An extended abstract of this work will appear in Public Key Cryptography — PKC 2012.
This is the full version.

Improved Security for Linearly Homomorphic Signatures: A Generic Framework[*]

DAVID MANDELL FREEMAN[†]
Stanford University, USA
dfreeman@cs.stanford.edu

March 9, 2012

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.

[*] Full version available as Cryptology ePrint Archive Report 2012/060.

[†] 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.

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?

Homomorphic signatures [JMSW02, BFKW09, GKKR10, BF11a, AL11, BF11b] 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 [JMSW02] and were adapted for the above application by Boneh, Freeman, Katz, and Waters [BFKW09], whose motivation was to authenticate packets in *network coding* protocols [ACLY00, LYC03]. 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 [BF11b, §2]..

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 [GKKR10] is based on RSA and authenticates linear functions on vectors over the integers, while the system of Boneh and Freeman [BF11a] is based on lattice assumptions and authenticates linear functions on vectors over small fields. In a recent breakthrough, Boneh and Freeman [BF11b] 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 [AL11], which is based on the Lewko-Waters identity-based encryption scheme [LW10] 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 [CFW11a], 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 following signature schemes have the required properties and thus give rise to secure homomorphic signatures:

- The scheme of Waters [Wat05], secure under the (co-)computational Diffie-Hellman asumption in bilinear groups.

- The scheme of Boneh and Boyen [BB08], secure under the q-strong Diffie-Hellman assumption in bilinear groups.
- The scheme of Gennaro, Halevi, and Rabin [GHR99], secure under the strong RSA assumption.
- The scheme of Hohenberger and Waters [HW09b], secure under the RSA assumption.

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 [AL11] or [CFW11a]. 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 [GS02]) 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 [BFKW09, GKKR10], 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 [AL11] and Catalano-Fiore-Warinschi [CFW11a] to work against our stronger adversary; in Appendix B 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 [AL11] to the Lewko-Waters signatures on which they are based [LW10]. 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

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

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

$$\begin{aligned}
\hat{e}(\sigma_1, \textstyle\prod h_i^{v_i + w_i}) &= \hat{e}(\sigma_1, \textstyle\prod h_i^{v_i}) \cdot \hat{e}(\sigma_1, \textstyle\prod h_i^{w_i}) && \text{(by bilinearity of } \hat{e}\text{)} \\
&= \hat{e}(\sigma_3, g) \cdot \hat{e}(\sigma_3', g) && \text{(by the verification property for } \sigma \text{ and } \tau\text{)} \\
&= \hat{e}(\sigma_3\sigma_3', g) && \text{(by bilinearity of } \hat{e}\text{).}
\end{aligned}$$

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 [KFM04], and the idea of signing such a hash is used in several previous constructions [BFKW09, GKKR10, BF11b]. 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 3.4.

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 [GS02], Boneh-Boyen [BB04], and Waters [Wat05], where signatures have the same general form as in the Lewko-Waters system, as well as that of Boneh-Boyen [BB08], 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 [GHR99] 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 [HW09b] 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. [BFKW09] for network coding and adapted to the more general homomorphic setting by Boneh and Freeman [BF11b], 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*). (See Section 2.1 for formal definitions.)

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 [GHR99, Fis03, HK08, HW09a, HW09b]: 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(Lemma A.1) , 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. Details of the simulators for our example systems appear in Section 6. In Appendix C 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 [CFW11b] 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. In our construction the σ_1 and σ_2 components are used to bind the file identifier to the signature; in [CFW11b] this is not necessary since the signing function already makes use of the file identifier. Signatures in [CFW11b] are thus 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 [GKKR10] require an upper bound on vector length.)

While the constructions in [CFW11b] 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 [HW09b] 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 [CFW11b] 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 Preliminaries

2.1 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 ([BF11b]). 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 $\pi_i(m_1, \dots, m_k) = m_i$ that projects onto the ith component. We require that $\pi_1, \dots, \pi_k \in \mathcal{F}$ 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 $i \in \{1, \dots, k\}$ be any index. If $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \tau, m, i)$, then with overwhelming probability
$\mathsf{Verify}(\mathsf{pk}, \tau, m, \sigma, \pi_i) = 1$.
2. Let $\tau \in \{0,1\}^\lambda$ be any tag, let $\vec{\mu} = (\mu_1, \dots, \mu_k) \in \mathcal{M}^k$ be any tuple of messages, let $\vec{\sigma} = (\sigma_1, \dots, \sigma_k) \in \Sigma^k$ 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 $(f_1, \dots, f_k, g) \in \mathcal{F}^{k+1}$ be any tuple of admissible functions. Let $g \circ \vec{f}$ denote the function that sends $\vec{x} = (x_1, \dots, x_k)$ to $g(f_1(\vec{x}), \dots, f_k(\vec{x}))$. If $\mathsf{Verify}(\mathsf{pk}, \tau, m_i, f_i) = 1$ for some $m_1, \dots, m_k \in \mathcal{M}$, the message $g(m_1, \dots, m_k)$ is in $\mathcal{M}$, and the function $g \circ \vec{f}$ is admissible, then with overwhelming probability
$$\mathsf{Verify}\Big(\mathsf{pk},\ \tau,\ g(\vec{m}),\ \mathsf{Eval}\big(\mathsf{pk}, \tau, g, \vec{\sigma}\big),\ g \circ \vec{f}\Big) = 1.$$

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 $i = 1, \dots, k$, 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 $f(\mathbf{v}_1, \dots, \mathbf{v}_k) = \sum c_k \mathbf{v}_i$ as the vector $(c_1, \dots, c_k) \in R^k$.

Relationship to Network Coding. Definition 2.1 generalizes the definition of Boneh, Freeman, Katz and Waters for signatures in *network coding* systems [BFKW09, Definition 1]. In network coding, a file is parsed as a set of vectors $\mathbf{v}'_1, \dots, \mathbf{v}'_k \in \mathbb{F}_p^n$. Each vector $\mathbf{v}'_i$ is then "augmented" by appending the ith unit vector $\mathbf{e}_i$, creating k "augmented vectors" $\mathbf{v}_1, \dots, \mathbf{v}_k \in \mathbb{F}_p^{n+k}$. 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 $f(x_1, \dots, x_k) = \sum_i c_i x_i$, then the "augmentation component" of $\mathbf{w} = f(\mathbf{v}_1, \dots, \mathbf{v}_k)$ (i.e., the last k entries) is exactly $(c_1, \dots, c_k)$. 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. [BF11b]).

2.2 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 [BF11b])**.** 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* HomSig-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. For a general homomorphic signature scheme, it may not be possible for the challenger to efficiently detect a type 3 forgery (i.e., determine whether the function f^*

output by the adversary is well-defined on F). However, for *linearly* homomorphic schemes, it is easy to detect such a forgery. As a result, we can convert an adversary that ouputs a Type 3 forgery into one that outputs a Type 2 forgery.

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.*

Proof. Let $\mathcal{A}$ be an adversary that attacks $\mathcal{H}$ and outputs a Type 3 forgery. We exhibit an algorithm $\mathcal{B}$ that outputs a Type 2 forgery. Algorithm $\mathcal{B}$ first runs algorithm $\mathcal{A}$ against the challenger. Let $(\tau^*, \mathbf{w}^*, \sigma^*, f^*)$ be the Type 3 forgery output by $\mathcal{A}$. Let $\mathbf{v}_1, \dots, \mathbf{v}_t$ be the queries made by $\mathcal{A}$ for the file corresponding to τ^*, and let $f^* = (c_1, \dots, c_k)$. There are two cases:

1. $\sum_{i=1}^{t} c_i \mathbf{v}_i \neq \mathbf{w}^*$. In this case $\mathcal{B}$ queries file τ^* with vectors $\mathbf{v}_i = 0$ for all $i > t$.
2. $\sum_{i=1}^{t} c_i \mathbf{v}_i = \mathbf{w}^*$. The fact that f^* is not well-defined on F means there is some $j > t$ such that $c_j \neq 0$. Then $\mathcal{B}$ chooses some nonzero vector $\mathbf{e}$, queries file τ^* with vectors $\mathbf{v}_i = 0$ for $i > t$, $i \neq j$, and queries vector $\mathbf{v}_j = \mathbf{e}$.

In both cases $\mathcal{B}$ outputs the same forgery as $\mathcal{A}$. We verify that this is a Type 2 forgery. In the first case this is immediate since $f^*(\mathbf{v}_1, \dots, \mathbf{v}_k) = \sum_{i=1}^{t} c_i \mathbf{v}_i \neq \mathbf{w}^*$. In the second case we have

$$f^*(\mathbf{v}_1, \dots, \mathbf{v}_k) = \textstyle\sum_{i=1}^{t} c_i \mathbf{v}_i + c_j \mathbf{e} = \mathbf{w}^* + \mathbf{e} \neq \mathbf{w}^*.$$

□

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 $m' = f(m_1, \dots, m_k)$ reveals nothing about the values of the m_i beyond what can be ascertained from the values of m' and f. In Appendix C we give the formal definition of this property, which was introduced by Boneh and Freeman [BF11a].

2.3 Background on Signatures and Complexity Assumptions

A *signature scheme* is a tuple $\mathcal{S}$ of three probabilistic polynomial-time algorithms $\mathcal{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. KeyGen takes a security parameter λ (in unary) and returns public and secret keys. Sign takes a secret key and a message and returns a signature. If Sign is randomized, we will often write the randomness explicitly as an additional input. Verify takes a public key, a message, and a signature, and returns 1 (accept) or 0 (reject). We require that for every $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, if $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ then $\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1$. In all of our examples we assume that the message space is $\{0,1\}^\lambda$; one can expand the message space by first applying a collision-resistant hash function $F \colon \{0,1\}^* \to \{0,1\}^\lambda$ to the message.

The security of a signature scheme $\mathcal{S}$ is captured in the following "unforgeability game" between a challenger and an adversary $\mathcal{A}$:

- The challenger computes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and sends pk to $\mathcal{A}$.
- Proceeding adaptively, $\mathcal{A}$ submits messages m_i to the challenger and receives signatures $\sigma_i = \mathsf{Sign}(\mathsf{sk}, m_i)$
- $\mathcal{A}$ outputs a message m^* and a signature σ^*.

We say that (m^*, σ^*) is a *forgery* if $\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = 1$ and $m^* \neq m_i$ for all i. We say that (m^*, σ^*) is a *strong forgery* if $\mathsf{Verify}(\mathsf{pk}, m^*, \sigma^*) = 1$ and $(m^*, \sigma^*) \neq (m_i, \sigma_i)$ for all i. (A forgery is a valid signature on a previously unseen message; a strong forgery is a *new* signature on *any* message.)

We define $\text{Sig-Adv}[\mathcal{A}, \mathcal{S}]$ to be the probability that $\mathcal{A}$ outputs a forgery, and we define $\text{Sig}_\text{s}\text{-Adv}[\mathcal{A}, \mathcal{S}]$ to be the probability that $\mathcal{A}$ outputs a strong forgery. We say that $\mathcal{S}$ is *unforgeable* (respectively, *strongly unforgeable*) if $\text{Sig-Adv}[\mathcal{A}, \mathcal{S}]$ (repsectively, $\text{Sig}_\text{s}\text{-Adv}[\mathcal{A}, \mathcal{S}]$) is negligible in the security parameter λ for all polynomial-time adversaries $\mathcal{A}$. Note that if $\mathcal{S}$ has the property that each message has a unique signature, then unforgeable and strongly unforgeable are equivalent.

We also consider a weaker notion of security, captured in the following "weak unforgeability game":

- $\mathcal{A}$ submits messages $m_1, \ldots, m_q$ to the challenger.
- The challenger computes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}, m_i)$ for $i = 1, \ldots, q$ and sends pk and the σ_i to $\mathcal{A}$.
- $\mathcal{A}$ outputs a message m^* and a signature σ^*.

We define forgery and strong forgery as above, and define $\text{Sig}_\text{w}\text{-Adv}[\mathcal{A}, \mathcal{S}]$ and $\text{Sig}_\text{s,w}\text{-Adv}[\mathcal{A}, \mathcal{S}]$ to be the probability that $\mathcal{A}$ outputs a forgery and a strong forgery, respectively. If these quantities are negligible in λ, we say $\mathcal{A}$ is *(strongly) unforgeable against a weak adversary*.

2.4 Signatures and Assumptions in Bilinear Groups

Let BGen be an algorithm that takes input 1^λ and outputs a tuple $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$ with the following properties:

- p is a prime in $[2^\lambda, 2^{\lambda+1}]$;
- $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of order p in which group operations are efficently computable;
- $\hat{e} \colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable, nondegenerate, bilinear map (or "pairing").

