Anamorphic Signatures: Secrecy from a Dictator Who Only Permits Authentication!

Miroslaw Kutylowski, Giuseppe Persiano, Duong Hieu Phan, Moti Yung, Marcin Zawada

Anamorphic Signatures: Secrecy From a Dictator Who Only Permits Authentication!

Mirosław Kutyłowski
Wrocław University of Science and Technology and NASK - National Research Institute

Giuseppe Persiano
Università di Salerno and Google

Duong Hieu Phan
Telecom Paris, Institut Polytechnique de Paris

Moti Yung
Google and Columbia University

Marcin Zawada
Wrocław University of Science and Technology

Abstract

The goal of this research is to raise technical doubts regarding the usefulness of the repeated attempts by governments to curb Cryptography (aka the "Crypto Wars"), and argue that they, in fact, cause more damage than adding effective control. The notion of *Anamorphic Encryption* was presented in Eurocrypt'22 for a similar aim. There, despite the presence of a Dictator who possesses all keys and knows all messages, parties can arrange a hidden "*anamorphic*" message in an otherwise indistinguishable from regular ciphertexts (wrt the Dictator).

In this work, we postulate a stronger cryptographic control setting where encryption does not exist (or is neutralized) since all communication is passed through the Dictator in, essentially, cleartext mode (or otherwise, when secure channels to and from the Dictator are the only confidentiality mechanism). Messages are only authenticated to assure recipients of the identity of the sender. We ask whether security against the Dictator still exists, even under such a strict regime which allows only authentication (i.e., authenticated/ signed messages) to pass end-to-end, and where received messages are determined by/ known to the Dictator, and the Dictator also eventually gets all keys to verify compliance of past signing. To frustrate the Dictator, this authenticated message setting gives rise to the possible notion of anamorphic channels inside signature and authentication schemes, where parties attempt to send undetectable secure messages (or other values) using signature tags which are indistinguishable from regular tags. We define and present implementation of schemes for anamorphic signature and authentication; these are applicable to existing and standardized signature and authentication schemes which were designed independently of the notion of anamorphic messages. Further, some cornerstone constructions of the foundations of signatures, in fact, introduce anamorphism.

Contents

List of Theorems

1 Introduction

The notion of *anamorphic encryption* [PPY22] was introduced recently to deal with a very restricted cryptographic setting, where the dictatorial government requests to know all the keys. Yet, the notion allows parties to exploit the existing (and so severely debilitated) encrypted communication to exchange secret messages that remain hidden from the dictator while keeping the dictator's constraints.

Whereas in the above, parties are authorized to employ encryption, in this paper we consider a dictator who is much more restrictive with respect to encryption. In fact, in this new setting the use of encryption is totally prohibited except for private channels to and from the dictator.

The scenario we envision here assumes that all messages that are exchanged are sent via a central clearing house (i.e., through the dictator itself), which gets the message from the sender over a secure channel and forwards the message to the receiver on another secure channel. In this configuration encryption is completely neutralized, and, depending on the implementation of secure channels, may essentially not exist. Specifically, if the secure channel is physically protected (e.g., via quantum communication) there is no encryption indeed. If instead the secure channel is implemented with cryptographic tools, the ciphertext sent by the sender to the dictator is first decrypted and thus the plaintext is revealed, and the dictator and effectively re-encrypts it and eventually sends it to the receiver, who recovers the message. The two ciphertexts from the sender to the dictator and from the dictator to the receiver are completely independent, but for the fact that they conceal the same plaintext. Note that this configuration kills any steganographic channel or any anamorphic channel which the sender might have employed; simply, what the receiver gets is independent of the sender's ciphertext randomness, and this very restrictive configuration essentially overcomes the anamorphic encryption of [PPY22].

We note that in many configurations `https` communication is broken by a middlebox which inspects plaintext (for various safety purposes) away from the final user, so having intermediate channels between sender and receiver is not a completely outrageous and unusual configuration (see, for example, [DMS+17]).

Note, however, that if we simply trust the dictator as a built-in man-in-the-middle, then it has absolute power to control the messages sent, as he can send any message at anytime to anyone on behalf of anyone else. Of course, no one and certainly not a dictator can be trusted to not employ such an unlimited power, hence the above configuration on its own does not make sense. To minimally make sense, authenticity of the messages (against the dictator) must be assured. Then, if messages (even those re-encrypted by the dictator) are authenticated (signed) by the sender, then a message can include a header which includes: "*the sender name*," "*the receiver name*," "*time and date*," and even the "*hash of all prior messages in this exchange*." Using this extended message, the dictator is forced to send all messages from sender to receiver and the receiver can check the authenticity of a stream of messages. Now, secrecy (with respect to the dictator) is lost, but the dictator has to be faithful due to the authenticity assured by signing the stream of messages with a key they do not possess. So, in case we need to give up privacy, to have a minimal level of trust we need authenticated enhanced messages as above.

Assume we are then interested in reintroducing anamorphic communication that is hidden for the dictator, in spite of the loss of privacy in this scenario. Namely, we are interested in people obeying the dictator's rules while still interested in having an additional secret channel for communication that remains hidden from the dictator. In the above system, where the dictator is the clearing house for ciphertexts, signature schemes (or authentication and identification procedures)

are the only remaining cryptographic tool shared end-to-end between senders and receivers used to keep the dictator from impersonating users and creating fake messages. This gives rise to the question we tackle in this work:

- Can we have anamorphic channels inside signature schemes? and
- Can we implement such schemes and employ them in existing established signature and authentication mechanisms?

Further, when the dictator needs to analyze compliance of the parties with his demands, he gets to know (or even dictates) the messages, and he gets the signing key to check the validity of the messages regarding compliance. In fact, view this checking as done after the messaging is over (so the dictator cannot forge messages, but can eventually check the compliance of signed messages, say, when a new signing key is certified and the old key is revoked, and in any event these signing keys are merely for authentication and are not used for signing legally binding documents, etc.). It goes without saying that non-compliance has dire consequences for users who are caught!

This strict configuration is the subject of this work in which we answer the above in the *positive*; we further point at various issues of characterizing the anamorphic channels in this setting; and we develop methods to build such channels in various existing families of signature, authentication, and identification schemes. Moreover, we show how existing cornerstone method of the theory of signatures, which allow building signatures from basic simpler components also allow the introduction of anamorphism!

Obviously, the notion of *Anamorphic Cryptography* deals with possible abuses, misuses, and new uses of cryptosystems beyond their primary goals, which is an interesting take of cryptographic systems, after their primary goals are understood, formalized, and proved. Anamorphism is one way to view what side uses cryptographic systems enable (directly in the hands of the users themselves). This is a different way from and requires more than simply having a subliminal channel [Sim83], and even different from kleptographic abuse of cryptosystems which exploit such channels in a way hidden from the user [YY96] and also different from countermeasures against them (e.g., [RTYZ16]). Such unplanned readily available anamorphic uses constitute a note against controlling cryptography by governments attempting control of the primary use of cryptography. Next, we can see that anamorphic systems may further lead to other notions and applications.

An Implication: Watermarking and Anamorphic Signatures. Anamorphic channels obviously demonstrate the futility of direct control over encryption keys. Additionally, anamorphic methods have applications beyond subverting the dictator's limitations. In particular, let us concentrate on watermarking.

The aim of watermarking is to insert some information, a *mark*, in a digital object in order to be able to trace its origin. The marked object should be indistinguishable from an unmarked one and the mark should be difficult to remove. Hopper et al. [HMW07] first formalized the goal, and further work has been done on this important concept (for more recent work, see e.g. [GKM+19]). We note that watermarking can be seen as an anamorphic message to oneself (or to a designated checker), thus in fact, watermarking can be reduced from anamorphism.

Consider the concept of an anamorphic signatures as proposed in this work, and suppose that one party signs and wants to distribute a confidential document to a set of users and to guarantee authenticity of the document which is digitally signed. The signer is also afraid that the document will be leaked in its signed version as it could not be denied. For this reason the signer decides to

put in place a mechanism by which it would be possible to trace the leak. Anamorphic signatures come to help in this case. Indeed the party can sign the document for each receiver and insert a different anamorphic message in the signature. By extracting the anamorphic message from the leaked signed document, the signer can trace the leak. The leakage of the document without the signature is less dangerous as the document cannot be attributed to the signer with certainty (i.e., the document can be repudiated). As this example shows, watermarking is similar to an anamorphic message to oneself.

Using this method, one can insert different watermarking scheme (i.e., a different pseudorandom function to derive the mark and insert it in the anamorphic channel) to each of signing devices holding the same signing key, so the source of the signature can be determined from the inserted watermark.

We can employ the watermarking method to also protect the leakage or cryptanalysis of the signing key (since the one who gets/ cryptanalyzes the key cannot use a proper watermarking), hence the forgery will be caught.

Chaffing and Winnowing [Riv98] *Chaffing and Winnowing* is a technique that provides message confidentiality without using encryption or steganography. The two parties that want to establish a confidential channel share a secret *authentication* key. Each message is authenticated using the authentication key and then other fake messages (the *chaff*) are added with an incorrect authentication tag. In other words, the real messages (the *wheat*) are carried by correctly authenticated packets that can be winnowed by the legitimate receiver that knows the authentication key. A third party that sees all the messages flowing between the two parties cannot tell the wheat from the chaff without the authentication key. It is thus crucial for the confidentiality of the communication that dictator does not have the authentication key. The author of [Riv98] justifies this assumption as he considers a dictator seeking "access to all authentication keys as well, a truly frightening prospect." In this paper, we consider the frightening prospect in which the dictator has access to the authentication keys (in our setting, the signing keys) and we show that anamorphic signatures comes to the rescue.

1.1 Our Contribution

As said, our work is primarily geared toward developing anamorphism as a tool for answering issues that are raised in the "crypto wars." In the setting considered, encryption is completely neutralized and only message authentication through signatures (like TLS signing) is allowed. We show that, even in this extremely restricted setting, existing digital signature schemes and design techniques allow for private communication, despite a dictator that has the power to request all signing keys. In other words, to pursue the futile goal of disallowing private communication, the dictator must disallow not only encryption but also digital signatures with obvious dire consequences for digital communication: not only all communication is public, but nobody knows whom they are talking to, and, in fact, the dictator can impersonate anyone!

Defining anamorphic signatures. In Section 2, we formally define the concept of an *anamorphic signature scheme*. This is a special type of signature scheme that allows the signer to embed an *anamorphic* message in a signature. The anamorphic message can only be read by a set of trusted parties that have received a special *double key* from the signer. To every other party, a signature carrying an anamorphic message, an *anamorphic signature*, is indistinguishable from a

regular signature. And, this holds also w.r.t. parties, like the dictator, that can demand to see all secret material (including the signing keys).

Let us contrast anamorphic signatures with regular signatures. In the typical usage of a signature scheme, a signer Bob runs the *key generation* algorithm to obtain a *verification key* `svk` and a *signing key* `ssk`. Bob publishes the verification key `svk` in a public directory and keeps the signing key `ssk` private. Whenever Bob wants to sign a message `msg`, he runs the *signing* algorithm on input `ssk` and `msg` and obtains a signature that can be verified by running the *verification* algorithm on input the signature and `msg` and `svk`. An anamorphic signature scheme consists of three *anamorphic* algorithms: the *anamorphic key generation* algorithm, the *anamorphic signing* algorithm, and the *anamorphic decryption* algorithm. Their usage is slightly different. More specifically, the *anamorphic key generation* algorithm returns (`svk`, `ssk`), as in a regular signature scheme, and, additionally, a *double key* `dkey`. As before, `svk` is published and `ssk` is kept secret and the `dkey` is distributed by Bob to a trusted set of users with which Bob wants to establish an hidden communication channel. Whenever Bob wishes to secretly send a message `amsg`, the *anamorphic* message, to his trusted circle, Bob picks an innocent looking message `msg`, the *regular* message, and signs `msg` by running the *anamorphic signing* algorithm on input the two messages `msg` and `amsg`, the signing key `ssk` and the double key `dkey`. The signature `sig` produced has the following anamorphic property: it can be successfully verified using the verification key `svk` and the message `msg`; if instead it is given as input to the *anamorphic decryption* algorithm along with `dkey`, it returns the anamorphic message `amsg`. In other words, the same signature `sig` gives different results depending on the key used to operate on it. The security requirement posits that key pairs (`svk`, `ssk`) and signatures `sig` returned by the anamorphic algorithms are indistinguishable from those returned by the regular algorithms. And this holds also not only for parties that have access to the verification key `svk` but also for the dictator that has access to the signing key `ssk`.

Many-to-many vs. One-to-many. We consider two types of anamorphic signatures that implement two different types of communication channel hidden from the dictator: *many-to-many* and *one-to-many*.

By looking ahead, the formal definition of an anamorphic signature scheme (see Def. 2) does not make any claim about the unforgeability of the signature scheme by the parties that hold the double key (the circle of trusted users). Indeed, the possibility that that every member of the circle is able produce an anamorphic signature by relying only on the double key is not ruled-out by the definition. If this is the case, that is if knowledge of the double key allows to produce valid signatures, then we call the anamorphic signature *symmetric*. The name *symmetric* indicates that the set of users holding the double key can all produce signatures carrying anamorphic message; in other words, symmetric anamorphic signatures implement a *many-to-many* communication channel hidden from the dictator. As we shall see, this is the case for the anamorphic ElGamal signature scheme in which the double key contains the signing key. We note that the signer shares his secret key with the members of the group which is interested in a clandestine many-to-many secret communication under the mask of non-repudiated authentication (analogous to TLS authentication keys not serving as contract signing keys!). Interestingly, Davies [Dav83], already in 1983, was the first to consider giving up non-repudiation for other, more nefarious, reasons.

Another possible option is where the signature scheme remains unforgeable even if the double key is revealed. Such an anamorphic signature implements a *one-to-many* communication channel hidden from the dictator in which only the owner of the signing key can produce an anamorphic

signature. All other members can read anamorphic messages that come with an implicit origin authentication as they are part of a signature that can only be produced by the owner of the signing key. We call this type of anamorphic signatures *private* because the privacy of the signing key is preserved.

Constructions. The main message of this paper is that anamorphic signatures do exist, both symmetric and private, and they do not exist by accident but they systematically emerge from central design techniques.

To start, in Section 3, we show that two of the oldest and most well known signature schemes, the ElGamal signature schemes and the Schnorr signature scheme, are indeed *symmetric* anamorphic. These two signature schemes exemplify our general technique to obtain anamorphism. Specifically, we show that some of the randomness used by the signer can be recovered by the verifier and it can be used to hide the anamorphic message by means of an encryption with pseudorandom ciphertexts. The encryption key is the *double key* that is shared by the signer with a restricted group of trusted users.

However, anamorphism is not simply a special property enjoyed by a sparse group of construction from the literature. Indeed, in this paper, we make the stronger point of showing that two cornerstone design techniques for signature schemes give anamorphic signature schemes (for all their widely used instantiations). In other words, anamorphism seems to be a basic property of signature schemes that emerges naturally in general design techniques. More specifically, we prove the following.

1. In Section 4, we introduce the concept of an *anamorphic three-message public-coin* protocol and show that several instances of this important class of protocols are anamorphic. In other words, an anamorphic protocol can be used by the prover to send an hidden message to whomever has the double key and receives a transcript of the protocol. The message remnains hidden even with respect to a Dictator that has the secret information of the prover (i.e., the witness) and the dictator itself plays the role of the verifier.

   Having introduced the concept of an anamorphic three-message public-coin protocols, we give a sufficient condition for a protocol to be anamorphic and show that the most well-known protocols (e.g., the protocol for proving knowledge of discrete log and for graph isomorphism) do enjoy this property.

2. Besides being of its own independent interest, the concept of an *anamorphic three-message public-coin* protocol is important because it is at the basis of the construction of anamorphic signature schemes. That is, we look at the Fiat-Shamir heuristics that transforms three-message public-coin protocols into signature schemes that can be proved unforgeable in the Random Oracle Model. In Section 4.2, we prove that if the Fiat-Shamir heuristics is applied to an anamorphic three-message public-coin protocol the resulting signature scheme is *symmetric* anamorphic. In other words, Fiat-Shamir preserves anamorphism. This general result includes as special cases our warm-up examples, ElGamal and Schnorr, and it is obtained by using the encrypted anamorphic message as randomness in the signing process.

3. In Section 5, we present the notion of *private* anamorphism. In a private anamorphic signature scheme we require that knowledge of the *double-key* does not give signing capabilities. Private anamorphism provides a one-to-many channel hidden from the dictator in which there is one

designated sender, the one holding the signing key, that can send messages to the rest of the group.

4. In Section 7, we show that the Naor-Yung paradigm for constructing unforgeable many-time signature schemes from unforgeable one-time signature scheme gives private anamorphic signature schemes. More specifically, in Section 6 we introduce the notion of *weak anamorphism* for one-time signatures; we show that several well-known one-time signatures schemes, including Lamport's tagging system, enjoy weak anamorphism; and, finally, we prove that the Naor-Yung paradigm, when instantiated with a weakly anamorphic one-time signatures yields private anamorphic many-time signature schemes in the standard model. We extend this to the tree-based extension of the NY paradigm (see Appendix G).

   As a corollary, we obtain that private anamorphic signature schemes can be constructed in the standard model under the sole assumption of the existence of one-way functions. This is obtained by using Lamport's tagging systems (that can be proved to be one-time unforgeable by assuming only one-way functions) and upgrading it to many-time with the NY paradigm that only requires universal one-way hash functions (which, in turn, can be shown to exist by assuming one-way functions by using Rompel's construction).

5. Applications: since signature schemes are used as a component in encryption schemes, we demonstrate (in Section 8) anamorphic channels in encryption schemes which contain a signature component.

We have additional results in the appendix. In Appendix I we show private anamorphic schemes with standardized random-oracle based schemes (PSS RSA in particular). In Appendix J, we present an anamorphic scheme where the double key is generated jointly with the signing key (as it contains part of it) but still it cannot be used to forge signatures.

Bandwidth of the anamorphic channel. We stress that all our constructions have polynomial bandwidth. That is, the size of the anamorphic message is lower bounded by a polynomial in the size of the signature carrying it. The underlying technique of our construction is to replace the randomness that is used in the generation of the signing and verification keys (for the case of weak anamorphism) or in the generation of the signature (for the case of anamorphism) with a pseudorandom ciphertext encrypting the anamorphic message using the double key. The randomness must be extractable from the signature at verification, and depending on whether the signing key is needed for extraction or not the scheme gives rise to a different type of anamorphism.

Double-Key Distribution. As we have observed above, and as it will be made clear by our formal definitions, all security guarantees are void if the dictator obtains the double key. This is obvious as the double key allows to extract the anamorphic message from an anamorphic signature. In other words, an anamorphic signature symmetrically encrypts the anamorphic message.

Previous work on anamorphic encryption [PPY22] also considered the setting in which parties do not share any prior information; that is, the sender and the receiver of an anamorphic ciphertext do not share any private information. In other words, the anamorphic message is asymmetrically encrypted. This seems to be a very natural and important setting as it dispenses of the need of a secure channel for the distribution of the double key. It is thus natural to ask if an equivalent notion could be considered for anamorphic signatures as well. Note, however, that for anamorphic

signatures, the receiver of the anamorphic message coincides with the one with only the public information (the verification key). The fact that no double key is used in signatures would mean that everybody can read the anamorphic message, which is not what we want (as this will include the dictator!). In the context of encryption, things are different since the receiver of the anamorphic message is the owner of the secret information. In this case, having an anamorphic encryption scheme with no double key means that any user can send an anamorphic message exclussively to the owner of a public key.

2 Definitions and Models

In this section we introduce the notion of an *anamorphic signature* scheme and of a *symmetric anamorphic signature* scheme. We postpone the formal definition of a *private anamorphic signature* scheme to Section 5. In Appendix A, we review the concept of an unforgeable signature scheme.

Informally, an unforgeable signature scheme is anamorphic if there exists an *anamorphic* triplet of algorithms that allows to generate a key pair of signing/verification key along with a *double key*. The double key is shared by the signer with one or more selected trusted verifiers and it allows the signer to embed a secret message, called the *anamorphic message*, into a signature. The correctness condition is that the anamorphic message is readable by verifiers that have the double key. The security condition requires that no PPT dictator is able to distinguish whether the keys and the signatures are produced by the normal triplet of algorithm or by the anamorphic triplet. This indistinguishability implies that the anamorphic message is indistinguishable from a random field (hence it has semantic security) and the signature is unforgeable (were it not, it would give a way to distinguish the anamorphic one from the regular signature which is unforgeable!). We make this intuition formal in Theorem 1 and Theorem 2 in Section 2.1.

Let us start by defining the concept of an *anamorphic triplet* which will be used to define the concept of an *anamorphic signature scheme*.

Definition 1 (Anamorphic Triplet)**.** *An* anamorphic triplet $\mathsf{T} = (\mathsf{aKG}, \mathsf{aSig}, \mathsf{aDec})$ *consists of three PPT algorithms such that*

1. *the* anamorphic key generation algorithm aKG *on input security parameter* 1^λ *outputs the triplet* $(\mathtt{asvk}, \mathtt{assk}, \mathtt{dkey})$ *composed of the* anamorphic verification key $\mathtt{asvk}$*, the* anamorphic signing key $\mathtt{assk}$*, and the* double key $\mathtt{dkey}$*;*

2. *the* anamorphic signing algorithm aSig *takes as input a* regular *message* $\mathtt{msg}$*, an* anamorphic *message* $\mathtt{amsg}$*, an anamorphic signing key* $\mathtt{assk}$*, and a double key* $\mathtt{dkey}$ *and outputs an* anamorphic signature $\mathtt{asig}$*;*

3. *the* anamorphic decryption algorithm aDec *takes as input an anamorphic signature* $\mathtt{asig}$ *and a* double key $\mathtt{dkey}$ *and outputs an anamorphic message* $\mathtt{amsg}$*;*

and that satisfy the following correctness requirement

- *For every pair of messages* $(\mathtt{msg}, \mathtt{amsg})$*, the probability that* $\mathsf{aDec}(\mathtt{asig}, \mathtt{dkey}) \neq \mathtt{amsg}$ *is negligible, where* $(\mathtt{asvk}, \mathtt{assk}, \mathtt{dkey}) \leftarrow \mathsf{aKG}(1^\lambda)$ *and* $\mathtt{asig} \leftarrow \mathsf{aSig}(\mathtt{msg}, \mathtt{amsg}, \mathtt{assk}, \mathtt{dkey})$*, and the probability is taken over the random coin tosses of* aKG *and* aSig*.*

The concept of an anamorphic signature is formalized by means of two games: the *real game*, in which the adversary receives keys and signatures generated by the signature scheme S, and the *anamorphic game*, in which keys and signatures are output by the anamorphic triplet T. The definition requires the two games to be indistinguishable.

