Sharing a Secret Anamorphically: Secret Shares Disguised as Signatures

Gennaro Avitabile[1], Vincenzo Botta[2], and Daniele Friolo[3]

[1] University of Surrey, Surrey Centre for Cyber Security, Guildford, UK
avitabilegenn@gmail.com
[2] Independent Researcher
botta.vin@gmail.com
[3] Roma Tre University, Rome, Italy
daniele.friolo@gmail.com

Abstract. Anamorphic Encryption (Persiano, Phan and Yung, Eurocrypt '22) allows private communication in a challenging setting where encryption is severely controlled by a central authority (henceforth the dictator) who can demand the users to surrender their secret keys. Anamorphic Signatures (AS) (Kutylowski, Persiano, Phan, Yung, Zawada, Crypto '23) face the even more restrictive world where only authentication is allowed but users still want to send secret messages despite the repressive control of the dictator holding their signing key. Several constructions have been proposed so far, but they all come with some limitations.
We propose a new flexible setting where digital signatures are used to covertly embed secret shares that t out of N designated receivers can combine to recover a covert secret. We formalize this new primitive called Anamorphic Secret Sharing Signatures (ASSS) together with new robustness, forward/backward secrecy, and private unforgeability notions and we give a construction for Schnorr-like signatures overcoming the limitations of previous constructions. ASSS additionally imply (regular single-receiver) AS, but two signatures (instead of one) are needed to recover the covert message.

1 Introduction

Anamorphic Encryption (AE) [43] enables private communication in environments where encryption is tightly regulated by a central authority. AE allows a sender to covertly transmit a confidential message to a receiver without raising suspicion, even if the authority has the receiver's secret decryption key. This concept is particularly relevant in dictatorships, where encryption may remain legally permitted to protect against external threats, yet individuals can be compelled to hand over their private keys. In [31] an even stricter scenario is considered where the dictator forbids encryption altogether and only allows authentication via the use of digital signatures. This stronger scenario automatically relieves the dictator from the problem of covert communication via AE, but the possibility of exchanging secret messages in spite of the repressive control of the dictator is still opened up by the presence of anamorphic signatures.

Anamorphic Signatures (AS) operate under two distinct modes: standard and anamorphic. In the standard mode, AS behave like any typical signature scheme. In anamorphic mode, however, key generation produces a public key pk along with two secret keys: a conventional signing key sk and a "double key" dk. Alice shares dk with Bob through a private one-time channel, while using pk publicly. If compelled to reveal her secret key, Alice can safely disclose only sk. Alice, who knows dk and sk, can craft a signature that both authenticates a message m under her public key and simultaneously encodes a hidden message $\hat{m}$, which only Bob can extract using dk. The core security goal is that the anamorphic mode is indistinguishable from the standard mode, even when the dictator knows sk. Schemes that additionally allow the double key to be sampled independently of the conventional keys are called Anamorphic Extensions [5]. As noted in [43], designing entirely new schemes that are compatible with anamorphic communication offers limited benefit. A dictator could simply mandate any scheme they prefer, potentially to suppress anamorphic messaging. Consequently, several works proved that existing encryption schemes are inherently anamorphic [43, 5, 14, 32, 44]. Many of these constructions are not only efficient but also have an exponentially large anamorphic message space. Analogously, recent works [31, 4, 18] have shown that several existing signature schemes are anamorphic, unfortunately such schemes come with some limitations that we discuss next.

On unique signatures. The most effective way for a dictator to suppress anamorphic communication is through unique signature schemes. In such schemes, each message can only produce a single valid signature under a given key, eliminating the possibility of embedding covert information. However, practical adoption may be hindered by compatibility requirements with existing systems (e.g., Bitcoin). This trade-off often dictates the choice of cryptographic primitives. For instance, Schnorr multi-signatures [29, 40, 2], despite notable drawbacks compared to BLS [12], have gained significant traction precisely because of their compatibility with standard Schnorr signatures.

Constructing AS. In [31] various techniques for AS are presented. Among those, rejection sampling is a technique for getting an AS extension on top of any signature scheme whose signatures have high min-entropy. The double key is simply a PRF key K and the sender samples a new signature σ until $\mathsf{PRF}(K, \sigma)$ is equal to the desired message. However, if the signer wants to embed a k-bit message, 2^k samples are necessary on average and thus, to keep signing in polynomial time, the size of the message must be logarithmic.

It is instead more interesting to look for AS with an exponentially large anamorphic message space. In [31], a sufficient condition for anamorphism of signature schemes called randomness recoverability is given. That is, a signature scheme is randomness recoverable if it allows, with sufficient knowledge (e.g., with the secret key), to recover from the signature (a part of) the random coins used during the signing process. Given a randomness-recoverable signature scheme it is straightforward to encrypt an anamorphic message $\hat{m}$. The signer en-

crypts $\hat{m}$ using a symmetric encryption scheme with pseudo-random ciphertexts and uses the obtained ciphertext as the random coins of the signing process. The double key simply consists of the secret key of the symmetric encryption scheme and all the information that allows to recover the randomness. This technique gives an exponential anamorphic message space.

Based on this insight, we can then divide signature schemes in different categories depending on what information is necessary to recover the signing randomness:

Recovering the randomness requires no secret: Clearly randomness-recoverable signatures are schemes directly including uniformly random bit-strings in $\{0,1\}^{O(\lambda)}$ in the signature itself. For example, in [31] the Boneh-Boyen [9], the RSA-PSS signatures [7], and the Naor-Yung transform [39] applied to the Lamport tagging scheme [33] - three schemes with such a property - are shown to be anamorphic. There are many other, widely known, schemes with such a property (e.g., [10, 28, 8]). Although the techniques shown in [31] to embed anamorphic messages in such signature schemes are elegant and sophisticated, the schemes they are built on top quite clearly embed a steganographic channel and are easily forbidden by a dictator eager to prevent covert communication.

Recovering the randomness requires the secret key: Other constructions are randomness recoverable only given the signing secret key. This constraint has two disadvantages: (i) the sender has to trust the anamorphic receivers, as they can sign messages on her behalf using the signing secret key, even though the sender did not ever send an anamorphic message and (ii) every time the sender has to update her public key, the demanding secure exchange of the double key (e.g., in person) must be repeated to enable anamorphic communication. For example, in [31] the ElGamal [24, 45] and the Schnorr [46] signatures are shown to be randomness recoverable, but only with the help of the signing key. For example, the Schnorr scheme includes a uniformly random group element a in the signature, but it cannot be directly used to embed a pseudo-random ciphertext since the signer needs to know the discrete log of a to finalize the signature.

Currently, apart from schemes that embed uniformly random bit-strings directly into the signature, there are basically no AS constructions, for widely used schemes, providing an exponentially large anamorphic message space while allowing anamorphic channel setup without revealing the secret key to the receiver. The recent work of Deo and Libert [18] proposes a remarkable bandwidth-efficient construction for the RSA-based Guillou-Quisquater scheme. However, such scheme is not very used in practice. Instead, the concretely more efficient dlog-based protocols are now widely deployed and used across numerous applications. The same authors [18] further propose a dlog-based high-rate anamorphic signature over the Okamoto-Schnorr signature scheme [42]. However, their solution needs the users to define some public parameters together with a trapdoor, which is used to create anamorphic signatures. Realistically, a dictator would choose such public parameters (as a system manager could do according to [42])

and force the users to use them; thus making the technique of [18] unusable. More details are given in Sec. 2.

In the following section, we present a new setting that introduces additional features and overcomes the above limitations on a class of widely used schemes, including the Schnorr signature, albeit at the cost of requiring two signatures (rather than one) to recover the anamorphic message.

1.1 Anamorphic Secret Sharing Signatures

We study the scenario of a dealer D who privately sends double keys $\mathsf{dk}_1, \ldots, \mathsf{dk}_N$, sampled *independently from the signing key*, to N different receivers $R_1, \ldots, R_N$, and anamorphically encodes secret shares of an anamorphic secret $\hat{s}$ in digital signatures $\sigma_1, \ldots, \sigma_n$ authenticating (arbitrary) regular messages $m_1, \ldots, m_n$.

Using the double key dk_i and the signature σ_i, R_i can extract a share from the signature. Less than t shares computationally hide $\hat{s}$, while t or more shares allow to recover $\hat{s}$. We call this primitive Anamorphic Secret Sharing Signature (ASSS). As in AS, we require that regular signatures are indistinguishable from signatures anamorphically encoding shares.

ASSS enrich the functionality of AS ensuring that the anamorphic message can only be reconstructed by a quorum of receivers. In a dictatorship, anamorphic communication can be seen as a last resort to communicate and coordinate within groups despite the repressive regime. For example, the dealer could be a relevant party (e.g., another authority close to the dictator) who wants to send signatures[4] to whistleblow secrets to a series of parties (e.g, the press) with the guarantee that a minimum number of parties will learn the leaked information about the dictator.

In such a scenario, we require ASSS to satisfy some additional properties w.r.t. AS. First, the ASSS should have a form of *robustness*. Indeed, we support the even more repressive setting where only the dealer is allowed to generate signatures and thus reconstruction must happen in person[5]. Therefore, it is crucial that the receivers are aware of whether a signature encodes a share or not, as just *attempting* a reconstruction that is going to fail might be wasteful or even dangerous. This notion of robustness is different from the one defined in [5] (see Sec. 1.3 for more details). Second, we require some form of *forward/backward secrecy*, meaning that a single demanding[6] setup phase of the double key is sufficient to share an unbounded number of secrets; and a successful reconstruction phase does not impact on the privacy of the previously/subsequently shared anamorphic messages. We call such property *adaptive secrecy*, as it holds even in the presence of adaptive corruptions. Lastly, we require a property we call

[4] These signatures might be involved in her normal activity and even be posted on a public bulletin board.

[5] If everyone is allowed to generate signatures, the parties could just send each other the shares using anamorphic signatures. Indeed, in the next paragraph we show that ASSS imply a form of AS.

[6] The dealer may be a relevant party who is closely watched by the dictator.

weak recipient unforgeability meaning that the signature remains unforgeable by the receiver (e.g., the signing key is not leaked to the receivers) until a reconstruction is performed[7]. Weak recipient unforgeability becomes very handy when the creation of the double key is independent from the creation of the signing key. Indeed, after having shared a secret, the dealer can ask for a new signing key[8], avoiding impersonation attacks without losing her ability to communicate anamorphically in the future.

ASSS with one receiver. We remark that ASSS are useful even for one-to-one communication. Indeed, the dealer can make a 2-out-of-2 secret sharing giving both dk_1 and dk_2 to the same receiver. In this way, the receiver can recover the anamorphic message after having received 2 signatures. If the ASSS has good bandwidth, this solution is preferable to a regular AS with small message space (e.g., via rejection sampling), as the latter would require sending much more signatures to convey the same amount of information.

This solution is similar to the notion of anamorphic signature multi-message extensions recently proposed in [4]. However, their constructions do not overcome the limitation highlighted in Sec. 1. In fact, they either require users to generate public parameters with a trapdoor that could normally be set up by the dictator precisely so that users do not possess such a trapdoor, or they rely on schemes that directly embed a random bit-string in $\{0,1\}^{O(\lambda)}$ into the signature itself (see Sec. 2 for more details).

1.2 Our Contribution

We construct a class of ASSS schemes based on the Fiat-Shamir transform of a class of Σ-protocols called pre-image Σ-protocols [36, 17, 16], which include the widely used Schnorr identification scheme.

Our scheme has the following features:

1. The generation of the double keys is independent of the generation of the signing key. Therefore, the sender can generate a signing key with the regular key generation procedure and can re-use the same double keys even if he needs to update his signing key.
2. The anamorphic message space is exponentially large.
3. It enjoys the robustness, adaptive secrecy, and weak recipient unforgeability notions explained in Sec. 1.1.

We remark that the previous construction of AS for Schnorr [31] does not enjoy weak recipient unforgeability, as the double key simply consists of the signing secret key. In this sense, before our results, a dictator could have felt comfortable with allowing Schnorr signatures since the only known way to make them

[7] Recipient unforgeability was first introduced in [18] for AS. It states that the signature scheme remains unforgeable by the receiver, no matter how many anamorphic signatures they received.

[8] A key revocation procedure should always be made available since the risk of compromising secret keys does not disappear if a dictatorship takes place.

randomness recoverable, and thus to encrypt big anamorphic messages, was to give the secret key to the receiver even if no anamorphic message would ever be encrypted in the future. This is a big disincentive to anamorphic communication if the ability to sign messages is valuable, e.g. it allows to spend money on the dictatorship's digital currency.

In our setting, instead, even if the signing key gets leaked once the anamorphic communication takes place, the sender can simply perform a key revocation procedure and get a new signing key. This is without the need to perform a fresh (demanding and possibly risky) exchange of a double key should further anamorphic communications take place.

1.3 Technical Overview

Let us consider the Schnorr Σ-Protocol for proving knowledge of a discrete log. The works of Cramer [16], Cramer and Damgard [17], and Maurer [36] showed that a protocol (called Pre-Image Σ-protocol) for proving knowledge of a pre-image of a group homomorphism generalizes a large number of Σ-protocols in the literature including the Schnorr's protocol [46] and the Guillou-Quisquater protocol [26]. For simplicity we will now focus on the Schnorr Σ-Protocol, but our techniques apply to the more general Pre-Image Σ-protocol.

In the Schnorr Σ-protocol, the common input consists of a group $\mathbb{G}$ of a prime order p, a generator g and $x \in \mathbb{G}$. The prover knows w such that $x = g^w$. The protocol proceeds in three rounds: (i) the prover sends to the verifier a value $a = g^r$ where r is sampled at random, (ii) the verifier sends to the prover a uniformly random challenge c, (iii) the prover replies with $z = r + cw$. Finally, the verifier accepts iff $g^z = a \cdot x^c$. In [31] it is observed that Schnorr (as many other Σ-Protocols) is randomness recoverable if the witness w is known. Indeed, the randomness r used to produce first-round message a can be obtained from z as $r = z - cw$. Such strategy works identically if the Σ-protocol is turned into a signature using the Fiat-Shamir transform.

