Efficient Ring Signatures without Random Oracles

Hovav Shacham[*]
Weizmann Institute of Science
`hovav.shacham@weizmann.ac.il`

Brent Waters
SRI International
`bwaters@csl.sri.com`

August 24, 2006

Abstract

We describe the first efficient ring signature scheme secure, without random oracles, based on standard assumptions. Our ring signatures are based in bilinear groups. For l members of a ring our signatures consist of $2l + 2$ group elements and require $2l + 3$ pairings to verify. We prove our scheme secure in the strongest security model proposed by Bender, Katz, and Morselli: namely, we show our scheme to be anonymous against full key exposure and unforgeable with respect to insider corruption. A shortcoming of our approach is that all the users' keys must be defined in the same group.

[*]Supported by a Koshland Scholars Program fellowship.

1 Introduction

Ring signatures were introduced by Rivest, Shamir, and Tauman [18, 19]. Each user in the system generates and publishes a public key. (This key can be, for example, the description of an RSA permutation.) In generating a ring signature, a user can choose, arbitrarily, some other users to implicate in the signature. The public keys of these implicated users, along with the signer's public key, are said to form the ring for that signature. A verifier is convinced that someone in the ring is responsible for the signature, but cannot tell who.

In this paper we present the first efficient ring signature scheme secure, without random oracles, based on standard assumptions. Our scheme gives $O(l)$ signatures, with no a priori bound on ring size. Our ring signatures are based in bilinear groups. In particular, for l members of a ring our signatures consist of $2l + 2$ group elements and require $2l + 3$ pairings to verify. We now outline our approach.

In our ring signature scheme each user generates her own public-private keypair from the Waters [20] signature scheme defined over a group G of a composite order n, where the group is set up by a trusted authority. When a user wants to sign a message M on a ring R she first creates a ciphertext C, which is a BGN [9] encryption of her public signing key. Next, she proves that C is an encryption of exactly one of the public keys in the the ring R. We use proofs similar to that of Groth, Ostrovsky, and Sahai [16] to do this efficiently. Finally, she gives an encrypted signature of the message M using her private signing key and proves that the signature verifies under the encrypted key.

A shortcoming of our approach is that all the users' keys must be defined in the group G. This is unlike the generic construction of Bender, Katz, and Morselli [4], which does not place restrictions on the user keys, and also unlike some of the random-oracle–based schemes (discussed below) that allow for independently generated RSA user keys. In compensation, however, we obtain an efficient scheme provably secure, without random oracles, in the strongest security model proposed by Bender, Katz, and Morselli [4]: namely, we show our scheme to be anonymous against full key exposure and unforgeable with respect to insider corruption.

Related Work. The construction of Rivest, Shamir, and Tauman, requires that the users' public keys be for trapdoor-permutation–based schemes, i.e., include descriptions of a trapdoor permutation. Subsequently, ring signature constructions were presented where the underlying keys are discrete-log–based [17], discrete-log–based in the bilinear map setting [8], factoring-based [15], or a mix of trapdoor-permutation–type and discrete-log–type [1].

Ring signatures are also related to group signatures, introduced by Chaum and Van Heyst [13] and themselves the subject of much subsequent research. The two concepts differ in two main ways. First, the ring is determined by the signer and can be different for each signatures; in a group signature, group membership is controlled by a group manager and, at any given time, is fixed.[1] Second, no one can determine which member of a ring generated a signature; in a group signature, a tracing party possesses a special trapdoor that allows it to determine which group member is responsible for a signature.

[1]Dodis et al. [15, Section 6.3] describe ad-hoc group signatures, a primitive for which this difference is less pronounced.

Applications. The canonical application for ring signatures is secret leaking: A signature by the ring of all cabinet ministers on a leaked memo is credible, but doesn't incriminate any particular minister for the leak. Other applications have been proposed [4, 19].

Ring Signatures in the Standard Model. The security of the ring signatures proposed by Rivest, Shamir, and Tauman and in most subsequent papers holds in the random oracle model [2].[2] Some recent papers have considered how to construct ring signatures that are provably secure in the standard model.

Xu, Zhang, and Feng [21] describe a ring signature secure in the standard model, but the proof presented is not rigorous and is apparently flawed [4, n. 1]. Chow et al. [14] give a ring signature scheme with proof in the standard model, but based on a strong new assumption. Bender, Katz, and Morselli present a ring signature secure in the standard model assuming trapdoor permutations exist, but the scheme uses generic ZAPs for NP as a building block, and is thus impractical. In addition, they give two ring signature schemes secure in the standard model but which allow only two-signer rings: one based on the Waters signature [20], a second based on the Camenisch-Lysyanskaya signature [12].

Our ring signature scheme is related to a recent group signature secure without random oracles due to Boyen and Waters [10]. One important difference is that in their group signature paper the master public key, which belongs to the group manager, is in the clear and the first level message, which corresponds to the user's identity, is encrypted and then proved to be well formed. In our scheme, on the other hand, the message to be signed is public, but the verification key — which belongs to the user who generated the signature — is encrypted and then a proof is given that *it* is well formed. (In our case, "well-formed" means "in the ring.") The Boyen-Waters group signature is itself based on two lines of research: the identity-based encryption scheme in the standard model due to Waters [20], which follows up on earlier schemes by Boneh and Boyen [5, 6]; and the perfect non-interactive zero knowledge proofs of Groth, Ostrovsky, and Sahai [16], which are based on the homomorphic encryption scheme proposed by Boneh, Goh, and Nissim [9].

2 Mathematical Setting

Like Boyen and Waters, we make use of bilinear groups of composite order. These were introduced by Boneh, Goh, and Nissim [9]. Let n be a composite with factorization $n = pq$. We have:

- G is a multiplicative cyclic group of order n;
- G_p is its cyclic order-p subgroup, and G_q is its cyclic order-q subgroup;
- g is a generator of G, while h is a generator of G_q;
- G_T is a multiplicative group of order n;
- $e : G \times G \to G_T$ is an efficiently computable map with the following properties:
  - Bilinear: for all $u, v \in G$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$;
  - Non-degenerate: $\langle e(g, g)\rangle = G_T$ whenever $\langle g\rangle = G$;

[2]More precisely, Rivest, Shamir, and Tauman analyzed their construction in the ideal-cipher model; Bresson, Stern, and Szydlo [11] later showed that random oracles suffice for proving its security.

- $G_{T,p}$ and $G_{T,q}$ are the G_T-subgroups of order p and q, respectively;
- the group operations on G and G_T can be performed efficiently; and
- bitstrings corresponding to elements of G (and of G_T) can be recognized efficiently.

In our ring signature scheme, the description of such a group G, including the generators g and h, is given in the common reference string generated by the setup authority.

2.1 Complexity Assumptions

For our ring signature, we assume that two problems are difficult to solve in the setting described above: computational Diffie-Hellman in G_p and the Subgroup Decision Problem.

Computational Diffie-Hellman in G_p. Given the tuple (η, η^a, η^b), where $\eta \xleftarrow{\text{R}} G_p$ and $a, b \xleftarrow{\text{R}} \mathbb{Z}_p$, compute and output η^{ab}. In the composite setting one is additionally given the description of the larger group G, including the factorization (p, q) of its order n.

Subgroup Decision. Given w selected at random either from G (with probability 1/2) or from G_q (with probability 1/2), decide whether w is in G_q. For this problem one is given the description of G, but *not* given the factorization of n.

The assumptions are formalized by measuring an adversary's success probability for computational Diffie-Hellman and an adversary's guessing advantage for the subgroup decision problem. Note that if CDH in G_p as we have formulated it is hard then so is CDH in G. The assumption that the subgroup decision problem is hard is called the Subgroup Hiding (SGH) assumption, and was introduced by Boneh, Goh, and Nissim [9].

3 Underlying Signature

The underlying signature scheme is the Waters signature [20]. This signature was adapted for composite order groups by Boyen and Waters [10]. The variant we describe differs from theirs in retaining the original Waters formulation for the public key: $g_1, g_2 \in G$ rather than $e(g_1, g_2) \in G_T$.

Suppose that messages to be signed are encoded as elements of $\{0,1\}^k$ for some k. (For example, as the output of a k-bit collision-resistant hash function.) In addition to the system parameters of Section 2 above, the Waters scheme makes use of random generators $u', u_1, u_2, \ldots, u_k$ in G.

The scheme is as follows.

WC.Kg. Pick random $\alpha, \beta \xleftarrow{\text{R}} \mathbb{Z}_n$ and set $g_1 \leftarrow g^\alpha$ and $g_2 \leftarrow g^\beta$. The public key pk is $(g_1, g_2) \in G^2$. The private key sk is (α, β).

