Improved Security for Linearly Homomorphic Signatures: A Generic Framework

David Mandell Freeman*

Stanford University, dfreeman@cs.stanford.edu

Abstract. We propose a general framework that converts (ordinary) signature schemes having certain properties into linearly homomorphic signature schemes, i.e., schemes that allow authentication of linear functions on signed data. The security of the homomorphic scheme follows from the same computational assumption as is used to prove security of the underlying signature scheme. We show that the following signature schemes have the required properties and thus give rise to secure homomorphic signatures in the standard model:

- The scheme of Waters (Eurocrypt 2005), secure under the computational Diffie-Hellman asumption in bilinear groups.
- The scheme of Boneh and Boyen (Eurocrypt 2004, *J. Cryptology* 2008), secure under the q-strong Diffie-Hellman assumption in bilinear groups.
- The scheme of Gennaro, Halevi, and Rabin (Eurocrypt 1999), secure under the strong RSA assumption.
- The scheme of Hohenberger and Waters (Crypto 2009), secure under the RSA assumption.

Our systems not only allow weaker security assumptions than were previously available for homomorphic signatures in the standard model, but also are secure in a model that allows a stronger adversary than in other proposed schemes.
Our framework also leads to efficient linearly homomorphic signatures that are secure against our stronger adversary under weak assumptions (CDH or RSA) in the random oracle model; all previous proofs of security in the random oracle model break down completely when faced with our stronger adversary.

Keywords. Homomorphic signatures, standard model, bilinear groups, CDH, RSA.

1 Introduction

Suppose Alice has some set of data $m_1, \ldots, m_k$ that she signs with a digital signature and stores in a database. At some later point in time Bob queries the database for the mean $\overline{m}$ of the data. Since Bob suspects the database might be malicious, he also wants Alice's signature on $\overline{m}$ to prove that the mean was computed correctly. Bob's bandwidth is limited, so he can't simply download the whole database, verify the signature, and compute the mean himself. Or maybe he has the bandwidth but Alice has requested that the data be kept private, with only the mean to be made public. What is Bob to do?

* This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0382. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the Department of Defense or the U.S. Government.

Homomorphic signatures [19,5,14,7,2,6] are a cryptographic primitive that addresses this problem. In a homomorphic signature scheme, a user signs messages $m_1, \ldots, m_k$ in some message space $\mathcal{M}$, producing signatures $\sigma_1, \ldots, \sigma_k$; verification is performed as usual for a signature scheme. The "homomorphic" property is as follows: given this set of signatures and a function $f : \mathcal{M}^k \to \mathcal{M}$ from a set of "admissible" functions $\mathcal{F}$, anyone can produce a signature on the pair $(f, f(m_1, \ldots, m_k)) \in \mathcal{F} \times \mathcal{M}$. Validation of the signature asserts that the claimed value is indeed the result of applying f to the underlying data; if the system is secure, then a malicious adversary cannot compute a valid signature on (f, m^*) for any $m^* \neq f(m_1, \ldots, m_k)$.

Homomorphic signatures were originally proposed by Johnson, Molnar, Song, and Wagner [19] and were adapted for the above application by Boneh, Freeman, Katz, and Waters [5], whose motivation was to authenticate packets in *network coding* protocols [1,23]. Other applications of homomorphic signatures include computing statistics, Fourier transforms, or least-squares fits on authenticated data, all of which can be done using "linearly homomorphic" signatures; i.e., those that authenticate linear functions.

The construction of Boneh et al. uses bilinear groups and authenticates linear functions on vectors over large prime fields. Follow-up work by Gennaro, Katz, Krawczyk, and Rabin [14] is based on RSA and authenticates linear functions on vectors over the integers, while the system of Boneh and Freeman [7] is based on lattice assumptions and authenticates linear functions on vectors over small fields. In a recent breakthrough, Boneh and Freeman [6] showed how to use "ideal lattices" to authenticate *polynomial* functions on data; this system is currently the only one that goes beyond linear functions.

In all of the above systems security is proven only in the random oracle model. At present there are only two homomorphic signature schemes proven secure in the standard model. The first is a scheme of Attrapadung and Libert [2], which is based on the Lewko-Waters identity-based encryption scheme [22] and uses bilinear groups of composite order. Signatures consist of three group elements of size at least 1024 bits, and security is proven using three nonstandard (fixed-size) assumptions, two of which are decisional and one of which is computational. The second is a scheme of Catalano, Fiore, and Warinschi [8], which is based on the general framework of "adaptive pseudo-free groups." In the instantiation based on the strong RSA assumption, signatures consist of two integers of size at least 1024 bits.

1.1 Our Contributions

A general framework for homomorphic signatures. Motivated by a desire to construct efficient systems with stronger security, we propose a general framework that converts (ordinary) signature schemes having certain properties into linearly homomorphic signature schemes. The security of the homomorphic scheme follows from the same computational assumption as is used to prove security of the underlying signature scheme. We show that the schemes of Waters [24], Boneh and Boyen [4]; Gennaro, Halevi, and Rabin [13]; and Hohenberger and Waters [18] all have the required properties and thus give rise to secure homomorphic signatures. The resulting homomorphic constructions are all secure under a computational (as opposed to a decisional) assumption in the standard model, and the pairing-based constructions offer shorter signatures than those

of [2] or [8]. Our framework also leads to a variant of the construction of Attrapadung and Libert, as the signature scheme derived from Lewko-Waters IBE has the required properties; the security proof, however, still requires decisional assumptions.

A stronger security model. Not only do our systems allow weaker security assumptions than were previously available for homomorphic signatures, but our schemes are proven secure in a model that allows a stronger adversary than in other proposed schemes. Specifically, in all previous schemes the adversary could adaptively query signatures on many data sets but was required to submit all messages belonging to a given data set at the same time, after which he would receive signatures on all of the messages at once. In our security model the adversary is allowed to adaptively query one *message* at a time, and even to intersperse queries from different data sets. It was not previously known how to construct a homomorphic signature scheme that is secure against such an adversary.

We also observe that certain of our constructions are also secure in the random oracle model under weak assumptions: the Waters-based scheme (actually the same as that of Gentry and Silverberg [15]) under (co-)CDH in bilinear groups, and the Gennaro-Halevi-Rabin scheme under RSA. While these random-oracle schemes are less efficient than current homomorphic schemes that use the same assumptions [5,14], they are secure against our stronger adversary. All previous proofs of security in the random oracle model break down completely when faced with our stronger adversary.

It is possible to modify the proofs of the standard-model schemes of Attrapadung-Libert [2] and Catalano-Fiore-Warinschi [8] to work against our stronger adversary; in the full version of this paper [12] we address a variant of the former.

Many schemes. Our framework gives users a wide range of options when choosing a homomorphic signature scheme, including variability of the underlying vector space (vectors over $\mathbb{F}_p$ for pairing-based systems, vectors over $\mathbb{Z}$ for RSA-based ones) and tradeoffs between security and efficiency (the most efficient systems require stronger assumptions). We also expect our framework to be applicable to other signature schemes, both existing and not yet proposed.

1.2 Overview of Our Construction

We consider *linearly homomorphic* signature schemes, in which messages are vectors $\mathbf{v}$ with coordinates in some ring R and functions are R-linear combinations of messages. Using network coding terminology, we call a set of vectors that can be linearly combined with each other a "file."

The impetus for our framework comes from comparing the Attrapadung-Libert homomorphic signatures [2] to the Lewko-Waters signatures on which they are based [22]. The Lewko-Waters system uses a cyclic group $\mathbb{G}$ whose order $N = pqr$ is a product of three distinct primes, along with a nondegenerate, symmetric bilinear map $\hat{e}$ on $\mathbb{G}$. A signature on a message m consists of two group elements $(\sigma_1, \sigma_2) = \left(g^r h^s,\ g^\alpha H(m)^r h^{s'}\right)$, where g, h are public group elements of prime order p, q, respectively; g^α is the secret key; H is a hash function; and r, s, s' are random in $\mathbb{Z}_N$. Verification can be carried out by testing whether $\hat{e}(\sigma_2, g)/\hat{e}(\sigma_1, H(m))$ is equal to $e(g,g)^\alpha$, where this last value is also public. (Here g and h are constructed so that $\hat{e}(g,h) = 1$.)

Attrapadung and Libert convert this scheme to a homomorphic scheme that signs n-dimensional vectors defined over $\mathbb{Z}_N$. The main idea is that to sign a vector $\mathbf{v} = (v_1, \ldots, v_n)$ belonging to a file F, we use the underlying scheme to sign the filename F (or more precisely, a "tag" chosen at random to identify F) and then add on a signed "homomorphic hash" of the vector $\mathbf{v}$ using *the same randomness* on the g part. Specifically, the signature has the form

$$(\sigma_1, \sigma_2, \sigma_3) = \left(g^r h^s,\ g^\alpha H(F)^r h^{s'},\ (h_1^{v_1} \cdots h_n^{v_n})^r h^{s''}\right)$$

where $h_1, \ldots, h_n$ are additional public group elements in $\langle g \rangle$ and s'' is random. To verify, we check whether the first two components form a valid signature on F, and whether $\hat{e}(\sigma_1, \prod h_i^{v_i}) = \hat{e}(\sigma_3, g)$.