We instead exploit another property of Σ-protocols to extract the randomness in a different way without being directly provided with the witness. That is, Σ-protocols satisfy a property called special soundness stating that from any two accepting transcripts (a_0, c_0, z_0) and (a_1, c_1, z_1) for a statement x such that $a_0 = a_1$ and $c_0 \neq c_1$, one can extract the witness w. For example, the extractor of Schnorr computes the witness as $w = (z_0 - z_1)(c_0 - c_1)^{-1}$.

A basic idea for a 2-out-of-2 anamorphic secret sharing protocol involves sending to two receivers two accepting transcripts sharing the same a so that they can put together the transcripts to recover w and hence r. However, aside being insecure, producing two transcripts/signatures with the same a makes anamorphic communication easily detectable by the dictator! Furthermore, this approach does not generalize to arbitrary values of t and N.

Adding anamorphic security. To prevent the dictator from detecting anamorphic communication we exploit the homomorphic properties of Schnorr i.e., that $g^{r_1} \cdot g^{r_2} = g^{r_1 + r_2}$. A 2-out-of-2 anamorphic secret sharing protocol consists in

generating two PRF keys K_0 and K_1 as double keys and to share a synchronized counter ctr with the receivers. To encode an anamorphic share of a secret $\hat{s}$ for the i-th receiver with $i \in \{0,1\}$, the dealer produces as a first round message $a_i = g^{r_i}$ where $r_i = \hat{s} + \mathsf{PRF}(K_i, ctr)$. The receivers can both compute the same first round message $a = a_i \cdot g^{-\mathsf{PRF}(K_i, ctr)}$ and consequently adapt their z_i as $z_i' = z_i - \mathsf{PRF}(K_i, ctr)$. They can then run the extractor algorithm on the transcripts (a, c_0, z_0') and (a, c_1, z_1') to get w and consequently $r_i - \mathsf{PRF}(K_i, ctr) = \hat{s}$.

Generalizing to arbitrary thresholds. One can easily extend the above template to N receivers, but this comes with the inconvenience that the reconstruction threshold is fixed to $t = 2$. Indeed, any two receivers can get the same first-round message and come together to run the extractor. Our approach to overcome this limitation is having the dealer producing the first-round messages a_i with $i \in [N]$ in a way that t receivers must cooperate to unblind their a_i and get the same a. After this unblinding procedure, any two transcripts can be used to run the Σ-protocol's extractor and recover the anamorphic message $\hat{s}$. To achieve this goal we use distributed pseudorandom functions (DPRF) [37, 11, 38, 34]. In DPRF, a secret key SK_0 is secret shared among N parties generating key shares $SK_1, \ldots, SK_n$ so that the party i can locally compute a partial evaluation $\mathsf{DPRF.PEval}(SK_i, X)$ on some input X. Given t partial evaluations it is possible to reconstruct $\mathsf{DPRF.Eval}(SK_0, X)$, i.e. the evaluation of the DPRF under the initial secret key SK_0. Roughly, the security of the DPRF mandates that, under possibly adaptive corruptions, less than t partial evaluations computationally leak nothing about the evaluation of the DPRF using the initial secret key SK_0.

The modified protocol works as follows, the dealer generates a DPRF secret key SK_0 and shares it into $SK_1, \ldots, SK_n$. The double key of the receiver $i \in [N]$ is $\mathsf{dk}_i = SK_i$. When generating a share for the message $\hat{s}$ for the receiver i, the dealer produces $a_i = g^{r_i}$ with $r_i = \hat{s} + \mathsf{DPRF.Eval}(SK_0, ctr||i)$. During the reconstruction phase, each party i provides its transcript along with the set of their partial evaluations $\{\mathsf{DPRF.PEval}(SK_i, ctr||j)\}_{j \in [N]}$. From t of these sets it is possible to reconstruct $\mathsf{DPRF.Eval}(SK_0, ctr||j)$ and $\mathsf{DPRF.Eval}(SK_0, ctr||k)$ for some $j \neq k \in [N]$ and thus obtain two transcripts to run the extraction procedure and recover $\hat{s}$. Notice that the dictator does not catch the dealer because every party gets a different a_i and subsequent interactions with the same receiver j also produce a random looking a_j since the counter ctr gets incremented at each interaction. Furthermore, adaptive secrecy is obtained because the DPRF's security guarantees that an evaluation on point X tells nothing about the evaluation on another point $Y \neq X$. Unlike the constructions in [31], our construction requires parties to keep a small state, however it comes with additional properties and features.

Adding robustness. In [5] and [27] the notion of robustness for anamorphic encryption and signatures respectively is proposed. In a nutshell, it requires that ciphertexts/signatures generated using the regular scheme do not accidentally decrypt to an incorrect anamorphic message. This property is readily satisfied by our scheme since unblinding two regular first-round messages and getting the

same value only happens with negligible probability. However, we ask for a somewhat stricter requirement that is parties should not even *attempt* to reconstruct a secret if no secret was encoded by the dealer. This is motivated by the fact that the dealer may need to produce/distribute signatures even when no secret to be shared is available. Since the reconstruction should happen via a secure channel (e.g., in person) it might be too costly or even dangerous for the receivers to come together only to discover no secret was encoded in the signatures. We achieve this property by adding a key K of a 1-bit output PRF in the double keys. The dealer rejection samples, over a counter u, the first-round messages $a_i = g^{r_i}$ with $r_i = \hat{s} + \mathsf{DPRF}(SK_0, ctr||i||u)$ until $\mathsf{PRF}(K, a_i) = 0$ if a secret is encoded; she rejection samples over fresh random coins r_i until $\mathsf{PRF}(K, a_i) = 1$ otherwise. The dealer aborts if more than λ attempts are performed, which will happen only with negligible probability.

The receivers simply evaluate $\mathsf{PRF}(K, a)$ to check whether a signature encodes a share or not. During the reconstruction phase, each party now provides $\{\mathsf{DPRF.PEval}(SK_i, ctr||j||u)\}_{j\in[N],u\in[\lambda]}$. The reconstructing party now looks for values of u_j and u_k such that the unblinding of a_j and a_k gives the same a. The expected number of trials for encoding a share is 2, while it is 4 for reconstructing a secret.

2 Related Work

Anamorphic Encryption (AE) is conceptually related to, though distinct from, several previously studied notions. A detailed comparison is provided in [43]. Moreover, [47, 13] established a strong connection between AE and Algorithm Substitution Attacks (ASA) [6], enabling insights from one domain to inform and reinterpret results in the other. The concept has been extended in [32, 14] to address stronger privacy guarantees, particularly in scenarios where adversaries may have access to the double key. The notion of robust Anamorphic Encryption (AE) was introduced in [5] and later generalized to the sender-AE setting in [47]. A further enhancement of robustness, termed anamorphic unforgeability, was proposed in [21]. Another line of work studies the structural limits of anamorphic bandwidth of AE [15, 21, 13, 3]. The concept of anamorphic broadcast mode along with a construction was introduced in [20].

The concept of anamorphic signatures was introduced in [31] along with several constructions. In [27] some revised definition for the security of anamorphic signatures are proposed. In [4], besides other contributions for AE, the concept of anamorphic signature extensions is proposed. In anamorphic signature extensions a constant number of anamorphic signatures is used, together with the double key, to recover an anamorphic message. In particular, they provide clever asymmetric constructions (i.e., where the double key is asymmetric like in public key encryption) for Water's signatures [48] and a construction for BBS signatures [10]. The first one requires the user to fix their own parameters for the Water's hash, which the authors acknowledge to be a strong assumption in a dictatorship. The second one exploits the fact that BBS signatures directly em-

bed a random bit-string in $\mathbb{Z}_p$ in the signature itself. The recent work of Deo and Libert [18] proposes several anamorphic signature constructions with recipient unforgeability. One construction is based on the generalized Okamoto-Schnorr signature scheme [42] which is derived by applying the Fiat-Shamir transform to the following Σ-Protocol. The prover proves knowledge of the secret key $(x_1, \ldots, x_n)$ such as $X = \prod_{j=1}^{n} g_i^{x_i}$, where X is the corresponding public key. To make this scheme anamorphic [18] assumes the public key to be structured as $(X, g_1, \ldots, g_n)$ allowing the user to generate g_i with a trapdoor w_i s.t. $g_i = g^{w_i}$, even though this trapdoor information is not needed for regular signing. Indeed, as also stated in the original paper by Okamoto [42], the values $g_1, \ldots, g_n$ can be generated by a third party and shared by all users. Therefore, it is reasonable to think that the dictator could simply generate $g_1, \ldots, g_n$ s.t. the users do not have the trapdoor information. A second construction is based on RSA-based Guillou-Quisquater scheme however, such scheme is not very used in practice. Finally, a modification of the Lyubashevsky scheme [35] is also proposed, but it crucially requires to use much bigger system parameters than the commonly used ones.

3 Preliminaries

Given a pair of interactive Turing machines, P and V, we denote by $\mathsf{trans}(\langle \mathsf{P}(w), \mathsf{V} \rangle(x))$ the transcript of the interaction (i.e., the set of exchanged messages) between the machines P and V with w being P's private input and x being common input. For two machines M, A, we let $\mathsf{M}^{\mathsf{A}}(x)$ denote the output of machine M on input x and given oracle access to A. We denote by $x \leftarrow\!\!\$\ \mathcal{X}$ the process of uniformly choosing an element x in a set $\mathcal{X}$. When considering a distribution D w.r.t a set $\mathcal{X}$, we denote $x \leftarrow\!\!\$\ D$ the process of sampling from the distribution D. We implicitly assume that all the distributions we consider are efficiently sampleable. Given two distributions D_1 and D_2, we use $D_1 \equiv D_2$ to denote that D_1 and D_2 are identically distributed, $D_1 \approx_s D_2$ to denote that D_1 is statistically indistinguishable from D_2, and $D_1 \approx_c D_2$ to denote computational indistinguishability of the two distributions.

With a little abuse of notation, we assume, when needed, that a set $\mathcal{X}$ is ordered and we write $\mathcal{X}[i]$ to indicate the i-th element of $\mathcal{X}$.

When writing $b \leftarrow \mathsf{A}(x)$, we mean that b is the output of the probabilistic polynomial-time (PPT) algorithm A on input x. Furthermore, we denote with $\mathsf{A}(x; r)$ an execution of the PPT algorithm A with fixed random coins r. Any element with position (i, j) in a matrix can be accessed with $\mathsf{M}[i][j]$. The number of rows of M can be retrieved with $\mathsf{M}.\mathsf{length}$. An entire row of elements can be added to a matrix with $\mathsf{M}.\mathsf{append}(el_1, \ldots, el_n)$ with n the number of columns in the matrix. When, during a nested cycle, the command break is invoked, i.e. **for** $i \in [N]$ {**for** $j \in [N]${ break}}, it automatically breaks *all* the nested cycles.

3.1 Signatures

A signature scheme is a tuple of poly-time algorithms $\mathsf{SIG} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ with message space $\mathcal{M}$, described as follows:

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$**:** On input the security parameter, outputs a verification key/signing key pair $(\mathsf{vk}, \mathsf{sk})$.
$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$**:** On input a signing key sk and a message $m \in \mathcal{M}$, outputs a signature σ.
$b \leftarrow \mathsf{Verify}(\mathsf{pk}, m, \sigma)$**:** On input a public key pk and a message $m \in \mathcal{M}$, outputs 1 if the signature verifies, and 0 otherwise.

Definition 1 (Correctness). SIG *is correct if, for all* $m \in \mathcal{M}$, $\Pr[\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1]$, *where* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$.

Definition 2 (EUF-CMA). SIG *is existentially unforgeable under chosen message attacks if, for all valid PPT adversaries* A,

$$\Pr\left[\mathsf{V}(\mathsf{pk}, m, \sigma) = 1 \ \wedge m \notin \mathcal{Q}_{\mathsf{Sign}}(\mathsf{sk}, \cdot) \,\middle|\, \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda); \\ (m, \sigma) \leftarrow \mathsf{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(\mathsf{vk})\end{array}\right] \leq \mathsf{negl}(\lambda).$$

where $\mathcal{Q}_{\mathsf{Sign}}$ *is the set of all the queries made by the adversary to the* $\mathsf{Sign}(\mathsf{sk}, \cdot)$ *oracle.*

3.2 Σ-protocols

A Σ-protocol is a *3-round* public-coin protocol consisting of a tuple of PPT algorithms $(\mathsf{P}_0, \mathsf{P}_1, \mathsf{V})$. It is defined w.r.t. an NP language $\mathcal{L}$ with a poly-time relation $\mathcal{R}_\mathcal{L}$, a challenge space C_Σ, and a randomness space R_Σ. The algorithms work as follows:

- $a \leftarrow \mathsf{P}_0(x, w; r)$ on input statement x, witness w, randomness $r \in \mathsf{R}_\Sigma$, outputs a first-round message a.
- $z \leftarrow \mathsf{P}_1(x, w, a, c, r)$ on input statement x, witness w, first-round message a, randomness r used in the first round, and challenge $c \in \mathsf{C}_\Sigma$, outputs the third-round message z.
- $1/0 \leftarrow \mathsf{V}(x, a, c, z) = 1$, on input statement x, first-round message a, challenge $c \in \mathsf{C}_\Sigma$, outputs 1 to accept and 0 to reject.

A transcript (a, c, z) w.r.t. a statement x is called *accepting* if $\mathsf{V}(x, a, c, z) = 1$.

Throughout the paper, we refer to $\langle \mathsf{P}(\cdot), \mathsf{V} \rangle(\cdot)$ as an execution of a Σ-protocol Σ between a prover P and a verifier V, where a is computed using P_0 and sent to V, c is sampled at random from C_Σ by V and sent to P and finally z is computed by P using P_1 and then sent to V. In this case, the output of trans will be of the form (a, c, z).