WC.Sig(sk, M)**.** Parse the user's private key sk as $(\alpha, \beta) \in \mathbb{Z}_n^*$ and the message M as a bitstring $(m_1, \ldots, m_k) \in \{0,1\}^k$. Pick a random $r \xleftarrow{\text{R}} \mathbb{Z}_n$ and compute

$$S_1 \leftarrow g^{\alpha\beta} \cdot \Big(u' \prod_{i=1}^{k} u_i^{m_i}\Big)^r \qquad \text{and} \qquad S_2 \leftarrow g^r \ .$$

The signature is $\sigma = (S_1, S_2) \in G^2$.

WC.Vf(pk, M, σ)**.** Parse the user's public key pk as $(g_1, g_2) \in G^2$, the message M as a bitstring $(m_1, \dots, m_k) \in \{0,1\}^k$, and the signature σ as $(S_1, S_2) \in G^2$. Verify that

$$e(S_1, g) \cdot e\Big(S_2^{-1}, u' \prod_{i=1}^{k} u_i^{m_i}\Big) \stackrel{?}{=} e(g_1, g_2) \tag{1}$$

holds; if so, output `valid`; if not, output `invalid`.

This signature is existentially unforgeable if computational Diffie-Hellman holds on G.

The Waters Signature in G_p. One can also restrict the Waters signature to the subgroup G_p, obtaining a signature scheme to which we refer as $\mathcal{WP}$. In this case, the generator g is replaced by a generator η of G_p, and exponents are drawn from $\mathbb{Z}_p$ rather than $\mathbb{Z}_n$. In particular, the verification equation is

$$e(\hat{S}_1, \eta) \cdot e\Big(\hat{S}_2^{-1}, \hat{u}' \prod_{i=1}^{k} \hat{u}_i^{m_i}\Big) \stackrel{?}{=} e(\eta_1, \eta_2) \ . \tag{2}$$

This variant is secure assuming CDH is hard in G_p, and is used in our reductions.

4 Ring Signature Definitions

Informally, a ring signature scheme should satisfy two security properties. First, it should be anonymous: an adversary should not be able to determine which member of a ring generated a signature. Second, it should be unforgeable: an adversary should be able to construct a valid signature on a ring of public keys only if he knows the secret key corresponding to one of them. Formalizing this intuition is tricky. Rivest, Shamir, and Tauman [18] gave a formalization which has been used in much subsequent work. Recently, Bender, Katz, and Morselli [4] described several possible stronger formulations of each notion.

Below, we show our scheme to be anonymous against full key exposure and unforgeable with respect to insider corruption. For both anonymity and unforgeability these are the strongest formulations considered by Bender, Katz, and Morselli. We now recall these formulations; see [4] for additional details and motivation.

RS.Kg. This randomized algorithm outputs a public verification key pk and a private signing key sk.

RS.Sig(pk, sk, R, M)**.** This algorithm takes as input a keypair (pk, sk) and a set of public keys R that constitutes the ring, along with a message M in some message space to be signed. It is required that $pk \in R$ hold. The algorithm returns a signature σ on M for the ring R.

RS.Vf(R, M, σ)**.** The verification algorithm takes as input a set of public keys R that constitutes the ring and a purported signature σ on a message M. It returns either `valid` or `invalid`.

Anonymity. Anonymity against full key exposure for a ring signature scheme $\mathcal{RS}$ is defined using the following game between a challenger and an adversary $\mathcal{A}$:

Setup. The challenger runs algorithm Kg l times to obtain keypairs $(pk_1, sk_1), \ldots, (pk_l, sk_l)$. In addition, the challenger records the random coins $\{\omega_i\}$ used in generating each keypair. Here l is a game parameter. The adversary $\mathcal{A}$ is given the public keys $\{pk_i\}$.

Signing Queries. Algorithm $\mathcal{A}$ is allowed to make ring signing queries of the form (s, R, M). Here M is the message to be signed, R is a set of public keys, and s is an index such that $pk_s \in R$ holds. (The other keys in R need not be keys in the set $\{pk_i\}$.) The challenger responds with $\sigma = \mathsf{Sig}(pk_s, sk_s, R, M)$.

Challenge. Algorithm $\mathcal{A}$ requests a challenge by sending to the challenger the values (i_0, i_1, R, M). Here M is to be signed with respect to the ring R, and i_0 and i_1 are indices such that $pk_{i_0}, pk_{i_1} \in R$. (The other keys in R need not be keys in the set $\{pk_i\}$.) The challenger chooses a bit $b \overset{\mathrm{R}}{\leftarrow} \{0,1\}$, computes the challenge signature $\sigma \leftarrow \mathsf{Sig}(pk_{i_b}, sk_{i_b}, R, M)$, and provides $\mathcal{A}$ with σ. In addition, the challenger provides $\mathcal{A}$ with the coins $\{\omega_i\}$ used to generate the keys; from these, $\mathcal{A}$ can recompute $\{sk_i\}$.

Output. Algorithm $\mathcal{A}$ finally outputs its guess b' for b, and wins if $b = b'$.

We define $\mathbf{Adv}^{\text{rsig-anon-ke}}_{\mathcal{RS},\mathcal{A}}$ to be the advantage over 1/2 of $\mathcal{A}$ in the above game.

Unforgeability. Unforgeability with respect to insider corruption for a ring signature scheme $\mathcal{RS}$ is defined using the following game between a challenger and an adversary $\mathcal{A}$:

Setup. The challenger runs algorithm Kg l times to obtain keypairs $(pk_1, sk_1), \ldots, (pk_l, sk_l)$. Here l is a game parameter. The adversary $\mathcal{A}$ is given the public keys $\{pk_i\}$. The challenger also initializes the set C of corrupted users as $C \leftarrow \emptyset$.

Queries. Algorithm $\mathcal{A}$ is allowed to make ring signing and corruption queries. A ring signing query is of the form (s, R, M). Here M is the message to be signed, R is a set of public keys, and s is an index such that $pk_s \in R$ holds. (The other keys in R need not be keys in the set $\{pk_i\}$.) The challenger responds with $\sigma = \mathsf{Sig}(pk_s, sk_s, R, M)$. A corruption query is of the form s, where s is again an index. The challenger provides sk_s to $\mathcal{A}$ and adds pk_s to C.

Output. Eventually, $\mathcal{A}$ outputs a tuple (R^*, M^*, σ^*) and wins the game if (1) it never made a ring signing query (s, R^*, M^*) for any s; (2) $R^* \subseteq \{pk_i\} \setminus C$; and (3) $\mathsf{Vf}(R^*, M^*, \sigma^*) = \texttt{valid}$.

We define $\mathbf{Adv}^{\text{rsig-uf-ic}}_{\mathcal{RS},\mathcal{A}}$ to be the probability that $\mathcal{A}$ wins in the above game.

Trusted Setup In our model we allow for a trusted global setup by an authority. This is a stronger setup assumption than what was used in previous results. However, this setup allows us to realize the benefits of an efficient scheme provably secure in the standard model. In practice the authority role can be split amongst several parties. For example, using techniques like those of Boneh and Franklin [7] several parties could generate a shared modulus n and group description efficiently for our scheme.

5 On Bender, Katz, and Morselli's Two-User Ring Signatures

Bender, Katz, and Morselli propose a ring signature secure without random oracles based on the Waters signature. This ring signature allows only two-signer rings, but this suffices for some applications of ring signatures, in particular designated-verifier signatures. Unlike the scheme we present, the BKM ring signature is proven unforgeable only against chosen-subring attacks. In this section, we recall the BKM ring signature and show that it is, in fact, insecure with respect to insider corruption.

We stress that Bender, Katz, and Morselli do not claim that their scheme is secure with respect to insider corruption. They prove security against chosen-subring attacks, a weaker notion, and this proof is correct. Our contribution in this section is to demonstrate a practical attack against the scheme in the more general model.

Consider a group G of prime order p, together with a bilinear map $e : G \times G \to G_T$, where G_T is also of size p. (This is unlike the composite-order setup of our paper.) Each user has a private key $\alpha \in \mathbb{Z}_p$ and a public key that includes $g_1 = g^\alpha$ and her own Waters hash generators $u', u_1, \ldots, u_k \in G$. Now, if Alice wishes to sign a message $M = (m_1, \ldots, m_k)$ in a ring that comprising her and Bob, whose public key is $(\bar{g}_1,\ \bar{u}', \bar{u}_1, \ldots, \bar{u}_k)$, she picks $r \xleftarrow{\text{R}} \mathbb{Z}_p$ and computes

$$S_1 \leftarrow (\bar{g}_1)^\alpha \cdot \big(u' \prod\nolimits_{i=1}^{k} u_i^{m_i}\big)^r \cdot \big(\bar{u}' \prod\nolimits_{i=1}^{k} \bar{u}_i^{m_i}\big)^r \qquad \text{and} \qquad S_2 \leftarrow g^r \ .$$

For any two users, the values $(g_1, \bar{g}_1)$ act like the Waters public key (g_1, g_2); the value $g^{\alpha\bar{\alpha}}$ acts as a shared signing key. Since either user is capable of computing this value, anonymity is unconditional. Unforgeability against chosen-subring attacks follows from the security of the underlying Waters signature.