Definition 2 (Anamorphic Signature Scheme)**.** *An unforgeable signature scheme* $\mathsf{S} = (\mathsf{KG}, \mathsf{Sig}, \mathsf{Verify})$ *is* anamorphic *if there exists an anamorphic triplet* $\mathsf{T} = (\mathsf{aKG}, \mathsf{aSig}, \mathsf{aDec})$ *such that for every PPT dictator* $\mathcal{D}$ *there exists a negligible function* negl *such that*

$$|\text{Prob}\left[\mathsf{RealG}_{\mathsf{S},\mathcal{D}}(\lambda) = 1\right] - \text{Prob}\left[\mathsf{AnamorphicG}_{\mathsf{T},\mathcal{D}}(\lambda)\right]| \leq \mathsf{negl}(\lambda),$$

where

$\mathsf{RealG}_{\mathsf{S},\mathcal{D}}(\lambda)$

1. $(\mathtt{svk}, \mathtt{ssk}) \leftarrow \mathsf{KG}(1^\lambda)$;
2. return $\mathcal{D}^{\mathsf{Os}(\cdot,\cdot,\mathtt{ssk})}(\mathtt{svk}, \mathtt{ssk})$, where
   $\mathsf{Os}(\mathtt{msg}, \mathtt{amsg}, \mathtt{ssk}) = \mathsf{Sig}(\mathtt{msg}, \mathtt{ssk})$.

$\mathsf{AnamorphicG}_{\mathsf{T},\mathcal{D}}(\lambda)$

1. $(\mathtt{asvk}, \mathtt{assk}, \mathtt{dkey}) \leftarrow \mathsf{aKG}(1^\lambda)$;
2. return $\mathcal{D}^{\mathsf{Oa}(\cdot,\cdot,\mathtt{assk},\mathtt{dkey})}(\mathtt{asvk}, \mathtt{assk})$, where
   $\mathsf{Oa}(\mathtt{msg}, \mathtt{amsg}, \mathtt{assk}, \mathtt{dkey}) = \mathsf{aSig}(\mathtt{msg}, \mathtt{amsg}, \mathtt{assk}, \mathtt{dkey})$.

In the next section, we consider two consequences of anamorphism. Specifically, the anamorphic message is IND-CPA secure with respect to a party, like the dictator, that does not have the double key; and that if the keys are generated anamorphically, the signature scheme is unforgeable with respect to parties that do not have the signing key and the double key. As we shall see, whether the signature scheme remains unforgeable with respect to parties that have the double key is independent from the security guarantee of anamorphism. In Section 2.2, we define the notion of a *symmetric anamorphic signature scheme*, an anamorphic signature scheme for which the double key enables signing. In Section 5, we define the notion of a *private anamorphic signature scheme*, an anamorphic signature scheme that remains unforgeable even if the double key is available.

2.1 Security of the associated schemes

An anamorphic signature scheme is naturally associated with two schemes: the signature scheme in which keys and signatures are anamorphic (that is, computed by the anamorphic triplet); and the symmetric encryption scheme that hides the anamorphic message by means of the double key. Quite obviously, one would like the signature to be unforgeable and the anamorphic message to be IND-CPA secure. Indeed, the purpose of an anamorphic scheme is to hide the anamorphic message. The formal definition of anamorphic signatures (Definition 2) makes no explicit security guarantee neither about the IND-CPA security of the anamorphic message nor about the security of the signature scheme when the verification key is anamorphic. Note though that the indistinguishability of the two games as required by Definition 2 means in particular that the mere existence

of an anamorphic message is indistinguishable from its non-existence which, intuitively, should be sufficient for semantic security of the anamorphic message. Also, anamorphism should imply that forging with respect to an anamorphic verification key is as hard as forging messages in the original signature (as forging in both cases is without the key(s) and if it becomes easy for the anamorphic version, due to the indistinguishability it is easy for the original scheme (a contradiction)). We next give formal proofs for the two intuitions above.

We start by proving that for an anamorphic signature scheme the anamorphic message is hidden from a party that has access to the signing and verification keys but not, obviously, to the double key. This is made formal by means of the IND-CPA game for anamorphic messages, that we call AcpaG. This game is the adaptation of the IND-CPA game (see Appendix B)o the anamorphic signature setting in which the anamorphic message is "encrypted" by computing a signature of the regular message. This is reflected in the working of the encryption oracle Oe and of the challenge oracle Oc. In game AcpaG the adversary $\mathcal{A}$ has access to the anamorphic keys, both verification and signing, and can ask to see anamorphic signatures for pairs $(\mathtt{msg}, \mathtt{amsg})$ of regular/anamorphic messages of their choice; that is, $\mathcal{A}$ can ask for "encryptions" of anamorphic messages of their choice just as in chosen plaintext attack. Once ready, $\mathcal{A}$ outputs the regular message $\mathtt{msg}$ and the pair of anamorphic messages $(\mathtt{amsg}_0, \mathtt{amsg}_1)$ on which they want to be tested. Finally, one of the two anamorphic messages is encrypted and $\mathcal{A}$ should not be able to distinguish which one has been used to produce the anamorphic signature. Below we define the anamorphic IND-CPA game AcpaG for an anamorphic triplet $\mathsf{T} = (\mathsf{aKG}, \mathsf{aSig}, \mathsf{aDec})$, PPT adversary $\mathcal{A}$, and bit $\beta \in \{0, 1\}$.

$\mathsf{AcpaG}^{\beta}_{\mathsf{T},\mathcal{A}}(\lambda)$

1. $(\mathtt{asvk}, \mathtt{assk}, \mathtt{dkey}) \leftarrow \mathsf{aKG}(1^{\lambda})$;
2. $(\mathtt{msg}, \mathtt{amsg}_0, \mathtt{amsg}_1, \mathtt{st}) \leftarrow \mathcal{A}^{\mathsf{Oe}(\cdot,\cdot,\mathtt{assk},\mathtt{dkey})}(\mathtt{asvk}, \mathtt{assk})$;
3. $\mathtt{asig} = \mathsf{Oc}^{\beta}(\mathtt{msg}, \mathtt{amsg}_0, \mathtt{amsg}_1, \mathtt{assk}, \mathtt{dkey})$;
4. return $\mathcal{A}^{\mathsf{Oe}(\cdot,\cdot,\mathtt{assk},\mathtt{dkey})}(\mathtt{asig}, \mathtt{st})$;

where

- $\mathsf{Oe}(\mathtt{msg}, \mathtt{amsg}, \mathtt{assk}, \mathtt{dkey}) = \mathsf{aSig}(\mathtt{msg}, \mathtt{amsg}, \mathtt{assk}, \mathtt{dkey})$;
- $\mathsf{Oc}^{\beta}(\mathtt{msg}, \mathtt{amsg}_0, \mathtt{amsg}_1, \mathtt{assk}, \mathtt{dkey}) = \mathsf{aSig}(\mathtt{msg}, \mathtt{amsg}_{\beta}, \mathtt{assk}, \mathtt{dkey})$;

The following theorem holds.

Theorem 1. *Let* S *be anamorphic signature scheme and let* T *be the associated anamorphic triplet. Then for all PPT adversaries* $\mathcal{A}$ *we have*

$$\left|\text{Prob}\left[\mathsf{AcpaG}^{0}_{\mathsf{T},\mathcal{A}}(\lambda) = 1\right] - \text{Prob}\left[\mathsf{AcpaG}^{1}_{\mathsf{T},\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Proof. Towards a contradiction, suppose there exists a PPT adversary $\mathcal{A}$ such that

$$\text{Prob}\left[\mathsf{AcpaG}^{1}_{\mathsf{T},\mathcal{A}}(\lambda) = 1\right] \geq \text{Prob}\left[\mathsf{AcpaG}^{0}_{\mathsf{T},\mathcal{A}}(\lambda) = 1\right] + 1/\mathsf{poly}(\lambda)$$

for some polynomial $\mathsf{poly}(\cdot)$. We construct the following dictator $\mathcal{D}$ that distinguishes games RealG and $\mathsf{AnamorphicG}$. Dictator $\mathcal{D}$ receives a pair of keys $(\mathtt{svk}, \mathtt{ssk})$ (that is either regular or anamorphic) and has access to an oracle $O(\cdot,\cdot)$ (that is either Os or Oa) and uses $\mathcal{A}$ in the following way. $\mathcal{D}$ runs $\mathcal{A}$ on input $(\mathtt{svk}, \mathtt{ssk})$ and replies to queries $(\mathtt{msg}, \mathtt{amsg})$ by returning $O(\mathtt{msg}, \mathtt{amsg})$. When $\mathcal{A}$ return $(\mathtt{msg}, \mathtt{amsg}_0, \mathtt{amsg}_1)$, $\mathcal{D}$ randomly selects β from $\{0,1\}$ and sets $\mathtt{ct} = O(\mathtt{msg}, \mathtt{amsg}_\beta)$. Then $\mathcal{D}$ runs $\mathcal{A}$ on input $\mathtt{ct}$ and replies to $\mathcal{A}$'s oracle queries as before. Finally, $\mathcal{A}$ returns bit η and $\mathcal{D}$ outputs 1 iff $\beta = \eta$.

Let us denote by $p_{\alpha,\beta}$ the probability that $\mathcal{A}$ outputs α in game AcpaG^β. By hypothesis we have that $p_{11} \geq p_{10} + 1/\mathsf{poly}(\lambda)$. Suppose that $\mathcal{D}$ is playing the anamorphic game and thus the pair of keys received in input is anamorphic (i.e., output by aKG) and $O = \mathsf{Oa}$. Then observe that $\mathcal{D}$ is providing $\mathcal{A}$ with a view from AcpaG^β and therefore the probability that $\mathcal{D}$ outputs 1 is

$$\frac{1}{2}(p_{11} + p_{00}) = \frac{1}{2}(p_{11} + 1 - p_{10}) \geq \frac{1}{2} + \frac{1}{2 \cdot \mathsf{poly}(\lambda)}.$$

Suppose instead that $\mathcal{D}$ is playing the real game and thus the pair of keys receives in input is regular (i.e., output by KG) and $O = \mathsf{Os}$. Then in this case, the view of $\mathcal{A}$ is independent of β as Os ignores its second argument. Therefore in this case $\mathcal{D}$ outputs 1 with at most probability $1/2$.

We can thus conclude that $\mathcal{D}$ violates the anamorphism of S. Contradiction. □

Next we define the concept of the *associated* signature scheme. In the associated signature scheme, the pair of signing and verification keys are anamorphically generated and the regular signing and verifying algorithms are used for the other operations. Then we show that the associated signature scheme is unforgeable as well.

Definition 3. *Let* $\mathsf{S} = (\mathsf{KG}, \mathsf{Sig}, \mathsf{Verify})$ *be an* anamorphic *signature scheme and let* $(\mathsf{aKG}, \mathsf{aSig}, \mathsf{aDec})$ *be the associated anamorphic triplet. The signature scheme* associated *with* S *is* $\mathsf{S}^\star = (\mathsf{aKG}^\star, \mathsf{aSig}, \mathsf{Verify})$*, where* $\mathsf{aKG}^\star(1^\lambda)$ *is the algorithm that obtains* $(\mathtt{asvk}, \mathtt{assk}, \mathtt{dkey}) \leftarrow \mathsf{aKG}(1^\lambda)$ *and outputs* $(\mathtt{asvk}, \mathtt{assk})$.

Theorem 2. *Let* $\mathsf{S} = (\mathsf{KG}, \mathsf{Sig}, \mathsf{Verify})$ *an* anamorphic *signature scheme and let* $(\mathsf{aKG}, \mathsf{aSig}, \mathsf{aDec})$ *be its anamorphic triplet. Then the associated signature scheme* $\mathsf{S}^\star = (\mathsf{aKG}^\star, \mathsf{aSig}, \mathsf{Verify})$ *is a secure signature scheme.*

Proof. For sake of contradiction, suppose that there exists a PPT adversary $\mathcal{A}$ such that game $\mathsf{sigG}_{\mathsf{S}^\star}$ has a non-negligible probability of outputting 1 and consider the following dictator $\mathcal{D}$ that receives as input a pair of keys $(\mathtt{svk}, \mathtt{ssk})$ and runs $\mathcal{A}$ on input $\mathtt{svk}$. Whenever $\mathcal{A}$ issues a query for message m, $\mathcal{D}$ replies by returning the pair $(m, \mathsf{Sig}(m, \mathtt{ssk}))$. $\mathcal{D}$ outputs 1 if and only if $\mathcal{A}$ outputs a pair $(\mathtt{msg}, \mathtt{sig})$ that is accepted by Verify and that was not returned as a reply to one of $\mathcal{A}$'s queries.

Now observe that if the input pair $(\mathtt{svk}, \mathtt{ssk})$ is anamorphic, that is it is output by $\mathsf{aKG}^\star$, then $\mathcal{D}$ is actually playing game $\mathsf{AnamorphicG}$ while simulating game $\mathsf{sigG}_{\mathsf{S}^\star}$ for $\mathcal{A}$. Thus, by our assumption, the probability that $\mathcal{D}$ outputs 1 is non-negligible. On the other hand, if the input pair $(\mathtt{svk}, \mathtt{ssk})$ is output by KG, then $\mathcal{D}$ is actually playing game RealG while simulating game $\mathsf{sigG}_{\mathsf{S}}$ for $\mathcal{A}$. Since S is secure the probability that $\mathcal{D}$ outputs 1 is negligible.

Therefore, dictator $\mathcal{D}$ described above contradicts the anamorphism of S. □

2.2 Symmetric Anamorphism

We next define the concept of a *symmetric* anamorphic signature scheme. As we discussed in the introduction, in a symmetric anamorphic scheme, the double key allows to produce signatures and not just to extract the anamorphic message from a signature. This implies that every user that has the double key can send anamorphic messages that will be read by all other members of the trusted circle, thus implementing a many-to-many communication channel hidden from the dictator. The definition we present below requires the existence of an extraction algorithm that extracts the signing key from the double. This is not the most general definition but all symmetric schemes we present satisfy this definition.

Definition 4 (Symmetric Anamorphic Triplet)**.** *An anamorphic triplet* $\mathsf{T} = (\mathsf{aKG}, \mathsf{aSig}, \mathsf{aDec})$ *is* symmetric *if there exists an efficient extraction algorithm* $\mathsf{Extract}$ *such that, for* $(\mathtt{svk}, \mathtt{ssk}, \mathtt{dkey}) \leftarrow \mathsf{aKG}(1^\lambda)$*, we have that* $\mathsf{Extract}(\mathtt{svk}, \mathtt{dkey}) = \mathtt{ssk}$ *except with probability negligible in* λ*, over the coin tosses of* aKG*.*

Definition 5 (Symmetric Anamorphic Signature Scheme)**.** *An anamorphic signature scheme* $\mathsf{S} = (\mathsf{KG}, \mathsf{Sig}, \mathsf{Verify})$ *with anamorphic symmetric triplet* $\mathsf{T} = (\mathsf{aKG}, \mathsf{aSig}, \mathsf{aDec})$ *is* symmetric *if* T *is symmetric.*

3 First Examples of Anamorphic Signatures

In this section we give the first concrete examples of anamorphic signature schemes by showing that some of the most well-known signature schemes, namely the Boneh-Boyen, the ElGamal, and the Schnorr signature, are symmetric anamorphic.

We start by reviewing the concept of a symmetric encryption scheme with pseudo-random ciphertexts.

3.1 Pseudorandom Ciphertexts

In this section, we review the notion of a symmetric encryption scheme $\mathsf{prE} = (\mathsf{prKG}, \mathsf{prEnc}, \mathsf{prDec})$ with *pseudorandom ciphertexts* by means the following game $\mathsf{PRCtG}^\beta_{\mathsf{prE},\mathcal{A}}$, where $\beta \in \{0,1\}$, prE is a symmetric encryption scheme, and $\mathcal{A}$ is a PPT adversary. We assume that prE for security parameter λ encrypts $n(\lambda)$-bit plaintexts into $\ell(\lambda)$-bit ciphertexts.

$\mathsf{PRCtG}^\beta_{\mathsf{prE},\mathcal{A}}(\lambda)$

1. Set $K \leftarrow \mathsf{prKG}(1^\lambda)$
2. Return $\mathcal{A}^{\mathsf{OPr}^\beta(K,\cdot)}()$, where, for $n(\lambda)$-bit plaintext $\mathtt{msg}$,

   $\mathsf{OPr}^0(K, \mathtt{msg})$ returns a randomly selected $\ell(\lambda)$-bit string;

   $\mathsf{OPr}^1(K, \mathtt{msg}) = \mathsf{prEnc}(K, \mathtt{msg})$.

Definition 6. *Let* $\mathsf{prE} = (\mathsf{prKG}, \mathsf{prEnc}, \mathsf{prDec})$ *be an IND-CPA symmetric encryption scheme. We say that* prE *has* pseudorandom ciphertexts *if for every PPT adversary* $\mathcal{A}$ *we have*

$$\left|\text{Prob}\left[\mathsf{PRCtG}^0_{\mathsf{prE},\mathcal{A}}(\lambda) = 1\right] - \text{Prob}\left[\mathsf{PRCtG}^1_{\mathsf{prE},\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Symmetric encryption schemes with pseudorandom ciphertexts can be constructed starting from one-way functions. More specifically, let $m(\cdot)$ and $n(\cdot)$ be two polynomials and let $\mathcal{F} : \{0,1\}^\lambda \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}$ be a pseudorandom function with λ-bit seed, $m(\lambda)$-bit argument and $n(\lambda)$-bit output. Consider the encryption scheme in which the key generation algorithm, for security parameter λ, randomly selects a λ-bit key K and the encryption algorithm, on input $n(\lambda)$-bit message $\mathtt{msg}$, randomly selects $r \leftarrow \{0,1\}^{m(\lambda)}$ and outputs the pair $\mathtt{ct} = (r, \mathtt{msg} \oplus \mathcal{F}(K, r))$ of length $m(\lambda) + n(\lambda)$. It is easy to see that the scheme is IND-CPA secure and that the ciphertext $\mathtt{ct}$ is indistinguishable from a randomly selected string of length $m(\lambda) + n(\lambda)$ bits.

Theorem 3. *Assuming existence of one-way functions, there exists and IND-CPA secure symmetric encryption scheme with* pseudorandom ciphertexts*.*

3.2 Boneh-Boyen Signatures

In this section we show that the Boneh-Boyen signature scheme [BB08, BB04] is anamorphic. The proof of anamorphism will also exemplify our main technique for showing anamorphism. Specifically, we identify randomness in the signature that can be set by the signer and extracted by the verifier. The anamorphic signing algorithm replaces the randomness with a ciphertext carrying the anamorphic message. To ensure that the dictator does not detect the existence of the anamorphic message, an encryption scheme with pseudorandom ciphertexts (see Section 3.1).

We start by defining the concept of a bilinear group generator and then we describe the Boneh-Boyen signature scheme.

Definition 7. *A* bilinear group generator $\mathcal{G}$ *is a Probabilistic Polynomial Time (PPT) algorithm that, on input* 1^λ*, outputs a* λ*-bit prime* p*, the descriptions of cyclic groups* G_1, G_2, G_T *of order* p *and a bilinear map* $\mathsf{e} : G_1 \times G_2 \to G_T$ *so that:*

- *For all* $u \in G_1, v \in G_2$ *and integers* a *and* b*,*

$$\mathsf{e}(u^a, v^b) = \mathsf{e}(u, v)^{a \cdot b}.$$

We assume that the descriptions of a group allows to randomly sample elements of the group and to efficiently perform the group operation.

The Boneh-Boyen signature scheme $\mathsf{BB} = (\mathsf{bbKG}, \mathsf{bbSig}, \mathsf{bbVerify})$ is described as follows.

1. The key generation algorithm bbKG, on input security parameter 1^λ, randomly selects $(G_1, G_2, G_T, \mathsf{e}, p) \leftarrow \mathcal{G}(1^\lambda)$. Then the algorithm randomly selects generators $g_1 \in G_1$ and $g_2 \in G_2$ and random $x, y \leftarrow \mathbb{Z}_p$ and sets $z = \mathsf{e}(g_1, g_2)$, $u = g_2^x$ and $v = g_2^y$. The verification key is the tuple $\mathtt{svk} = (g_1, g_2, u, v, z)$ and the signing key is the triple $\mathtt{ssk} = (g_1, x, y)$.

2. The signing algorithm bbSig, on input a signing key $\mathtt{ssk} = (g_1, x, y)$ and a message $\mathtt{msg} \in \mathbb{Z}_p$, randomly selects $r \leftarrow \mathbb{Z}_p$. If $r = -(x + m)/y$ then the signing algorithm fails. Otherwise it sets $\sigma = g_1^{1/(x+m+yr)}$. The signature is the pair $\mathtt{sig} = (r, \sigma)$. Note that the probability of failure is negligible in the security parameter λ.

3. The verification algorithm $\mathsf{bbVerify}$ on input a signature $\mathtt{sig} = (r, \sigma)$ checks that

$$\mathsf{e}(\sigma, u \cdot g_2^m \cdot v^r) = z.$$

Definition 8. *The* Strong Diffie-Hellman assumption *for bilinear group generator $\mathcal{G}$ posits that for every PPT algorithm $\mathcal{A}$ and every polynomially bounded $q = q(\lambda)$ the following probability*

$$\text{Prob}\left[(G_1, G_2, G_T, \mathsf{e}, p) \leftarrow \mathcal{G}(1^\lambda); g_1 \leftarrow G_1; g_2 \leftarrow G_2; x \leftarrow \mathbb{Z}_p : \mathcal{A}(g_1, g_1^x, \ldots, g_1^{x^q}, g_2, g_2^x) = (c, g_1^{1/(c+x)})\right]$$

is negligible in λ.

The following theorem holds.

Theorem 4 ([BB08, BB04])**.** *The Boneh-Boyen signature scheme is unforgeable under the Strong Diffie-Hellman assumption.*

To prove that the Boneh-Boyen signature scheme is actually an anamorphic signature scheme we make the following observation. The r component of a signature is a randomly selected element of $\mathbb{Z}_p$. In an anamorphic signature instead r will be the ciphertext of the anamorphic message $\mathtt{amsg}$ computed using an encryption scheme with pseudorandom ciphertexts. More precisely, let $\mathsf{prE} = (\mathsf{prKG}, \mathsf{prEnc}, \mathsf{prDec})$ be an encryption scheme, with pseudorandom ciphertexts that for security parameter λ, encrypts $\lambda/2$-bit plaintexts into λ-bit ciphertexts. For example, this can be obtained by using the scheme described in Section 3.1 with $n(\lambda) = m(\lambda) = \lambda/2$. Let us consider the following triplet $\mathsf{T} = (\mathsf{abbKG}, \mathsf{abbSig}, \mathsf{aDec})$.

1. The anamorphic key generation algorithm abbKG, on input security parameter 1^λ, computes $(\mathtt{svk}, \mathtt{ssk}) \leftarrow \mathsf{bbKG}(1^\lambda)$ and randomly selectes encryption key $K \leftarrow \mathsf{prKG}(1^\lambda)$. The anamorphic verification key is $\mathtt{asvk} := \mathtt{svk}$, the anamorphic signing key is $\mathtt{assk} := \mathtt{ssk}$, and the double key is $\mathtt{dkey} := K$.