We refer to the tuple $\mathcal{G}$ as a *bilinear group*. We assume that the descriptions of $\mathbb{G}_1$ and $\mathbb{G}_2$ include explicit generators. We define the following computational problems in bilinear groups:

- **Computational co-Diffie-Hellman (co-CDH):** an instance of the co-CDH problem is a tuple $(g_1, g_1^\alpha, h_1, g_2, g_2^\alpha)$ for randomly chosen $g_1 \xleftarrow{\text{R}} \mathbb{G}_1$, $g_2 \xleftarrow{\text{R}} \mathbb{G}_2$, and $\alpha \xleftarrow{\text{R}} \mathbb{Z}_p$. A solution is the element $h_1^\alpha \in \mathbb{G}_1$.
- **q-Strong Diffie-Hellman (q-SDH):** an instance of the q-SDH problem for $q \geq 1$ is a tuple $(g_1, g_1^\alpha, g_1^{\alpha^2}, \ldots, g_1^{\alpha^q}, g_2, g_2^\alpha)$ for randomly chosen $g_1 \xleftarrow{\text{R}} \mathbb{G}_1$, $g_2 \xleftarrow{\text{R}} \mathbb{G}_2$, and $\alpha \xleftarrow{\text{R}} \mathbb{Z}_p$. A solution is a pair $(r, g_1^{1/(\alpha+r)}) \in \mathbb{Z}_p \times \mathbb{G}_1$. (We use the convention $1/0 = 0$ so the pair $(-\alpha, 1_{\mathbb{G}_1})$ is a valid solution.)

If $\mathcal{A}$ is an algorithm that takes an instance of the co-CDH problem, we define $\text{co-CDH-Adv}[\mathcal{A}, \mathcal{G}]$ to be the probability that $\mathcal{A}$ outputs a solution. We say the co-CDH assumption holds for BGen if for all polynomial-time algorithms $\mathcal{A}$, $\text{co-CDH-Adv}[\mathcal{A}, \mathcal{G}]$ is a negligible function of λ. We define $q\text{-SDH-Adv}[\mathcal{A}, \mathcal{G}]$ and the q-SDH assumption analogously.

If there is an efficiently computable isomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$, then our definition is equivalent to that of [BLS04]; when $\mathbb{G}_1 = \mathbb{G}_2$ we recover the usual CDH problem. We note that an algorithm that solves co-CDH can solve q-SDH for any $q \geq 1$.

2.4.1 Signatures from co-CDH

The following construction is due originally to Gentry and Silverberg [GS02], who proved security in the random oracle model under the co-CDH assumption; Boneh and Boyen [BB04] and Waters [Wat05] later gave variants that are secure in the standard model under the same assumption.

Let $\mathcal{G}$ be a bilinear group and let $H\colon \{0,1\}^\lambda \to \mathbb{G}_1$ be a hash function.[1] We define a signature scheme $\mathsf{GS}(H)$ as follows:

$\mathsf{GS}(H).\mathsf{Setup}(1^\lambda)$: Run $\mathsf{BGen}(1^\lambda)$ and let $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$ be the output. Choose random $g_1 \xleftarrow{\text{R}} \mathbb{G}_1$, $g_2 \xleftarrow{\text{R}} \mathbb{G}_2$, and $\alpha \xleftarrow{\text{R}} \mathbb{Z}_p$. The public key is $\mathsf{pk} = (\mathcal{G}, g_2, \hat{e}(g_1, g_2)^\alpha, H)$, and the secret key is $\mathsf{sk} = g_1^\alpha$.

$\mathsf{GS}(H).\mathsf{Sign}(\mathsf{sk}, m)$: Choose random $r \xleftarrow{\text{R}} \mathbb{Z}_p$, and output $\sigma \leftarrow (g_2^r, g_1^\alpha \cdot H(m)^r)$.

$\mathsf{GS}(H).\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: Write $\sigma = (\sigma_1, \sigma_2) \in \mathbb{G}_2 \times \mathbb{G}_1$. Output 1 if $\hat{e}(\sigma_2, g_2) = \hat{e}(H(m), \sigma_1) \cdot \hat{e}(g_1, g_2)^\alpha$; otherwise output 0.

Theorem 2.4 ([GS02]). *If the co-CDH assumption holds for* BGen *and H is modeled as a random oracle, then* $\mathsf{GS}(H)$ *is unforgeable.*

Boneh and Boyen [BB04] showed that by using the hash function $H_{\mathsf{BB}}(x) = u^x v$ for random (public) $u, v \in \mathbb{G}_1$, one can prove "selective" security under co-CDH in the standard model. Waters [Wat05] constructed a hash function H_{W} which, when used in the GS scheme, allows a proof of full security under co-CDH in the standard model. The hash function H_{W} takes messages $m \in \{0,1\}^\lambda$ and is defined as follows:

$H_{\mathsf{W}}.\mathsf{Setup}(\mathbb{G})$: Choose random $u_0, u_1, \ldots, u_\lambda \xleftarrow{\text{R}} \mathbb{G}_1$, and output $\mathsf{hk} = \{u_i\}$.

$H_{\mathsf{W}}.\mathsf{Eval}(\mathsf{hk}, m)$: Let $m[i]$ be the ith bit of m. Output $u_0 \cdot \prod_{i=1}^{\lambda} u_i^{m[i]}$.

Theorem 2.5 ([Wat05]). *If the co-CDH assumption holds for* BGen*, then* $\mathsf{GS}(H_{\mathsf{W}})$ *is unforgeable.*

2.4.2 Strongly Unforgeable co-CDH signatures

Boneh, Shen, and Waters [BSW06] showed how to modify the Waters construction to obtain the strong unforgeability property. We give a variant that is strongly unforgeable against a weak adversary, which is all we will need to construct homomorphic signatures. We call this variant $\mathsf{GS}'(H)$:

$\mathsf{GS}'(H).\mathsf{Setup}(1^\lambda)$: Run $\mathsf{GS}(H).\mathsf{Setup}(1^\lambda)$. Output pk, sk, and a collision-resistant hash function $F\colon \{0,1\}^* \to \{0,1\}^\lambda$.

$\mathsf{GS}'(H).\mathsf{Sign}(\mathsf{sk}, m)$: Choose random $r \xleftarrow{\text{R}} \mathbb{Z}_p$, and output $\sigma \leftarrow (g_2^r, g_1^\alpha \cdot H\big(F(m \| g_2^r)\big)^r)$.

$\mathsf{GS}'(H).\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: Write $\sigma = (\sigma_1, \sigma_2) \in \mathbb{G}_2 \times \mathbb{G}_1$. Output $\mathsf{GS}(H).\mathsf{Verify}(\mathsf{pk}, F(m \| \sigma_1), \sigma)$.

Theorem 2.6 ([BSW06]). *If* $\mathsf{GS}(H)$ *is unforgeable, then* $\mathsf{GS}'(H)$ *is strongly unforgeable against a weak adversary.*

We note that this transformation was also applied by Attrapadung and Libert [AL11] to the signatures derived from the Lewko-Waters IBE [LW10] in order to prevent signatures from being rerandomized.

[1] H is actually a family of hash functions parametrized by $\mathbb{G}_1$; for readability we suppress this dependency in the notation.

2.4.3 Boneh-Boyen Signatures

Boneh and Boyen [BB08] proposed a different pairing-based signature in the standard model. As compared with the Waters signatures $\mathsf{GS}(H_{\mathsf{W}})$, Boneh-Boyen signatures are shorter and require a smaller public key, but are secure under an (apparently) stronger assumption, the q-SDH assumption. We give the version that is secure against a weak adversary, and we denote the scheme BB.

$\mathsf{BB.Setup}(1^\lambda)$: Run $\mathsf{BGen}(1^\lambda)$ and let $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e})$ be the output. Choose random $\alpha \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_p$ and generators $g_1 \overset{\mathrm{R}}{\leftarrow} \mathbb{G}_1$, $g_2 \overset{\mathrm{R}}{\leftarrow} \mathbb{G}_2$. The public key is $\mathsf{pk} = (\mathcal{G}, g_1, g_2, g_2^\alpha)$, and the secret key is $\mathsf{sk} = \alpha$.

$\mathsf{BB.Sign}(\mathsf{sk}, m)$: Given a message $m \in \mathbb{Z}_p$, output $\sigma = g_1^{1/(\alpha+m)}$. (We use the convention $1/0 = 0$ so $\mathsf{BB.Sign}(\mathsf{sk}, -\alpha) = 1_{\mathbb{G}_1}$.)

$\mathsf{BB.Verify}(\mathsf{pk}, m, \sigma)$: Output 1 if $\hat{e}(\sigma, g_2^m \cdot g_2^\alpha) = \hat{e}(g_1, g_2)$; otherwise output 0.

Theorem 2.7 ([BB08])**.** *If the q-SDH assumption holds for $\mathcal{G}$, then* BB *is strongly unforgeable against a weak adversary making at most q signature queries.*

2.5 Signatures and Assumptions in RSA Groups

Let RSAGen be an algorithm that takes input 1^λ and outputs two primes $p, q \in [2^\lambda, 2^{\lambda+1}]$ with the property that $(p-1)/2$ and $(q-1)/2$ are both prime. We let $N = pq$ and we refer to the group $\mathbb{Z}_N^*$ as an *RSA group*. We define the following computational problems in RSA groups:

- **(Random-Exponent) RSA Problem:** an instance of the RSA problem is a tuple (N, e, g) for N output by RSAGen, a randomly chosen integer e less than and relatively prime to $\varphi(N) = (p-1)(q-1)$, and a random $g \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_N^*$. A solution is the element $g^{1/e} \in \mathbb{Z}_N^*$.
- **Strong RSA Problem:** an instance of the strong RSA problem is a tuple (N, g) for N output by RSAGen and a random $g \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_N^*$. A solution is a pair $(z, e) \in \mathbb{Z}_N^* \times \mathbb{Z}$ with $e > 1$ such that $z = g^{1/e}$.

If $\mathcal{A}$ is an algorithm that takes an instance of the RSA problem, we define $\mathrm{RSA\text{-}Adv}[\mathcal{A}, N]$ to be the probability that $\mathcal{A}$ outputs a solution. We say the RSA assumption holds for RSAGen if for all polynomial-time algorithms $\mathcal{A}$, $\mathrm{RSA\text{-}Adv}[\mathcal{A}, N]$ is a negligible function of λ. We define $\mathrm{SRSA\text{-}Adv}[\mathcal{A}, N]$ and the strong RSA assumption analogously.

2.5.1 Gennaro-Halevi-Rabin Signatures

Gennaro, Halevi, and Rabin [GHR99] gave the first "hash-and-sign" signature proved secure without random oracles, using the strong RSA assumption. We give the version that is secure against a weak adversary. We also note that the same construction is secure in the random oracle model under the (random-exponent) RSA assumption.

Let $H\colon \{0,1\}^\lambda \to \mathbb{Z}$ be a hash function. We define a signature scheme $\mathsf{GHR}(H)$ as follows:

$\mathsf{GHR}(H)\mathsf{.Setup}(1^\lambda)$: Run $\mathsf{RSAGen}(1^\lambda)$, let $(p, q, N = pq)$ be the output, and let $\mathbb{G}$ be the subgroup of squares in $\mathbb{Z}_N^*$. Choose a random $g \overset{\mathrm{R}}{\leftarrow} \mathbb{G}$. The public key is $\mathsf{pk} = (N, g)$ and the secret key is $\mathsf{sk} = (p, q)$.

$\mathsf{GHR}(H)\mathsf{.Sign}(\mathsf{sk}, m)$: Output $\sigma = g^{1/H(m)}$.

$\mathsf{GHR}(H)\mathsf{.Verify}(\mathsf{pk}, m, \sigma)$: If $\sigma^{H(m)} = g$, output 1; otherwise output 0.

Theorem 2.8 ([GHR99]). *If the strong RSA assumption holds for* RSAGen *and* H *is a collision-resistant hash function that outputs odd primes less than* $2^{2\lambda-2}$*, then* $\mathsf{GHR}(H)$ *is strongly unforgeable against a weak adversary.*

Theorem 2.9. *If the RSA assumption holds for* RSAGen *and* H *is modeled as a random oracle that outputs odd primes less than* $2^{2\lambda-2}$*, then* $\mathsf{GHR}(H)$ *is strongly unforgeable against a weak adversary.*

Gennaro, Halevi, and Rabin [GHR99, §6] show how to construct a hash function in the standard model that is collision-resistant and outputs odd primes.

2.5.2 Hohenberger-Waters Signatures

Hohenberger and Waters [HW09b] define a hash function H_{HW} under which $\mathsf{GHR}(H)$ can be proven secure under the (random-exponent) RSA assumption in the standard model. Instead of outputting primes as in the GHR scheme, the HW hash function outputs a product of λ primes, one for each message bit. The hash function takes messages $m \in \{0,1\}^\lambda$ and is defined as follows:

$H_{\mathsf{HW}}.\mathsf{Setup}(1^\lambda)$: Choose a pseudorandom function $F\colon \{0,1\}^* \times \mathbb{Z} \to \{0,1\}^{2\lambda-2}$, a random key K for F, and a random value $c \in \{0,1\}^\lambda$. Output $\mathsf{hk} = (F, K, c)$.

$H_{\mathsf{HW}}.\mathsf{Eval}(\mathsf{hk}, m)$: Define the function $G : \{0,1\}^* \to \{0,1\}^{2\lambda-2}$ by $G(z) = F_K(z, i(z)) \oplus c$, where $i(z)$, called the *resolving index for* z, is the smallest positive integer such that $F_K(z, i(z)) \oplus c$ is an odd prime. Let $m^{(j)}$ denote the first j bits of m, and let $e_j = G(m^{(j)})$. Output $\prod_{i=1}^{\lambda} e_j$.