This scheme has the advantage that it requires no shared setup beyond the group description. This justifies making each signer generate and publish her own Waters hash generators, since a third party trusted with generating them for all users could use its knowledge of their discrete logs to recover the shared signing key $g^{\alpha\bar{\alpha}}$ from any signature.

The unforgeability condition under which Bender, Katz, and Morselli prove their scheme secure does not allow for adversarially generated keys. We show that the scheme is in fact insecure against such attacks, which, for a two-user ring signature, have the following form: Alice and Bob publish their public keys. Then Veronica publishes her key and tricks Alice into signing a message for the Alice-Veronica ring; what she learns from this signature allows her to forge an Alice-Bob ring signature.

Suppose Alice's public key is $(g_1,\ u', u_1, \ldots, u_k)$ and Bob's is $(\bar{g}_1,\ \bar{u}', \bar{u}_1, \ldots, \bar{u}_k)$. In our attack, Veronica picks $s, t', t_1, \ldots, t_k \xleftarrow{\text{R}} \mathbb{Z}_p$ and sets

$$\hat{g}_1 \leftarrow \bar{g}_1 \cdot g^s \qquad \text{and} \qquad \hat{u}' \leftarrow g^{t'}/u' \qquad \text{and} \qquad \hat{u}_i \leftarrow g^{t_i}/u_i \quad 1 \le i \le k \ .$$

Now when Alice generates an Alice-Veronica ring signature on a message $M = (m_1, \ldots, m_k)$ we will have

$$S_1 = (\hat{g}_1)^\alpha \cdot \big(u'\hat{u}' \prod\nolimits_{i=1}^{k} (u_i \hat{u}_i)^{m_i}\big)^r = (\bar{g}_1)^\alpha (g^s)^\alpha (g^t)^r$$

where $t = t' + \sum_{i=1}^{k} m_i t_i$, and Veronica recovers the shared Alice-Bob signing key $g^{\alpha\bar{\alpha}}$ as $S_1/(g_1^s S_2^t)$.

Note that Veronica need not know the discrete logarithms of all her Waters generators. It suffices for her to pick $\hat{u}'$ specially while letting the rest be globally specified. In this variant, Veronica picks ahead of time a message $M^* = (m_1^*, \ldots, m_k^*)$ that she thinks she can trick Alice into signing. She then chooses $s, t' \xleftarrow{\text{R}} \mathbb{Z}_p$, and computes

$$\hat{g}_1 \leftarrow \bar{g}_1 \cdot g^s \cdot \qquad \text{and} \qquad \hat{u}' \leftarrow g^{t'} \,/\, \left(u' \prod_{i=1}^{k} (u_i \hat{u}_i)^{m_i^*}\right) \ .$$

Now, when Alice generates an Alice-Veronica ring signature on M^*, we have $S_1 = (\bar{g}_1)^\alpha (g^s)^\alpha (g^{t'})^r$, from which Veronica can recover $g^{\alpha\bar{\alpha}}$.

The attack described above is prevented if all users share the same Waters generators $(u', u_1, \ldots, u_k)$; but even in this case Veronica can still obtain from Alice an Alice-Bob ring signature when Alice thinks she is generating an Alice-Veronica ring signature. To achieve this, Veronica chooses $s \xleftarrow{\text{R}} \mathbb{Z}_p$ and sets $\hat{g}_1 \leftarrow (\bar{g}_1)^s$. Now an Alice-Veronica ring signature on $M = (m_1, \ldots, m_k)$ will have the form

$$S_2 = g^r \qquad \text{and} \qquad S_1 = (\hat{g}_1)^\alpha \cdot \left(u' \prod_{i=1}^{k} u_i^{m_i}\right)^r = (\bar{g}_1^\alpha)^s \cdot \left(u' \prod_{i=1}^{k} u_i^{m_i}\right)^r \ ,$$

and therefore $(S_1^{1/s}, S_2^{1/s})$ is an Alice-Bob ring signature on M with randomness r/s.

Attack on the Camenisch-Lysyanskaya–Based Scheme. In the full version of their paper [3], Bender, Katz, and Morselli also give a two-user ring signature based on Camenisch-Lysyanskaya signatures [12]. As with their Waters-based scheme, they claim and prove security against chosen-subring attacks. Here, we show an attack on this ring signature similar to the attack above, again with respect to insider corruption. We stress once more that Bender, Katz, and Morselli do not claim security in this stronger model.

Suppose that Alice and Bob have respective secret keys x and y, and public keys $X = g^x$ and $Y = g^y$. Their ring signature on a message $m \in \mathbb{Z}_p$ is (a, a^y, a^{x+mxy}), where a is random in G. If $a = g^r$ with $r \in \mathbb{Z}_p$ then Alice computes the ring signature as $(a, Y^r, a^x Y^{mxr})$ and Bob as (a, a^y, X^{r+mry}). If Veronica plays the part of Alice, she publishes as her key $\hat{X} = X^s$ for $s \xleftarrow{\text{R}} \mathbb{Z}_p$. Bob then generates the Veronica-Bob signature $(S_1, S_2, S_3) = (a, a^y, a^{sx+msxy})$, from which Veronica can produce an Alice-Bob ring signature on m as $(S_1, S_2, S_3^{1/s})$. If Veronica plays the part of Bob, she publishes as her key $\hat{Y} = Y^s$ for $s \xleftarrow{\text{R}} \mathbb{Z}_p$. Alice then generates the Alice-Veronica signature $(S_1, S_2, S_3) = (a, a^{sy}, a^{x+mxsy})$, from which Veronica can produce an Alice-Bob ring signature on $m' = ms$ as $(S_1, S_2^{1/s}, S_3)$.

Implications for Designated-Verifier Signatures. The attack described above demonstrates a trade-off between our Waters-based ring signature and the BKM one. Our scheme requires a trusted setup, but achieves security even in the presence of adversarially generated keys. This is important for designated-verifier signatures, the main proposed application for two-user ring signatures, since there is no reason that Alice will only wish to designate as verifiers users whose keys she trusts to have been properly generated.

6 Our Ring Signature Construction

In this section, we describe our ring signature scheme. As noted in the introduction, in our ring signature all the users' keys must be defined in a group G of composite order. That group must be set up by a trusted authority, since the factorization of its order n must be kept secret. In addition to setting up the group G, the setup authority must also set up some additional parameters, using a global setup algorithm we now describe.

Global Setup. The trusted ring signing setup algorithm first constructs a group G of composite order $n = pq$ as described in Section 2 above. It then chooses exponents $a, b_0 \xleftarrow{\text{R}} \mathbb{Z}_n$ and sets

$$A \leftarrow g^a \qquad \text{and} \qquad B_0 \leftarrow g^{b_0} \qquad \text{and} \qquad \hat{A} \leftarrow h^a \ .$$

Let $H : \{0,1\}^* \to \{0,1\}^k$ be a collision-resistant hash function. The setup algorithm picks Waters hash generators

$$u', u_1, u_2, \ldots, u_k \xleftarrow{\text{R}} G \ .$$

The published common reference string includes a description of the group G and of the collision-resistant hash H, along with $(A, B_0, \hat{A})$ and $(u', u_1, \ldots, u_k)$. The factorization of n is not revealed. Note that anyone can use the pairing to verify that the pair $(A, \hat{A})$ is properly formed.

The Scheme. Individual users now use the public parameters published by the setup algorithm in generating their keys, signing, and verifying. The algorithms they use are as follows.

LRS.Kg. Choose a random exponent $b \xleftarrow{\text{R}} \mathbb{Z}_n$; set $pk \leftarrow g^b \in G$ and $sk \leftarrow A^b \in G$.

Recall that in the variant of the Water's signature scheme that we use a public key is a pair of group elements in G. Here, one of the two group elements for a user's key is always the global setup value A. In effect the user's public key is like the Water's public key A, g^b. However, all users share the element A.

LRS.Sig(pk, sk, R, M)**.** The signing algorithm takes as input a message $M \in \{0,1\}^*$, a ring R of public keys, and a keypair $(pk, sk) \in G^2$. No key may appear twice in R, and R must include pk.