To make signatures on different vectors within a file compatible, we need to use the same randomness r in the underlying signature each time, so the σ_1 and σ_2 components are the same for each vector in the file. Attrapadung and Libert achieve this property by applying a pseudorandom function to the filename F to produce r. Once the randomness is the same across all vectors within a file, the homomorphic property follows: given two vectors $\mathbf{v}, \mathbf{w}$ in the same file F and two signatures $\sigma_{\mathbf{v}} = (\sigma_1, \sigma_2, \sigma_3)$ and $\sigma_{\mathbf{w}} = (\sigma_1, \sigma_2, \sigma_3')$ produced with the same value of r, the triple $(\sigma_1, \sigma_2, \sigma_3\sigma_3')$ is a valid signature on the vector $\mathbf{v} + \mathbf{w}$. Specifically, we have

$$\hat{e}(\sigma_1, \prod h_i^{v_i+w_i}) = \hat{e}(\sigma_1, \prod h_i^{v_i}) \cdot \hat{e}(\sigma_1, \prod h_i^{w_i}) = \hat{e}(\sigma_3, g) \cdot \hat{e}(\sigma_3', g) = \hat{e}(\sigma_3\sigma_3', g).$$

This property generalizes in the obvious way to authenticate $\mathbb{Z}_N$-linear combinations of arbitrary numbers of vectors in $(\mathbb{Z}_N)^n$.

Pre-homomorphic signatures. The idea of using a homomorphic hash to authenticate linear combinations of vectors goes back to Krohn, Freedman, and Mazières [21], and the idea of signing such a hash is used in several previous constructions [5,14,6]. The key idea here — and the one that we can generalize to other systems — is signing the filename and the hash separately and tying them together with the signing function.

Specifically, the abstract properties of the Lewko-Waters scheme that make the homomorphic scheme work are as follows:

- The signature contains a component $\sigma_1 = g^{f(m,r)}$ for some fixed group element g and some function f of the message m and randomness r. (In Lewko-Waters we take $f(m, r) = r$, modulo h components.)
- Given σ_1, m, and two group elements x and y, there is an efficient algorithm to test whether $y = x^{f(m,r)}$. (In Lewko-Waters we use the pairing.)

In Section 3 we formalize these properties in the notion of a *pre-homomorphic signature*.

Our main construction is as follows: given a pre-homomorphic signature, we form a homomorphic signature on a vector $\mathbf{v}$ in a file F by generating signing randomness r using a PRF, signing the tag τ identifying F to produce the component $\sigma_1 = g^{f(m,r)}$ (and perhaps some other component σ_2), and then forming the component $\sigma_3 = (\prod h_i^{v_i})^{f(m,r)}$. The signature on $\mathbf{v}$ is $(\sigma_1, \sigma_2, \sigma_3)$. As in the Attrapadung-Libert scheme, homomorphic operations within the same file can be carried out by multiplying σ_3 components, and verification can be carried out using the testing algorithm. As stated

this system is "weakly" secure, and we must add some kind of "chameleon hash" to obtain full security; details are in Section 3.

Examples. Surveying the literature, we see that many pairing-based schemes have the "pre-homomorphic" structure we define. These include the CDH-based schemes of Gentry-Silverberg [15], Boneh-Boyen [3], and Waters [24], where signatures have the same general form as in the Lewko-Waters system, as well as that of Boneh-Boyen [4], where signatures have the form $g^{1/(x+m+yr)}$ and security is based on the q-strong Diffie-Hellman problem. In all cases we can use the pairing to determine whether two pairs of elements have the same discrete log relationship.

Expanding into the RSA space, we see that the signatures of Gennaro, Halevi, and Rabin [13] also have our "pre-homomorphic" form: signatures are of the form $g^{1/H(m)} \bmod N$, and we can easily test whether $y = x^{1/H(m)}$ by raising both sides to the power $H(m)$. GHR signatures are secure under the strong RSA assumption; Hohenberger and Waters [18] demonstrate a hash function H that allows for a proof of security of the same construction under the (standard) RSA assumption.

Security. As formalized by Boneh et al. [5] for network coding and adapted to the more general homomorphic setting by Boneh and Freeman [6], an attacker tries to break a homomorphic signature scheme by adaptively submitting signature queries to a challenger and outputting a forgery. The forgery is a tuple $(\tau^*, \mathbf{w}^*, \sigma^*, f^*)$ consisting of a "tag" τ^* that identifies a file, a vector $\mathbf{w}^*$, a signature σ^*, and a function f^*. There are two winning conditions: either τ^* does not identify one of the files queried to the challenger (a *Type 1 forgery*), or τ^* does identify such a file F, but $\mathbf{w}^*$ is not equal to $f(\mathbf{v}_1, \ldots, \mathbf{v}_k)$, where $\mathbf{v}_1, \ldots, \mathbf{v}_k$ are the vectors in F (a *Type 2 forgery*).

For our general construction, we give a direct reduction that shows that a Type 1 forgery leads to a break of the underlying signature scheme. Furthermore we show that if the underlying signature scheme is *strongly* unforgeable, then certain Type 2 forgeries also break the underlying scheme. We also observe that since the identifying tags are chosen by the challenger, the underlying scheme need only be unforgeable against a *weak* adversary, i.e., one that submits all of its message queries before receiving the public key. This relaxation allows for improved efficiency in our construction.

For the remaining Type 2 forgeries we do not have a black-box reduction to the underlying signature scheme. However, we can do the next best thing: we can abstract out properties of the scheme's security proof that allow us to use a forgery in the homomorphic system to solve the computational problem used to prove the underlying scheme secure. Specifically, suppose we have a simulator that takes an instance of a computational problem and mimics the underlying signature scheme. Let f be the "pre-homomorphic" signing function discussed above, and suppose that the simulator can produce two group elements x, y with the following properties:

- The simulator can compute $x^{f(m,r)}$ for all message queries.
- The simulator can compute $y^{f(m,r)}$ for all but one message query m^*.
- If r^* is the randomness used to sign m^*, then the value of $y^{f(m^*,r^*)}$ can be used to solve the computational problem.

A typical example of such a simulator is the kind used in security proofs of (strong-)RSA signatures [13,11,16,17,18]: if $\{e_i\}$ is the set of integers that need to be inverted mod

$\varphi(N)$ to answer signature queries, we compute $E = \prod e_i$ and $E^* = \prod_{i \neq \ell} e_i$ for a random ℓ and set $x = g^E \bmod N$, $y = g^{E^*} \bmod N$. Using Shamir's trick, given y^{1/e_ℓ} we can recover g^{1/e_ℓ} and in many cases solve the computational problem.

Given such a simulator, we "program" the homomorphic hash function so that for all vectors queried by the adversary, $H_{\mathsf{hom}}(\mathbf{v})$ consists of x factors only and therefore all signatures can be computed. However, if the adversary produces a linear function f^* described by coefficients $(c_1, \ldots, c_k)$ and a vector $\mathbf{w}^*$ such that $\mathbf{w}^* \neq \sum c_i \mathbf{v}_i$, then we can show that with noticeable probability the hash of $\mathbf{w}^*$ has a nontrivial y factor, and therefore a forged signature can be used to solve the computational problem.

Our general security theorem appears in Section 5. An example instantiation, based on Boneh-Boyen signatures, appears in Section 6. In the full version of this paper [12] we show how to modify our schemes in bilinear groups to achieve privacy; specifically, a derived signature on $m' = f(m_1, \ldots, m_k)$ reveals nothing about the values of the m_i that cannot be obtained from the value of m' and the knowledge of f. (We also show that our RSA schemes do not have this property.)

1.3 Concurrent Work

In concurrent and independent work, Catalano, Fiore, and Warinschi [9] have proposed two new linearly homomorphic signature schemes that are secure in the standard model: one based on Boneh-Boyen signatures and secure under the q-SDH assumption, and one based on Gennaro-Halevi-Rabin signatures and secure under the strong-RSA assumption. Signatures in these schemes consist only of the σ_3 component of our corresponding schemes, and thus signatures are shorter than those arising from our construction. The strong-RSA construction also has the feature that the length of integer vectors to be signed is unbounded. (Our RSA constructions as well as that of [14] require an upper bound on vector length.)

While the constructions in [9] are proved secure only against an adversary that queries entire files at once, it is possible to modify the proofs to work against our stronger adversary. We also expect that if the hash function from [18] is used in the strong-RSA scheme, the resulting scheme is secure under the (standard) RSA assumption. However, it does not appear that the techniques of [9] can be used to produce linearly homomorphic signatures based on Waters signatures and the co-CDH assumption.