Theorem 2.10 ([HW09b]). *If the RSA assumption holds for* RSAGen*, then* $\mathsf{GHR}(H_{\mathsf{HW}})$ *is strongly unforgeable against a weak adversary.*

3 Building Blocks

3.1 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

$$\mathsf{Sign}_{\mathsf{sk}}\colon \mathcal{M} \times \mathcal{R} \to \mathbb{G} \times \{0,1\}^*, \tag{3.1}$$

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

$$\sigma_1 = g^{f_{\mathsf{sk}}(m,r)} \tag{3.2}$$

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.)

Examples. We now verify that all of the signature schemes discussed in Section 2.3 are pre-homomorphic according to Definition 3.1. In each case we explicitly identify the groups $\mathbb{G}$ and $\mathbb{G}'$, the function f_{sk}, and the Test algorithm.

- $\mathsf{GS}(H)$: we have $\mathbb{G} = \mathbb{G}_2$ with generator $g = g_2$, and $\mathbb{G}' = \mathbb{G}_1$. The signing function is $f_{\mathsf{sk}}(m, r) = r$. We define $\mathsf{GS}(H).\mathsf{Test}(\mathsf{pk}, m, (\sigma_1, \sigma_2), x, y)$ to output 1 if and only if $\hat{e}(x, \sigma_1) = \hat{e}(y, g_2)$. The bilinearity and nondegeneracy of $\hat{e}$ ensures that the latter is true if and only if $\log_x(y) = \log_{g_2}(\sigma_1)$ (regardless of the output of $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$).

- BB: we have $\mathbb{G} = \mathbb{G}_1$ with generator $g = g_1$, and $\mathbb{G}' = \mathbb{G}_2$. The (deterministic) signing function is $f_{\mathsf{sk}}(m) = 1/(\alpha + m) \pmod p$. We define $\mathsf{BB}.\mathsf{Test}(\mathsf{pk}, m, \sigma, x, y)$ to output 1 if and only if $\hat{e}(\sigma, x) = \hat{e}(g_1, y)$.

- $\mathsf{GHR}(H)$: we have $\mathbb{G} = \mathbb{G}' =$ squares in $\mathbb{Z}_N^*$, with generator g. The (deterministic) signing function is $f_{\mathsf{sk}}(m) = 1/H(m) \pmod{\varphi(N)/4}$. We define $\mathsf{GHR}(H).\mathsf{Test}(\mathsf{pk}, m, \sigma, x, y)$ to output 1 if and only if $y^{H(m)} = x$. If $\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1$ then $\sigma^{H(m)} = g$ and therefore $\log_g(\sigma) = \log_x(y)$.

In fact, as a referee pointed out to us, *any* signature scheme is pre-homomorphic: take $\mathbb{G} = \{1\}$, $f_{\mathsf{sk}} = 0$, and let σ_2 be the signature. The usefulness of the pre-homomorphic property only becomes apparent when it is combined with the "samplable" and "extractable" properties discussed in Section 5.2.

3.2 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 [KFM04, CJL09, BFKW09, AL11] as well as the RSA-based construction of Gennaro et al. [GKKR10].

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 $h_1, \ldots, h_n \overset{\text{R}}{\leftarrow} \mathbb{G}$ 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 $H_{\mathsf{hom}}(\mathbf{v}) = \mathsf{HomHash.Eval}(\mathsf{hk}, \mathbf{v})$.

We can shorten the description of H_{hom} by using a hash function $O \colon \mathbb{Z} \to \mathbb{G}$ to produce the values $h_1, \ldots, h_n$. If O is modeled as a random oracle, then in our security proofs we can program O so that we know the discrete logs of the h_i to some base g.

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.)

3.3 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 $x \overset{\text{R}}{\leftarrow} \chi$, the distribution of $g^{\phi(x)}$ is statistically close[2]to the uniform distribution on $\mathbb{G}$; and
2. If the order of $\mathbb{G}$ is not efficiently computable, then for $x \overset{\text{R}}{\leftarrow} \chi$, 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 $a = N \cdot 2^m$.

3.4 Chameleon Hashing

As defined by Krawczyk and Rabin [KR00], 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: given m, s, m', and the trapdoor, one can compute s' such that $C(m, s) = C(m', s')$. Furthermore, the distribution of s' conditioned on the values of $(m', C(m', s'))$ is the same regardless of whether we choose s' at random (the "forward" direction) or compute it using the trapdoor (the "backward" direction). Chameleon hashes can be used to turn any weakly unforgeable signature scheme into an unforgeable one; see [HW09b, Appendix A] for a proof.

[2] We say two distributions parametrized by an integer n are *statistically close* if their statistical distance is a negligible function of n (i.e., smaller than $1/\text{poly}(n)$)

To show unforgeability of our homomorphic signature scheme (as opposed to weak unforgeability) we will embed a "homomorphic" chameleon hash function C.[3]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 5.2) 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 $\mathbb{G} \subset \mathbb{Z}_N^*$.)

[3] One might ask why we do not give a generic conversion from weakly secure homomorphic signatures. The reason is that we are turning the homomorphic hash function that is already in the weakly secure signature into a chameleon hash, rather than adding a new chameleon hash function on top of the homomorphic hash.

2. If the order of $\mathbb{G}$ is efficiently computable from $\mathsf{pk}_{\mathcal{S}}$, 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 $t_1, \ldots, t_k, u \xleftarrow{\text{R}} \mathbb{G}'$.
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 $\kappa \xleftarrow{\text{R}} \mathcal{K}$.[4]
6. Choose a pseudorandom function $F : \mathcal{K}' \times \{0,1\}^\lambda \times \mathbb{Z} \to R$, and choose a random key $\mu \xleftarrow{\text{R}} \mathcal{K}'$.
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 $\mathcal{M} = \{\mathbf{v} \in R^n : \|\mathbf{v}\| \leq B_1\}$, 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.)

- 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 $s \xleftarrow{\text{R}} \mathbb{Z}_{|\mathbb{G}|}$.
4. Compute $\sigma_3 \leftarrow \big(t_i \cdot H_{\mathsf{hom}}(\mathbf{v}) \cdot u^s\big)^{f_{\mathsf{sk}}(\tau,r)}$, 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 $\sigma_3' \leftarrow \prod_{i=1}^k \big(\sigma_3^{(i)}\big)^{c_i}, \quad s' \leftarrow \sum_{i=1}^k c_i s^{(i)}$.

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

2. Output $\sigma' = (\sigma_1^{(1)}, \sigma_2^{(1)}, \sigma_3', s')$.

Lemma 4.1. *The homomorphic signature scheme* $\mathsf{HomSig}(\mathcal{S})$ *satisfies the correctness properties of Definition 2.1.*

Proof. Let $\tau \in \{0,1\}^\lambda$ be a tag, $\mathbf{v} \in R^n$ be a vector with $\|\mathbf{v}\| \leq B_1$, and $i \in \{1, \ldots, k\}$ be an index. Suppose $\sigma = (\sigma_1, \sigma_2, \sigma_3, s) \leftarrow \mathsf{Sign}(\mathsf{sk}, \tau, m, i)$. Note that the function π_i corresponds to the unit vector $\mathbf{e}_i$ with a 1 in the ith place and 0 elsewhere. First, since $(\sigma_1, \sigma_2) \leftarrow \mathcal{S}.\mathsf{Sign}(\tau, r)$, the correctness of $\mathcal{S}$ implies that $\zeta_1 = 1$. Next, we compute

$$\zeta_2 \leftarrow \mathsf{Test}\big(\mathsf{pk}_{\mathcal{S}},\ \tau,\ (\sigma_1, \sigma_2),\ x,\ \sigma_3\big),$$

where $x = t_i \cdot H_{\mathsf{hom}}(\mathbf{v}) \cdot u^s$. By construction, we have $\sigma_3 = x^{f_{\mathsf{sk}}(\tau, r)}$. The form of the Sign algorithm (Definition 3.1 (2)) then implies that $\log_g(\sigma_1) = f_{\mathsf{sk}}(\tau, r)$. Since $\mathcal{S}.\mathsf{Verify}(\mathsf{pk}_{\mathcal{S}}, \tau, (\sigma_1, \sigma_2)) = 1$, the correctness property of Test implies that $\zeta_2 = 1$. The condition $\|\mathbf{v}\| \leq B$ implies $\zeta_3 = 1$. It follows that $\mathsf{Verify}(\mathsf{pk}, \tau, \mathbf{v}, \sigma, \pi_i) = 1$.

Now let $\tau \in \{0,1\}^\lambda$ be a tag, $\mathbf{v}_1, \ldots, \mathbf{v}_k \in R^n$ be a tuple of vectors with $\|\mathbf{v}_i\| \leq B_1$, and $f_1, \ldots, f_k, g$ be linear functions represented as vectors in R^k, with $g = (c_1, \ldots, c_k)$. Suppose $\vec{\sigma} = (\sigma^{(1)}, \ldots, \sigma^{(k)})$ is a vector of signatures with $\sigma^{(i)} = (\sigma_1^{(i)}, \sigma_2^{(i)}, \sigma_3^{(i)}, s^{(i)})$ for $i = 1, \ldots, k$, produced by applying Eval iteratively zero or more times to the output of $\mathsf{Sign}(\mathsf{sk}, \tau, \mathbf{v}_i, i)$. Suppose $\mathsf{Verify}(\mathsf{pk}, \tau, \mathbf{w}_i, \sigma_i, f_i) = 1$ for $i = 1, \ldots, k$ and some $\mathbf{w}_1, \ldots, \mathbf{w}_k \in R^n$. Let $\sigma' = (\sigma_1', \sigma_2', \sigma_3', s') \leftarrow \mathsf{HomSig}(\mathcal{S}).\mathsf{Eval}(\mathsf{pk}, \tau, f, \vec{\sigma})$, let $\mathbf{y} = g(\vec{\mathbf{w}}) = \sum_{i=1}^k c_i \mathbf{w}_i$, and let $f' = g \circ \vec{f} = \sum_{i=1}^k c_i f_i$. We must show that

$$\mathsf{Verify}\left(\mathsf{pk}, \tau, \mathbf{y}, \sigma', f'\right) = 1. \tag{4.1}$$

First, since the $\sigma_1' = \sigma_1^{(1)}$ and $\sigma_2' = \sigma_2^{(1)}$, the fact that $\sigma^{(1)}$ verifies implies that $\zeta_1 = 1$. Next, represent the functions f_i as $(d_{i1}, \ldots, d_{ik})$, and arrange the d_{ij} in a matrix $D \in R^{k \times k}$. Then the function $f' = g \circ \vec{f}$ is represented by $(c_1, \ldots, c_k) \cdot D \in R^k$, and the x computed in Step (2) of Verify is given by

$$x = \left(\prod_{j=1}^{k} t_j^{\sum_i c_i d_{ij}} \right) \cdot H_{\mathsf{hom}}(\mathbf{y}) \cdot u^{s'},$$

Using the homomorphic property of H_{hom} and the fact that $s' = \sum c_i s^{(i)}$, we can rewrite this as

$$x = \prod_{i=1}^{k} \left(\left(\prod_{j=1}^{k} t_j^{d_{ij}} \right) \cdot H_{\mathsf{hom}}(\mathbf{v}_i) \cdot u^{s^{(i)}} \right)^{c_i} = \prod_{i=1}^{k} x_i^{c_i}, \tag{4.2}$$

where x_i is the value computed to verify $\sigma^{(i)}$ for the function f_i.

Now the fact that all of the $\sigma^{(i)}$ are produced using the Sign and Eval algorithms with tag τ implies that $\sigma_1^{(i)} = \sigma_1^{(j)}$ for all i, j, and therefore $\sigma_1^{(i)} = g^e$ for some integer e independent of i. Since all of the $\sigma^{(i)}$ verify, it must hold that $\sigma_3^{(i)} = x_i^e$ for all i. Since $\sigma_3' = \prod_{i=1}^k (\sigma_3^{(i)})^{c_i}$ by construction, it follows from (4.2) that $\sigma_3' = x^e$, and therefore $\zeta_2 = 1$.

Finally, since $\|\mathbf{y}\| \leq k \cdot \|f\| \cdot \max_i(\{\|\mathbf{v}_i\|\}) \leq kB_1B_2$, we have $\zeta_3 = 1$. We conclude that (4.1) holds. $\square$

5 Security

Recall that an adversary can break a homomorphic signature scheme by computing any of three types of forgeries in Definition 2.2: a Type 1 forgery consists of a valid signature on a message-function pair belonging to a file the adversary *has not* queried to the challenger, while a Type 2 forgery consists of a valid signature on a message-function pair belonging to a file the adversary *has* queried to the challenger, but for which the message is not equal to the function applied to the queried messages. A Type 3 forgery consists of a valid signature on a message-function pair belonging to a file the adversary has queried to the challenger, but for which the adversary has not queried enough messages for the function to be well-defined on the input data set.

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.

