Ring Signatures: Stronger Definitions, and Constructions without Random Oracles[*]

Adam Bender[†] Jonathan Katz[†‡] Ruggero Morselli[†§]

Abstract

Ring signatures, first introduced by Rivest, Shamir, and Tauman, enable a user to sign a message so that a *ring* of possible signers (of which the user is a member) is identified, without revealing exactly *which member* of that ring actually generated the signature. In contrast to group signatures, ring signatures are completely "ad-hoc" and do not require any central authority or coordination among the various users (indeed, users do not even need to be aware of each other); furthermore, ring signature schemes grant users fine-grained control over the level of anonymity associated with any particular signature.

This paper has two main areas of focus. First, we examine previous definitions of security for ring signature schemes and suggest that most of these prior definitions are too weak, in the sense that they do not take into account certain realistic attacks. We propose new definitions of anonymity and unforgeability which address these threats, and give separation results proving that our new notions are strictly stronger than previous ones. Second, we show the first constructions of ring signature schemes in the standard model. One scheme is based on generic assumptions and satisfies our strongest definitions of security. Two additional schemes are more efficient, but achieve weaker security guarantees and more limited functionality.

1 Introduction

Ring signatures enable a user to sign a message so that a "ring" of possible signers (of which the user is a member) is identified, without revealing exactly which member of that ring actually generated the signature. This notion was first formally introduced by Rivest, Shamir, and Tauman [22], and ring signatures — along with the related notion of ring/ad-hoc identification schemes — have been studied extensively since then [6, 21, 1, 25, 5, 18, 13, 24, 20, 2]. Ring signatures are related, but incomparable, to the notion of group signatures [8]. On the one hand, group signatures have the additional feature that the anonymity of a signer can be revoked (i.e., the signer can be traced) by a designated group manager. On the other hand, ring signatures allow greater flexibility: no centralized group manager or coordination among the various users is required (indeed, users may be unaware of each other at the time they generate their public keys), rings may be formed completely "on-the-fly" and in an ad-hoc manner, and users are given fine-grained control over the level of anonymity associated with any particular signature (via selection of an appropriate ring).

[*]An extended abstract of this paper appeared in the 3rd Theory of Cryptography Conference, March 4-7 2006, New York, NY, USA [4].

[†]Dept. of Computer Science, University of Maryland. {bender,jkatz,ruggero}@cs.umd.edu

[‡]Supported in part by NSF Trusted Computing Grants #0310499 and #0310751, NSF-ITR #0426683, and NSF CAREER award #0447075.

[§]Supported by NSF Trusted Computing Grant #0310499 and NSF-ITR #0426683.

Ring signatures naturally lend themselves to a variety of applications which have been suggested already in previous work (see especially [22, 21, 13, 2]). The original motivation was to allow secrets to be leaked anonymously. Here, for example, a high-ranking government official can sign information with respect to the ring of *all* similarly high-ranking officials; the information can then be verified as coming from *someone* reputable without exposing the actual signer. Ring signatures can also be used to provide a member of a certain class of users access to a particular resource without explicitly identifying this member; note that there may be cases when third-party verifiability is required (e.g., to prove that the resource has been accessed) and so ring signatures, rather than ad-hoc identification schemes, are needed. Finally, we mention the application to designated-verifier signatures [19] especially in the context of e-mail. Here, ring signatures enable the sender of an e-mail to sign the message with respect to the ring containing the sender and the receiver; the receiver is then assured that the e-mail originated from the sender but cannot prove this to any third party. For this latter application it is sufficient to use a ring signature scheme which supports only rings of size two. Chen et al. [9] propose another application of ring signatures where rings of size two suffice.

1.1 Our Contributions in Relation to Previous Work

This paper focuses on both definitions and constructions. We summarize our results in each of these areas, and relate them to prior work.