Definition 3 (Σ-protocol). *A Σ-protocol is required to have the following properties:*

Completeness: *For all x and w such that $(x, w) \in \mathcal{R}_{\mathcal{L}}$ it holds that:*

$$\Pr\left[\mathsf{V}(x, a, c, z) = 1 \;\middle|\; \begin{array}{c} r \leftarrow\!\!\$\ \mathsf{R}_{\Sigma}; c \leftarrow\!\!\$\ \mathsf{C}_{\Sigma}; \\ a \leftarrow \mathsf{P}_0(x, w; r); \\ z \leftarrow \mathsf{P}_1(x, w, a, c, r) \end{array}\right] = 1.$$

Special Soundness: *There exists a PPT extractor* Ext*, such that on input x and 2 accepting transcripts $(a, c_0, z_0), (a, c_1, z_1)$ for x, where $c_0 \neq c_1$, it holds that:*

$$\Pr[(x, w) \in \mathcal{R}_{\mathcal{L}} | w \leftarrow \mathsf{Ext}(x, a, c_0, z_0, c_1, z_1)] = 1.$$

Special HVZK (SHVZK): *There exists a PPT algorithm* Sim_{Σ} *such that for all $c \in \mathsf{C}_{\Sigma}$, and all $(x, w) \in \mathcal{R}_{\mathcal{L}}$:*

$$\Big\{(a, c, z) \mid r \leftarrow\!\!\$\ \mathsf{R}_{\Sigma}, a \leftarrow \mathsf{P}_0(x, w; r), z \leftarrow \mathsf{P}_1(x, w, a, c, r)\Big\} \approx_c$$
$$\Big\{(a, c, z) \mid (a, z) \leftarrow \mathsf{Sim}_{\Sigma}(x, c)\Big\}.$$

The work of [1] establishes the minimal requirement for the security of the Fiat-Shamir [25] transform when applied to three-message public-coin identification schemes. Here, the statement x is the prover's identity (e.g., a public key) and w the prover's secret key. In security against passive attacks, an impersonator tries to convince a verifier without knowing the secret key. The impersonator is further allowed to obtain a polynomial number of transcripts of honest execution of the protocol before trying to impersonate the prover. This attack is considered *passive* since the impersonator can obtain only honest transcripts. If a PPT impersonator wins this game with negligible probability, then the protocol is secure against passive attacks. We defer to [1] for the formal definition.

Theorem 1 ([1]). *If a 3-round protocol* (P, V) *is secure against* passive attacks*, then the signature scheme obtained by applying the Fiat-Shamir heuristics is EUF-CMA-secure (Def.2) in the random oracle model (ROM).*

It is known that Σ-protocols for hard relations (i.e., relations for which given the statement it is hard to find the witness) are secure against passive attacks.

3.3 Pre-Image Σ-Protocol

Let $(\mathcal{G}, *)$ and $(\mathcal{H}, \otimes)$ be two groups of super-polynomial size with efficiently computable operations, and let $f : \mathcal{G} \to \mathcal{H}$ be a one-way homomorphism, i.e., $f(x * y) = f(x) \otimes f(y)$. The pre-image Σ-protocol for the relation $\mathcal{R} = \{(x, w) : x = f(w)\}$ is defined as follows:

- **Common input:** Description of groups $\mathcal{G}$, $\mathcal{H}$, and element $x \in \mathcal{H}$.
- **Prover's private input:** $w \in \mathcal{G}$ such that $x = f(w)$.

The algorithms work as follows:

1. $a \leftarrow \mathsf{P}_0(x, w; k)$**:** on input $(x, w) \in \mathcal{R}$ and random coins $k \in \mathcal{G}$, compute $a \leftarrow f(k)$, and output a.
2. $z \leftarrow \mathsf{P}_1(x, w, c, k)$**:** on input $(x, w) \in \mathcal{R}$, k, and challenge $c \in \mathsf{C}_\Sigma \subseteq \mathbb{N}$, compute and output $z = k * w^c$.
3. $1/0 \leftarrow \mathsf{V}(x, a, c, z)$**:** on input (x, a, c, z), output 1 if and only if $f(z) = a \otimes x^c$.

In [36] it is proven that pre-image Σ-protocols are SHVZK. The simulator Sim, on input instance x and challenge c, operates as follows: (i) Randomly pick $z \leftarrow G$; (ii) Compute $a = f(z) \otimes x^{-c}$; (iii) Output (a, z).

Additionally, Theorem 3 of [36] describes the condition under which a pre-image Σ-protocol is special sound. That is values $\ell \in \mathbb{Z}$ and $u \in \mathcal{G}$ are known such that:

1. $gcd(c_1 - c_2, \ell) = 1$ for all $c_1, c_2 \in \mathsf{C}_\Sigma, c_1 \neq c_2$
2. $f(u) = x^\ell$.

Whenever such conditions are met, it is easy to see that the pre-image Σ-protocol also satisfies the following stronger flavor of special soundness.

Definition 4 (Special Soundness with Randomness Recoverability). *A Σ-protocol $\Sigma = (\mathsf{P}_0, \mathsf{P}_1, \mathsf{V})$ is special sound with randomness recoverability if there exists a PPT extractor ExtR, such that on input x and 2 accepting transcripts $(a, c_0, z_0), (a, c_1, z_1)$ for x, where $c_0 \neq c_1$, it holds that:*

$$\Pr[(x, w) \in \mathcal{R}_\mathcal{L} \wedge a = \mathsf{P}_0(x, w; r) | (w, r) \leftarrow \mathsf{ExtR}(x, a, c_0, z_0, c_1, z_1)] = 1.$$

Indeed, ExtR simply runs the extractor guaranteed by theorem 3 of [36] to recover w and then computes $k = z * w^{-c}$.

3.4 Pseudo-Random Functions

A pseudorandom function (PRF) family is specified by poly-time algorithms $(\mathsf{KeyGen}, \mathsf{Eval})$, where KeyGen is a randomized key generation algorithm that takes in input a security parameter 1^λ and outputs a random key $K \in \mathcal{K}$. Eval is a deterministic evaluation algorithm, which takes in input a key $K \in \mathcal{K}$ and an input X in a domain $\mathcal{X}$ and evaluates a function $\mathsf{Eval}(K, X)$ in a range $\mathcal{Y}$.

The standard security definition for PRFs requires that the adversary is unable to distinguish an oracle outputting real pseudorandom values from an oracle using a random functions.

Definition 5 (Real-or-Random Security). *A pseudorandom function $\mathsf{PRF} = (\mathsf{KeyGen}, \mathsf{Eval})$ with keyspace $\mathcal{K}$, input space $\mathcal{X}$ and output space $\mathcal{Y}$. Let Ω be the set of all functions that map inputs in the space $\mathcal{X}$ to outputs in the space $\mathcal{Y}$. The pseudorandom function PRF is called secure if for all PPT adversaries A:*

$$\left| \Pr[\mathsf{A}^{\mathsf{Eval}(K,\cdot)}(1^\lambda) = 1 \mid K \leftarrow\$ \mathsf{KeyGen}(1^\lambda)] - \Pr[\mathsf{A}^{R(\cdot)}(1^\lambda) = 1 \mid R \leftarrow\$ \Omega] \right| \leq \mathsf{negl}(\lambda).$$

3.5 Distributed Pseudo-Random Functions (DPRFs)

A Distributed PRF [9] (DPRF) is a tuple of poly-time algorithms (Setup, KeyGen, Share, PEval, Eval, Combine) with the following specification.

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^N, 1^t, 1^\ell)$: On input a security parameter 1^λ, a number of parties 1^N, a threshold 1^t, and a desired input length 1^ℓ, outputs public parameters pp, which are considered implicit input to all other algorithms. The public parameters implicitly define a key space $\mathcal{K}$, an input space $\mathcal{X} = \{0,1\}^\ell$, and an evaluation space $\mathcal{Y}$.
- $K \leftarrow \mathsf{KeyGen}(1^\lambda)$: On input the security parameter 1^λ, output a key $K \in \mathcal{K}$.
- $(SK_1, \ldots, SK_n) \leftarrow \mathsf{Share}(SK_0)$: it takes as input a random master secret key $SK_0 \in \mathcal{K}$ and outputs a tuple of shares $(SK_1, \ldots, SK_N) \in \mathcal{K}^N$, which form a (t, N)-threshold secret sharing of SK_0.
- $Y_i \leftarrow \mathsf{PEval}(SK_i, X)$: On input a key share $SK_i \in \mathcal{K}$ and an input $X \in \mathcal{X}$ and outputs a partial evaluation $Y_i \in \mathcal{Y}$.
- $Y \leftarrow \mathsf{Combine}(\mathcal{S}, \{Y_i\}_{i\in\mathcal{S}})$: On input a t-subset $\mathcal{S} \subset [N]$ together with t partial evaluations $\{Y_i\}_{i\in\mathcal{S}}$, where $Y_i \in \mathcal{Y}$ for all $i \in \mathcal{S}$, and outputs a value $Y \in \mathcal{Y}$.
- $Y \leftarrow \mathsf{Eval}(SK_0, X)$: it takes as input the master secret key $SK_0 \in \mathcal{K}$ and $X \in \mathcal{X}$ and outputs a value $Y \in \mathcal{Y}$.

Definition 6 (Consistency). *Let integers $\ell, t, N \in \mathsf{poly}(\lambda)$ s.t. $2 \le t \le N$. We say that* DPRF *is consistent if for any* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^N, 1^t, 1^\ell)$*, any master key* $SK_0 \leftarrow \mathsf{KeyGen}(1^\lambda)$ *shared according to* $(SK_1, \ldots, SK_N) \leftarrow \mathsf{Share}(SK_0)$*, any t-subset $\mathcal{S} = \{i_1, \ldots, i_t\} \subset [N]$ and any input $X \in \mathcal{X}$, and $Y_{i_j} = \mathsf{PEval}(SK_{i_j}, X)$ for each $j \in [t]$:*

$$\Pr[\mathsf{Eval}(SK_0, X) = \mathsf{Combine}(\mathcal{S}, (Y_{i_1}, \ldots, Y_{i_t}))] \ge 1 - \mathsf{negl}(\lambda).$$

We say that a DPRF provides adaptive security if it remains secure against an adversary that can adaptively choose which parties it wants to corrupt. In particular, the adversary can arbitrarily interleave evaluation and corruption queries as long as they do not allow it to trivially win.

Definition 7 (Adaptive Security). *Let integers $\ell, t, N \in \mathsf{poly}(\lambda)$ s.t. $2 \le t \le N$. We say that a* DPRF *is pseudorandom under adaptive corruptions if for all PPT* A*:*

$$\Pr\left[\mathsf{Exp}^{\mathsf{sec}}_{\mathsf{DPRF}}(1^\lambda, 1^N, 1^t, 1^\ell) = 1\right] \le \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{sec}}_{\mathsf{DPRF}}$ *is defined in Fig. 1.*

[9] This definition is the real-or-random version of the adaptive find-then-guess security of [34]. The two definitions can be shown to be equivalent with a standard hybrid argument over evaluation queries with a multiplicative gap between the advantage functions.

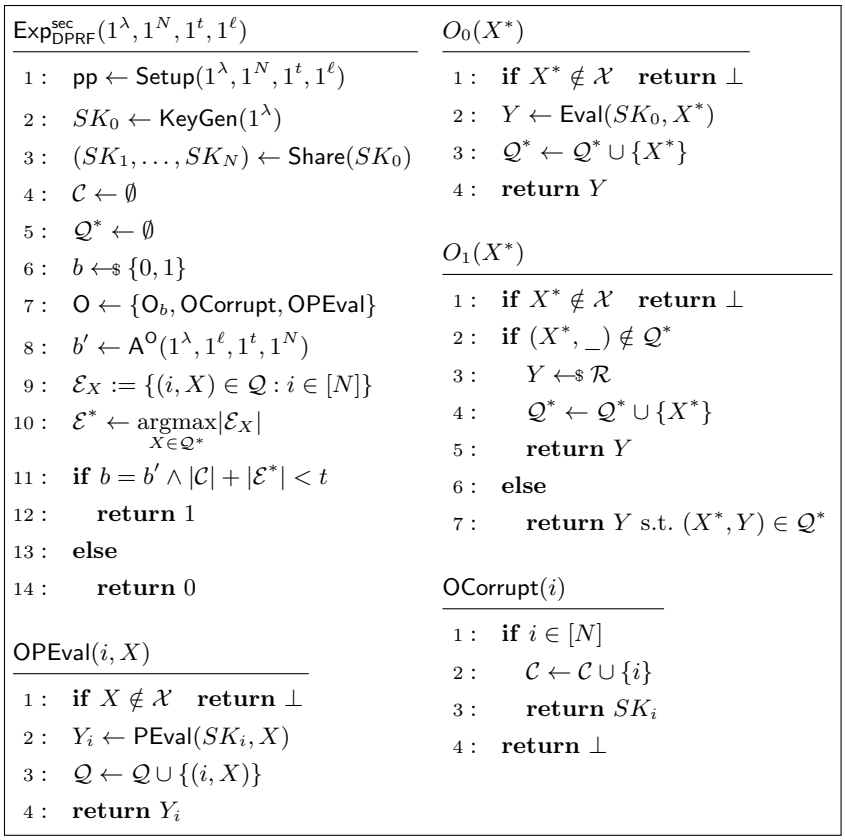

Fig. 1: Adaptive Security Game for Distributed PRFs. For any pair (i, X) in the set of queries $\mathcal{Q}$ made to OPEval, the set $\mathcal{E}_X$ contains all the queries made for different indices for a fixed $X \in \mathcal{Q}$. Then, the operator argmax outputs the set $\mathcal{E}^*$, that is the $\mathcal{E}_X$ with the greatest cardinality given that X is taken from the oracle queries in $\mathcal{Q}^*$. The check of line 10 guarantees that the adversary cannot trivially win using the corrupted keys and the partial evaluation queries.

4 Anamorphic Secret Sharing Σ-Protocol

An anamorphic secret sharing Σ-protocol $\mathsf{a}\Sigma$ is a Σ-protocol for relation $\mathcal{R}$ in which the prover and the verifier have access to an additional private input called the double key. More specifically, the prover P wants to share an anamorphic secret $\hat{s}$ via a t-out-of-N secret sharing by talking to N verifiers $V_1, \ldots, V_N$ individually. For this purpose, P shares with each of the verifiers a specific double key, i.e. P and V_i use dk_i during their interaction. Additionally, P holds a private master double key dk.

Let $\hat{S}$ be the anamorphic secret space, R_Σ the randomness space of Σ, C_Σ the challenge space of Σ. Consider the algorithms $\mathsf{a}\Sigma = (\mathsf{dKeyGen}, \mathsf{aP}_0, \mathsf{aP}_1, \mathsf{ExtShare}, \mathsf{Recon})$ that work as follow w.r.t. a relation $\mathcal{R}$.

