signature* σ^{a}*, outputs a partial decryption* pdec_i*.*

$\{\mathsf{amsg}, \perp\} \leftarrow \mathsf{TRAS.Combine}(\mathsf{pdec}_{i_1}, \ldots, \mathsf{pdec}_{i_t})$: *A partial anamorphic decryption combination algorithm that on input a set of partial decryption* $(\mathsf{pdec}_{i_1}, \ldots, \mathsf{pdec}_{i_t})$*, outputs either the anamorphic message* amsg *or* $\perp$*.*

We say that a TRAS *scheme is* correct *if, for all public parameters* 1^λ*, all message pairs* $(\mathsf{msg}, \mathsf{amsg})$*, the following event holds:*

$$\Pr\left[\begin{array}{c|l} |\mathcal{J}| \geq t & \sigma^{\mathsf{a}} \leftarrow \mathsf{TRAS.Sign}(\mathsf{sk}^S, \mathsf{adk}^S, \mathsf{msg}, \mathsf{amsg}) \\ \mathsf{amsg} = \mathsf{amsg}' & \{\mathsf{pdec}_i \leftarrow \mathsf{TRAS.Dec}(\mathsf{sk}^S, \mathsf{adk}_i, \sigma^{\mathsf{a}})\}_{i \in \mathcal{J}} \\ & \mathsf{amsg}' \leftarrow \mathsf{TRAS.Combine}(\{\mathsf{pdec}_i\}_{i \in \mathcal{J}}) \end{array}\right] = 1,$$

where the probability is over $(\mathsf{pk}^S, \mathsf{sk}^S) \leftarrow \mathsf{KGen}(1^\lambda)$ *and* $(\mathsf{adk}^S, \mathsf{adk}_1, \ldots, \mathsf{adk}_n) \leftarrow \mathsf{TRAS.KGen}(1^\lambda, t)$*, and the random coins of* $\mathsf{TRAS.Sign}$*.*

[5] The $\mathsf{TRAS.KGen}$ algorithm is performed in advance in a distributed manner across the anamorphic set, similar to a distributed key generation protocol [23]. This eliminates the need for a trusted setup, which is impractical in a totalitarian setting.

Definition 7 (Anamorphic). *We say that a* TRAS *scheme is anamorphic if for all parameter 1^λ, for any polynomial time algorithm $\mathcal{A}$, the following holds:*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{TR\text{-}Anam}}(1^\lambda) \leq \mathsf{negl}(\lambda),$$

where $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{TR\text{-}Anam}}(1^\lambda) = \left|\Pr\left[\mathsf{ExpTR\text{-}Anam}^{\mathcal{A}}_{\mathsf{AS}}(1^\lambda) = 1\right] - \frac{1}{2}\right|$ *with the experiment* ExpTR-Anam *defined in Figure 2. The experiment is defined similarly to the experiment* ExpAnam *of an anamorphic signature scheme defined in [28], which is also presented in Definition 23. We allow the adversary $\mathcal{A}$ to pick the threshold parameter (n, t). We also allow the adversary $\mathcal{A}$ to select the sender's signature key pair $(\mathsf{sk}^S, \mathsf{pk}^S)$ while the anamorphic network generates the anamorphic keys. The adversary then interacts with either an oracle producing a standard signature or one producing an anamorphic signature. Eventually, its goal is to determine which oracle it interacted with. The differences compared to* ExpAnam*, defined in Definition 23, is that we allow the adversary to pick the threshold parameter (n, t) and the signature key pair $(\mathsf{sk}^S, \mathsf{pk}^S)$, replace the anamorphic key generation algorithm* AS.KGen *with the key generation algorithm* TRAS.KGen *and the anamorphic signing algorithm* AS.Sign *in the oracle* aSign_1 *with the anamorphic signing algorithm* TRAS.Sign *of a* TRAS *scheme.*

$\mathsf{ExpTR\text{-}Anam}^{\mathcal{A}}_{\mathsf{AS}}(1^\lambda)$	$\mathsf{aSign}_0(\mathsf{msg}, \mathsf{amsg})$
$(n, t, \mathsf{sk}^S, \mathsf{pk}^S) \leftarrow \mathcal{A}(1^\lambda)$	$\sigma \leftarrow \mathsf{S.Sign}(\mathsf{sk}^S, \mathsf{msg}, \mathsf{st})$
$(\mathsf{adk}^S, \mathsf{adk}_1, \ldots, \mathsf{adk}_n) \leftarrow \mathsf{TRAS.KGen}(1^\lambda, n, t)$	**return** σ
$b \leftarrow\!\!\$\ \{0, 1\}$	$\mathsf{aSign}_1(\mathsf{msg}, \mathsf{amsg})$
$b' \leftarrow \mathcal{A}^{\mathsf{aSign}_b}(\mathsf{sk}, \mathsf{pk})$	$\sigma \leftarrow \mathsf{TRAS.Sign}(\mathsf{sk}^S, \mathsf{adk}^S, \mathsf{msg}, \mathsf{amsg})$
return $b = b'$	**return** σ

Fig. 2. Anamorphic experiment $\mathsf{ExpTR\text{-}Anam}^{\mathcal{A}}_{\mathsf{AS}}$.

Definition 8 (Semantic security). *For a threshold-recipient anamorphic signature scheme* TRAS *and a stateful, polynomial time adversary $\mathcal{A}$, we define the anamorphic IND-CPA experiment* ExpTR-CPA *as follows:*

- **Setup phase.** *The adversary $\mathcal{A}$ first specifies the number n of overall recipients and the threshold t. This is followed by challenger sampling the anamorphic keys using the* TRAS.KGen *algorithm of a* TRAS *scheme. We also let the adversary $\mathcal{A}$ choose the sender's signature key pair $(\mathsf{sk}^S, \mathsf{pk}^S)$.*
- **Query phase.** *The adversary $\mathcal{A}$ then interacts with the challenger using two types of queries: key corruption queries and anamorphic signing queries.*
  - $\mathsf{Corr}(i)$: *This oracle reveals to $\mathcal{A}$ the secret decryption key of recipient i who was not queried before and blocks further oracle access when the number of corrupted recipients exceeds the predetermined threshold t.*
  - $\mathsf{aSign}(\mathsf{msg}, \mathsf{amsg})$: *This oracle executes an anamorphic signing algorithm* TRAS.Sign *on a given pair of regular-anamorphic message* $(\mathsf{msg}, \mathsf{amsg})$.

– **Challenge phase.** *The adversary $\mathcal{A}$, given the sender double key adk^S and the list of corrupted recipient $\mathcal{J}$, determines a message msg to be signed by the challenger and two anamorphic messages $(\mathsf{msg}_0, \mathsf{msg}_1)$. The challenger in the experiment chooses a random bit b and executes the anamorphic signing algorithm on the message msg and the corresponding anamorphic message amsg_b. Finally, the challenger sends the anamorphic signature σ_b to $\mathcal{A}$.*
– **Output phase.** *The adversary $\mathcal{A}$ outputs a guess b' on the chosen bit b.*

We say that an TRAS *scheme has anamorphic semantic security if, for all parameter 1^λ, for any polynomial-time algorithm $\mathcal{A}$, the following holds:*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{TR\text{-}CPA}}(1^\lambda) \leq \mathsf{negl}(\lambda),$$

where $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{TR\text{-}CPA}}(1^\lambda) = \left|\Pr\left[\mathsf{ExpTR\text{-}CPA}^{\mathcal{A}}_{\mathsf{TRAS}}(1^\lambda) = 1\right] - \frac{1}{2}\right|$ *with the experiment* ExpTR-CPA *defined in Figure 3.*

$\mathsf{ExpTR\text{-}CPA}^{\mathcal{A}}_{\mathsf{TRAS}}(1^\lambda)$	$\mathsf{Corr}(i)$
$\mathcal{J} \leftarrow \emptyset;\ (n, t, \mathsf{sk}^S, \mathsf{pk}^S) \leftarrow \mathcal{A}(1^\lambda)$	**if** $i \notin [n]$ **or** $i \in \mathcal{J}$ **or** $\lvert\mathcal{J}\rvert >= t-1$
$(\mathsf{adk}^S, \mathsf{adk}_1, \ldots, \mathsf{adk}_n) \leftarrow \mathsf{TRAS.KGen}(1^\lambda, n, t)$	**then return** $\perp$
$(\mathsf{msg}, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}^{\mathsf{Corr},\mathsf{aSign}}(\mathsf{adk}^S, \mathcal{J})$	$\mathcal{J} \leftarrow \mathcal{J} \cup \{i\}$
$b \leftarrow\!\!\$\ \{0,1\}$	**return** adk_i
$\sigma_b \leftarrow \mathsf{TRAS.Sign}(\mathsf{sk}^S, \mathsf{adk}^S, \mathsf{msg}, \mathsf{amsg}_b)$	$\mathsf{aSign}(\mathsf{msg}, \mathsf{amsg})$
$b' \leftarrow \mathcal{A}^{\mathsf{Corr},\mathsf{aSign}}(\sigma_b)$	$\sigma \leftarrow \mathsf{TRAS.Sign}(\mathsf{sk}^S, \mathsf{adk}^S, \mathsf{msg}, \mathsf{amsg})$
return $b = b'$	**return** σ

Fig. 3. Threshold-recipient anamorphic IND-CPA security experiment $\mathsf{ExpTR\text{-}CPA}^{\mathcal{A}}_{\mathsf{TRAS}}$.

3.2 Generic construction

We propose a generic construction for TRAS. Our idea is straightforward and relies on a threshold encryption scheme TE and a signature scheme S that allows for random coin extraction from a signature while holding the signing key. The idea is to set the double key of the sender as the threshold encryption public key $\mathsf{adk}^S = \mathsf{pk}^{\mathsf{TE}}$ and give the threshold secret key to users as their double keys $\mathsf{adk}_i = \mathsf{sk}_i^{\mathsf{TE}}$. To encrypt an anamorphic message, the sender first encrypts amsg using the TE scheme for public key adk^S and then uses this ciphertext as the random coins. Here, we assume that the TE ciphertext is pseudorandom, and the ciphertext space of the TE scheme coincides with the random coin space of the signature, i.e., its representation "looks" like the random coin used in the signing process of the signature scheme S. The recipients then use the extraction algorithm and the sender's signing key to get the TE ciphertext. They then apply the TE partial decryption algorithm to get the partial decryptions that can be combined using the TE combination algorithm. We give more details in Figure 4.

Definition 9 (Generic TRAS construction). *A generic threshold-recipient anamorphic signature scheme uses a threshold encryption scheme* $\mathsf{TE} = (\mathsf{TE.KGen},$

$\mathsf{TE.Enc}, \mathsf{TE.Combine}, \mathsf{TE.Dec}$) *that has pseudorandom ciphertext and an anamorphic signature* $\mathsf{AS} = (\mathsf{AS.KGen}, \mathsf{AS.Sign}, \mathsf{AS.Dec})$ *associates with a random coin extractable digital signature scheme* $\mathsf{S} = (\mathsf{S.KGen}, \mathsf{S.Sign}, \mathsf{S.Vf})$ *and a pair of anamorphic sender's signature signing and verification keys* $(\mathsf{pk}^S, \mathsf{sk}^S) \leftarrow \mathsf{S.KGen}(1^\lambda)$ *as the main building blocks. Our generic* TRAS *scheme is then defined as follows. We also give a high-level overview of the particular algorithms in Figure 4.*

$(\mathsf{adk}^S, \mathsf{adk}_1, \ldots, \mathsf{adk}_n) \leftarrow \mathsf{TRAS.KGen}(1^\lambda, n, t)$: *The anamorphic key generation algorithm runs a threshold encryption key generation algorithm* $(\mathsf{pk}^{\mathsf{TE}}, \mathsf{sk}_1^{\mathsf{TE}}, \ldots, \mathsf{sk}_n^{\mathsf{TE}}) \leftarrow \mathsf{TE.KGen}(1^\lambda, n, t)$. *The algorithm then assigns* $(\mathsf{adk}^S, \mathsf{adk}_1, \ldots, \mathsf{adk}_n) := (\mathsf{pk}^{\mathsf{TE}}, \mathsf{sk}_1^{\mathsf{TE}}, \ldots, \mathsf{sk}_n^{\mathsf{TE}})$ *and outputs the key tuple.*

$\sigma^{\mathsf{a}} \leftarrow \mathsf{TRAS.Sign}(\mathsf{sk}^S, \mathsf{adk}^S, \mathsf{msg}, \mathsf{amsg})$: *The anamorphic signing algorithm first embeds the anamorphic message* amsg *into the random coin* st *by computing* $\mathsf{st} \leftarrow \mathsf{TE.Enc}(\mathsf{adk}^S, \mathsf{amsg})$. *It then executes the signing algorithm* $\sigma^{\mathsf{a}} \leftarrow \mathsf{S.Sign}(\mathsf{sk}^S, \mathsf{msg}; \mathsf{st})$ *and outputs the anamorphic signature* σ^{a}. *Similar to prior work, we assume the output space of* TE.Enc *is identical to the randomness space of* S.Sign. *More precisely, we assume that the* TE *for security parameter* 1^λ *encrypts* $n(\lambda)$*-bit plaintexts into* $\ell(\lambda)$*-bit ciphertexts, with* $\ell(\lambda)$ *be the bit-size of the random coin space in the signature scheme. This can be enforced with appropriate encoding.*

$\mathsf{pdec}_i \leftarrow \mathsf{TRAS.Dec}(\mathsf{sk}^S, \mathsf{adk}_i, \sigma^{\mathsf{a}})$: *The partial anamorphic decryption algorithm first extracts the random coin* st *from the anamorphic signature* σ^{a} *by computing* $\mathsf{st} \leftarrow \mathsf{S.RanExt}(\mathsf{sk}^S, \mathsf{msg}, \sigma^{\mathsf{a}})$. *The algorithm then computes* $\mathsf{pdec} \leftarrow \mathsf{TE.Dec}(\mathsf{adk}_i, \mathsf{st})$ *to retrieves the partial decryption* pdec_i.

$\{\mathsf{amsg}, \perp\} \leftarrow \mathsf{TRAS.Combine}(\{\mathsf{pdec}_i\}_{i \in \mathcal{J}})$: *The partial decryption combination algorithm checks if the threshold* t *is met. It then runs* $\mathsf{amsg} \leftarrow \mathsf{TE.Combine}(\{\mathsf{pdec}_i\}_{i \in \mathcal{J}})$ *to reconstruct the anamorphic message* amsg.

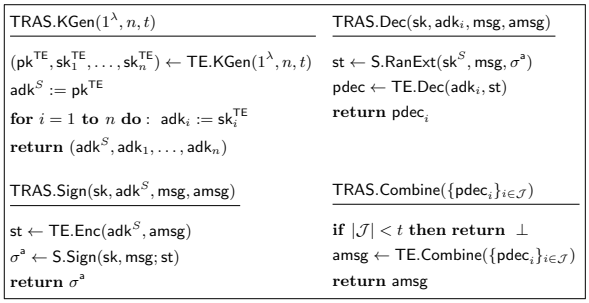

Fig. 4. A generic construction for threshold-receiver anamorphic signature scheme.

Theorem 2. *The construction in Figure 4 is correct and anamorphic.*

Theorem 3. *For any p.p.t.* $\mathcal{A}$*, it holds:* $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{TR\text{-}CPA}}(1^\lambda) \leq \mathbf{Adv}^{\mathcal{A}_1}_{\mathsf{TE\text{-}CPA}}(1^\lambda)$.

We deferred the proof of Theorem 2 and Theorem 3 to Appendix C.

4 Anamorphic Signatures under Strong Dictatorship

We consider a strong model for anamorphic signature. We assume that the dictator can control not only the cryptographic keys of the citizen but also limits the citizen's power only to use basic cryptography, i.e., a signature scheme and a Diffie-Hellman(DH) key exchange. One might think that this is a very strong setup; however, a dictator can issue electronic identity cards (eIDs) with pre-generated signing and DH keys. The dictator can then regulate that only signatures and key exchanges using the eID are allowed. It is important to note that the majority of government-regulated eID systems support DH or ECDH keys as part of the Password Authenticated Connection Establishment (PACE) protocol within their public key infrastructure. Furthermore, support for DH or ECDH keys is mandated by the International Civil Aviation Organization (ICAO) standard for ePassports [25], which all 193 member states are required to follow [26]. Therefore, it is reasonable to assume that an eID issued by the dictator supports DH or ECDH keys. We show that even in this model, a (sufficiently large) subgroup of citizens can exchange anamorphic messages, and the only "secret" information the dictator does not know is the group member's identities. The exciting part is that security will hold without relying on specified classical computational assumptions, providing a post-quantum secure solution.