Acknowledgments. The author thanks Nuttapong Attrapadung, Dan Boneh, and Benoît Libert for helpful discussions, and the anonymous referees for their feedback.

2 Homomorphic Signatures

In a homomorphic signature scheme we can sign messages m in some message space $\mathcal{M}$ and apply functions f to signed messages for f in some set of "admissible" functions $\mathcal{F}$. Each set of messages is grouped together into a "data set" or "file," and each file is equipped with a "tag" τ that serves to bind together the messages in that file. Formally, we have the following.

Definition 2.1 ([6]). A *homomorphic signature scheme* is a tuple of probabilistic, polynomial-time algorithms $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Eval})$ as follows:

- $\mathsf{Setup}(1^\lambda, k)$. Takes a security parameter λ and a maximum data set size k. Outputs a public key pk and a secret key sk. The public key pk defines a message space $\mathcal{M}$, a signature space Σ, and a set $\mathcal{F}$ of "admissible" functions $f\colon \mathcal{M}^k \to \mathcal{M}$.
- $\mathsf{Sign}(\mathsf{sk}, \tau, m, i)$. Takes a secret key sk, a tag $\tau \in \{0,1\}^\lambda$, a message $m \in \mathcal{M}$ and an index $i \in \{1, \ldots, k\}$, and outputs a signature $\sigma \in \Sigma$. (The index i indicates that this is the ith message in the file.)
- $\mathsf{Verify}(\mathsf{pk}, \tau, m, \sigma, f)$. Takes a public key pk, a tag $\tau \in \{0,1\}^\lambda$, a message $m \in \mathcal{M}$, a signature $\sigma \in \Sigma$, and a function $f \in \mathcal{F}$, and outputs either 0 (reject) or 1 (accept).
- $\mathsf{Eval}(\mathsf{pk}, \tau, f, \vec{\sigma})$. Takes a public key pk, a tag $\tau \in \{0,1\}^\lambda$, a function $f \in \mathcal{F}$, and a tuple of signatures $\vec{\sigma} \in \Sigma^k$, and outputs a signature $\sigma' \in \Sigma$.

Let $\pi_i\colon \mathcal{M}^k \to \mathcal{M}$ be the function $\pi_i(m_1, \ldots, m_k) = m_i$ that projects onto the ith component. We require that $\pi_1, \ldots, \pi_k \in \mathcal{F}$ for all pk output by $\mathsf{Setup}(1^\lambda, k)$.

Informally, the correctness conditions of our scheme are that (a) a signature produced by Sign on message m with index i verifies for the projection function π_i, and (b) if Eval is given a function g and signatures that verify for messages m_i and functions f_i, then the signature output by Eval verifies for the message $g(\vec{m})$ and the function obtained by composing g with the f_i.

Formally, we require that for each $(\mathsf{pk}, \mathsf{sk})$ output by $\mathsf{Setup}(1^\lambda, k)$, we have:

1. Let $\tau \in \{0,1\}^\lambda$ be any tag, let $m \in \mathcal{M}$ be any message, and let $i \in \{1, \ldots, k\}$ be any index. If $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \tau, m, i)$, then $\mathsf{Verify}(\mathsf{pk}, \tau, m, \sigma, \pi_i) = 1$.
2. Let $\tau \in \{0,1\}^\lambda$ be any tag, let $\vec{\mu} = (\mu_1, \ldots, \mu_k) \in \mathcal{M}^k$ be any tuple of messages, let $\vec{\sigma} = (\sigma_1, \ldots, \sigma_k) \in \Sigma^k$ be signatures produced by zero or more iterative applications of Eval on the outputs of $\mathsf{Sign}(\mathsf{sk}, \tau, \mu_i, i)$, and let $(f_1, \ldots, f_k, g) \in \mathcal{F}^{k+1}$ be any tuple of admissible functions. Let $g \circ \vec{f}$ denote the function that sends $\vec{x} = (x_1, \ldots, x_k)$ to $g(f_1(\vec{x}), \ldots, f_k(\vec{x}))$. If $\mathsf{Verify}(\mathsf{pk}, \tau, m_i, f_i) = 1$ for some $m_1, \ldots, m_k \in \mathcal{M}$, the message $g(m_1, \ldots, m_k)$ is in $\mathcal{M}$, and the function $g \circ \vec{f}$ is admissible, then $\mathsf{Verify}\left(\mathsf{pk},\ \tau,\ g(\vec{m}),\ \mathsf{Eval}\big(\mathsf{pk}, \tau, g, \vec{\sigma}\big),\ g \circ \vec{f}\right) = 1$.

Note that if $f_i = \pi_i$ is the ith projection function, then the function $g \circ \vec{f}$ in condition (2) is equal to g. Thus condition (2) says that if we apply Eval to the function g and signatures $\sigma_i = \mathsf{Sign}(\mathsf{pk}, \tau, \mu_i, i)$ for $i = 1, \ldots, k$, then the resulting signature verifies for the message $g(\vec{\mu})$ and the function g.

A *linearly homomorphic* signature scheme is a homomorphic signature scheme where the message space $\mathcal{M}$ consists of n-dimensional vectors over a ring R, and the set of admissible functions $\mathcal{F}$ consists of R-linear functions from $(R^n)^k$ to R. We identify $\mathcal{F}$ with a subset of R^k by representing the function $f(\mathbf{v}_1, \ldots, \mathbf{v}_k) = \sum c_k \mathbf{v}_i$ as the vector $(c_1, \ldots, c_k) \in R^k$.

Relationship to Network Coding. Definition 2.1 generalizes the definition of Boneh, Freeman, Katz and Waters for signatures in *network coding* systems [5, Definition 1].

In network coding, a file is parsed as a set of vectors $\mathbf{v}'_1, \ldots, \mathbf{v}'_k \in \mathbb{F}_p^n$. Each vector $\mathbf{v}'_i$ is then "augmented" by appending the ith unit vector $\mathbf{e}_i$, creating k "augmented vectors" $\mathbf{v}_1, \ldots, \mathbf{v}_k \in \mathbb{F}_p^{n+k}$. It is these augmented vectors that are transmitted through the network.

In the network coding protocol, each router in the network creates random linear combinations of its incoming vectors and passes the resulting vectors downstream. The vectors' augmentation carries information about the function that has been applied. Specifically, the ith unit vector that we append to the ith data vector represents the projection function π_i. If we apply the linear function $f\colon (\mathbb{F}_p^{n+k})^k \to \mathbb{F}_p^{n+k}$ given by $f(x_1, \ldots, x_k) = \sum_i c_i x_i$, then the "augmentation component" of $\mathbf{w} = f(\mathbf{v}_1, \ldots, \mathbf{v}_k)$ (i.e., the last k entries) is exactly $(c_1, \ldots, c_k)$. Thus there are two equivalent ways of viewing a signature on a derived vector $\mathbf{w}$: as a signature on the entire vector $\mathbf{w}$, or as a signature on the pair $(\mathbf{w}', f)$ where $\mathbf{w}' = \sum_i c_i \mathbf{v}'_i$ is the first n components of $\mathbf{w}$. Our definition takes the latter view, as it is the one that generalizes more readily to nonlinear functions (see e.g. [6]).

Security. The goal of an adversary attacking a homomorphic signature scheme is to produce a signature on a message-function pair that cannot be derived from previously seen data and signatures. This can be done in two ways: the adversary can produce a signature on a function-message pair that doesn't correspond to a previously seen data set (a *Type 1 forgery*), or the adversary can authenticate an *incorrect* value of a function on a previously seen data set (a *Type 2 forgery*).

In our model, the adversary is allowed to make adaptive queries on data sets of his choice. Our adversary is allowed to query one message at a time and proceed adaptively *within* each data set, or even to intersperse queries from different data sets. In contrast, in previous works the adversary was required to submit all messages in a given data set at once. This new flexibility implies a third type of forgery: the adversary might output a function-message pair that corresponds to a previously seen data set, but for which the adversary has not queried enough messages for the function's output to be well-defined on the input data set. We call this forgery a *Type 3 forgery*.

In our model (and in our constructions) we must avoid collisions between tags τ, so we have the challenger choose them uniformly from $\{0,1\}^\lambda$. Since the adversary can intersperse queries from different files, the signer must maintain a state to ensure that each query is signed with the correct tag and index.

Definition 2.2 (adapted from [6]). A homomorphic signature scheme $\mathcal{S} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Eval})$ is *unforgeable against an adaptive per-message attack* (or simply *unforgeable*) if for all k the advantage of any probabilistic, polynomial-time adversary $\mathcal{A}$ in the following game is negligible in the security parameter n:

Setup: The challenger runs $\mathsf{Setup}(1^\lambda, k)$ to obtain $(\mathsf{pk}, \mathsf{sk})$ and gives pk to $\mathcal{A}$. The public key defines a message space $\mathcal{M}$, a signature space Σ, and a set $\mathcal{F}$ of admissible functions $f\colon \mathcal{M}^k \to \mathcal{M}$.