- $(\mathsf{dk}, \mathsf{dk}_1, \ldots, \mathsf{dk}_N) \leftarrow \mathsf{dKeyGen}(1^\lambda, 1^t, 1^N)$: on input the security parameter 1^λ, the threshold 1^t, and the number of parties 1^N , it outputs a master double key dk and individual double key for the verifiers $\mathsf{dk}_1, \ldots, \mathsf{dk}_N$.
- $a/\mathsf{fail} \leftarrow \mathsf{aP}_0(x, w, \mathsf{dk}, \hat{s}, \mathsf{st}; r)$ on input statement x, witness w, master double key dk, anamorphic secret $\hat{s} \in \hat{S} \cup \{\perp\}$, a state st, and random coins $r \in \mathsf{R}_\Sigma$ outputs a first-round message a, or an error symbol fail in case of failure.
- $z \leftarrow \mathsf{aP}_1(x, w, a, c, \mathsf{dk}, \hat{s}, \mathsf{st}, r)$: on input statement x, witness w, first-round message a, challenge $c \in \mathsf{C}_\Sigma$, master double key dk, anamorphic secret $\hat{s} \in \hat{S} \cup \{\perp\}$, state st, and randomness r used in the first round, outputs the third-round message z.
- $\mathsf{st}' \leftarrow \mathsf{UpdState}(\mathsf{st})$: on input a state st, outputs an updated state st'. The initial state is indicated with the symbol ϵ.
- $\hat{s}_i/\perp \leftarrow \mathsf{ExtShare}(x, a, c, z, \mathsf{dk}_i, \mathsf{st})$: on input statement x, first-round message a, challenge $c \in \mathsf{C}_\Sigma$, third-round message z, double key of the i-th verifier dk_i, a state st, outputs the i-th share $\hat{s}_i$, or $\perp$ if no share was encoded in the transcript.
- $\hat{s}/\perp/\mathsf{fail} \leftarrow \mathsf{Recon}(\mathcal{S}, \{\hat{s}_i\}_{i\in\mathcal{S}})$: on input a t-subset $\mathcal{S} \subset [N]$ together with t shares $\{\hat{s}_i\}_{i\in\mathcal{S}}$, outputs a value $\hat{s} \in \hat{S}$, or a special symbol $\perp$, or an error symbol fail in case of failure.

Definition 8 (Anamorphic Secret Sharing Σ-protocol). *We say that $\mathsf{a}\Sigma$ is an anamorphic secret sharing tuple for Σ if Def. 9, Def. 10, Def. 11, and Def. 12 are satisfied. Similarly, Σ is said to be anamorphic if it admits an anamorphic secret sharing tuple $\mathsf{a}\Sigma$ satisfying that properties.*

As for Σ-protocols, we refer to $\langle \mathsf{aP}(\cdot), \mathsf{V}\rangle(\cdot)$ as the execution of an anamorphic Σ-protocol Σ between a prover aP and a verifier V, where a is computed using aP_0 and sent to V, c is sampled at random from C_Σ by V or adversarially chosen if V is corrupted and sent to aP and finally z is computed by aP using aP_1 and then sent to V. If aP_0 outputs fail, then the output of trans is fail, otherwise the output of trans will be of the form (a, c, z).

As done in [5] our games will take care of the proper update of the state variables. For this reason also correctness it is more easily formalized via a distinguishing game.

The adversary gets oracle access to O_b for a uniformly sampled bit b. In case $b = 0$, the adversary, querying the oracle with an anamorphic secret $\hat{s}$ and a set of reconstruction indices $\mathcal{S}$, receives back a value which is the result of the share generation and the subsequent reconstruction using the set $\mathcal{S}$ (when $|\mathcal{S}| = t$). If $b = 1$ the oracle simply returns the very same $\hat{s}$ back. In particular, the check of line 10 guarantees our robustness property. That is, in case no share is encoded in a transcript, every party will always realize it.

Definition 9 (Correctness). *Let integers $t, N \in \mathsf{poly}(\lambda)$ s.t. $2 \leq t \leq N$.$\mathsf{a}\Sigma$ is correct w.r.t a Σ-protocol Σ if for all PPT* A*:*

$$\Pr\left[\mathsf{Exp}^{\mathsf{Cor}}_{\mathsf{A},\mathsf{a}\Sigma}(1^\lambda, 1^t, 1^N) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{Cor}}_{\mathsf{A},\mathsf{a}\Sigma}(1^\lambda, 1^t, 1^N)$ *is defined in Fig. 2.*

$\mathsf{Exp}^{\mathsf{Cor}}_{\mathsf{A},\mathsf{a}\Sigma}(1^\lambda, 1^t, 1^N)$

1 : Sample random $(x, w) \in \mathcal{R}_{\mathcal{L}}$
2 : $(\mathsf{dk}, \mathsf{dk}_1, \ldots, \mathsf{dk}_N) \leftarrow \mathsf{dKeyGen}(1^\lambda, 1^t, 1^N)$
3 : **for** $i \in [N] : \mathsf{st}_i \leftarrow \epsilon_i$
4 : $b \leftarrow\!\!\$\ \{0, 1\}$
5 : $b' \leftarrow \mathsf{A}^{\mathsf{O}_b}(1^\lambda, 1^t, 1^N, x, w)$
6 : **if** $b = b'$
7 : **return** 1
8 : **else**
9 : **return** 0

$\mathsf{O}_1(\hat{s}, \mathcal{S})$

1 : **return** $\hat{s}$

$\mathsf{O}_0(\hat{s}, \mathcal{S})$

1 : **if** $\hat{s} \notin \hat{S} \cup \{\perp\} \vee |\mathcal{S}| \neq t \vee \mathcal{S} \not\subset [N]$
2 : **return** $\hat{s}$
3 : $count = 0$
4 : **for** $i \in [N]$
5 : $\mathsf{tx}_i \leftarrow \mathsf{trans}(\langle \mathsf{aP}(w, \mathsf{dk}, \hat{s}, \mathsf{st}_i), \mathsf{V}\rangle(x))$
6 : **if** $\mathsf{tx}_i = \mathsf{fail}$
7 : **return** fail
8 : $\hat{s}_i \leftarrow \mathsf{ExtShare}(x, \mathsf{tx}_i, \mathsf{dk}_i, \mathsf{st}_i)$
9 : **if** $\hat{s}_i = \perp$
10 : $count++$
11 : $\mathsf{st}_i \leftarrow \mathsf{UpdState}(\mathsf{st}_i)$
12 : **if** $\hat{s} = \perp$
13 : **if** $count = N$
14 : **return** $\perp$
15 : **else return** fail
16 : **return** $\mathsf{Recon}(\mathcal{S}, \{\hat{s}_i\}_{i \in \mathcal{S}})$

Fig. 2: Correctness Game for Anamorphic Secret Sharing Σ-protocols.

For the following definitions we take inspiration from Sec. 4.1 of [30]. The anamorphic security property requires that the dictator having access to $(x, w, \hat{s})$ but not to the double keys $(\mathsf{dk}, \mathsf{dk}_1, \ldots, \mathsf{dk}_N)$, cannot tell whether it is observing interactions produced by a real prover or by an anamorphic prover. More formally, we introduce the following definition.

Definition 10 (Anamorphic Security). *Let integers $t, N \in \mathsf{poly}(\lambda)$ s.t. $2 \leq t \leq N$. $\mathsf{a}\Sigma$ is anamorphic secure w.r.t. a Σ-protocol Σ for a relation $\mathcal{R}_{\mathcal{L}}$ if, for all PPT* A*:*

$$\Pr\left[\mathsf{Exp}^{\mathsf{Ana}}_{\mathsf{A},\Sigma,\mathsf{a}\Sigma}(1^\lambda, 1^t, 1^N) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

$\mathsf{Exp}^{\mathsf{Ana}}_{\mathsf{A},\Sigma,\mathsf{a}\Sigma}(1^\lambda, 1^t, 1^N)$

1 : Sample random $(x, w) \in \mathcal{R}_{\mathcal{L}}$
2 : $(\mathsf{dk}, \mathsf{dk}_1, \ldots, \mathsf{dk}_N) \leftarrow \mathsf{dKeyGen}(1^\lambda, 1^t, 1^N)$
3 : **for** $i \in [N] : \mathsf{st}_i \leftarrow \epsilon_i$
4 : $b \leftarrow\!\!\$\ \{0, 1\}$
5 : $b' \leftarrow \mathsf{A}^{\mathsf{O}_b(\cdot,\cdot)}(1^\lambda, 1^t, 1^N, x, w)$
6 : **if** $b = b'$
7 : **return** 1
8 : **else**
9 : **return** 0

$\mathsf{O}_0(i, \hat{s})$

1 : **if** $i \notin [N] \vee \hat{s} \notin \hat{S} \cup \{\perp\}$
2 : **return** $\perp$
3 : $\mathsf{tx} \leftarrow \mathsf{trans}(\langle \mathsf{P}(w), \mathsf{A}(1^\lambda, 1^t, 1^N, x, w)\rangle(x))$
4 : **return** tx

$\mathsf{O}_1(i, \hat{s})$

1 : **if** $i \notin [N] \vee \hat{s} \notin \hat{S} \cup \{\perp\}$
2 : **return** $\perp$
3 : $\mathsf{tx} \leftarrow \mathsf{trans}(\langle \mathsf{aP}(w, \mathsf{dk}, \hat{s}, \mathsf{st}_i), \mathsf{A}(1^\lambda, 1^t, 1^N, w)\rangle(x))$
4 : $\mathsf{st}_i \leftarrow \mathsf{UpdState}(\mathsf{st}_i)$
5 : **return** tx

Fig. 3: Anamorphic Security Game for Anamorphic Secret Sharing Σ-protocols.

where $\mathsf{Exp}^{\mathsf{Ana}}_{\mathsf{A},\Sigma,\mathsf{a}\Sigma}(1^\lambda, 1^t, 1^N)$ *is defined in Fig. 3.*

We model the secrecy notion as a fully adaptive game where the adversary has access to three oracles: a corruption oracle $\mathsf{OCorrupt}(i)$, a share generation oracle $\mathsf{OShare}(\hat{s})$, and a share extraction oracle $\mathsf{OExtShare}(i, j)$. $\mathsf{OCorrupt}$ returns the double key of party i, OShare returns N protocol transcripts resulting from sharing the queried value $\hat{s}$, and $\mathsf{OExtShare}$ returns the extracted share from the j-th transcript of the i-th call to OShare. After a polynomial number of queries to these oracles, A outputs two values and gets N transcripts corresponding to the sharing of a randomly selected one. A wins the game if it can guess the shared valued with probability significantly better than $1/2$, under the condition it did not corrupt more than $t - 1$ keys. Notice that before outputting its guess, A is still allowed to perform new sharing queries and to extract the shares for all the secret sharings, except the challenged one.

Definition 11 (Adaptive Secrecy). *Let integers* $t, N \in \mathsf{poly}(\lambda)$ *s.t.* $2 \leq t \leq N$. $\mathsf{a}\Sigma$ *satisfies adaptive secrecy w.r.t. a* Σ*-protocol* Σ *for a relation* $\mathcal{R}_{\mathcal{L}}$ *if for all PPT* A*:*

$$\Pr\left[\mathsf{Exp}^{\mathsf{AdSec}}_{\mathsf{A},\mathsf{a}\Sigma}(1^\lambda, 1^t, 1^N) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{AdSec}}_{\mathsf{A},\mathsf{a}\Sigma}(1^\lambda, 1^t, 1^N)$ *is defined in Fig. 4.*

We define a simulatability property to guarantee that any coalition of less than t receivers learns nothing more about the witness of what it is already leaked by Σ. We do that by requiring the existence of a simulator Sim, which acts as filter. The simulatability game interacts with the ITM running the proving

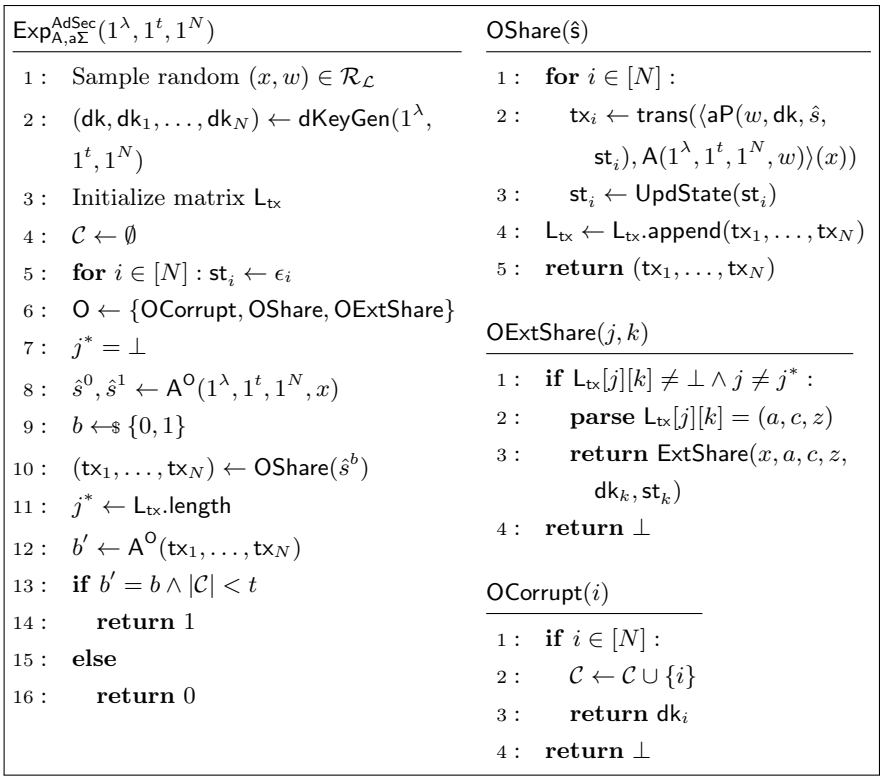

Fig. 4: Adaptive Secrecy Game for Anamorphic Secret Sharing Σ-protocols.

algorithms $\mathsf{P}_0, \mathsf{P}_1$ of Σ with hardcoded statement x and witness w. Only based on the anamorphic secret, the double key of the party and the corresponding states, Sim decides whether to forward the first-round message to A. A suitable first-round message must be found within λ invocations of Sim. After that, the simulatability game simply acts as a proxy between A and the regular prover. We require that the distribution of the transcripts generated from an execution between an anamorphic prover and the adversary is indistinguishable from the interaction mediated by the simulatability game.