Definitions of security. Prior work on ring signature/identification schemes provides definitions of security that are either rather informal or seem (to us) unnaturally weak, in that they do not address a number of seemingly valid security concerns. One example is the failure to consider the possibility of *adversarially-chosen* public keys. Specifically, both the anonymity and unforgeability definitions in most prior work assume that honest users always sign with respect to rings consisting entirely of *honestly-generated* public keys; no security is provided if users sign with respect to a ring containing even one adversarially-generated public key. Clearly, however, a scheme which is not secure in the latter case is of limited use; this is especially true since rings are constructed in an ad-hoc fashion using keys of (possibly unknown) users which are not validated by any central authority as being correctly constructed. We formalize security against such attacks (as well as others), and show separation results proving that our definitions are strictly stronger than those considered in previous work. In addition to the new, strong definitions we present, the *hierarchy* of definitions we give is useful for characterizing the security of ring signature constructions.

Constructions. We present three ring signature schemes which are provably secure in the standard model. We stress that these are the *first* such constructions, as all previous constructions of which we are aware rely on random oracles/ideal ciphers.[1] It is worth remarking that ring identification schemes are somewhat easier to construct (using, e.g., techniques of Cramer et al. [11]); ring signatures can then easily be derived from such schemes using the Fiat-Shamir methodology in the random oracle model [16]. This approach, however, is no longer viable (at least, based on our current understanding) when working in the standard model.

Our first construction is based on generic assumptions, and satisfies the strongest definitions of anonymity and unforgeability considered here. Moreover, this construction is completely ad-hoc and requires no setup assumptions of any kind. This construction is inspired by the generic

[1]Although Xu, Zhang, and Feng [24] claim a ring signature scheme in the standard model based on specific assumptions, their proof was later found to be flawed (personal communication from J. Xu, March 2005). Concurrently to our work, Chow, Liu and Yuen [10] show a ring signature scheme that they prove secure in the standard model (for rings of *constant* size) based on a new number-theoretic assumption.

construction of group signatures due to Bellare et al. [3] and, indeed, the constructions share some similarities at a high level. However, a number of subtleties arise in our context that do not arise in the context of group signatures, and the construction given in [3] does not immediately lend itself to a ring signature scheme. Two issues in particular that we need to deal with are the fact that we have no central group manager to issue "certificates" as in [3], and that we additionally need to take into account the possibility of adversarially-generated public keys as discussed earlier (this is not a concern in [3] where there is only a single group public key published by a (semi-)trusted group manager).

Our other two constructions are more efficient than the first, and rely on specific number-theoretic assumptions. These have various disadvantages with regard to the generic solution discussed previously. First, these schemes require some system-wide public parameters (essentially a description of a cyclic group along with a generator) that must be shared by all users of the scheme. Importantly, however, we stress that there is no possibility of embedding any "trapdoor" in these parameters; thus, reliance on such parameters is much less problematic than relying on a "common reference string" that typically does have some trapdoor information associated with it, and must therefore be generated by a trusted party (who can violate security of the scheme if they are corrupted). Our final two schemes also provide more limited functionality and security guarantees than our first construction; most limiting is that they only support rings of size two. Such schemes are still useful for certain applications (as discussed earlier); furthermore, constructing an efficient 2-user ring signature scheme without random oracles is still difficult, as we do not have the Fiat-Shamir methodology available in our toolbox. These two schemes are based, respectively, on the recent (standard) signature schemes of Waters [23] and Camenisch and Lysyanskaya [7].

2 Preliminaries

We use the standard definitions of public-key encryption schemes and semantic security; signature schemes and existential unforgeability under adaptive chosen-message attacks; and computational indistinguishability. In this paper we will assume public-key encryption schemes for which, with all but negligible probability over (pk, sk) generated at random using the specified key generation algorithm, $\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(M)) = M$ holds with probability 1.