Queries: $\mathcal{A}$ specifies a filename $F \in \{0,1\}^*$ and a message $\mathbf{v} \in \mathcal{M}$. If $\mathbf{v}$ is the first query for F, the challenger chooses a tag τ_F uniformly from $\{0,1\}^\lambda$, gives it to $\mathcal{A}$, and sets a counter $i_F = 1$. Otherwise, the challenger looks up the value of τ_F previously

chosen and increments the counter i_F by 1. The challenger then gives to $\mathcal{A}$ the signature $\sigma^{(F,i_F)} \leftarrow \mathsf{Sign}(\mathsf{sk}, \tau_F, \mathbf{v}, i_F)$.
The above interaction is repeated a polynomial number of times, subject to the restriction that at most k messages can be queried for any given filename F. We let $\mathbf{V}_F$ denote the tuple of elements $\mathbf{v}$ queried for filename F, listed in the order they were queried.

Output: $\mathcal{A}$ outputs a tag $\tau^* \in \{0,1\}^\lambda$, a message $\mathbf{w}^* \in \mathcal{M}$, a signature $\sigma^* \in \Sigma$, and a function $f^* \in \mathcal{F}$.

We say a function f is *well-defined on* F if either $i_F = k$ or $i_F < k$ and $f(\mathbf{V}_F, \mathbf{v}_{i_F+1}, \ldots, \mathbf{v}_k)$ takes the same value for all possible choices of $(\mathbf{v}_{i_F+1}, \ldots, \mathbf{v}_k) \in \mathcal{M}^{k-i_F}$. The adversary *wins* if $\mathsf{Verify}(\mathsf{pk}, \tau^*, \mathbf{w}^*, \sigma^*, f^*) = 1$ and one of the following hold:

(1) $\tau^* \neq \tau_F$ for all filenames F queried by $\mathcal{A}$ (a *Type 1 forgery*),
(2) $\tau^* = \tau_F$ for filename F, f^* is well-defined on F, and $\mathbf{w}^* \neq f^*(\mathbf{V}_F)$ (a *Type 2 forgery*), or
(3) $\tau^* = \tau_F$ for filename F and f^* is not well-defined on F (a *Type 3 forgery*).

The *advantage* $\mathrm{HomSig\text{-}Adv}[\mathcal{A}, \mathcal{S}]$ of $\mathcal{A}$ is the probability that $\mathcal{A}$ wins the game.

For $t \in \{1,2,3\}$, we say that the scheme is *secure against type t forgeries* if the winning condition in Definition 2.2 is restricted to type t forgeries only. The proof of the following result can be found in the full version of this paper [12].

Proposition 2.3. *Let $\mathcal{H}$ be a linearly homomorphic signature scheme with message space $\mathcal{M} \subset R^n$ for some ring R. If $\mathcal{H}$ is secure against Type 2 forgeries, then $\mathcal{H}$ is secure against Type 3 forgeries.*

Privacy. In addition to the unforgeability property described above, one may wish homomorphic signatures to be *private*, in the sense that a derived signature on $m' = f(m_1, \ldots, m_k)$ reveals nothing about the values of the m_i beyond what can be ascertained from the values of m' and f. We discuss this property in the full version of this paper [12].

3 Building Blocks

Pre-homomorphic Signatures. Our generic conversion applies to "hash-and-sign" signatures with a specific form. Namely, a signature on a message m with randomness r must have a component $g^{f(m,r)}$, where g is some fixed generator of a cyclic group $\mathbb{G}$ and f is some function that may depend on the secret key. Furthermore, if we are given a valid signature on m with randomness r, then given x and y there is an efficient algorithm that tests whether $y = x^{f(m,r)}$.

Definition 3.1. Let $\mathcal{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme. Let $\mathcal{M}$ be the space of messages and $\mathcal{R}$ be the space of randomness sampled by the signing algorithm. We say that $\mathcal{S}$ is *pre-homomorphic* if the following three conditions hold for each key pair $(\mathsf{pk}, \mathsf{sk})$ output by KeyGen:

1. There is a finite cyclic group $\mathbb{G}$ such that Sign defines a map $\mathsf{Sign}_{\mathsf{sk}}\colon \mathcal{M} \times \mathcal{R} \to \mathbb{G} \times \{0,1\}^*$, where $\mathcal{M}$ is the message space and $\mathcal{R}$ is the space of randomness used by Sign. We decompose a signature σ as (σ_1, σ_2) with $\sigma_1 \in \mathbb{G}$, and we allow the σ_2 component to be empty.
2. The public key pk contains a generator g of the group $\mathbb{G}$ in (1), and there is an efficiently computable function $f_{\mathsf{sk}}\colon \mathcal{M} \times \mathcal{R} \to \mathbb{Z}$ such that for each signature $(\sigma_1, \sigma_2) \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m, r)$, we have $\sigma_1 = g^{f_{\mathsf{sk}}(m,r)}$
3. There is an efficient algorithm $\mathsf{Test}(\mathsf{pk}, m, \sigma, x, y)$ that takes input the public key pk, a message $m \in \mathcal{M}$, a signature $\sigma = (\sigma_1, \sigma_2)$, and group elements $x, y \in \mathbb{G}'$ for some group $\mathbb{G}'$ of the same order as $\mathbb{G}$. Suppose $\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1$. Then the algorithm outputs 1 if and only if $\log_g(\sigma_1) = \log_x(y)$; otherwise, the algorithm outputs 0. (If $\mathsf{Verify}(\mathsf{pk}, m, \sigma) \neq 1$ then the algorithm's output is unspecified.)

Homomorphic Hashing. A *homomorphic hash* is a linear function that maps vectors defined over some ring R to elements of some finite group $\mathbb{G}$. The ring R is interpreted as "exponents" of the group $\mathbb{G}$; the following definition makes this concept precise.

Definition 3.2. Let $\mathbb{G}$ be a finite cyclic group, R be a ring, and $\phi\colon R \to \mathbb{Z}$ be an injective function. We say (R, ϕ) is a *ring of exponents for* $\mathbb{G}$ if $\phi(r) \bmod |\mathbb{G}|$ defines a ring homomorphism from R to $\mathbb{Z}_{|\mathbb{G}|}$.

We shall assume from now on that the map ϕ is understood, in which case we say R itself is a ring of exponents for $\mathbb{G}$ and we identify R with its image under ϕ. In particular, for $g \in \mathbb{G}$ and $r \in R$, we interpret g^r to mean $g^{\phi(r)}$.

While Definition 3.2 is abstract, it is very concrete in our two principal examples:

- If $\mathbb{G}$ is a cyclic group of order p and ϕ is the map that lifts elements of $\mathbb{F}_p$ to integer representatives in $[0, p-1]$, then $(\mathbb{F}_p, \phi)$ is a ring of exponents for $\mathbb{G}$.
- If $\mathbb{G}$ is *any* finite cyclic group and ϕ is the identity map on $\mathbb{Z}$, then $(\mathbb{Z}, \phi)$ is a ring of exponents for $\mathbb{G}$. (In our constructions $\mathbb{G}$ will be a cyclic subgroup of $\mathbb{Z}_N^*$.)

In both cases our interpretation of g^r for $r \in R$ agrees with standard usage.

We now define the homomorphic hash used in our conversion. Our definition incorporates, in a single abstract framework, the homomorphic hash from previous linearly homomorphic signatures using discrete log groups [21,10,5,2] as well as the RSA-based construction of Gennaro et al. [14].

Definition 3.3. Let $\mathbb{G}$ be a finite cyclic group and let R be a ring of exponents for $\mathbb{G}$. For any positive integer n, define the following algorithms:

$\mathsf{HomHash.Setup}(\mathbb{G}, n)$: Choose random elements $h_1, \ldots, h_n \xleftarrow{\mathrm{R}} \mathbb{G}$ and output $\mathsf{hk} = (h_1, \ldots, h_n)$.

$\mathsf{HomHash.Eval}(\mathsf{hk}, \mathbf{v})$**:** Given a key $\mathsf{hk} = (h_1, \ldots, h_n)$ and a vector $\mathbf{v} = (v_1, \ldots, v_n) \in R^n$, output $\prod_{j=1}^{n} h_j^{v_j}$.

For a fixed value of hk, we define $H_{\mathsf{hom}}\colon R^n \to \mathbb{G}$ by $H_{\mathsf{hom}}(\mathbf{v}) = \mathsf{HomHash.Eval}(\mathsf{hk}, \mathbf{v})$.

As the name implies, the key property of HomHash is that it is homomorphic: for $\mathbf{v}, \mathbf{w} \in R^n$ and $a, b \in R$,

$$H_{\mathsf{hom}}(\mathbf{v})^a \cdot H_{\mathsf{hom}}(\mathbf{w})^b = \left(\prod_{j=1}^n h_j^{v_j}\right)^a \cdot \left(\prod_{j=1}^n h_j^{w_j}\right)^b = \prod_{j=1}^n h_j^{av_j+bw_j} = H_{\mathsf{hom}}(a\mathbf{v}+b\mathbf{w}).$$

(In the middle equality we have used the homomorphic property of Definition 3.2.)