Definition 12 (Simulatability). *Let integers $t, N \in \mathsf{poly}(\lambda)$ s.t. $2 \leq t \leq N$. $\mathsf{a}\Sigma$ is simulatable w.r.t $\Sigma = (\mathsf{P}, \mathsf{V})$ if there exists a PPT $\mathsf{Sim}_{\mathsf{a}\Sigma}$ such that for all PPT A:*

$$\Pr\left[\mathsf{Exp}^{\mathsf{Sim}}_{\mathsf{A},\mathsf{a}\Sigma}(1^\lambda, 1^t, 1^N) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{Sim}}_{\mathsf{A},\mathsf{a}\Sigma}(1^\lambda, 1^t, 1^N)$ is defined in Fig. 5.

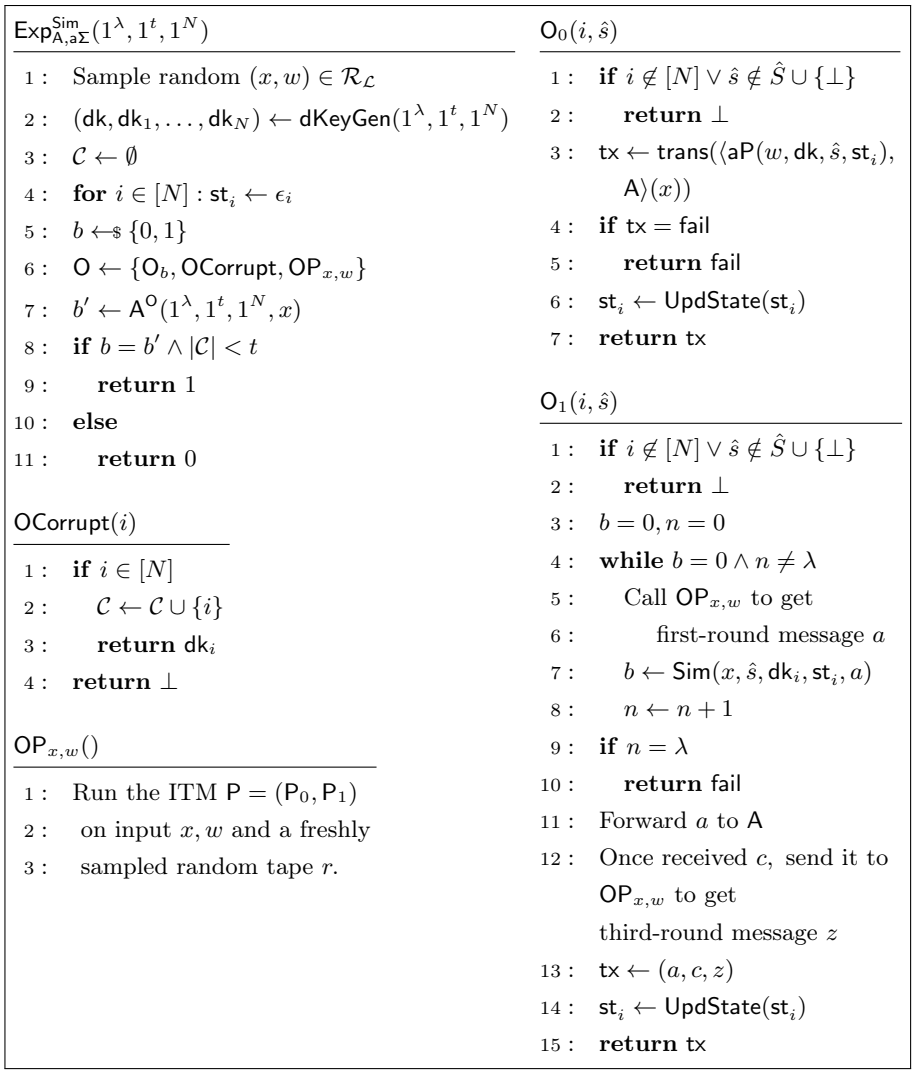

Fig. 5: Simulatability Game for Anamorphic Secret Sharing Σ-protocols.

4.1 Anamorphic Secret Sharing for Pre-Image Σ-Protocols

Let Σ be a pre-image Σ-protocol with associated one-way homomorphism $f : \mathcal{G} \to \mathcal{H}$, and DPRF be a distributed pseudo-random function with output space $\mathcal{G}$, PRF be a pseudo-random function with input space $\mathcal{H}$ and output space $\{0, 1\}$. We construct an anamorphic secret sharing Σ-protocol Σ_{an} as described in Fig. 6. For Σ_{an}, the initial state is $\mathsf{st}_i = (0, i)$.

$\mathsf{aP}_0(x, w, \mathsf{dk}, \hat{s}, \mathsf{st}; r)$

1 : **parse** $\mathsf{st} = (ctr, i)$
2 : **parse** $\mathsf{dk} = (\mathsf{pp}, K, SK_0)$
3 : **if** $\hat{s} = \bot : b = 1$ **else** $: b = 0$
4 : $v \leftarrow 1 - b, u \leftarrow 0$
5 : **while** $v \neq b \wedge u \neq \lambda$
6 : **if** $b = 0$
7 : $\rho_i \leftarrow \hat{s} * \mathsf{DPRF.Eval}(SK_0, ctr||i||u)$
8 : **else**
9 : $\rho_i \leftarrow\!\!\$\ \mathcal{G}$
10 : $a_i \leftarrow f(\rho_i)$
11 : $v \leftarrow \mathsf{PRF.Eval}(K, a_i)$
12 : $u \leftarrow u + 1$
13 : **if** $u = \lambda :$ **return** fail **else** $:$ **return** a_i

$\mathsf{aP}_1(x, w, a, c, \mathsf{dk}, \hat{s}, \mathsf{st}, r)$

1 : **parse** $\mathsf{st} = (ctr, i)$
2 : Recover ρ_i as in aP_0
3 : $z_i \leftarrow \rho_i * w^c$
4 : **return** z_i

$\mathsf{ExtShare}(x, a, c, z, \mathsf{dk}_i, \mathsf{st})$

1 : **parse** $\mathsf{st} = (ctr, i), \mathsf{dk}_i = (\mathsf{pp}, K, SK_i)$
2 : **if** $\mathsf{PRF.Eval}(K, a) = 1$
3 : **return** $\bot$
4 : **for** $j \in [N]$:
5 : **for** $u \in [0, \lambda - 1]$:
6 : $E_{j,u} \leftarrow \mathsf{DPRF.PEval}(SK_i, ctr||j||u)$
7 : **return** $(a, c, z, \{E_{j,u}\}_{j\in[N],u\in[\lambda]}, K)$

$\mathsf{UpdState}(\mathsf{st})$

1 : **parse** $\mathsf{st} = (ctr, i)$
2 : **return** $(ctr + 1, i)$

$\mathsf{dKeyGen}(1^\lambda, 1^t, 1^N)$

1 : $\mathsf{pp} \leftarrow \mathsf{DPRF.Setup}(1^\lambda, 1^\lambda, 1^t, 1^N)$
2 : $K \leftarrow \mathsf{PRF.KeyGen}(1^\lambda)$
3 : $SK_0 \leftarrow \mathsf{DPRF.KeyGen}(1^\lambda)$
4 : $(SK_1, \ldots, SK_N) \leftarrow \mathsf{DPRF.Share}(SK_0)$
5 : **for** $i \in [N]$
6 : $\mathsf{dk}_i \leftarrow (\mathsf{pp}, K, SK_i)$
7 : $\mathsf{dk} \leftarrow (\mathsf{pp}, K, SK_0)$
8 : **return** $(\mathsf{dk}, \mathsf{dk}_1, \ldots, \mathsf{dk}_N)$

$\mathsf{Recon}(\mathcal{S}, \{\hat{s}_i\}_{i\in\mathcal{S}})$

1 : **for** $i \in \mathcal{S}$:
2 : **if** $\hat{s}_i = \bot$
3 : **return** $\bot$
4 : **parse** $\hat{s}_i = (a_i, c_i, z_i, \{E^i_{j,u}\}_{j\in[N],u\in[\lambda]}, K)$
5 : $k \leftarrow \mathcal{S}[1], m \leftarrow \mathcal{S}[2], n \leftarrow 0$
6 : **while** $n \neq \lambda :$ // for efficiency
7 : **for** $u_k \in [0, n]$:
8 : **for** $u_m \in [0, n]$:
9 : $R_k \leftarrow \mathsf{DPRF.Combine}(\mathcal{S}, \{E^j_{k,u_k}\}_{j\in\mathcal{S}})$
10 : $R_m \leftarrow \mathsf{DPRF.Combine}(\mathcal{S}, \{E^j_{m,u_m}\}_{j\in\mathcal{S}})$
11 : $a'_k \leftarrow a_k \otimes f(R_k^{-1})$
12 : $a'_m \leftarrow a_m \otimes f(R_m^{-1})$
13 : **if** $a'_k = a'_m$ **then** break
14 : $n \leftarrow n + 1$
15 : **if** $n = \lambda$
16 : **return** fail
17 : $z'_k \leftarrow z_k * R_k^{-1}$
18 : $z'_m \leftarrow z_m * R_m^{-1}$
19 : $\hat{s} \leftarrow \Sigma.\mathsf{ExtR}(x, a'_k, c_k, c_m, z'_k, z'_m)$
20 : **return** $\hat{s}$

Fig. 6: Anamorphic Secret Sharing tuple Σ_{an} for any Pre-Image Σ-Protocol Σ. The **while** in line 6 of Recon is for efficiency reasons due to the fact that u_k and u_m would be, in practice, very small numbers.

Theorem 2. *Let Σ be a pre-image Σ-protocol (Def. 3 and Sec. 3.3) with associated function $f : \mathcal{G} \to \mathcal{H}$, DPRF be a Distributed PRF, with output space $\mathcal{G}$, enjoying consistency (Def. 6) and adaptive security (Def. 7), PRF be a PRF (Def. 5) with input space $\mathcal{H}$ and output space $\{0,1\}$. Then Σ_{an} (Fig. 6) is an Anamorphic Secret Sharing tuple for Σ (Def. 8) with anamorphic secret space $\hat{S} = \mathcal{G}$.*

Proof. We will prove, in the following lemmas, that Σ_{an} is an anamorphic secret sharing tuple for Σ by proving correctness, adaptive secrecy, and simulatability.

Lemma 1 (Correctness). *If PRF is secure (Def. 5), DPRF is consistent (Def. 6) and adaptive secure (Def. 7), and Σ is a pre-image Σ-protocol, then Σ_{an} is correct (Def. 9).*

Proof. Let us first evaluate the probability that O_0 returns fail when $\hat{s} \neq \perp$. This event can happen because of the checks at lines 7 of O_0 is satisfied or if Recon returns fail. The check of line 7 of O_0 (Fig. 2) is satisfied iff for any of the N possible receivers, aP_0 did not find a first-round message a_k within λ attempts s.t. $\mathsf{PRF.Eval}(K, a_k) = 0$. This must happen with probability at most $(1/2)^\lambda + \mathsf{negl}(\lambda)$, otherwise a distinguisher for the PRF's security can be easily built using the adversary of the correctness game. Given that the check of line 7 of O_0 (Fig. 2) did not pass for any $i \in [N]$ (i.e., there was no a_i s.t. $a_i = \mathsf{fail}$), Recon will return fail iff $n = \lambda$ at line 15 (Fig. 6). However, by construction this event will not happen, except with negligible probability, as any a_k, a_m with $k \neq m \in [N]$ are s.t. $a_k = f(\rho_k)$ with $\rho_k = \hat{s} * \mathsf{DPRF.Eval}(SK_0, ctr||i||u_k)$ and $a_m = f(\rho_m)$ with $\rho_k = \hat{s} * \mathsf{DPRF.Eval}(SK_0, ctr||i||u_m)$ with $u_k, u_m \in \{0, \ldots, \lambda\}$ and all possible values u_m and u_k are eventually tested. Thus, the unblinding procedure of lines $9 - 13$ of Recon (Fig. 6) is successful, except with negligible probability due the negligible correctness error of the DPRF. Let us now evaluate the probability that O_0 returns fail when $\hat{s} = \perp$. This event can happen because of the checks at line 7 or at line 16 of O_0 (Fig. 2) are satisfied. Similarly to what argued before, due to the PRF's security, the check of line 7 is only satisfied with negligible probability. Furthermore, given that the checks of line 7 in O_0 (Fig. 2) passed for all a_i with $i \in [n]$, $\mathsf{ExtShare}$ will return $\perp$ for all $i \in [N]$ and thus *count* will be equal to N in O_0 (Fig. 2). As a result, line 16 of in O_0 (Fig. 2) will not be executed. The above reasoning also shows that if $\hat{s} = \perp$, then O_0 will return $\perp$, except with negligible probability. What remains to show is that whenever $\hat{s} \neq \perp$, the Recon algorithm executed at line 17 (Fig. 6) returns $\hat{s}$. The only cause of error, aside the negligible correctness errors of the DPRF and of $\Sigma.\mathsf{ExtR}$, is due to finding a'_k and a'_m s.t. $a'_k = a'_m \neq f(\hat{s})$. However, it is easy to show that due to the pseudo-randomness of DPRF this can only happen with negligible probability.

Lemma 2 (Anamorphic Security). *If PRF is secure (Def. 5) and DPRF is adaptively secure (Def. 7), then Σ_{an} satisfies anamorphic security (Def. 10).*

Proof. We now prove our theorem via a series of indistinguishable hybrids starting from the game with $b = 0$ and ending with the same game with $b = 1$. The proof goes through the following hybrids.