To formally define this strong model, we reuse a significant part of the syntax from the previous section, since we also assume here that we have a recipient-threshold scheme. The main difference is that in all algorithms, we eliminate the double keys (since those do not exist here), and the sender's signing key is also not given to the recipient (e.g., it is hardware-protected on the eID). The only secret information used in the signing algorithm is a public key for the anamorphic subgroup of citizens, which we can compute using the algorithm $\mathsf{adk}^{\mathcal{J}} \leftarrow \mathsf{eTRAS.KCombine}(\{\mathsf{pk}_i\}_{i\in\mathcal{J}})$. Note that we assume citizens are allowed to exchange their public keys. Another difference is in the IND-CPA experiment for this new notion. We enable the adversary to specify the number of all citizens n, the size of the anamorphic subgroup ℓ, the number $Q_{\mathsf{cr}} \leq \ell$ of anamorphic subgroup members it is allowed to compromise, and the number $Q_{\mathsf{ch}} \leq n$ of citizens it can investigate. Since the only "secret" is the anamorphic group $\mathcal{J}$, the corruption query reveals a "global identifier" for one of the ℓ members, i.e., it gives a pointer for the adversary to the secret key of that member. On the other hand, Q_{ch} that relates to a checking oracle allows the adversary to check if a citizen identified via an index from 1 to n is part of the anamorphic subgroup.

Since we allow the adversary to pick the parameters $(n, \ell, Q_{\mathsf{cr}}, Q_{\mathsf{ch}})$, the experiment checks if the adversary can trivially break CPA security. Note that since we assume that all keys are known to the adversary, we need to assume that the secret $\mathcal{J}$ has enough entropy. In other words, from the perspective of the adversary, there need to be enough potential subgroup for the security to hold. In the experiment, the adversary is also is allowed to pick message msg and two challenge anamorphic messages amsg_0 and amsg_1 that the challenger of the IND-CPA experiment uses to generate the challenge signature $\sigma_b \leftarrow$

$\mathsf{eTRAS.Sign}(\mathsf{sk}^S, \mathsf{adk}^{\mathcal{J}}, \mathsf{msg}, \mathsf{amsg}_b)$. The rest of the experiment is similar to a standard IND-CPA, i.e., the adversary must guess bit b.

Another significant difference is that because no secret can be shared between group members, the only way to exchange information between the sender and group members is to use *rejection sampling*, i.e., check that the signature encodes the correct amsg (e.g., the last bits of signature match the amsg) by first creating and then running $\mathsf{Check}(\sigma^{\mathsf{a}}, \mathsf{amsg}) = 1$ and repeating the signing process otherwise. To make this work, constructions of such a strong primitive need to ensure that amsg can be viewed as pseudorandom bits; otherwise, an adversary can easily distinguish the anamorphic signatures. This process only allows encoding short messages amsg since to encode k bits we have to reject around 2^k signatures. We, therefore, extend the signing algorithm "sign" many messages $\{\mathsf{msg}\}_{i\in[\kappa]}$ at once, i.e., signers use $\{\sigma_i\}_{i\in[\kappa]} \leftarrow \mathsf{eTRAS.Sign}(\mathsf{sk}^S, \mathsf{adk}^{\mathcal{J}}, \{\mathsf{msg},\}_{i\in[\kappa]}, \mathsf{amsg})$ to sign. The parameter κ will be scheme specific.

Interestingly, since the secret we use is the $\mathcal{J}$, if all group members know it, then a very efficient scheme exists without needing multiple recipients and a threshold. Recipients can hash the $\mathcal{J}$ together with some random bits and then let the sender transfer the anamorphic message xor-ed with the output of the random oracle. The reason we define this primitive with multiple receives is because, this way, the sender can be the only party knowing the entire group $\mathcal{J}$. Members can independently generate partial decryptions and later exchange them by meeting in person or using other means of propagation. We provide more details on the syntax and the security definitions below.

4.1 Syntax and Security

Below we introduce the syntax of an extended threshold-recipient anamorphic signature scheme that differs from a TRAS scheme associated with a digital signature scheme $\mathsf{S} = (\mathsf{S.KGen}, \mathsf{S.Sign}, \mathsf{S.Vf})$ and a pair of sender's signing and verification keys $(\mathsf{pk}^S, \mathsf{sk}^S) \leftarrow \mathsf{S.KGen}(1^\lambda)$ in Definition 6.

$\mathcal{J} \leftarrow \mathsf{eTRAS.Select}(1^\lambda, n, \ell)$: The quorum selecting algorithm on input security parameter 1^λ the number of participants n and subgroup size ℓ, first verifies that $n > \lambda$ to ensure the security requirement is met and then outputs the quorum $\mathcal{J} := \{j_k\}_{k=1}^{\ell}$ of size ℓ that specifies members of the group.

$\mathsf{adk}^{\mathcal{J}} \leftarrow \mathsf{eTRAS.KCombine}(\{\mathsf{pk}_i\}_{i\in\mathcal{J}})$: The quorum public key combine algorithm that on input the public keys of members in quorum $\mathcal{J}$, outputs the combined sender double key $\mathsf{adk}^{\mathcal{J}}$ that correspond to the quorum $\mathcal{J}$.

$\mathsf{pdec}_i \leftarrow \mathsf{eTRAS.Dec}(\mathsf{sk}_i, \mathsf{pk}^S, \sigma^{\mathsf{a}})$: A partial anamorphic decryption algorithm that on input a secret key sk_i, the sender's public key pk^S, and an anamorphic signature σ^{a}, outputs a partial decryption pdec_i.

We say that an eTRAS scheme is *correct* if, for all public parameters 1^λ, all message pairs $(\mathsf{msg}, \mathsf{amsg})$, the following event holds:

$$\Pr\left[\begin{array}{c|l} |\mathcal{J}| = \ell & \sigma^{\mathsf{a}} \leftarrow \mathsf{eTRAS.Sign}(\mathsf{sk}^S, \mathsf{adk}^{\mathcal{J}}, \mathsf{msg}, \mathsf{amsg}) \\ \mathsf{amsg} = \mathsf{amsg}' & \{\mathsf{pdec}_i \leftarrow \mathsf{eTRAS.Dec}(\mathsf{sk}_i, \mathsf{pk}^S, \sigma^{\mathsf{a}})\}_{i\in\mathcal{J}} \\ & \mathsf{amsg}' \leftarrow \mathsf{eTRAS.Combine}(\{\mathsf{pdec}_i\}_{i\in\mathcal{J}}) \end{array}\right] = 1,$$

where the probability is over $(\mathsf{pk}^S, \mathsf{sk}^S) \leftarrow \mathsf{KGen}(1^\lambda)$, and $\{(\mathsf{sk}_i, \mathsf{pk}_i)\}_{i=1}^n \leftarrow \mathsf{KGen}(1^\lambda)$, and $\mathcal{J} \leftarrow \mathsf{eTRAS.Select}(1^\lambda, n, \ell)$, and the random coins of $\mathsf{eTRAS.Sign}$.

Definition 10 (Anamorphic). *We say that an* eTRAS *scheme is anamorphic if for all parameter* 1^λ*, for any polynomial time algorithm* $\mathcal{A}$*, the following holds:*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{eTR\text{-}Anam}}(1^\lambda) \leq \mathsf{negl}(\lambda),$$

where $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{eTR\text{-}Anam}}(1^\lambda) = \left| \Pr\left[\mathsf{ExpeTR\text{-}Anam}^{\mathcal{A}}_{\mathsf{AS}}(1^\lambda) = 1\right] - \frac{1}{2} \right|$ *with the experiment* ExpeTR-Anam *defined in Figure 5. The adversary first selects the parameters* (n, ℓ) *and generates all keys (sender and recipient) for the citizen. We abort the experiment when the guessing probability* $\binom{n}{\ell}$ *is trivial. The anamorphic sender then chooses an anamorphic quorum* $\mathcal{J}$ *of size* ℓ *and computes the corresponding double key for the quorum by executing* $\mathsf{adk}^{\mathcal{J}} \leftarrow \mathsf{eTRAS.KCombine}(\{\mathsf{pk}_i\}_{i\in\mathcal{J}})$*. The rest of the experiment is defined similarly as in* ExpTR-Anam*.*

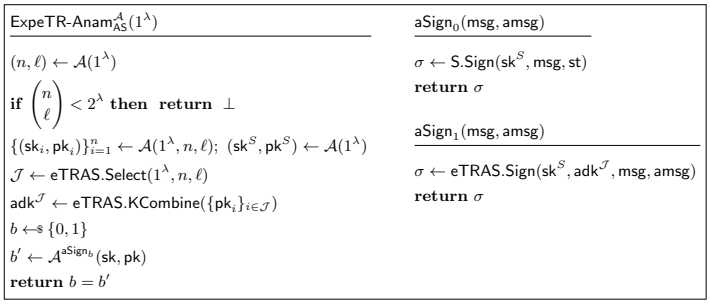

Fig. 5. Anamorphic experiment $\mathsf{ExpeTR\text{-}Anam}^{\mathcal{A}}_{\mathsf{AS}}$.

Definition 11 (Semantic security). *For an extended threshold-recipient anamorphic signature scheme* eTRAS *and a stateful, polynomial time adversary* $\mathcal{A}$*, we define the anamorphic IND-CPA experiment* ExpeTR-CPA *as follows:*

- **Setup phase.** *The adversary* $\mathcal{A}$ *first specifies the number* n *of overall participants and the size* ℓ *of the anamorphic group as well as the number of queries* $Q_{\mathsf{cr}}, Q_{\mathsf{ch}}$ *to the* Corr, Check *oracles respectively. The challenger checks whether the correct anamorphic group guessing probability for the adversary is negligible. This is followed by the challenger forming the anamorphic group* $\mathcal{J} := \{j_k\}_{k=1}^{\ell}$ *using the* eTRAS.Select *algorithm and computing the group anamorphic key* $\mathsf{adk}^{\mathcal{J}} \leftarrow \mathsf{eTRAS.KCombine}(\{\mathsf{pk}_i\}_{i\in\mathcal{J}})$*. We also let the adversary* $\mathcal{A}$ *choose the signature key pair of every participant in the network.*
- **Query phase.** *The adversary* $\mathcal{A}$ *then interacts with the challenger using two types of queries: key corruption queries and anamorphic signing queries.*
  - $\mathsf{Corr}(i)$: *This oracle reveals to* $\mathcal{A}$ *the identity of a member in the determined anamorphic group and blocks further oracle access when the number of corrupted members exceeds the predetermined threshold* Q_{cr}*.*

- $\mathsf{Check}(i)$: *This corruption oracle reveals to the adversary* $\mathcal{A}$ *whether the user* i *belongs to the anamorphic group or not. The oracle blocks access when the number of queries reaches* Q_{ch}.
- $\mathsf{aSign}(\mathsf{msg}, \mathsf{amsg})$: *This oracle is defined as in the* ExpTR-CPA.

– **Challenge phase.** *The adversary* $\mathcal{A}$*, given the list of corrupted group member* $\mathcal{J}_{\mathsf{cr}}$*, determines a message* msg *to be signed by the challenger and two anamorphic messages* $(\mathsf{msg}_0, \mathsf{msg}_1)$*. The challenger in the experiment chooses a random bit* b *and executes the anamorphic signing algorithm on the message* msg *and the corresponding anamorphic message* amsg_b *using the anamorphic key* $\mathsf{adk}^{\mathcal{J}}$*. Finally, the challenger sends the anamorphic signature* σ_b *to* $\mathcal{A}$.

– **Output phase.** *The adversary* $\mathcal{A}$ *outputs a guess* b' *on the chosen bit* b.

We say that an eTRAS *scheme has anamorphic semantic security if, for all parameter* 1^λ*, for any polynomial-time algorithm* $\mathcal{A}$*, with* Q_{cr} *queries to the oracle* Corr *and* Q_{ch} *queries to the oracle* Check*, the following holds:* $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{eTR\text{-}CPA}}(1^\lambda) \leq \mathsf{negl}(\lambda)$*, where* $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{eTR\text{-}CPA}}(1^\lambda) = \left| \Pr\left[\mathsf{ExpeTR\text{-}CPA}^{\mathcal{A}}_{\mathsf{TRAS}}(1^\lambda) = 1\right] - \frac{1}{2} \right|$ *with the experiment* ExpeTR-CPA *defined in Figure 6.*

$\mathsf{ExpeTR\text{-}CPA}^{\mathcal{A}}_{\mathsf{eTRAS}}(1^\lambda)$

$\mathcal{J}_{\mathsf{cr}} \leftarrow \emptyset;\ \mathsf{cnt} \leftarrow 0;\ (n, \ell, Q_{\mathsf{cr}}, Q_{\mathsf{ch}}) \leftarrow \mathcal{A}(1^\lambda)$

if $\binom{n - Q_{\mathsf{cr}} - Q_{\mathsf{ch}}}{\ell - Q_{\mathsf{cr}} - \lambda} < 2^\lambda$ **then**

  return $\perp$

$\{(\mathsf{sk}_i, \mathsf{pk}_i)\}_{i=1}^n \leftarrow \mathcal{A}(1^\lambda, n, \ell)$

$(\mathsf{sk}^S, \mathsf{pk}^S) \leftarrow \mathcal{A}(1^\lambda)$

$\mathcal{J} \leftarrow \mathsf{eTRAS.Select}(1^\lambda, n, \ell)$

$\mathsf{adk}^{\mathcal{J}} \leftarrow \mathsf{eTRAS.KCombine}(\{\mathsf{pk}_i\}_{i \in \mathcal{J}})$

$(\{\mathsf{msg},\}_{i \in [\kappa]}, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}^{\mathsf{Corr},\mathsf{Check},\mathsf{aSign}}(\mathcal{J}_{\mathsf{cr}})$

$b \leftarrow\$ \{0, 1\}$

$\{\sigma_{b,i}\}_{i \in [\kappa]} \leftarrow \mathsf{eTRAS.Sign}(\mathsf{sk}^S, \mathsf{adk}^{\mathcal{J}}, \{\mathsf{msg},\}_{i \in [\kappa]}, \mathsf{amsg}_b)$

$b' \leftarrow \mathcal{A}^{\mathsf{Corr},\mathsf{Check},\mathsf{aSign}}(\{\sigma_{b,i}\}_{i \in [\kappa]})$

return $b = b'$

$\mathsf{Corr}(i)$

Parse: $\{j_k\}_{k=1}^{\ell} := \mathcal{J}$

if $i \notin [\ell]$ **or** $|\mathcal{J}_{\mathsf{cr}}| >= Q_{\mathsf{cr}} - 1$ **then return** $\perp$

$\mathcal{J}_{\mathsf{cr}} \leftarrow \mathcal{J}_{\mathsf{cr}} \cup \{j_i\}$

return j_i

$\mathsf{Check}(i)$

if $\mathsf{cnt} > Q_{\mathsf{ch}}$ **then return** $\perp$

$\mathsf{cnt} \leftarrow \mathsf{cnt} + 1$

if $i \in \mathcal{J}$ **then return** 1

else return 0

$\mathsf{aSign}(\mathsf{msg}, \mathsf{amsg})$

$\sigma \leftarrow \mathsf{eTRAS.Sign}(\mathsf{sk}^S, \mathsf{adk}^{\mathcal{J}}, \mathsf{msg}, \mathsf{amsg})$

return σ

Fig. 6. IND-CPA security experiment $\mathsf{ExpeTR\text{-}CPA}^{\mathcal{A}}_{\mathsf{eTRAS}}$ for eTRAS.

4.2 Strongly Secure Recipient-Threshold Anamorphic Signatures

We show how to instantiate this strong model with simple cryptographic primitives. The construction relies on a non-deterministic signature scheme $\mathsf{S} = (\mathsf{S.KGen}, \mathsf{S.Sign}, \mathsf{S.Vf})$ that allows for the mentioned rejection sampling technique.