2. The anamorphic signing algorithm abbSig on input the regular message $\mathtt{msg} \in \mathbb{Z}_p$, the anamorphic message $\mathtt{amsg} \in \{0,1\}^{\lambda/2}$, the anamorphic signing key $\mathtt{ssk} = (g_1, x, y)$ and the double key $\mathtt{dkey} = K$ proceeds as follows. First, it encrypts $\mathtt{amsg}$ by running $\mathtt{act} = \mathsf{prEnc}(K, \mathtt{amsg})$ until $\mathtt{act} \in \mathbb{Z}_p$. Then, it sets $r = \mathtt{act}$ and if $r = -(x+m)/y$ then the algorithm fails. Otherwise, it computes σ as $\sigma = g_1^{1/(x+m+yr)}$ and it outputs $\mathtt{asig} = (r, \sigma)$.

3. The anamorphic decryption algorithm aDec, on input anamorphic signature $\mathtt{asig} = (r, \sigma)$, double key $\mathtt{dkey} = K$ recovers the anamorphic message $\mathtt{amsg}$ as $\mathtt{amsg} = \mathsf{prDec}(K, r)$.

We are now ready to prove that the Boneh-Boyen signature scheme is anamorphic.

Theorem 5. *Under the Strong Diffie-Hellman assumption, the Boneh-Boyen signature scheme is anamorphic.*

Proof. First of all, observe that, under the Strong Diffie-Hellman assumption, the Boneh-Boyen signature is unforgeable by Theorem 5. Moreover, under the Strong Diffie-Hellman assumption there exists a one-way function and thus we can construct and encryption scheme with pseudorandom ciphertexts.

Let us now consider the triplet $\mathsf{T} = (\mathsf{abbKG}, \mathsf{abbSig}, \mathsf{aDec})$ described above. We start by observing that the pair of verification and signing key output by the anamorphic key generation algorithm abbKG has the same distribution as the pair output by bbKG.

Suppose, for sake of contradiction, that there exists a PPT dictator $\mathcal{D}$ and a polynomial $\mathsf{poly}(\cdot)$ for which

$$\left|\text{Prob}\left[\mathsf{RealG}_{\mathsf{BB},\mathcal{D}}(\lambda)=1\right]-\text{Prob}\left[\mathsf{AnamorphicG}_{\mathsf{T},\mathcal{D}}(\lambda)=1\right]\right|\geq 1/\mathsf{poly}(\lambda)$$

and consider the following PPT adversary $\mathcal{A}$ for the pseudorandomness of the ciphertext of prE. $\mathcal{A}$ has access to an oracle O that, on input $\lambda/2$ bits returns λ bits. $\mathcal{A}$ runs $\mathcal{D}$ on input randomly generated $(\mathtt{svk},\mathtt{ssk})\leftarrow\mathsf{bbKG}(1^\lambda)$. Whenever $\mathcal{D}$ issues a query for $(\mathtt{msg},\mathtt{amsg})$, $\mathcal{A}$ sets $r=O(\mathtt{amsg})$. If $r=-(x+m)/y$ then $\mathcal{A}$ stops and returns a random bit. Otherwise, $\mathcal{A}$ sets $\sigma=g_1^{1/(x+m+yr)}$. and returns (r,σ) to $\mathcal{D}$.

Now observe that if $\mathcal{A}$ is playing the $\mathsf{PRCtG}^0_{\mathsf{prE},\mathcal{A}}$ game then r is random and thus $\mathcal{A}$ is simulating the real game to $\mathcal{D}$. On the other hand, if $\mathcal{A}$ is playing the $\mathsf{PRCtG}^1_{\mathsf{prE},\mathcal{A}}$ game then r is a ciphertext carrying $\mathtt{amsg}$ and thus $\mathcal{A}$ is simulating the anamorphic game to $\mathcal{D}$. Therefore we have that

$$\begin{aligned}&\left|\text{Prob}\left[\mathsf{PRCtG}^0_{\mathsf{prE},\mathcal{A}}(\lambda)=1\right]-\text{Prob}\left[\mathsf{PRCtG}^1_{\mathsf{prE},\mathcal{A}}(\lambda)=1\right]\right|=\\&\left|\text{Prob}\left[\mathsf{RealG}_{\mathsf{BB},\mathcal{D}}(\lambda)=1\right]-\text{Prob}\left[\mathsf{AnamorphicG}_{\mathsf{T},\mathcal{D}}(\lambda)=1\right]\right|\geq 1/\mathsf{poly}(\lambda)\end{aligned}$$

Contradiction. □

3.3 ElGamal Signatures

In Figure 1, we describe the ElGamal [ElG84] signature scheme $\mathsf{ElS}=(\mathsf{ElKG},\mathsf{ElSig},\mathsf{ElVerify})$ as modified by [PS96]. The signature scheme is proved secure in the Random Oracle Model under the assumption of hardness of the discrete logarithm problem. A variation of this scheme, called the DSA, constitutes the Digital Signature Standard [DSS13] and adapting our proof of anamorphism for ElGamal to DSA is straightforward.

1. The key generation algorithm ElKG, on input security parameter 1^λ, randomly selects (the description of) a cyclic group $\mathbb{G}$ of prime order p, for p of length $\Theta(\lambda)$, a generator g for $\mathbb{G}$, and a hash function $H:\{0,1\}^\star\times\mathbb{G}\to\mathbb{Z}_p$. In addition the algorithm randomly selects $x\leftarrow\mathbb{Z}_p^\star$ and sets $y=g^x$.

   Finally, the algorithm outputs the verification key $\mathtt{svk}:=(\mathbb{G},g,H,y)$ and the signing key $\mathtt{ssk}:=(\mathbb{G},g,H,x)$.

2. The signing algorithm ElSig, on input message m and signing key $\mathtt{ssk}=(\mathbb{G},g,H,x)$, outputs $\mathtt{sig}=(r,s)$ computed as follows.

   Randomly select $\kappa\leftarrow\mathbb{Z}_p$, set $r:=g^\kappa$ and $s:=(H(m,r)-x\cdot r)/\kappa \bmod (p-1)$.

3. The verification algorithm $\mathsf{ElVerify}$, on input message m, signature (r,s), and verification key $\mathtt{svk}$, checks if $g^{H(m,r)}=y^r\cdot r^s$.

Figure 1: The ElGamal signature scheme ElS.

To show that the ElGamal signature scheme is anamorphic, we use the technique of hiding the anamorphic message in the randomness used to produce the signature by means of an encryption

scheme with pseudorandom ciphertexts. These are special IND-CPA symmetric encryption schemes whose ciphertexts are pseudorandom; that is, indistinguishable from truly random strings of the same length. See Appendix 3.1 for formal definitions.pecifically, instead of being selected at random, κ is set equal to an encryption of the anamorphic message computed using an encryption scheme with pseudorandom ciphertexts. The encryption key K used to compute the ciphertext constitutes the double key. To construct a valid signature, we observe that the ElGamal signature scheme imposes the following relation:

$$H(m, r) = x \cdot r + s \cdot \kappa.$$

Therefore, if κ and, consequently, $r = g^{\kappa}$ are fixed and x is the secret key, then the above equation can be solved for s so to produce a legal signature (r, s).

Let us proceed more formally and denote by $\mathsf{prE} = (\mathsf{prKG}, \mathsf{prEnc}, \mathsf{prDec})$ an IND-CPA secure encryption scheme with pseudorandom ciphertexts. Consider the following anamorphic triplet $\mathsf{EIT} = (\mathsf{aKG}, \mathsf{aSig}, \mathsf{aDec})$.

1. The anamorphic key-generation algorithm aKG on input 1^{λ} runs EIKG to obtain $(\mathtt{asvk} := (\mathbb{G}, g, H, y), \mathtt{assk} := (\mathbb{G}, g, H, x))$. In addition, the algorithm randomly selects $K \leftarrow \mathsf{prKG}(1^{\lambda})$ and sets $\mathtt{dkey} := (K, x)$. Finally, the algorithm outputs $(\mathtt{asvk}, \mathtt{assk}, \mathtt{dkey})$.

2. The anamorphic signing algorithm aSig, on input messages $(\mathtt{msg}, \mathtt{amsg})$, signing key $\mathtt{assk} = (p, h, H, x)$ and double key K, computes ciphertext $\mathtt{prct} = \mathsf{prEnc}(K, \mathtt{amsg})$, and sets $\kappa := \mathtt{prct}$ and $r = g^{\kappa}$. Finally, s is computed as $s = (H(\mathtt{msg}, r) - x \cdot r) \cdot \kappa^{-1}$ and the pair (r, s) is output.

3. The anamorphic decryption algorithm aDec receives a signature (r, s) for normal message $\mathtt{msg}$ and double key (K, x) and computes $\mathtt{act}$ as $\mathtt{prct} = (H(\mathtt{msg}, r) - x \cdot r) \cdot s^{-1}$ and $\mathtt{amsg}$ as $\mathtt{amsg} = \mathsf{prDec}(K, \mathtt{prct})$.

The following theorem follows from the the general proof of the anamorphism of the Fiat-Shamir heuristics. We provide a proof for this special case as a warm-up to the general proof.

Theorem 6. *Given that the ElGamal signature is unforgeable in the Random Oracle Model, the ElGamal signature scheme is a symmetric anamorphic signature scheme in the Random Oracle Model.*

Proof. Correctness is straightforward and we also observe that the double key contains the signing key x so the extract algorithm is trivial. To complete the proof, suppose for the sake of contradiction that there exists a PPT dictator $\mathcal{D}$ that breaks the anamorphism of EIS and EIT. That is, there exists a polynomial poly such that

$$\left|\mathrm{Prob}\left[\mathsf{RealG}_{\mathsf{EIS},\mathcal{D}}(\lambda) = 1\right] - \mathrm{Prob}\left[\mathsf{AnamorphicG}_{\mathsf{EIT},\mathcal{D}}(\lambda) = 1\right]\right| \geq 1/\mathsf{poly}(\lambda).$$

Consider now the following PPT adversary $\mathcal{A}$ for the pseudorandomness of the ciphertexts of prE. According to Definition 6 (see Appendix 3.1), $\mathcal{A}$ is provided with access to an oracle O that is either equal to OPr^0, that returns random strings, or equal to OPr^1, that returns encryptions of the input with respect to a randomly selected key of prE. $\mathcal{A}$ constructs $(\mathtt{svk}, \mathtt{ssk})$ by running EIKG and runs $\mathcal{D}$ on input $(\mathtt{svk}, \mathtt{ssk})$. Whenever $\mathcal{D}$ makes a query $(\mathtt{msg}, \mathtt{amsg})$, $\mathcal{A}$ prepares the reply by following algorithm EISig with the only exception that κ is obtained by invoking O; that is, $\kappa = O(\mathtt{amsg})$. When $\mathcal{D}$ stops and outputs a bit, $\mathcal{A}$ outputs the same bit.

Now observe that if $O = \mathsf{OPr}^0$ then $\mathcal{A}$ is playing game $\mathsf{PRCtG}^0_{\mathsf{prE},\mathcal{A}}$ and is providing $\mathcal{D}$ with a view of RealG. Indeed the signatures provided have the same distribution of the output of ElSig on input $\mathtt{msg}$. On the other hand, if $O = \mathsf{OPr}^1$ then $\mathcal{A}$ is playing game $\mathsf{PRCtG}^1_{\mathsf{prE},\mathcal{A}}$ and is providing $\mathcal{D}$ with a view of $\mathsf{AnamorphicG}$. Indeed in this case the signatures provided have the same distribution as the output of aSig on input $\mathtt{msg}$ and $\mathtt{amsg}$. Therefore we have that

$$\text{Prob}\left[\mathsf{PRCtG}^0_{\mathsf{prE},\mathcal{A}}(\lambda) = 1\right] = \text{Prob}\left[\mathsf{RealG}_{\mathsf{ElS},\mathcal{D}}(\lambda) = 1\right]$$

and

$$\text{Prob}\left[\mathsf{PRCtG}^1_{\mathsf{prE},\mathcal{A}}(\lambda) = 1\right] = \text{Prob}\left[\mathsf{AnamorphicG}_{\mathsf{ElT},\mathcal{D}}(\lambda) = 1\right]$$

which, together with our assumptions about $\mathcal{D}$, contradicts the pseudorandomness of prE. □

Remark. In the construction of the anamorphic triplet for ElGamal we have made the implicit assumption that the ciphertexts of prE are randomly distributed in $\mathbb{Z}_p$. Indeed, the anamorphic signing algorithm should sample ciphertexts until one in $\mathbb{Z}_p$ is obtained and this on average will require at most 2 trials. Similar considerations apply to other constructions prensented in this paper.

On using asymmetric encryption schemes with pseudorandom ciphertexts. The acute reader might wonder why we use a *symmetric* encryption scheme with pseudorandom ciphertexts instead of an *asymmetric* one. Indeed for a symmetric encryption scheme the same key must be used to encrypt and to decrypt and thus it must be securely shared between two parties. Asymmetric encryption indeed does not require any secure channel as no key must be securely shared. Unfortunately, this would not be feasible in our settings. In particular, the dictator has disallowed encryption and thus there is no public-key directory for public keys. Even though this were not a problem, the moment a user publishes their public key, the dictator will request to see the associated secret key. In other words, even if we use an asymmetric encryption scheme the dictator must be aware of the existence of a public key; therefore it must be shared privately, which bring us back to the symmetric key setting.

3.4 Schnorr Signatures

The Schnorr signature scheme [Sch90, Sch91] is proved secure in the Random Oracle Model under the assumption of hardness of the discrete logarithm problem (see [PS96] and [Seu12]). A formal description of the signature scheme is recalled in Figure 2.

Let us now convince that the Schnorr signature scheme is anamorphic by showing that the randomness κ used by the signer can be extracted by the verifier. Then as done for ElGamal, the randomness can be replaced with a ciphertext of a symmetric encryption scheme with pseudorandom ciphertexts whose key is part of the double key.

Let us proceed more formally and denote by $\mathsf{prE} = (\mathsf{prKG}, \mathsf{prEnc}, \mathsf{prDec})$ an IND-CPA secure encryption scheme with pseudorandom ciphertexts.

1. The anamorphic key-generation algorithm aScKG on input 1^λ runs ScKG to obtain $(\mathtt{svk}, \mathtt{ssk})$. In addition, the algorithm randomly selects $K \leftarrow \mathsf{prKG}(1^\lambda)$.

   Finally, the algorithm outputs $(\mathtt{asvk} := \mathtt{svk}, \mathtt{assk} := \mathtt{ssk}, \mathtt{dkey} := (K, \mathtt{ssk}))$.

1. The key generation algorithm $\mathsf{ScKG}(1^\lambda)$ randomly selects $\mathbb{G}$, a cyclic group of prime order q, for q of length $\Theta(\lambda)$, a generator g for $\mathbb{G}$, and a hash function $H : \{0,1\}^\star \times \mathbb{G} \to \mathbb{Z}_q$.

   Then the algorithm randomly selects $x \leftarrow \mathbb{Z}_q$ and sets $y = g^x$. The *signing key* is set equal to $\mathtt{ssk} := (\mathbb{G}, g, H, x)$ and the *verification key* is set equal to $\mathtt{svk} := (\mathbb{G}, g, H, y)$.

2. To sign message $\mathtt{msg}$, the signing algorithm ScSig, randomly selects $\kappa \leftarrow \mathbb{Z}_q$, computes $r = g^\kappa$, $c = H(\mathtt{msg}, r)$ and $s = \kappa + c \cdot x$.

   The signature for $\mathtt{msg}$ is the pair (r, s).

3. To verify a signature (r, s) against message $\mathtt{msg}$ and verification key $\mathtt{svk} := (\mathbb{G}, g, H, y)$, algorithm $\mathsf{ScVerify}$ first computes $c = H(\mathtt{msg}, r)$ and then checks that $r = g^s \cdot y^{-c}$.

Figure 2: The Schnorr signature scheme.

2. The anamorphic signing algorithm aSig, on input messages $(\mathtt{msg}, \mathtt{amsg})$, anamorphic signing key $\mathtt{ssk}$ and double key $(K, \mathtt{ssk})$, computes the ciphertext $\mathtt{prct} = \mathsf{prEnc}(K, \mathtt{amsg})$ and sets $\kappa := \mathtt{prct}$. Then the algorithm proceeds as ScSig; that is, it sets $r = g^\kappa$, $s = \kappa + x \cdot c$, where $c = H(\mathtt{msg}, r)$ and outputs signature (r, s).

3. The anamorphic decryption algorithm aDec, on input signature (r, s), and K and $\mathtt{ssk} = (\mathbb{G}, g, H, x)$, computes $c = H(\mathtt{msg}, r)$, and obtains κ as $\kappa = s - x \cdot c$. Finally, the anamorphic message $\mathtt{amsg}$ is obtained as $\mathtt{amsg}\mathsf{prDec}(K, \kappa)$.

We have the following theorem whose proof is omitted and derives easily from the general theorem about the anamorphism of the Fiat-Shamir heuristics. See Theorem 9.

Theorem 7. *If the Schnorr signature is unforgeable in the ROM, then the Schnorr signature scheme is a symmetric anamorphic signature scheme in the ROM.*

4 Three-Message Public-Coin Protocols

In this section, we study the Fiat-Shamir heuristics [FS87] that constructs signature schemes from three-message public-coin protocols. If the protocol enjoys a specific security property (see Theorem 8 below), the resulting signature scheme is unforgeable in the Random Oracle Model. We show that this general technique preserves anamorphism; that is, if the starting protocol is anamorphic (in a sense that will be formally defined below) then the resulting signature scheme is also anamorphic. We then give a simple sufficient condition for anamorphism of a protocol and use it to show that several well known three-message public-coin protocols are anamorphic and thus so are the signature schemes obtained from them via Fiat-Shamir heuristics.

Let us set up our terminology (see also Appendix D) Let (P, V) be two PPT machines playing a three round protocol for a polynomial relation $\mathcal{R}$. If $(x, w) \in \mathcal{R}$ we say that x is the *instance* and w is the *witness*. Following the notation used for Sigma protocols [CDS94], we let $(a, \mathtt{st}) \leftarrow P(x, w)$ be the pair of the first message a of the interaction and P's state. Message a is sent to V which responds with a random string e of length $r(\lambda)$, for some polynomially bounded function $r(\cdot)$. P concludes the interaction by computing and sending message $z \leftarrow P(x, w, \mathtt{st}, e)$. Finally, V outputs a bit $b \leftarrow V(x, a, e, z)$. We denote by $[P(x, w) \leftrightarrow V(x)]$ the distribution of the transcripts (a, e, z)

of interactions between P and V over random coin tosses of P and V. We say that a transcript $(a, e, z) \leftarrow [P(x, w) \leftrightarrow V(x)]$ is accepting for x if $V(x, a, e, z)$.

In the next section, we define the notion of an anamorphic three-message public-coin protocol. In Appendix E, we discuss a simple two-message anamorphic protocol.

4.1 Anamorphic Three-Message Public-Coin Protocols

Roughly speaking, a *prover-to-verifier anamorphic* three-message public-coin protocol for relation $\mathcal{R}$ is a protocol in which prover and verifier have access to a private input called the *double key* $\mathtt{dkey}$. Like in a regular protocol (P, V), (x, w) is sampled from $\mathcal{R}$ and the prover has (x, w) and the verifier has x. In addition, both prover and verifier share a *double key* and the prover has an *anamorphic message* $\mathtt{amsg}$ that would like to send the verifier in a secure way. At the end of an execution, in which the anamorphic prover runs on input $(x, w, \mathtt{amsg}, \mathtt{dkey})$ and the anamorphic verifier on input $(x, \mathtt{dkey})$, the verifier extracts $\mathtt{amsg}$ from the transcript. The security property requires that the dictator that has access to $(x, w, \mathtt{amsg})$, but not to $\mathtt{dkey}$, cannot tell whether it is interacting with a real prover or with an anamorphic prover. Let us proceed more formally.

Definition 9. *Let* $\mathsf{3Prot} = (P, V)$ *be a three-message public-coin protocol for relation* $\mathcal{R}$*. We say that* $\mathsf{RRT} = (\mathsf{dKG}, \mathsf{aP}, \mathsf{aDec})$ *is an* anamorphic triplet *for* $\mathsf{3Prot}$ *if*

1. *The* double-key generation *algorithm* dKG*, on input security parameter* 1^λ *and* (x, w) *in the support of* $\mathcal{R}(\lambda)$*, returns* double key $\mathtt{dkey}$*.*
2. aP *is the* anamorphic prover *algorithm that, on input a pair instance-witness* (x, w)*, an anamorphic message* $\mathtt{amsg}$*, and a double key* $\mathtt{dkey}$*, plays the protocol with a verifier. That is,* $\mathsf{aP}(x, w, \mathtt{amsg}, \mathtt{dkey})$ *outputs* $(a, \mathtt{st})$ *and* $\mathsf{aP}(x, w, \mathtt{amsg}, \mathtt{dkey}, a, \mathtt{st}, e)$ *outputs* z*.*

and the following correctness *condition is satisfied:* $\mathsf{aDec}(x, \mathtt{dkey}, \mathtt{tx}) = \mathtt{amsg}$*, except with negligible probability, where* $(x, w) \leftarrow \mathcal{R}(\lambda)$*,* $\mathtt{dkey} \leftarrow \mathsf{dKG}(1^\lambda, x, w)$ *and* $\mathtt{tx} \leftarrow [\mathsf{aP}(x, w, \mathtt{amsg}, \mathtt{dkey}) \leftrightarrow V(x)]$*.*

We are now ready for the definition of a *prover-to-verifier anamorphic three-message public-coin protocol.*

Definition 10. *We say that three-message public-coin protocol* $\mathsf{3Prot} = (P, V)$ *is* prover-to-verifier anamorphic *(or simply,* anamorphic*) if there exists an* anamorphic triplet $\mathsf{T} = (\mathsf{dKG}, \mathsf{aP}, \mathsf{aDec})$ *such that, for all PPT dictators* $\mathcal{D}$*,*

$$\left|\text{Prob}\left[\mathsf{ProtG}^0_{\mathsf{3Prot},\mathsf{T},\mathcal{D}}(\lambda) = 1\right] - \text{Prob}\left[\mathsf{ProtG}^1_{\mathsf{3Prot},\mathsf{T},\mathcal{D}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

where

$\mathsf{ProtG}^\beta_{\mathsf{3Prot},\mathsf{T},\mathcal{D}}(\lambda)$

1. Set $(x, w) \leftarrow \mathcal{R}(\lambda)$ and $\mathtt{dkey} \leftarrow \mathsf{dKG}(1^\lambda, x, w)$;
2. return $\mathcal{D}(x, w)^{O^\beta(\mathtt{amsg})}$, where
   $O^0(\mathtt{amsg})$ samples a transcript from $[P(x, w) \leftrightarrow \mathcal{D}(x, w, \mathtt{amsg})]$; and
   $O^1(\mathtt{amsg})$ samples a transcript from $[\mathsf{aP}(x, w, \mathtt{amsg}, \mathtt{dkey}) \leftrightarrow \mathcal{D}(x, \mathtt{amsg})]$.