$\mathcal{H}_0$**:** It is the anamorphic game with $b = 0$.
$\mathcal{H}_1$**:** It is identical to $\mathcal{H}_0$ except of the following changes: 1) P_0 sets the values b and v as aP_0 does in line 3 (Fig. 6), 2) it executes the while loop as aP_0 but in line 7 (Fig. 6) the value ρ_i is computed as $\hat{s} * r$ where r is a uniformly random value from $\mathcal{G}$, 3) it samples the bit v of line 11 (Fig. 6) uniformly at random, 4) it has the same fail condition of aP_0, returning fail if lambda attempts were reached.
$\mathcal{H}_2$**:** It is identical to $\mathcal{H}_1$ except that instead of just sampling v at random for each iteration, aP_0 keeps track of (a_i, v_i) and re-uses the same v_i should the same a_i occur again.
$\mathcal{H}_3$**:** It is identical to $\mathcal{H}_2$ except that v is computed as $\mathsf{PRF.Eval}(K, a_i)$
$\mathcal{H}_4$**:** It is identical to $\mathcal{H}_3$ except that r is computed as $\mathsf{DPRF.Eval}(SK_0, ctr||i||u)$.

The proof is concluded by observing that $\mathcal{H}_4$ is identical to the anamorphic security game with $b = 1$ and that $\mathcal{H}_0 \approx_s \mathcal{H}_1 \approx_s \mathcal{H}_2 \approx_c \mathcal{H}_3 \approx_c \mathcal{H}_4$. Let us prove the above claims:

$\mathcal{H}_0 \approx_s \mathcal{H}_1$**:** $\mathcal{H}_1$ has the same distribution of $\mathcal{H}_0$ except for the fact that the prover could return fail in $\mathcal{H}_1$ while that would never happen in $\mathcal{H}_0$. However, the probability that the prover outputs fail in $\mathcal{H}_1$ is $(1/2)^\lambda$. Hence, A distinguishes with negligible probability.
$\mathcal{H}_1 \approx_s \mathcal{H}_2$**:** the two hybrids differ only if an a_i value repeats across all attempts, however this only happens with negligible probability given that $\mathcal{G}$ has superpoly size.
$\mathcal{H}_2 \approx_c \mathcal{H}_3$**:** this can be shown via a trivial reduction to the security of the PRF.
$\mathcal{H}_1 \approx_c \mathcal{H}_2$**:** this can be shown via a trivial reduction to the security of the DPRF.

Lemma 3 (Adaptive Secrecy). *If* DPRF *is adaptively secure (Def. 7) then* Σ_{an} *satisfies adaptive secrecy (Def. 11).*

Proof. The proof goes through the following hybrids.

$\mathcal{H}_0$**:** it is the adaptive game of Fig. 4, with secret fixed with $\hat{s}_0$.
$\mathcal{H}_1$**:** identical to $\mathcal{H}_0$ except that when executing the call $\mathsf{OShare}(\hat{s}^0)$ at line 10 of $\mathsf{Exp}^{\mathsf{AdSec}}_{\mathsf{A},\mathsf{a}\Sigma}(1^\lambda, 1^t, 1^N)$ (Fig. 4), the prover aP_0 samples a random value instead of computing, for each receiver and each attempt, $\mathsf{DPRF.Eval}(SK_0, ctr||i||u)$.
$\mathcal{H}_2$**:** identical to $\mathcal{H}_1$, except that the secret is fixed with $\hat{s}_1$.
$\mathcal{H}_3$**:** identical to $\mathcal{H}_2$, except that when executing the call $\mathsf{OShare}(\hat{s}^0)$ at line 10 of $\mathsf{Exp}^{\mathsf{AdSec}}_{\mathsf{A},\mathsf{a}\Sigma}(1^\lambda, 1^t, 1^N)$, aP_0 computes the first round message a as in the original game, i.e. changes the random value to $\mathsf{DPRF.Eval}(SK_0, ctr||i||u)$.

The proof is concluded by observing that $\mathcal{H}_3$ is identical to the adaptive security game with $b = 1$ and that $\mathcal{H}_0 \approx_c \mathcal{H}_1 \equiv \mathcal{H}_2 \approx_c \mathcal{H}_3$. Let us prove that $\mathcal{H}_0 \approx_c \mathcal{H}_1$ and $\mathcal{H}_2 \approx_c \mathcal{H}_3$.

$\mathcal{H}_0 \approx_c \mathcal{H}_1$**:** Let us assume that $\mathcal{H}_0$ and $\mathcal{H}_1$ are computationally distinguishable and that A wins with non-negligible probability, then we construct B breaking adaptive security of DPRF (Fig. 7). B works as follows.

- Upon receiving a $\mathsf{Corrupt}$ query from A for the receiver i, forward i to the corruption oracle of the adaptive security of the DPRF, receiving back SK_i. Craft a double key dk_i containing SK_i and send dk_i to A.
- Upon receiving a OShare query with input $\hat{s}$, replace all the calls to $\mathsf{DPRF.Eval}(SK_0, ctr||i||u)$ in aP_0 by querying $\mathsf{OPEval}(i, ctr||i||u)$ for all $j \in [N]$ and all the required attempts of $u \in [\lambda]$ and then use $\mathsf{Combine}$ to obtain $\mathsf{Eval}(SK_0, ctr||i||u)$.
- B creates the transcripts $\mathsf{tx}_1, \ldots, \mathsf{tx}_N$ of the challenge sharing normally produced in line 10, without calling OShare. Instead, to simulate the first-round messages of such transcripts, B queries the oracle O_b of the adaptive security of DPRF (Fig. 7) with $j^*||i||u_i$, for all $i \in [N]$ and all the required attempts of $u_i \in [\lambda]$. B then completes all such transcripts interacting with A to get the challenge values and using the witness to compute the third-round messages.
- Upon receiving a query to $\mathsf{OExtShare}$ (Fig. 4) with input j, k, retrieve the partial evaluations by querying OPEval (Fig. 7) with input $(k, j||p||u)$ for each $p \in [N]$ and $u \in [\lambda]$, receiving back the partial evaluations $\{E^k_{p,u}\}_{p\in[N],u\in[\lambda]}$. Then, craft the extracted share by including the partial evaluations $\{E^k_{p,u}\}_{p\in[N],u\in[\lambda]}$.
- Finally, output whatever A outputs.

B perfectly simulates $\mathcal{H}_0$ when the challenge oracle is O_0 and perfectly simulates $\mathcal{H}_1$ when the challenge oracle is O_1. Thus, if A distinguishes between $\mathcal{H}_0$ and $\mathcal{H}_1$ with non-negligible advantage, B wins the DPRF adaptive security game with the same advantage. Indeed, B behaves as an admissible adversary for $\mathsf{Exp}^{\mathsf{sec}}_{\mathsf{DPRF}}$ whenever A behaves as an admissible adversary for $\mathsf{Exp}^{\mathsf{AdSec}}_{\mathsf{A},\mathsf{a}\Sigma}$. This is because B will not invoke $\mathsf{Corrupt}$ for more than $t-1$ indexes as required by $\mathsf{Exp}^{\mathsf{sec}}_{\mathsf{DPRF}}$, and will not ask for any partial evaluations of the type $(_, j^*||_||_)$ because oracle calls to $\mathsf{OExtShare}$ on j^* are forbidden to A.

$\mathcal{H}_2 \approx_c \mathcal{H}_3$**:** Works exactly as in $\mathcal{H}_0 \approx_c \mathcal{H}_1$.

Lemma 4 (Simulatability). *If* DPRF *is adaptively secure (Def. 11), then* $\mathsf{a}\Sigma$ *is simulatable (Def. 12).*

Proof. We start by defining $\mathsf{Sim}_{\mathsf{a}\Sigma}$ in Fig. 7.

We now prove our theorem via a series of indistinguishable hybrids starting from the simulatability game (Fig. 5) with $b = 0$ and ending with the same game with $b = 1$.

$\mathcal{H}_0$: It coincides with the simulatability game with $b = 0$

$\mathcal{H}_1$: It is identical to $\mathcal{H}_0$ except that aP_0, when $\hat{s} \neq \perp$, uses the random coins r to compute ρ_i instead of using $\mathsf{DPRF.Eval}$. In particular, the looping condition on u is removed and ρ_i is computed as $\rho_i \leftarrow \hat{s} * r$ instead of $\rho_i \leftarrow \hat{s} * \mathsf{DPRF.Eval}(SK_0, ctr||i||u)$.

$\mathsf{Sim}_{\mathsf{a}\Sigma}(x, \hat{s}, \mathsf{dk}_i, \mathsf{st}_i, a)$

1 : **if** $\hat{s} = \perp : b \leftarrow 1$ **else** $: b \leftarrow 0$
2 : $b' \leftarrow \mathsf{PRF}(K, a)$
3 : **if** $b = b'$
4 : **return** 1
5 : **else**
6 : **return** 0

Fig. 7: Simulator for $\mathsf{a}\Sigma$.

$\mathcal{H}_2$: It is identical to $\mathcal{H}_1$ except that aP_0, when $\hat{s} \neq \perp$ computes $\rho_i \leftarrow r$ ignoring $\hat{s}$.

The proof is concluded by observing that $\mathcal{H}_2$ coincides with the simulatability game with $b = 1$ and that $\mathcal{H}_0 \approx_c \mathcal{H}_1 \equiv \mathcal{H}_2$. Let us prove that $\mathcal{H}_0 \approx_c \mathcal{H}_1$.

$\mathcal{H}_0 \approx_c \mathcal{H}_1$**:** Let A be an adversary breaking the simulatability game with non-negligible advantage, we can construct an adversary breaking the adaptive security game of DPRF. B forwards any corruption query to the corruption oracle of the adaptive security game of the DPRF (Fig. 7). Every time B receives an oracle call to generate a transcript, B computes $\rho_i \leftarrow \hat{s} * Y_i$ where Y_i is obtained by querying the evaluation oracle O_b in Fig. 5 on the current state st_i, then B normally continues its interaction with A. Notice that if $b = 0$, then B perfectly simulates $\mathcal{H}_0$ since Y_i is computed with $\mathsf{DPRF.Eval}(SK_0, \mathsf{st}_i)$, otherwise it perfectly simulates $\mathcal{H}_1$ since Y_i is generated at random. Thus B can output whatever A outputs and win whenever A wins. Notice that if A is admissible, B is an admissible adversary for the secrecy game of DPRF since it does not make too many corruption or partial evaluation queries.

5 Anamorphic Secret Sharing Signatures

We define anamorphic secret sharing signatures as follows. The definitions for standard anamorphic signatures are reported in [27]. An anamorphic secret sharing signature is a tuple of poly-time algorithms $\mathsf{ASIG} = (\mathsf{dKeyGen}, \mathsf{aSig}, \mathsf{ExtShare}, \mathsf{Recon})$ w.r.t a signature scheme $\mathsf{SIG} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is described as follows:

- $(\mathsf{dk}, \mathsf{dk}_1, \ldots, \mathsf{dk}_N) \leftarrow \mathsf{dKeyGen}(1^\lambda, 1^t, 1^N)$: on input the security parameter 1^λ, the threshold 1^t, and the number of parties 1^N, it outputs a master double key dk and individual double key for the receivers $\mathsf{dk}_1, \ldots, \mathsf{dk}_N$.
- $\hat{\sigma}/\mathsf{fail} \leftarrow \mathsf{aSig}(\mathsf{sk}, m, \hat{s}, \mathsf{dk}, \mathsf{st})$ takes as input the signing key sk, a *regular* message m, an *anamorphic* message $\hat{s}$, a master double key dk, a state st and outputs an *anamorphic signature* $\hat{\sigma}$, or an error symbol fail in case of failure.

– $\mathsf{st}' \leftarrow \mathsf{UpdState}(\mathsf{st})$: on input a state st, outputs an updated state st'. The initial state is indicated with the symbol ϵ.
– $\hat{s}_i/\perp \leftarrow \mathsf{ExtShare}(\mathsf{pk}, \hat{\sigma}, \mathsf{dk}_i, \mathsf{st})$: on input a public key pk, a signature $\hat{\sigma}$, a double key of the i-th verifier dk_i, a state st, outputs the i-th share $\hat{s}_i$, or $\perp$ if no share was encoded in the transcript.
– $\hat{s}/\perp/\mathsf{fail} \leftarrow \mathsf{Recon}(\mathcal{S}, \{\hat{s}_i\}_{i\in\mathcal{S}})$: on input a t-subset $\mathcal{S} \subset [N]$ together with t shares $\{\hat{s}_i\}_{i\in\mathcal{S}}$, outputs a value $\hat{s} \in \hat{S}$, or a special symbol $\perp$, or an error symbol fail in case of failure.

Definition 13 (Anamorphic Secret Sharing Signatures). *We say that* ASIG *is an anamorphic secret sharing signature w.r.t a signature* SIG *if Def. 14, Def. 15, Def. 16, and Def. 17 are satisfied. Similarly,* SIG *is said to be anamorphic if it admits an anamorphic secret sharing tuple* ASIG *satisfying that properties.*

Definition 14 (Correctness). *Let integers $t, N \in \mathsf{poly}(\lambda)$ s.t. $2 \leq t \leq N$.* ASIG *is correct if for all PPT* A*:*

$$\Pr\Big[\mathsf{Exp}^{\mathsf{CorSig}}_{\mathsf{A},\mathsf{ASIG}}(1^\lambda, 1^t, 1^N) = 1\Big] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{CorSig}}_{\mathsf{A},\mathsf{ASIG}}(1^\lambda, 1^t, 1^N)$ is defined in Fig. 8.

Definition 15 (Anamorphic Security). *Let integers $t, N \in \mathsf{poly}(\lambda)$ s.t. $2 \leq t \leq N$.* ASIG *satisfies anamorphic security w.r.t to a signature* SIG *if, for all PPT* A*:*

$$\Pr\Big[\mathsf{Exp}^{\mathsf{AnaSig}}_{\mathsf{A},\mathsf{SIG},\mathsf{ASIG}}(1^\lambda, 1^t, 1^N) = 1\Big] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{AnaSig}}_{\mathsf{A},\mathsf{SIG},\mathsf{ASIG}}(1^\lambda, 1^t, 1^N)$ is defined in Fig. 9.

Definition 16 (Adaptive Secrecy). *Let integers $t, N \in \mathsf{poly}(\lambda)$ s.t. $2 \leq t \leq N$.* ASIG *satisfies adaptive secrecy if for all PPT* A*:*

$$\Pr\Big[\mathsf{Exp}^{\mathsf{AdSecSig}}_{\mathsf{A},\mathsf{ASIG}}(1^\lambda, 1^t, 1^N) = 1\Big] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{AdSecSig}}_{\mathsf{A},\mathsf{ASIG}}(1^\lambda, 1^t, 1^N)$ is defined in Fig. 10.