In the key generation algorithm eID.KGen of the eID, we assume that user is given a keypair $(\mathsf{sk}^{\mathsf{S}}, \mathsf{pk}^{\mathsf{S}})$ for the scheme S and receives a DH keypair $(\mathsf{sk}_{\mathsf{DH}}, \mathsf{pk}_{\mathsf{DH}} = g^{\mathsf{sk}_{\mathsf{DH}}})$. To align with the model, we permit $(\mathsf{sk}^{\mathsf{S}}, \mathsf{pk}^{\mathsf{S}})$ to be generated by $\mathcal{A}$. However, given that DH key generation is a supported functionality within the eID

and is required for correct operation, we allow these DH keys to be generated honestly by the sender rather than by the dictator.

To send an anamorphic message to the anamorphic subgroup $\mathcal{J}$, the signer first computes the public key $\mathsf{adk}^{\mathcal{J}} = \prod_{i \in \mathcal{J}} \mathsf{pk}_{i,\mathsf{DH}}$. Given this public key, it computes random bits $r \leftarrow\!\!\$\ \{0,1\}^{\ell_q + \lambda}$, where ℓ_q is the bit size of the order q of the DH group. The signer then computes $r_q = r \mod q$ and the share key $K = (\mathsf{adk}^{\mathcal{J}})^{r_q \cdot \mathsf{sk}_{S,\mathsf{DH}}}$. Note that we use the DH keys in a key encapsulation mechanism for K, where the recipients will share their partial decapsulation values. The key K is the hashed using a random oracle and used as a one-time pad for the anamorphic message $\mathsf{ct} = \mathsf{H}(K) \oplus \mathsf{amsg}$. The message encoded in the signature S is $\mathsf{amsg}' = (r, \mathsf{ct})$.

Each recipient decodes amsg' from the sender's message and computes its share as $\mathsf{pdec}_i = ((\mathsf{pk}_{S,\mathsf{DH}})^{r_q \cdot \mathsf{sk}_{i,\mathsf{DH}}}, \mathsf{ct})$. The recipients also include the ciphertext ct in the partial decryptions. Note that this is only for the consistency of the algorithms. In an actual implementation, the ciphertext ct would be retained by each recipient and later used together with all other partial decryptions. The anamorphic group member can then use the same rejection sampling technique to broadcast pdec_i to others. Once all partial values are known, the members can compute the key $K = \prod_{i \in \mathcal{J}} \mathsf{share_i}$, where $\mathsf{pdec}_i = (\mathsf{share}_i, \mathsf{ct})$ and decrypt the anamorphic message of the sender by computing $\mathsf{amsg} = \mathsf{ct} \oplus \mathsf{H}(K)$.

Before we go into the details in Figure 7 we define three algorithms Encode, Decode, Check. We first define the Check function that takes as input a signature σ_i^{a} and short anamorphic message amsg_i and checks if amsg_i corresponds to some bit on predetermined positions. By re-signing the actual message msg_i, we receive a fresh σ_i^{a}, and the Check function can be used to implement the rejection sampling idea [36]. Note that the number of bits checked in this function influences the number of re-signing we have to do, i.e., the size of $\ell_\sigma^{\mathsf{a}} = \sigma_i^{\mathsf{a}}$ needs to be small so that $2^{\ell_\sigma^{\mathsf{a}}}$ is feasible computation for the sender. Unfortunately, the anamorphic message $\mathsf{amsg}' = (r, \mathsf{ct})$ in our scheme is long and cannot directly be encoded into a single signature. We, therefore, will use the function $\mathsf{Encode}(\mathsf{amsg}')$ to decompose amsg' into smaller chunks $\mathsf{amsg}_1, \ldots, \mathsf{amsg}_\kappa$. The decoding algorithm Decode will take as input all amsg_i chunks and output the full anamorphic message amsg'. Encode and Decode are functions that truncate and combine bits. This is possible since, contrary to the original idea of rejection sampling [36], the anamorphic message in our scheme consists of a bit string indistinguishable from a uniformly random string in the adversary's view. The original rejection sampling idea used a keyed PRF to "encode" a bit string with non-uniform distribution into an indistinguishable from uniform one. However, in our scheme, citizens do not share additional keys (i.e., double keys); we can only use this technique with amsg' distributed indistinguishable from uniform bit strings.

Theorem 4. *The construction in Figure 7 is correct and anamorphic.*

Theorem 5. *The construction in Figure 7 is IND-CPA secure according to Definition 11. More formally, for any polynomial-time adversary $\mathcal{A}$ making at most*

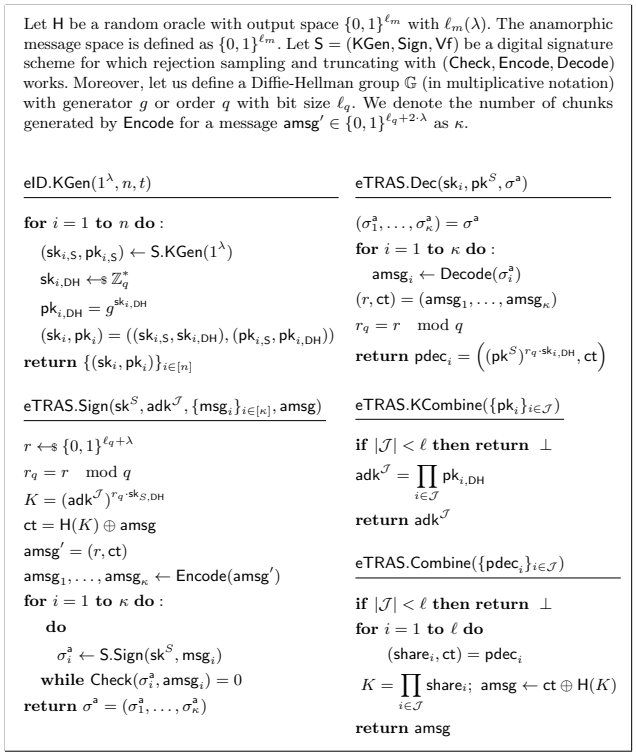

Fig. 7. Strongly Secure Threshold-Receiver Anamorphic Signature Scheme.

q_h queries to oracle H*, the following holds:*

$$\mathsf{ExpeTR\text{-}CPA}^{\mathcal{A}}_{\mathsf{eTRAS}}(1^\lambda) \leq q_h \cdot (1/\mathsf{bino} + 1/q) + q_h^2/2^{\ell_m},$$

where $\mathsf{bino} = \binom{n-Q_{\mathsf{cr}}-Q_{\mathsf{ch}}}{\ell-Q_{\mathsf{cr}}-\lambda}$*,* $(n, \ell, Q_{\mathsf{cr}}, Q_{\mathsf{ch}})$ *are the parameters output by* $\mathcal{A}$ *in the* $\mathsf{ExpeTR\text{-}CPA}$ *experiment and assuming* $Q_{\mathsf{ch}} \cdot (\ell - Q_{\mathsf{cr}})/(n - Q_{\mathsf{cr}} - Q_{\mathsf{ch}}) < 1$ *and* q *is the order of the group* $\mathbb{G}$ *generated by* g*.*

We deferred all omitted proofs to Appendix E.

Remark 1. Let us assume that the number of citizens is $n = 2^{20}$ (≈ 1 mln) and the dictator can check around $1/4$ of the whole population n, i.e., $Q_{\mathsf{ch}} = n/4$, and can corrupt the same amount of anamorphic members $Q_{\mathsf{cr}} = \ell/4$. We then have $\mathsf{bino} = \binom{n-n/4-\ell/4}{\ell-\ell/4-\lambda} = \binom{3n/4-\ell/4}{3\ell/4-\lambda}$. For $\lambda = 128$ and $q_h = 2^\lambda$, 512-bit anamorphic message, the group must only be of size $\ell = 192$ to achieve 128-bit security.

5 Threshold-Sender Anamorphic Signatures

We investigate the concept of a threshold-sender anamorphic signature scheme, or in other words, a primitive that seeks to embed a secret anamorphic message within a threshold signature framework. The motivation for adding anamorphism

to a threshold signature setting is that anamorphic users can hide their identities, even if the signature is suspected of containing such a message. Since a group of signers produces a threshold signature, any user embedding the anamorphic message can hope to remain anonymous within that group. However, embedding a hidden message in a threshold signature introduces significant challenges. In addition to making the signature appear ordinary to outsiders, the anamorphic message must remain secure and correctly embedded, even in the presence of corrupted signers who could compromise the signing process.

A key challenge in formalizing the concept of a threshold-sender anamorphic signature (TSAS) lies in dealing with signer corruption. Threshold signatures are inherently designed to tolerate the corruption of signers up to a certain threshold, and this property should extend to TSAS. However, in TSAS, the corruption of the sender immediately compromises the anamorphic message, particularly in symmetric schemes where the same double key is used for both encryption and decryption. To address this, we propose a model where corrupted signers can still use the ordinary signing algorithm, while honest signers use the anamorphic signing algorithm to inject the anamorphic message. This model is designed to preserve the confidentiality of the anamorphic message even in the presence of corrupt signers. We next present the syntax of TSAS in Section 5.1, followed by a detailed discussion of the security requirements in Section 5.2.

In the end, we come up with a conclusion that constructing such a TSAS scheme is impossible within a broad class of efficient, standardizing-candidate threshold signature schemes: the two-round FROST (see Section 5.3), the three-round Sparkle (see Appendix D.3), a multi-signature MuSig2 (see Appendix D.2) and even the three-round lattice-based TRaccoon (see Appendix D.4).

5.1 Threshold-sender Anamorphic Signature Syntax

Let n be the number of users in the network and t be the threshold. The formal syntax and correctness definition of TSAS is given in Definitions 12 and 13.

Definition 12 (Threshold-sender Anamorphic Signature Scheme). *A two-round threshold-sender anamorphic signature scheme* $\mathsf{TSAS} = (\mathsf{TSAS.KGen}, \mathsf{TSAS.Sign}_1, \mathsf{TSAS.Sign}_2, \mathsf{TSAS.Combine}, \mathsf{TSAS.Dec})$ *associated with a threshold signature scheme* $\mathsf{TS} = (\mathsf{TS.KGen}, \mathsf{TS.Sign}, \mathsf{TS.Combine}, \mathsf{TS.Vf})$ *and a list of signing and verification keys* $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TS.KGen}(1^\lambda, n, t)$ *of the threshold signature scheme, consists of the following p.p.t. algorithms:*

- $(\mathsf{adk}^S, \mathsf{adk}) \leftarrow \mathsf{TSAS.KGen}(1^\lambda, n, t)$: *A key generation algorithm that on input security parameter* 1^λ*, a number of participants and a threshold* t*, outputs a double key* adk^S *for the sender and a double key* adk *for the recipient.*
- $(\mathsf{msg}_{i,1}, \mathsf{St}_{i,1}) \leftarrow \mathsf{TSAS.Sign}_1(\mathsf{sk}_i, \mathsf{msg}, \mathsf{amsg}, \mathcal{J}, \mathsf{adk}^S)$: *on input a secret key* sk_i*, a message* msg*, an anamorphic message* amsg*, a subset* $\mathcal{J}$ *of indices and a sender double key* adk^S*, returns a state* $\mathsf{St}_{i,1}$ *and a first message* $\mathsf{msg}_{i,1}$ *to be sent during round 1 of the protocol to all signers in* $\mathcal{J} \setminus \{i\}$*.*
- $s_i \leftarrow \mathsf{TSAS.Sign}_2(\mathsf{St}_{i,1}, \{\mathsf{msg}_{j,1}\}_{j \in \mathcal{J} \setminus \{i\}})$: *a deterministic algorithm on input a state* $\mathsf{St}_{i,1}$ *and incoming messages* $\{\mathsf{msg}_{j,1}\}_{j \in \mathcal{J} \setminus \{i\}}$*, outputs a share* s_i*. Since*

amsg *and* adk^S *were already stored in the state* $\mathsf{St}_{i,1}$ *during the execution of* $\mathsf{TSAS.Sign}_1$*, the algorithm is permitted to omit these values as explicit inputs and instead provide them implicitly through* $\mathsf{St}_{i,1}$*.*

$\{\sigma^{\mathsf{a}}, \perp\} \leftarrow \mathsf{TSAS.Combine}(\mathsf{pk}, s_{i_1}, \ldots, s_{i_t})$: *A share combination algorithm that on input the public key* pk *of the threshold signature scheme and a set of signature shares* $(s_{i_1}, \ldots, s_{i_t})$*, outputs either a signature* σ *or* $\perp$*. For the anamorphic setting,* $\mathsf{TSAS.Combine}$ *and* $\mathsf{TS.Combine}$ *should be identical and can be used interchangeably.*

$\mathsf{amsg} \leftarrow \mathsf{TSAS.Dec}(\mathsf{adk}, \sigma^{\mathsf{a}})$: *an anamorphic decryption algorithm on input a signing key* sk*, a recipient's double key* adk *and an anamorphic signature* σ^{a}*, outputs an anamorphic message* amsg*.*

Remark 2. Since both $\mathsf{TSAS.Sign}_{1,2}$ *are executed locally by the anamorphic sender, it is permitted to embed the anamorphic message in both algorithms. This syntax maximizes the sender's capacity for anamorphic embedding.*

Definition 13. *We say that a* TSAS *scheme is correct if for all public parameters* 1^λ*, all message pairs* $(\mathsf{msg}, \mathsf{amsg})$ *in the associated message space, all positive integers* n, t *such that* $t \leq n$*, the following event holds:*

$$\Pr\left[\begin{array}{c} |\mathcal{J}| \geq t \\ \mathsf{amsg} = \mathsf{amsg}' \end{array} \middle| \begin{array}{c} \mathsf{msg}_{i,1} \leftarrow \mathsf{TSAS.Sign}_1(\mathsf{sk}, \mathsf{msg}, \mathsf{amsg}, \mathcal{J}, \mathsf{adk}^S), \forall i \in \mathcal{J} \\ s_i \leftarrow \mathsf{TSAS.Sign}_2(\mathsf{St}_{i,1}, \{\mathsf{msg}_{j,1}\}_{j \in \mathcal{J} \setminus \{i\}}), \forall i \in \mathcal{J} \\ \sigma^{\mathsf{a}} \leftarrow \mathsf{TSAS.Combine}(\mathsf{pk}, \{s_j\}_{j \in \mathcal{J}}) \\ \mathsf{amsg} \leftarrow \mathsf{TSAS.Dec}(\mathsf{sk}, \mathsf{adk}, \sigma^{\mathsf{a}}) \end{array}\right] = 1,$$

where the probability is over $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow\!\!\$\ \mathsf{TS.KGen}(1^\lambda, n, t)$*,* $(\mathsf{adk}^S, \mathsf{adk}) \leftarrow \mathsf{TSAS.KGen}(1^\lambda, n, t)$ *and the random coins of* $\mathsf{TSAS.Sign}$*.*

5.2 TSAS Indistinguishability with Corruptions

Adversary Model. In our setting, we examine an adversarial model in which we establish an impossibility result. To strengthen this impossibility result to the greatest extent, we limit the power and the malicious behaviors of the adversary as much as possible. The motivation is that, even when the dictator acts solely as a passive signer (who signs any given message), the anamorphic sender still cannot embed the anamorphic message within the signature without being noticed.

The adversary is allowed to choose one secret key sk_k and specifies a signing quorum $\mathcal{J}$ at the beginning. We model the case in which it actively participates in the signing protocol. We highlight some insights for the adversary as follows.

Predefined Corruption . We stress that it must declare the index of the corrupted signer beforehand. Otherwise, it leads to a trivial attack where the adversary could corrupt a signer running an anamorphic signing algorithm, extract the double key, and distinguish it from those using the normal signing algorithm.

Semi-honest Adversary . We follow the semi-honest adversary model introduced in [24]. In this setting, we consider only a semi-honest adversary who follows the protocol (e.g., using the standard signing algorithm). This rules out trivial denial-of-service attacks on the standard threshold signature, i.e., preventing the protocol from outputting a valid signature.

Rushing Adversary . We follow the rushing adversary model introduced in [14,38]. In the same round, $\mathcal{A}$ waits for the other signers to send their respective responses before sending its own. By responding afterward, $\mathcal{A}$ is able to act adaptively based on the other signers' messages.