Uniform Sampling. To sample uniformly random elements of $\mathbb{G}$, we raise a generator to a random exponent. The following definition captures the property this exponent needs to have.

Definition 3.4. Let $\mathbb{G}$ be a finite cyclic group and (R, ϕ) be a ring of exponents for $\mathbb{G}$. We say a distribution χ on R is $\mathbb{G}$-*uniform* if:

1. For $x \xleftarrow{\mathrm{R}} \chi$, the distribution of $g^{\phi(x)}$ is statistically closeto the uniform distribution on $\mathbb{G}$; and
2. If the order of $\mathbb{G}$ is not efficiently computable, then for $x \xleftarrow{\mathrm{R}} \chi$, the distribution of $\phi(x) \bmod e$ is statistically close to the uniform distribution on $\mathbb{Z}_e$ for all $e \in [|\mathbb{G}|/16, |\mathbb{G}|]$.

If $R = \mathbb{Z}_p$ and $\mathbb{G}$ is a group of (known) order p, we can take χ to be the uniform distribution on R. If $R = \mathbb{Z}$ and $\mathbb{G}$ is the multiplicative group of nonzero squares mod $N = pq$, we can take χ to be the uniform distribution on $[0, a]$ for any $a \gg |\mathbb{G}|$. To obtain a statistical distance of at most 2^{-m}, it suffices to take $a = N \cdot 2^m$.

Chameleon Hashing. As defined by Krawczyk and Rabin [20], a *chameleon hash function* is a function C that takes two inputs: a message m and randomness s. It is collision-resistant and has the additional property that there is a "trapdoor" that allows collisions to be computed.

To show unforgeability of our homomorphic signature scheme (as opposed to weak unforgeability) we will embed a "homomorphic" chameleon hash function C.Since the underlying messages are vectors, the randomness will be an additional vector component s, and we define $C(\mathbf{v}, s) = H_{\mathsf{hom}}(\mathbf{v}) \cdot u^s$ for some fixed (public) $u \in \mathbb{G}$. Note that (up to relabeling) this is simply H_{hom} applied to the $(n+1)$-dimensional vector $(\mathbf{v}, s)$.

Let us a try a first attempt at embedding a "trapdoor" in the homomorphic hash. We can generate hk and u such that we know discrete logs of the h_i and u to some base g; e.g., $h_i = g^{\beta_i}$, $u = g^{\eta}$. When $\mathbb{G}$ has prime order p, to evaluate C we can choose a uniformly random $s \in \mathbb{Z}_p$, and to hit a fixed target $C(\mathbf{v}, s) = g^a$ we simply compute s in $\mathbb{Z}_p$ such that $\langle \vec{\beta}, \mathbf{v} \rangle + \eta s = a$. Since this s is unique, the distribution of s conditioned on $(\mathbf{v}, C(\mathbf{v}, s) = g^a)$ is the same in both cases.

However, if $\mathbb{G}$ is a group of unknown order then this attempt fails. To begin, we cannot sample s from the uniform distribution on $\mathbb{Z}_{|\mathbb{G}|}$; in addition, we can't invert in the exponent to compute s. To get around these obstacles, we choose s from the distribution that the simulator in our security proof needs to sample (see Section 5) and we set $\eta = 1$. Specifically, the trapdoor information is $\beta_1, \ldots, \beta_n$ and $\delta_1, \ldots, \delta_k$ sampled from a $\mathbb{G}$-uniform distribution χ (Definition 3.4). To produce a signature on the ith file vector $\mathbf{v}$, the simulator uses the trapdoor to set $s = \delta_i + \langle \vec{\beta}, \mathbf{v} \rangle$. Thus in the "forward" direction we compute a random s by sampling δ_i and β_j from the same distribution χ.

More precisely, s is chosen from the following distribution:

Definition 3.5. Let χ be a $\mathbb{G}$-uniform distribution on R and $\mathbf{v} \in R^n$ be a vector. Let $F : \mathcal{K} \times \{0,1\}^\lambda \times \mathbb{Z} \to R$ be a pseudorandom function whose outputs are indistinguishable from samples from χ. For a fixed key $\mu \in \mathcal{K}$, define the distribution $\Xi_{\tau,\mathbf{v}}$ on R as follows:

1. Compute $\beta_j \leftarrow F_\mu(\tau, j)$ for $j = 1, \ldots, n$.
2. Sample $\delta \leftarrow \chi$.
3. Output $\delta + \langle \vec{\beta}, \mathbf{v} \rangle$.

(The distribution $\Xi_{\tau,\mathbf{v}}$ depends on μ, but we suppress this in the notation for readability.)

Since our simulator only needs to evaluate the chameleon hash for one file, it does not need to reuse the values of δ, so we can choose a new uniform δ each time. Note that if R is finite and χ is the uniform distribution on R, then $\Xi_{\tau,\mathbf{v}}$ is the uniform distribution on R. In particular, the distribution does not depend on the PRF F, so we have recovered our "first attempt" above.

4 A Generic Conversion

Let $\mathcal{S} = (\mathcal{S}.\mathsf{KeyGen}, \mathcal{S}.\mathsf{Sign}, \mathcal{S}.\mathsf{Verify})$ be a pre-homomorphic signature scheme. Define a homomorphic signature scheme $\mathsf{HomSig}(\mathcal{S})$ as follows:

$\mathsf{HomSig}(\mathcal{S}).\mathsf{Setup}(1^\lambda, k, n)$: On input a security parameter λ, a maximum data set size k, and a dimension n, do the following:

1. Compute $(\mathsf{pk}_\mathcal{S}, \mathsf{sk}_\mathcal{S}) \leftarrow \mathcal{S}.\mathsf{KeyGen}(1^\lambda)$. Let $\mathbb{G}, \mathbb{G}'$ be the groups in Definition 3.1 and let R be a ring of exponents for $\mathbb{G}$.
   (In our instantitaions, we use $R = \mathbb{F}_p$ if $\mathbb{G}$ has order p, and $R = \mathbb{Z}$ if $\mathbb{G} \subset \mathbb{Z}_N^*$.)
2. If the order of $\mathbb{G}$ is efficiently computable from pk_S, set $B_1 = B_2 = |\mathbb{G}|$. Otherwise, choose B_1, B_2 such that $kB_1B_2 < |\mathbb{G}|/32$. (We assume that a lower bound on $|\mathbb{G}|$ can be efficiently computed.)
3. Compute $\mathsf{hk} \leftarrow \mathsf{HomHash.Setup}(\mathbb{G}', n)$.
4. Choose random $t_1, \ldots, t_k, u \xleftarrow{\mathrm{R}} \mathbb{G}'$.
5. Choose a pseudorandom function $\Psi : \mathcal{K} \times \{0,1\}^\lambda \to \mathcal{R}$, where $\mathcal{R}$ is the space of randomness sampled by $\mathcal{S}.\mathsf{Sign}$, and choose a random key $\kappa \xleftarrow{\mathrm{R}} \mathcal{K}$.[1]
6. Choose a pseudorandom function $F : \mathcal{K}' \times \{0,1\}^\lambda \times \mathbb{Z} \to R$, and choose a random key $\mu \xleftarrow{\mathrm{R}} \mathcal{K}'$.
7. Output the public key $\mathsf{pk} = (\mathsf{pk}_\mathcal{S}, \mathsf{hk}, \{t_i\}_{i=1}^k, u, R, B_1, B_2)$ and the secret key $\mathsf{sk} = (\mathsf{sk}_\mathcal{S}, \Psi, \kappa, F, \mu, \mathsf{pk})$.

– The message space is $\mathcal{M} = \{\mathbf{v} \in R^n : \|\mathbf{v}\| \le B_1\}$, where we define $\|\mathbf{v}\| = \max_j\{|v_j|\}$. (Recall that we are identifying R with a subset of $\mathbb{Z}$ as remarked after Definition 3.2. If $|\mathbb{G}|$ is efficiently computable, then $\mathcal{M}$ is all of R^n.)

[1] If $\mathcal{S}.\mathsf{Sign}$ is deterministic, then we do not need the PRF Ψ.

– We represent an R-linear function $f : R^n \to R$ as a k-tuple of elements of R; specifically, the function $f(\mathbf{v}_1, \ldots, \mathbf{v}_k) = \sum c_i \mathbf{v}_i$ is represented by the vector $(c_1, \ldots, c_k) \in R^k$. We define $\|f\| = \max_i\{|c_i|\}$.
– The set of admissible functions $\mathcal{F}$ is all R-linear functions on k-tuples of vectors in R^n with $\|f\| \leq B_2$. (Note that when $R = \mathbb{Z}_{|\mathbb{G}|}$ this is all R-linear functions from $(R^n)^k$ to R.)
– We use $H_{\mathsf{hom}}(\mathbf{v})$ to denote $\mathsf{HomHash.Eval}(\mathsf{hk}, \mathbf{v})$.