Compute $(m_1, \ldots, m_k) \leftarrow H(M, R)$. Let $l = |R|$; parse the elements of R as $v_i \in G$, $1 \le i \le l$. Let i^* be the index such that $v_{i^*} = pk$. Define $\{f_i\}_{i=1}^{l}$ as

$$f_i = \begin{cases} 1 & \text{if } i = i^*, \\ 0 & \text{otherwise.} \end{cases}$$

Now for each i, $1 \le i \le l$, choose a random exponent $t_i \xleftarrow{\text{R}} \mathbb{Z}_n$ and set

$$C_i \leftarrow (v_i/B_0)^{f_i} h^{t_i} \qquad \text{and} \qquad \pi_i \leftarrow \big((v_i/B_0)^{2f_i - 1} h^{t_i}\big)^{t_i} \ .$$

As in the papers of Groth, Ostrovsky, and Sahai [16] and Boyen and Waters [10], the value π_i acts as a proof that C_i is well-formed — here, specifically, that $f_i \in \{0,1\}$. Let $C \leftarrow \prod_{i=1}^{l} C_i$ and $t \leftarrow \sum_{i=1}^{l} t_i$. Observe that, when there is exactly one non-zero value amongst $\{f_i\}$, viz., f_{i^*}, we have $B_0 C = (v_{i^*})(h^t)$, so C serves as an encryption of the user's public key. (The role of B_0 is discussed below.) Finally, choose $r \xleftarrow{\text{R}} \mathbb{Z}_n$ and compute

$$S_1 \leftarrow sk \cdot \big(u' \prod_{j=1}^{k} u_j^{m_j}\big)^r \cdot \hat{A}^t \qquad \text{and} \qquad S_2 \leftarrow g^r$$

The signature is output as $\sigma = \big((S_1, S_2),\ \{(C_i, \pi_i)\}_{i=1}^{l}\big) \in G^{2l+2}$.

LRS.Vf(R, M, σ)**.** Compute $(m_1, \ldots, m_k) \leftarrow H(M, R)$. Let $l = |R|$; parse the elements of R as $v_i \in G$, $1 \le i \le l$. Verify that no element is repeated in R and reject otherwise. Parse the signature σ as $\big((S_1, S_2), \{(C_i, \pi_i)\}_{i=1}^{l}\big) \in G^{2l+2}$. (If this parse fails, reject.) Check first that the proofs $\{\pi_i\}$ are valid: for each i, $1 \le i \le l$, that

$$e\big(C_i,\, C_i/(v_i/B_0)\big) \overset{?}{=} e(h, \pi_i) \tag{3}$$

holds. If any of the proofs is invalid, reject. Otherwise, set $C \leftarrow \prod_{i=1}^{l} C_i$. Accept if the following equation is satisfied:

$$e(A, B_0 C) \overset{?}{=} e(S_1, g) \cdot e\big(S_2^{-1},\, u' \prod_{j=1}^{k} u_j^{m_j}\big) \ . \tag{4}$$

Discussion As outlined in the introduction, in our ring signature scheme we wish to prove that the value C which is computed by multiplying all C_i values together contains an encryption of exactly one key from the ring. This can be done by both using GOS proofs to show that each C_i is either an encryption of the proper public key or the identity element and that exactly one of these is not the identity element. (If every C_i were an encryption of the identity element everywhere, the public key encrypted in C would be the identity element and trivial for the adversary to forge under.)

Instead of directly proving this, which would require larger — though still $O(l)$-sized[3] — proofs, we have the user prove that each C_i is an encryption of the identity element or the i-th public key in the ring times some group element B_0 given by the setup algorithm. Thus, C will be an encryption of a public key times B_0. Now a signer will instead prove that the signature verifies under the encrypted key divided by B_0, which is the signers original public key.. In this way if a forger attempts to forge by letting all C_i be encryptions of the identity element, he will need to forge under the public key B_0.

7 Security

We now prove that our ring signature scheme is anonymous against full key exposure and unforgeable with respect to insider corruption.

7.1 Anonymity

The anonymity proof closely follows that given by Boyen and Waters for their group signature [10].

Theorem 7.1. *Our ring signature scheme is anonymous against full key exposure if SGH is hard.*

Proof. The proof proceeds in games. We define Games 0 and 1 as follows. In Game 0, h is chosen uniformly from G_q; in Game 1, h is chosen uniformly from G.

[3] A possible circuit is as follows. Let $\{f_i\}$ be the indicator variables. Let $c_0^1 = c_0^2 = 0$, and for $i \ge 1$ compute c_i^1 and c_i^2 as $c_i^1 \leftarrow c_{i-1}^1 \vee f_i$ and $c_i^2 \leftarrow c_{i-1}^2 \vee (f_i \wedge c_{i-1}^1)$. Finally, prove that $c_l^1 = 1$ and $c_l^2 = 0$.

Games 0 and 1. Algorithm $\mathcal{B}$ is given: the group order n (but not its factorization); the description of the group G, together with generators g of G and h; in Game 0, h is chosen from G_q; in Game 1, h is chosen from all of G. Algorithm $\mathcal{B}$ chooses a collision resistant hash function $H : \{0,1\}^* \to \{0,1\}^k$. It follows the setup algorithm above to obtain system parameters $(A, B_0, \hat{A})$ and $(u', u_1, \dots, u_k)$. Algorithm $\mathcal{B}$ then runs Kg l times to obtain keypairs $\{(pk_i, sk_i)\}_{i=1}^{l}$, recording in addition the randomnesses $\{b_i\}$ used in each run.

Algorithm $\mathcal{B}$ runs $\mathcal{A}$, providing to it the following: the description of the group G, including the generators g and h; the common parameters $(A, B_0, \hat{A})$ and $(u', u_1, \dots, u_k)$, along with the description of H; and the challenge public keys $\{pk_i\}_{i=1}^{l}$. When $\mathcal{A}$ makes a signing query of the form (s, R, M), $\mathcal{A}$ responds with $\sigma = \mathsf{Sig}(pk_s, sk_s, R, M)$. Finally, $\mathcal{A}$ requests a challenge with the values (i_0, i_1, R, M). Algorithm $\mathcal{B}$ chooses a bit $b \xleftarrow{R} \{0,1\}$, computes the challenge signature $\sigma \leftarrow \mathsf{Sig}(pk_{i_b}, sk_{i_b}, R, M)$, and provides $\mathcal{A}$ with σ. In addition, the challenger provides $\mathcal{A}$ with the random coins $\{b_i\}$ used to generate the private keys. Algorithm $\mathcal{A}$ finally outputs its guess b' for b; $\mathcal{B}$ outputs 1 if $b = b'$, 0 otherwise.

Discussion. Denote by $\mathbf{Adv}_{\mathcal{B}}^{\text{game-0}}$ the advantage $\mathcal{B}$ has over 1/2 in Game 0, and by $\mathbf{Adv}_{\mathcal{B}}^{\text{game-1}}$ the advantage over 1/2 it has in Game 1. Clearly, we have

$$\mathbf{Adv}_{\mathcal{B}}^{\text{game-0}} = \mathbf{Adv}_{\mathcal{LRS},\mathcal{A}}^{\text{rsig-anon-ke}} \quad , \tag{5}$$

since in Game 0 $\mathcal{A}$'s environment is exactly as specified in the anonymity game. Moreover, suppose that $\mathcal{B}$́s output were different in the two games. Then we could use $\mathcal{B}$, with $\mathcal{A}$ as a subroutine, to solve SGH: given generators (g, h) to test, we provide them to $\mathcal{B}$ and output 1 if $\mathcal{B}$ does. This gives a new algorithm $\mathcal{C}$ for which we have

$$\begin{aligned}\mathbf{Adv}_{\mathcal{C}}^{\text{sgh}} &= \Big|\Pr\big[\mathcal{B} = 1 \mid h \xleftarrow{R} G_p\big] - \Pr\big[\mathcal{B} = 1 \mid h \xleftarrow{R} G\big]\Big| \\ &= \frac{1}{2}\Big|\big(2\Pr\big[\mathcal{B} = 1 \mid h \xleftarrow{R} G_p\big] - 1\big) - \big(2\Pr\big[\mathcal{B} = 1 \mid h \xleftarrow{R} G\big] - 1\big)\Big| \\ &= \frac{1}{2}\big|\mathbf{Adv}_{\mathcal{B}}^{\text{game-0}} - \mathbf{Adv}_{\mathcal{B}}^{\text{game-1}}\big| \quad . \end{aligned}\tag{6}$$