Quorum size is strictly controlled . In this setting, we enforce the size of the signing quorum to be exactly equal to the threshold t of the TS scheme. If the quorum's size is smaller than the threshold, the quorum trivially is incapable of signing. On the other hand, if the quorum's size is larger than the threshold, the honest signers can always form a sub-quorum that excludes the adversary while still issuing a valid signature. This setup ensures that the adversary must actively participate in the signing protocol to have an influence.

Security Model. We formalize two properties: malicious correctness and anamorphic indistinguishability in Definition 14 and Definition 15. To add more details, both experiments are parameterized by (n, t), and the threshold key generation is executed honestly. The sender then executes $\mathsf{TSAS.KGen}$ to retrieve the double keys and send the other double key to the receiver (only in the anamorphic experiment for the indistinguishability model). Also, similar to a requirement in most of the threshold signature schemes [30, 39], we only allow at most one query to signer i in the second round, enforcing by using the set $S_{i,1}$. Finally, oracle access remains the same as in the standard setting, except that in anamorphic oracles, we replace $\mathsf{TS.Sign}_j$ with its corresponding $\mathsf{TSAS.Sign}_j$ variant.

Definition 14 (Malicious correctness). *We say that an* TSAS *scheme has malicious correctness (or correctness with corruptions) if for all* 1^λ*, for any p.p.t* $\mathcal{A}$*, for all threshold parameter* (n, t)*, the following holds:* $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{TS\text{-}Correct}}(1^\lambda, n, t) \leq \mathsf{negl}(\lambda)$*, where* $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{TS\text{-}Correct}}(1^\lambda, n, t) = \left|\Pr\left[\mathsf{ExpTS\text{-}Correct}^{\mathcal{A}}_{\mathsf{TSAS}}(1^\lambda, n, t) = 1\right]\right|$ *with the experiments* $\mathsf{ExpTS\text{-}Correct}$ *defined in Figure 8.*

$\mathsf{ExpTS\text{-}Correct}^{\mathcal{A}}_{\mathsf{TSAS}}(1^\lambda, n, t)$

$(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TS.KGen}(1^\lambda, n, t)$
$(\mathsf{Cor}, k, \mathsf{msg}^*, \mathsf{amsg}^*) \leftarrow \mathcal{A}(\mathsf{pk})$;
$(\mathsf{adk}^S, \mathsf{adk}) \leftarrow \mathsf{TSAS.KGen}(1^\lambda, n, t)$
if $\mathsf{Cor} \not\subset [n]$ **or** $|\mathsf{Cor}| \neq t$ **or** $k \notin \mathsf{Cor}$ **then** :
  Abort
for $i \in [n] \setminus \mathsf{Cor}$ **do**
  $\mathsf{msg}_{i,1}, \mathsf{St}_{i,1} \leftarrow \mathsf{TSAS.Sign}_1(\mathsf{sk}_i, \mathsf{msg}, \mathsf{amsg}, \mathcal{J}, \mathsf{adk}^S)$
$\{\mathsf{msg}_{i,1}\}_{i \in \mathsf{Cor}} \leftarrow \mathcal{A}(\{\mathsf{msg}_{i,1}\}_{i \in [n] \setminus \mathsf{Cor}})$
for $i \in [n] \setminus \mathsf{Cor}$ **do**
  $s_i \leftarrow \mathsf{TSAS.Sign}_2(\mathsf{St}_{i,1}, \{\mathsf{msg}_{j,1}\}_{j \in \mathcal{J} \setminus \{i\}})$
$\{\sigma^{\mathsf{a}}, \perp\} \leftarrow \mathsf{TSAS.Combine}(\mathsf{pk}, s_{i_1}, \ldots, s_{i_t})$
return $\mathsf{TSAS.Dec}(\mathsf{adk}, \sigma^{\mathsf{a}}) \neq \mathsf{amsg}^*$

Fig. 8. Malicious correctness experiment $\mathsf{ExpTS\text{-}Correct}^{\mathcal{A}}_{\mathsf{TSAS}}$ for TSAS .

We can trivially send anamorphic messages by adding ciphertext to threshold signatures, but then the dictator can distinguish such an anamorphic signature

from "real" ones. Therefore, we need to formalize the concept of anamorphism in the threshold setting into the indistinguishability of the two following games. First, a real game in which the adversary receives keys generated by and interactively signs within algorithms of a TS scheme. Second, an anamorphic game in which algorithms of TSAS generate additional double keys and signing messages.

We refer to the beginning of Section 5.2 for the adversary model.

Definition 15 (Threshold-sender anamorphic indistinguishability). *We say that an* TSAS *scheme has anamorphic indistinguishability if for all* 1^λ*, for all threshold parameter* (n, t)*, for any p.p.t. algorithm* $\mathcal{D}$*, the following holds:*

$$\mathbf{Adv}^{\mathcal{D}}_{\mathsf{TS\text{-}Anam}}(1^\lambda, n, t) \leq \mathsf{negl}(\lambda),$$

where $\mathbf{Adv}^{\mathcal{D}}_{\mathsf{TS\text{-}Anam}}(1^\lambda, n, t) = \left|\Pr\left[\mathsf{ExpTS\text{-}Real}^{\mathcal{D}}_{\mathsf{TSAS}}(1^\lambda, n, t) = 1\right] - \Pr\left[\mathsf{ExpTS\text{-}Anam}^{\mathcal{D}}_{\mathsf{TSAS}}(1^\lambda, n, t) = 1\right]\right|$

with the experiments ExpTS-Real *and* ExpTS-Anam *defined in Figure 9.*

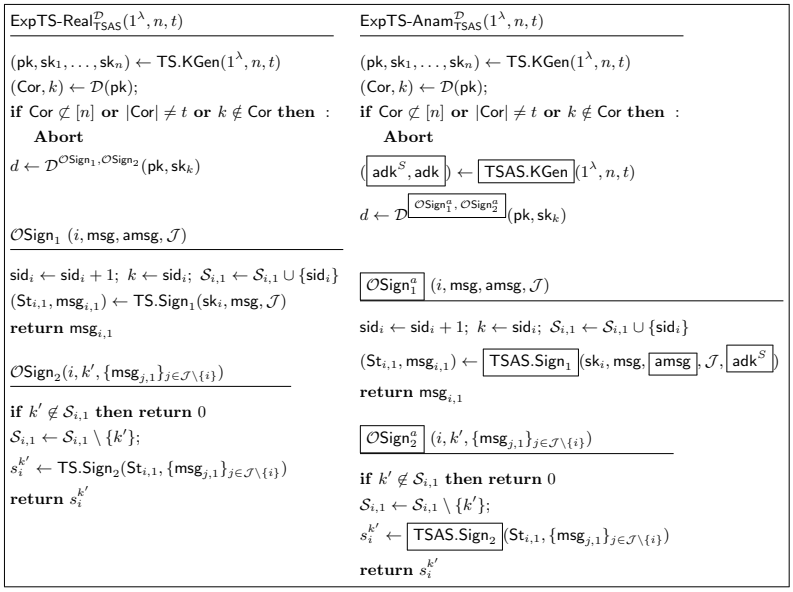

Fig. 9. TSAS security experiments ExpTS-Real$^{\mathcal{D}}_{\mathsf{TSAS}}$ and ExpTS-Anam$^{\mathcal{D}}_{\mathsf{TSAS}}$.

In Sections 5.3, D.2 - D.4, we present the impossibility of constructing TSAS schemes associated with multiple state-of-the-art threshold signature schemes. Our results rule out an entire class of anamorphic embedding techniques that embed anamorphic messages within the output signatures, thereby excluding the random-coin embedding subclass (Figure 1) as well.

5.3 FROST

FROST is a Schnorr threshold signature scheme initially proposed by Komlo and Goldberg. It is currently undergoing standardization by the IETF [12] and NIST [33], with multiple independent implementations already done. FROST's signature protocol is semi-interactive and has well-studied optimizations.

We choose the FROST3 [39] variant for presentation, but in fact, the results in this subsection immediately work for other variants of FROST as well. A concrete description for FROST3 is provided in Figure 16.Briefly, the scheme consists of two rounds in which signers distributedly compute: (1) D and E that later combines to the nonce $R = DE^b$ with some binding coefficient b; and (2) the response s of the final Schnorr signature (R, s).

Theorem 6. FROST3 *is not anamorphic. In particular, there is no threshold-sender anamorphic signature scheme associated with* FROST3 *as in Definition 12 that is maliciously correct and has anamorphic indistinguishability.*

The theorem holds for any choice of (n, t), in which the dictator only needs to compromise only ONE single party ($|\mathsf{Cor}| = 1$). For the sake of simplicity, WLOG, we only present the case $n = t = 2$; however, the proof extends straightforwardly to any parameters (n, t) of a FROST3 threshold signature scheme.

Sketch proof. We outline the proof intuition as follows: assuming FROST3 satisfies anamorphic indistinguishability, we subsequently demonstrate that it fails to achieve malicious correctness, thereby preventing the sender from embedding the anamorphic message in the final signature. We consider a semi-honest, rushing adversary $\mathcal{A}$. Assuming anamorphic indistinguishability, the malicious correctness of the anamorphic FROST3 can be deduced solely by analyzing the structure of the standard version, as the two schemes must behave similarly. The impossibility arises from the fact that the final signature σ appears to be a valid Schnorr signature under the joint public key, making its output distribution unpredictable to the honest signer (Lemma 2). Since the signer must assign each signature to a decryption label (e.g., 0 or 1), and these labels must partition the signature space, any such labeling maps at most half of the signatures to a given message. But because the signer cannot predict the final signature, since a rushing $\mathcal{A}$ can always add a signature share that acts as random noise on the combined anamorphic shares, it must abort with probability at least $1/2$ to maintain decryption correctness - a behavior noticeable to the dictator. It is worth noting that this strategy may be viewed as a general framework for assessing the feasibility of transforming a threshold signature scheme into its sender-anamorphic counterpart. The evaluation relies solely on analyzing whether the signature output distribution of the underlying standard scheme is unpredictable or not.

Lemma 1. FROST3 *is signature-preserving as in Definition 4.*

Proof of Lemma 1. Define the following algorithms TS.SkAgg and TS.StAgg

$$\mathsf{TS.SkAgg}(\{\mathsf{sk}_i\}_{i\in\mathcal{J}}) := \sum_{i\in\mathcal{J}} \Lambda_i \mathsf{sk}_i;\ \ \mathsf{TS.StAgg}(\{(d_i, e_i)\}_{i\in\mathcal{J}}) := \sum_{i\in\mathcal{J}} d_i + b\sum_{i\in\mathcal{J}} e_i$$

where $\Lambda_i = \mathsf{Lagrange}(\mathcal{J}, i)$ and $b \leftarrow \mathsf{H}_{\mathrm{non}}(X, \mathcal{J}, (g^{\sum_{i\in\mathcal{J}} d_i}, g^{\sum_{i\in\mathcal{J}} e_i}), \mathsf{msg})$. A simple calculation shows that Equation (1) holds. □

Lemma 2. *There exists $\mathcal{D}$ in the real experiment $\mathsf{ExpTS\text{-}Real}^{\mathcal{D}}_{\mathsf{TSAS}}(1^\lambda, n, t)$ defined in Definition 15 such that, for every security parameter 1^λ, for every quorum $\mathcal{J}$, and for all messages msg and anamorphic messages amsg, the resulting signature σ has a distribution Dist_0, which is perfectly indistinguishable from the distribution of Schnorr signatures $\mathsf{Dist}_1 = \{\sigma \mid r \leftarrow\!\$\ \mathbb{Z}_q\}$ under the joined key $x = \mathsf{TS.SkAgg}(\{\mathsf{sk}_i\}_{i\in\mathcal{J}})$, where $\sigma \leftarrow r + cx$ and $c \leftarrow \mathsf{H}(g^x, g^r, \mathsf{msg})$.*

Proof. From Lemma 1, the final signature σ can be rewritten as a standard Schnorr signature $\sigma \leftarrow r+cx$ where $r = d_1+d_2+(e_1+e_2)b$, $b \leftarrow \mathsf{H}_{\mathrm{non}}(X, \mathcal{J}, \rho, \mathsf{msg})$ and x is the DL of the combined public key X. As (d_2, e_2) is uniform over $\mathbb{Z}_q$ and only known after (d_1, e_1) is fixed, $(d_1 + d_2)$ and $(e_1 + e_2)$ are uniform over $\mathbb{Z}_q$. As $\mathsf{H}_{\mathrm{non}}$ is modeled as a random oracle, b is uniform over $\mathbb{Z}_q$ and so is r. □

Intuitively, this lemma tells us that only a rushing adversary can make the threshold signature unpredictable to honest signers. In the proof of Theorem 6 below, we show that the adversary can similarly influence the distribution of the anamorphic signature.

Proof of Theorem 6. Let TSAS be a threshold sender anamorphic signature scheme associated with $\mathsf{FROST3}$. Assume that TSAS has anamorphic indistinguishability. We prove this theorem by constructing an adversary $\mathcal{A}$ against the malicious correctness of TSAS. $\mathcal{A}$ controls signer S_2, while S_1 is honest, acting as an anamorphic sender. S_2 starts two signing executions on $(\mathsf{msg}_1, \mathsf{amsg}_1)$ and $(\mathsf{msg}_2, \mathsf{amsg}_2)$ even chosen by honest signer S_1. $\mathcal{A}$ controls S_2 to follow the protocol almost perfectly, except that it always sends its first-round message after S_1 does. We will show that the only possibility for the correctness equation to hold for both executions is that S_1 aborts at least one execution, with probability at least $1/2$.

Fix one execution among the two above. Let $(m^a_{1,1}, m^a_{1,2})$ be the first-round and second-round messages of S_1 using $\mathsf{TSAS.Sign}_1$ and $\mathsf{TSAS.Sign}_2$. Assuming anamorphic indistinguishability, we deduce that $(m^a_{1,1}, m^a_{1,2})$ must be of the form: (i) $m^a_{1,1} = (D_1, E_1)$ where $D_1, E_1 \in \mathbb{G}$ close to uniform distribition; and (ii) $m^a_{1,2} = s_1 \in \mathbb{Z}_q$ satisfying partial verification $(g^{s_1} = D_1 E_1^b \mathsf{pk}_1^{c\Lambda_1})$. In other words, these messages from anamorphic signing oracles are computationally indistinguishable from those output from $\mathsf{TS.Sign}_1$ and $\mathsf{TS.Sign}_2$.

Therefore, if we construct $\mathcal{A}$ exactly as $\mathcal{D}$ in Lemma 2, but acting in $\mathsf{ExpTS\text{-}Anam}^{\mathcal{D}}_{\mathsf{TSAS}}(1^\lambda, n, t)$, the lemma holds for the resulting anamorphic signature σ^{a} as well, i.e.,

$$\begin{aligned}&\Pr[\mathsf{amsg}_i \leftarrow \mathsf{TSAS.Dec}(\sigma^{\mathsf{a}}_i) \mid \sigma^{\mathsf{a}}_i \leftarrow \mathcal{A}]\\ =&\Pr\left[\mathsf{amsg}_i \leftarrow \mathsf{TSAS.Dec}(\sigma^{\mathsf{a}}_i) \;\middle|\; \sigma^{\mathsf{a}}_i \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg}; r^i), r^i \leftarrow\!\$\ \mathbb{Z}_q\right] + \mathsf{negl}(\lambda)\end{aligned}$$

for $i = 1, 2$ and $\mathsf{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ being the Schnorr digital signature scheme. Define the right-hand side by $\Pr[E_i]$ for $i = 1, 2$. As E_1 and E_2 are mutually exclusive events w.r.t. two distinct amsg_1 and amsg_2. Then we have either $\Pr[E_1] < 1/2 + \mathsf{negl}(\lambda)$ or $\Pr[E_2] < 1/2 + \mathsf{negl}(\lambda)$. Assume the former holds, σ^{a} does not decrypt to amsg_1 with probability at least $1/2 - \mathsf{negl}(\lambda)$, except that S_1 aborts. □

6 Discussion

Anamorphism Implies CPA. One of the observations made in the original anamorphic signature paper [28] is that anamorphism implies CPA security. In the CPA experiment, the adversary provides two anamorphic messages, and while given the anamorphic signature, the adversary must decide which signatures are hidden inside the signature. It is easy to see that CPA uses hybrid arguments based on the anamorphism property. In other words, one can first replace the signature in the CPA experiment with a standard one without anamorphism. The adversary cannot notice this change since otherwise, it would be able to distinguish standard from anamorphic signatures. In the next step, we replace the standard signature with an anamorphic one, hiding the other message.