$\mathsf{HomSig}(\mathcal{S}).\mathsf{Sign}(\mathsf{sk}, \tau, \mathbf{v}, i)$: On input a secret key sk, a tag $\tau \in \{0,1\}^\lambda$, a vector $\mathbf{v} \in R^n$, and an index $i \in \{1, \ldots, k\}$, do the following:

1. Compute $r \leftarrow \Psi_\kappa(\tau)$.
2. Compute $(\sigma_1, \sigma_2) \leftarrow \mathcal{S}.\mathsf{Sign}(\mathsf{sk}_\mathcal{S}, \tau, r)$.
3. Using the PRF F, choose $s \leftarrow \Xi_{\tau,\mathbf{v}}$ (Definition 3.5).
   If $|\mathbb{G}|$ is known, this is equivalent to choosing $s \xleftarrow{\mathrm{R}} \mathbb{Z}_{|\mathbb{G}|}$.
4. Compute $\sigma_3 \leftarrow \left(t_i \cdot H_{\mathsf{hom}}(\mathbf{v}) \cdot u^s\right)^{f_{\mathsf{sk}}(\tau,r)}$, where f_{sk} is the function in Definition 3.1 (2).
5. Output $\sigma = (\sigma_1, \sigma_2, \sigma_3, s)$.

$\mathsf{HomSig}(\mathcal{S}).\mathsf{Verify}(\mathsf{pk}, \tau, \mathbf{w}, \sigma, f)$: On input a public key pk, a tag $\tau \in \{0,1\}^\lambda$, a vector $\mathbf{w} \in R^n$, a signature $\sigma = (\sigma_1, \sigma_2, \sigma_3, s)$, and a function $f = (c_1, \ldots, c_k)$, do the following:

1. Compute $\zeta_1 \leftarrow \mathcal{S}.\mathsf{Verify}(\mathsf{pk}_\mathcal{S}, \tau, (\sigma_1, \sigma_2))$.
2. Let $x \leftarrow (\prod_{i=1}^k t_i^{c_i}) \cdot H_{\mathsf{hom}}(\mathbf{w}) \cdot u^s$ and compute $\zeta_2 \leftarrow \mathsf{Test}\big(\mathsf{pk}_\mathcal{S},\ \tau,\ (\sigma_1, \sigma_2),\ x,\ \sigma_3\big)$, where Test is the algorithm from Definition 3.1 (3).
3. If $\|\mathbf{w}\| \leq kB_1B_2$, set $\zeta_3 = 1$; otherwise set $\zeta_3 = 0$.
4. If $\zeta_1 = \zeta_2 = \zeta_3 = 1$, output 1; otherwise output 0.

$\mathsf{HomSig}(\mathcal{S}).\mathsf{Eval}(\mathsf{pk}, \tau, f, \vec{\sigma})$: On input a public key pk, a tag $\tau \in \{0,1\}^\lambda$, a function $f = (c_1, \ldots, c_k)$, and a vector of signatures $\vec{\sigma} = (\sigma^{(1)}, \ldots, \sigma^{(k)})$ where $\sigma^{(i)} = (\sigma_1^{(i)}, \sigma_2^{(i)}, \sigma_3^{(i)}, s^{(i)})$, do the following:

1. Compute $\sigma_3' \leftarrow \prod_{i=1}^k \left(\sigma_3^{(i)}\right)^{c_i}$, $\quad s' \leftarrow \sum_{i=1}^k c_i s^{(i)}$.
2. Output $\sigma' = (\sigma_1^{(1)}, \sigma_2^{(1)}, \sigma_3', s')$.

Lemma 4.1 (Proof in full version [12].). *The homomorphic signature scheme* $\mathsf{HomSig}(\mathcal{S})$ *satisfies the correctness properties of Definition 2.1.*

5 Security

Recall that an adversary can break a homomorphic signature scheme by computing any of three types of forgeries in Definition 2.2. By Proposition 2.3, a Type 3 forgery in a linearly homomorphic scheme implies a Type 2 forgery. In our security analysis we consider the remaining two types separately. We further split Type 2 into two subtypes. In a Type 2 forgery for $\mathsf{HomSig}(\mathcal{S})$, the adversary outputs a forged signature $(\sigma_1^*, \sigma_2^*, \sigma_3^*, s^*)$ and a tag τ^* equal to one of the tags τ_ℓ returned from a previous query. By our construction of Eval, any signature derived from the queried signatures corresponding to τ_ℓ will have the same σ_1 and σ_2 components as in the queried signatures. This motivates the following definition:

– *Type 2a*: The pair (σ_1^*, σ_2^*) output by the adversary is *not* equal to $(\sigma_1, \sigma_2) \leftarrow \mathcal{S}.\mathsf{Sign}(\mathsf{sk}_{\mathcal{S}}, \tau^*, r^*)$ computed by the challenger. (Here $r^* = \Psi_\kappa(\tau^*)$.)
– *Type 2b*: The pair (σ_1^*, σ_2^*) output by the adversary *is* equal to $(\sigma_1, \sigma_2) \leftarrow \mathcal{S}.\mathsf{Sign}(\mathsf{sk}_{\mathcal{S}}, \tau^*, r^*)$ computed by the challenger.

Type 1, 2a Forgeries. We show that Type 1 forgery in our homomorphic scheme $\mathsf{HomSig}(\mathcal{S})$ leads to a forgery of the underlying signature scheme $\mathcal{S}$; i.e., a valid signature on a previously unseen message. In addition, a Type 2a forgery leads to a *strong* forgery of the underlying signature scheme, i.e., a *new* valid signature on a previously queried message. Since the underlying scheme $\mathcal{S}$ is used to sign random messages chosen by the challenger, we only require that $\mathcal{S}$ be unforgeable against a weak adversary.

Theorem 5.1. *If $\mathcal{S}$ is strongly unforgeable against a weak adversary and Ψ is a secure PRF, then $\mathsf{HomSig}(\mathcal{S})$ is secure against Type 1 and Type 2a forgeries.*

Sketch of Proof. We simulate the public key for $\mathsf{HomSig}(\mathcal{S})$ using the public key for $\mathcal{S}$ and elements $t_i = g^{\gamma_i}$, $h_j = g^{\alpha_j}$, and $u = g^{\delta}$ with known discrete logarithms. To sign vector $\mathbf{v}_i$, we query τ to the $\mathcal{S}$ challenger to obtain (σ_1, σ_2), and we compute $\sigma_3 = \sigma_1^{\gamma_i + \langle \vec{\alpha}, \mathbf{v}_i \rangle + \delta s}$ for $s \xleftarrow{\mathrm{R}} \Xi_{\tau, \mathbf{v}_i}$. Given a Type 1 or Type 2a forgery $(\tau^*, \mathbf{w}^*, f^*, \sigma^*$, the (σ_1, σ_2) component of σ^* is a valid forgery for $\mathcal{S}$ on the message τ^*. □

Type 2b Forgeries. In this case we do not have a black-box reduction to the underlying signature scheme. However, we do not have to prove each instance separately, as we can abstract out properties of the underlying scheme's security proof — or more specifically, of the simulator used in the reduction — that allow our reduction to go through. These properties are captured in the following definition:

Definition 5.2. Let $\mathcal{S}$ be a pre-homomorphic signature scheme and $\mathcal{P}$ be a computational problem. We say that $\mathcal{S}$ is *δ-simulatable and γ-extractable for $\mathcal{P}$* if there is a simulator Sim that takes an instance I of $\mathcal{P}$, interacts with a signature adversary $\mathcal{A}$ that makes at most q message queries, and has the following properties:

1. The probability that Sim aborts is at most $1 - \delta$.
2. Conditioned on Sim not aborting, the public key $\mathsf{pk}_{\mathcal{S}}$ produced by Sim is statistically indistinguishable from a real public key for $\mathcal{S}$.
3. Conditioned on Sim not aborting and for any public key $\mathsf{pk}_{\mathcal{S}}$, the signatures produced by Sim are statistically indistinguishable from real signatures produced by $\mathcal{S}$.
4. Let $(\sigma_1^{(\ell)}, \sigma_2^{(\ell)})$ be the signature produced by Sim on the ℓth message query, and let $\omega_\ell = \log_g(\sigma_1^{(\ell)})$. (If Sim simulates signatures perfectly, then $\omega_\ell = f_{\mathsf{sk}}(m_\ell, r_\ell)$ for implicit randomness r_ℓ.) Then Sim can efficiently compute generators x and y of $\mathbb{G}'$ such that
   – Sim can efficiently compute x^{ω_ℓ} for all ℓ;
   – Sim can efficiently compute y^{ω_ℓ} for all $\ell \neq \ell^*$, where ℓ^* is a value in $1, \ldots, q$ randomly chosen by Sim.