Two remarks are in order. First of all, we let the dictator $\mathcal{D}$ pick the anamorphic message $\mathtt{amsg}$ and the challenge e, and have access to the witness w for x. This in the spirit of anamorphism that requires the dictator to have access to any secret information related to a public information they are aware of. Moreover, we also stress that the double key depends on the pair (x, w) and thus they must be generated and securely shared for each different pair (x, w). This does not affect the applicability of the notion. Indeed, we note that in the two main applications to identifications protocols and to signature schemes, the prover's input (x, w) is fixed as they constitute the id and the authentication key in the case of identification protocols, and the verification and the signing keys in the case of signature schemes.

4.2 Fiat-Shamir gives Symmetric Anamorphic Signatures

The Fiat-Shamir heuristics [FS87] constructs a signature scheme from any three-message public-coin protocol for a relation $\mathcal{R}$ satisfying a specific security condition. See Theorem 8 below.

Before proceeding further, let us fix a three-message public-coin protocols $\mathsf{3Prot} = (P, V)$ for a relation $\mathcal{R}$ and describe the signature scheme $\mathsf{FS} = (\mathsf{fsKG}, \mathsf{fsSig}, \mathsf{fsVerify})$ obtained by applying the Fiat-Shamir heuristics to $\mathsf{3Prot}$. 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 a pair $(x, w) \in \mathcal{R}$ with $|x| = \lambda$. The transform uses hash function H.

1. The key-generation algorithm $\mathsf{fsKG}(1^\lambda)$ samples a pair $(x, w) \leftarrow \mathcal{R}(1^\lambda)$. The instance x is the verification key $\mathtt{fsvk} = x$ and the pair (x, w) is the signing key $\mathtt{fssk} = (x, w)$.

2. The signing algorithm fsSig, on input message $\mathtt{msg}$ and signing key $\mathtt{fssk} = (x, w)$, runs the prover P on input (x, w) and produces the first message a along with state $\mathtt{st}$. Then, it sets $e = H(a, \mathtt{msg})$ and computes message $z = P(x, w, a, \mathtt{st})$. The signature of $\mathtt{msg}$ is the transcript $\mathtt{sig} = (a, e, z)$.

3. The verification algorithm $\mathsf{fsVerify}$ on input signature verification key $\mathtt{fsvk} = x$, message $\mathtt{msg}$ and signature $\mathtt{sig} = (a, e, z)$ checks if $V(x, a, e, z) = 1$ and if $e = H(a, \mathtt{msg})$.

There is a long series of results starting from [FS87] that establish the security of the transform. The minimal conditions for the security of FS were given by [AABN02] and are motivated by the use of three-message public-coin protocols as identification protocols. Here the prover picks a pair $(x, w) \in \mathcal{R}$ and x will be the prover's identity and w their secret key. The prover successfully manages to identify themselves if the verifier accepts. A possible security notion for identification schemes is *security against under passive attacks*, where an *impersonator* tries to convince the verifier without the knowledge of the secret key. The impersonator is allowed to obtain any (polynomial) number of of transcripts of honest executions of the protocol, after which he can try to impersonate the legitimate prover. The attack is passive as only honest executions of the protocol are made available to the impersonator. If any PPT impersonator has only negligible probability of success, then we say that the protocol is secure against passive attacks (see formal definition in Section D).

Theorem 8 ([AABN02])**.** *If* $\mathsf{3Prot} = (P, V)$ *is secure against passive attacks, then the signature scheme obtained by applying the Fiat-Shamir heuristics is secure against chosen-message attacks in the ROM.*

Next, we show that if $\mathsf{3Prot}$ is anamorphic, then the signature resulting from applying Fiat-Shamir is anamorphic.

Theorem 9. *Suppose that* $\mathsf{3Prot}$ *is anamorphic and secure against passive attacks. Then the signature scheme* $\mathsf{FS} = (\mathsf{fsKG}, \mathsf{fsSig}, \mathsf{fsVerify})$ *obtained by applying the Fiat-Shamir transform to* $\mathsf{3Prot}$ *is symmetric anamorphic.*

Proof. First of all observe that if $\mathsf{3Prot}$ is secure against passive attacks then, by Theorem 8, FS is unforgeable. Consider now the following anamorphic triplet $\mathsf{afsT} = (\mathsf{afsKG}, \mathsf{afsSig}, \mathsf{afsDec})$ that is, essentially, the Fiat-Shamir transform applied to the anamorphic prover. More precisely, let $\mathsf{T} = (\mathsf{dKG}, \mathsf{aP}, \mathsf{aDec})$ be the anamorphic triplet associated with $\mathsf{3Prot}$.

1. The anamorphic key generation algorithm $\mathsf{afsKG}(1^\lambda)$ runs $\mathsf{fsKG}(1^\lambda)$ to get verification key $\mathtt{svk} := x$ and signing key $\mathtt{ssk} := (x, w)$, and it then selects $d \leftarrow \mathsf{dKG}(1^\lambda, x, w)$ and sets $\mathtt{dkey} := (d, x, w)$.

   Note that $\mathtt{dkey}$ contains the signing key $\mathtt{ssk}$ and thus the scheme is symmetrically anamorphic.

2. The anamorphic signing algorithm afsSig takes as input message $\mathtt{msg}$, anamorphic message $\mathtt{amsg}$, signing key $\mathtt{ssk} = (x, w)$, and double key $\mathtt{dkey} = (d, x, w)$ and computes the anamorphic signature in the following way. Set $(a, \mathtt{st}) \leftarrow \mathsf{aP}(x, w, \mathtt{amsg}, d)$, $e = H(a, \mathtt{msg})$, and $z \leftarrow \mathsf{aP}(x, w, \mathtt{amsg}, d, a, \mathtt{st}, e)$. Finally, $\mathtt{asig} = (a, e, z)$ is the anamorphic signature produced.

3. The anamorphic decryption algorithm afsDec receives anamorphic signature $\mathtt{asig} = (a, e, z)$ and double key $\mathtt{dkey} = (d, x, w)$ and runs algorithm $\mathsf{aDec}(\mathtt{asig}, d)$ to obtain $\mathtt{amsg}$.

Correctness follows from the correctness property of the anamorphic triplet of (P, V). Let us now assume for the sake of contradiction that there exists a dictator $\mathcal{D}$ that breaks anamorphism of FS and consider the following adversary $\mathcal{A}$ that breaks the anamorphism of (P, V) and of the triplet T.

$\mathcal{A}$ receives a pair $(x, w) \leftarrow \mathcal{R}(1^\lambda)$ and interacts with an oracle O that is either prover P or anamorphic prover aP. Upon receiving (x, w) as input, $\mathcal{A}$ prepares $(\mathtt{svk}, \mathtt{ssk}) = (x, (x, w))$ and runs $\mathcal{D}$ on input $(\mathtt{svk}, \mathtt{ssk})$. When $\mathcal{D}$ issues query $(\mathtt{msg}, \mathtt{amsg})$, $\mathcal{A}$ interacts with the oracle O on input $(x, w, \mathtt{amsg})$ and in the interaction $\mathcal{A}$ sets $e = H(a, \mathtt{msg})$. Finally, $\mathcal{A}$ returns the transcript of the interaction as a response to $\mathcal{D}$'s query $(\mathtt{msg}, \mathtt{amsg})$. When $\mathcal{D}$ stops, $\mathcal{A}$ returns $\mathcal{D}$'s output has its own output.

Now observe that if $\mathcal{A}$ plays ProtG^0, and thus $\mathcal{A}$ interacts with prover P, then every query by $\mathcal{D}$ is answered with a regular signature. Therefore, the probability that $\mathcal{A}$ returns 1 in game ProtG^0 is equal to the probability that $\mathcal{D}$ returns 1 in game RealG. On the other hand, if $\mathcal{A}$ plays ProtG^1, and thus $\mathcal{A}$ interacts with anamorphic prover aP, then every query by $\mathcal{D}$ is answered with an anamorphic signature. Therefore, the probability that $\mathcal{A}$ returns 1 in game ProtG^1 is equal to the probability that $\mathcal{D}$ returns 1 in game $\mathsf{AnamorphicG}$. Therefore

$$\begin{aligned}
&\left|\text{Prob}\left[\mathsf{ProtG}^0_{\mathsf{3Prot},\mathsf{T},\mathcal{A}}(\lambda) = 1\right] - \text{Prob}\left[\mathsf{ProtG}^1_{\mathsf{3Prot},\mathsf{T},\mathcal{A}}(\lambda) = 1\right]\right| \\
=&\left|\text{Prob}\left[\mathsf{RealG}_{\mathsf{FS},\mathcal{D}}(\lambda) = 1\right] - \text{Prob}\left[\mathsf{AnamorphicG}_{\mathsf{afsT},\mathcal{D}}(\lambda) = 1\right]\right| \geq 1/\mathsf{poly}(\lambda).
\end{aligned}$$

We thus reached a contradiction since we had assumed that $\mathsf{3Prot}$ was anamorphic. □

4.3 A Sufficient Condition for Anamorphism

In this section we give a sufficient condition for a three-message public-coin protocol to be anamorphic. Roughly speaking, we say that a prover P is *randomness recovering*, if there exists a way to recover the randomness used by P to produce the first message a. As we shall see, randomness recovering are easily proved to be anamorphic. Let us now proceed more formally.

Definition 11 (Randomness Recovering Protocol.)**.** *We say that a prover P of a three-message public-coin protocol is* randomness recovering*, if there exists a* recovering *algorithm* RRecov *that, on input $(x, w) \in \mathcal{R}$ and a transcript (a, e, z) for (x, w), outputs the randomness used to create a.*

We next show that a protocol $\mathsf{3Prot} = (P, V)$ with randomness recovering prover is (prover-to-verifier) anamorphic by using an encryption scheme $\mathsf{prE} = (\mathsf{prKG}, \mathsf{prEnc}, \mathsf{prDec})$ with pseudorandom ciphertexts. More precisely, consider the following "randomness-recovering" anamorphic triplet $\mathsf{RRT} = (\mathsf{dKG}, \mathsf{aP}, \mathsf{aDec})$.

1. The *double-key generation* algorithm dKG on input (x, w), randomly selects $K \leftarrow \mathsf{prKG}(1^\lambda)$ and outputs $\mathtt{dkey} := (x, w, K)$.

2. The *anamorphic prover* aP on input $(x, w, K, \mathtt{amsg})$, computes $\mathtt{act}$ equal to $\mathsf{prEnc}(K, \mathtt{amsg})$ and then aP runs the code of P by using $\mathtt{act}$ as a random tape.

3. The *anamorphic decryption* algorithm aDec, on input a transcript $\mathtt{tx} = (a, e, z)$ and $\mathtt{dkey} = (x, w, K)$, runs the recovering algorithm $\mathsf{RRecov}(x, w, \mathtt{tx})$ to obtain $\mathtt{act}$ and gets $\mathtt{amsg}$ by decrypting $\mathtt{act}$ using K.

We have the following theorem.

Theorem 10. *Any Randomness Recovering protocol is (prover-to-verifier) anamorphic.*

Proof. We observe that the only difference between the games $\mathsf{AnamorphicG}$ and RealG is in the randomness used to construct the first message as in the former true randomness is used, whereas in the latter a pseudorandom ciphertext is employed. Any adversary that distinguishes the two games can be used to distinguish a ciphertext of prE from true randomness thus breaking the pseudorandomness property of prE (see Definition 6). □

4.4 Examples of Three-message Protocols with Randomness Recovering

Several 3-message protocols can be shown to be *randomness recovering* (see, for example, [Bet88, BM91, FS87, Gir91, GQ90, MS90, Oka93, OS91, Poi95, Sch91, Ste94]). In this section, we give two explicit examples, one from the number theoretic domain that can be used to derive the ElGamal signature scheme via the Fiat-Shamir transform (see Section 4.2) and one from the graph theoretic domain.

Discrete Log. Consider the protocol for Discrete Log by Chaum and Pedersen [CP93]. In this 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 holds a the discrete w such that $x = g^w$. He produces a commitment a by picking r at random and setting $a = g^r$. After receiving a random challenge e from the verifier, the prover replies with $z = r + ew$. Finally, the verifier accepts the proof if $g^z = a \cdot x^e$.

To see that the protocol is randomness extracting, we observe that the randomness r used to produce first message a can be obtained from z. Indeed, $r = z - ew$ and the value of e is in the transcript and w is the witness. This is very similar to how we proved that ElGamal and Schnorr signatures and this is no coincidence as the ElGamal and Schnorr signatures derive from the protocol above through the Fiat-Shamir transform. In the next section we show that this general technique that derives signature schemes from three-message public-coin protocols when applied to protocol with randomness recovery yield anamorphic signature schemes.

Graph Isomorphism. This is a classical proof system by [GMW86] that is not used as part of the Fiat-Shamir heuristic as the resulting signature scheme would be inefficient. This is mainly due to the fact that a hard instance would require very large graphs and that the verifier's challenge is only 1-bit long and thus several instances must be executed in parallel to achieve a small error probability. Nonetheless, we give it as an example of a proof system that is not number-theoretic in nature.

Here the common input is a pair of graphs (G_0, G_1) and the prover holds as a witness an isomorphism π such that $\pi(G_0) = G_1$. To produce the first message, the prover picks a random permutation σ and sends the graph $H = \sigma(G_0)$ to the verifier. The verifier's challenge is a random bit e and the final prover's message is an isomorphism $\tau : G_e \to H$.

To see that the protocol is randomness extracting, we observe that the randomness σ used to produce the first message H is equal to τ if $e = 0$; otherwise, $\sigma = \tau \circ \pi$. Note that τ is part of the transcript and π is the witness for the common input.

5 Private Anamorphic Signatures

In this section we introduce the notion of a *private* anamorphic signature scheme, that captures the idea that the anamorphic signing key remains *private*, even if the double key is released. More precisely, the signature scheme remains unforgeable even by an adversary that has access to the double key. Whereas symmetric anamorphic signatures can be used to implement a many-to-many communication channel hidden from the dictator, private anamorphic signatures yield one-to-many channels in which the owner of the signer key is the designated sender and the parties with the double key are the receivers. Of course, every private anamorphic signature can be made symmetric by appending the signing key to the double key.

Requiring that the signature scheme remains unforgeable even if the double key is released is similar to the requirements studied in Theorem 2 that proved that the anamorphic verification keys do not weaken the unforgeability of the signature scheme and this latter property is a direct consequence of the anamorphism property. A *private* anamorphic signature makes the same requirement with respect to adversaries that have access to the *double key* but, of course, not to the anamorphic signing key.

We formally define this concept by strengthening experiment sigG used to formalize unforgeability by letting the adversary also receive the double key; then we present a simple condition (that we call *separability*) on the anamorphic key generation algorithm that is readily seen to be sufficient for private anamorphism. This will allow us to simplify the proof of private anamorphism for our constructions. In Section 7, we show that the Naor-Yung paradigm [NY89] that lifts one-time signatures to many-time signatures gives private anamorphic signature schemes when instantiated with one-time signature schemes that enjoy a weaker form of anamorphism. As a special case, NY

instantiated with Lamport's tagging systems [Lam79] gives private anamorphic signature schemes under the sole assumption of existence of one-way functions.

5.1 Formal Definition of Private Anamorphism

To formalize the concept of a private anamorphic signature, we consider the following game DsigG for an anamorphic signature scheme $\mathsf{S} = (\mathsf{KG}, \mathsf{Sig}, \mathsf{Verify})$, associated triplet $\mathsf{T} = (\mathsf{aKG}, \mathsf{aSig}, \mathsf{aDec})$ and PPT adversary $\mathcal{A}$. The game is similar to the game used to define unforgeability (see Definition 18 of Appendix A)nd the goal of the adversary is to produce a new signature that has not been returned by the signature oracle. However, here we consider a stronger adversary that has access to the double key $\mathtt{dkey}$.

$\mathsf{DsigG}^{\mathcal{A}}_{\mathsf{S},\mathsf{T}}(\lambda)$

1. $(\mathtt{asvk}, \mathtt{assk}, \mathtt{dkey}) \leftarrow \mathsf{aKG}(1^\lambda)$;
2. $(\mathtt{msg}, \mathtt{sig}) \leftarrow \mathcal{A}^{\mathsf{Os}(\cdot, \mathtt{assk})}(\mathtt{asvk}, \mathtt{dkey})$,
   where $\mathsf{Os}(m, \mathtt{assk}) = (m, \mathsf{Sig}(m, \mathtt{assk}))$;
3. if $\mathsf{Verify}(\mathtt{sig}, \mathtt{msg}, \mathtt{asvk}) = 1$ and $(\mathtt{msg}, \mathtt{sig})$ has not been returned by Os then return 1; else return 0.

Definition 12. *An anamorphic signature scheme* $\mathsf{S} = (\mathsf{KG}, \mathsf{Sig}, \mathsf{Verify})$ *with anamorphic triplet* $\mathsf{T} = (\mathsf{aKG}, \mathsf{aSig}, \mathsf{aDec})$ *is* private anamorphic *if, for every PPT adversary* $\mathcal{A}$*, the probability that* $\mathsf{DsigG}^{\mathcal{A}}_{\mathsf{S},\mathsf{T}}(\lambda)$ *returns* 1 *is negligible in* λ*.*

5.2 ElGamal and Schnorr Signature Schemes

In this section we want to briefly discuss the ElGamal and Schnorr signature schemes and how to make them private. It is not difficult to see that the anamorphic triplet presented in Sections 3.3 and 3.4 are not private as they both contain a crucial component of the signing key: the exponent x. We want to briefly describe how to make them private by using the *rejection sampling technique*, that has been used in the context of anamorphic encryption by [PPY22]. As we shall see, this technique only allows to encrypt messages of size $O(\log \lambda)$ whereas in this section we will show private signature schemes with polynomial bandwidth. We concentrate on the ElGamal signature scheme as similar reasoning holds for the Schnorr signature scheme. The double key is the seed K of a pseudorandom function F with one-bit output. To embed a one-bit anamorphic message, the signer samples signatures (r, s) until they obtain a signature with $F(K, r) = b$. On average two samples suffice. If the signer wants to embed an ℓ-bit message m, they consider a pseudorandom function F with ℓ-bit outputs and sample signatures until $F(K, r) = m$. On average, 2^ℓ samples are necessary and thus, to keep signing polynomial, it must be $\ell = O(\log \lambda)$.

5.3 A Sufficient Condition for Private Anamorphism

In the definition of anamorphic triplet, we allow the anamorphic key generation algorithm to jointly generate the anamorphic verification and signing keys, and the double key. This means that the double key and the signing key could share random coin tosses used in their generation and in

this case it is not clear whether unforgeability still holds if an adversary has access to the double key. We consider a special case of anamorphic key generation algorithms that can be seen as the parallel (and independent) composition of the normal key-generation algorithm KG and of the *double-key* generation algorithm dKG that are run on fresh and independent randomness. We call these algorithms and the anamorphic triplets with such key generation algorithms *separable*. Since separable anamorphic key generation algorithms unlink the generation of signing and double key, the same double key can be used across multiple signing keys. This is crucial for anamorphic signature schemes that update the signing (and the verification) key as new signatures are produced (e.g., the Naor-Yung construction discussed in Sect. 7).

Definition 13 (Separable Anamorphic Key Generation.)**.** *Let* S *be an anamorphic signature scheme with anamorphic triplet* $\mathsf{T} = (\mathsf{aKG}, \mathsf{aSig}, \mathsf{aDec})$*. We say that the* T *is* separable *if* aKG *is the parallel and independent composition of algorithm* KG*, that outputs the signing and verification keys, and of PPT algorithm* dKG *that outputs the double key.*

We have the following theorem .

Theorem 11. *Let* S *be an anamorphic signature scheme with anamorphic triplet* T*. If* T *is separable then* S *is private anamorphic.*

Proof. Suppose that there exits a PPT adversary $\mathcal{A}$ for which $\mathsf{DsigG}^{\mathcal{A}}_{\mathsf{S},\mathsf{T}}$ outputs 1 with non-negligible probability. Now observe that for a separable triplet T the DsigG game can be written as follows:

$\mathsf{DsigG}^{\mathcal{A}}_{\mathsf{S},\mathsf{T}}(\lambda)$

1. $(\mathtt{svk}, \mathtt{ssk}) \leftarrow \mathsf{KG}(1^\lambda)$;
2. $\mathtt{dkey} \leftarrow \mathsf{dKG}(1^\lambda)$;
3. $(\mathtt{msg}, \mathtt{sig}) \leftarrow \mathcal{A}^{\mathsf{Os}(\cdot,\mathtt{ssk})}(\mathtt{svk}, \mathtt{dkey})$,
   where $\mathsf{Os}(m, \mathtt{ssk}) = (m, \mathsf{Sig}(m, \mathtt{ssk}))$;
4. if $\mathsf{Verify}(\mathtt{sig}, \mathtt{msg}, \mathtt{asvk}) = 1$ and
   $(\mathtt{msg}, \mathtt{sig})$ has not been returned by Os
   then return 1;
   else return 0.

Let us now consider the following adversary $\mathcal{B}$. $\mathcal{B}$ receives $\mathtt{svk}$ in input and has access to oracle $\mathsf{Os}(\mathtt{ssk}, \cdot)$. $\mathcal{B}$ generates $\mathtt{dkey}$ by running dKG and runs $\mathcal{A}$ on input $(\mathtt{svk}, \mathtt{dkey})$ and relays all $\mathcal{A}$'s queries to Os. It is straightforward to see that $\mathcal{B}$ has the same probability as $\mathcal{A}$ of returning 1. □

6 One-Time Signature Schemes

In this section we look at one-time signature schemes and define the notion of a *weakly anamorphic* one-time signature where the anamorphic message must be available at key-generation time and not at signature time like in anamorphic signatures. From a technical point of view, this is due to the fact that one-time signature schemes have often deterministic signature algorithms and randomization is only used in the selection of the signing and verification keys. On the positive

side, as we shall see below, weak anamorphism allows for a limited decoupling of the generation of the double key and of the pair of anamorphic signing and verification key and thus the same double key can be used in conjuction with more than one anamorphic pair of keys.

Clearly, weakly anamorphic signatures have limited interest *per se*. The reason for studying this concept is in its applications. We shall see, in Section 7, that weakly anamorphic one-time signatures give full private anamorphism when lifted to many-time signatures by the Naor-Yung paradigm. In Appendix G this is extended to its tree-based version. In addition, in Section 8 we show that weakly anamorphic one-time signature can be used in the context of the CHK [CHK04] CCA encryption scheme to make the resulting encryption scheme anamorphic. Let us now proceed in order.

Syntax for One-Time Signature Schemes. We modify slightly the syntax of a one-time signature $\mathsf{OneTSig} = (\mathsf{oKG}, \mathsf{oSig}, \mathsf{oVerify})$ by having the key-generation algorithm oKG accept as input the security parameter 1^λ as well as the length parameter 1^ℓ that determines the length of the one message that will be signed. Also, the security game sigG is modified to reflect the one-time nature of the scheme by allowing the adversary $\mathcal{A}$ at most one query to the signature oracle Os.