Unfortunately, the same argumentation does not work for TRAS. The main reason is that in the CPA experiment, we now provide the adversary with a corruption oracle that leaks double keys. While the above idea works for standard anamorphic signatures, it might be the case for some schemes that an adversary can distinguish a standard signature from an anamorphic signature with just one double key. A different view on this could be to look at a standard anamorphic signature as a threshold scheme with one recipient, and then the corruption oracle cannot work since the threshold is one user. In other words, if we did not provide a corruption oracle in the CPA definition of threshold-recipient anamorphic signatures, then CPA would directly be implied by anamorphism.

Potential Side Channel Attack for TRAS. A potential threat to TRAS is side-channel attack. While the security analysis assumes the adversary has access to the output signature and keys, it overlooks the fact that, in real-world scenarios, the adversary may exploit metadata to compromise the anamorphism. A key observation is that the computational workload of the TRAS signing algorithm is substantially more than that of a standard signature, resulting in a significant increase in signing time. This arises from the need for the sender to encrypt the anamorphic message amsg using primitives like threshold encryption, which is more computationally expensive than generating fresh random coins. Thus, a powerful dictator with a mass oversight system or malware devices can benchmark citizens' signing times, enabling a breach of the anamorphic property. To address this issue, one can precompute the anamorphic ciphertext or utilize efficient symmetric primitives while shifting the workload to the recipient's side.

7 Conclusion

Anamorphic signatures let users use signatures to communicate covertly when there are restrictions placed on encryption technologies. Prior solutions let signers

transmit covert messages to individual recipients which is prone to failures when an authoritarian entity compromises the recipient. This work studies how threshold cryptography might address the problem and explores different settings, presenting an initial characterization of the feasibility regime. In doing so, we also present stronger adversarial models where even the secret keys of users are fully controlled by the adversary. However, a full characterization of anamorphic threshold signatures remains open as we only consider a restricted class of techniques that embed messages in the randomness of signatures. Going beyond signatures, an interesting direction of research is to study other cryptographic objects in daily use that offer support for anamorphic messaging.

References

1. Atzei, N., Bartoletti, M., Lande, S., Zunino, R.: A formal model of bitcoin transactions. In: Meiklejohn, S., Sako, K. (eds.) FC 2018. LNCS, vol. 10957, pp. 541–560. Springer, Berlin, Heidelberg (Feb / Mar 2018). https://doi.org/10.1007/978-3-662-58387-6_29
2. Banfi, F., Gegier, K., Hirt, M., Maurer, U., Rito, G.: Anamorphic encryption, revisited. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part II. LNCS, vol. 14652, pp. 3–32. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58723-8_1
3. Bendlin, R., Damgård, I.: Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 201–218. Springer, Berlin, Heidelberg (Feb 2010). https://doi.org/10.1007/978-3-642-11799-2_13
4. Boneh, D., Ding, X., Tsudik, G., Wong, C.M.: A method for fast revocation of public key certificates and security capabilities. In: Wallach, D.S. (ed.) USENIX Security 2001. USENIX Association (Aug 2001)
5. Braun, L., Damgård, I., Orlandi, C.: Secure multiparty computation from threshold encryption based on class groups. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, vol. 14081, pp. 613–645. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38557-5_20
6. Canetti, R., Goldwasser, S.: An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 90–106. Springer, Berlin, Heidelberg (May 1999). https://doi.org/10.1007/3-540-48910-X_7
7. Catalano, D., Giunta, E., Migliaro, F.: Anamorphic encryption: New constructions and homomorphic realizations. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part II. LNCS, vol. 14652, pp. 33–62. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58723-8_2
8. Catalano, D., Giunta, E., Migliaro, F.: Limits of black-box anamorphic encryption. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part II. LNCS, vol. 14921, pp. 352–383. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68379-4_11
9. CBP: Border search of electronic devices at ports of entry. United States Customs and Border Protection (2024), https://www.cbp.gov/travel/cbp-search-authority/border-search-electronic-devices

10. CBSA: Examining personal digital devices at the canadian border. Canada Border Services Agency (2024), https://www.cbsa-asfc.gc.ca/travel-voyage/edd-ean-eng.html
11. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO'92. LNCS, vol. 740, pp. 89–105. Springer, Berlin, Heidelberg (Aug 1993). https://doi.org/10.1007/3-540-48071-4_7
12. Connolly, D., Komlo, C., Goldberg, I., Wood, C.A.: Two-Round Threshold Schnorr Signatures with FROST. Internet-Draft draft-irtf-cfrg-frost-15, Internet Engineering Task Force (Sep 2023), https://datatracker.ietf.org/doc/draft-irtf-cfrg-frost/15/, work in Progress
13. Cremers, C., Loss, J., Wagner, B.: A holistic security analysis of Monero transactions. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part III. LNCS, vol. 14653, pp. 129–159. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58734-4_5
14. Crites, E.C., Komlo, C., Maller, M.: Fully adaptive Schnorr threshold signatures. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, vol. 14081, pp. 678–709. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38557-5_22
15. Dao, Q., Miller, J., Wright, O., Grubbs, P.: Weak fiat-shamir attacks on modern proof systems. In: 2023 IEEE Symposium on Security and Privacy. pp. 199–216. IEEE Computer Society Press (May 2023). https://doi.org/10.1109/SP46215.2023.10179408
16. De Feo, L., Delpech de Saint Guilhem, C., Fouotsa, T.B., Kutas, P., Leroux, A., Petit, C., Silva, J., Wesolowski, B.: Séta: Supersingular encryption from torsion attacks. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part IV. LNCS, vol. 13093, pp. 249–278. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-030-92068-5_9
17. del Pino, R., Katsumata, S., Prest, T., Rossi, M.: Raccoon: A masking-friendly signature proven in the probing model. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part I. LNCS, vol. 14920, pp. 409–444. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68376-3_13
18. Desmedt, Y.: Society and group oriented cryptography: A new concept. In: Pomerance, C. (ed.) CRYPTO'87. LNCS, vol. 293, pp. 120–127. Springer, Berlin, Heidelberg (Aug 1988). https://doi.org/10.1007/3-540-48184-2_8
19. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 307–315. Springer, New York (Aug 1990). https://doi.org/10.1007/0-387-34805-0_28
20. Devevey, J., Libert, B., Nguyen, K., Peters, T., Yung, M.: Non-interactive CCA2-secure threshold cryptosystems: Achieving adaptive security in the standard model without pairings. In: Garay, J. (ed.) PKC 2021, Part I. LNCS, vol. 12710, pp. 659–690. Springer, Cham (May 2021). https://doi.org/10.1007/978-3-030-75245-3_24
21. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: A lattice-based digital signature scheme. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2018**, 238–268 (2018), https://api.semanticscholar.org/CorpusID:3593118
22. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 10–18. Springer, Berlin, Heidelberg (Aug 1984). https://doi.org/10.1007/3-540-39568-7_2

23. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 295–310. Springer, Berlin, Heidelberg (May 1999). https://doi.org/10.1007/3-540-48910-X_21
24. Hazay, C., Lindell, Y.: A note on the relation between the definitions of security for semi-honest and malicious adversaries. Cryptology ePrint Archive, Paper 2010/551 (2010), https://eprint.iacr.org/2010/551
25. ICAO: Part 11: Security mechanisms for mrtds. Doc 9303: Machine Readable Travel Documents (2021), https://www.icao.int/publications/pages/publication.aspx?docnum=9303
26. ICAO: Member states (2025), https://www.icao.int/about-icao/pages/member-states.aspx
27. Komlo, C., Goldberg, I.: FROST: Flexible round-optimized Schnorr threshold signatures. In: Dunkelman, O., Jr., M.J.J., O'Flynn, C. (eds.) SAC 2020. LNCS, vol. 12804, pp. 34–65. Springer, Cham (Oct 2020). https://doi.org/10.1007/978-3-030-81652-0_2
28. Kutylowski, M., Persiano, G., Phan, D.H., Yung, M., Zawada, M.: Anamorphic signatures: Secrecy from a dictator who only permits authentication! In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 759–790. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38545-2_25
29. Mackey, R.: French scientist denied us entry after phone messages critical of trump found. The Guardian (2025), https://www.theguardian.com/us-news/2025/mar/19/trump-musk-french-scientist-detained
30. Nick, J., Ruffing, T., Seurin, Y.: MuSig2: Simple two-round Schnorr multi-signatures. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 189–221. Springer, Cham, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84242-0_8
31. NIST: Nist call for additional digital signature schemes for the post-quantum cryptography standardization process (2022), https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf
32. NIST: National institute of standards and technology: Digital signature standard (dss) - fips 186-5. Tech. rep., U.S. Department of Commerce (2023), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf
33. NIST: Nist first call for multi-party threshold schemes (2023), https://csrc.nist.gov/pubs/ir/8214/c/ipd
34. Pedersen, T.P.: A threshold cryptosystem without a trusted party (extended abstract) (rump session). In: Davies, D.W. (ed.) EUROCRYPT'91. LNCS, vol. 547, pp. 522–526. Springer, Berlin, Heidelberg (Apr 1991). https://doi.org/10.1007/3-540-46416-6_47
35. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 129–140. Springer, Berlin, Heidelberg (Aug 1992). https://doi.org/10.1007/3-540-46766-1_9
36. Persiano, G., Phan, D.H., Yung, M.: Anamorphic encryption: Private communication against a dictator. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 34–63. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3_2
37. Persiano, G., Phan, D.H., Yung, M.: Public-key anamorphism in (CCA-secure) public-key encryption and beyond. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024,

Part II. LNCS, vol. 14921, pp. 422–455. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68379-4_13
38. Pino, R.D., Katsumata, S., Maller, M., Mouhartem, F., Prest, T., Saarinen, M.J.O.: Threshold raccoon: Practical threshold signatures from standard lattice assumptions. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part II. LNCS, vol. 14652, pp. 219–248. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58723-8_8
39. Ruffing, T., Ronge, V., Jin, E., Schneider-Bensch, J., Schröder, D.: ROAST: Robust asynchronous schnorr threshold signatures. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 2551–2564. ACM Press (Nov 2022). https://doi.org/10.1145/3548606.3560583
40. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 239–252. Springer, New York (Aug 1990). https://doi.org/10.1007/0-387-34805-0_22
41. Wang, Y., Chen, R., Huang, X., Yung, M.: Sender-anamorphic encryption reformulated: Achieving robust and generic constructions. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part VI. LNCS, vol. 14443, pp. 135–167. Springer, Singapore (Dec 2023). https://doi.org/10.1007/978-981-99-8736-8_5

A Discussion

A.1 A possible way to overcome the limitation in real-world setting

Embedding anamorphic messages within the signature is infeasible in the context of threshold signatures. A similar challenge arises when attempting to embed the message within the sender's public key. This is primarily due to the distributed nature of threshold signature's key generation, where no single party has complete control over the final public key. Instead, the dictator can influence the process in ways that make the final public key unpredictable to the user, preventing reliable encoding of hidden information. Given these constraints, the only practical approach is to embed the anamorphic message in the regular message.

A possible real-world setting. We begin by analyzing whether an anamorphic message can be embedded within a regular message in the context of a standard digital signature scheme. This approach requires incorporating an auxiliary bit string into the message to facilitate the embedding process. However, this introduces a fundamental challenge: the dictator—who has significant influence over the system—can easily detect the presence of the auxiliary bit string. If the dictator identifies this irregularity, they may reject or request to modify the message, compromising the anamorphic property and rendering this approach ineffective. To circumvent this issue, we seek real-world scenarios where non-linguistic strings can naturally be appended to a message without raising suspicion. Specifically, we focus on two settings: *signature schemes utilizing the Fiat-Shamir transformation* and *cryptocurrency transactions*, both of which inherently allow for the inclusion of additional structured data.

Many widely used signature schemes, such as those in [11] and [40], rely on the transformation of an interactive proof system into a non-interactive one via

the Fiat-Shamir transformation [15]. In the original interactive protocol, the signing process follows these steps:

1. *Commitment:* The prover generates and sends a commitment to the verifier.
2. *Challenge:* The verifier responds with a randomly chosen challenge.
3. *Response:* The prover computes a response for the challenge and their secret key.

To eliminate the need for interaction, the Fiat-Shamir transformation replaces the verifier's challenge with a deterministic hash function. Instead of relying on a third-party verifier, the prover generates the challenge by hashing the commitment and the message. The prover then computes the response based on this challenge and their private key. Recent research [15] has identified a crucial security weakness in this transformation, termed Weak Fiat-Shamir Attacks. To mitigate this risk, it is recommended that the hash function incorporate not only the commitment and message but also additional public parameters, including the public key. This refinement enhances security and has been widely adopted in deployed-proof systems. This adjustment is particularly relevant for embedding anamorphic messages. Since public parameters (e.g., public key) are now incorporated into the hash commitment, they allow encoding hidden information within the signing process without disrupting the scheme's integrity.

Cryptocurrency transactions are another setting where auxiliary information can be naturally embedded into a message—without dictator interference. Unlike standard digital signatures, cryptocurrency transactions inherently involve structured metadata, offering a channel for including additional data. While we do not assume the presence of an explicit auxiliary bit string in the transaction metadata, we observe that at least the recipient's public key remains outside the dictator's control. This is particularly important because the dictator generally does not dictate the recipient's choice of public key as it might be outside the dictator's jurisdiction. Additionally, many cryptocurrencies implement multi-user accounts, requiring payment transactions to be confirmed using multi-signature or threshold signature schemes. These mechanisms involve multiple parties, further diluting the dictator's ability to control every aspect of the signing process.

Both Fiat-Shamir-based signature schemes and cryptocurrency transactions provide structured opportunities to embed anamorphic messages within a regular message. These approaches enable the sender to append an additional bit string—either within the public key hashing process or within transaction metadata—without raising suspicion. However, these methods come with limitations:

- Single message embedding: They allow for embedding only a single anamorphic message within the public key, restricting the amount of hidden information that can be transmitted.
- Dependence on key control: If the dictator controls key generation, even these approaches become infeasible, as they can manipulate the keys to disrupt the embedding process.

Despite these constraints, these settings provide valuable mechanisms for enabling undetected anamorphic communication.

Stealth address to the rescue. In cryptocurrency transactions, the sender signs the transaction details, including the transfer amount and recipient's public key (address) [1]. The recipient then uses the corresponding private key to authorize a new transaction. To ensure privacy, Monero proposes their signature with stealth addresses as a privacy-enhancing feature that ensures transaction recipients remain untraceable on the blockchain. As described in [13], when a payee initiates a transaction, they generate a unique, one-time public address $P = g^{H(A^r)} \cdot B$ (a stealth address) derived from the recipient's public key $(A, B) = (g^a, g^b)$ and a random value $R = g^r$ generated on the fly. This ensures that each transaction appears as though it is sent to a new, unlinkable address, even though the recipient's wallet (holding the master secret key (a, b)) ultimately controls all funds. Given a so-called view key(a, B), anyone can scan the blockchain to detect stealth addresses, but only the holder of (a, b) can spend those funds. This prevents observers from linking multiple transactions to a single recipient, significantly enhancing anonymity and fungibility.

Senders can exploit the randomness of R to encode amsg via rejection sampling, i.e., repeatedly generating R until it satisfies a predefined predicate. For example, senders can enforce that the first ℓ bits of R match the encryption of amsg. However, in practice, ℓ is limited, allowing only a modest payload of approximately 30–40 bits, making this inefficient for larger messages.

Another approach is to encode the anamorphic message amsg directly within the public key P by modifying its computation as follows: $P = g^{H(A^r)} \cdot B \cdot g^{\mathsf{amsg}}$. This effectively "shifts" P by g^{amsg}, rendering the standard view key obsolete. However, if the recipient already knows the expected stealth address (e.g., it was communicated by the sender), they can reconstruct the original address: $P' = g^{H(A^r)} \cdot B$ and extract amsg by computing: $X = P/P' = g^{\mathsf{amsg}}$. The recipient can efficiently recover amsg by solving the discrete logarithm of X using the baby-step giant-step algorithm. Crucially, the modified address P remains fully spendable by the recipient, ensuring that the embedded message and embedding process remain undetectable to an external observer, including the dictator.