5. For y and ℓ^* as above, there is an efficient algorithm $\mathsf{Extract}$ that given an integer b, a value $z = y^{b \cdot \omega_{\ell^*}}$, and the internal state of Sim, outputs either $\perp$ or a solution to the instance I of $\mathcal{P}$. Furthermore, if the distribution of b is $\mathbb{G}$-uniform, then the probability (over the instances of $\mathcal{P}$ and the random coins of Sim) that $\mathsf{Extract}$ outputs $\perp$ is at most $1 - \gamma$.

Theorem 5.3. *Suppose $\mathcal{S}$ is a δ-simulatable, γ-extractable pre-homomorphic signature scheme for $\delta, \gamma \geq 1/\operatorname{poly}(\lambda)$. If there is no efficient algorithm to solve problem $\mathcal{P}$ in the group $\mathbb{G}$ and Ψ and F are secure PRFs, then $\mathsf{HomSig}(\mathcal{S})$ is secure against Type 2b forgeries.*

Proof. We describe an algorithm $\mathcal{B}$ that takes an instance I of problem $\mathcal{P}$ and interacts with an adversary $\mathcal{A}$ in the unforgeability game for $\mathsf{HomSig}(\mathcal{S})$. $\mathcal{B}$ runs as follows:

Setup: $\mathcal{B}$ does the following:

1. Run Sim on instance I to generate a (simulated) public key $\mathsf{pk}_{\mathcal{S}}$ and elements $x, y \in \mathbb{G}'$ and $\ell^* \in \{1, \ldots, q\}$ as in Definition 5.2; abort if Sim aborts.
2. Let R, B_1, B_2 be as in $\mathsf{HomSig}(\mathcal{S}).\mathsf{Setup}$ and let χ be a $\mathbb{G}'$-uniform distribution on R.
3. For $j = 1, \ldots, n$, choose $\alpha_j, \beta_j \xleftarrow{\mathrm{R}} \chi$ and set $h_j \leftarrow x^{\alpha_j} y^{-\beta_j}$. Let $\vec{\alpha}, \vec{\beta}$ be the vectors of α_j and β_j, respectively, and let $\mathsf{hk} = (h_1, \ldots, h_n)$.
4. For $i = 1, \ldots, k$, choose $\gamma_i, \delta_i \xleftarrow{\mathrm{R}} \chi$ and set $t_i \leftarrow x^{\gamma_i} y^{-\delta_i}$. Let $\vec{\gamma}, \vec{\delta}$ be the vectors of γ_i, δ_i, respectively.
5. Choose $\eta \xleftarrow{\mathrm{R}} \chi$ and set $u = x^{\eta} y$.
6. Choose random tags $\tau_1, \ldots, \tau_\ell \xleftarrow{\mathrm{R}} \{0,1\}^\lambda$, and abort if $\tau_i = \tau_j$ for $i \neq j$. Initialize an empty array A and counters $c_\ell = 1$ for $\ell = 1, \ldots, q$.
7. Send $\mathcal{A}$ the public key $\mathsf{pk} = (\mathsf{pk}_{\mathcal{S}}, \mathsf{hk}, \{t_i\}_{i=1}^k, u, R, B_1, B_2)$.

Queries: When $\mathcal{A}$ makes a query for filename $F \in \{0,1\}^*$ and a vector $\mathbf{v} \in R^n$, $\mathcal{B}$ does the following:

1. If F is not in the array A, append F to A. Let ℓ be the index of F in A and let $i = c_\ell$. If $c_\ell = 1$, send the tag τ_ℓ to the adversary.
2. Run Sim to produce (simulated) $\mathcal{S}$ signatures $(\sigma_1^{(\ell)}, \sigma_2^{(\ell)})$ on the message τ_ℓ, using (perhaps implicit) randomness r_ℓ; abort if Sim aborts.
3. If $\ell \neq \ell^*$, choose $s^{(\ell,i)} \xleftarrow{\mathrm{R}} \Xi_{\ell,\mathbf{v}}$ (Definition 3.5).
   If $\ell = \ell^*$, set $s^{(\ell,i)} = \delta_i + \langle \vec{\beta}, \mathbf{v} \rangle$
4. Compute the third component of $\mathsf{Sign}(\mathsf{sk}, \tau_\ell, \mathbf{v}, i)$ as

$$\sigma_3^{(\ell,i)} \leftarrow (t_i \cdot H_{\mathsf{hom}}(\mathbf{v}_i) \cdot u^s)^{\omega_\ell} = \left(x^{\gamma_i + \langle \vec{\alpha}, \mathbf{v}_i \rangle + \eta s^{(\ell,i)}} y^{s^{(\ell,i)} - \delta_i - \langle \vec{\beta}, \mathbf{v}_i \rangle} \right)^{\omega_\ell}$$

   Property 4 of Definition 5.2 implies that we can efficiently compute this value for all ℓ. (Note that when $\ell = \ell^*$, there is no y term due to our choice of s.)
5. Send the signature $\sigma^{(\ell,i)} = (\sigma_1^{(\ell)}, \sigma_2^{(\ell)}, \sigma_3^{(\ell,i)}, s^{(\ell,i)})$ to the adversary.
6. Set $c_\ell \leftarrow c_\ell + 1$.

Forgery: When $\mathcal{A}$ outputs a Type 2b forgery $(\tau^*, \mathbf{w}^*, \sigma^*, f^*)$ with f^* represented by $\mathbf{c} = (c_1, \ldots, c_k)$ and $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, s^*)$, $\mathcal{B}$ does the following:

1. If $\tau^* \neq \tau_{\ell^*}$, abort.
2. Let $\mathbf{v}_1, \ldots, \mathbf{v}_k$ be the vectors queried with tag τ^*. Compute

$$a = \langle \vec{\gamma}, \mathbf{c} \rangle + \langle \vec{\alpha}, \mathbf{w}^* \rangle + \eta s^*, \quad b = -\langle \vec{\delta}, \mathbf{c} \rangle - \langle \vec{\beta}, \mathbf{w}^* \rangle + s^*, \quad z = \sigma_3^* / x^{a \cdot \omega_{\ell^*}}.$$

   Property 4 of Definition 5.2 implies that we can efficiently compute z.
3. Run $\mathsf{Extract}(b, z, \mathsf{Sim})$ and output the result.

In the full version of this paper [12], we analyze the simulation using a series of games; here we give a sketch of the analysis.

Let W_0 be the event that $\mathcal{A}$ wins the unforgeability game when interacting with a real challenger for $\mathsf{HomSig}(\mathcal{S})$ and W_1 be the event that $\mathcal{A}$ wins when interacting with our simulator. Then we can show that under the hypotheses of the theorem statement, $\Pr[W_1] \geq \frac{\delta}{q} \cdot \Pr[W_0] - \epsilon$ for some negligible ϵ. (The δ factor represents the simulator not aborting, and the $1/q$ factor reflects the simulator guessing the correct ℓ^*.)

If W_1 occurs, then the fact that the forgery is a valid signature for the tag τ_{ℓ^*} implies that the element z computed in the forgery is equal to $y^{b \cdot \omega_{\ell^*}}$. Under the assumption that b is $\mathbb{G}'$-uniform, property (5) of Definition 5.2 implies that $\mathcal{B}$ outputs a solution to the instance I of problem $\mathcal{P}$ with probability at least γ, which completes the proof.

It remains only to show that b is $\mathbb{G}'$-uniform. Let $\mathbf{y} = \sum c_i \mathbf{v}_i - \mathbf{w}^* \in R^n$ and let $\hat{s} = \sum_{i=1}^{k} c_i s^{(\ell^*, i)}$. It follows from our construction of the $s^{(\ell^*, i)}$ that $b = \langle \vec{\beta}, \mathbf{y} \rangle + s^* - \hat{s}$. Since the property of being $\mathbb{G}'$-uniform is invariant under translation by a scalar, it suffices to show that (1) $\mathbf{y} \neq 0 \bmod |\mathbb{G}|$, and (2) the vector $\vec{\beta}$ comes from a distribution statistically close to χ^n even when conditioned on the adversary's view. Property (1) follows from the fact that $\mathcal{A}$ outputs a Type 2b forgery, while property (2) can be verified by looking at the information about $\vec{\alpha}, \vec{\beta}, \vec{\gamma}, \vec{\delta}$ available to the adversary. □

6 Example Instantiation: Boneh-Boyen Signatures

We now describe how our construction can be instantiated using the signatures of Boneh and Boyen [4]; we describe additional instantiations in the full version of this paper [12].

Let $\mathbb{G}, \mathbb{G}_T$ be group of prime order p with an efficiently computable, nondegenerate bilinear map $\hat{e} \colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. (For simplicity we assume here that the pairing is symmetric; in the full version we consider a general pairing $\hat{e} \colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.) The signature scheme BB consists of the following algorithms:

BB.Setup: Choose random $\alpha \xleftarrow{R} \mathbb{Z}_p$ and $g \xleftarrow{R} \mathbb{G}$. The public key is $\mathsf{pk} = (g, g^\alpha)$, and the secret key is $\mathsf{sk} = \alpha$.