Definition 17 (Weak Recipient Unforgeability). *Let integers $t, N \in \mathsf{poly}(\lambda)$ s.t. $2 \leq t \leq N$.* ASIG *is weak recipient unforgeable w.r.t a signature* SIG *if for all* PPT A*:*

$$\Pr\big[\mathsf{Exp}^{\mathsf{WReUnf}}_{\mathsf{A},\mathsf{SIG},\mathsf{ASIG}}(1^\lambda, 1^t, 1^N) = 1\big] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{WReUnf}}_{\mathsf{A},\mathsf{SIG},\mathsf{ASIG}}(1^\lambda, 1^t, 1^N)$ is defined in Fig. 11.

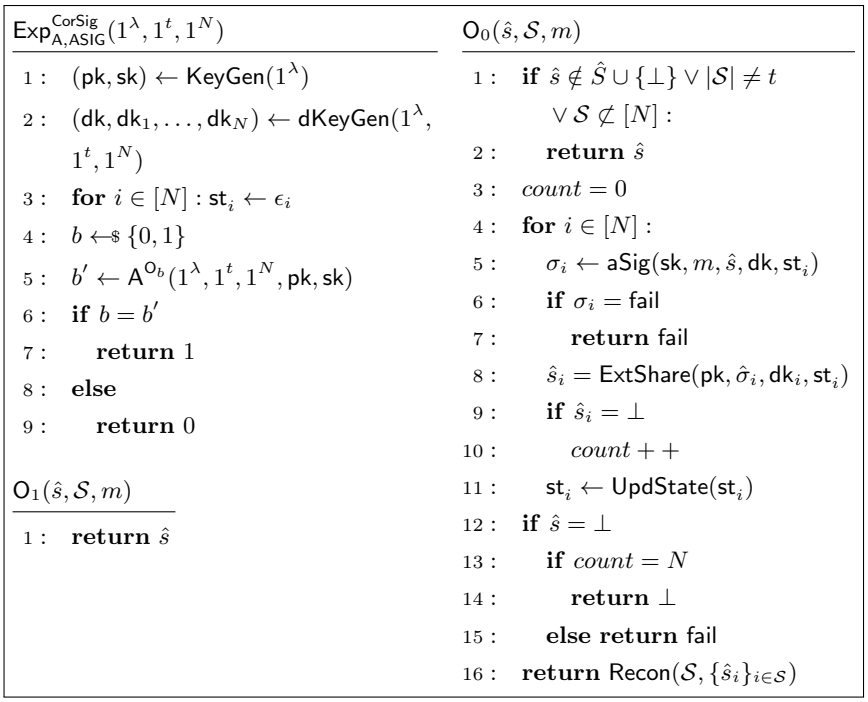

Fig. 8: Correctness Game for Anamorphic Secret Sharing Signatures.

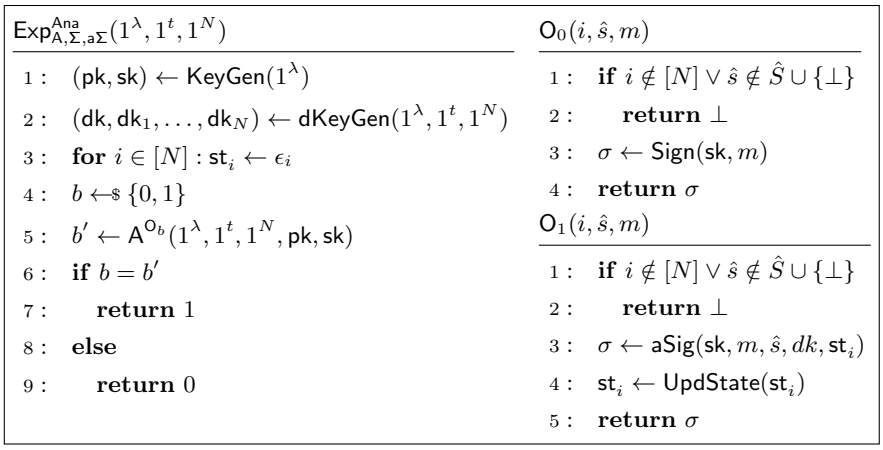

Fig. 9: Anamorphic Security Game for Secret Sharing Signatures.

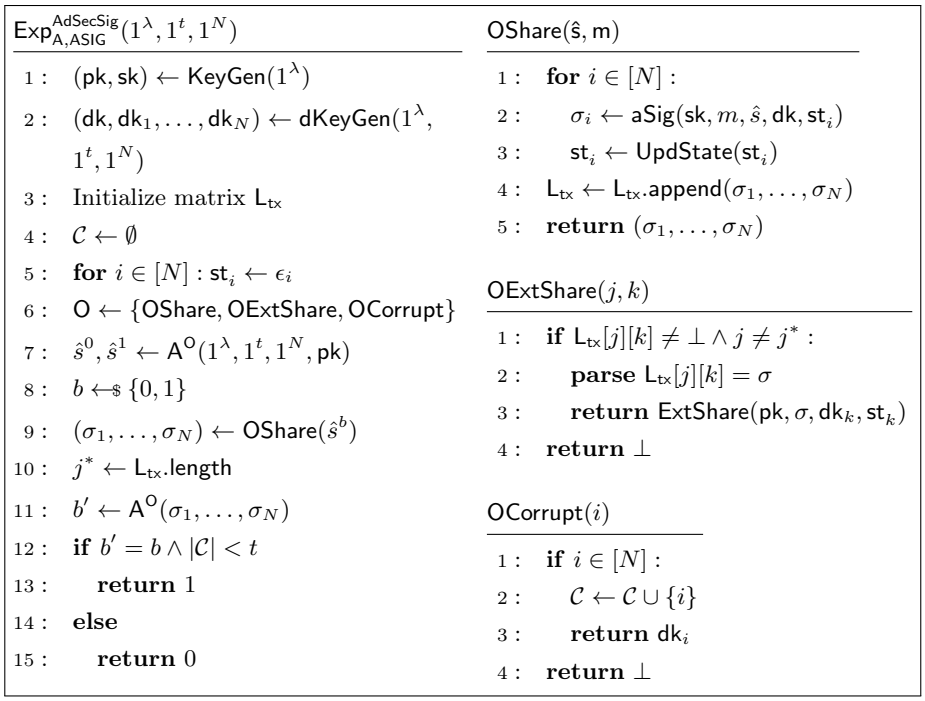

Fig. 10: Adaptive Secrecy Game for Anamorphic Secret Sharing Signatures.

5.1 Fiat-Shamir gives Anamorphic Secret Sharing Signatures

The well-known Fiat-Shamir transform compiles any Σ-Protocol $\Sigma = (\mathsf{P}_0, \mathsf{P}_1, \mathsf{V})$ for a relation $\mathcal{R}$ that is secure against passive attacks (Thm. 1) into a signature scheme $\mathsf{SIG} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. The transform uses an hash function H modeled as a random oracle. Below, with a slight abuse of notation, we identify the relation $\mathcal{R}$ with the sampling algorithm $\mathcal{R}$ that on input 1^λ returns an appropriate random pair $(x, w) \in \mathcal{R}$. In more details, SIG is defined as follows:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$: $(x, w) \leftarrow \mathcal{R}(1^\lambda)$, return $(\mathsf{pk} = x, \mathsf{sk} = (x, w))$.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$: parse $\mathsf{sk} = (x, w)$, $r \leftarrow\$ \mathsf{R}_\Sigma$, $a \leftarrow \Sigma.\mathsf{P}_0(x, w; r)$, $c \leftarrow H(x||a||m)$, $z \leftarrow \Sigma.\mathsf{P}_1(x, w, a, c, r)$ return $\sigma = (a, c, z)$.
- $0/1 \leftarrow \mathsf{Verify}(\mathsf{pk}, m, \sigma)$: parse $\mathsf{pk} = x$ and $\sigma = (a, c, z)$, return 1 iff $c = H(x||a||m)$ and $\mathsf{V}(x, a, c, z) = 1$.

Given an anamorphic tuple $\mathsf{a}\Sigma$ for Σ we can apply the same transformation to $(\mathsf{aP}_0, \mathsf{aP}_1, \mathsf{V})$ to get an anamorphic tuple ASIG for the signature scheme SIG above. In more details, ASIG is defined as follows:

- $(\mathsf{dk}, \mathsf{dk}_1, \ldots, \mathsf{dk}_N) \leftarrow \mathsf{dKeyGen}(1^\lambda, 1^t, 1^N)$: return $\mathsf{a}\Sigma.\mathsf{dKeyGen}(1^\lambda, 1^t, 1^N)$.

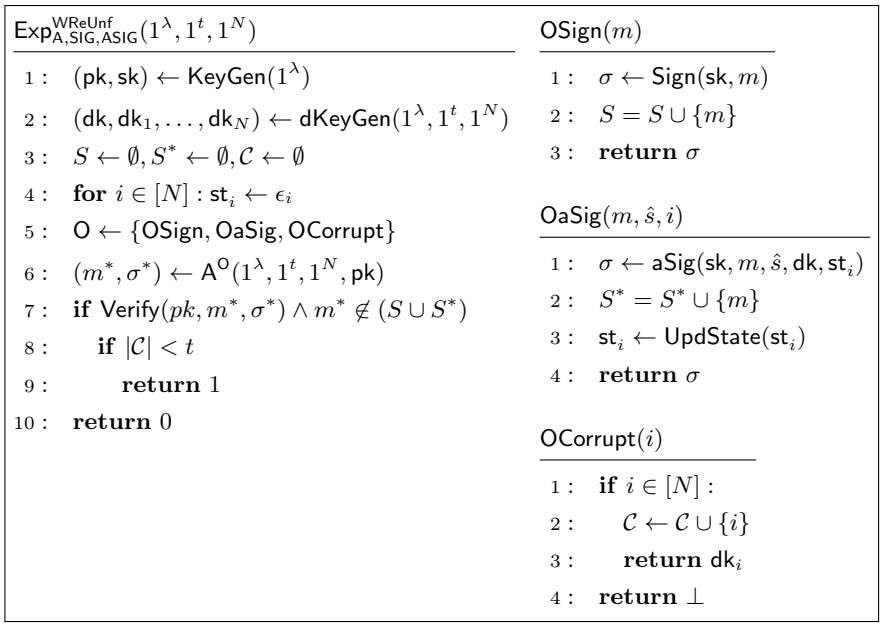

Fig. 11: Weak Recipient Unforgeability Game for Anamorphic Secret Sharing Signatures.

- $\sigma/\mathsf{fail} \leftarrow \mathsf{aSig}(\mathsf{sk}, m, \hat{s}, \mathsf{dk}, \mathsf{st})$: parse $\mathsf{sk} = (x, w)$, choose $r \leftarrow\!\!\$\ \mathsf{R}_\Sigma$, and compute $a \leftarrow \mathsf{aP}_0(x, w, \mathsf{dk}, \hat{s}, \mathsf{st}; r)$. If $a = \mathsf{fail}$, then return fail. Else, compute $c \leftarrow H(x||a||m)$, $z \leftarrow \mathsf{aP}_1(x, w, a, c, \mathsf{dk}, \hat{s}, \mathsf{st}, r)$ and return $\sigma = (a, c, z)$.
- $\mathsf{st}' \leftarrow \mathsf{UpdState}(\mathsf{st})$: return $\mathsf{a}\Sigma.\mathsf{UpdState}(\mathsf{st})$.
- $\hat{s}_i/\perp \leftarrow \mathsf{ExtShare}(\mathsf{pk}, \sigma, \mathsf{dk}_i, \mathsf{st})$: parse $\mathsf{pk} = x$ and $\sigma = (a, c, z)$ and return $\mathsf{a}\Sigma.\mathsf{ExtShare}(x, a, c, z, \mathsf{dk}_i, \mathsf{st})$.
- $\hat{s}/\perp/\mathsf{fail} \leftarrow \mathsf{Recon}(\mathcal{S}, \{\hat{s}_i\}_{i\in\mathcal{S}})$: return $\mathsf{a}\Sigma.\mathsf{Recon}(\mathcal{S}, \{\hat{s}_i\}_{i\in\mathcal{S}})$.

Theorem 3. *Let* $\mathsf{a}\Sigma$ *be an Anamorphic Secret Sharing tuple for* Σ *(Def. 8). Then,* ASIG *described above is an Anamorphic Secret Sharing Signature (Def. 13) for* SIG *in the ROM.*

Proof. We prove the above theorem via different lemmas, one per each required property. We omit correctness as it trivially follows from the correctness of $\mathsf{a}\Sigma$ (Def. 9).

Lemma 5 (Anamorphic Security). *If* $\mathsf{a}\Sigma$ *is anamorphic secure (Def. 10) w.r.t.* Σ*, then* ASIG *is anamorphic secure (Def. 15) w.r.t* SIG.

Proof. Let us assume for the sake of contradiction that there exists a PPT adversary A breaking anamorphism of ASIG with non-negligible advantage and consider the following PPT adversary B that breaks the anamorphism of $\mathsf{a}\Sigma$. B

receives a pair $(x, w) \leftarrow \mathcal{R}(1^\lambda)$ and interacts with an oracle O_b (Fig. 3) that is either prover P or the anamorphic prover aP. Upon receiving $(1^\lambda, 1^t, 1^N, x, w)$, B sets $(\mathsf{pk}, \mathsf{sk}) = (x, (x, w))$ and runs A on input $(1^\lambda, 1^t, 1^N, \mathsf{pk}, \mathsf{sk})$. Whenever A issues a query, B interacts with O_b to reply to A. In particular, whenever A queries $(i, \hat{s}, m)$, B interacts with its oracle O_b setting as challenge $c = H(x||a||m)$ where a was the first-round message B received from O_b, finally B gets a third-round message z from O_b. B returns the transcript of interaction (a, c, z) with O_b to A. When A stops, B returns the same output returned by A. Since B is perfectly simulating $\mathsf{Exp}^{\mathsf{AnaSig}}_{\mathsf{A,SIG,ASIG}}(1^\lambda, 1^t, 1^N)$ to A, B breaks the anamorphism of $\mathsf{a}\Sigma$ with the same advantage of A.