Integrating anamorphic messages into signature schemes involving stealth addresses introduces several technical challenges that require a more rigorous and formalized framework. A comprehensive analysis must address security guarantees, such as ensuring that the embedding process does not compromise the anonymity and unlinkability properties of Monero transactions. Additionally, formal definitions are needed to characterize the indistinguishability of anamorphic message embedding from standard transaction generation. Given these open questions, we leave the concrete formalization and rigorous security analysis of anamorphic message embedding in stealth-address-based signature schemes as an open problem for future research.

B Background

B.1 Chernoff Bound

Theorem 7. *For $i = 1, \ldots, n$ let X_i be independent random variables that take the value 1 with probability p_i and the value 0 with probability $1 - p_i$. Suppose at least one p_i is non-zero. Let $X = \sum_{i \in [n]} X_i$ and let $\mu = E[X] = \sum_{i \in [n]} p_i$. For $\delta > 2 \cdot e - 1$ and the Euler's number e, we then have:*

$$\Pr[X > (1 + \delta) \cdot \mu] < 2^{-\delta \cdot \mu}.$$

B.2 Digital Signature Scheme

Definition 16 (Correctness). *We say that a digital signature scheme S is correct if, for all public parameter 1^λ and all message msg in the associated message space, the following event holds:*

$$\Pr\left[\mathsf{S.Vf}(\mathsf{pk}, \mathsf{msg}, \sigma) \middle| \sigma \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; \mathsf{st})\right] = 1,$$

where the probability is over the random variable $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{S.KGen}(1^\lambda)$ and the random coins of $\mathsf{S.Sign}$.

Definition 17 (Unforgeability). *We say that a digital signature scheme S scheme achieves unforgeability security if for all parameter 1^λ, for any polynomial time algorithm $\mathcal{A}$, the following holds:*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{Unf}}(1^\lambda) \leq \mathsf{negl}(\lambda),$$

where $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{Unf}}(1^\lambda) = \Pr\left[\mathsf{ExpUnf}^{\mathcal{A}}_{\mathsf{AS}}(1^\lambda) = 1\right]$ with the experiment ExpUnf defined in Figure 10.

$\mathsf{ExpUnf}^{\mathcal{A}}_{\mathsf{S}}(1^\lambda)$	$\mathsf{oSign}(\mathsf{msg})$
$\mathcal{M} \leftarrow \{\emptyset\};\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{S.KGen}(1^\lambda)$	**if** $\mathsf{msg} \in \mathcal{M}$ **then**
$(\mathsf{msg}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{oSign}}(\mathsf{sk}, \mathsf{pk})$	**return** $\perp$
$b_1 \leftarrow \mathsf{S.Vf}(\mathsf{pk}, \mathsf{msg}^*, \sigma^*)$	$\mathcal{M} \leftarrow \mathcal{M} \cup \{\mathsf{msg}\}$
$b_2 \leftarrow \mathsf{msg}^* \notin \mathcal{M}$	$\sigma \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}, \mathsf{st})$
return $b_1 \wedge b_2$	**return** σ

Fig. 10. Unforgeability security experiment $\mathsf{ExpUnf}^{\mathcal{A}}_{\mathsf{S}}$.

B.3 Symmetric Encryption Scheme

We say that a symmetric encryption scheme is correct if for all public parameters 1^λ, all plaintext pt in the associated plaintext space, and all $\mathsf{ct} \leftarrow \mathsf{E.Enc}(\mathsf{sk}, \mathsf{pt})$, the following event holds:

$$\Pr\left[\mathsf{pt} \leftarrow \mathsf{E.Dec}(\mathsf{sk}, \mathsf{ct})\right] = 1,$$

where the secret key is generated from $\mathsf{sk} \leftarrow \mathsf{E.KGen}(1^\lambda)$.

Definition 18 (Semantic security). *We say that symmetric encryption scheme* E *has semantic security (or IND-CPA secure) if for all parameter* 1^λ*, for any polynomial-time algorithm* $\mathcal{A}$*, the following holds:*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{CPA}}(1^\lambda) \leq \mathsf{negl}(\lambda),$$

where $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{CPA}}(1^\lambda) = \left|\Pr\left[\mathsf{Exp\text{-}CPA}^{\mathcal{A}}_{\mathsf{E}}(1^\lambda) = 1\right] - \frac{1}{2}\right|$ *with the IND-CPA experiment* $\mathsf{Exp\text{-}CPA}$ *defined in Figure 11.*

Definition 19 (Pseudorandom ciphertext). *We say that symmetric encryption scheme* E *has pseudorandom ciphertext if for all parameter* 1^λ*, for any polynomial-time algorithm* $\mathcal{A}$*, the following holds:*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{PRC}}(1^\lambda) \leq \mathsf{negl}(\lambda),$$

where $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{PRC}}(1^\lambda) = \left|\Pr\left[\mathsf{Exp\text{-}PRC}^{\mathcal{A}}_{\mathsf{E}}(1^\lambda) = 1\right] - \frac{1}{2}\right|$ *with the pseudorandom ciphertext experiment* $\mathsf{Exp\text{-}PRC}$ *defined in Figure 11. Here, we assume that the encryption scheme* E *for security parameter* 1^λ *encrypts* $n(\lambda)$*-bit plaintexts into* $\ell(\lambda)$*-bit ciphertexts.*

B.4 Threshold Encryption Scheme

We say that a threshold encryption scheme is correct if for all public parameters 1^λ, all plaintext pt in the associated plaintext space, all positive integers n, t such that $t \leq n$, and all subsets $\mathcal{J} \subseteq [n]$ with $|\mathcal{J}| \geq t$ where $\mathsf{ct} \leftarrow \mathsf{TE.Enc}(\mathsf{pk}, \mathsf{pt})$, the following event holds:

$$\Pr\left[\mathsf{pt} \leftarrow \mathsf{TE.Combine}(\mathsf{pk}, \mathsf{ct}, \{\mathsf{TE.Dec}(\mathsf{pk}, \mathsf{sk}_j, \mathsf{ct})\}_{j \in \mathcal{J}})\right] = 1,$$

where the probability is over random variables $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow\!\!\$\ \mathsf{TE.KGen}(1^\lambda, n, t)$ and the random coins of $\mathsf{TE.Enc}$.

$\mathsf{Exp\text{-}CPA}^{\mathcal{A}}_{\mathsf{E}}(1^\lambda)$	$\mathsf{Exp\text{-}PRC}^{\mathcal{A}}_{\mathsf{E}}(1^\lambda)$	$\mathsf{Enc}_0(\mathsf{pt})$
$\mathsf{sk} \leftarrow \mathsf{E.KGen}(1^\lambda)$	$\mathsf{sk} \leftarrow \mathsf{E.KGen}(1^\lambda)$	$\mathsf{ct} \leftarrow \mathsf{E.Enc}(\mathsf{sk}, \mathsf{pt})$
$(\mathsf{pt}_0, \mathsf{pt}_1) \leftarrow \mathcal{A}^{\mathsf{Enc}_0}(1^\lambda)$	$b \leftarrow\!\!\$\ \{0,1\}$	**return** ct
$b \leftarrow\!\!\$\ \{0,1\}$	$b' \leftarrow \mathcal{A}^{\mathsf{Enc}_b}(\mathsf{pk}, \mathsf{ct}_b)$	$\mathsf{Enc}_1(\mathsf{pt})$
$\mathsf{ct}_b \leftarrow \mathsf{E.Enc}(\mathsf{sk}, \mathsf{pt}_b)$	**return** $b = b'$	$\mathsf{ct} \leftarrow\!\!\$\ \{0,1\}^{\ell(\lambda)}$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc}_0}(\mathsf{ct}_b)$		**return** ct
return $b = b'$		

Fig. 11. The IND-CPA and pseudorandom ciphertext security experiments of a symmetric encryption scheme E.

Definition 20 (Semantic security). *We say that an* TE *scheme has semantic security (or IND-CPA secure) if for all parameter* 1^λ*, for any polynomial-time algorithm* $\mathcal{A}$*, the following holds:*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{TE\text{-}CPA}}(1^\lambda) \leq \mathsf{negl}(\lambda),$$

where $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{TE\text{-}CPA}}(1^\lambda) = \left|\Pr\left[\mathsf{ExpTE\text{-}CPA}^{\mathcal{A}}_{\mathsf{TE}}(1^\lambda) = 1\right] - \frac{1}{2}\right|$ *with the IND-CPA experiment* ExpTE-CPA *defined in Figure 12.*

$\mathsf{ExpTE\text{-}CPA}^{\mathcal{A}}_{\mathsf{TE}}(1^\lambda)$	$\mathsf{Corr}(i)$
$\mathcal{J} \leftarrow \emptyset;\ (n,t) \leftarrow \mathcal{A}(1^\lambda)$	**if** $i \notin [n]$ **or** $i \in \mathcal{J}$ **or** $\lvert\mathcal{J}\rvert \geq t-1$
$(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TE.KGen}(1^\lambda, n, t)$	**then return** $\perp$
$(\mathsf{pt}_0, \mathsf{pt}_1) \leftarrow \mathcal{A}^{\mathsf{Corr}}(\mathsf{pk}, \mathcal{J})$	$\mathcal{J} \leftarrow \mathcal{J} \cup \{i\}$
$b \leftarrow\!\!\$\ \{0,1\}$	**return** sk_i
$\mathsf{ct}_b \leftarrow \mathsf{TE.Enc}(\mathsf{pk}, \mathsf{pt}_b)$	
$b' \leftarrow \mathcal{A}^{\mathsf{Corr}}(\mathsf{ct}_b)$	
return $b = b'$	

Fig. 12. Threshold encryption IND-CPA security experiment $\mathsf{ExpTE\text{-}CPA}^{\mathcal{A}}_{\mathsf{TE}}$.

Definition 21 (Pseudorandom ciphertext). *We say that an* TE *scheme has pseudorandom ciphertext if for all parameter* 1^λ*, for any polynomial-time algorithm* $\mathcal{A}$*, the following holds:*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{TE\text{-}PRC}}(1^\lambda) \leq \mathsf{negl}(\lambda),$$

where $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{TE\text{-}PRC}}(1^\lambda) = \left|\Pr\left[\mathsf{ExpTE\text{-}PRC}^{\mathcal{A}}_{\mathsf{TE}}(1^\lambda) = 1\right] - \frac{1}{2}\right|$ *with the pseudorandom ciphertext experiment* ExpTE-PRC *defined in Figure 13. Here, we assume that the* TE *for security parameter* 1^λ *encrypts* $n(\lambda)$*-bit plaintexts into* $\ell(\lambda)$*-bit ciphertexts.*

$\mathsf{ExpTE\text{-}PRC}^{\mathcal{A}}_{\mathsf{TE}}(1^\lambda)$	$\mathsf{Enc}_0(\mathsf{pt})$
$\mathcal{J} \leftarrow \emptyset;\ (n,t) \leftarrow \mathcal{A}(1^\lambda)$	$\mathsf{ct} \leftarrow \mathsf{TE.Enc}(\mathsf{pk}, \mathsf{pt})$
$(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TE.KGen}(1^\lambda, n, t)$	**return** ct
$b \leftarrow\!\!\$\ \{0,1\}$	$\mathsf{Enc}_1(\mathsf{pt})$
$b' \leftarrow \mathcal{A}^{\mathsf{Enc}_b}(\mathsf{pk}, \mathsf{ct}_b)$	$\mathsf{ct} \leftarrow\!\!\$\ \{0,1\}^{\ell(\lambda)}$
return $b = b'$	**return** ct

Fig. 13. The pseudorandom ciphertext experiment $\mathsf{ExpTE\text{-}PRC}^{\mathcal{A}}_{\mathsf{TE}}$ of a threshold encryption scheme.

B.5 Threshold Signature Scheme

We say that a threshold signature scheme is correct if for all public parameters 1^λ, all messages msg in the associated message space, all positive integers n, t such that $t \leq n$, and all subsets $\mathcal{J} \subseteq [n]$ with $|\mathcal{J}| \geq t$, the following event holds:

$$\Pr\left[\begin{array}{c}\mathsf{TS.Vf}(\mathsf{pk},\\ \mathsf{msg},\sigma) = 1\end{array}\middle|\begin{array}{c}\sigma \leftarrow \mathsf{TS.Combine}(\mathsf{pkc}, \{s_j\}_{j\in\mathcal{J}})\\ s_j \leftarrow \mathsf{TS.Sign}(\mathsf{sk}_j, \mathsf{msg})\end{array}\right] = 1,$$

where the probability is over the random variables $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow\!\!\$\ \mathsf{TS.KGen}(1^\lambda, n, t)$ and the random coins of TS.Sign.

Definition 22 (Unforgeability). *We say that a threshold signature scheme* TS *achieves unforgeability security if for all parameter* 1^λ*, for any polynomial time algorithm* $\mathcal{A}$*, the following holds:*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{TS\text{-}UF}}(1^\lambda) \leq \mathsf{negl}(\lambda),$$

where $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{TS\text{-}UF}}(1^\lambda) = \Pr\left[\mathsf{ExpTS\text{-}UF}^{\mathcal{A}}_{\mathsf{TS}}(1^\lambda) = 1\right]$ *with the experiment* ExpTS-UF *defined in Figure 14.*

$\mathsf{ExpTS\text{-}UF}^{\mathcal{A}}_{\mathsf{TS}}(1^\lambda)$

$\mathcal{J}_{\mathsf{cr}} \leftarrow \emptyset;\ (n,t) \leftarrow \mathcal{A}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{TS.KGen}(1^\lambda, n, t)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Corr}, \mathcal{O}\mathsf{Sign}_1, \mathcal{O}\mathsf{Sign}_2}(\mathsf{pk}, \mathcal{J}_{\mathsf{cr}})$
$b_1 \leftarrow \mathsf{TS.Vf}(\mathsf{pk}, \mathsf{msg}^*, \sigma^*)$
$b_2 \leftarrow |Sigs[m^*]| = 0$
return $b_1 \wedge b_2$

$\mathsf{Corr}(i)$

if $i \notin [n]$ **or** $i \in \mathcal{J}_{\mathsf{cr}}$ **or** $|\mathcal{J}_{\mathsf{cr}}| \geq t$ **then**
  return $\perp$
$\mathcal{J}_{\mathsf{cr}} \leftarrow \mathcal{J}_{\mathsf{cr}} \cup \{i\}$
return sk_i

$\mathcal{O}\mathsf{Sign}_1\ (i, \mathsf{msg}, \mathsf{amsg}, \mathcal{J})$

$\mathsf{sid}_i \leftarrow \mathsf{sid}_i + 1;\ k \leftarrow \mathsf{sid}_i$
$\mathcal{S}_{i,1} \leftarrow \mathcal{S}_{i,1} \cup \{\mathsf{sid}_i\}$
$\mathsf{St}^k_{i,1}, \mathsf{msg}^k_{i,1} \leftarrow \mathsf{TS.Sign}_1(\mathsf{sk}_i, \mathsf{msg}, \mathcal{J})$
return $\mathsf{msg}^k_{i,1}$

$\mathcal{O}\mathsf{Sign}_2(i, k', \mathsf{msg}'_{i_1,1}, \ldots, \mathsf{msg}'_{i_\ell,1})$

if $k' \notin \mathcal{S}_{i,1}$ **then return** 0
$\mathcal{S}_{i,1} \leftarrow \mathcal{S}_{i,1} \setminus \{k'\}$;
$s_i^{k'} \leftarrow \mathsf{TS.Sign}_2(\mathsf{St}^k_{i,1}, \mathsf{msg}'_{i_1,1}, \ldots, \mathsf{msg}'_{i_\ell,1})$
$Sigs[m] \leftarrow Sigs[m] \cup \{i\}$
return $s_i^{k'}$

Fig. 14. Threshold signature unforgeability experiment $\mathsf{ExpTS\text{-}UF}^{\mathcal{A}}_{\mathsf{TS}}$.