But now, we argue that $\mathbf{Adv}_{\mathcal{B}}^{\text{game-1}} = 0$, even if $\mathcal{A}$ is computationally unbounded. Consider the distinguishing challenge $\big((S_1, S_2), \{(C_i, \pi_i)\}_{i=1}^{l'}\big) \in G^{2l'+2}$. For each i, we have $C_i = (v_i/B_0)^{f_i} h^{t_i}$ with $f_i \in \{0,1\}$ and $t_i \in \mathbb{Z}_n$. But when h is a generator of G there exist $\tau_{i0}, \tau_{i1} \in \mathbb{Z}_n$ such that $C_i = (v_i/B_0) h^{\tau_{i1}} = h^{\tau_{i0}}$ and, moreover, denoting by $(\pi_i \mid f_i = b)$ the value which π_i is assigned if f_i is set to $b \in \{0,1\}$, we have

$$(\pi_i \mid f_i = 1) = ((v_i/B_0)^1 h^{\tau_{i1}})^{\tau_{i1}} = (h^{\tau_{i0}})^{\tau_{i1}} = (h^{\tau_{i1}})^{\tau_{i0}} = ((v_i/B_0)^{-1} h^{\tau_{i0}})^{\tau_{i0}} = (\pi_i \mid f_i = 0) \quad ,$$

so for each i the pair (C_i, π_i) is consistent with either $f_i = 0$ or $f_i = 1$, and $\mathcal{A}$ can gain no information from this part of the signature. The value $S_2 = g^r$ is unrelated to the choice of signer. Thus if $\mathcal{A}$ can gain information, it is only from S_1. But, having fixed S_2 and $\{(C_i, \pi_i)\}$, S_1 is the unique value satisfying (4). Specifically, letting $A = g^a$, $S_2 = g^r$, and $C/B_0 = g^c$ (all of which a computationally unbounded adversary can calculate), we have $S_1 = g^{ac} \cdot \big(u' \prod_{j=1}^{k} u_j^{m_j}\big)^r$. Thus this value gives no information about whether sk_{i_0} or sk_{i_1} was used to generate the challenge signature, and $\mathcal{A}$ can do no better than guess b. This establishes

$$\mathbf{Adv}_{\mathcal{B}}^{\text{game-1}} = 0 \tag{7}$$

Putting equations (5), (6), and (7) together, we see that

$$\mathbf{Adv}^{\text{rsig-anon-ke}}_{\mathcal{LRS},\mathcal{A}} \le 2\mathbf{Adv}^{\text{sgh}}_{\mathcal{C}} \ .$$

To interpret this result concretely, we note that the algorithm $\mathcal{B}$ used in the reduction took $O(1)$ operations to set up and obtain the result, and $O(1)$ time to answer each of $\mathcal{A}$'s queries. To interpret it asymptotically, we introduce the security parameter that we have suppressed, note that the reduction is polynomial-time in that parameter, and observe that if $\mathbf{Adv}^{\text{rsig-anon-ke}}_{\mathcal{LRS},\mathcal{A}}$ is non-negligible, then so is $\mathbf{Adv}^{\text{sgh}}_{\mathcal{C}}$. Either interpretation implies the result stated informally in the theorem. □

7.2 Unforgeability

We show that our ring signature scheme is unforgeable. We present a proof sketch here, with the full proof relegated to Appendix A.

Theorem 7.2. *Our ring signature scheme is unforgeable with respect to insider corruption if H is collision resistant and CDH is hard in G_p.*

Proof sketch. The algorithm which makes the reduction is given the factorization of n. Using this and standard Chinese Remaindering techniques, it can decompose an element of G into its G_p and G_q constituents. This allows it to undo BGN blinding with h^t terms, and to recover from a signature the values f_i used in generating it. Each of these must be in the set $\{0,1\}$ by the perfect soundness of GOS proofs.

First, we ensure that any forgery (M^*, R^*) is such that $H(M^*, R^*) \neq H(M, R)$ for the adversary's previous signing queries. This is easy: an adversary for which this is not true would break the collision resistance of H.

Having disposed of this possibility, we distinguish between two types of adversaries. Consider the values $\{f_i\}$ which we recover from the forgery. For the first type of adversary, the number of i's such that $f_i = 0$ is either 0 or more than 1. For the second type, exactly one f_i equals 1.

For the first type of adversary, we note that each C_i such that $f_i = 1$ contributes a (v_i/B_0) term to the encrypted Waters key $C = \prod C_i$. Thus the Waters key under which the encrypted signature (S_1, S_2) is verified, B_0C, will include a B_0^{1-f} term, where $f = \sum f_i \neq 1$ for this type of adversary. Thus if we embed a CDH challenge in A and B_0, but construct the Waters hash generators $(u', u_1, \ldots, u_k)$ and user keys $\{v_i\}$ so that we know their discrete logarithms, we will obtain from the forgery values (S_1, S_2) such that $e(A,\, B_0^{1-f} \cdot \eta^r) = e(S_1, \eta) \cdot e\big(S_2^{-1}, \eta^x\big)$, where r and x are numbers we compute. From this we easily obtain the CDH solution. (Because we must project into G_p to recover the f_i's, we obtain a CDH solution in G_p rather than G, which is why the generator η of G_p has replaced g in the verification equation above.)

For the second type of adversary, we obtain a Waters signature forgery. Given the challenge Waters public key (η_1, η_2), which again is in G_p, and the Waters hash generators $(\hat{u}', \hat{u}_1, \ldots, \hat{u}_k)$, we place η_1 in A, adding a random G_q component so that A spans G, and pick B_0 arbitrarily. We similarly extend the challenge waters hash generators to G. We pick all the user keys arbitrarily except one, which we instantiate using η_2, properly extended to G. Now we can handle corruption queries for every user except the special one. Signing queries we can answer directly for the normal users, and can answer for the special user using our Waters signing oracle. This oracle returns a signature $(\hat{S}_1, \hat{S}_2) \in G_p^2$; extending this to a properly-blinded signature in G takes a bit of work, but isn't terribly difficult. The index of the special user is kept hidden from the adversary, so with probability $1/l$ he doesn't make a corruption query for that user but then does make his forgery for that user. (Recall that this type of adversary always has exactly one user for which $f_i = 1$.) We convert the adversary's forgery to a Waters signature forgery in G_p. Because $H(M^*, R^*)$ is different for this forgery than for previous signing queries, the forgery is nontrivial.

Thus we obtain from a ring signature forging adversary a break of either the collision resistance of H or the CDH hardness of G_p or (with a $1/l$ loss of advantage) to the unforgeability of the Waters signature in G_p. However, the Waters signature in G_p is unforgeable if CDH is hard in G_p, and the theorem statement follows. $\square$

8 Conclusions and Open Problems

We presented the first efficient ring signatures that are provably secure without random oracles under standard assumptions. Signatures in our scheme are of size $2l + 2$ group elements for l members in a ring. We showed our signatures to be secure for the strongest definitions of security. Two interesting open problems remain: to obtain a ring signature secure without random oracles where (1) user keys need not be generated in a particular shared group; or (2) signature length is independent of the number of signers implicated in the ring.

References

[1] M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. In Y. Zheng, editor, *Proceedings of Asiacrypt 2002*, volume 2501 of *LNCS*, pages 415–32. Springer-Verlag, Dec. 2002.

[2] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. Denning, R. Pyle, R. Ganesan, R. Sandhu, and V. Ashby, editors, *Proceedings of CCS 1993*, pages 62–73. ACM Press, Nov. 1993.

[3] A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. Cryptology ePrint Archive, Report 2005/304, 2005. `http://eprint.iacr.org/`.

[4] A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In S. Halevi and T. Rabin, editors, *Proceedings of TCC 2006*, volume 3876 of *LNCS*, pages 60–79. Springer-Verlag, Mar. 2006.

[5] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 223–38. Springer-Verlag, May 2004.

[6] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In M. Franklin, editor, *Proceedings of Crypto 2004*, volume 3152 of *LNCS*, pages 443–59. Springer-Verlag, Aug. 2004.

[7] D. Boneh and M. Franklin. Efficient generation of shared RSA keys. *J. ACM*, 48(4):702–22, July 2001.

[8] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, *Proceedings of Eurocrypt 2003*, volume 2656 of *LNCS*, pages 416–32. Springer-Verlag, May 2003.

[9] D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In J. Kilian, editor, *Proceedings of TCC 2005*, number 3378 in LNCS, pages 325–41. Springer-Verlag, Feb. 2005.

[10] X. Boyen and B. Waters. Compact group signatures without random oracles. In S. Vaudenay, editor, *Proceedings of Eurocrypt 2006*, volume 4004 of *LNCS*, pages 427–44. Springer-Verlag, May 2006.

[11] E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. In M. Yung, editor, *Proceedings of Crypto 2002*, volume 2442 of *LNCS*, pages 465–80. Springer-Verlag, Aug. 2002.

[12] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In M. Franklin, editor, *Proceedings of Crypto 2004*, volume 3152 of *LNCS*, pages 56–72. Springer-Verlag, Aug. 2004.

[13] D. Chaum and E. van Heyst. Group signatures. In D. W. Davies, editor, *Proceedings of Eurocrypt 1991*, volume 547 of *LNCS*, pages 257–65. Springer-Verlag, Apr. 1991.

[14] S. Chow, J. Liu, V. Wei, and T. H. Yuen. Ring signatures without random oracles. In S. Shieh and S. Jajodia, editors, *Proceedings of ASIACCS 2006*, pages 297–302. ACM Press, Mar. 2006.

[15] Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad hoc groups. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 609–26. Springer-Verlag, May 2004.

[16] J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for NP. In S. Vaudenay, editor, *Proceedings of Eurocrypt 2006*, volume 4004 of *LNCS*, pages 339–58. Springer-Verlag, May 2006.