5.1 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.*

Specifically, let $\mathcal{A}$ be an adversary that attacks the homomorphic signature scheme and outputs a Type 1 forgery, a Type 2a forgery, or $\perp$. Then there is an adversary $\mathcal{B}$ that attacks $\mathcal{S}$, an adversary $\mathcal{B}'$ that attacks the PRF Ψ, and a negligible value of ϵ such that

$$\text{HomSig-Adv}[\mathcal{A}, \mathsf{HomSig}(\mathcal{S})] \leq \text{Sig}_{\text{s,w}}\text{-Adv}[\mathcal{B}, \mathcal{S}] + \text{PRF-Adv}[\mathcal{B}', \Psi] + \epsilon.$$

Proof. Our algorithm $\mathcal{B}$ will play the role of the (weak) adversary in the unforgeability game for $\mathcal{S}$ and that of the challenger in the unforgeability game for $\mathsf{HomSig}(\mathcal{S})$. The algorithm works as follows:

Setup:

1. Choose random tags $\tau_1, \ldots, \tau_q \overset{\text{R}}{\leftarrow} \{0,1\}^\lambda$. Send the set of queries $T = \{\tau_1, \ldots, \tau_q\}$ to the challenger for $\mathcal{S}$, and receive $\mathsf{pk}_\mathcal{S}$ and signatures $(\sigma_1^{(\ell)}, \sigma_2^{(\ell)}) \leftarrow \mathcal{S}.\mathsf{Sign}(\mathsf{sk}_\mathcal{S}, \tau_\ell)$ for each ℓ in $1, \ldots, q$.
2. Let $\mathbb{G}, \mathbb{G}', R, B_1, B_2$ be as in $\mathsf{HomSig}(\mathcal{S}).\mathsf{Setup}$, let g be the generator for $\mathbb{G}$ defined by $\mathsf{pk}_\mathcal{S}$, and let χ be a $\mathbb{G}'$-uniform distribution on R.

3. For $i = 1, \ldots, k$, choose $\gamma_i \xleftarrow{\text{R}} \chi$ and set $t_i = g^{\gamma_i}$.
4. For $j = 1, \ldots, n$, choose $\alpha_j \xleftarrow{\text{R}} \chi$ and set $h_j = g^{\alpha_j}$. Let $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n)$ and $\mathsf{hk} = (h_1, \ldots, h_n)$.
5. Choose $\delta \xleftarrow{\text{R}} \chi$ and set $u = g^{\delta}$.
6. 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 vector $\mathbf{v} \in R^n$, do 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. Let (σ_1, σ_2) be the signature on τ_ℓ obtained from the $\mathcal{S}$ challenger.
3. Choose $s \xleftarrow{\text{R}} \Xi_{\tau_\ell, \mathbf{v}}$.
4. Compute $\sigma_3 = \sigma_1^{\gamma_i + \langle \vec{\alpha}, \mathbf{v} \rangle + \delta s}$ and send the signature $\sigma^{(\ell,i)} = (\sigma_1, \sigma_2, \sigma_3, s)$ to the adversary.
5. Set $c_\ell \leftarrow c_\ell + 1$.

Forgery: When $\mathcal{A}$ outputs a forgery $(\tau^*, \mathbf{w}^*, f^*, \sigma^*)$ with $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, s^*)$, output τ^* and (σ_1^*, σ_2^*).

We analyze the simulation using a series of games. Let W_i be the event that $\mathcal{A}$ wins the unforgeability game in Game i.

Game 0. This is the real unforgeability game, where $\mathcal{A}$ interacts with a challenger for $\mathsf{HomSig}(\mathcal{S})$, and therefore $\Pr[W_0] = \text{HomSig-Adv}[\mathcal{A}, \mathsf{HomSig}(\mathcal{S})]$.

Game 1. This is the same as Game 0, except we choose tags τ_ℓ and compute signatures $(\sigma_1^\ell, \sigma_2^\ell)$ at the beginning of the game, instead of at the time signatures are requested. We implement the arrays F and A as described above, and when the adversary makes a query on the ℓth file, we look up and return the appropriate values. This change is purely conceptual, so $\Pr[W_1] = \Pr[W_0]$.

Game 2. This is the same as Game 1, except we choose t_i, h_j, u as described above instead of as random elements of $\mathbb{G}'$. Since χ is a $\mathbb{G}$-uniform distribution, the values of t_i, h_j, and u are statistically close to uniform in $\mathbb{G}'$. It follows that $\Pr[W_2] \geq \Pr[W_1] - \epsilon$ for some negligible ϵ

Game 3. To compute a signature on vector $\mathbf{v}$ with tag τ_ℓ and counter i, we compute the σ_3 component as $\sigma_3 = \sigma_1^{\gamma_i + \langle \vec{\alpha}, \mathbf{v} \rangle + \delta s}$. It follows that

$$\sigma_3 = \left(g^{f_{\mathsf{sk}}(\tau_\ell, r_\ell)}\right)^{\gamma_i + \langle \vec{\alpha}, \mathbf{v} \rangle + \delta s} = \left(t_i \cdot H_{\mathsf{hom}}(\mathbf{v}) \cdot u^s\right)^{f_{\mathsf{sk}}(\tau_\ell, r_\ell)}$$

for $r_\ell = \Psi_\kappa(\tau_\ell)$, and therefore this change is only conceptual. (Note that this value of r_ℓ is also used to compute σ_1 and σ_2. Thus $\Pr[W_3] = \Pr[W_2]$.

Game 4. We replace the PRF Ψ_κ (used to produce the randomness r for $\mathcal{S}.\mathsf{Sign}$) with a truly random function. (In particular, if two of the τ_ℓ values are equal then we output the same r for both.) Since this replacement is the only difference between Games 3 and 4, there is a PRF adversary $\mathcal{B}'$ that satisfies

$$\text{PRF-Adv}[\Psi, \mathcal{B}'] = |\Pr[W_4] - \Pr[W_3]| \geq \text{HomSig-Adv}[\mathcal{A}, \mathsf{HomSig}(\mathcal{S})] - \epsilon - \Pr[W_4]. \tag{5.1}$$

Observe that in Game 4, the adversary $\mathcal{A}$ is interacting with our simulator $\mathcal{B}$. If the event W_4 occurs, then $\mathsf{HomSig}(\mathcal{S}).\mathsf{Verify}(\mathsf{pk}, \tau^*, m^*, \sigma^*, f) = 1$. In particular, this means that $\mathcal{S}.\mathsf{Verify}(\mathsf{pk}_{\mathcal{S}}, \tau^*, (\sigma_1^*, \sigma_2^*)) = 1$. If $\mathcal{A}$ outputs a Type 1 forgery, then $\tau^* \neq \tau_\ell$ for all ℓ, and thus $\mathcal{B}$'s output wins the unforgeability game for $\mathcal{S}$. If $\mathcal{A}$ outputs a Type 2a forgery, then $\tau^* = \tau_\ell$ for some ℓ, but (σ_1^*, σ_2^*) is not the signature on τ_ℓ received from the $\mathcal{S}$ challenger; thus $\mathcal{B}$'s output also wins the unforgeability game in this case.

We conclude that $\mathrm{Sig}_{\mathrm{s,w}}\text{-}\mathrm{Adv}[\mathcal{B}, \mathcal{S}] = \Pr[W_4]$, and the theorem follows from (5.1). □

Note that since the signatures are deterministic (due to the use of the PRF Ψ), a collision in tags τ_ℓ does not help the adversary produce a Type 1 or Type 2a forgery. In addition, it was pointed out to us by a referee that the proof of Theorem 5.1 holds for arbitrary signature schemes $\mathcal{S}$, with the trivial pre-homomorphic structure that sets $f_{\mathsf{sk}} = 0$ and puts the signature in the σ_2 component. Thus Type 1 and 2a forgeries do not exploit the homomorphic properties of the system.

5.2 Type 2b Forgeries

We now consider 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 $1 - \gamma$.

We say that $\mathcal{S}$ is *computationally* δ-simulatable and γ-extractable if properties (2) and (3) hold only in a computational sense; i.e., if no efficient adversary can distinguish the public key and signatures produced by $\mathcal{S}$ from the real public key and signatures.

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.*

Specifically, let $\mathcal{A}$ be an adversary that attacks the homomorphic signature scheme while making at most $q = \operatorname{poly}(\lambda)$ queries and outputs a Type 2b forgery or $\perp$. Then there is an adversary $\mathcal{B}$ that solves $\mathcal{P}$, an adversary $\mathcal{B}'$ that attacks the PRF Ψ, an adversary B'' that attacks the PRF F, and a negligible value of ϵ, such that

$$\text{HomSig-Adv}[\mathcal{A}, \mathsf{HomSig}(\mathcal{S})] \leq \frac{q}{\gamma\delta} \cdot \mathcal{P}\text{-Adv}[\mathcal{B}, \mathbb{G}] + \text{PRF-Adv}[\mathcal{B}', \Psi] + \text{PRF-Adv}[\mathcal{B}'', F] + \frac{q^2}{2^\lambda} + \epsilon.$$

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 $\mathsf{HomSig}(\mathcal{S})$. The algorithm works 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 $\alpha_j, \beta_j \xleftarrow{\text{R}} \chi$ 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 $\gamma_i, \delta_i \xleftarrow{\text{R}} \chi$ 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 $\eta \xleftarrow{\text{R}} \chi$ and set $u = x^\eta y$.
6. Choose random tags $\tau_1, \ldots, \tau_\ell \xleftarrow{\text{R}} \{0,1\}^\lambda$, 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 $s^{(\ell,i)} \xleftarrow{\text{R}} \Xi_{\ell,\mathbf{v}}$ (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

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

   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.

We analyze the simulation using a series of games. Let W_i be the event that $\mathcal{A}$ wins the unforgeability game in Game i.

Game 0. This is the real unforgeability game, where $\mathcal{A}$ interacts with a challenger for $\mathsf{HomSig}(\mathcal{S})$, and thus $\Pr[W_0] = \text{HomSig-Adv}[\mathcal{A}, \mathsf{HomSig}]$.

Game 1. Instead of choosing t_i, h_j, and u uniformly at random from $\mathbb{G}'$, we compute them as in Steps (3)–(5) of the public key generation prodecure above. Since χ is a $\mathbb{G}'$-uniform distribution on R, the values of t_i and h_j, and u are statistically close to uniform in $\mathbb{G}'$. It follows that

$$\Pr[W_1] \geq \Pr[W_0] - \epsilon_1 \tag{5.2}$$

for some negligible ϵ_1.

Game 2. We replace the PRF Ψ_κ (used to produce the randomness r_ℓ for $\mathcal{S}.\mathsf{Sign}$) with a truly random function, taking care to use the same value of r_ℓ for each vector signed with tag τ_ℓ. Since this replacement is the only difference between Game 1 and Game 2, there is a PRF adversary $\mathcal{B}'$ that satisfies

$$\Pr[W_2] \geq \Pr[W_1] - \text{PRF-Adv}[\Psi, \mathcal{B}']. \tag{5.3}$$

Game 3. We abort and raise an event τ-coll if two of the tags τ_ℓ, τ'_ℓ chosen by the challenger are equal. By a birthday bound, we have

$$\Pr[W_3] \geq \Pr[W_2 \wedge \neg\tau\text{-}\mathsf{coll}] \geq \Pr[W_2] - \frac{q^2}{2^\lambda} \tag{5.4}$$

Game 4. We now use Sim to produce the public key $\mathsf{pk}_{\mathcal{S}}$ and to compute $\mathcal{S}$ signatures on the τ_ℓ. If Sim aborts, we abort and raise an event Sim-abort. By Properties (1)–(3) of Definition 5.2, we have

$$\Pr[W_4] = \Pr[W_3 \wedge \neg\mathsf{Sim}\text{-}\mathsf{abort}] \geq \delta \cdot \Pr[W_3] - \epsilon_4 \tag{5.5}$$

for some negligible ϵ_4.

Game 5. We now abort and raise an event wrong-ℓ if $\tau^* \neq \tau_{\ell^*}$, where ℓ^* is the value guessed by Sim. Since the adversary's view is independent of ℓ^*, we have

$$\Pr[W_5] \geq \Pr[W_4 \wedge \neg\mathsf{wrong}\text{-}\ell] = \frac{1}{q} \cdot \Pr[W_4]. \tag{5.6}$$

Game 6. This is identical to Game 5, except instead of choosing $s^{(\ell^*,i)}$ from $\Xi_{\ell^*,\mathbf{v}}$ as in the real game, we choose $s^{(\ell^*,i)}$ from the distribution $\Xi'_{\ell^*,\mathbf{v}}$ obtained by replacing the PRF F in Definition 3.5 with a truly random function. It follows that there is a PRF adversary $\mathcal{B}''$ that satisfies

$$\Pr[W_6] \geq \Pr[W_5] - \text{PRF-Adv}[F, \mathcal{B}'']. \tag{5.7}$$

Game 7. Finally, instead of choosing $s^{(\ell^*,i)}$ from $\Xi'_{\ell^*,i}$ as in Game 6, we set $s^{(\ell^*,i)} = \delta_i + \langle \vec{\beta}, \mathbf{v}_i \rangle$, where $\mathbf{v}_i$ is the ith vector from the ℓ^*th file. Observe that from the adversary's point of view the variables α_j, γ_i are defined only mod $|\mathbb{G}|$. Since for each choice of the variables β_j, δ_i there is a unique setting of $\alpha_j, \gamma_i \bmod |\mathbb{G}|$ consistent with the adversary's view, these two ways of choosing $s^{(\ell^*,i)}$ are equivalent, and thus

$$\Pr[W_7] = \Pr[W_6] \tag{5.8}$$

Observe that in this game $\mathcal{A}$ is interacting with our simulator $\mathcal{B}$.