Lemma 6 (Adaptive Secrecy). *If $\mathsf{a}\Sigma$ satisfies adaptive secrecy (Def. 11), then ASIG satisfies adaptive secrecy (Def. 16).*

Proof. The proof is basically identical to the one above. Assume for the sake of contradiction that there exists a PPT adversary A breaking the adaptive secrecy of ASIG with non-negligible advantage, then we construct PPT B breaking the adaptive secrecy of $\mathsf{a}\Sigma$ with the same advantage. B runs A setting $\mathsf{pk} = x$ and replies to its queries by simply forwarding queries to its own oracles. The only consideration to make is that on receiving an OShare query (Fig. 10), B will set $c = H(x||a||m)$ in its interaction with aP. When A stops, B returns the same output returned by A. Since B is perfectly simulating $\mathsf{Exp}^{\mathsf{AdSecSig}}_{\mathsf{A,ASIG}}(1^\lambda, 1^t, 1^N)$ (Def. 4) to A, B breaks the adaptive secrecy of $\mathsf{a}\Sigma$ with the same advantage of A.

Lemma 7 (Weak Recipient Unforgeability). *If SIG is EUF-CMA-secure (Def.2) and $\mathsf{a}\Sigma$ satisfies simulatability (Def. 12), then ASIG is weak recipient unforgeable (Def. 17).*

Proof. The proof goes through the following hybrids.

$\mathcal{H}_0$: It is the weak recipient unforgeability game.

$\mathcal{H}_1$: It is identical to $\mathcal{H}_0$ except that the code of OaSig is modified by replacing the call to aSig in the following way. First it executes $\Sigma.\mathsf{P}_0(x, w; r)$ getting back the first-round message a, until $\mathsf{Sim}(x, \hat{s}, \mathsf{dk}_i, \mathsf{st}_i, a)$ returns 1, then it sets $c = H(x||a||m)$ and runs $\Sigma.\mathsf{P}_1(x, w, a, c, r)$ getting back z. It then constructs $\sigma = (a, c, z)$.

It is easy to show that $\mathcal{H}_0 \approx_c \mathcal{H}_1$ via a trivial reduction to simulatability (Def. 12). To conclude the proof, we now assume that A wins in $\mathcal{H}_1$ with non-negligible probability and use it to build an adversary B breaking the unforgeability of SIG. B runs A simulating $\mathcal{H}_1$ as follows:

- B generates the double keys and handles the corruption queries in the natural way.
- For every OSign (Fig. 11) query made by A, B replies by forwarding the query to the signing oracle of the unforgeability game of SIG (Fig. 2).

– For every OaSig (Fig. 11) query made by A, B keeps making signing queries to the signing oracle of the unforgeability game of SIG until B gets back a signature $\sigma = (a, c, z)$ s.t. $\mathsf{Sim}(x, \hat{s}, \mathsf{dk}_i, \mathsf{st}_i, a) = 1$. B then forwards such σ to a.
– Whenever A outputs a forgery, B outputs the same forgery.

To conclude notice that B perfectly simulates $\mathcal{H}_1$ to A, B is an admissible adversary whenever A is, and B wins whenever A does.

6 Conclusions

In this paper, we introduced the notion of Anamorphic Secret Sharing Signatures (ASSS) and showed a construction for signatures based on pre-image Σ-protocols combined with the Fiat-Shamir transform. Our ASSS introduces the possibility of sharing a secret anamorphically and, unlike the Schnorr-based AS of [31], does not reveal the secret key before the anamorphic communication takes place. A future direction involves studying ASSS for post-quantum signatures. A relevant case is CRYSTALS-Dilithium [22, 23] that was recently standardized by NIST [41]. In particular, CRYSTALS-Dilithium is based on Lyubashevky's Σ-protocol [35, 19] combined with the Fiat-Shamir with abort. Lyubashevky's Σ-protocol is a post-quantum Schnorr-like Σ-protocol with homomorphic properties and thus is a possible candidate for our techniques. However, our current analysis only applies to pre-image Σ-protocols paired with the standard Fiat-Shamir transform.

Acknowledgments

We thank the anonymous reviewers who reviewed this paper for their valuable comments. Gennaro Avitabile did part of the work while affiliated with the IMDEA Software Institute, Madrid, Spain. Vincenzo Botta did the majority of the work while affiliated with the Sapienza University of Rome, Rome, Italy. Daniele Friolo did part of the work while affiliated with the Sapienza University of Rome, Rome, Italy.

References

1. Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 418–433, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-46035-7_28.
2. Handan Kilinç Alper and Jeffrey Burdges. Two-round trip schnorr multi-signatures via delinearized witnesses. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 157–188, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland. doi:10.1007/978-3-030-84242-0_7.

3. Gennaro Avitabile, Vincenzo Botta, Emanuele Giunta, Marcin Mielniczuk, and Francesco Migliaro. The malice of ELFs: Practical anamorphic-resistant encryption without random oracles. In Joan Daemen and Emmanuel Thomé, editors, *Advances in Cryptology - EUROCRYPT 2026 - 45th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Rome, Italy, May 10-14, 2026, Proceedings, Part V*, volume 16545 of *Lecture Notes in Computer Science*, pages 303–333. Springer, 2026. doi:10.1007/978-3-032-25330-9_11.
4. Shalini Banerjee, Tapas Pal, Andy Rupp, and Daniel Slamanig. Simple asymmetric anamorphic encryption and signature using multi-message extensions. In Nadia Heninger and Mike Rosulek, editors, *Advances in Cryptology - CRYPTO 2026 - 46th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-20, 2026, Proceedings, Part X*, volume 16809 of *Lecture Notes in Computer Science*, pages 432–464. Springer, 2026. doi:10.1007/978-3-032-35428-0_14.
5. Fabio Banfi, Konstantin Gegier, Martin Hirt, Ueli Maurer, and Guilherme Rito. Anamorphic encryption, revisited. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part II*, volume 14652 of *Lecture Notes in Computer Science*, pages 3–32, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-58723-8_1.
6. Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 1–19, Santa Barbara, CA, USA, August 17–21, 2014. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-662-44371-2_1.
7. Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416, Saragossa, Spain, May 12–16, 1996. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-68339-9_34.
8. Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS$^+$ signature framework. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 2129–2146, London, UK, November 11–15, 2019. ACM Press. doi:10.1145/3319535.3363229.
9. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73, Interlaken, Switzerland, May 2–6, 2004. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-24676-3_4.
10. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55, Santa Barbara, CA, USA, August 15–19, 2004. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-540-28628-8_3.
11. Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-40041-4_23.

12. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532, Gold Coast, Australia, December 9–13, 2001. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-45682-1_30.
13. Davide Carnemolla, Dario Catalano, Emanuele Giunta, and Francesco Migliaro. Anamorphic resistant encryption: the good, the bad and the ugly. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part III*, volume 16002 of *Lecture Notes in Computer Science*, pages 472–503. Springer, 2025. doi:10.1007/978-3-032-01881-6_15.
14. Dario Catalano, Emanuele Giunta, and Francesco Migliaro. Anamorphic encryption: New constructions and homomorphic realizations. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part II*, volume 14652 of *Lecture Notes in Computer Science*, pages 33–62, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-58723-8_2.
15. Dario Catalano, Emanuele Giunta, and Francesco Migliaro. Limits of black-box anamorphic encryption. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part II*, volume 14921 of *Lecture Notes in Computer Science*, pages 352–383. Springer, 2024. doi:10.1007/978-3-031-68379-4_11.
16. Ronald Cramer. *Modular design of secure yet practical cryptographic protocols*. Phd thesis, University of Amsterdam, 1996.
17. Ronald Cramer and Ivan Damgård. Zero-knowledge proofs for finite field arithmetic; or: Can zero-knowledge be for free? In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 424–441, Santa Barbara, CA, USA, August 23–27, 1998. Springer Berlin Heidelberg, Germany. doi:10.1007/BFb0055745.
18. Amit Deo and Benoît Libert. Anamorphic signatures with dictator and recipient unforgeability for long messages. In Goichiro Hanaoka and Bo-Yin Yang, editors, *Advances in Cryptology - ASIACRYPT 2025 - 31st International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, VIC, Australia, December 8-12, 2025, Proceedings, Part VI*, volume 16250 of *Lecture Notes in Computer Science*, pages 370–401. Springer, 2025. doi:10.1007/978-981-95-5119-4_12.
19. Julien Devevey, Pouria Fallahpour, Alain Passelègue, and Damien Stehlé. A detailed analysis of Fiat-Shamir with aborts. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part V*, volume 14085 of *Lecture Notes in Computer Science*, pages 327–357, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-38554-4_11.
20. Xuan Thanh Do, Giuseppe Persiano, Duong Hieu Phan, and Moti Yung. Anamorphism beyond one-to-one messaging: Public-key with anamorphic broadcast mode. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part III*, volume 15603 of *Lecture Notes in Computer Science*, pages 429–455. Springer, 2025. doi:10.1007/978-3-031-91131-6_15.

21. Yevgeniy Dodis and Eli Goldin. Anamorphic-resistant encryption; or why the encryption debate is still alive. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2025, Proceedings, Part III*, volume 16002 of *Lecture Notes in Computer Science*, pages 440–471. Springer, 2025. doi:10.1007/978-3-032-01881-6_14.
22. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018. doi:10.13154/TCHES.V2018.I1.238-268.
23. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium. https://pq-crystals.org/dilithium/, 2024.
24. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, Santa Barbara, CA, USA, August 19–23, 1984. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-39568-7_2.
25. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-47721-7_12.
26. Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231, Santa Barbara, CA, USA, August 21–25, 1990. Springer, New York, USA. doi:10.1007/0-387-34799-2_16.
27. Joseph Jaeger and Roy Stracovsky. Dictators? Friends? Forgers. - breaking and fixing unforgeability definitions for anamorphic signature schemes. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024, Part II*, volume 15485 of *Lecture Notes in Computer Science*, pages 105–137, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore. doi:10.1007/978-981-96-0888-1_4.
28. Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 525–537, Toronto, ON, Canada, October 15–19, 2018. ACM Press. doi:10.1145/3243734.3243805.
29. Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson, Jr., and Colin O'Flynn, editors, *SAC 2020: 27th Annual International Workshop on Selected Areas in Cryptography*, volume 12804 of *Lecture Notes in Computer Science*, pages 34–65, Halifax, NS, Canada (Virtual Event), October 21-23, 2020. Springer, Cham, Switzerland. doi:10.1007/978-3-030-81652-0_2.
30. Mirek Kutylowski, Giuseppe Persiano, Duong Hieu Phan, Moti Yung, and Marcin Zawada. Anamorphic signatures: Secrecy from a dictator who only permits authentication! *IACR Cryptol. ePrint Arch.*, page 356, 2023. URL: https://eprint.iacr.org/2023/356.

31. Miroslaw Kutylowski, Giuseppe Persiano, Duong Hieu Phan, Moti Yung, and Marcin Zawada. Anamorphic signatures: Secrecy from a dictator who only permits authentication! In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part II*, volume 14082 of *Lecture Notes in Computer Science*, pages 759–790, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-38545-2_25.
32. Miroslaw Kutylowski, Giuseppe Persiano, Duong Hieu Phan, Moti Yung, and Marcin Zawada. The self-anti-censorship nature of encryption: On the prevalence of anamorphic cryptography. *Proc. Priv. Enhancing Technol.*, 2023(4):170–183, 2023. doi:10.56553/popets-2023-0104.
33. Leslie Lamport. Constructing digital signatures from a one way function. *Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, 1979*, 1979.
34. Benoît Libert, Damien Stehlé, and Radu Titiu. Adaptively secure distributed PRFs from LWE. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018: 16th Theory of Cryptography Conference, Part II*, volume 11240 of *Lecture Notes in Computer Science*, pages 391–421, Panaji, India, November 11–14, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-030-03810-6_15.
35. Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616, Tokyo, Japan, December 6–10, 2009. Springer Berlin Heidelberg, Germany. doi:10.1007/978-3-642-10366-7_35.
36. Ueli Maurer. Zero-knowledge proofs of knowledge for group homomorphisms. *Des. Codes Cryptogr.*, 77(2-3):663–676, 2015. doi:10.1007/S10623-015-0103-5.
37. Silvio Micali and Ray Sidney. A simple method for generating and sharing pseudo-random functions, with applications to clipper-like escrow systems. In Don Coppersmith, editor, *Advances in Cryptology – CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 185–196, Santa Barbara, CA, USA, August 27–31, 1995. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-44750-4_15.
38. Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and KDCs. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 327–346, Prague, Czech Republic, May 2–6, 1999. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-48910-X_23.
39. Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 33–43. ACM, 1989. doi:10.1145/73007.73011.
40. Jonas Nick, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 189–221, Virtual Event, August 16–20, 2021. Springer, Cham, Switzerland. doi:10.1007/978-3-030-84242-0_8.
41. NIST. Fips 204 - module-lattice-based digital signature standard. https://csrc.nist.gov/pubs/fips/204/final, 2024.
42. Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53, Santa Barbara, CA, USA, August 16–20, 1993. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-48071-4_3.

43. Giuseppe Persiano, Duong Hieu Phan, and Moti Yung. Anamorphic encryption: Private communication against a dictator. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 34–63, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland. doi:10.1007/978-3-031-07085-3_2.
44. Giuseppe Persiano, Duong Hieu Phan, and Moti Yung. Public-key anamorphism in (CCA-secure) public-key encryption and beyond. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part II*, volume 14921 of *Lecture Notes in Computer Science*, pages 422–455, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-68379-4_13.
45. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398, Saragossa, Spain, May 12–16, 1996. Springer Berlin Heidelberg, Germany. doi:10.1007/3-540-68339-9_33.
46. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252, Santa Barbara, CA, USA, August 20–24, 1990. Springer, New York, USA. doi:10.1007/0-387-34805-0_22.
47. Yi Wang, Rongmao Chen, Xinyi Huang, and Moti Yung. Sender-anamorphic encryption reformulated: Achieving robust and generic constructions. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part VI*, volume 14443 of *Lecture Notes in Computer Science*, pages 135–167, Guangzhou, China, December 4–8, 2023. Springer, Singapore, Singapore. doi:10.1007/978-981-99-8736-8_5.
48. Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer Berlin Heidelberg, Germany. doi:10.1007/11426639_7.