In the next section, we formalize the notion of *weak anamorphism*

6.1 Weak Anamorphic Signature Schemes

Just as the notion of an anamorphic triplet is crucial for anamorphism, for weak anamorphism we have the concept of a *weakly anamorphic triplet*. There are two differences between the concepts. First of all, since we intend to embed the anamorphic message in the verification key, the signing algorithm needs not to have knowledge of the anamorphic nature of the signature being produced and, thus, there is no need for an anamorphic signing algorithm. In addition, however, to model the limited form of decoupling of double and anamorphic keys we have two algorithms for key generation: algorithm odKG that selects the double-key $\mathtt{odkey}$ and the algorithm oaKG that computes the anamorphic pair of keys on input the double-key $\mathtt{dkey}$ and the anamorphic message $\mathtt{amsg}$. Note that the decoupling is not complete as the one guaranteed by separability (see Definition 13) but, as we shall see, it is sufficient to obtain private anamorphism. Our definition is tailored for one-time signature schemes as all weakly anamorphic schemes that we will consider in this paper will be one-time signature schemes.

Definition 14 (Weakly Anamorphic Triplet.)**.** *A* weakly anamorphic triplet $\mathsf{T} = (\mathsf{odKG}, \mathsf{oaKG}, \mathsf{oaDec})$ *for one-time signature scheme* $\mathsf{OneTSig} = (\mathsf{oKG}, \mathsf{oSig}, \mathsf{oVerify})$ *consists of three PPT algorithms such that*

1. *the* double-key generation algorithm odKG *that, on input security parameter* 1^λ*, outputs the double-key* $\mathtt{odkey}$*;*
2. *the* anamorphic key-generation algorithm oaKG *that, on input security parameter* 1^λ*, length parameter* 1^ℓ*, double-key* $\mathtt{odkey}$*, and anamorphic message* $\mathtt{amsg}$ *outputs the pair* $(\mathtt{asvk}, \mathtt{assk})$ *composed of the* anamorphic verification key $\mathtt{asvk}$ *and the* anamorphic signing key $\mathtt{assk}$*;*
3. *the* anamorphic decryption algorithm oaDec *takes as input a signature* $\mathtt{asig}$ *and a* double key $\mathtt{odkey}$ *and outputs an anamorphic message* $\mathtt{amsg}$*;*

and that satisfy the following correctness requirement

- *For every pair of ℓ-bit messages* $(\mathtt{msg}, \mathtt{amsg})$ *the probability that* $\mathsf{oaDec}(\mathtt{sig}, \mathtt{dkey}) \neq \mathtt{amsg}$ *is negligible, where* $\mathtt{odkey} \leftarrow \mathsf{odKG}(1^\lambda)$, $(\mathtt{asvk}, \mathtt{assk}) \leftarrow \mathsf{oaKG}(1^\lambda, 1^\ell, \mathtt{dkey}, \mathtt{amsg})$, *and* $\mathtt{sig} = \mathsf{Sig}(\mathtt{msg}, \mathtt{assk})$ *and the probability is taken over the random coin tosses of* odKG *and* oaKG.

We next present the definition of a weakly anamorphic signature scheme.

Definition 15. *An unforgeable one-time signature scheme* $\mathsf{OneTSig} = (\mathsf{oKG}, \mathsf{oSig}, \mathsf{oVerify})$ *is* weakly anamorphic *with weakly anamorphic triplet* $\mathsf{T} = (\mathsf{odKG}, \mathsf{oaKG}, \mathsf{oaDec})$ *if for every PPT dictator* $\mathcal{D}$*, and for every* $\ell = \mathsf{poly}(\lambda)$*, there exists a negligible function* negl *such that*

$$\left|\text{Prob}\left[\mathsf{WeakG}^0_{\mathsf{S},\mathsf{T},\mathcal{D}}(\lambda,\ell)=1\right] - \text{Prob}\left[\mathsf{WeakG}^1_{\mathsf{S},\mathsf{T},\mathcal{D}}(\lambda,\ell)=1\right]\right| \leq \mathsf{negl}(\lambda)$$

where

$\mathsf{WeakG}^\beta_{\mathsf{S},\mathsf{T},\mathcal{D}}(\lambda,\ell)$

1. $\mathtt{odkey} \leftarrow \mathsf{odKG}(1^\lambda)$;
2. return $\mathcal{D}^{\mathsf{OracleKG}^\beta(1^\lambda,1^\ell,\mathtt{odkey},\cdot)}(1^\lambda)$;

   where $\mathsf{OracleKG}^0(1^\lambda, 1^\ell, \mathtt{odkey}, \mathtt{amsg}) = \mathsf{oKG}(1^\lambda, 1^\ell)$
   and $\mathsf{OracleKG}^1(1^\lambda, 1^\ell, \mathtt{odkey}, \mathtt{amsg}) = \mathsf{oaKG}(1^\lambda, 1^\ell, \mathtt{odkey}, \mathtt{amsg})$.

In the game above the dictator has access to an oracle that returns pairs of verification and signing keys that are generated either regularly by oKG or anamorphically by oaKG. We note that if keys are generated anamorphically they are with respect to the same double key odkey. Also note that the dictator can compute all signatures of their choice since the signing key is provided by the oracle. At the end the dictator outputs its guess to whether the keys received are anamorphic or regular.

6.2 Lamport's Tagging Systems

We describe Lamport's tagging system $\mathsf{L} = (\mathsf{LKG}, \mathsf{LSig}, \mathsf{LVerify})$, a one-time signature scheme that uses a one-way function $f : \{0,1\}^\star \to \{0,1\}^\star$ (see [Lam79]). For sake of compactness and without loss of generality, we assume that f is length preserving. The idea behind Lamport's tagging system is to select a secret $x_{b,j}$ for each bit position j and for each bit-value b of the message and then hide it by means of a one-way function f by computing $y_{b,j} = f(x_{b,j})$. The $y_{b,j}$'s constitute the verification key and to sign a message one reveals the secrets for each bit position j and value m_j of the message m in that position. Verification then just consists in re-applying the function f to the revealed secrets and checking that the value obtained appears in the verification key. See Figure 3 for a formal description of L.

We have the following theorem.

Theorem 12 ([Lam79])**.** *Assuming existence of one-way functions, Lamport's tagging system is an unforgeable one-time signature scheme.*

Note that for one-time signatures the PPT adversary $\mathcal{A}$ is allowed to make at most one call to oracle Os in game sigG.

1. $\mathsf{LKG}(1^\lambda, 1^\ell)$ randomly selects $x_{0,j}, x_{1,j} \leftarrow \{0,1\}^\lambda$ and sets $y_{0,j} = f(x_{0,j})$ and $y_{1,j} = f(x_{1,j})$, for $j = 1, \ldots, \ell$. The verification key is the sequence $\mathsf{Lvk} = ((y_{0,j}, y_{1,j}))_{j=1}^{\ell}$ and the signing key is the sequence $\mathsf{Lsk} = ((x_{0,j}, x_{1,j}))_{j=1}^{\ell}$.

2. LSig on input message $\mathtt{msg} = m_1, \ldots, m_\ell$ and signing key $\mathsf{Lsk} = ((x_{0,j}, x_{1,j}))_{j=1}^{\ell}$ computes the signature by setting $\mathtt{sig} = (x_{m_j,j})_{j=1}^{\ell}$.

3. To verify signature $\mathtt{sig} = (s_j)_{j=1}^{\ell}$ of message $\mathtt{msg} = m_1, \ldots, m_\ell$ against verification key $\mathsf{Lvk} = ((y_{0,j}, y_{1,j}))_{j=1}^{\ell}$, the $\mathsf{LVerify}$ algorithm checks that $f(s_j) = y_{m_j,j}$ for $j = 1, \ldots, \ell$.

Figure 3: Lamport's tagging system L [Lam79]

How to make Lamport's tagging system weakly anamorphic? Essentially, instead of picking $x_{0,1}$ and $x_{1,1}$ at random, the anamorphic key generation algorithm picks them as the ciphertext of the anamorphic message $\mathtt{amsg}$ computed with respect to an encryption scheme with pseudorandom ciphertexts (message randomized and then encrypted, each time randomized differently). Note that every signature will include exactly one of $x_{0,1}$ and $x_{1,1}$. See Definition 6 for the notion of an encryption scheme with pseudorandom ciphertexts.his embeds the anamorphic message in the verification and signing key. We also note that the encryption key K used to encrypt the anamorphic message in the $x_{0,1}$ and $x_{1,1}$ can be chosen prior and independently from the anamorphic keys. On the other hand the anamorphic keys do depend on K as they contain ciphertexts encrypted with K.

More formally, let $\mathsf{prE} = (\mathsf{prKG}, \mathsf{prEnc}, \mathsf{prDec})$ be a symmetric encryption scheme with pseudorandom ciphertexts. We assume that prE for security parameter λ encrypts $n(\lambda)$-bit plaintexts into λ-bit ciphertexts and that $\ell = n(\lambda)$. Consider the following weakly anamorphic triplet $\mathsf{T} = (\mathsf{LdKG}, \mathsf{LaKG}, \mathsf{LaDec})$.

1. Algorithm $\mathsf{LdKG}(1^\lambda)$ randomly samples a key $K \leftarrow \mathsf{prKG}(1^\lambda)$. The key K is the double key.

2. Algorithm LaKG, on input security parameter 1^λ, length parameter 1^ℓ, double key K, and anamorphic $\mathtt{amsg}$, constructs the signing and verification key just as $\mathsf{LKG}(1^\lambda, 1^\ell)$ with the only exceptions that $x_{0,1} \leftarrow \mathsf{prEnc}(K, \mathtt{amsg})$ and $x_{1,1} \leftarrow \mathsf{prEnc}(K, \mathtt{amsg})$. The algorithm returns $(\mathtt{asvk}, \mathtt{assk})$ so computed.

3. Algorithm LaDec receives a signature $(x_1, \ldots, x_\ell)$ for message $\mathtt{msg}$ and double key K and returns the anamorphic message computed by decrypting x_1 with key K. That is, aDec returns $\mathsf{prDec}(K, x_1)$.

Theorem 13. *Assuming existence of one-way functions, Lamport's tagging system is weakly anamorphic.*

Proof. Under the assumption of existence of a one-way function, Theorem 12 guarantees that Lamport's tagging system is unforgeable and Theorem 3 guarantees that there exists prE with pseudorandom ciphertexts.

To complete the proof, we assume, for the sake of contradiction, that there exists a PPT dictator $\mathcal{D}$ that contradicts the weak anamorphism of L. That is,

$$\left|\text{Prob}\left[\mathsf{WeakG}^0_{\mathsf{L},\mathsf{T},\mathcal{D}}(\lambda)=1\right]-\text{Prob}\left[\mathsf{WeakG}^1_{\mathsf{L},\mathsf{T},\mathcal{D}}(\lambda)=1\right]\right|\geq 1/\mathsf{poly}(\lambda).$$

We construct an adversary $\mathcal{A}$ that breaks the pseudorandomness of the ciphertexts of prE. $\mathcal{A}$ plays one of two games, PRCtG^0 or PRCtG^1, and it has access to an oracle $O(\cdot)$. $\mathcal{A}$ runs $\mathcal{D}$ and answers $\mathcal{D}$'s queries for pairs of verification and signing keys relative to anamorphic message $\mathtt{amsg}$ by running algorithm LKG with the only difference that $x_{0,1}$ and $x_{1,1}$ are computed by querying oracle O twice on $\mathtt{amsg}$. Finally $\mathcal{A}$ returns $\mathcal{D}$'s output.

Let us analyze two cases depending on whether $\mathcal{A}$ is playing PRCtG^0 or PRCtG^1 and thus on the type of oracle O it has access to. If $\mathcal{A}$ is playing PRCtG^0, then O returns random strings and thus $\mathcal{A}$ is providing $\mathcal{D}$ with a view from game $\mathsf{WeakG}^0_{\mathsf{L},\mathsf{T},\mathcal{D}}$. Therefore,

$$\text{Prob}\left[\mathsf{PRCtG}^0_{\mathsf{prE},\mathcal{A}}(\lambda)=1\right]=\text{Prob}\left[\mathsf{WeakG}^0_{\mathsf{L},\mathsf{T},\mathcal{D}}(\lambda)=1\right].$$

On the other hand, if $\mathcal{A}$ is playing PRCtG^1, then O returns encryptions of its input with respect to a randomly chosen key K and thus $\mathcal{A}$ is providing $\mathcal{D}$ with a view from game WeakG^1. Therefore,

$$\text{Prob}\left[\mathsf{PRCtG}^1_{\mathsf{prE},\mathcal{A}}(\lambda)=1\right]=\text{Prob}\left[\mathsf{WeakG}^1_{\mathsf{L},\mathsf{T},\mathcal{D}}(\lambda)=1\right].$$

By our assumption on $\mathcal{D}$, we contradict the assumed pseudorandomness of the ciphertexts of prE. □

In Appendix F, we will discuss other one-time signature schemes that are weakly anamorphic.

7 The Naor-Yung Paradigm for Signatures

The Naor-Yung paradigm [NY89] is a general paradigm to lift one-time signatures to many-time signatures in the standard model and it relies on the notion of a *Universal One-Way Hash functions (UOWHF)*. We show that when instantiated with weakly anamorphic one-time signature schemes, the NY paradigm yields many-time private anamorphic signature schemes. Specifically, the NY paradigm instantiated with Lamport's tagging systems [Lam79] and Rompel's UOWHF based on one-way functions [Rom90] gives a private anamorphic signature scheme based on the minimal assumption of one-way functions.

7.1 Universal One-Way Hash functions

We consider $\mathbb{H}=\{\mathbb{H}_\lambda\}_{\lambda>0}$ where, for each λ, $\mathbb{H}_\lambda$ is a family of functions such that each function $H\in\mathbb{H}_\lambda$ is $H:\{0,1\}^{p_1(k)}\to\{0,1\}^{p_2(k)}$, for some polynomials p_1 and p_2.

Definition 16 (UOWHF [NY89])**.** $\mathbb{H}$ *is a family of* Universal One-Way Hash functions *if*

1. *each $H\in\mathbb{H}_\lambda$ has a description of length $h(\lambda)$, for some polynomial h, and there is a polynomial time algorithm that on input x and the description of H outputs $H(x)$;*
2. *for every PPT algorithm $\mathcal{A}$, the probability that $\mathcal{A}$, on input x and a randomly chosen H from $\mathbb{H}_\lambda$, outputs y such that $H(x)=H(y)$ is negligible in λ;*

3. it is possible to uniformly select H from family $\mathbb{H}_\lambda$ in polynomial time.

We have the following theorem.

Theorem 14 ([NY89, Rom90])**.** *If one-way functions exist then it is possible to construct a family of Universal One-Way Hash functions.*

Roughly speaking, the NY paradigm upgrades a one-time signature scheme to many-time signature by updating the verification key and, consequently, the signing key before each use of the signing algorithm. The i-th key is used to sign the hash of two new keys: one will be used to sign the $(i+1)$-st message and the other is the $(i+1)$-st key. In this way, a one-time key is only used once. A positive side effect of this is that, even if the one-time signature scheme is only weakly anamorphic, NY can be shown to be anamorphic (that is, without the limitation that the anamorphic message is fixed at key-generation time): key-generation time for the one-time signature coincides with the signature time of the many-time signature. In Appendix G we will show that the extensions to a tree-based approach of the NY paradigm can give stateless private anamorphic signature schemes.

Going into more details, let us fix a weakly anamorphic one-time signature $\mathsf{OneTSig} = (\mathsf{oKG}, \mathsf{oSig}, \mathsf{oVerify})$. The many-time scheme obtained from applying the NY paradigm to $\mathsf{OneTSig}$ will have, at any give time, a public state $\mathtt{st} = (H, \mathtt{Svk})$, consisting of (the description of) a UOWHF H and of a one-time *state* verification key $\mathtt{Svk}$; and a private state consisting of the *state* signing key $\mathtt{Ssk}$ associated with $\mathtt{Svk}$. Now let $\mathtt{st}_{i-1} = (H_{i-1}, \mathtt{Svk}_{i-1})$ be the public state after the $(i-1)$-st message has been signed and let $\mathtt{Ssk}_{i-1}$ be the associated signing key. To sign the next message $\mathtt{msg}_i$, the signing algorithm selects a new UOWHF H_i and runs oKG twice to select two pairs of one-time verification and signing keys: the new *state* pair $(\mathtt{Svk}_i, \mathtt{Ssk}_i) \leftarrow \mathsf{oKG}(1^\lambda)$ and an ephemeral *message* pair $(\mathtt{Mvk}_i, \mathtt{Msk}_i) \leftarrow \mathsf{oKG}(1^\lambda)$. The oSig algorithm is then run twice. The first time on input the message signing key $\mathtt{Msk}_i$ to compute a signature of $\mathtt{msg}_i$ and the second time on input the current state signing key $\mathtt{Ssk}_{i-1}$ to sign a hash computed with respect to H_{i-1} of the new state $\mathtt{st}_i = (H_i, \mathtt{Svk}_i)$ and of the ephemeral message verification key $\mathtt{Mvk}_i$. A formal description is found in Figure 4.

Let $\mathsf{T} = (\mathsf{odKG}, \mathsf{oaKG}, \mathsf{oaDec})$ be the weakly anamorphic triplet associated with $\mathsf{OneTSig}$ and let us describe the *separable* anamorphic triplet $(\mathsf{aNYKG}, \mathsf{aNYSig}, \mathsf{aNYDec})$. The anamorphic key-generation aNYKG consists of the parallel execution of algorithm odKG that returns $\mathtt{dkey}$ and of NYKG that returns the initial public state $(H_0, \mathtt{Svk}_0)$ with initial signing key $\mathtt{Ssk}_0$. To encrypt the i-th anamorphic message $\mathtt{amsg}_i$ as part of the signature of the i-th regular message $\mathtt{msg}_i$, the anamorphic signing algorithm aNYSig first runs oaKG on input $\mathtt{dkey}$ and $\mathtt{amsg}_i$ and obtains the ephemeral message pair $(\mathtt{Mvk}_i, \mathtt{Msk}_i)$ and then runs oKG to generate the new state pair $(\mathtt{Svk}_i, \mathtt{Ssk}_i)$. The rest of the signature algorithm stays unchanged; that is, oSig is used with key $\mathtt{Msk}_i$ to sign $\mathtt{msg}_i$ and with key $\mathtt{Ssk}_{i-1}$ to sign $H_{i-1}(H_i, \mathtt{Svk}_i, \mathtt{Mvk}_i)$. Algorithm aNYDec, on input the signature of $\mathtt{msg}_i$, runs aDec to extract the anamorphic message $\mathtt{amsg}_i$. A formal description is found in Figure 5. The following theorem is due to [NY89].

Theorem 15 ([NY89])**.** *If Universal One-Way Hash functions exist then* NYS *is an unforgeable signature scheme.*

Let us now convince ourselves that the Naor-Yung paradigm yields a private anamorphic many-time signature scheme whenever $\mathsf{OneTSig}$ is weakly anamorphic. First of all, observe that in the NY paradigm a new key pair is generated before each signature is produced and this allows to embed the anamorphic message into the verification key. In other words the signature time for the many-time

- The *key-generation* algorithm $\mathsf{NYKG}(1^\lambda)$ runs $\mathsf{oKG}(1^\lambda, 1^\ell)$ and let $\mathtt{Svk}_0$ and $\mathtt{Ssk}_0$ be the verification and the signing key obtained.

  NYKG then randomly selects a UOWHF $H_0 : \{0,1\}^{2\ell+h(\lambda)} \to \{0,1\}^\ell$ from $\mathbb{H}_\lambda$ and outputs initial $\mathtt{st}_0 = (H_0, \mathtt{Svk}_0)$ and signing key $\mathtt{Ssk}_0$. (Here, $h(\lambda)$ is the length of the description of a UOWHF.)

- The *signing algorithm* NYSig takes as input the i-th ℓ-bit message $\mathtt{msg}_i$, the current state $\mathtt{st}_{i-1} = (H_{i-1}, \mathtt{Svk}_{i-1})$ and the current signing key $\mathtt{Ssk}_{i-1}$ and proceeds as follows.

  1. Randomly select a UOWHF $H_i \leftarrow \mathbb{H}_\lambda$ and construct the following pairs of keys
     - $(\mathtt{Svk}_i, \mathtt{Ssk}_i) \leftarrow \mathsf{oKG}(1^\lambda, 1^\ell)$;
     - $(\mathtt{Mvk}_i, \mathtt{Msk}_i) \leftarrow \mathsf{oKG}(1^\lambda, 1^\ell)$;

     The new state and signing key are $\mathtt{st}_i = (H_i, \mathtt{Svk}_i)$ and $\mathtt{Ssk}_i$.
  2. Construct the following two signatures:
     - signature $\mathtt{stSig}_i$ of $H_{i-1}(\mathtt{st}_i, \mathtt{Mvk}_i)$ by running oSig on input signing key $\mathtt{Ssk}_{i-1}$;
     - signature $\mathtt{mSig}_i$ of $\mathtt{msg}_i$ by running oSig on input signing key $\mathtt{Msk}_i$.
  3. Signature $\mathtt{sig}_i = (\mathtt{st}_i, \mathtt{Mvk}_i, \mathtt{stSig}_i, \mathtt{mSig}_i)$ is output.

- The *verification algorithm* $\mathsf{NYVerify}$ takes as input signature $\mathtt{sig}_i = (\mathtt{st}_i, \mathtt{Mvk}_i, \mathtt{stSig}_i, \mathtt{mSig}_i)$ of $\mathtt{msg}_i$ and the current state $\mathtt{st}_{i-1} = (H_{i-1}, \mathtt{Svk}_{i-1})$.

  First it checks $\mathtt{stSig}_i$ by running $\mathsf{oVerify}$ on input message $H_{i-1}(\mathtt{st}_i, \mathtt{Mvk}_i)$ and $\mathtt{Svk}_{i-1}$. Then it checks $\mathtt{mSig}_i$ by running $\mathsf{oVerify}$ on input message $\mathtt{msg}_i$ and $\mathtt{Mvk}_i$.

Figure 4: The NY signature scheme $\mathsf{NYS} = (\mathsf{NYKG}, \mathsf{NYSig}, \mathsf{NYVerify})$ obtained from one-time signature $\mathsf{OneTSig} = (\mathsf{oKG}, \mathsf{oSig}, \mathsf{oVerify})$.

signature scheme produced by NY coincides with the key-generation time of the one-time signature scheme and this allows us to upgrade from weak anamorphism to anamorphism. Moreover, observe that the anamorphic key-generation algorithm is indeed separable as the anamorphic keys are obtained by the regular key-generation algorithm and the double key by an independent execution of algorithm odKG that is guaranteed to exist by weak anamorphism. This guarantees private anamorphism.

We have the following theorem.