Let W_8 be the event that $\mathcal{B}$ outputs a solution to the instance I of problem $\mathcal{P}$, so $\Pr[W_8] = \mathcal{P}\text{-Adv}[\mathcal{B}, \mathbb{G}]$. We wish to calculate the probability of W_8 conditioned on W_7 occuring; i.e., conditioned on $\mathcal{A}$ winning the unforgeability game when interacting with $\mathcal{B}$. To simplify notation, in the following analysis we let $\tau = \tau^* = \tau_{\ell^*}$ be the tag in question and $\omega = \omega_{\ell^*}$.

Define $X = \left(\prod_{i=1}^k t_i^{c_i}\right) \cdot H_{\mathsf{hom}}(\mathbf{w}^*) \cdot u^{s^*}$. Since W_7 occurs we have $\mathsf{HomSig}(\mathcal{S}).\mathsf{Verify}(\mathsf{pk}, \tau, \mathbf{w}^*, \sigma^*, f) = 1$, and therefore both $\mathcal{S}.\mathsf{Verify}(\mathsf{pk}_{\mathcal{S}}, \tau, (\sigma_1^*, \sigma_2^*)) = 1$ and $\mathsf{Test}(\mathsf{pk}_{\mathcal{S}}, \tau, (\sigma_1^*, \sigma_2^*), X, \sigma_3^*) = 1$. Property (3) of Definition 3.1 now implies that $\sigma_3^* = X^\omega$.

By the homomorphic property of $\mathsf{HomHash}$ (3.2), we have

$$\begin{aligned} z &= \left(\frac{\left(\prod_{i=1}^k t_i^{c_i}\right) \cdot H_{\mathsf{hom}}(\mathbf{w}^*) \cdot u^{s^*}}{x^a} \right)^\omega \\ &= \left(\frac{x^{\langle \vec{\gamma}, \mathbf{c} \rangle + \langle \vec{\alpha}, \mathbf{w}^* \rangle + \eta s^*} y^{-\langle \vec{\delta}, \mathbf{c} \rangle - \langle \vec{\beta}, \mathbf{w}^* \rangle + s^*}}{x^{\langle \vec{\gamma}, \mathbf{c} \rangle + \langle \vec{\alpha}, \mathbf{w}^* \rangle + \eta s^*}} \right)^\omega \\ &= y^{b \cdot \omega}. \end{aligned}$$

Now 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. If this is the case, then

$$\Pr[W_8] \geq \gamma \cdot \Pr[W_7]. \tag{5.9}$$

Compiling the results from equations (5.2)–(5.9) proves the theorem.

It remains only to show that b is $\mathbb{G}'$-uniform. Recall that $s^{(\ell^*,i)} = \delta_i + \langle \vec{\beta}, \mathbf{v}_i \rangle$, where $\mathbf{v}_i$ is the ith vector queried with tag τ, and let $\hat{s} = \sum_{i=1}^k c_i s^{(\ell^*,i)} = \langle \vec{\delta}, \mathbf{c} \rangle + \langle \vec{\beta}, \sum c_i \mathbf{v}_i \rangle$, and observe that the value of $\hat{s}$ is known to the adversary. Let $\mathbf{y} = \sum c_i \mathbf{v}_i - \mathbf{w}^* \in R^n$; then $b = \langle \vec{\beta}, \mathbf{y} \rangle + s^* - \hat{s}$. 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.

If both (1) and (2) hold, then the value $b = \langle \vec{\beta}, \mathbf{y} \rangle + s^* - \hat{s}$ is statistically close to uniform mod $|\mathbb{G}'|$. Furthermore, if $|\mathbb{G}|$ is not efficiently computable, then the same argument shows that b is statistically close to uniform mod e for all $e \in (|\mathbb{G}|/16, |\mathbb{G}|]$.

To prove condition (1), we observe that since $\mathcal{A}$ outputs a Type 2 forgery, we have $\mathbf{y} \neq 0$ in R^n. If $R = \mathbb{Z}_{|\mathbb{G}|}$ then clearly $\mathbf{y}$ is nonzero mod $|\mathbb{G}|$, while if $R = \mathbb{Z}$ then $\|\mathbf{w}^*\|$ and $\|\sum c_i \mathbf{v}_i\|$ are both less than kB_1B_2. In the latter case, our choice of B_1 and B_2 implies $\|\mathbf{y}\| \leq |\mathbb{G}|/16$, and therefore $\mathbf{y}$ is also nonzero mod $|\mathbb{G}|$.

To prove condition (2), we first note that since the variables α_j, γ_i appear only in the exponent, from the adversary's viewpoint they are only defined mod $|\mathbb{G}|$. The uniformity property of χ means that we can replace α_j, γ_i with uniform samples from $\mathbb{Z}_{|\mathbb{G}|}$ with only negligible effect on the adversary's view. Now let $\vec{\beta}^*$ be any fixed value of $\vec{\beta}$. The value of $\vec{\beta}^*$ and the values of $s^{(\ell^*,i)}$ determine a unique value of $\vec{\delta}$, which comes from the distribution χ^k by construction. The values of h_j determine a unique value of $\vec{\alpha} \bmod |\mathbb{G}|$. Finally, the value of $\vec{\delta}$ determines a unique value of $\vec{\gamma} \bmod |\mathbb{G}|$.

We conclude that for each possible value of $\vec{\beta}$, there is a unique assignment to the other variables mod $|\mathbb{G}|$ that is consistent with the adversary's view. Thus the adversary obtains no information about $\vec{\beta}$, so it follows that $\vec{\beta}$ comes from a distribution statistically close to χ^n even when conditioned on the adversary's view. □

6 Instantiations

We now show that for each of the signature schemes described in Section 2.3, there exists a simulator that has the properties enumerated in Definition 5.2. Given this result, it follows from Theorems 5.1 and 5.3 and a straightforward hybrid argument that the homomorphic signature scheme HomSig instantiated with each signature scheme is secure.

We begin with the Gentry-Silverberg co-CDH signatures, in the strongly unforgeable version proposed by Boneh, Shen, and Waters. This example is unique in that the simulator $\mathsf{Sim}_{\mathsf{GS}}$ that we describe is *not* the simulator used in the security proof of the signature scheme. In the security proofs for GS-type signatures, the solution to the co-CDH challenge is implicitly used as the secret key in the simulation. In our simulation, on the other hand, the solution is used only in the conversion to a homomorphic scheme.

We now define the simulator $\mathsf{Sim}_{\mathsf{GS'}}(H)$, which takes an instance $(g_1, g_1^\alpha, h_1, g_2, g_2^\alpha)$ of the co-CDH problem and interacts with a weak signature adversary.

Setup: Choose a random $\beta \leftarrow \mathbb{Z}_p$ and let $\mathsf{sk} = g_2^\beta$. Define the hash function H so that $\log_{g_1}(H(F(m_i \| g_2^{r_i})) = d_i$ is known to the simulator for all m_i. Specifically, if H is a random oracle then we simply program $H(m)$ with g^{d_i} for random d_i; if H is the Waters hash function H_{W} (see page 10) then we choose $u_j = g_1^{a_j}$ for random a_j, and then $d_i = \sum a_j \cdot F(m_i \| g_2^{r_i})[j]$. The public key is $\mathsf{pk} = (\mathcal{G}, g_2, \hat{e}(g_1, g_2)^\beta, H)$.

Signatures: Given messages $m_1, \ldots, m_q$ queried by the adversary, pick a random $\ell^* \in 1, \ldots, q$. For $\ell \neq \ell^*$, compute the signature on message m_ℓ as usual. For $\ell = \ell^*$, compute $\sigma^{(\ell^*)} = \left(g_2^\alpha,\ g_1^\beta \cdot (g_1^\alpha)^{d_{\ell^*}}\right)$.

Proposition 6.1. *If H is either a random oracle or H_{W}, then $\mathsf{Sim}_{\mathsf{GS'}}(H)$ is 1-simulatable and $(1 - 1/p)$-extractable for the co-CDH problem.*

Proof. We check the conditions of Definition 5.2, using the simulator $\mathsf{Sim}_{\mathsf{GS}}$.

1. $\mathsf{Sim}_{\mathsf{GS}}$ never aborts.

2. The only change to the public key is in defining the hash function H; the simulated H is indistinguishable from the real H in both cases.

3. Clearly signatures produced by $\mathsf{Sim}_{\mathsf{GS}}$ are identical to real signatures for $\ell \neq \ell^*$. Since $(g_1^\alpha)^{d_{\ell^*}} = H(m_{\ell^*})^\alpha$, the signature $\sigma^{(\ell^*)}$ is identical to a real signature computed with randomness $r_{\ell^*} = \alpha$. Since α is random in $\mathbb{Z}_p$ and does not appear anywhere else in the adversary's view, this signature also is distributed identically to a real signature.

4. Let $x = g_1$ and $y = h_1$. For $\ell \neq \ell^*$ we have $\omega_\ell = r_\ell$, the randomness chosen by the simulator to compute the signature component $\sigma_1^{(\ell)} = g_2^{r_\ell}$, while we have $\omega_\ell = \alpha$. Since $\omega_\ell r_\ell$ is known to the simulator for all $\ell \neq \ell^*$, clearly the simulator can compute x^{ω_ℓ} and y^{ω_ℓ} for all $\ell \neq \ell^*$. Furthermore, since $\omega_{\ell^*} = \alpha$, the simulator also knows $x^{\omega_{\ell^*}} = g_1^{\alpha}$.

5. Given an integer b and the element $z = y^{b\cdot\omega_{\ell^*}} = h_1^{b\cdot\alpha}$, we define $\mathsf{Extract}$ to output $z^{1/b}$, or $\perp$ if $b = 0 \bmod p$. Thus for uniform b in $\mathbb{Z}_p$, $\mathsf{Extract}$ outputs a solution to the co-CDH problem with probability $1 - 1/p$.

□

We note that the challenge element h_1 is not used to construct simulated signatures — it is only used to simulate the σ_3 component of homomorphic signatures. Combining the results of Theorems 2.4, 2.5, 2.6, 5.1, and 5.3 gives the following:

Corollary 6.2. *If the co-CDH assumption holds for* BGen, *then* $\mathsf{HomSig}(\mathsf{GS}'(H))$ *is unforgeable if either* $H = H_{\mathsf{W}}$ *is the Waters hash function, or* H *is modeled as a random oracle.*

The system $\mathsf{HomSig}(\mathsf{GS}'(H))$ authenticates vectors defined over $\mathbb{Z}_p$. Signatures consist of two elements of $\mathbb{G}$, one element of $\mathbb{G}'$, and one element of $\mathbb{Z}_p$.

Boneh-Boyen signatures The simulator we describe is essentially the same one used in the proof of security against a weak adversary [BB08, Sec. 4.3]. Let $\mathsf{Sim}_{\mathsf{BB}}$ take an instance $(g_1, g_1^{\alpha}, g_1^{\alpha^2}, \ldots, g_1^{\alpha^q}, g_2, g_2^{\alpha})$ of the q-SDH problem and interact 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 $x = g_1^{P(\alpha)}$. We output the public key $\mathsf{pk} = (\mathcal{G}, x, g_2, g_2^{\alpha})$; the (implicit) secret key is α.

Signatures: Let $P_\ell(t) = \prod_{i\neq\ell}(t + m_i)$. The signature on m_ℓ is $\sigma^{(\ell)} = x^{1/(\alpha+m_\ell)} = g_1^{P_\ell(\alpha)}$, which can be computed from the q-SDH challenge.

Proposition 6.3. $\mathsf{Sim}_{\mathsf{BB}}$ *is* 1*-simulatable and* $(1 - 1/p)$*-extractable for the* q*-SDH problem.*

Proof. We check the conditions of Definition 5.2.

1. $\mathsf{Sim}_{\mathsf{BB}}$ never aborts.

2. Since g_1 is uniformly random in $\mathbb{G}_1$, the simulated public key is distributed identically to the real public key.

3. It is clear that signatures produced by $\mathsf{Sim}_{\mathsf{BB}}$ are identical to real signatures.

4. Let $P(t)$ and $P_\ell(t)$ be as above. Let $x = g_1^{P(\alpha)}$ and $y = g_1^{P_{\ell^*}(\alpha)}$. 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 $\ell \neq \ell^*$.

5. Using polynomial division, we can 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

$$z = y^{b\cdot\omega_{\ell^*}} = g_1^{b\cdot P^*(\alpha)/(\alpha+m_{\ell^*})} = g_1^{b\cdot(Q(\alpha)+c/(\alpha+m_{\ell^*}))},$$

we define Extract to output

$$\left(m_{\ell^*}, \left(\frac{z^{1/b}}{g_1^{Q(\alpha)}}\right)^{1/c}\right),$$

or $\perp$ if $b = 0 \bmod p$. Thus for uniform b in $\mathbb{Z}_p$, Extract outputs a solution to the q-SDH problem with probability $1 - 1/p$.

□

Combining the results of Theorems 2.7, 5.1, and 5.3 gives the following:

Corollary 6.4. *If the q-SDH assumption holds for* BGen, *then* HomSig(BB) *is unforgeable (in the standard model).*

The system HomSig(BB) authenticates vectors defined over $\mathbb{Z}_p$. Signatures consist of one element of $\mathbb{G}$, one element of $\mathbb{G}'$, and one element of $\mathbb{Z}_p$.