BB.Sign: Given a message $m \in \mathbb{Z}_p$, output $\sigma = g^{1/(\alpha + m)}$.

BB.Verify: Output 1 if $\hat{e}(\sigma, g^m \cdot g^\alpha) = \hat{e}(g, g)$; otherwise output 0.

An instance of the *q-strong Diffie-Hellman problem* is a tuple $(g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^q})$ for randomly chosen $g \xleftarrow{R} \mathbb{G}$ and $\alpha \xleftarrow{R} \mathbb{Z}_p$. A solution is a pair $(r, g^{1/(\alpha + r)}) \in \mathbb{Z}_p \times \mathbb{G}$.

Boneh and Boyen [4, Lemma 9] show that if the q-SDH assumption holds for $\mathbb{G}$, then BB is strongly unforgeable against a weak adversary making at most q signature queries.

The BB scheme is pre-homomorphic (Definition 3.1): the (deterministic) signing function is $f_{\mathsf{sk}}(m) = 1/(\alpha + m) \pmod p$, and we define $\mathsf{BB.Test}(\mathsf{pk}, m, \sigma, x, y)$ to output 1 if and only if $\hat{e}(\sigma, x) = \hat{e}(g_1, y)$ (regardless of the output of $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$).

We now describe the Boneh-Boyen simulator $\mathsf{Sim}_{\mathsf{BB}}$ that takes an instance $(g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^q})$ of the q-SDH problem and interacts with a weak signature adversary.

Setup: Given distinct messages $m_1, \ldots, m_q \in \mathbb{Z}_p$ queried by the adversary, form the polynomial $P(t) = \prod_{i=1}^{q}(t+m_i) \in \mathbb{Z}_p[t]$. Since $P(t)$ has degree at most q, we can use the q-SDH instance to compute $x = g^{P(\alpha)}$. Output the public key $\mathsf{pk} = (x, g^\alpha)$; the (implicit) secret key is α.

Signatures: Let $P_\ell(t) = \prod_{i \neq \ell}(t + m_i)$. The signature on m_ℓ is $\sigma^{(\ell)} = x^{1/(\alpha+m_\ell)} = g^{P_\ell(\alpha)}$, which can be computed from the q-SDH challenge.

Proposition 6.1. $\mathsf{Sim}_{\mathsf{BB}}$ *is* 1*-simulatable &* $(1 - \frac{1}{p})$*-extractable for the q-SDH problem.*

Proof. Conditions 1–3 of Definition 5.2 are obviously satisfied. To verify condition 4, let $P(t)$ and $P_\ell(t)$ be as above. Let $x = g^{P(\alpha)}$ and $y = g^{P_{\ell^*}(\alpha)}$. We have $\sigma^{(\ell)} = x^{1/\alpha+m_\ell}$, so $\omega_\ell = 1/(\alpha + m_\ell)$ and the simulator can compute x^{ω_ℓ} for all ℓ and y^{ω_ℓ} for all $\ell \neq \ell^*$.

To verify the last condition, write $P_{\ell^*}(t)/(t + m_{\ell^*}) = Q(t) + c/(t + m_{\ell^*})$ for some polynomial $Q \in \mathbb{Z}_p[t]$ of degree at most $q - 2$ and $c \in \mathbb{Z}_p$. Since the messages m_i are distinct, we have $c \neq 0$. Given an integer b and the element

$$z = y^{b \cdot \omega_{\ell^*}} = g^{b \cdot P^*(\alpha)/(\alpha+m_{\ell^*})} = g^{b \cdot (Q(\alpha) + c/(\alpha+m_{\ell^*}))},$$

we let $\mathsf{Extract}$ output $\left(m_{\ell^*}, (z^{1/b}/g^{Q(\alpha)})^{1/c}\right)$, or $\perp$ if $b = 0 \bmod p$. Thus for uniform b in $\mathbb{Z}_p$, $\mathsf{Extract}$ outputs a solution to the q-SDH problem with probability $1 - 1/p$. □

Corollary 6.2. *If the q-SDH assumption holds for* $\mathbb{G}$*, then* $\mathsf{HomSig}(\mathsf{BB})$ *is unforgeable.*

References

1. Ahlswede, R., Cai, N., Li, S., Yeung, R.: Network information flow. IEEE Transactions on Information Theory 46(4), 1204–1216 (2000)
2. Attrapadung, N., Libert, B.: Homomorphic network coding signatures in the standard model. In: Public Key Cryptography — PKC '11. Springer LNCS, vol. 6571, pp. 17–34 (2011)
3. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Advances in Cryptology — EUROCRYPT '04. Springer LNCS, vol. 3027, pp. 223–238 (2004)
4. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. J. Cryptology 21, 149–177 (2008), extended abstract in *Advances in Cryptology — EUROCRYPT '04*
5. Boneh, D., Freeman, D., Katz, J., Waters, B.: Signing a linear subspace: Signature schemes for network coding. In: Public-Key Cryptography — PKC '09. LNCS, vol. 5443, pp. 68–87. Springer (2009)

6. Boneh, D., Freeman., D.M.: Homomorphic signatures for polynomial functions. In: Paterson, K. (ed.) Advances in Cryptology — EUROCRYPT 2011. Springer LNCS, vol. 6632, pp. 149–168 (2011), full version available at http://eprint.iacr.org/2011/018
7. Boneh, D., Freeman, D.M.: Homomorphic signatures over binary fields and new tools for lattice-based signatures. In: Public Key Cryptography — PKC 2011. Springer LNCS, vol. 6571, pp. 1–16 (2011), full version available at http://eprint.iacr.org/2010/453
8. Catalano, D., Fiore, D., Warinschi, B.: Adaptive pseudo-free groups and applications. In: Paterson, K.G. (ed.) Advances in Cryptology — EUROCRYPT 2011. Springer LNCS, vol. 6632, pp. 207–223 (2011)
9. Catalano, D., Fiore, D., Warinschi, B.: Efficient network coding signatures in the standard model. Cryptology ePrint Archive, Report 2011/696 (2011), http://eprint.iacr.org/2011/696
10. Charles, D., Jain, K., Lauter, K.: Signatures for network coding. International Journal of Information and Coding Theory 1(1), 3–14 (2009)
11. Fischlin, M.: The Cramer-Shoup strong-RSA signature scheme revisited. In: Public Key Cryptography — PKC 2003. Springer LNCS, vol. 2567, pp. 116–129 (2003)
12. Freeman, D.M.: Improved security for linearly homomorphic signatures: A generic framework. Cryptology ePrint Archive, Report 2012/060 (2012), http://eprint.iacr.org/2012/060
13. Gennaro, R., Halevi, S., Rabin, T.: Secure hash-and-sign signatures without the random oracle. In: Advances in Cryptology — Eurocrypt 1999. LNCS, vol. 1592, pp. 123–139 (1999)
14. Gennaro, R., Katz, J., Krawczyk, H., Rabin, T.: Secure network coding over the integers. In: Public Key Cryptography — PKC '10. Springer LNCS, vol. 6056, pp. 142–160 (2010)
15. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Advances in Cryptology — ASIACRYPT 2002, Springer LNCS, vol. 2501, pp. 548–566. Springer, Berlin (2002)
16. Hofheinz, D., Kiltz, E.: Programmable hash functions and their applications. In: Advances in Cryptology — CRYPTO 2008. Springer LNCS, vol. 5157, pp. 21–38 (2008)
17. Hohenberger, S., Waters, B.: Realizing hash-and-sign signatures under standard assumptions. In: Advances in Cryptology — EUROCRYPT '09. Springer LNCS, vol. 5479, pp. 333–350 (2009)
18. Hohenberger, S., Waters, B.: Short and stateless signatures from the RSA assumption. In: Advances in Cryptology — CRYPTO '09. Springer LNCS, vol. 5677, pp. 654–670 (2009)
19. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Topics in Cryptology — CT-RSA 2002. Springer LNCS, vol. 2271, pp. 244–262 (2002)
20. Krawczyk, H., Rabin, T.: Chameleon signatures. In: Network and Distributed System Security Symposium (2000)
21. Krohn, M., Freedman, M., Mazières, D.: On-the-fly verification of rateless erasure codes for efficient content distribution. In: Proc. of IEEE Symposium on Security and Privacy. pp. 226–240 (2004)
22. Lewko, A.B., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Theory of Cryptography — TCC 2010. Springer LNCS, vol. 5978, pp. 455–479 (2010)
23. Li, S.Y.R., Yeung, R.W., Cai, N.: Linear network coding. IEEE Trans. Info. Theory 49(2), 371–381 (2003)
24. Waters, B.: Efficient identity-based encryption without random oracles. In: Advances in Cryptology — EUROCRYPT '05. Springer LNCS, vol. 3494, pp. 320–329 (2005)