[17] J. Herranz and G. Sáez. Forking lemmas for ring signature schemes. In T. Johansson and S. Maitra, editors, *Proceedings of Indocrypt 2003*, volume 2904 of *LNCS*, pages 266–79. Springer-Verlag, Dec. 2003.

[18] R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In C. Boyd, editor, *Proceedings of Asiacrypt 2001*, volume 2248 of *LNCS*, pages 552–65. Springer-Verlag, Dec. 2001.

[19] R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret: Theory and applications of ring signatures. In O. Goldreich, A. Rosenberg, and A. Selman, editors, *Essays in Theoretical Computer Science: in Memory of Shimon Even*, volume 3895 of *LNCS Festschrift*. Springer-Verlag, 2006. To appear. Online: `http://www.mit.edu/~tauman/ringsig-book.pdf`.

[20] B. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *Proceedings of Eurocrypt 2005*, volume 3494 of *LNCS*, pages 114–27. Springer-Verlag, May 2005.

[21] J. Xu, Z. Zhang, and D. Feng. A ring signature scheme using bilinear pairings. In C. H. Lim and M. Yung, editors, *Proceedings of WISA 2004*, volume 3325 of *LNCS*, pages 160–9. Springer-Verlag, Aug. 2004.

A Proof of Theorem 7.2

Proof. We show how to construct, based on a forger algorithm $\mathcal{A}$, an algorithm $\mathcal{B}$ that breaks one of our complexity assumptions. Algorithm $\mathcal{B}$ is given the factorization $n = pq$.

We begin by making some observations about working with a group of composite order; these follow trivially from the Chinese Remainder Theorem.

1. Let δ_p be such that $\delta_p = 0 \bmod q$ and $\delta_p = 1 \bmod p$. Now for all $z \in G$ we have $z^{\delta_p} \in G_p$; specifically, $z^{\delta_p} = 1$ iff $z \in G_q$. Now, for some ring signature, consider a pair (C_i, π_i) that satisfies (3). By the perfect soundness of Groth-Ostrovsky-Sahai proofs, we know that $C_i = (v_i/B_0)^{f_i} h^{t_i}$ with $f_i \in \{0, 1\}$. But by the observation above we have $C_i^{\delta_p} = 1$ iff $f_i = 0$. Thus, given a valid ring signature, $\mathcal{B}$ can recover the values $\{f_i\}$. (This is provided that $v_i/B_0 \notin G_q$. For a nontrivial forgery, however, each v_i will come from the set of challenge keys, which $\mathcal{B}$ will choose so that this property holds.)

2. Given generators η of G_p and h of G_q, $\eta^{r_1} h^{r_2}$ is a generator of G for all $r_1 \in \mathbb{Z}_p^*$, $r_2 \in \mathbb{Z}_q^*$. By drawing r_1, r_2 at random, we obtain a random generator of G. If η is kept independent of the adversary's view, we can set $r_1 = 1$ and maintain the randomness of the generator; similarly for h and r_2.

3. Given some element $u \in G$ and generator g as above, the representations $u = g^a$ with $a \in \mathbb{Z}_n$ and $u = \eta^x h^y$ with $(x, y) \in \mathbb{Z}_p \times \mathbb{Z}_q$ are one-to-one; and u's projections into G_p and G_q are η^x and h^y, respectively.

4. For all $u \in G_p$ and $v \in G_q$, we have $e(u, v) = 1$. This implies that for all $u_1, u_2 \in G_p$ and $v_1, v_2 \in G_q$, we have $e(u_1 v_1, u_2 v_2) = e(u_1, u_2) \cdot e(v_1, v_2)$.

With this in mind, we distinguish three types of adversaries:

Type I. Amongst its signing queries and its forgery, $\mathcal{A}$ issues two message–ring pairs (M, R) and (M', R') such that $(M, R) \neq (M', R')$ but $H(M, R) = H(M', R')$.

Type II. Algorithm $\mathcal{A}$ never causes a hash collision as above. Further, its forgery is *not* such that exactly one of the exponents $\{f_i\}$ equals 1: either $f_i = 0$ for all i or $\sum_i f_i > 1$.

Type III. Algorithm $\mathcal{A}$ never causes a hash collision as above, and its forgery is such that exactly one of the exponents $\{f_i\}$ equals 1.

We handle each type of adversary separately.

Type-I Adversary. We convert a Type-I forger $\mathcal{A}$ to a break of the collision-resistance of H. Algorithm $\mathcal{B}$ simply follows the unforgeability game of Section 4 using the algorithms specified above for our ring signature. In addition, $\mathcal{B}$ keeps track of the pairs $(M^{(i)}, R^{(i)})$ included in $\mathcal{A}$'s ring signing queries, and the pair (M^*, R^*) included in its forgery. Since $\mathcal{A}$ is a Type-I adversary, these include two pairs (M, R) and (M', R') such that $H(M, R) = H(M', R')$. Algorithm $\mathcal{B}$ outputs this pair as a collision. Clearly, $\mathcal{B}$ succeeds whenever $\mathcal{A}$ does, so we have established that — for a Type-I adversary — we have

$$\mathbf{Adv}^{\text{rsig-uf-ic}}_{\mathcal{LRS},\mathcal{A}} \le \mathbf{Adv}^{\text{coll}}_{H,\mathcal{B}} \ . \tag{8}$$

Type-II Adversary. We convert a Type-II forger $\mathcal{A}$ to a CDH breaker for the subgroup G_p. Algorithm $\mathcal{B}$ is given: the group order n and its factorization $n = pq$; the description of the group G, together with generators η of G_p and h of G_q; and, in addition, a pair $(\eta^\alpha, \eta^\beta) \in G_p^2$. Its goal is to compute $\eta^{\alpha\beta}$. Algorithm $\mathcal{B}$ chooses a collision resistant hash function $H : \{0,1\}^* \to \{0,1\}^k$ as above. It picks $r_1 \overset{\text{R}}{\leftarrow} \mathbb{Z}_q^*$ and sets $g \leftarrow \eta h^{r_1}$; this is a random generator of G, by the observation above. Now $\mathcal{B}$ picks $r_2, r_3 \overset{\text{R}}{\leftarrow} \mathbb{Z}_q$ and sets

$$A \leftarrow \eta^\alpha h^{r_2} \qquad \text{and} \qquad \hat{A} \leftarrow h^{r_2/r_1} \qquad \text{and} \qquad B_0 \leftarrow \eta^\beta h^{r_3} \ ;$$

here $\hat{A}$ is correctly formed by the observation that begins the proof; indeed, we have $e(A, h) = e(h, h)^{r_2} = e(h^{r_2/r_1}, h^{r_1}) = e(\hat{A}, g)$, where we have twice used the fact that $e(\eta, h) = 1$.

Algorithm $\mathcal{B}$ picks exponents $x', x_1, x_2, \ldots, x_k \overset{\text{R}}{\leftarrow} \mathbb{Z}_n$, and sets $u' \leftarrow g^{x'}$ and, for $1 \le j \le k$, $u_j \leftarrow g^{x_j}$. Finally, $\mathcal{B}$ generates the user keys by choosing for each i, $1 \le i \le l$, random exponents $b_i \overset{\text{R}}{\leftarrow} \mathbb{Z}_n$, and setting $pk_i \leftarrow g^{b_i}$ and $sk_i \leftarrow A^{b_i}$. Algorithm $\mathcal{B}$ runs $\mathcal{A}$, providing to it the following: the description of the group G, including the generators g and h; the common parameters $(A, B_0, \hat{A})$ and $(u', u_1, \ldots, u_k)$, along with the description of H; and the challenge public keys $\{pk_i\}_{i=1}^{l}$.

When $\mathcal{A}$ makes a corruption query at index s, $1 \le s \le l$, $\mathcal{B}$ responds with sk_s. When $\mathcal{A}$ makes a ring signing query of the form (s, R, M), $\mathcal{A}$ responds by computing $\sigma \leftarrow \mathsf{Sig}(pk_s, sk_s, R, M)$. If the signing algorithm returns an error condition, indicating that the request is invalid, $\mathcal{B}$ declines to answer the signing query; otherwise it responds with σ.

Finally, $\mathcal{A}$ outputs a forgery σ^* on a ring R^* for message M^*. Let $l^* = |R^*|$. Parse the ring R^* as $(v_1, \ldots, v_{l^*})$. Since the forgery is nontrivial, none of the public keys is repeated, and there is a mapping $t : [1, l^*] \to [1, l]$ such that $v_i = pk_{t(i)}$ for each i, $1 \le i \le l^*$. Moreover, $\mathcal{A}$ must not have made a corruption query at any of the indices $\{t(i)\}_{i=1}^{l^*}$, though this is not important in this case.