Gennaro-Halevi-Rabin Signatures. As with Boneh-Boyen signatures, the simulator we describe is essentially the same one used in the proof of security against a weak adversary (without random oracles). Let $\mathsf{Sim}_{\mathsf{GHR}}(H)$ take an instance (N, g) of the strong RSA problem and interact with a weak signature adversary. Assume H is a hash function that outputs odd primes in $[2^{2\lambda-4}, 2^{2\lambda-2}]$. (In our earlier description we had no lower bound on the output of H, but this condition is easy to add with minimal additional computational cost.)

Setup: Given distinct messages $m_1, \ldots, m_q \in \{0,1\}^\lambda$ queried by the adversary, form the integer $E = \prod_{i=1}^q H(m_i)$ and compute $x = g^{2E} \bmod N$. Abort if $H(m_i) = H(m_j)$ for some $i \neq j$ or if $H(m_i) \mid \varphi(N)$ for some i; otherwise output the public key $\mathsf{pk} = (N, x)$. The (implicit) secret key is the factorization of N.

Signatures: Let $E_\ell = \prod_{i \neq \ell} H(m_i)$. The signature on m_ℓ is $\sigma^{(\ell)} = x^{1/H(m_\ell)} = g^{2E_\ell}$.

Proposition 6.5. *Suppose the hash function H is ϵ_1-collision-resistant (i.e., the probability of a an efficient adversary finding a collision in H is at most ϵ_1) and the probability of an efficient adversary factoring N is at most ϵ_2. Then there is a negligible η such that* $\mathsf{Sim}_{\mathsf{GHR}}(H)$ *is $(1 - \epsilon_1 - \epsilon_2)$-simulatable and $(1 - \eta)$-extractable for the strong RSA problem.*

Proof. We check the conditions of Definition 5.2.

1. There are two abort conditions: $\mathsf{Sim}_{\mathsf{GHR}}(H)$ aborts if either $\mathcal{A}$ finds a collision in the hash function H or if $\mathcal{A}$ finds a value of m such that $H(m) \mid \varphi(N)$. The former event occurs with probability at most ϵ_1, while the latter occurs with probability at most ϵ_2. (If $\hat{p}$ is an odd factor of $\varphi(N)$, then $2\hat{p} + 1$ is a factor of N.)

2. If $\mathsf{Sim}_{\mathsf{GHR}}(H)$ doesn't abort, then g is uniformly random in $\mathbb{Z}_N^*$ and $H(m_i)$ is prime to $\varphi(N)$ for all i, so x is a uniformly random square in $\mathbb{Z}_N^*$.

3. It is clear that signatures produced by $\mathsf{Sim}_{\mathsf{GS}}$ are identical to real signatures.

4. Let E and E_ℓ be as above. Let $x = g^{2E}$ and $y = g^{2E_{\ell^*}}$. Since $\sigma^{(\ell)} = x^{1/H(m_\ell)}$, we have $\omega_\ell = 1/H(m_\ell)$. Thus the simulator can compute x^{ω_ℓ} for all ℓ and y^{ω_ℓ} for all $\ell \neq \ell^*$.

5. Let $e^* = H(m_{\ell^*})$. Given an integer b and the element

$$z = y^{b \cdot \omega_{\ell^*}} = g^{2b \cdot E_{\ell^*}/e^*},$$

we observe that $z^{e^*} = g^{2bE_{\ell^*}}$. Using Shamir's trick (see Lemma A.1), if $\gcd(e^*, 2bE_{\ell^*}) = 1$ then we can compute $g^{1/e^*} \in \mathbb{Z}_N^*$.

If Sim does not abort, then all of the factors of E_{ℓ^*} are distinct from e^*, so $\gcd(e^*, 2bE_{\ell^*}) = 1$ if and only if e^* divides b. Since $\phi(N)$ is not efficiently computable (otherwise we could factor N), the assumption that b is $\mathbb{G}$-uniform implies that $b \bmod e$ is uniform for all $e \in [\varphi(N)/2^6, \varphi(N)/4]$. Since $2^{2\lambda} < \varphi(N) < 2^{2\lambda+2}$ with overwhelming probability, e^* satisfies this condition and thus $b \bmod e^*$ is statistically close to uniform. Therefore the probability that e^* divides b is (statistically close to) $1/e^*$, which is negligible.

□

Essentially the same simulator works for the RSA problem when H is a random oracle; the main difference is that we abort if the challenge exponent e^* is not a prime in $[2^{2\lambda-4}, 2^{2\lambda-2}]$; it follows from explicit bounds on the prime number function $\pi(x)$ (see Theorem A.2) that this happens with probability less than $1/50\lambda$. Given the challenge e, we choose a random ℓ^* and program $H(m_{\ell^*}) = e^*$; we program the other values of $H(m_\ell)$ with random primes. The rest of the analysis proceeds as above; for a polynomial number of queries the probability of a collision in the hash function is negligible in λ. We obtain the following:

Proposition 6.6. *Suppose the hash function H is modeled as a random oracle. Then there is a negligible η such that $\mathsf{Sim}_{\mathsf{GHR}}(H)$ is $1/50\lambda$-simulatable and $(1-\eta)$-extractable for the RSA problem.*

For the Hohenberger-Waters version of GHR signatures we modify $\mathsf{Sim}_{\mathsf{GHR}}$ as follows: we pick a random c' and find some i, j such that i is the smallest positive integer such $F_K(m_{\ell^*}^{(j)}, i) \oplus c'$ is an odd prime e_0. We abort if either e^* or e_0 is not in $[2^{2\lambda-4}, 2^{2\lambda-2}]$; otherwise we define the public value $c = c' \oplus e_0 \oplus e^*$. The value E in the simulator is now the product of all primes produced in the computation of the $H_{\mathsf{HW}}(m_i)$, and E_ℓ is the product of all such primes except for those produced in the computation of $H_{\mathsf{HW}}(m_\ell)$. We abort if there is a collision among these primes, which still happens with negligible probability. We obtain the following:

Proposition 6.7. *There is a negligible η such that $\mathsf{Sim}_{\mathsf{GHR}}(H_{\mathsf{HW}})$ is $1/(50\lambda)^2$-simulatable and $(1-\eta)$-extractable for the RSA problem.*

Combining the results of Theorems 2.8, 2.9, 2.10, 5.1, and 5.3 gives the following:

Corollary 6.8. *If the strong RSA assumption holds for RSAGen and H is a collision-resistant hash function that outputs primes in $[2^{2\lambda-4}, 2^{2\lambda-2}]$, then $\mathsf{HomSig}(\mathsf{GHR}(H))$ is unforgeable (in the standard model).*

If the RSA assumption holds for RSAGen, then $\mathsf{HomSig}(\mathsf{GHR}(H))$ is unforgeable if either $H = H_{\mathsf{HW}}$ is the Hohenberger-Waters hash function, or H is modeled as a random oracle that outputs primes in $[2^{2\lambda-4}, 2^{2\lambda-2}]$.

The system $\mathsf{HomSig}(\mathsf{GHR}(H))$ authenticates vectors over $\mathbb{Z}$ of bounded length. Signatures consist of two elements of $\mathbb{Z}_N$ and one integer of size at most $N^2 n \cdot 2^{80}$. These are about twice as large as the strong-RSA signatures of Catalano, Fiore, and Warinschi [CFW11a], which consist of one element of $\mathbb{Z}_N$ and one integer of size at most $N \cdot 2^{80}$. However, our signatures are unforgeable against an adaptive per-message attack, while the security proof of the scheme in [CFW11a] requires the adversary to submit all vectors for a file at once.

7 Further Directions

We have described a general framework for converting signature schemes with certain properties to linearly homomorphic signatures. Security of the converted scheme is based on the same assumption as the underlying scheme. The security model we use allows a stronger adversary than in prior constructions. We gave instantiations based on the schemes of Waters, Boneh-Boyen, Gennaro-Halevi-Rabin, and Hohenberger-Waters.

One direction for further work is to improve the efficiency of our schemes, especially on the RSA side. The fact that we require a "homomorphic chameleon hash in the exponent" dictates that we append a very large integer to vectors over $\mathbb{Z}$ signed with our RSA systems. We would like to remove this requirement, ideally proving security without a chameleon hash. Catalano, Fiore, and Warinschi [CFW11b] have recently made significant progress in this direction, proposing more efficient linearly homomorphic signatures based on q-SDH and strong RSA. It is an open problem to extend these results to weaker assumptions (co-CDH and RSA) and to the security model with our stronger adversary.

Another direction is to strengthen the adversary in our model even further. At present we require that tags τ used to identify files be chosen at random by the challenger. Allowing the adversary to choose tags — or even to use simple counters or filenames — would improve the efficiency of the system in practice. While randomly chosen tags seem to be a requirement for proofs in the random oracle model (the hash of the tag must be computed at signing time, so the tag must be unpredictable to the adversary), their presence in this work serves only to allow us to use an underlying signature that is secure against a weak adversary. Using a weakly secure underlying signature is more efficient and allows for simpler analysis of our construction, but it would be worthwhile to investigate the tradeoff between tag choice and efficiency.

Finally, our framework does not incorporate any homomorphic signature schemes based on lattice assumptions [BF11a, BF11b], and indeed at present there exist no such systems secure in the standard model. An adaptation of our work to lattice-based signatures would not only give us new linearly homomorphic systems but may also give systems with a larger class of admissible functions, such as the polynomial scheme of [BF11b]. In addition, perhaps the insights gained by adapting our framework to lattices would allow us to move closer to a *fully homomorphic* signature scheme — one that could authenticate *any* function on computed data.

References

[ACLY00] R. Ahlswede, N. Cai, S. Li, and R. Yeung. "Network information flow." *IEEE Transactions on Information Theory* **46**:4 (2000), 1204–1216.

[ABC+12] J. H. Ahn, D. Boneh, J. Camenisch, S. Hohenberger, a. shelat, and B. Waters. "Computing on authenticated data." In *Theory of Cryptography — TCC 2012*, Springer LNCS **7194** (2012), 1–20.

[AL11] N. Attrapadung and B. Libert. "Homomorphic network coding signatures in the standard model." In *Public Key Cryptography — PKC '11*, Springer LNCS **6571** (2011), 17–34.

[BB04] D. Boneh and X. Boyen. "Efficient selective-ID secure identity-based encryption without random oracles." In *Advances in Cryptology — EUROCRYPT '04*, Springer LNCS **3027** (2004), 223–238.

[BB08] D. Boneh and X. Boyen. "Short signatures without random oracles and the SDH assumption in bilinear groups." *J. Cryptology* **21** (2008), 149–177. Extended abstract in *Advances in Cryptology — EUROCRYPT '04*.

[BFKW09] D. Boneh, D. Freeman, J. Katz, and B. Waters. “Signing a linear subspace: Signature schemes for network coding.” In *Public-Key Cryptography — PKC ’09*, LNCS **5443**. Springer (2009), 68–87.

[BF11a] D. Boneh and D. M. Freeman. “Homomorphic signatures over binary fields and new tools for lattice-based signatures.” In *Public Key Cryptography — PKC 2011*, Springer LNCS **6571** (2011), 1–16. Full version available at http://eprint.iacr.org/2010/453.

[BF11b] D. Boneh and D. M. Freeman. “Homomorphic signatures for polynomial functions.” In *Advances in Cryptology — EUROCRYPT 2011*, ed. K. Paterson, Springer LNCS **6632** (2011), 149–168. Full version available at http://eprint.iacr.org/2011/018.

[BLS04] D. Boneh, B. Lynn, and H. Shacham. “Short signatures from the Weil pairing.” *Journal of Cryptology* **17** (2004), 297–319. Extended abstract in *ASIACRYPT 2001*, Springer LNCS **2248** (2001), 514–532.

[BSW06] D. Boneh, E. Shen, and B. Waters. “Strongly unforgeable signatures based on computational Diffie-Hellman.” In *Public key cryptography—PKC 2006*, Springer LNCS **3958**. Springer, Berlin (2006), 229–240.

[BFF+09] C. Brzuska, M. Fischlin, T. Freudenreich, A. Lehmann, M. Page, J. Schelbert, D. Schröder, and F. Volk. “Security of sanitizable signatures revisited.” In *Public Key Cryptography — PKC ’09*, Springer LNCS **5443** (2009), 317–336.

[CFW11a] D. Catalano, D. Fiore, and B. Warinschi. “Adaptive pseudo-free groups and applications.” In *Advances in Cryptology — EUROCRYPT 2011*, ed. K. G. Paterson, Springer LNCS **6632** (2011), 207–223.

[CFW11b] D. Catalano, D. Fiore, and B. Warinschi. “Efficient network coding signatures in the standard model.” Cryptology ePrint Archive, Report 2011/696 (2011). Available at http://eprint.iacr.org/2011/696. To appear in *PKC 2012*.

[CJL09] D. Charles, K. Jain, and K. Lauter. “Signatures for network coding.” *International Journal of Information and Coding Theory* **1**:1 (2009), 3–14.