B.6 Anamorphic Signature Scheme

We say that an anamorphic signature scheme AS is *correct* if, for all public parameters 1^λ, all message pairs $(\mathsf{msg}, \mathsf{amsg})$, the following event holds:

$$\Pr\left[\mathsf{amsg} = \mathsf{amsg}' \middle| \begin{array}{c} \sigma^{\mathsf{a}} \leftarrow \mathsf{AS.Sign}(\mathsf{sk}, \mathsf{adk}, \mathsf{msg}, \mathsf{amsg}) \\ \mathsf{amsg}' \leftarrow \mathsf{AS.Dec}(\mathsf{sk}, \mathsf{adk}, \sigma^{\mathsf{a}}) \end{array}\right] = 1,$$

where the probability is over $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{S.KGen}(1^\lambda)$ and $\mathsf{adk} \leftarrow \mathsf{AS.KGen}(1^\lambda)$, and the random coins of AS.Sign.

The following security definitions are borrowed from [28].

Definition 23 (Anamorphic). *We say that an* AS *scheme is anamorphic if for all parameter* 1^λ*, for any polynomial time algorithm* $\mathcal{A}$*, the following holds:*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{Anam}}(1^\lambda) \leq \mathsf{negl}(\lambda),$$

where $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{Anam}}(1^\lambda) = \left|\Pr\left[\mathsf{ExpAnam}^{\mathcal{A}}_{\mathsf{AS}}(1^\lambda) = 1\right] - \frac{1}{2}\right|$ *with the experiment* ExpAnam *defined in Figure 15.*

Definition 24 (Semantic security). *We say that an* AS *scheme has anamorphic semantic security (or anamorphic IND-CPA secure) if for all parameter* 1^λ*, for any polynomial-time algorithm* $\mathcal{A}$*, the following holds:*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{A\text{-}CPA}}(1^\lambda) \leq \mathsf{negl}(\lambda),$$

where $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{A\text{-}CPA}}(1^\lambda) = \left|\Pr\left[\mathsf{ExpA\text{-}CPA}^{\mathcal{A}}_{\mathsf{AS}}(1^\lambda) = 1\right] - \frac{1}{2}\right|$ *with the IND-CPA experiment* ExpA-CPA *defined in Figure 15.*

$\mathsf{ExpAnam}^{\mathcal{A}}_{\mathsf{AS}}(1^\lambda)$	$\mathsf{ExpA\text{-}CPA}^{\mathcal{A}}_{\mathsf{AS}}(1^\lambda)$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{S.KGen}(1^\lambda)$	$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{S.KGen}(1^\lambda)$
$\mathsf{adk} \leftarrow \mathsf{AS.KGen}(1^\lambda)$	$\mathsf{adk} \leftarrow \mathsf{AS.KGen}(1^\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$	$(\mathsf{msg}, \mathsf{amsg}_0, \mathsf{amsg}_1) \leftarrow \mathcal{A}^{\mathsf{aSign}_1}(\mathsf{sk}, \mathsf{pk})$
$b' \leftarrow \mathcal{A}^{\mathsf{aSign}_b}(\mathsf{sk}, \mathsf{pk})$	$b \leftarrow\!\!\$\ \{0,1\}$
return $b = b'$	$\sigma_b \leftarrow \mathsf{AS.Sign}(\mathsf{sk}, \mathsf{adk}, \mathsf{msg}, \mathsf{amsg}_b)$
	$b' \leftarrow \mathcal{A}^{\mathsf{aSign}_1}(\sigma_b)$
	return $b = b'$
$\mathsf{aSign}_0(\mathsf{msg}, \mathsf{amsg})$	$\mathsf{aSign}_1(\mathsf{msg}, \mathsf{amsg})$
$\sigma \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}, \mathsf{st})$	$\sigma \leftarrow \mathsf{AS.Sign}(\mathsf{sk}, \mathsf{adk}, \mathsf{msg}, \mathsf{amsg})$
return σ	**return** σ

Fig. 15. Anamorphic experiment $\mathsf{ExpAnam}^{\mathcal{A}}_{\mathsf{AS}}$ and CPA security experiment $\mathsf{ExpA\text{-}CPA}^{\mathcal{A}}_{\mathsf{AS}}$.

C Threshold-Recipient Anamorphic Signatures

Theorem 2. *The generic* TRAS *construction in Figure 4 is correct and anamorphic.*

Proof. The correctness of the generic TRAS scheme directly follows from the correctness of the underlying TE scheme. We observe that the only difference between a real signature and an anamorphic signature created in TRAS is the random coin used in the signing process. Hence, as the threshold encryption scheme TE has pseudorandom ciphertext and the random coin space of the signature coincides with the ciphertext space of the threshold encryption scheme, using the same argument as in [28], we conclude that the generic TRAS scheme presented in Figure 4 is anamorphic. □

Theorem 3. *For any polynomial-time adversary A, the following holds:*

$$\mathbf{Adv}^{\mathcal{A}}_{\mathsf{TR\text{-}CPA}}(1^\lambda) \leq \mathbf{Adv}^{\mathcal{A}_1}_{\mathsf{TE\text{-}CPA}}(1^\lambda).$$

Proof. We will show that if there exists a PPT adversary $\mathcal{A}$ against the $\mathsf{ExpTR\text{-}CPA}$ experiment, then we can construct an adversary $\mathcal{A}_1$ against the $\mathsf{ExpTE\text{-}CPA}$ of the underlying threshold encryption scheme TE.

- In the setup phase, $\mathcal{A}_1$ start by interacting with $\mathcal{A}$ and receives the public parameter (n, t) as well as the sender's key pair $(\mathsf{sk}^S, \mathsf{pk}^S)$ from $\mathcal{A}$. On receiving the public key of the TE scheme in the $\mathsf{ExpTE\text{-}CPA}$ experiment, $\mathcal{A}_1$ assigns its sender double key as $\mathsf{adk}^S \leftarrow \mathsf{pk}^{\mathsf{TE}}$.
- To answer queries for the oracle Corr, $\mathcal{A}_1$ queries to the oracle Corr in the $\mathsf{ExpTE\text{-}CPA}$ experiment with the same input and outputs what this oracle outputs. On the other hand, to answer queries for the oracle aSign, $\mathcal{A}_1$ runs the algorithm $\mathsf{TRAS.Sign}$ with the secret key sk^S and the double key adk^S and uses the outputs as the oracle's answer.
- When receiving the tuple $(\mathsf{msg}, \mathsf{amsg}_0, \mathsf{amsg}_1)$ from $\mathcal{A}$, $\mathcal{A}_1$ assigns $\mathsf{msg}_0 \leftarrow \mathsf{amsg}_0$, $\mathsf{msg}_1 \leftarrow \mathsf{amsg}_1$ and let the pair $(\mathsf{msg}_0, \mathsf{msg}_1)$ be the challenge in the $\mathsf{ExpTE\text{-}CPA}$ experiment.
- On the receiving the ciphertext ct from the $\mathsf{ExpTE\text{-}CPA}$ experiment, $\mathcal{A}_1$ computes the anamorphic signature as $\sigma^{\mathsf{a}} \leftarrow \mathsf{S.Sign}(\mathsf{sk}^S, \mathsf{msg}; \mathsf{ct})$ by using the ciphertext ct as the random coin and forwards the signature to $\mathcal{A}$. It then outputs what $\mathcal{A}$ outputs.

It is clear that $\mathcal{A}_1$ wins whenever $\mathcal{A}$ wins. Therefore, we have: $\mathbf{Adv}^{\mathcal{A}}_{\mathsf{TR\text{-}CPA}}(1^\lambda) \leq \mathbf{Adv}^{\mathcal{A}_1}_{\mathsf{TE\text{-}CPA}}(1^\lambda)$. □

D Threshold-Sender Anamorphic Signatures

D.1 FROST

FROST3 is the most efficient variant thanks to the ability to aggregate protocol messages before broadcasting them to the signers. In more detail, one can easily add an PreAgg algorithm that aggregates the presignatures shares $\{D_i, E_i\}_{i\in\mathcal{J}}$ from first round into a compact presignature ρ, which consists only of the two products $(D, E) = (\prod_{i\in\mathcal{J}} D_i, \prod_{i\in\mathcal{J}} E_i)$. This presignature can later be input to the second round instead of the entire list of $\{D_i, E_i\}_{i\in\mathcal{J}}$. We omit this optimization in Figure 16 for consistency with the syntax of threshold signatures in Definition 3. A concrete description for FROST3 is provided in Figure 16.

D.2 MuSig2

MuSig2 [30] is the first Schnorr multi-signature scheme that incorporates key aggregation and achieves security under concurrent signing sessions. MuSig2 has a semi-interactive signing protocol that resembles FROST very closely. It also shares the ability to aggregate pre-signature shares with FROST3. Therefore,

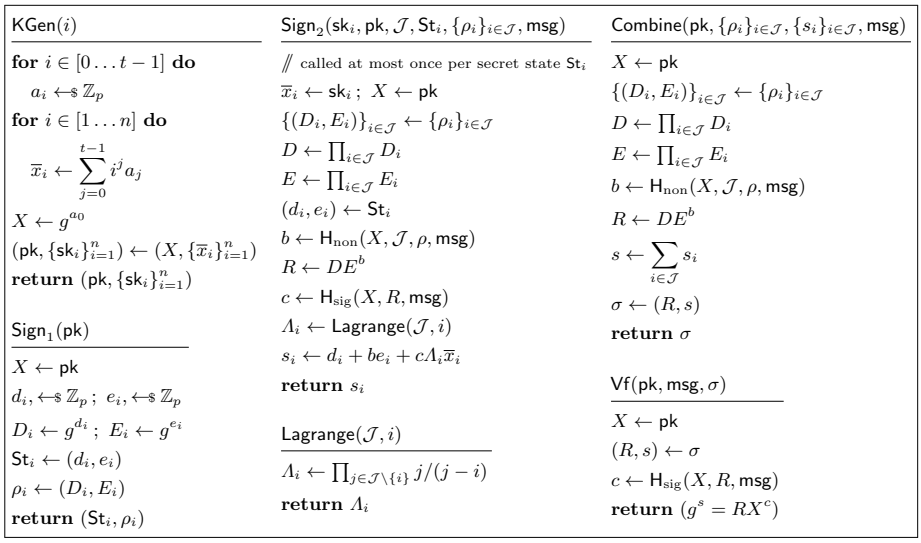

Fig. 16. Threshold Signature Scheme FROST3.

despite the slightly different context of n-out-of-n multi-signatures from t-out-of-n threshold signatures, we can obtain a similar result for MuSig2. That is to say, if defined malicious correctness and anamorphic indistinguishability for a n-out-of-n multi-sender anamorphic signatures (MSAS) similar to those in Section 5.2, we rule out a class of such an MSAS associated with MuSig2.

Theorem 8. *(Informal)* MuSig2 *is not anamorphic. In particular, there is no multi-sender anamorphic signature scheme associated with* MuSig2 *that is maliciously correct and has anamorphic indistinguishability.*

Proof. As MuSig2 shares the same forms of nonce and pre-signature shares ((D_i, E_i) and s_i) with FROST3, we can show that MuSig2 is not anamorphic by similarly applying the proof of Theorem 6. Note that the argument using anamorphic indistinguishability holds similarly due to the common partial verification. □

D.3 Sparkle and three-round Schnorr threshold signatures

Sparkle [14] is a practical threshold Schnorr signature scheme that allows one round of pre-processing and two online signing rounds.

Theorem 9. *The threshold scheme in Figure 17 is not anamorphic. In particular, there is no threshold-sender anamorphic signature scheme associated with the above threshold scheme as in Definition 12 that is maliciously correct and has anamorphic indistinguishability.*

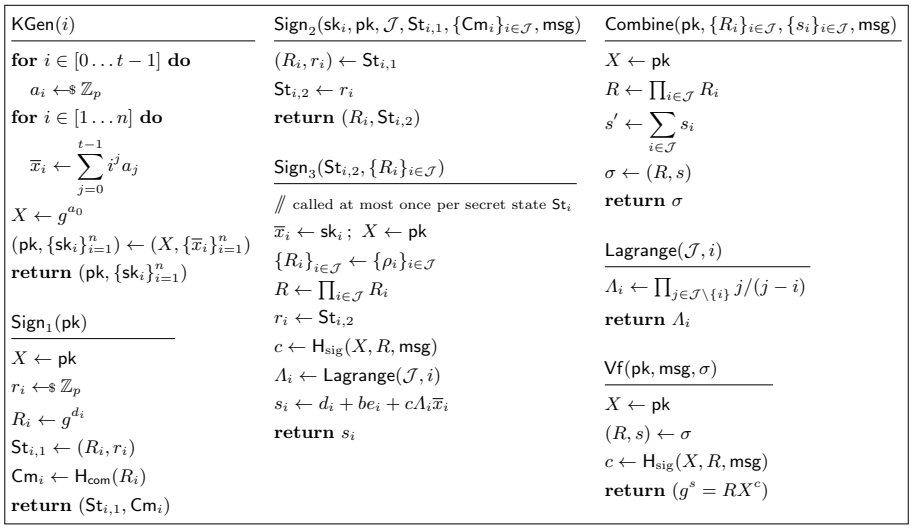

Fig. 17. Three-round Schnorr Threshold Signature Scheme Sparkle.

The proof of Theorem 9 follows the same approach as Theorem 6, leveraging two components: the signature-preserving property of Sparkle and an adversary that can randomize the final signature. Also, the argument using anamorphic indistinguishability holds due to partial verification of Sparkle. We refer the reader to that proof for details, and here, we present only the two core lemmas that substantiate these components.

Lemma 3. *The threshold scheme in Figure 17 is signature-preserving w.r.t. Schnorr signature scheme as in Definition 3.*

Proof of Lemma 3. Define the following algorithms TS.SkAgg and TS.StAgg

$$\mathsf{TS.SkAgg}(\{\mathsf{sk}_i\}_{i\in\mathcal{J}}) := \sum_{i\in\mathcal{J}} \Lambda_i \mathsf{sk}_i$$
$$\mathsf{TS.StAgg}(\{(R_i, r_i)\}_{i\in\mathcal{J}}) := \sum_{i\in\mathcal{J}} r_i,$$

where $\Lambda_i = \mathsf{Lagrange}(\mathcal{J}, i)$. Simple calculation shows that Equation (1) holds. □

Lemma 4. *Let* $\mathsf{H}_{\mathsf{com}}$ *be a collision-resistant hash function, and* $\mathsf{H}_{\mathsf{sig}}$ *be modeled as a random oracle. There exists an adversary* $\mathcal{A}$ *such that, for every security parameter* 1^λ*, and for all messages* msg *and anamorphic messages* amsg*, where the adversary participates in the signing process, the final signature* σ^{a} *is, with all but negligible probability, determined as a standard single-signer Schnorr signature, i.e.,* $\sigma^{\mathsf{a}} = r + cx$*, where* r *is uniformly distributed over* $\mathbb{Z}_q$*.*

Proof of Lemma 4. It suffices to consider the case $n = t = 2$. First, let us show that with all but a negligible probability, r_1 is fixed right after S_1 sends out Cm_1 during the first round. Indeed, we can assume that the output state $\mathsf{St}_{i,1}$ S_1 obtain when running $\mathsf{TSAS.Sign}_1(\mathsf{sk}_i, \mathsf{msg}, \mathsf{amsg}, \mathcal{J}, \mathsf{adk}^S)$ contains some r_1'. Let bad_1 and bad_2 be the events that $r_1' \notin \mathbb{Z}_q$ and that $r_1' \in \mathbb{Z}_q \wedge r_1' \neq r_1$, respectively. Note that if either occurs, we break the preimage or collision resistance of $\mathsf{H}_{\mathsf{com}}$. Therefore, $r_1' = r_1$ with all but a negligible probability.

By Lemma 3, the final signature σ^{a} is determined as a standard single-signer Schnorr signature $\sigma^{\mathsf{a}} = r + cx$, where $r = r_1 + r_2$ and x is the DL of the combined public key X.

As r_2 is uniform over $\mathbb{Z}_q$ and only known after r_1 is fixed, $(r_1 + r_2)$ is uniform over $\mathbb{Z}_q$. □

D.4 TRacoon

TRaccoon [38] is the first practical three-round threshold signature scheme from the standard lattice at EC'24. It is a thresholdized version of Raccoon [17], a lattice-based signature scheme by del Pino et al., a candidate for the additional NIST call for proposals [31].