We will also use the notion of a *ZAP*, which is a 2-round, public-coin, witness-indistinguishable proof system for any language $L \in \mathcal{NP}$. ZAPs were introduced by Dwork and Naor [14], who show that ZAPs can be constructed based on any non-interactive zero-knowledge (NIZK) proof system; the latter, in turn, can be constructed based on trapdoor permutations [15].

Formally, a ZAP is a triple $(\ell, \mathcal{P}, \mathcal{V})$ such that (1) the initial message r from the verifier $\mathcal{V}$ has length $\ell(k)$ (where k is the security parameter); (2) the prover $\mathcal{P}$, on input the verifier-message r, statement x, and witness w, outputs $\pi \leftarrow \mathcal{P}_r(x, w)$; finally, (3) $\mathcal{V}_r(x, \pi)$ outputs 1 or 0, indicating acceptance or rejection of the proof.

A ZAP is used in the following way: The verifier generates a random first message $r \leftarrow \{0,1\}^{\ell(k)}$ and sends it to the prover $\mathcal{P}$. The prover, given r, a statement x, and associated witness w, sends $\pi \leftarrow \mathcal{P}_r(x, w)$ to the verifier. The verifier then runs $\mathcal{V}_r(x, \pi)$ and accepts iff the output is 1.

A formal definition of the security properties guaranteed by a ZAP follows. Let L be an $\mathcal{NP}$ language with associated polynomial-time and polynomially-bounded *witness relation* $\mathcal{R}_L$ (i.e., such that $L \stackrel{\text{def}}{=} \{x \mid \exists w : (x, w) \in \mathcal{R}_L\}$). If $(x, w) \in \mathcal{R}_L$ we refer to x as the *statement* and w as the associated *witness* for x.

Definition 1 [**ZAP**] A ZAP for an $\mathcal{NP}$ language L (with associated witness relation $\mathcal{R}_L$) is a triple

$(\ell, \mathcal{P}, \mathcal{V})$, where $\ell(\cdot)$ is a polynomial, $\mathcal{P}$ is a PPT algorithm, and $\mathcal{V}$ is polynomial-time deterministic algorithm such that.

Completeness For any $(x, w) \in \mathcal{R}_L$ and any $r \in \{0,1\}^{\ell(k)}$:

$$\mathbf{Pr}\left[\pi \leftarrow \mathcal{P}_r(x, w) : \mathcal{V}_r(x, \pi) = 1\right] = 1 \ .$$

(The original definition [14] allows for a negligible completeness error, but their construction achieves perfect completeness when instantiated using the NIZK of [15].)

Adaptive soundness There exists a negligible function ε such that

$$\mathbf{Pr}\left[r \leftarrow \{0,1\}^{\ell(k)} : \exists(x, \pi) : \quad x \notin L \text{ and } \mathcal{V}_r(x, \pi) = 1\right] \leq \varepsilon(k) \, .$$

Witness indistinguishability (Informal) For any $x \in L$, any pair of witnesses w_0, w_1 for x, and any $r \in \{0,1\}^{\ell(k)}$, the distributions $\{\mathcal{P}_r(x, w_0)\}$ and $\{\mathcal{P}_r(x, w_1)\}$ are computationally indistinguishable. (Formally, we need to speak in terms of sequences $\{r_k \in \{0,1\}^{\ell(k)}\}$, $\{x_k\}$, and $\{(w_{k,0}, w_{k,1})\}$ but we avoid doing so for simplicity of exposition.) $\diamondsuit$

3 Definitions

We begin by presenting the functional definition of a ring signature scheme. We refer to an ordered list $R = (PK_1, \ldots, PK_n)$ of public keys as a *ring*, and let $R[i] = PK_i$. We will also freely use set notation, and say, e.g., that $PK \in R$ if there exists an index i such that $R[i] = PK$. We will always assume, without loss of generality, that the keys in a ring are ordered lexicographically.