[Fis03] M. Fischlin. “The Cramer-Shoup strong-RSA signature scheme revisited.” In *Public Key Cryptography — PKC 2003*, Springer LNCS **2567** (2003), 116–129.

[GHR99] R. Gennaro, S. Halevi, and T. Rabin. “Secure hash-and-sign signatures without the random oracle.” In *Advances in Cryptology — Eurocrypt 1999*, LNCS **1592** (1999), 123–139.

[GKKR10] R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin. “Secure network coding over the integers.” In *Public Key Cryptography — PKC ’10*, Springer LNCS **6056** (2010), 142–160.

[GS02] C. Gentry and A. Silverberg. “Hierarchical ID-based cryptography.” In *Advances in Cryptology — ASIACRYPT 2002*, Springer LNCS **2501**. Springer, Berlin (2002). 548–566.

[HK08] D. Hofheinz and E. Kiltz. “Programmable hash functions and their applications.” In *Advances in Cryptology — CRYPTO 2008*, Springer LNCS **5157** (2008), 21–38.

[HW09a] S. Hohenberger and B. Waters. “Realizing hash-and-sign signatures under standard assumptions.” In *Advances in Cryptology — EUROCRYPT ’09*, Springer LNCS **5479** (2009), 333–350.

[HW09b] S. Hohenberger and B. Waters. “Short and stateless signatures from the RSA assumption.” In *Advances in Cryptology — CRYPTO ’09*, Springer LNCS **5677** (2009), 654–670.

[JMSW02] R. Johnson, D. Molnar, D. Song, and D. Wagner. “Homomorphic signature schemes.” In *Topics in Cryptology — CT-RSA 2002*, Springer LNCS **2271** (2002), 244–262.

[KR00] H. Krawczyk and T. Rabin. “Chameleon signatures.” In *Network and Distributed System Security Symposium* (2000).

[KFM04] M. Krohn, M. Freedman, and D. Mazières. “On-the-fly verification of rateless erasure codes for efficient content distribution.” In *Proc. of IEEE Symposium on Security and Privacy* (2004), 226–240.

[LW10] A. B. Lewko and B. Waters. “New techniques for dual system encryption and fully secure HIBE with short ciphertexts.” In *Theory of Cryptography — TCC 2010*, Springer LNCS **5978** (2010), 455–479.

[LYC03] S.-Y. R. Li, R. W. Yeung, and N. Cai. “Linear network coding.” *IEEE Trans. Info. Theory* **49**:2 (2003), 371–381.

[RS62] J. B. Rosser and L. Schoenfeld. “Approximate formulas for some functions of prime numbers.” *Illinois J. Math.* **6** (1962), 64–94.

[Sha83] A. Shamir. “On the generation of cryptographically strong pseudorandom sequences.” *ACM Transactions on Computer Systems* **1** (1983), 38–44.

[Wat05] B. Waters. “Efficient identity-based encryption without random oracles.” In *Advances in Cryptology — EUROCRYPT ’05*, Springer LNCS **3494** (2005), 320–329.

A Useful Facts From Number Theory

Shamir showed that it is easy to compute $y^{1/a} \bmod N$ given $y^{b/a}$:

Lemma A.1 ([Sha83])**.** *Given $x, y \in \mathbb{Z}_N$ together with $a, b \in \mathbb{Z}$ such that $x^a = y^b$ and $\gcd(a, b) = 1$, there is an efficient algorithm for computing $z \in \mathbb{Z}_N$ such that $z^a = y$.*

The algorithm is to use Euclid’s algorithm to compute $f, g \in \mathbb{Z}$ such that $af + bg = 1$, and then compute $z = x^g y^f$.

For our RSA proofs we need estimates on the density of prime numbers.

Theorem A.2 ([RS62])**.** *For any positive x, let $\pi(x)$ be the number of primes less than or equal to x. Then for $x \geq 17$,*

$$\frac{x}{\log x} < \pi(x) < \frac{1.26\,x}{\log x},$$

where $\log$ *is the natural logarithm.*

B Lewko-Waters Signatures in our Framework

In this section we show that the signature scheme derived from the Lewko-Waters identity-based encryption scheme [LW10] has the properties necessary for applying our generic framework. The resulting linearly homomorphic signature scheme is almost identical to that of Attrapadung and Libert [AL11]. However, our proof in Section 5 shows that the scheme is secure against an adversary that can adaptively query vectors; the proof of Attrapadung and Libert only holds against an adversary that queries entire files at once.

B.1 Composite-Order Bilinear Groups.

The Lewko-Waters construction uses bilinear groups whose order is a product N of three primes. Formally, let CGen be an algorithm that takes input 1^λ and outputs a tuple $\mathcal{G} = (N, G, G_T, \hat{e})$ with the following properties:

- $N = p_1 p_2 p_3$ is a product of three distinct primes $p_i \in [2^\lambda, 2^{\lambda+1}]$;
- G, G_T are groups of order N in which group operations are efficently computable;
- $\hat{e}\colon G \times G \to G_T$ is an efficiently computable, nondegenerate, bilinear map (or "pairing").

We refer to the tuple $\mathcal{G}$ as a *bilinear group of composite order*. We assume that the description of $\mathcal{G}$ includes explicit generators g_1, g_2, g_3 of the three prime-order subgroups. These generators satisfy the following "orthogonality property": for $i \neq j$, we have $\hat{e}(g_i, g_j) = 1$. For any t dividing N, we denote by G_t the cyclic subgroup of G of order t.

We define the following computational assumptions in bilinear groups of composite order:

- **Assumption 1:** An instance of the Assumption 1 problem is a tuple (g, X_3, T), where $g \xleftarrow{\text{R}} G_{p_1}$, $X_2 \xleftarrow{\text{R}} G_{p_2}$, and either $T = T_1 \xleftarrow{\text{R}} G_{p_1 p_2}$ or $T = T_2 \xleftarrow{\text{R}} G_{p_2}$. A solution is the integer $i \in \{1, 2\}$ such that $T = T_i$.
- **Assumption 2:** An instance of the Assumption 2 problem is a tuple $(g, X_1 X_2, X_3, Y_2 Y_3, T)$, where $g, X_1 \xleftarrow{\text{R}} G_{p_1}$, $X_2, Y_2 \xleftarrow{\text{R}} G_{p_2}$, $X_3, Y_3 \xleftarrow{\text{R}} G_{p_3}$, and either $T = T_1 \xleftarrow{\text{R}} G$ or $T = T_2 \xleftarrow{\text{R}} G_{p_1 p_3}$. A solution is the integer $i \in \{1, 2\}$ such that $T = T_i$.
- **Assumption 3:** An instance of the Assumption 3 problem is a tuple $(g, g^\alpha X_2, X_3, g^s Y_2, Z_2)$, where $g \xleftarrow{\text{R}} G_{p_1}$, $X_2, Y_2, Z_2 \xleftarrow{\text{R}} G_{p_2}$, $X_3 \xleftarrow{\text{R}} G_{p_3}$, and $\alpha, s \xleftarrow{\text{R}} \mathbb{Z}_N$. A solution is the element $e(g,g)^{\alpha s} \in G_T$.

If $\mathcal{A}$ is an algorithm that takes an instance of Assumption i, we define $\mathrm{Asmp}(i)\text{-}\mathrm{Adv}[\mathcal{A}, \mathcal{G}]$ to be the probability that $\mathcal{A}$ outputs a solution. We say that Assumption i holds for CGen if for all polynomial-time algorithms $\mathcal{A}$, $\mathrm{Asmp}(i)\text{-}\mathrm{Adv}[\mathcal{A}, \mathcal{G}]$ is a negligible function of λ.

Note that Lewko and Waters use the decisional version of Assumption 3 in their IBE scheme; for signatures we use the computational version (as do Attrapadung and Libert).

B.2 Lewko-Waters Signatures.

We now give the Lewko-Waters signature scheme and its security theorem. The signature scheme is similar in form to the Gentry-Silverberg scheme discussed in Section 2.4.

$\mathsf{LW.Setup}(1^\lambda)$: Run $\mathsf{CGen}(1^\lambda)$ and let $\mathcal{G} = (N, G, G_T, \hat{e})$ be the output. Choose random $u, g, h \xleftarrow{\text{R}} G_{p_1}$, $v \xleftarrow{\text{R}} g_{p_3}$ and $\alpha \xleftarrow{\text{R}} \mathbb{Z}_p$. The public key is $\mathsf{pk} = (\mathcal{G}, g, h, u, v, e(g,g)^\alpha)$, and the secret key is $\mathsf{sk} = \alpha$.

$\mathsf{LW.Sign}(\mathsf{sk}, m)$: Given a message $m \in \mathbb{Z}_N$, choose random $r, t, t' \xleftarrow{\text{R}} \mathbb{Z}_N$, and output $\sigma \leftarrow \big(g^r v^t, g^\alpha (u^m h)^r v^{t'}\big)$.

$\mathsf{LW.Verify}(\mathsf{pk}, m, \sigma)$: Write $\sigma = (\sigma_1, \sigma_2) \in G^2$. Output 1 if $\hat{e}(\sigma_2, g) = \hat{e}(u^m h, \sigma_1) \cdot \hat{e}(g,g)^\alpha$; otherwise output 0.

Theorem B.1 ([LW10]). *If Assumptions 1, 2, and 3 hold for* CGen*, then* LW *is unforgeable.*

In the original Lewko-Waters proposal the element $v \in G_{p_3}$ is part of the secret key. However, we observe that publishing v has no effect on the system's security. In particular, all of the simulators in the hybrid games of the security proof are given a random element of G_{p_3}, which the simulators can use as the element v.

The transformation of Boneh, Shen, and Waters [BSW06] allows us to convert the LW scheme into a scheme LW' that is strongly unforgeable against a weak adversary. Briefly, we pick a collision-resistant hash function F and replace the exponent r in the second component with $F(m \| g^r v^t)$. We omit the details.

B.3 Applying our Framework

The pre-homomorphic property. We first show that the Lewko-Waters signatures are pre-homomorphic according to Definition 3.1:

1. $\mathsf{LW.Sign}$ defines a function $\mathsf{Sign}_{\mathsf{sk}} : \mathbb{Z}_N \times \mathbb{Z}_N^3 \to G_{p_1 p_3}^2$.
2. Let r, t, t' be the randomness chosen by the Sign algorithm. Compute a, b such that $ap_1 + bp_3 = 1$. Let $f_{\mathsf{sk}} \colon \mathbb{Z}_N \times \mathbb{Z}_N^3 \to \mathbb{Z}$ be given by

$$f_{\mathsf{sk}}(m, (r, t, t')) = rbp_3 + tap_1. \qquad \text{(B.1)}$$

   Since g and v have order p_1 and p_3, respectively, we have

$$(gv)^{f_{\mathsf{sk}}(m,(r,t,t'))} = g^{rbp_3} v^{tap_1} = g^r v^t,$$

   which is the first component of the signature.
3. The Test function is evaluated by computing a pairing: given $(\mathsf{pk}, m, \sigma, x, y)$ with $x, y \in G_{p_1 p_3}$, output 1 if and only if $\hat{e}(x, \sigma_1) = \hat{e}(gv, y)$. The bilinearity and nondegeneracy of $\hat{e}$ ensure that the latter holds if and only if $\log_x(y) = \log_{gv}(\sigma_1)$.

Since the group involved in the signing algorithm is cyclic of (known) order N, the vectors signed using the derived linearly homomorphic scheme will be defined over the ring $R = \mathbb{Z}_N$. The distribution χ used for sampling in signatures is the uniform distribution on $\mathbb{Z}_N$.

In the simulations used in the security proof, signatures will be elements of the full group G rather than the subgroup $G_{p_1 p_3}$. In this case we want Test to act on inputs from G as if it was given inputs from the subgroup $G_{p_1 p_3}$.

Formally, we extend the requirements on Test as follows: let $\mathbb{G}$ be the group from Definition 3.1 and suppose for simplicity that $\mathbb{G}' = \mathbb{G}$. Let $\mathbb{H}$ be a group containing $\mathbb{G}$ and let $\pi \colon \mathbb{H} \to \mathbb{G}$ be a surjective homomorphism. Then we say that Test is *compatible with* $(\mathbb{H}, \pi)$ if

- Test satisfies Definition 3.1 (3) for inputs $\sigma_1, x, y \in \mathbb{G}$.
- When inputs σ_1 and y are in $\mathbb{H}$, $\mathsf{Test}(\mathsf{pk}, m, (\sigma_1, \sigma_2), x, y) = \mathsf{Test}(\mathsf{pk}, m, (\pi(\sigma_1), \sigma_2), x, \pi(y))$.

To apply this construction to the Lewko-Waters scheme we let $\mathbb{G} = G_{p_1 p_3}$ and $\mathbb{H} = G$. If $\pi \colon G \to G_{p_1 p_3}$ is the surjective homorphism that maps G_{p_2} to 1 and is the identity on $G_{p_1 p_3}$, then Test is compatible with (G, π). In particular, since g and x are in $G_{p_1 p_3}$, the pairing is oblivious to the G_{p_2} components of σ_1 and y.

The simulatable property. We now demonstrate a simulator $\mathsf{Sim}_{\mathsf{LW}}$ for the Lewko-Waters scheme (in its strongly unforgeable version) that satisfies (a generalization of) Definition 5.2. The simulator takes an instance $(g, g^\alpha X_2, X_3, g^s Y_2, Z_2)$ of the Assumption 3 problem and interacts with a weak signature adversary.