Algorithm $\mathcal{B}$ parses σ^* as $\big((S_1, S_2), \{(C_i, \pi_i)\}_{i=1}^{l^*}\big) \in G^{2l^*+2}$. Again, since the forgery is nontrivial, this parse must succeed. Each pair (C_i, π_i) must satisfy (3), and all the conditions hold for recovering $\{f_i\}_{i=1}^{l^*}$ from $\{C_i\}$ as described at the beginning of the proof, and we have, for each i,

$$C_i^{\delta_p} = \big(v_i^{\delta_p}/B_0^{\delta_p}\big)^{f_i} = \big((g^{b_{t(i)}})^{\delta_p}/B_0^{\delta_p}\big)^{f_i} = \big(\eta^{b_{t(i)}}/\eta^\beta\big)^{f_i}\big) \ . \tag{9}$$

Let $b = \sum_{i=1}^{l^*} f_i b_{t(i)}$ and $f = \sum_{i=1}^{l^*} f_i$. Then by (9)

$$C^{\delta_p} = \prod_{i=1}^{l^*} C_i^{\delta_p} = \eta^b/(\eta^\beta)^f \ . \tag{10}$$

Finally, $\mathcal{B}$ computes $(m_1, \ldots, m_k) \leftarrow H(M^*, R^*)$ and lets $x \leftarrow x' + \sum_{j=1}^{k} m_i x_i$. This value is such that $u' \prod_{j=1}^{k} u_j^{m_j} = g^x$. Since the forgery is valid, the verification equation (4) must hold;

raising both sides to power δ_p and using (10) gives[4]

$$e((\eta^\alpha),\,(\eta^\beta)\cdot\eta^b/(\eta^\beta)^f) = e(S_1^{\delta_p},\eta)\cdot e\big((S_2^{\delta_p})^{-1},\,\eta^x\big)$$

or, rearranging,

$$e(\eta^\alpha,\eta^\beta)^{1-f}\cdot e(\eta^\alpha,\eta)^b = e((\eta^\alpha)^{-b}\cdot S_1^{\delta_p}\cdot(S_2^{\delta_p})^{-x},\,\eta)\ ,$$

so the answer to the CDH problem is

$$\big[(\eta^\alpha)^{-b}\cdot S_1^{\delta_p}\cdot(S_2^{\delta_p})^{-x}\big]^{1/(1-f)}\ ;$$

we are allowed to take $(1-f)$-th roots of both sides because we are guaranteed, for a Type-II adversary, that $f\neq 1$. Algorithm $\mathcal{B}$ performs this calculation and outputs the result.

Clearly, $\mathcal{B}$ succeeds whenever $\mathcal{A}$ does, so we have established that — for a Type-II adversary — we have

$$\mathbf{Adv}^{\text{rsig-uf-ic}}_{\mathcal{LRS},\mathcal{A}} \le \mathbf{Adv}^{\text{cdh}}_{G_p,\mathcal{B}}\ . \tag{11}$$

Type-III Adversary. We convert a Type-III forger $\mathcal{A}$ to a Waters signature forger in the subgroup G_p.

Algorithm $\mathcal{B}$ is given: the group order n and its factorization $n=pq$; the description of the group G, together with generators η of G_p and h of G_q; and, in addition, WP public parameters $\hat{u}',\hat{u}_1,\ldots,\hat{u}_k$ all in G_p, and a WP public key $(\eta_1,\eta_2)\in G_p^2$. Algorithm $\mathcal{B}$ chooses a collision resistant hash function $H:\{0,1\}^*\to\{0,1\}^k$ as above. It picks $r_1\xleftarrow{\text{R}}\mathbb{Z}_q^*$ and sets $g\leftarrow\eta h^{r_1}$. Now $\mathcal{B}$ picks $r_2,r_3\xleftarrow{\text{R}}\mathbb{Z}_q$ and $x\xleftarrow{\text{R}}\mathbb{Z}_p$ sets

$$A\leftarrow\eta_1 h^{r_2}\qquad\text{and}\qquad\hat{A}\leftarrow h^{r_2/r_1}\qquad\text{and}\qquad B_0\leftarrow\eta^x h^{r_3}\ .$$

Note that in this case B_0 is unrelated to the challenge WP key. Algorithm $\mathcal{B}$ picks random exponents $s',s_1,s_2,\ldots,s_k\xleftarrow{\text{R}}\mathbb{Z}_q$ and sets $u'\leftarrow\hat{u}'h^{s'}$ and, for $1\le j\le k$, $u_j\leftarrow\hat{u}h^{s_j}$.

Finally, $\mathcal{B}$ generates the user keys. It picks $i^*\xleftarrow{\text{R}}\{1,2,\ldots,l\}$. For each $i\neq i^*$, it chooses a random exponent $b_i\xleftarrow{\text{R}}\mathbb{Z}_n$ and sets $pk_i\leftarrow g^{b_i}$ and $sk_i\leftarrow A^{b_i}$. For user i^*, algorithm $\mathcal{B}$ picks $r_4\xleftarrow{\text{R}}\mathbb{Z}_q$ and sets $pk^*=pk_{i^*}\leftarrow\eta_2 h^{r_4}$. Algorithm $\mathcal{B}$ does not know b_{i^*} and sk_{i^*}, and sets these to placeholder values.

It is easy to see, using the observations made at the beginning of the proof, that if the WP challenge given to $\mathcal{B}$ is proper and uniformly distributed then the ring forging challenge given to $\mathcal{A}$ is also proper and uniformly distributed.

Now algorithm $\mathcal{B}$ runs $\mathcal{A}$, providing to it the following: the description of the group G, including the generators g and h; the common parameters $(A,B_0,\hat{A})$ and $(u',u_1,\ldots,u_k)$, along with the description of H; and the challenge public keys $\{pk_i\}_{i=1}^l$.

When $\mathcal{A}$ makes a corruption query at index s, $1\le s\le l$, $\mathcal{B}$ responds with sk_s unless s equals i^*, in which case $\mathcal{B}$ declares failure and exits.

When $\mathcal{A}$ makes a ring signing query of the form (s,R,M), $\mathcal{A}$ responds as follows. It first checks that the query is properly formed, and declines to answer otherwise. Now if $s\neq i^*$, $\mathcal{B}$ knows sk_s, so it computes $\sigma\leftarrow\mathsf{Sig}(pk_s,sk_s,R,M)$, and responds to $\mathcal{A}$ with this value.

[4] Note that raising the pairings to power δ_p suffices: forcing one argument to G_p makes the G_q component of the other argument irrelevant, since $e(u,v)=1$ for all $u\in G_p$ and $v\in G_q$.

If $s = i^*$, $\mathcal{B}$ computes σ by querying its own WP signing oracle for an ordinary signature in G_p and computing based on this a blind signature in G, as follows. It first requests from its signing oracle a WP signature on $(m_1, \ldots, m_k) = H(M, R)$. The signing oracle responds with $(\hat{S}_1, \hat{S}_2) \in G_p^2$. Let $l' = |R|$, and parse R as $(v_1, \ldots, v_{l'}$. Now, for $1 \le i \le l'$, algorithm $\mathcal{B}$ sets $f_i \leftarrow 0$ if $v_i \neq pk^*$, and $f_i \leftarrow 1$ for the single i such that $v_i = pk^*$. (We are guaranteed that $\{v_i\}$ will behave this way by the well-formedness of the query.) For each i again, $\mathcal{B}$ picks a random exponent $t_i \xleftarrow{\text{R}} \mathbb{Z}_n$ and sets

$$C_i \leftarrow (v_i/B_0)^{f_i} h^{t_i} \qquad \text{and} \qquad \pi_i \leftarrow \big((v_i/B_0)^{2f_i - 1} h^{t_i}\big)^{t_i} \ .$$

These are formed exactly as in the description of Sig. Algorithm $\mathcal{B}$ sets $t \leftarrow \sum_{i=1}^{l} t_i$. Then we have

$$C = \prod_{i=1}^{l} C_i = (pk^*/B_0) h^t \ . \tag{12}$$

Now $\mathcal{B}$ chooses $r \xleftarrow{\text{R}} \mathbb{Z}_q$ and computes

$$S_1 \leftarrow \hat{S}_1 \cdot h^{r_2 r_4/r_1} \cdot \big(h^{s'/r_1} \prod_{j=1}^{k} h^{s_j m_j/r_1}\big)^r \cdot h^{r_2 t/r_1} \qquad \text{and} \qquad S_2 \leftarrow \hat{S}_2 \cdot h^r \ .$$