Definition 2 [**Ring signature**] A *ring signature scheme* is a triple of PPT algorithms (Gen, Sign, Vrfy) that, respectively, generate keys for a user, sign a message, and verify the signature of a message. Formally:

- $\mathsf{Gen}(1^k)$, where k is a security parameter, outputs a public key PK and secret key SK.
- $\mathsf{Sign}_{s,SK}(M, R)$ outputs a signature σ on the message M with respect to the ring $R = (PK_1, \ldots, PK_n)$. We assume the following: (1) ($R[s]$, SK) is a valid key-pair output by Gen; (2) $|R| \geq 2$ (since a ring signature scheme is not intended to serve as a standard signature scheme); and (3) each public key in the ring is distinct.

  The first of the above conditions simply models ring signature usage (where a signer "knows" their index s in the ring). The latter two conditions are without much loss of generality: it is easy to modify any ring signature scheme to allow signatures with $|R| = 1$ by including a special key for just that purpose, and given a ring R with repeated keys the signer/verifier can simply take the sub-ring of distinct keys in R and correctness (see below) will be unaffected.
- $\mathsf{Vrfy}_R(M, \sigma)$ outputs a single bit indicating validity or invalidity of a purported signature σ on a message M with respect to the ring of public keys R.

We require the following correctness condition: for any k, any $\{(PK_i, SK_i)\}_{i=1}^n$ output by $\mathsf{Gen}(1^k)$, any $s \in [n]$, and any M, we have $\mathsf{Vrfy}_R(M, \mathsf{Sign}_{s,SK_s}(M, R)) = 1$ where $R = (PK_1, \ldots, PK_n)$.

A *c-user ring signature scheme* is a variant of the above that only supports rings of fixed size c (i.e., the Sign and Vrfy algorithms only take as input rings R for which $|R| = c$, and correctness is only required to hold for such rings). $\diamondsuit$

To improve readability, we will generally omit the input "s" to the signing algorithm (and simply write $\sigma \leftarrow \mathsf{Sign}_{SK}(M, R)$), with the understanding that the signer can determine an index s for which SK is the secret key corresponding to public key $R[s]$. Strictly speaking, there may not be a unique such s (especially when R contains incorrectly-generated keys); in real-world usage of a ring signature scheme, though, a signer will certainly be able to identify his own public key.

A ring signature scheme is used as follows: At various times, different users run the key-generation algorithm Gen to generate public and secret keys. We stress that no coordination among these users is assumed or required. When a user with secret key SK wishes to generate an anonymous signature on a message M, he chooses a ring R of public keys which includes his own, computes $\sigma \leftarrow \mathsf{Sign}_{SK}(M, R)$ and outputs (σ, R); in such a case, we refer to the holder of SK as the *signer* of the message. Assuming the scheme is secure, anyone can now verify that this signature was generated by *someone* holding a secret key corresponding to a public key in R by running $\mathsf{Vrfy}_R(M, \sigma)$.

Although our functional definition of a ring signature scheme requires users to generate keys specifically for that purpose (in contrast to the requirements of [1, 2]), our first construction can be easily modified to work with any ring of users as long as they each have a public key for both encryption and signing (see Section 5).

As discussed in the Introduction, ring signatures must satisfy two independent notions of security: anonymity and unforgeability. There are various ways each of these notions can be defined (and various ways these notions have been defined in the literature); we present our definitions in Sections 3.1 and 3.2, and compare them to previously-suggested definitions in Section 4.

3.1 Definitions of Anonymity

The anonymity condition requires, informally, that an adversary should not be able to tell which member of a ring generated a particular signature. (All the anonymity definitions that follow can be phrased in either a *computational* or an *unconditional* sense where, informally, anonymity holds for polynomial-time adversaries in the former case and all-powerful adversaries in the latter case. For simplicity, we present only the computational versions.) We begin with a basic definition of anonymity which is already stronger than that considered in most previous work in that we give the adversary access to a signing oracle (this results in a stronger definition even in the case of unconditional anonymity).