Setup: Choose random $a, b, \beta \xleftarrow{\text{R}} \mathbb{Z}_N$ and set $u = g^a$, $h = g^b$, $v = X_3$. Let F be a collision-resistant hash function. Output the public key $(\mathcal{G}, g, h, u, v, \hat{e}(g,g)^\beta, F)$. The secret key is β.

Signatures: Given messages $m_1, \dots, m_q$ queried by the adversary, pick a random $\ell^* \in 1, \dots, q$. For $\ell \neq \ell^*$, compute the signature on m_ℓ using the real Sign algorithm with randomness r_ℓ, t_ℓ, t'_ℓ. For $\ell = \ell^*$, choose random $t^*, t'^*, \mu \xleftarrow{\text{R}} \mathbb{Z}_N$ and compute

$$\sigma^{(\ell^*)} = \left(g^\alpha X_2 v^{t^*},\ g^\beta \cdot (g^\alpha X_2)^{aF(m_{\ell^*} \| g^\alpha X_2 v^{t^*}) + b} Z_2^\mu v^{t'^*}\right).$$

We now wish to show that $\mathsf{Sim}_{\mathsf{LW}}$ satisfies Definition 5.2. As above, we must modify the definition slightly to allow for inputs in a larger group $\mathbb{H}$. Specifically, we replace conditions (4) and (5) of Definition 5.2 with the following:

4'. Let $(\sigma_1^{(\ell)}, \sigma_2^{(\ell)})$ be the signature produced by Sim on the ℓth message query, and let $\omega_\ell = \log_g(\pi(\sigma_1)^{(\ell)})$. Then Sim can efficiently compute elements x and y in $\mathbb{H}$ such that

   - $\pi(x)$ and $\pi(y)$ generate $\mathbb{G}$.
   - For all ℓ, Sim can efficiently compute an element $X_\ell \in \mathbb{H}$ such that $\pi(X_\ell) = \pi(x^{\omega_\ell})$.
   - Let $\ell^* \in \{1, \dots, q\}$ be randomly chosen by Sim. For all $\ell \neq \ell^*$, Sim can efficiently compute an element $Y_\ell \in \mathbb{H}$ such that $\pi(Y_\ell) = \pi(y^{\omega_\ell})$.

5'. Condition (5) holds under the weaker requirement that $\pi(z) = \pi(y^{b \cdot \omega_{\ell^*}})$.

As before, in our application we have $\mathbb{G} = G_{p_1 p_3}$ and $\mathbb{H} = G$, and the map π projects away from the G_{p_2} component. The revised condition (4') simply says that condition (4) holds modulo G_{p_2}.

Proposition B.2. $\mathsf{LW}'(H)$ *is computationally* 1*-simulatable and* $(1 - \frac{3}{2^\lambda})$*-extractable for the Assumption 3 problem.*

Proof. We check the conditions of Definition 5.2 (with revised conditions (4') and (5')), using the simulator $\mathsf{Sim}_{\mathsf{LW}}$.

1. $\mathsf{Sim}_{\mathsf{LW}}$ never aborts.

2. The simulated public key is distributed identically to the real public key.

3. Clearly for $\ell \neq \ell^*$, the simulated signature on m_ℓ is distributed identically to a real signature. By [LW10, Lemmas 5–7], if Assumptions 1 and 2 hold then the signature on m_{ℓ^*} is computationally indistinguishable from a real signature.

4'. Let $x = gv$ and $y = g^s Y_2 v$. Then the simulator can compute $x^{\omega_\ell} = g^{r_\ell} v^{t_\ell}$ and $y^{\omega_\ell} = (g^s Y_2)^{r_\ell} v^{t_\ell}$ for all $\ell \neq \ell^*$ (recall the definition of f_{sk} in (B.1)). Furthermore, since we implicitly set $r_{\ell^*} = \alpha$ in the ℓ^*th signature, we have $x^{\omega_{\ell^*}} = g^\alpha v^{t^*}$. Thus if we set $X_{\ell^*} = g^\alpha X_2 v^{t^*}$, then $\pi(X_{\ell^*}) = \pi(g^\alpha v^{t^*})$.

5'. Suppose we are given an integer b and an element z with $\pi(z) = \pi(y^{b \cdot \omega_{\ell^*}}) = g^{bs} v^{bt^*}$. Then we have $z = g^{bs} R_2 v^{bt^*}$ for some $R_2 \in \mathbb{G}_{p_2}$. We define $\mathsf{Extract}$ to output $\hat{e}(z,g)^{1/b}$, or $\perp$ if b is not invertible mod N. Since $e(z,g) = e(g,g)^{bs}$, if b is uniform in $\mathbb{Z}_N$ then $\mathsf{Extract}$ outputs a solution to the Assumption 3 problem with probability at least $1 - \frac{3}{2^\lambda}$.

□

Combining the results of Theorems B.1, 5.1, and 5.3 gives the following:

Corollary B.3. *If Assumptions 1, 2, and 3 hold for* CGen*, then* $\mathsf{HomSig}(\mathsf{LW}')$ *is unforgeable.*

The system $\mathsf{HomSig}(\mathsf{LW}')$ authenticates vectors defined over $\mathbb{Z}_N$. Signatures consist of three elements of $\mathbb{G}$ and one element of $\mathbb{Z}_N$. The triple of elements is essentially the same as an Attrapadung-Libert signature (with σ_3 not rerandomized in G_{p_3}); the additional element of $\mathbb{Z}_N$ is a chameleon hash that allows us to prove security against our stronger adversary.

C Privacy

A linearly homomorphic signature scheme is *private* if given signatures on data $m_1, \dots, m_k \in \mathcal{M}$, a derived signature on $m' = f(m_1, \dots, m_k)$ produced by Eval does not leak any information about the underlying m_i beyond what is revealed by the derived message m' and the function f. While this privacy property ensures that the original data is hidden, we are not hiding the fact that derivation took place.

The formal definition of this property was proposed by Boneh and Freeman [BF11a], building on a definition of Brzuska et al. [BFF+09]. The motivating concept is that given signatures on messages derived from one of two different files, even an attacker who knows the secret key cannot determine which file contains the original data. This property is called *weak context hiding*; the "weak" refers to the fact that the original signatures on the file are not made public, and distinguishes this property from the stronger notion proposed by Ahn et al. [ABC+12].[5]

Definition C.1. A homomorphic signature scheme $\mathcal{S} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Eval})$ is *weakly context hiding* 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 $\mathsf{Setup}(1^n, k)$ to obtain $(\mathsf{pk}, \mathsf{sk})$ and gives pk and sk to $\mathcal{A}$. The public key defines a message space $\mathcal{M}$, a signature space Σ, and a set $\mathcal{F}$ of admissible functions $f \colon \mathcal{M}^k \to \mathcal{M}$.

Challenge: $\mathcal{A}$ outputs $(\vec{m}_0^*, \vec{m}_1^*, f_1, \dots, f_s)$ with $\vec{m}_0^*, \vec{m}_1^* \in \mathcal{M}^k$. The functions $f_1, \dots, f_s$ are in $\mathcal{F}$ and satisfy

$$f_i(\vec{m}_0^*) = f_i(\vec{m}_1^*) \qquad \text{for all } i = 1, \dots, s.$$

In response, the challenger generates a random bit $b \in \{0,1\}$ and a random tag $\tau \in \{0,1\}^n$. It signs the messages in $\vec{m}_b^*$ using the tag τ and obtains a vector $\vec{\sigma}$ of k signatures. Next, for $i = 1, \dots, s$ the challenger computes a signature $\sigma_i := \mathsf{Eval}(\mathsf{pk}, \tau, f_i, \vec{\sigma})$ on $f_i(\vec{m}_b^*)$. It sends the tag τ and the signatures $\sigma_1, \dots, \sigma_s$ to $\mathcal{A}$. Note that the functions $f_1, \dots, f_s$ can be output adaptively after $\vec{m}_0^*, \vec{m}_1^*$ are output.

Output: $\mathcal{A}$ outputs a bit b'.

The adversary $\mathcal{A}$ wins the game if $b = b'$. The *advantage* of $\mathcal{A}$ is the probability that $\mathcal{A}$ wins the game.

It follows from the definition that an adversary that wins the weak context hiding game can determine whether whether the challenge signatures were derived from signatures on $\vec{m}_0^*$ or from signatures on $\vec{m}_1^*$.

[5] The context-hiding properties proposed in [ABC+12] require that a derived signature be indistinguishable from a "fresh" signature on the same message (i.e., one produced by Sign). This notion is unrealizable in our context since the Verify algorithm takes as input the function f, which already reveals whether the signature is fresh or derived — a signature is produced by Sign if and only if its associated function is one of the projections π_i.

C.1 Achieving Privacy in Bilinear Group Schemes

Recall from Section 4 that signatures in our linearly homomorphic schemes are of the form $\sigma = (\sigma_1, \sigma_2, \sigma_3, s)$, where (σ_1, σ_2) is a signature on the tag τ using the underlying ordinary signature, $\sigma_3 = H_{\mathsf{hom}}(\mathbf{v}) \cdot u^s$ is a homomorphic hash of the vector $\mathbf{v}$ being signed, and s is randomness for a chameleon hash. The components σ_1, σ_2 are the same for all signatures associated with a given file and thus cannot help the adversary win the context-hiding game. Furthermore, we note that if we remove the chameleon hash (i.e., set $s = 0$), then the σ_3 component is unique for each vector $\mathbf{v}$, regardless of whether $\mathbf{v}$ was input to Sign or computed as a linear combination of other vectors. We conclude that the only information that can be useful to an adversary playing the context-hiding game is the value of s. However, when $|\mathbb{G}|$ is known, s is uniform in $\mathbb{Z}_{|\mathbb{G}|}$, so the value of s reveals nothing about the underlying messages.

We make these ideas precise in the following privacy theorem for our bilinear group systems.

Theorem C.2. *Suppose $|\mathbb{G}|$ is known. Then $\mathsf{HomSig}(\mathcal{S})$ is weakly context hiding.*

Proof. Let $\mathbf{v}_1, \ldots, \mathbf{v}_k \in R^n$ be one of the two sets of vectors sent by the adversary, and let $\mathbf{w} = f(\mathbf{v}_1, \ldots, \mathbf{v}_k) = \sum c_i \mathbf{v}_i$ for some $f = (c_1, \ldots, c_k) \in \mathcal{F}$. Suppose the challenger signs the vectors $\mathbf{v}_i$ using the tag τ and obtains a vector $\vec{\sigma}$ of k signatures. Next the challenger computes a signature $\sigma := \mathsf{Eval}(\mathsf{pk}, \tau, f, \vec{\sigma})$ on $\mathbf{w}$. As discussed above, this signature is of the form $(\sigma_1, \sigma_2, \sigma_3, s)$, where the σ_1 and σ_2 components are the same for all signatures associated with τ and $\sigma_3 = H_{\mathsf{hom}}(\mathbf{w}) \cdot u^s$. Since $|\mathbb{G}|$ is known, the distribution $\Xi_{\tau,\mathbf{v}}$ is the uniform distribution on $\mathbb{Z}_{|\mathbb{G}|}$. Thus in the adversary's view the value of s is uniformly distributed in $\mathbb{Z}_{|\mathbb{G}|}$ and independent of f and the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$. Thus the distribution of σ is identical for all sets of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ and functions f such that $f(\mathbf{v}_1, \ldots, \mathbf{v}_k) = \mathbf{w}$. □

Unfortunately, the above argument does not apply when $\mathbb{G}$ is a group of unknown order and $R = \mathbb{Z}$. In this case we choose the s component of signatures uniformly from some fixed interval of positive integers, so when we combine signatures the resulting value of s reveals information about the linear combination computed.

We make this observation precise by exhibiting an attack on the context-hiding property. Consider the system with $n = 1$ and $k = 2$, so data sets consist of pairs of integers. Suppose the attacker submits the two data sets $\vec{v}_0^* = (0, 0)$, $\vec{v}_1^* = (-1, 1)$ and the function $f = (1, 1)$; so $f(\vec{v}_0^*) = f(\vec{v}_1^*) = 0$. Suppose further that the distribution χ used in sampling s satisfies $\chi = -\chi$. When signing the first data set $(0, 0)$ the values of s_j come from the distribution χ, since $\beta_j \cdot 0 = 0$ for $j = 1, 2$. However, when signing the second data set $(-1, 1)$ the values of s_j come from $\chi \pm F_\mu(\tau, j)$ for $j = 1, 2$, which is computationally indistinguishable from $\chi \pm \chi$ under the assumption that F is a PRF. Furthermore, $\chi - \chi = \chi + \chi$ by our assumption on χ.

We conclude that for the file $\vec{v}_0^*$ the value of s in the derived signature on 0 is distributed as $\chi + \chi$, which has variance $2 \cdot \mathrm{Var}(\chi)$; while for the file $\vec{v}_1^*$ the value of s in the derived signature on 0 is distributed as $\chi + \chi + \chi + \chi$, which has variance $4 \cdot \mathrm{Var}(\chi)$. It follows that the attacker can distinguish the two cases with non-negligible probability.