Theorem 16. *If universal one-way hash functions exist, the NY signature scheme* $\mathsf{NYS} = (\mathsf{NYKG}, \mathsf{NYSig}, \mathsf{NYVerify})$ *of Figure 4 is private anamorphic whenever* $\mathsf{OneTSig}$ *is weakly anamorphic.*

Proof. First of all, note that if universal one-way hash functions exist then, by Theorem 15, NYS is an unforgeable signature scheme. Next, note that the triplet aNY is separable as aNYKG generates the double-key and the pair of verification and signing key by independent and parallel execution of odKG and of NYKG. See Figure 5.

To finish the proof, let us assume, for sake of contradiction, that there exists a PPT dictator $\mathcal{D}$ that breaks the anamorphism of $\mathsf{NY} = (\mathsf{NYKG}, \mathsf{NYSig}, \mathsf{NYVerify})$ and its triplet aNY and we show

- The *anamorphic key-generation* algorithm $\mathsf{aNYKG}(1^\lambda, 1^\ell)$ runs the normal key-generation NYKG algorithm and let $\mathtt{st}_0$ and $\mathtt{Ssk}_0$ be the initial verification and signing key, respectively. In addition the algorithm runs odKG to obtain double key $\mathtt{dkey}$.
- The *anamorphic signing algorithm* aNYSig takes as input the i-th ℓ-bit normal message $\mathtt{msg}_i$, the i-th anamorphic message $\mathtt{amsg}_i$, the current state $\mathtt{st}_{i-1} = (H_{i-1}, \mathtt{Svk}_{i-1})$, the current state signing key $\mathtt{Ssk}_{i-1}$ and the double key $\mathtt{dkey}$ and proceeds as follows.
  1. Randomly selects a UOWHF H_i;
  2. The algorithm selects the following key pairs:
     - Message pair $(\mathtt{Mvk}_i, \mathtt{Msk}_i)$ is obtained by running oaKG on input $\mathtt{dkey}$ and $\mathtt{amsg}_i$;
     - State pair $(\mathtt{Svk}_i, \mathtt{Ssk}_i)$ is obtained by running oKG.
  3. The algorithm constructs the following two signatures:
     - signature $\mathtt{stSig}_i$ of $H_{i-1}(\mathtt{st}_i, \mathtt{Mvk}_i)$ by running oSig on input signing key $\mathtt{Ssk}_{i-1}$;
     - signature $\mathtt{mSig}_i$ of $\mathtt{msg}_i$ by running oSig on input signing key $\mathtt{Msk}_i$.
  4. return $\mathtt{sig}_i = (\mathtt{st}_i, \mathtt{Mvk}_i, \mathtt{stSig}_i, \mathtt{mSig}_i)$.
- The *anamorphic decryption algorithm* aNYDec takes as input the anamorphic signature $\mathtt{sig}_i = (\mathtt{st}_i, \mathtt{Mvk}_i, \mathtt{stSig}_i, \mathtt{mSig}_i)$ of $\mathtt{msg}_i$ and the double key $\mathtt{dkey}$. It extracts the first component $\mathtt{act}$ of $\mathtt{mSig}_i$ and returns $\mathtt{amsg} = \mathsf{oaDec}(\mathtt{dkey}, \mathtt{act})$.

Figure 5: The anamorphic triplet $\mathsf{aNY} = (\mathsf{aNYKG}, \mathsf{aNYSig}, \mathsf{aNYDec})$ of the NY signature scheme NYS.

existence of PPT dictator $\mathcal{A}$ that breaks the weak anamorphism of $\mathsf{OneTSig} = (\mathsf{oKG}, \mathsf{oSig}, \mathsf{oVerify})$ and of its weakly anamorphic triplet $\mathsf{T} = (\mathsf{odKG}, \mathsf{oaKG}, \mathsf{oaDec})$

For clarity, in Figure 6 and 7, we instantiate the real game for NY and the anamorphic game for aNY by describing NYKG and NYSig in terms of the underlying one-time signature scheme $\mathsf{OneTSig}$ and the algorithms of the anamorphic triplet $\mathsf{aNY} = (\mathsf{aNYSig}, \mathsf{NYVerify}, \mathsf{aNYDec})$ in terms of the algorithms of the weakly anamorphic triplet $\mathsf{T} = (\mathsf{odKG}, \mathsf{oaKG}, \mathsf{oaDec})$ for the one-time signature scheme.

Simple inspection of the description of the two games shows that the only difference between the two games is in the way the message pair of keys $(\mathtt{Mvk}_i, \mathtt{Msk}_i)$ is computed. Now suppose that that there exists a PPT dictator $\mathcal{D}$ that distinguishes the two games. That is, there exists a polynomial poly such that

$$\left|\text{Prob}\left[\mathsf{RealG}_{\mathsf{NY},\mathcal{D}}(\lambda) = 1\right] - \text{Prob}\left[\mathsf{AnamorphicG}_{\mathsf{aNY},\mathcal{D}}(\lambda) = 1\right]\right| \geq 1/\mathsf{poly}(\lambda).$$

We construct an adversary $\mathcal{A}$ that contradicts the weak anamorphism of $\mathsf{OneTSig}$ and of its triplet T. $\mathcal{A}$ constructs $(\mathtt{svk}_0, \mathtt{ssk}_0)$ by running oKG, picks H_0 at random and runs $\mathcal{D}$ on input $((H_0, \mathtt{svk}_0), \mathtt{ssk}_0)$. $\mathcal{A}$ has access to an oracle $O(\cdot)$ that takes an anamorphic message $\mathtt{amsg}$ and returns either a regular pair of one-time verification and signing key or an anamorphic pair with $\mathtt{amsg}$ embedded in it. $\mathcal{A}$ uses O to compute the reply to $\mathcal{D}$ oracle queries for anamorphic message

$\mathsf{RealG}^{\mathcal{D}}_{\mathsf{NY}}(\lambda)$

- $(\mathtt{svk}_0, \mathtt{ssk}_0) \leftarrow \mathsf{oKG}(1^\lambda)$ and $H_0 \leftarrow \mathbb{H}_\lambda$;
- return $\mathcal{D}^{\mathsf{Os}(\cdot,\cdot,\mathtt{ssk}_0)}((H_0, \mathtt{svk}_0), \mathtt{ssk}_0)$.

where $\mathsf{Os}(\mathtt{msg}_i, \mathtt{amsg}_i, \mathtt{ssk}_0)$ computes its reply using NYSig and precisely as follows:

1. Let $\mathtt{st}_{i-1} = (H_{i-1}, \mathtt{Svk}_{i-1})$ be the current state and let $\mathtt{Ssk}_{i-1}$ be the current signing key.
2. Randomly select a UOWHF $H_i \leftarrow \mathbb{H}_\lambda$ and construct the following pairs of keys
   - $(\mathtt{Svk}_i, \mathtt{Ssk}_i) \leftarrow \mathsf{oKG}(1^\lambda, 1^\ell)$;
   - $(\mathtt{Mvk}_i, \mathtt{Msk}_i) \leftarrow \mathsf{oKG}(1^\lambda, 1^\ell)$;

   The new state and signing key are $\mathtt{st}_i = (H_i, \mathtt{Svk}_i)$ and $\mathtt{Ssk}_i$.
3. Construct the following two signatures:
   - signature $\mathtt{stSig}_i$ of $H_{i-1}(\mathtt{st}_i, \mathtt{Mvk}_i)$ by running oSig on input signing key $\mathtt{Ssk}_{i-1}$;
   - signature $\mathtt{mSig}_i$ of $\mathtt{msg}_i$ by running oSig on input signing key $\mathtt{Msk}_i$.
4. Signature $\mathtt{sig}_i = (\mathtt{st}_i, \mathtt{Mvk}_i, \mathtt{stSig}_i, \mathtt{mSig}_i)$ is returned.

Figure 6: The real game

$\mathsf{AnamorphicG}^{\mathcal{D}}_{\mathsf{aNY}}(\lambda)$

- $(\mathtt{svk}_0, \mathtt{ssk}_0) \leftarrow \mathsf{oKG}(1^\lambda)$ and $H_0 \leftarrow \mathbb{H}_\lambda$;
- $\mathtt{dkey} \leftarrow \mathsf{odKG}(1^\lambda)$.
- return $\mathcal{D}^{\mathsf{Oa}(\cdot,\cdot,\mathtt{assk}_0,\mathtt{dkey})}(\mathtt{svk}_0, \mathtt{ssk}_0)$.

where $\mathsf{Oa}(\mathtt{msg}_i, \mathtt{amsg}_i, \mathtt{assk}_0, \mathtt{dkey})$ computes its reply using aNYSig and precisely as follows:

1. Randomly select a UOWHF $H_i \leftarrow \mathbb{H}_\lambda$ and construct the following pairs of keys
   - $(\mathtt{Svk}_i, \mathtt{Ssk}_i) \leftarrow \mathsf{oKG}(1^\lambda, 1^\ell)$;
   - $(\mathtt{Mvk}_i, \mathtt{Msk}_i) \leftarrow \mathsf{oaKG}(1^\lambda, 1^\ell, \mathtt{dkey}, \mathtt{amsg}_i)$;

   The new state and signing key are $\mathtt{st}_i = (H_i, \mathtt{Svk}_i)$ and $\mathtt{Ssk}_i$.
2. Construct the following two signatures:
   - signature $\mathtt{stSig}_i$ of $H_{i-1}(\mathtt{st}_i, \mathtt{Mvk}_i)$ by running oSig on input signing key $\mathtt{Ssk}_{i-1}$;
   - signature $\mathtt{mSig}_i$ of $\mathtt{msg}_i$ by running oSig on input signing key $\mathtt{Msk}_i$.
3. Signature $\mathtt{sig}_i = (\mathtt{st}_i, \mathtt{Mvk}_i, \mathtt{stSig}_i, \mathtt{mSig}_i)$ is returned.

Figure 7: The anamorphic game

amsg. Specifically, $\mathcal{A}$ computes the reply just as in RealG (or $\mathsf{AnamorphicG}$) with the only exception that the pair $(\mathtt{Mvk}, \mathtt{Msk})$ is computed by querying O on amsg.

Now observe that if $\mathcal{A}$ is playing $\mathsf{WeakG}^0_{\mathsf{S},\mathsf{T},\mathcal{A}}$ then the oracle computes its output by running oKG and thus $\mathcal{A}$ is providing $\mathcal{D}$ with a view from $\mathsf{RealG}_{\mathsf{NY},\mathcal{D}}$. Therefore we have that

$$\text{Prob}\left[\mathsf{WeakG}^0_{\mathsf{S},\mathsf{T},\mathcal{A}}(1^\lambda)\right] = \text{Prob}\left[\mathsf{RealG}_{\mathsf{NY},\mathcal{D}}(1^\lambda) = 1\right].$$

On the other hand, if $\mathcal{A}$ is playing $\mathsf{WeakG}^1_{\mathsf{S},\mathsf{T},\mathcal{A}}$ then the oracle computes its output by running oaKG and thus $\mathcal{A}$ is providing $\mathcal{D}$ with a view from $\mathsf{AnamorphicG}_{\mathsf{aNY},\mathcal{D}}$. Therefore we have that

$$\text{Prob}\left[\mathsf{WeakG}^1_{\mathsf{S},\mathsf{T},\mathcal{A}}(1^\lambda) = 1\right] = \text{Prob}\left[\mathsf{AnamorphicG}_{\mathsf{aNY},\mathcal{D}}(1^\lambda) = 1\right].$$

By our assumption on $\mathcal{D}$, we obtain that $\mathcal{A}$ breaks the weak anamorphism of $\mathsf{OneTSig}$. Contradiction.

□

We observe, but do not elaborate further as optimizing efficiency is not a goal of this paper, that one NY signature can accommodate l anamorphic messages making the anamorphic communication very efficient. By instantiating the NY paradigm [NY89] with Lamport's tagging system [Rom90] and by using Rompel's construction for UOWHF [Rom90], we obtain the Naor-Yung-Lamport-Rompel private anamorphic signature scheme, whose security can be based on the existence of one-way functions.

Theorem 17. *If one-way functions exist, then there exists a private anamorphic signature scheme.*

8 Applications: From Private Anamorphic Signature to Anamorphic CCA Encryption

Signatures are often used as building blocks in more complex constructions and in several cases ephemeral keys are generated online to be used just once. If a private anamorphic signature is used to generate the ephemeral keys, then the new primitive becomes anamorphic. In this section, we exemplify this line of application by showing that it is possible to instantiate the construction of CCA secure encryption of [CHK04], that employs a different signature verification key per ciphertext, so to be anamorphic. The same holds for the construction by [DDN91].

The notion of an encryption scheme secure against chosen ciphertext attacks (CCA secure) was put forth by [RS92] and first implemented by [DDN91]. Roughly speaking, the adversary in a CCA attack has access to the decryption oracle which can be used to receive the decryption of any ciphertext of his choice except, obviously, the challenge ciphertext. Resilience to such attacks is also called non-malleability as it should not be able to efficiently maul a ciphertext into one with a related plaintext. Following the approach of Naor and Yung [NY90], CCA security (actually, the lunchtime or CCA-1 flavor of the attack) has been achieved using NIZK to force correctness of the ciphertext. In this section, we look at the construction of [CHK04] that achieves CCA security starting from selectively secure Identity Based Encryption (IBE). See Appendix H for a review of the notion of IBE and of selective security.

The Canetti-Halevi-Katz [CHK04] construction of a CCA secure encryption scheme leverages on the intrinsic non-malleability of a signature scheme. Roughly speaking, to encrypt a message

msg using CHK, the sender selects a pair of verification and signing key (svk, ssk) of a signature scheme, considers svk as the identity of an IBE scheme and encrypts the message with respect to that identity. The final ciphertext is obtained by concatenating the identity svk, the ciphertext with respect to svk and a signature of the ciphertext computed using ssk. See Figure 8 for a formal description. The construction $\mathsf{CHK}(\mathsf{I},\mathsf{S}) = (\mathsf{chkKG},\mathsf{chkEnc},\mathsf{chkDec})$ is parameterized by an IBE $\mathsf{I} = (\mathsf{Setup},\mathsf{Der},\mathsf{Enc},\mathsf{Dec})$ and by a one-time signature scheme $\mathsf{S} = (\mathsf{KG},\mathsf{Sig},\mathsf{Verify})$. We have the following theorem.

Theorem 18 ([CHK04])**.** *If IBE* I *is selectively secure and one-time signature scheme* S *is unforgeable then* $\mathsf{CHK}(\mathsf{I},\mathsf{S})$ *is CCA secure.*

1. The *key generation* algorithm $\mathsf{chkKG}(1^\lambda)$ runs algorithm $\mathsf{Setup}(1^\lambda, 1^{\ell(\lambda)})$ to obtain pp and msk. Here $\ell(\lambda)$ is the length of the verification keys of S for security parameter λ.

   The algorithm outputs pp as the public key chkpk and msk is the secret key chksk.

2. The *encryption* algorithm chkEnc, on input pp and msg, computes *ephemeral keys* by setting $(\mathtt{svk},\mathtt{ssk}) \leftarrow \mathsf{KG}(1^\lambda)$ and encrypts msg w.r.t. identity svk by setting $\mathtt{ct} = \mathsf{Enc}(\mathtt{svk},\mathtt{pp},\mathtt{msg})$. Finally, the algorithm computes $\mathtt{sig} = \mathsf{Sig}(\mathtt{ssk},\mathtt{ct})$ and outputs $\mathtt{chkct} = (\mathtt{svk},\mathtt{ct},\mathtt{sig})$.

3. The *decryption* algorithm chkDec takes a ciphertext $\mathtt{chkct} = (\mathtt{svk},\mathtt{ct},\mathtt{sig})$ and master secret key msk. First it checks that sig is a valid signature by running $\mathsf{Verify}(\mathtt{svk},\mathtt{ct})$. If the test is passed, the algorithm derives the secret key $\mathtt{sk}_{\mathtt{svk}}$ for identity svk by running algorithm $\mathsf{Der}(\mathtt{msk},\mathtt{svk})$ and decrypts by running $\mathsf{Dec}(\mathtt{sk}_{\mathtt{svk}},\mathtt{ct})$.

Figure 8: The $\mathsf{CHK}(\mathsf{I},\mathsf{S})$ construction.

We next show that if S is weakly anamorphic, then $\mathsf{CHK}(\mathsf{I},\mathsf{S})$ is an anamorphic encryption scheme. See Appendix C for the definition of anamorphic encryption scheme.

Theorem 19. *If* I *is a selectively secure IBE and* $\mathsf{S} = (\mathsf{KG},\mathsf{Sig},\mathsf{Verify})$ *is a weakly anamorphic unforgeable one-time signature scheme, then* $\mathsf{CHK}(\mathsf{I},\mathsf{S})$ *is an anamorphic encryption scheme.*

Proof. Theorem 18 gives that $\mathsf{CHK}(\mathsf{I},\mathsf{S})$ is a secure encryption scheme. Now let $\mathsf{T} = (\mathsf{odKG},\mathsf{oaKG},\mathsf{oaDec})$ be the weakly anamorphic triplet for one-time signature scheme S and let us construct an anamorphic triplet $\mathsf{aCHK} = (\mathsf{achkKG},\mathsf{achkEnc},\mathsf{achkDec})$ for CHK in the following way.

1. The anamorphic key-generation algorithm $\mathsf{achkKG}(1^\lambda)$ generates the double key dkey for S by running algorithm odKG. Moreover, the algorithm executes algorithm $\mathsf{chkKG}(1^\lambda)$ to obtain a pair (chkpk, chksk) of public and secret keys for CHK.

2. The anamorphic encryption algorithm achkEnc takes as input the public key chkpk, the double key dkey and the two messages msg and amsg. The algorithm executes the encryption algorithm chkEnc with the following exception

   - pair (svk, ssk) is generated by running oaKG on input dkey and amsg;

   Note that it is crucial that the signature scheme S is weakly anamorphic as we use the limited decoupling of dkey and anamorphic verification key. For otherwise the double key dkey

generated at the key-generation time might be incompatible with the ephemeral anamorphic signing key generated at encryption time.

3. The anamorphic decryption algorithm achkDec takes an anamorphic ciphertext $\texttt{chkct} = (\texttt{svk}, \texttt{ct}, \texttt{sig})$ and extracts the anamorphic message by running the anamorphic decryption algorithm oaDec of S on input the double key dkey and the anamorphic signature sig.

For the sake of contradiction, suppose that there exists a PPT dictator $\mathcal{D}$ that breaks the anamorphism of CHK and aCHK. That is, we assume that

$$\left|\text{Prob}\left[\mathsf{RealG}_{\mathsf{CHK},\mathcal{D}}(\lambda)=1\right]-\text{Prob}\left[\mathsf{AnamorphicG}_{\mathsf{aCHK},\mathcal{D}}(\lambda)=1\right]\right|\geq 1/\mathsf{poly}(\lambda),$$

for some polynomial poly. We use $\mathcal{D}$ to build a dictator $\mathcal{A}$ that breaks the weak anamorphism of one-time signature S.

$\mathcal{A}$ has access to an oracle O that takes an anamorphic message amsg and returns a pair of one-time signature keys $(\texttt{svk}, \texttt{ssk})$ that are either output by the key generation algorithm KG of S or by the anamorphic key generation algorithm oaKG of T on input amsg.

$\mathcal{A}$ starts by constructing a pair $(\texttt{chkpk}, \texttt{chksk})$ of public and secret keys for CHK and runs $\mathcal{D}$ on input the pair. We remind the reader that the public and secret key of CHK are, respectively, the public parameters pp and the master secret key msk of I. When $\mathcal{D}$ issues a query $(\texttt{msg}, \texttt{amsg})$, $\mathcal{A}$ prepares the reply as follows. $\mathcal{A}$ queries O on input amsg and obtains $(\texttt{svk}, \texttt{ssk})$. Then $\mathcal{A}$ constructs the ciphertext $\texttt{chkct} = (\texttt{svk}, \texttt{ct}, \texttt{sig})$ as specified by algorithm chkEnc and returns it to $\mathcal{D}$. Finally, $\mathcal{A}$ outputs $\mathcal{D}$'s output.

Now, we observe that if O computes its output by running algorithm KG, then $\mathcal{A}$ is playing $\mathsf{WeakG}^0_{\mathsf{S},\mathsf{T},\mathcal{A}}$ while simulating $\mathsf{RealG}_{\mathsf{CHK},\mathcal{D}}$. Therefore, we have

$$\text{Prob}\left[\mathsf{WeakG}^0_{\mathsf{S},\mathsf{T},\mathcal{A}}(\lambda)=1\right]=\text{Prob}\left[\mathsf{RealG}_{\mathsf{CHK},\mathcal{D}}(\lambda)=1\right].$$

On the other hand, if O computes its output by running algorithm oaKG, then $\mathcal{A}$ is playing $\mathsf{WeakG}^1_{\mathsf{S},\mathsf{T},\mathcal{A}}$ while simulating $\mathsf{AnamorphicG}_{\mathsf{aCHK},\mathcal{D}}$. Therefore, we have

$$\text{Prob}\left[\mathsf{WeakG}^1_{\mathsf{S},\mathsf{T},\mathcal{A}}(\lambda)=1\right]=\text{Prob}\left[\mathsf{RealG}_{\mathsf{CHK},\mathcal{D}}(\lambda)=1\right].$$

Therefore by our hypothesis, $\mathcal{A}$ breaks the weak anamorphism of S. Contradiction. □

The DDN and Sahai constructions. The DDN construction [DDN91] of CCA2 encryption is parameterized by a signature scheme and generates a new verification key per ciphertext. The bits of the verification key are used to select the random strings for which NIZKs will be produced. By the same argument used for the CHK construction above, we can prove that the DDN construction is anamorphic, if instantiated with a weakly anamorphic one-time signature scheme. Same argument holds for the CCA2 by Sahai [Sah99].

9 Conclusions

Naturally, when the primary reason(s) for designing a cryptosystem are well motivated, formalized, and understood, the next issues to think about are abuse and misuse of such systems in the overall systems security ecosystem. Anamorphic Cryptography is one concrete tool (in the user hand) to review such steps.

We demonstrated that even in scenarios where only authentication/ signatures are allowed and encryption is neutralized, there is a direct way to implement anamorphic channels between a signer and a receiver of the signed message (while employing already designed, existing, or standardized methods and systems). We formulated the conditions under which hidden anamorphic channels are possible, primarily to demonstrate some of the futility of restrictions imposed by governments on cryptographic systems, and we hope others will build on this initial study. However, we also noted that anamorphic signature schemes have other uses, and we showed that they can contribute to other cryptosystems being anamorphic, and be used to design watermarking of signature schemes in order to serve various protective purposes. More applications of the notion are expected to be discovered in future work.

Acknowledgment

The first and the last author have been supported by the National Centre for Research and Development (Warsaw), project ESCAPE PL-TW/VII/5/2020.

References

[AABN02] 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, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 418–433. Springer, Heidelberg, April / May 2002.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, Heidelberg, May 2004.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.

[BC93] Jurjen N. Bos and David Chaum. Provably unforgeable signatures. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 1–14. Springer, Heidelberg, August 1993.

[Bet88] Thomas Beth. Efficient zero-knowledge identification scheme for smart cards. In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, pages 77–84. Springer, Heidelberg, May 1988.