Definition 3 [**Basic anonymity**] Given a ring signature scheme (Gen, Sign, Vrfy), a polynomial $n(\cdot)$, and a PPT adversary $\mathcal{A}$, consider the following game:

1. Key pairs $\{(PK_i, SK_i)\}_{i=1}^{n(k)}$ are generated using $\mathsf{Gen}(1^k)$, and the set of public keys $S \overset{\text{def}}{=} \{PK_i\}_{i=1}^{n(k)}$ is given to $\mathcal{A}$.

2. $\mathcal{A}$ is given access to an oracle $\mathsf{OSign}(\cdot, \cdot, \cdot)$ such that $\mathsf{OSign}(s, M, R)$ returns $\mathsf{Sign}_{SK_s}(M, R)$, where we require $R \subseteq S$ and $PK_s \in R$.

3. $\mathcal{A}$ outputs a message M, distinct indices i_0, i_1, and a ring $R \subseteq S$ for which $PK_{i_0}, PK_{i_1} \in R$. A random bit b is chosen, and $\mathcal{A}$ is given the signature $\sigma \leftarrow \mathsf{Sign}_{SK_{i_b}}(M, R)$.

4. The adversary outputs a bit b', and succeeds if $b' = b$.

$(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ achieves *basic anonymity* if, for any PPT $\mathcal{A}$ and any polynomial $n(\cdot)$, the success probability of $\mathcal{A}$ in the above game is negligibly close to 1/2. $\diamondsuit$

(A variant is for the adversary to be given a signature computed by a randomly-chosen member of R, with the requirement that the adversary should be unable to guess the actual signer with probability better than $1/|R| + \mathsf{negl}(k)$. It is easy to see that this is equivalent to the above.)

Unfortunately, the above definition of basic anonymity leaves open the possibility of the following attack: (1) an adversary generates public keys in some arbitrary manner (which may possibly depend on the public keys of the honest users), and then (2) a legitimate signer generates a signature with respect to a ring containing some of these adversarially-generated public keys. The definition above offers no protection in this case! This attack, considered also in [21] (in a slightly different context) is quite realistic since, by their very nature, ring signatures are intended to be used in settings where there is no central authority checking validity of public keys. This motivates the following, stronger definition:

Definition 4 [**Anonymity w.r.t. adversarially-chosen keys**] Given a ring signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$, a polynomial $n(\cdot)$, and a PPT adversary $\mathcal{A}$, consider the following game:

1. As in Definition 3, key pairs $\{(PK_i, SK_i)\}_{i=1}^{n(k)}$ are generated using $\mathsf{Gen}(1^k)$ and the set of public keys $S \overset{\text{def}}{=} \{PK_i\}_{i=1}^{n(k)}$ is given to $\mathcal{A}$.
2. $\mathcal{A}$ is given access to an oracle $\mathsf{OSign}(\cdot,\cdot,\cdot)$ such that $\mathsf{OSign}(s, M, R)$ returns $\mathsf{Sign}_{SK_s}(M, R)$, where we require $PK_s \in R$. (In contrast to Definition 3, we no longer require $R \subseteq S$.)
3. $\mathcal{A}$ outputs a message M, distinct indices i_0, i_1, and a ring R for which $PK_{i_0}, PK_{i_1} \in R$. (Again, we no longer require $R \subseteq S$.) A random bit b is chosen, and $\mathcal{A}$ is given the signature $\sigma \leftarrow \mathsf{Sign}_{SK_{i_b}}(M, R)$.
4. The adversary outputs a bit b', and succeeds if $b' = b$.

$(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ achieves *anonymity w.r.t. adversarially-chosen keys* if for any PPT $\mathcal{A}$ and polynomial $n(\cdot)$, the success probability of $\mathcal{A}$ in the above game is negligibly close to 1/2. $\diamondsuit$

The above definition only guarantees anonymity of a particular signature as long as there are at least two honest users in the ring. In some sense this is inherent, since if an honest signer U chooses a ring in which *all other* public keys (i.e., except for the public key of U) were created by an adversary, then that adversary "knows" that U must be the signer (since the adversary did not generate the signature itself).