Given this, we have

$$\begin{aligned} e(S_1, g) &= e(\hat{S}_1, g) \cdot e(h^{r_2 r_4/r_1}, g) \cdot e(h^{s'/r_1} \prod_{j=1}^{k} h^{s_j m_j/r_1},\, g)^r \cdot e(h^{r_2 t/r_1}, g) \\ &= e(\hat{S}_1, \eta) \cdot e(h^{r_2 r_4/r_1}, h^{r_1}) \cdot e(h^{s'/r_1} \prod_{j=1}^{k} h^{s_j m_j/r_1},\, h^{r_1})^r \cdot e(h^{r_2 t/r_1}, h^{r_1}) \\ &= e(\hat{S}_1, \eta) \cdot e(h^{r_2}, h^{r_4}) \cdot e(h^{s'} \prod_{j=1}^{k} h^{s_j m_j},\, h)^r \cdot e(h^{r_2 t}, h) \end{aligned}$$

where we have used the decomposition $g = \eta h^{r_1}$ and the fact that $e(\eta, h) = 1$. Now, since $(\hat{S}_1, \hat{S}_2)$ satisfies the WP verification equation (2), we have

$$\begin{aligned} e(S_1, g) &= \cdots \\ &= e(\eta_1, \eta_2) \cdot e\Big(\hat{S}_2^{-1}, \hat{u}' \prod_{j=1}^{k} \hat{u}_j^{m_j}\Big)^{-1} \cdot e(h^{r_2}, h^{r_4}) \cdot e(h^{s'} \prod_{j=1}^{k} h^{s_j m_j},\, h)^r \cdot e(h^{r_2 t}, h) \\ &= e(A, pk^*) \cdot e\Big(\hat{S}_2, \hat{u}' \prod_{j=1}^{k} \hat{u}_j^{m_j}\Big) \cdot e\big(h^r,\, h^{s'} \prod_{j=1}^{k} h^{s_j m_j}\big) \cdot e(h^{r_2 t}, h) \ . \end{aligned}$$

Next, we use the fourth observation made at the beginning of the proof to combine the pairings dealing with message bits m_j:

$$\begin{aligned} e(S_1, g) &= \cdots \\ &= e(A, pk^*) \cdot e\Big(\hat{S}_2 h^r, (\hat{u}' h^{s'}) \cdot \prod_{j=1}^{k} (\hat{u}_j h^{s_j})^{m_j}\Big) \cdot e(h^{r_2 t}, h) \\ &= e(A, pk^*) \cdot e\big(S_2, u' \prod_{j=1}^{k} u_j^{m_j}\big) \cdot e(h^{r_2 t}, h) \end{aligned}$$

Finally, we substitute (12) for pk^*, obtaining

$$\begin{aligned} e(S_1, g) &= \cdots \\ &= e(A,\, CB_0 h^{-t}) \cdot e\big(S_2, u' \prod\nolimits_{j=1}^{k} u_j^{m_j}\big) \cdot e(h^{r_2 t}, h) \\ &= e(A, CB_0) \cdot e(h^{r_2}, h^{-t}) \cdot e\big(S_2, u' \prod\nolimits_{j=1}^{k} u_j^{m_j}\big) \cdot e(h^{r_2 t}, h) \\ &= e(A, CB_0) \cdot e\big(S_2, u' \prod\nolimits_{j=1}^{k} u_j^{m_j}\big) \enspace . \end{aligned}$$

But this is just the ring signature verification equation (4). Thus the signature as constructed is valid. We have already shown that the components $\{(C_i, \pi_i)\}$ are distributed exactly as in Sig. We need only show that (S_1, S_2) is properly distributed. Observe that $\hat{S}_2$ is properly distributed with some randomness $\hat{r} \in \mathbb{Z}_p$. This, combined with the randomness $r \in \mathbb{Z}_q$ we chose means that S_2 is properly distributed in G. Given a choice for S_2 and $\{(C_i, \pi_i)\}$, there is only one choice of S_1 that satisfies the verification equation. Thus the signature is properly formatted. Algorithm $\mathcal{B}$ computes and responds to $\mathcal{A}$ with $\sigma = \big((S_1, S_2),\, \{(C_i, \pi_i)\}_{i=1}^{l}\big) \in G^{2l+2}$.

Finally, $\mathcal{A}$ outputs a forgery σ^* on a ring R^* for message M^*. Let $l^* = |R^*|$. Parse the ring R^* as $(v_1, \ldots, v_{l^*})$. As before, since the forgery is nontrivial, none of the public keys is repeated, and there is a mapping $t : [1, l^*] \to [1, l]$ such that $v_i = pk_{t(i)}$ for each i, $1 \le i \le l^*$. Moreover, $\mathcal{A}$ must not have made a corruption query at any of the indices $\{t(i)\}_{i=1}^{l^*}$.

Algorithm $\mathcal{B}$ parses σ^* as $\big((S_1, S_2),\, \{(C_i, \pi_i)\}_{i=1}^{l^*}\big) \in G^{2l^*+2}$. Again, since the forgery is nontrivial, this parse must succeed. As before, we recover $\{f_i\}_{i=1}^{l^*}$ satisfying $C_i^{\delta_p} = \big(v_i^{\delta_p} / B_0^{\delta_p}\big)^{f_i}$. What is more, since $\mathcal{A}$ is Type III, we know that there is exactly one index $s \in \{1, 2, \ldots, l^*\}$ such that $f_s = 1$. If $t(s) \neq i^*$, $\mathcal{B}$ declares failure and exits. Otherwise we have

$$C^{\delta_p} = \prod_{i=1}^{l^*} C_i^{\delta_p} = (pk^*)^{\delta_p} / B_0^{\delta_p} \enspace . \tag{13}$$

Finally, $\mathcal{B}$ computes $(m_1, \ldots, m_k) \leftarrow H(M^*, R^*)$. Since the forgery is valid, the verification equation (4) must hold; raising both sides to power δ_p and then substituting for C^{δ_p} with (13), we obtain

$$e(S_1^{\delta_p}, \eta) \cdot e\big((S_2^{\delta_p})^{-1},\, \hat{u}' \prod\nolimits_{j=1}^{k} \hat{u}_j^{m_j}\big) = e(\eta_1, B_0^{\delta_p} C^{\delta_p}) = e\big(\eta_1, (pk^*)^{\delta_p}\big) = e(\eta_1, \eta_2) \enspace ,$$

So $(S_1^{\delta_p}, S_2^{\delta_p})$ is a valid WP signature on $H(M^*, R^*)$. Since $\mathcal{A}$ is Type III, we know that $H(M^*, R^*)$ is different from $H(M, R)$ for any (M, R) for which $\mathcal{A}$ made a ring signing query, and therefore different from any $H(M, R)$ for which $\mathcal{B}$ made a WP signing query. The forgery is thus nontrivial, and $\mathcal{A}$ outputs it and halts.

All that remains is to bound the probability that $\mathcal{B}$ aborts in the simulation above. But this is a standard argument. The special key is hidden at index i^*, which is uniformly chosen from the set $\{1, 2, \ldots, l\}$. Algorithm $\mathcal{B}$ can answer all signing queries correctly, so if it has not aborted after $\mathcal{A}$'s first c corruption queries, the value of i^* is independent of $\mathcal{A}$'s view, and the probability that $\mathcal{A}$ chooses it is $1/(l - c)$. Algorithm $\mathcal{A}$ may make at most $l - 1$ corruption queries, and its forgery must come from the set of uncorrupted keys where, again, i^* is independent of its view if $\mathcal{B}$ did not abort in the corruption phase. A simple calculation shows that regardless of $\mathcal{A}$'s behavior, $\mathcal{B}$

does not abort in either phase with probability $1/l$. Whenever algorithm $\mathcal{B}$ does not abort and $\mathcal{A}$ succeeds in forging a ring signature, $\mathcal{B}$ in turn succeeds in creating a WP forgery, so we have established that — for a Type-III adversary — we have

$$\mathbf{Adv}^{\text{rsig-uf-ic}}_{\mathcal{LRS},\mathcal{A}} \leq l \cdot \mathbf{Adv}^{\text{sig-uf-cma}}_{\mathcal{WP},\mathcal{B}} \ . \tag{14}$$

Summary. Putting together equations (8), (11), and (14), we obtain the following reduction:

$$\mathbf{Adv}^{\text{rsig-uf-ic}}_{\mathcal{LRS},\mathcal{A}} \leq \mathbf{Adv}^{\text{coll}}_{H,\mathcal{B}_1} + \mathbf{Adv}^{\text{cdh}}_{G_p,\mathcal{B}_2} + l \cdot \mathbf{Adv}^{\text{sig-uf-cma}}_{\mathcal{WP},\mathcal{B}_3} \ . \tag{15}$$

To interpret (15) concretely, we note that the reduction for each type of adversary took $O(1)$ operations to set up and obtain the result, and $O(1)$ time to answer each of $\mathcal{A}$'s queries. We can, moreover, guess which type of adversary we are dealing with, sustaining a $1/3$ loss in advantage. To interpret (15) asymptotically, we introduce the security parameter that we have suppressed, note that the reductions are all polynomial-time in that parameter, and observe that if $\mathbf{Adv}^{\text{rsig-uf-ic}}_{\mathcal{LRS},\mathcal{A}}$ is non-negligible, then one of the advantages on the right hand side must also be non-negligible. Either interpretation implies the result stated informally in the theorem, since WP is unforgeable if CDH is hard in G_p. $\square$