[BHH+15] Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. SPHINCS: Practical stateless hash-based signatures. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 368–397. Springer, Heidelberg, April 2015.

[BHK+19] 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*, pages 2129–2146. ACM Press, November 2019.

[BM91] Ernest F. Brickell and Kevin S. McCurley. An interactive identification scheme based on discrete logarithms and factoring. In Ivan Damgård, editor, *EUROCRYPT'90*, volume 473 of *LNCS*, pages 63–71. Springer, Heidelberg, May 1991.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416. Springer, Heidelberg, May 1996.

[BSI16] BSI. Technical guideline TR-03110 v2.21 – advanced security mechanisms for machine readable travel documents and eidas token. Available at: `https://www.bsi.bund.de/dok/6623528`, 2016.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 174–187. Springer, Heidelberg, August 1994.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222. Springer, Heidelberg, May 2004.

[CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, Heidelberg, August 1993.

[Dav83] Donald W. Davies. Applying the RSA digital signature to electronic mail. *Computer*, 16(2):55–62, 1983.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pages 542–552. ACM Press, May 1991.

[DMS+17] Zakir Durumeric, Zane Ma, Drew Springall, Richard Barnes, Nick Sullivan, Elie Bursztein, Michael Bailey, J. Alex Halderman, and Vern Paxson. The security impact of HTTPS interception. In *NDSS 2017*. The Internet Society, February / March 2017.

[DSS13] Digital Signature Standard (DSS). National Institute of Standards and Technology (NIST), FIPS PUB 186-4, U.S. Department of Commerce, July 2013.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, August 1984.

[Eur] European Commission. Proposal for a regulation of the european parliament and of the council amending Regulation (EU) No 910/2014 as regards establishing a framework for a European Digital Identity. Available at: `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri$=$CELEX%3A52021PC0281`.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[Gir91] Marc Girault. An identity-based identification scheme based on discrete logarithms modulo a composite number (rump session). In Ivan Damgård, editor, *EUROCRYPT'90*, volume 473 of *LNCS*, pages 481–486. Springer, Heidelberg, May 1991.

[GKM+19] Rishab Goyal, Sam Kim, Nathan Manohar, Brent Waters, and David J. Wu. Watermarking public-key cryptographic primitives. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 367–398. Springer, Heidelberg, August 2019.

[GMR84] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A "paradoxical" solution to the signature problem (extended abstract). In *25th FOCS*, pages 441–448. IEEE Computer Society Press, October 1984.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In *27th FOCS*, pages 174–187. IEEE Computer Society Press, October 1986.

[Gol87] Oded Goldreich. Two remarks concerning the Goldwasser-Micali-Rivest signature scheme. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 104–110. Springer, Heidelberg, August 1987.

[GQ90] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 216–231. Springer, Heidelberg, August 1990.

[HMW07] Nicholas Hopper, David Molnar, and David Wagner. From weak to strong watermarking. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 362–382. Springer, Heidelberg, February 2007.

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.

[MS90] Silvio Micali and Adi Shamir. An improvement of the Fiat-Shamir identification and signature scheme. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 244–247. Springer, Heidelberg, August 1990.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43. ACM Press, May 1989.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 31–53. Springer, Heidelberg, August 1993.

[OS91] H. Ong and Claus-Peter Schnorr. Fast signature generation with a Fiat-Shamir-like scheme. In Ivan Damgård, editor, *EUROCRYPT'90*, volume 473 of *LNCS*, pages 432–440. Springer, Heidelberg, May 1991.

[Poi95] David Pointcheval. A new identification scheme based on the perceptrons problem. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *EUROCRYPT'95*, volume 921 of *LNCS*, pages 319–328. Springer, Heidelberg, May 1995.

[PPY22] Giuseppe Persiano, Duong Hieu Phan, and Moti Yung. Anamorphic encryption: Private communication against a dictator. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 34–63. Springer, Heidelberg, May / June 2022.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, Heidelberg, May 1996.

[Riv98] Ronald L. Rivest. Chaffing and Winnowing: Confidentiality without Encryption. `http://people.csail.mit.edu/rivest/chaffing-980701.txt`, 1998.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990.

[RR02] Leonid Reyzin and Natan Reyzin. Better than BiBa: Short one-time signatures with fast signing and verifying. In Lynn Margaret Batten and Jennifer Seberry, editors, *ACISP 02*, volume 2384 of *LNCS*, pages 144–153. Springer, Heidelberg, July 2002.

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, Heidelberg, August 1992.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.

[RTYZ16] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Cliptography: Clipping the power of kleptographic attacks. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 34–64. Springer, Heidelberg, December 2016.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[Seu12] Yannick Seurin. On the exact security of Schnorr-type signatures in the random oracle model. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 554–571. Springer, Heidelberg, April 2012.

[Sim83] Gustavus J. Simmons. The prisoners' problem and the subliminal channel. In David Chaum, editor, *CRYPTO'83*, pages 51–67. Plenum Press, New York, USA, 1983.

[Ste94] Jacques Stern. A new identification scheme based on syndrome decoding. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 13–21. Springer, Heidelberg, August 1994.

[WS09] Jiang Wu and Douglas R. Stinson. An efficient identification protocol secure against concurrent-reset attacks. *J. Math. Cryptol.*, 3(4):339–352, 2009.

[YY96] Adam Young and Moti Yung. The dark side of "black-box" cryptography, or: Should we trust capstone? In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 89–103. Springer, Heidelberg, August 1996.

A Unforgeable Signature Schemes

In this section, we review the concept of an unforgeable signature scheme.

Definition 17 (Signature Scheme)**.** *A* Signature Scheme S *is a triplet of algorithms* $(\mathsf{KG}, \mathsf{Sig}, \mathsf{Verify})$ *with the following syntax:*

1. *the* key-generation *algorithm* KG *takes as input the security parameter* 1^λ *and outputs the pair* $(\mathtt{svk}, \mathtt{ssk})$ *consisting of a public* verification *key and of a secret* signing *key;*
2. *the* signing *algorithm* Sig *takes as input a message* $\mathtt{msg}$ *and the signing key* $\mathtt{ssk}$ *and outputs a* signature $\mathtt{sig}$*;*
3. *the* verification *algorithm* Verify *takes as input a signature* $\mathtt{sig}$*, a message* $\mathtt{msg}$ *and a verification key* $\mathtt{svk}$ *and accepts or rejects* $\mathtt{sig}$ *as a signature of* $\mathtt{msg}$*.*

and that satisfies the following correctness *requirement: for every* $\mathtt{msg}$*, it holds that*

$$\mathsf{Verify}(\mathsf{Sig}(\mathtt{msg}, \mathtt{ssk}), \mathtt{msg}, \mathtt{svk}) = \mathtt{Accept}$$

except with probability $\mathsf{negl}(\lambda)$*, where* $(\mathtt{svk}, \mathtt{ssk}) \leftarrow \mathsf{KG}(1^\lambda)$ *and the probability is taken over the coin tosses of* KG *and* Sig*.*

We next formally define the security of a signature scheme with respect to a *chosen-message attacks*. In this type of attack a PPT adversary $\mathcal{A}$ is given as input a randomly selected verification key $\mathtt{svk}$ and has access to an oracle Os that returns the signature of messages of $\mathcal{A}$'s choice. $\mathcal{A}$ is considered successful if it has a non-negligible probability of producing a pair $(\mathtt{msg}, \mathtt{sig})$ that was never output by the oracle. In particular, we consider $\mathcal{A}$ successful even if they manage to produce a new signature of a message for which he had seen a signature output by Os.

Definition 18. *A signature scheme* S *is* secure against chosen-message attacks *if for every PPT adversary* $\mathcal{A}$ *the probability that the following experiment* $\mathsf{sigG}^{\mathcal{A}}_{\mathsf{S}}(\lambda)$ *returns* 1 *is negligible in* λ*.*

$\mathsf{sigG}^{\mathcal{A}}_{\mathsf{S}}(\lambda)$

1. $(\mathtt{svk}, \mathtt{ssk}) \leftarrow \mathsf{KG}(1^\lambda)$*;*
2. $(\mathtt{msg}, \mathtt{sig}) \leftarrow \mathcal{A}^{\mathsf{Os}(\cdot, \mathtt{ssk})}(\mathtt{svk})$*,*
   where $\mathsf{Os}(m, \mathtt{ssk}) = (m, \mathsf{Sig}(m, \mathtt{ssk}))$*;*
3. *if* $\mathsf{Verify}(\mathtt{sig}, \mathtt{msg}, \mathtt{svk}) = 1$ *and*
   $(\mathtt{msg}, \mathtt{sig})$ *has not been returned by* Os
   then return 1*;*
   else return 0*.*

In the rest of the paper, whenever we say that a signature scheme is *secure* or *unforgeable*, we mean that the scheme is secure against chosen-message attacks as defined in Definition 18.

B Symmetric Encryption schemes

We start by reviewing the syntax of a symmetric encryption scheme.

Definition 19. *A* symmetric encryption scheme E *is a triplet* $\mathsf{E} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *of PPT algorithms with the following syntax*

1. *the* key-generator algorithm KG *takes as input the security parameter* 1^λ *and returns the* secret key $\mathtt{sk} \leftarrow \mathsf{KG}(1^\lambda)$*;*
2. *the* encryption algorithm Enc *takes as input the secret key* $\mathtt{sk}$ *and a message* $\mathtt{msg}$ *and returns a* ciphertext $\mathtt{ct} \leftarrow \mathsf{Enc}(\mathtt{sk}, \mathtt{msg})$*;*
3. *the* decryption algorithm Dec *takes as input the secret key* $\mathtt{sk}$ *and a ciphertext* $\mathtt{ct}$ *and returns a message* $\mathtt{msg} \leftarrow \mathsf{Dec}(\mathtt{sk}, \mathtt{ct})$*;*

that enjoys the following correctness property:

- *for every* $\mathtt{msg}$

$$\text{Prob}\left[\mathtt{sk} \leftarrow \mathsf{KG}(1^\lambda); \mathtt{ct} \leftarrow \mathsf{Enc}(\mathtt{sk}, \mathtt{msg}) \colon \mathsf{Dec}(\mathtt{sk}, \mathtt{ct}) \neq \mathtt{msg}\right] \leq \mathsf{negl}(\lambda).$$

When we wish to stress the random coin tosses R used by the encryption algorithm we will write $\mathtt{ct} \leftarrow \mathsf{Enc}(\mathtt{sk}, \mathtt{msg}; R)$.

Let us now review the notion of security against *chosen plaintext attacks* (IND-CPA security) for symmetric encryption schemes by means of the following game cpaG. More precisely, for an encryption scheme $\mathsf{E} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$, bit $\beta \in \{0, 1\}$, and PPT adversary $\mathcal{A}$, we consider the following security game $\mathsf{cpaG}^\beta_{\mathsf{E},\mathcal{A}}$ in which the adversary is given access to the encryption oracle Oe from which it can obtain the encryptions of messages of its choice. The adversary $\mathcal{A}$ works in two phases: in the first, it outputs the two messages on which it wants to be tested; in the second, it receives a ciphertext carrying one of the two messages and outputs a bit. Essentially, IND-CPA security requires the output of $\mathcal{A}$ to be independent from the message encrypted.

$\mathsf{cpaG}^\beta_{\mathsf{E},\mathcal{A}}(\lambda)$

1. $\mathtt{sk} \leftarrow \mathsf{KG}(1^\lambda)$;
2. $(\mathtt{msg}_0, \mathtt{msg}_1, \mathtt{st}) \leftarrow \mathcal{A}^{\mathsf{Oe}(\mathtt{sk},\cdot)}(1^\lambda)$;
3. $\mathtt{ct} = \mathsf{Oc}^\beta(\mathtt{sk}, \mathtt{msg}_0, \mathtt{msg}_1)$;
4. Return $\mathcal{A}^{\mathsf{Oe}(\mathtt{sk},\cdot)}(\mathtt{ct}, \mathtt{st})$.

where

- $\mathsf{Oc}^\beta(\mathtt{sk}, \mathtt{msg}_0, \mathtt{msg}_1) = \mathsf{Enc}(\mathtt{sk}, \mathtt{msg}_\beta)$;
- $\mathsf{Oe}(\mathtt{sk}, m) = \mathsf{Enc}(\mathtt{sk}, m)$.

We are ready for the formal definition of IND-CPA security.

Definition 20. *Symmetric encryption schem* E *is* IND-CPA secure *if for all PPT adversaries* $\mathcal{A}$ *we have*

$$\left|\text{Prob}\left[\mathsf{cpaG}^0_{\mathsf{E},\mathcal{A}}(\lambda) = 1\right] - \text{Prob}\left[\mathsf{cpaG}^1_{\mathsf{E},\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

C Anamorphic Encryption Schemes

We review the concept of a Anamorphic Encryption scheme introduced in [PPY22].

C.1 Anamorphic Triplets

We start with the syntactic definition of an *anamorphic triplet* of algorithms.

Definition 21 (Anamorphic Triplet)**.** *We say that a triplet* $\mathsf{AME} = (\mathsf{aKG}, \mathsf{aEnc}, \mathsf{aDec})$ *of PPT algorithms is an* anamorphic triplet *if*

- aKG *takes as input the security parameter* 1^λ *and returns a pair* $(\mathtt{apk}, \mathtt{ask})$ *of* anamorphic *keys and a* double *key* $\mathtt{dkey}$*;*
- aEnc *takes as input the* public key $\mathtt{apk}$*, the* double key $\mathtt{dkey}$*, and* two *messages, the* regular plaintext $\mathtt{msg}$ *and the* anamorphic plaintext $\mathtt{amsg}$*, and returns an* anamorphic *ciphertext* $\mathtt{act}$*;*
- aDec *takes as input the* secret key $\mathtt{ask}$*, the* double key $\mathtt{dkey}$*, and an anamorphic ciphertext* $\mathtt{act}$ *and returns a message* m*;*

and, in addition, the following correctness *requirement is satisfied*

- *for every regular message* $\mathtt{msg}$ *and anamorphic message* $\mathtt{amsg}$*, it holds that*

$$\mathsf{aDec}(\mathtt{ask}, \mathtt{dkey}, \mathtt{act}) = \mathtt{amsg}$$

except with negligible in λ *probability, where* $((\mathtt{apk}, \mathtt{ask}), \mathtt{dkey}) \leftarrow \mathsf{aKG}(1^\lambda)$ *and* $\mathtt{act} \leftarrow \mathsf{aEnc}(\mathtt{apk}, \mathtt{dkey}, \mathtt{msg}, \mathtt{amsg})$.

C.2 Anamorphic Encryption Schemes

We are now ready to define the notion of an *Anamorphic Encryption* scheme (or, simply, an *AM Encryption scheme*). Roughly speaking, we will say that a secure encryption scheme $\mathsf{E} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is an *Anamorphic Encryption* scheme if there exists an anamorphic triplet $\mathsf{AME} = (\mathsf{aKG}, \mathsf{aEnc}, \mathsf{aDec})$ such that no PPT dictator can distinguish whether E or AME is being used, even if given access to the secret key. We formalize the notion by means of the following two games involving a dictator $\mathcal{D}$.

$\mathsf{RealG}_{\mathsf{E},\mathcal{D}}(\lambda)$

1. Set $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{KG}(1^\lambda)$
2. Return $\mathcal{D}^{\mathsf{Oe}(\mathtt{pk},\cdot,\cdot)}(\mathtt{pk}, \mathtt{sk})$, where $\mathsf{Oe}(\mathtt{pk}, m, \mathtt{amsg}) = \mathsf{Enc}(\mathtt{pk}, m)$.

$\mathsf{AnamorphicG}_{\mathsf{AME},\mathcal{D}}(\lambda)$

1. Set $((\mathtt{apk}, \mathtt{ask}), \mathtt{dkey}) \leftarrow \mathsf{aKG}(1^\lambda)$
2. Return $\mathcal{D}^{\mathsf{Oa}(\mathtt{apk},\mathtt{dkey},\cdot,\cdot)}(\mathtt{apk}, \mathtt{ask})$, where $\mathsf{Oa}(\mathtt{pk}, \mathtt{dkey}, m, \mathtt{amsg}) = \mathsf{aEnc}(\mathtt{apk}, \mathtt{dkey}, m, \mathtt{amsg})$.

We have the following definition.

Definition 22. *We say that an encryption scheme* E *is an* Anamorphic Encryption *scheme if it is IND-CPA secure and there exists an anamorphic triplet* AME *such that for every PPT dictator* $\mathcal{D}$ *there exists a negligible function* negl *such that*

$$\left|\text{Prob}\left[\mathsf{RealG}_{\mathsf{E},\mathcal{D}}(\lambda) = 1\right] - \text{Prob}\left[\mathsf{AnamorphicG}_{\mathsf{AME},\mathcal{D}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

D Three-message public-coin protocols

In this section we are going to review the concept of a *three-message public-coin* protocol and set up our notation and give definitions for it.

We consider protocols for two PPT parties, the prover P and the verifier V for a NP relation $\mathcal{R}$. With a slight abuse of notation, we identify $\mathcal{R}$ with the distribution (x, w) of instances and witnesses for which the relation is considered.

Once $(x, w) \leftarrow \mathcal{R}(1^\lambda)$ is sampled, both P and V receive in input the instance x and the prover P also received the witness w. The protocol is a three-message protocol with P going first. Specifically, following the notation used for Sigma protocols [CDS94], we let $(a, \mathtt{st}) \leftarrow P(x, w)$ be the pair of the first message a of the interaction and P's state. Message a is sent to V which responds with a random string e of length $r(\lambda)$, for some polynomially bounded function $r(\cdot)$. P concludes the interaction by computing and sending message $z \leftarrow P(x, w, \mathtt{st}, e)$. Finally, V outputs a bit $b \leftarrow V(x, a, e, z)$. We denote by $[P(x, w) \leftrightarrow V(x)]$ the distribution of the transcripts (a, e, z) of interactions between P and V over random coin tosses of P and V. We say that a transcript $(a, e, z) \leftarrow [P(x, w) \leftrightarrow V(x)]$ is accepting for x if $V(x, a, e, z)$. We are interested in protocols that are complete.

Definition 23 (Completeness)**.** *We say that (P, V) is* complete *for $\mathcal{R}$ if for every (x, w) in the support of $\mathcal{R}$ and for every (a, e, z) in the support of $[P(x, w) \leftrightarrow V(x)]$, we have that $V(w, a, e, z) = 1$.*

Next we define the security property for three-message public-coin protocols that essentially says that, for $(x, w) \in \mathcal{R}(1^\lambda)$, all PPT adversaries $\mathcal{A}$ that have only access to x have negligible probability of making V accept, even if they have oracle access to distribution $[P(x, w) \leftrightarrow V(x)]$.

Definition 24 (Security)**.** *We say that (P, V) is* secure under passive attacks *for relation $\mathcal{R}$ if, for every PPT adversary $\mathcal{A}$, the probability that the following games $\mathsf{3RoundG}_{\mathcal{A}}(\lambda)$ outputs 1 is negligible in λ.*

$\mathsf{3RoundG}_{\mathcal{A}}(\lambda)$

1. $(x, w) \leftarrow \mathcal{R}(\lambda)$;
2. $(a, \mathtt{st}) \leftarrow \mathcal{A}^{O(x,w)}(x)$;
3. $e \leftarrow \{0,1\}^{r(\lambda)}$;
4. $z \leftarrow \mathcal{A}(x, a, \mathtt{st}, e)$;
5. *return* $V(x, a, e, z)$;

where $O(x, w)$ returns a transcript sampled according to $[P(x, w) \leftrightarrow V(x)]$.

Note that in definition above, $\mathcal{A}$ has instance x but not the associated witness w. However, $\mathcal{A}$ can get any polynomial number of transcript between P and V on (x, w) through oracle $O(x, w)$. The attack is passive as $\mathcal{A}$ is not allowed to actively engage with P running on (x, w) but can only observe its interaction with V.

E An Anamorphic Two-message Protocol

In this appendix, we look at a two-message protocol that is an interesting example of an identification protocol that is provably non-anamorphic in the prover-to-verifier direction and it is anamorphic in the opposite direction. Exactly as in the three-move protocols described in the previous sections, the verifier must have access to the prover's private input. Let us proceed more formally.

In the Stinson-Wu identification protocol [WS09], the prover holds a private key x and the public key $X = g^x$ (these operations are in an appropriate group $\mathcal{G}$). First, the verifier performs the following steps:

1. choose r at random, set $y_1 := g^r$, $y_2 := X^r$, and $h := H(y_2)$,
2. send (y_1, h) to the prover.

Then the prover runs the following steps:

1. $y_2' := y_1^x$,
2. abort if $h \neq H(y_2')$,
3. send y_2' to verifier.

Finally, the verifier accepts iff $y_2' = y_2$.

Note that the response y_2' of the prover is deterministic: y_2' must satisfy the equation $h = H(y_2')$ and as long as the hash function H is collision resistant it is infeasible to find y_2' different from y_2. Note that the response of the prover can be checked by an external observer since h and y_2' are sent in clear. The only possibility for the prover to deviate from the protocol so that it will not be detected by the observer is to abort, even if h and y_1 are correct.

To obtain an anamorphic channel, we observe that since r is random, so is y_1 and it can be substituted with the ciphertext $\mathtt{prct}$ of an encryption scheme with pseudorandom ciphertexts prE. We assume that the double key $\mathtt{dkey}$ consists of (x, K), where x is the prover's secret and K is a randomly selected encryption key for symmetric encryption scheme prE with psedurandom ciphertexts.

The anamorphic verifier $\mathsf{aVerifier}$ on input the anamorphic message $\mathtt{amsg}$ and the double key $\mathtt{dkey} = (x, K)$ proceeds as follows:

1. $y_2 := \mathsf{prEnc}(K, \mathtt{amsg})$,
2. $y_1 := y_2^{1/x}$, $h := H(y_2)$,
3. send (y_1, h) to the prover.

The anamorphic decryption algorithm aDec simply calculates $y_2 := y_1^x$ and returns $\mathsf{prDec}(K, y_2)$ We note that, by the pseudorandomness of ciphertexts of prE, the view of the dictator (that has access to the prover's secret x) is indistinguishable from the one in which no anamorphic communication is taking place.

F More One-Time Signature Schemes

In this section we describe more one-time signature schemes that can be proved weakly anamorphic.

F.1 The BC one-time signature

Bos and Chaum [BC93] proposed a variation on the Lamport scheme that improved efficiency. They observed that the tagging system revealed a subset S_m of the secrets that depended on the message m being signed and that one-time forgeability is guaranteed as long as no set S_m is contained in $S_{m'}$ for some $m' \neq m$. Based on this they proposed to pick t secrets and for each ℓ-bit message m to open a subset of size $k = t/2$ identified by the message m through some injective function. For this to work, it is enough that the number of k-subsets of set of size t is larger than the number of all ℓ-bit messages; that is, $\binom{t}{k} \geq 2^\ell$.