A weaker requirement one might consider when the signer U is the only honest user in a ring is that the other members of the ring should be unable to *prove* to a third party that U generated the signature (we call this an *attribution attack*). Preventing such an attack in general seems to require the involvement of a trusted party (or at least a common random string), something we would like to avoid. We instead define a slightly weaker notion which, informally, can be viewed as offering an honest user U some protection against attribution attacks as long as at least one other user U' in the ring was honest *at the time U' generated his public key*. However, we allow this user U' — as well as all other honest users in the ring (except for U) — to later collude with an adversary by revealing their secret keys in an attempt to attribute the signature to U.[2] (Actually, we even allow these users to reveal the *randomness* used to generate their secret keys. This ensures

[2]The idea is that everyone else in the ring is trying to "frame" U, but U is (naturally) refusing to divulge her secret key. Although this itself might arouse suspicion, the point is that it still cannot be proved — in court, say — that U was the signer.

security when erasure cannot be guaranteed, or when it cannot be guaranteed that all users will comply with the directive to erase their random coins.) Security in such a setting also ensures some measure of security in case honest users' secret keys are exposed or stolen.

In addition to the above, we consider also the stronger variant in which the secret keys of *all* honest users in the ring (i.e., including U) are exposed. This parallels the anonymity definition given by Bellare, et al. in the context of group signatures [3] (in fact, our definition is even stronger since we give the adversary the random coins of the corrupted users and not just their secret keys). For simplicity we also protect against adversarially-chosen keys, though one could consider the weaker definition that does not.

Definition 5 [**Anonymity against attribution attacks/full key exposure**] Given a ring signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$, a polynomial $n(\cdot)$, and a PPT adversary $\mathcal{A}$, consider the following game:

1. For $i = 1$ to $n(k)$, generate $(PK_i, SK_i) \leftarrow \mathsf{Gen}(1^k; \omega_i)$ for randomly-chosen ω_i. Give to $\mathcal{A}$ the set of public keys $\{PK_i\}_{i=1}^{n(k)}$. (This is functionally identical to the first step in previous definitions, except that we now make explicit the random coins used to generate keys.)

2. As in Definition 4, $\mathcal{A}$ is given access to an oracle $\mathsf{OSign}(\cdot,\cdot,\cdot)$ such that $\mathsf{OSign}(s, M, R)$ returns $\mathsf{Sign}_{SK_s}(M, R)$, where we require $PK_s \in R$.

3. $\mathcal{A}$ is also given access to an oracle $\mathsf{Corrupt}(\cdot)$ that, on input i, returns ω_i.

4. As in Definition 4, $\mathcal{A}$ outputs a message M, distinct indices i_0, i_1, and a ring R for which $PK_{i_0}, PK_{i_1} \in R$. A random bit b is chosen and $\mathcal{A}$ is given $\sigma \leftarrow \mathsf{Sign}_{SK_{i_b}}(M, R)$.

5. The adversary outputs a bit b', and succeeds if $b' = b$ and $|\{i_0, i_1\} \cap C| \leq 1$, where C is the set of queries to the corruption oracle.

$(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ achieves *anonymity against attribution attacks* if, for any PPT $\mathcal{A}$ and polynomial $n(\cdot)$, the success probability of $\mathcal{A}$ in the above game is at most $1/2 + \mathsf{negl}(k)$. If we allow $|\{i_0, i_1\} \cap C| = 2$, then we say $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ achieves *anonymity against full key exposure.* $\diamondsuit$

Linkability. Another desideratum of a ring signature scheme is that it be *unlinkable*; that is, it should be infeasible to determine whether two signatures (possibly generated with respect to different rings) were generated by the same signer. A detailed discussion on the relation between anonymity and unlinkability appears in [3, Section 3], and the conclusion reached there is that anonymity and unlinkability are equivalent under any "reasonable" formulation of these two notions. We concur with that assessment in our setting, and in particular remark that even our most basic definition of anonymity implies that signatures by the same signer cannot be meaningfully correlated. On the other hand, it is not hard to see that a weaker definition of anonymity whereby the adversary obtains only users' public keys and a single signature — but *cannot* obtain multiple other signatures via a signing oracle — does not imply unlinkability.