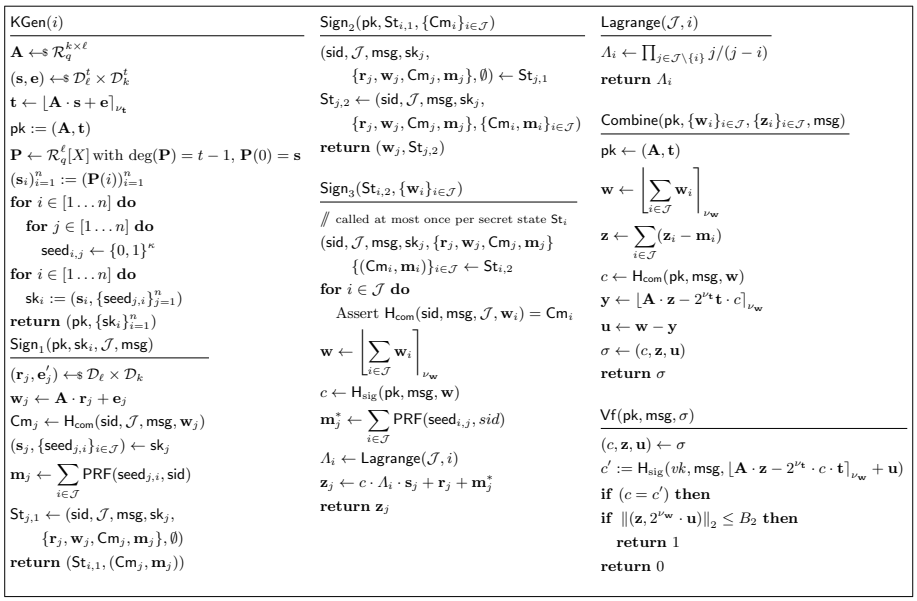

Fig. 18. TRaccoon Threshold Signature Scheme.

The following result holds for TRaccoon.

Theorem 10. TRaccoon *threshold scheme in Figure 18 is not anamorphic. In particular, there is no threshold-sender anamorphic signature scheme associated with* TRaccoon *as in Definition 12 that is maliciously correct and has anamorphic indistinguishability.*

The threshold signature scheme TRaccoon follows the folklore construction of a three-round threshold signature from Schnorr's signature scheme as in Appendix D.3. Therefore, the proof of Theorem 10 works similarly to Theorem 9, except for differences in the argument of anamorphic indistinguishability.

The key difference is that the third-round message $\mathbf{z}_i$ no longer satisfies partial verification w.r.t. the commitment $\mathbf{w}_i$, as TRaccoon has no partial public key. Instead, we base our argument on the verification of the combined (anamorphic) signature σ^{a}, which must have the same form as the threshold signature: $\sigma^{\mathsf{a}} = (c, \mathbf{z}, \mathbf{u})$. Once the honest signer outputs $\mathbf{w}_1$, the adversary can control the combined commitment by adding $\mathbf{w} = \mathbf{w}_1 + \mathbf{w}_2$, which is close to uniform from the honest signer's view. The full signature σ^{a} is then uniquely determined, assuming the underlying assumptions for security of TRaccoon from [38] with appropriate parameters.

We refer the reader to the proof of Theorem 9 for the remaining details and present below only the two core lemmas that substantiate the corresponding components.

Lemma 5. *The threshold scheme in Figure 18 is signature-preserving w.r.t.* Raccoon *signature scheme as in Definition 3.*

Proof of Lemma 5. Define the following algorithms TS.SkAgg and TS.StAgg

$$\mathsf{TS.SkAgg}(\{\mathsf{sk}_i\}_{i\in\mathcal{J}}) := \sum_{i\in\mathcal{J}} \Lambda_i \mathsf{sk}_i; \qquad \mathsf{TS.StAgg}(\{\mathsf{St}_{i,1}\}_{i\in\mathcal{J}}) := \sum_{i\in\mathcal{J}} \mathbf{r}_i,$$

where $\Lambda_i = \mathsf{Lagrange}(\mathcal{J}, i)$ and $\mathsf{St}_{i,1} = (\mathsf{sid}, \mathcal{J}, \mathsf{msg}, \mathsf{sk}_i, \{\mathbf{r}_i, \mathbf{w}_i, \mathsf{Cm}_i, \mathbf{m}_i\}, \emptyset)$. A simple calculation shows that Equation (1) holds. □

Lemma 6. *Let* $\mathsf{H}_{\mathsf{com}}$ *be a collision-resistant hash function, and* $\mathsf{H}_{\mathrm{sig}}$ *be modeled as a random oracle. There exists an adversary* $\mathcal{A}$ *such that, for every security parameter* 1^λ*, and for all messages* msg *and anamorphic messages* amsg*, where the adversary participates in the signing process, the final signature* σ^{a} *is, with all but negligible probability, determined as a single-signer* Raccoon *signature, i.e.,* $\sigma^{\mathsf{a}} = (c, \mathbf{z}, \mathbf{u}), \mathbf{z} = \mathbf{r} + c\mathbf{s}$*, where* $\mathbf{r}$ *is closely distributed to* $\mathcal{D}_\ell^t$*.*

The proof of Lemma 6 follows that of Lemma 4 but additionally leverages the convolution of discrete Gaussian distribution, due to the fact that $\mathbf{w}$ is a sum of discrete Gaussian vectors as explained in [38].

E Anamorphic Signatures under Strong Dictatorship

Theorem 4. *The construction in Figure 7 is correct and anamorphic.*

Proof. Our first observation is that the anamorphic message amsg' consists of a uniformly random string of $\ell_q + \lambda$ bits r and the ciphertext ct. To show anamorphism, we need to argue that ct is a uniformly distributed string $\{0,1\}^{\ell_m}$ in the view of the adversary. Our arguments follow by simply observing that the only way for the adversary to distinguish ct from a random string is in case it queries the random oracle H with the key K. There are two ways for the adversary to do it: either to try to hash group elements and succeed or to guess the anamorphic subgroup correctly and reconstruct the key K using the steps in $\mathsf{eTRAS.Combine}$. We will use a similar argument in the proof of CPA security, so we leave the formal steps for later and describe the intuition here. Note that since r_q is uniformly distributed over $\mathbb{Z}_q$ (this follows from picking r with enough additional bits), it follows that the key K is also uniformly distributed over $\mathbb{G}$. Therefore, for the first case, the adversary's chance is q_h/q to query the correct key K, where q_h is the upper bound on the number of hash queries of the adversary. The chances are $q_h/\binom{n}{\ell}$ in the second case. However, since we assume $\binom{n}{\ell} < 2^\lambda$, it follows that in both cases, the chances of the adversary querying K are negligible. Therefore, the ciphertext ct is indistinguishable from a uniformly random bit string for the adversary. We conclude that Figure 7 is anamorphic. Correctness follows by inspection. □

Theorem 5. *The construction in Figure 7 is IND-CPA secure according to Definition 11. More formally, for any polynomial-time adversary $\mathcal{A}$ making at most q_h queries to oracle H, the following holds:*

$$\mathsf{ExpeTR\text{-}CPA}^{\mathcal{A}}_{\mathsf{eTRAS}}(1^\lambda) \leq q_h \cdot (1/\mathsf{bino} + 1/q) + q_h^2/2^{\ell_m},$$

where $\mathsf{bino} = \binom{n-Q_{\mathsf{cr}}-Q_{\mathsf{ch}}}{\ell-Q_{\mathsf{cr}}-\lambda}$, $(n, \ell, Q_{\mathsf{cr}}, Q_{\mathsf{ch}})$ *are the parameters output by* $\mathcal{A}$ *in the* $\mathsf{ExpeTR\text{-}CPA}$ *experiment and assuming* $Q_{\mathsf{ch}} \cdot (\ell - Q_{\mathsf{cr}})/(n - Q_{\mathsf{cr}} - Q_{\mathsf{ch}}) < 1$ *and* q *is the order of the group* $\mathbb{G}$ *generated by* g.

Proof. The idea behind the proof is as follows. We first notice that since the adversary is allowed to generate all keys, it follows that IND-CPA security cannot follow from any scheme that would use those keys. In particular, the Diffie-Hellman keys. Instead, we want to argue that the adversary never asks the random oracle for the key K. The rationale here is that because the quorum $\mathcal{J}$ is unknown to the adversary, even if it can compute all decryption shares for anamorphic group members, they are hidden among the "fake" decryption shares of regular citizens. The adversary must guess the group, i.e., the correct $\mathcal{J}$, to get K and ask it to the random oracle. The adversary guesses the correct K with probability bino, which is negligible due to the experiment's conditions. While $\mathsf{H}(K)$ is not specified and there are no collisions in the hash function, we can

replace ct with a random bit string. We will show this proof using the game-based approach more formally below. Let us use $\mathbf{Adv}_{\mathrm{H}_i}(1^\lambda)$ to denote the probability:

$$\left|\Pr\left[\mathsf{ExpeTR\text{-}CPA}^{\mathcal{A}}_{\mathsf{TRAS}}(1^\lambda) = 1 \mid \mathrm{H}_i\right] - \frac{1}{2}\right|.$$

Hybrid $\mathbf{H}_0$: This is the original experiment $\mathsf{ExpeTR\text{-}CPA}^{\mathcal{A}}_{\mathsf{eTRAS}}(1^\lambda)$. We have $\mathbf{Adv}_{\mathrm{H}_0}(1^\lambda) = \mathbf{Adv}^{\mathcal{A}}_{\mathsf{eTR\text{-}CPA}}(1^\lambda)$

Hybrid $\mathbf{H}_1$: Similar to the previouse experiment, but we abort it in case there is a collision in the random oracle H.

Hybrid $\mathbf{H}_2$: Similar to the previous experiment, we abort if the adversary queries the key K to the oracle H.

Hybrid $\mathbf{H}_3$: We now replace the ciphertext ct by a uniformly random string from $\{0,1\}^{\ell_m}$.

Lemma 7. *We have* $|\mathbf{Adv}_{H_0}(1^\lambda) - \mathbf{Adv}_{H_1}(1^\lambda)| \leq q_h^2/2^{\ell_m}$.

Proof. The bound follows since the adversary $\mathcal{A}$ makes at most q_h queries to the random oracle, and collisions happen with probability at most $q_h^2/2^{\ell_m}$. □

Lemma 8. *We have* $|\mathbf{Adv}_{H_1}(1^\lambda) - \mathbf{Adv}_{H_2}(1^\lambda)| \leq q_h \cdot \left(\frac{1}{\binom{n-Q_{\mathsf{cr}}-Q_{\mathsf{ch}}}{\ell-Q_{\mathsf{cr}}-\lambda}} + 1/q\right)$.

Proof. We observe that the only information that the adversary does not learn is the $\mathcal{J}$. We also notice that the only way for $\mathcal{A}$ to compute the correct key K is to exactly guess $\mathcal{J}$, i.e., in case it uses a "fake" decryption share for one citizen that is outside of the group, it will receive an invalid key K. An alternative way to compute the correct key K is to pick a random element of the group $\mathbb{G}$. We will now show that the probability of guessing such an element directly is $1/q$ and focus on the first case later. We argue that the key K is uniformly distributed over $\mathbb{G}$ to show this probability. While the adversary can pick the public keys of citizens and influence the value $(\mathsf{adk}^{\mathcal{J}})^{\mathsf{sk}_{S,\mathsf{DH}}}$ the sender uses r_q to compute the key K, i.e., $K = \left((\mathsf{adk}^{\mathcal{J}})^{\mathsf{sk}_{S,\mathsf{DH}}}\right)^{r_q}$. Assuming r_q is uniformly samples over $\mathbb{Z}_q$, the uniformity of key K in $\mathbb{G}$ follows. Finally, notice that we pick r as a random bit string of size $\ell_q + \lambda$, and thanks to that, it follows that r_q are statistically close to a uniform distribution over $\mathbb{Z}_q$, which we assumed. Thus, we showed that using one query to the random oracle, the adversary can query the correct key K with probability $1/q$ if its strategy is to pick random elements from $\mathbb{G}$.

We now focus on the case where $\mathcal{A}$ randomly picks the subgroup and computes K based on its choice. Without access to Corr and Check oracles, there would be $\binom{n}{\ell}$ potential ℓ-size subgroups, so the changes would be $q_h/\binom{n}{\ell}$ to query the correct K. Access to those oracles increases $\mathcal{A}$ chances. In particular, the chances are maximized by first issuing all Corr and following with all Check queries.

After the adversary $\mathcal{A}$ issues all Corr queries, there are still $\ell - Q_{\mathsf{cr}}$ unknown anamorphic members and $n - Q_{\mathsf{cr}}$ candidates. So the probability of guessing a correct anamorphic member using the first Check query is $(\ell - Q_{\mathsf{cr}})/(n - Q_{\mathsf{cr}})$, which is maximized when the adversary guesses wrong all times $p_c = (\ell - Q_{\mathsf{cr}})/(n - Q_{\mathsf{cr}} - Q_{\mathsf{ch}})$ (i.e., $\frac{a}{b} < \frac{a}{b-1}$ and $\frac{a-1}{b} < \frac{a}{b}$ for all $a, b > 1$). Let us denote by Y_i the Bernoulli random value that the Check oracle outputs 1 on the i-th query. All those variables are dependent, and we cannot apply the Chernoff bound. However, we also know that for all $i \in [Q_{\mathsf{ch}}]$ $\Pr[Y_i = 1] \leq p_c$. Thus, let us consider independent Bernoulli random values X_i for which $\Pr[X_i = 1] = p_c$. It follows that $\Pr[Y_i = 1] \leq \Pr[X_i = 1]$. Consequently, the expected value $E[Y]$ for the sum Y of all variables is smaller than the corresponding expected value for the sum of all X values. Note here that $E[Y]$ corresponds to the expected number of times the Check will return 1, i.e., $\mathcal{A}$ guesses correctly a member of the group using the Check oracle. We will bound this number using the independent variables X_i and the Chernoff bound. The expected number is $\mu = E[X] = Q_{\mathsf{ch}} \cdot p_c$. Assuming $\mu < 1$ and using Theorem 7, we can bound the correct guesses of $\mathcal{A}$ by λ. We set $\delta = \lambda \cdot 1/\mu$ in the Chernoff bound. Note that then:

$$\Pr[X > (1 + \lambda \cdot 1/\mu) \cdot \mu] = \Pr[X > (\mu + \lambda)] < 2^{-\lambda \cdot 1/\mu \cdot \mu} = 2^{-\lambda}$$

and since we assumed $\mu < 1$ and X is the number of times Check will return 1, it follows that $\Pr[X > \lambda] < 2^{-\lambda}$.It follows that after all the Corr and Check queries, there are still $\binom{n-Q_{\mathsf{cr}}-Q_{\mathsf{ch}}}{\ell-Q_{\mathsf{cr}}-\lambda}$ potential subsets.

So the probability that the adversary guesses the correct subset to get key K is upper bounded by $q_h \cdot \left(\frac{1}{\binom{n-Q_{\mathsf{cr}}-Q_{\mathsf{ch}}}{\ell-Q_{\mathsf{cr}}-\lambda}} + 1/q \right)$. The claim follows. □

Lemma 9. *We have* $\mathbf{Adv}_{H_2}(1^\lambda) = \mathbf{Adv}_{H_3}(1^\lambda)$.

Proof. Because of the change in H_2, we observe that the adversary never queries the random oracle for K. Moreover, we know that there are no collisions in H. Therefore, the value $\mathsf{H}(K)$ is not specified, and the adversary cannot notice the change of ct to a uniformly random string. □

Lemma 10. *The advantage of adversary* $\mathcal{A}$ *in* H_3 *is* 0*, i.e.,* $\mathbf{Adv}_{H_3}(1^\lambda) = 0$.

Proof. Since the only element in the anamorphic signature that depends on the anamorphic message amsg is the ciphertext $\mathsf{ct} = \mathsf{H}(K) \otimes \mathsf{amsg}$ and with the changes in H_3, we replaced ct with a uniformly random string. It follows that the only strategy of the adversary is to guess bit b. Therefore, the advantage of $\mathcal{A}$ is 0. □

□