It is straightforward to see that the BC one-time signature can be made weakly anamorphic using the same technique used for Lamport's tagging system. The only difference that there we were guaranteed that, independently from the message, one of $x_{0,1}$ and $x_{1,1}$ would appear in the signature (as the first bit of the message is either 0 or 1) and thus we made both an encryption of the anamorphic message. Here, to be sure that at least one secret revealed by the signature is an encryption of the anamorphic message it is sufficient to select a set of $k/2+1$ secret and make them ciphertexts of the anamorphic message. Moreover, as for Lamport's tagging system, the double key consists of the encryption key of a symmetric encryption scheme with pseudorandom ciphertexts. We have thus the following theorem.

Theorem 20 ([BC93])**.** *Assuming existence of one-way functions, the BC one-time signature scheme is weakly anamorphic.*

F.2 The HORS one-time signature

In this section we describe the HORS one time signature of [RR02]. HORS departs from the original idea of [Lam79, BC93] of associating one subset S_m of the secrets to each message so that for no two messages $m' \neq m$ we have $S_m \subseteq S_{m'}$ but it requires only that it is difficult to find two such messages. The construction is parametrized by t, the number of secrets, and by k, an upper bound on the size of S_m. The signing key of HORS consists thus of t randomly selected λ-bit values $\mathtt{ssk} = (x_1, \ldots, x_t)$ and the verification key $\mathtt{svk} = (y_1, \ldots, y_t)$ where, $y_i = f(x_i)$, for $i = 1, \ldots, t$ and one-way function f. To sign message m, we compute $H(m)$ and we split the output into of k values $h_1, \ldots, h_k$ each of $\log_2 t$ bits. This natural identifies the at most k integers that constituted the set S_m associated with m that is used to produce a signature of m.

We have the following theorem.

Theorem 21 ([RR02])**.** *If H is chosen at random from a family of collision resistant hash functions and f is a one-way function, then HORS is a one-message unforgeable signature scheme.*

To make HORS weakly anamorphic it is sufficient to make all secrets x_i ciphertexts of the anamorphic message $\mathtt{amsg}$ w.r.t. an encryption scheme with pseudorandom ciphertexts.

Theorem 22. *Under the assumption of existence of a family of collision resistant hash functions and of a one-way function, HORS is weakly anamorphic.*

The verification key of the HORS signature scheme contains t values, which is usually picked to be a power of two. HORST [BHH+15] reduces the size of the verification key of HORS by using a binary hash-tree in which the t leaves contain each one value from HORS' verification key. HORST verification key includes only the root of the tree. A signature contains the paths from the root to each of the leaf that are opened as part of the signature. It is easy to see that HORST can be made weakly anamorphic as well.

G Tree-Based Hash-Based signatures

In this section we look at signatures based on trees; namely, using the NY recursive idea of signing the root of next tree to be used for signing. The idea of using the NY paradigm to lift one-time to many-time in the context of a tree (instead of using it along a line) was suggested in [NY89] for added efficiency (shorter root-to-signature paths) and for making the signature scheme memoryless. This idea is at the base of modern efficient signature schemes like SPHINCS [BHH+15] and SPHINCS+ [BHK+19], the current NIST choice for hash based signature.

We start by reviewing how to adapt NY to a tree structure and then we will show how to make it private anamorphic. A similar construction was proposed by Goldreich [Gol87] to make the RSA-based signature scheme [GMR84] memoryless.

G.1 Tree-Based Signatures

We now review treeS a tree-based signature scheme. The signing algorithm of a tree-based signature schemes implicitly defines a binary tree of depth λ in which each node contains a verification key of one-time signature scheme $\mathsf{OneTSig}$ that, except for the root, is signed along with the verification key of the sibling node by means of the key of the parent node. The root's verification key is the verification key of the whole system. More precisely, the key of a non-leaf node is used two sign the hash of the verification keys of the two children. To sign message $\mathtt{msg}$, the algorithm picks a random leaf and uses its key to sign $\mathtt{msg}$. The signature then consists of the signatures of all the keys on the path from the root to the selected leaf along with the actual signature of the message. To avoid having to store all exponentially many keys, the tree is built as keys are needed so that its size is polynomially bounded in the number of signatures produced.

Let us now give a more detailed description of the $\mathsf{treeS} = (\mathsf{tKG}, \mathsf{tSig}, \mathsf{tVerify})$ signature scheme. The construction is parametrized by a one-time signature scheme $\mathsf{OneTSig}$ and by a family of One-Way Hash Functions (OWHF) $\mathbb{H}$. We denote by $d(\lambda)$ the length of a verification key of $\mathsf{OneTSig}$ for security parameter λ and by $\ell(\lambda)$ the length of the messages that can be signed by $\mathsf{OneTSig}$. We assume that, for security parameter λ, hash functions H maps $2 \cdot d(\lambda)$ bits to $\ell(\lambda)$ bits.

1. The key generation algorithm tKG of treeS randomly selects OWFH $H \leftarrow \mathbb{H}_\lambda$. Moreover, it selects verification and signing key for $\mathsf{OneTSig}$ by setting $(\mathtt{oVK}, \mathtt{oSK}) \leftarrow \mathsf{oKG}(1^\lambda)$. Finally, the verification key is $\mathtt{tVK} = (\mathtt{oVK}, H)$ and the signing key is $\mathtt{tSK} = (\mathtt{oSK}, H)$.

2. The signing algorithm tSig, on input message $\mathtt{msg}$ and signing key $\mathtt{tSK} = (\mathtt{oSK}, H)$, randomly selects λ-bit string $B = b_1, \ldots, b_\lambda$ bits. Then, for $i = 1, \ldots, \lambda - 1$, let B_i denote the prefix $B_i = b_1 \cdots b_{i-1}$ and performs the following steps.

   First, randomly select two pairs $(\mathtt{oVK}_{B_i 0}, \mathtt{oSK}_{B_i 0}) \leftarrow \mathsf{oKG}(1^\lambda)$ and $(\mathtt{oVK}_{B_i 1}, \mathtt{oSK}_{B_i 1}) \leftarrow \mathsf{oKG}(1^\lambda)$ of verification and signing for the one-time signature scheme. Then, sign the hash $H(\mathtt{oVK}_{B_i 0}, \mathtt{oVK}_{B_i 1})$ of the verification by using the parent signing key $\mathtt{oSK}_{B_i}$. We let the root signing key $\mathtt{oSK}$ from $\mathtt{tSK}$ be the signing key corresponding to the empty string B_1. Finally, message $\mathtt{amsg}$ is signed using signing key $\mathtt{oSK}_B$. The algorithm returns as signature the string B, and the sequence of signatures of all the one-time verification keys and their sibling keys found along the path from root to leaf, as described by the string B.

3. The verification algorithm $\mathsf{tVerify}$ verifies all one-time signatures along the path from root to leaf.

How to avoid key re-use. There are two observations that concern the one-time nature of the keys generated. We first observe that the probability that two invocations of tSig for two different messages select the same random string B, and thus the same leaf, is negligible in λ. This does not hold however for non-leaf nodes that are close to the root as they might be selected more than once with non-negligible probability. Note that if the same key of a node is used to sign two different pairs of child keys, we lose the unforgeability of $\mathsf{OneTSig}$. To avoid this, the signing algorithm will store the tree as it is built by successive invocations and will generate keys for a node only upon the first visit. If a stateless signing algorithm is desidered, then keys can be generated pseudo-randomly so that successive visits to the same node will generate the same pair of child pairs. It is sufficient to pseudo-randomly generate keys only for the first, say, $\lambda/2$ levels as for deeper levels the probability that a node is visited more than once is negligible. We thus modify the key-generation and signing algorithm as follows to obtain the memory-less tree-based signature scheme $\mathsf{mltreeS}$.

1. The mltKG key-generation algorithm, on input 1^λ, sets $(\mathtt{tVK}, \mathtt{tSK}) \leftarrow \mathsf{tKG}(1^\lambda)$. Then it selects seed $s \leftarrow \{0,1\}^\lambda$ for a PRF $F_s : \{0,1\}^{d(\lambda)} \rightarrow \{0,1\}^{2r(\lambda)}$, where $d(\lambda)$ is the length on verification key of $\mathsf{OneTSig}$ for security parameter λ and $r(\lambda)$ is the number of random bits used by oKG for for security parameter λ. Seed s is added to the signing key $\mathtt{tSK}$.

2. The mltSig signing algorithm, on input 1^λ, executes algorithm tSig with the only difference being that, for every $i \leq \lambda/2$, $(\mathtt{oVK}_{B_i 0}, \mathtt{oSK}_{B_i 0})$ and $(\mathtt{oVK}_{B_i 1}, \mathtt{oSK}_{B_i 1})$ are obtained by running $\mathsf{oKG}(1^\lambda)$ using $F_s(\mathtt{oVK}_{B_i})$ as randon tape.

3. The $\mathsf{mltVerify}$ algorithm coincides with the $\mathsf{tVerify}$ algorithm.

The proof of the following theorem is similar to the the proof of Theorem 16.

Theorem 23. *If* $\mathsf{OneTSig}$ *is a one-time unforgeable signature, then* $\mathsf{mltreeS}$ *is a memory-less unforgeable many-time signature.*

G.2 Tree-Based Private Anamorphism

We use the same approach that we employed for the NY paradigm to obtain private anamorphism. Specifically, we instantiate the tree-based approach with a weakly anamorphic one-time signature and we notice that we need to anamorphically generate only the keys that belong to leaves because these are the only ones used to actually sign the messages. Contrast this with the NY transform in which every level of the linear structure has keys that sign messages.

Theorem 24. $\mathsf{mltreeS}$ *is a private anamorphic signature scheme whenever* $\mathsf{OneTSig}$ *is weakly anamorphic.*

By instantiating the $\mathsf{mltreeS}$ construction with Lamport's tagging system we obtain the following.

Theorem 25. *If one-way functions exist, there exists a private anamorphic signature scheme.*

H Identity Based Encryption

An Identity Based Encryption (IBE) is a special type of encryption scheme in which every string of a given length (also, called an *identity*) is a public key and the associated secret key is derived from the identity itself by means of a master secret key.

An IBE $\mathsf{I} = (\mathsf{Setup}, \mathsf{Der}, \mathsf{Enc}, \mathsf{Dec})$ consists of four algorithms with the following syntax.

1. The *setup* algorithm Setup that takes as input security parameter 1^λ and the identity length 1^ℓ and outputs a set of *public parameters* $\mathtt{pp}$ and the *master secret key* $\mathtt{msk}$.
2. The *key derivation* algorithm Der that takes as input an ℓ-bit ID $\mathtt{id}$, public parameters $\mathtt{pp}$, and the master secret key $\mathtt{msk}$ and returns the *identity secret key* $\mathtt{sk}_{\mathtt{id}}$.
3. The *encryption* algorithm Enc that takes public parameters $\mathtt{pp}$, identity $\mathtt{id}$, and a message $\mathtt{msg}$ and produces a ciphertext $\mathtt{ct}$.
4. The *decryption* algorithm Dec that takes a ciphertext $\mathtt{ct}$, an identity $\mathtt{id}$ and the associated secret key $\mathtt{sk}$ and outputs the message $\mathtt{msg}$.

It is required that for all $(\mathtt{pp}, \mathtt{msk})$ outputs by Setup, all $\mathtt{id}$, all $\mathtt{sk}$ output by Der on input $\mathtt{id}$, all messages $\mathtt{msg}$ and all ciphertexts $\mathtt{ct}$ output by $\mathsf{Enc}(\mathtt{msg}, \mathtt{id})$, it holds that $\mathsf{Dec}(\mathtt{id}, \mathtt{sk}_{\mathtt{id}}, \mathtt{ct}) = \mathtt{msg}$.

The notion of selective security for an IBE considers an experiment in which the adversary $\mathcal{A}$ starts by outputting the identity $\mathtt{id}^\star$ that they intend to attack. Then $\mathcal{A}$ receives in input public parameters $\mathtt{pp}$ and can adaptively ask the identity secret key for identities of their choice. Finally, $\mathcal{A}$ outputs two messages m_0, m_1, receives the encryption of one of them under $\mathtt{id}^\star$, and gives their final output. The notion of security requires that $\mathcal{A}$'s output in the game above is independent from the message encrypted. Let us proceed more formally and define the following game $\mathsf{SelectG}^\beta_{\mathsf{I},\mathcal{A}}$, for IBE I, PPT adversary $\mathcal{A}$ and $\beta \in \{0,1\}$.

$\mathsf{SelectG}^\beta_{\mathsf{I},\mathcal{A}}(\lambda)$

1. $(\mathtt{id}^\star, \mathtt{st}) \leftarrow \mathcal{A}(1^\lambda)$;
2. $(\mathtt{pp}, \mathtt{msk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)$;
3. $(m_0, m_1) \leftarrow \mathcal{A}(\mathtt{pp}, \mathtt{st})^{O(\mathtt{id}^\star, \mathtt{msk}, \cdot)}$;
4. $\mathtt{ct} \leftarrow \mathsf{Enc}(\mathtt{pp}, \mathtt{id}, m_\beta)$;
5. return $\mathcal{A}(\mathtt{pp})^{O(\mathtt{id}, \mathtt{msk}, \cdot)}$;

where $O(\mathtt{id}^\star, \mathtt{msk}, \mathtt{id}) = \mathsf{Der}(\mathtt{id}, \mathtt{pp}, \mathtt{msk})$ if $\mathtt{id} \neq \mathtt{id}^\star$ and $\perp$ otherwise.

Definition 25. *Identiy based encryption* $\mathsf{I} = (\mathsf{Setup}, \mathsf{Der}, \mathsf{Enc}, \mathsf{Dec})$ *is* selectively secure *if for all PPT adversaries* $\mathcal{A}$*,*

$$\left|\text{Prob}\left[\mathsf{SelectG}^0_{\mathsf{I},\mathcal{A}}(\lambda) = 1\right] - \text{Prob}\left[\mathsf{SelectG}^1_{\mathsf{I},\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

I Private Anamorphic Signatures secure in the ROM (RSA-PSS and more)

In this section, we discuss the *Probabilistic Signature Scheme* (PSS) [BR96], a signature scheme based on RSA signatures [RSA78] and proved secure in the Random Oracle model. In Figure 9, we recall the PSS signature scheme in order to be able to discuss its anamorphic properties. The PSS signature scheme uses two hash functions that are modeled as Random Oracles in the security proof. Given parameters λ_0 and λ_1, the hash functions are: $H : \{0,1\}^\star \to \{0,1\}^{\lambda_1}$ and $G : \{0,1\}^{\lambda_1} \to \{0,1\}^{\lambda-\lambda_1-1}$. Also, we denote by G_0 a function that on input w returns the first λ_0 bits of $G(w)$ and by G_1 a function that returns the remaining $\lambda - \lambda_0 - \lambda_1 - 1$ bits.

1. The key generation algorithm $\mathsf{pssKG}(1^\lambda)$ runs the RSA key generation algorithm to obtain (N, e, d) and outputs $\mathtt{svk} = (N, e)$ and $\mathtt{ssk} = (N, d)$.
2. The signing algorithm pssSig, on input (N, e) and message $\mathtt{msg}$, randomly selects $r \leftarrow \{0,1\}^{\lambda_0}$ and sets $w = H(\mathtt{msg}||r)$. Then it sets $r^\star = G_1(w) \oplus r$ and $y = 0||w||r^\star||G_2(w)$. Finally, the algorithm returns signature $\mathtt{sig} = y^d \pmod N$.
3. The verification algorithm $\mathsf{pssVerify}$, on input signature $\mathtt{sig}$ for message $\mathtt{msg}$ and verification key (N, e), computes y as $y = \mathtt{sig}^e$, parses it as $y = b||w||r^\star||z$ and it sets $r = r^\star \oplus G_1(w)$. Finally, the verification is successful iff $H(\mathtt{msg}||r) = w$, $G_2(w) = z$ and $b = 0$.

Figure 9: The RSA-PSS signature scheme. The parameters λ_0 and λ_1 have to satisfy $\lambda_0+\lambda_1 \leq \lambda-1$.

We use signature randomness r to achieve anamorphism and employ a symmetric encryption key $\mathsf{prE} = (\mathsf{prKG}, \mathsf{prEnc}, \mathsf{prDec})$ with pseudorandom ciphertexts. Indeed, observe that the value r is randomly chosen by the signer while signing and it is fully recovered as part of the verification process. There, once again, the anamorphic key generation algorithm apssKG generates a pair $((N, d), (N, e))$ of signing and verification keys, and randomly selects $K \leftarrow \mathsf{prKG}$ and sets $\mathtt{dkey} := K$. On input $\mathtt{msg}$ and $\mathtt{amsg}$, the anamorphic signing algorithm $\mathsf{apssSig}$ sets $r = \mathsf{prEnc}(K, \mathtt{amsg})$ and then executes signing algorithm pssSig. Finally, the anamorphic decryption algorithm aDec, on input $\mathtt{sig}$, follows the code of $\mathsf{pssVerify}$ and obtains the value of r. The anamorphic message is obtained by decrypting r using double key K. We have thus the following theorem.

Theorem 26. *If RSA-PSS is secure and* prE *is a symmetric encryption scheme with pseudorandom ciphertexts, then RSA-PSS is private anamorphic encryption scheme.*

The proof relies on the PSS randomness being indistinguishable from ciphertext of prE. Essentially, the same reasoning applies to PSS-R, the version of PSS with message recovery and we can thus claim that PSS-R is also private anamorphic signature scheme. In addition, we also note that the same applies to PRab, a version of provably secure Rabin signature, and to its message recovery version PRab-R. See [BR96] for further details.

Comment: Nonce based protocols, random values, randomized messages, etc.: We note that r in the above signatures scheme is a nonce which is recovered in signature verification. There are many protocols in which one party sends a nonce (random value) to the other party as a

challenge (which is typical in identification protocols). A random value which is otherwise structureless is an easy anamorphic channel as above to allow that party to send a hidden message with. The next section shows that even with more complicated protocols where the random value sent is not as readily available as a nonce, we can have a methodology to create anamorphic channels. Furthermore, note that if messages signed in a protocol are not fully deterministic (it is typical that participants have some freedom in choosing messages), anamorphic channel can be inserted in the message (in the rest of the paper we assume messages are adversarially chosen by the dictator, as a model even though messages can be chosen by users). In fact, partially randomized messages is the needed chosen method for embedding a nonce when we employ determinsitic signatures (even as they have a less tight security proofs), such as EdDSA and Full-Domain-Hash-RSA.

J Pseudonymous Signature

In this section we present an anamorphic scheme where the double key is generated jointly with the signing key (as it contains part of it) but still it cannot be used to forge signatures. Our example is Pseudonymous Signature scheme (PS for short) introduced in [BSI16]. The PS scheme provides an opportunity to create signatures using a single private key stored on a secure cryptographic device, but corresponding to multiple public keys - each key corresponding to a different *domain*. Moreover, the public keys and the signatures from different domains are unlinkable – there is no way to check whether they correspond to the same private key. Let us note that constructions of this type may become very important in the near future, providing technical means for realization of the idea of European Privacy Wallet from the new eIDAS Regulation [Eur].

In Figure 10, we recall the procedures of PS.

Anamorphic signing. As we can see, the system can be roughly described as a "double Schnorr" in which both keys are needed to sign and possession of only one allows to link signatures. Analogously to what we have done for Schnorr authentication scheme (see Theorem 7), the double key consists of K, a randomly selected encryption key of a ciphertext with pseudorandom ciphertexts, and of the second signing key $\mathtt{sk}_2$. The anamorphic signing algorithm for anamorphic message $\mathtt{amsg}$ sets $k_2 := \mathsf{prEnc}(K, \mathtt{amsg})$ and then proceeds as the normal signing algorithm. The anamorphic decryption algorithm solves the equation $s_2 = k_2 - c \cdot \mathtt{sk}_2 \bmod p$ for k_2 and uses K to decrypt k_2 thus obtaining $\mathtt{amsg}$.

1. At the setup time, a group $\mathcal{G}$ of a prime order p and a generator g are selected. System private keys $\mathtt{sk}_M, \mathtt{sk}_{\mathtt{ICC}} < p$ are chosen at random. The corresponding public keys $\mathtt{PK}_M = g^{\mathtt{sk}_M}$ and $\mathtt{PK}_{\mathtt{ICC}} = g^{\mathtt{sk}_{\mathtt{ICC}}}$ are published together with a description of $\mathcal{G}$ and the hash function H to be used.

2. The key generation algorithm PSKG chooses $\mathtt{sk}_2 < p$ at random and calculates $sk_2 < p$ that satisfies the equality $\mathtt{sk}_M = \mathtt{sk}_1 + \mathtt{sk}_2 \cdot \mathtt{sk}_{\mathtt{ICC}} \bmod p$. The keys $\mathtt{sk}_1, \mathtt{sk}_2$ are installed on the user's secure device.

3. The pseudonym generation algorithm PSPG, on input the sector public key $\mathtt{PK}_{\mathtt{sector}}$ and user's private keys $\mathtt{sk}_1$, $\mathtt{sk}_2$, generates user's pseudonyms for this sector: $I_1 := \mathtt{PK}_{\mathtt{sector}}^{sk_1}$ and $I_2 := \mathtt{PK}_{\mathtt{sector}}^{sk_2}$.

4. To sign message $\mathtt{msg}$ for sector with public key $\mathtt{PK}_{\mathtt{sector}}$, the signing algorithm PSSig, chooses $k_1, k_2 < p$ at random and calculates the following values:
   $Q := g^{k_1} \cdot \mathtt{PK}_M^{k_2}, \quad A_1 := \mathtt{PK}_{sector}^{k_1}, \quad A_2 := \mathtt{PK}_{sector}^{k_2},$
   $c := H(Q, I_1, A_1, I_2, A_2, \mathtt{PK}_{\mathtt{sector}}, \text{params}, m)$
   $s_1 := k_1 - c \cdot \mathtt{sk}_1 \bmod p, \quad s_2 := k_2 - c \cdot \mathtt{sk}_2 \bmod p.$
   (I_1, A_1 and I_2, A_2 are optional, params are additional parameters)

   The signature for $\mathtt{msg}$, the sector given by $\mathtt{PK}_{\mathtt{sector}}$, and user's pseudonym (I_1, I_2) is the tuple (c, s_1, s_2).

5. To verify a signature (c, s_1, s_2) against message $\mathtt{msg}$, sector given by $\mathtt{PK}_{\mathtt{sector}}$ and pseudonym I_1, I_2 and verification algorithm PSVerify calculates:
   $Q' := \mathtt{PK}_{\mathtt{ICC}}^{c} \cdot g^{s_1} \cdot \mathtt{PK}_M^{s_2}, \quad A_1' := I_1^{c} \cdot \mathtt{PK}_{\mathtt{sector}}^{s_1}, \quad A_2' := I_2^{c} \cdot \mathtt{PK}_{\mathtt{sector}}^{s_2},$
   $c' := H(Q', I_1, A_1', I_2, A_2', \mathtt{PK}_{\mathtt{sector}}, \text{params}, m)$
   and accepts if $c' = c$.

Figure 10: The Pseudonymous Signature scheme.