3.2 Definitions of Unforgeability

The intuitive notion of unforgeability is, as usual, that an adversary should be unable to output (R, M, σ) such that $\mathsf{Vrfy}_R(M, \sigma) = 1$ unless either (1) one of the public keys in R was generated by the adversary, or (2) a user whose public key is in R explicitly signed M previously (with respect to the same ring R). Some subtleties arise, however, when defining a chosen-message attack on the scheme. Many previous works (e.g., [22]), assume a definition like the following:

Definition 6 [**Unforgeability against fixed-ring attacks**] A ring signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *unforgeable against fixed-ring attacks* if for any PPT adversary $\mathcal{A}$ and for any polynomial $n(\cdot)$, the probability that $\mathcal{A}$ succeeds in the following game is negligible:

1. Key pairs $\{(PK_i, SK_i)\}_{i=1}^{n(k)}$ are generated using $\mathsf{Gen}(1^k)$, and the set of public keys $R \overset{\text{def}}{=} \{PK_i\}_{i=1}^{n(k)}$ is given to $\mathcal{A}$.
2. $\mathcal{A}$ is given access to a *signing oracle* $\mathsf{OSign}(\cdot, \cdot)$, where $\mathsf{OSign}(s, M)$ outputs $\mathsf{Sign}_{SK_s}(M, R)$.
3. $\mathcal{A}$ outputs (M^*, σ^*) and succeeds if $\mathsf{Vrfy}_R(M^*, \sigma^*) = 1$ and also $\mathcal{A}$ never made a query of the form $\mathsf{OSign}(\star, M^*)$. $\diamondsuit$

Note that not only is $\mathcal{A}$ restricted to making signing queries with respect to the *full* ring R, but its forgery is required to verify with respect to R as well. The following stronger, and more natural, definition was used in, e.g., [1]:

Definition 7 [**Unforgeability against chosen-subring attacks**] A ring signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *unforgeable against chosen-subring attacks* if for any PPT adversary $\mathcal{A}$ and for any polynomial $n(\cdot)$, the probability that $\mathcal{A}$ succeeds in the following game is negligible:

1. Key pairs $\{(PK_i, SK_i)\}_{i=1}^{n(k)}$ are generated using $\mathsf{Gen}(1^k)$, and the set of public keys $S \overset{\text{def}}{=} \{PK_i\}_{i=1}^{n(k)}$ is given to $\mathcal{A}$.
2. $\mathcal{A}$ is given access to a *signing oracle* $\mathsf{OSign}(\cdot, \cdot, \cdot)$, where $\mathsf{OSign}(s, M, R)$ outputs $\mathsf{Sign}_{SK_s}(M, R)$ and we require that $R \subseteq S$ and $PK_s \in R$.
3. $\mathcal{A}$ outputs (R^*, M^*, σ^*) and succeeds if $R^* \subseteq S$, $\mathsf{Vrfy}_{R^*}(M^*, \sigma^*) = 1$, and $\mathcal{A}$ never queried $(\star, M^*, R^*)$ to its signing oracle. $\diamondsuit$

While the above definition is an improvement, it still leaves open the possibility of an attack whereby honest users are "tricked" into generating signatures using rings containing adversarially-generated public keys. (Such an attack was also previously suggested by [21, 20].) The following definition takes this into account as well as, for completeness, an adversary who adaptively corrupts honest participants and obtains their secret keys. Since either of these attacks may be viewed as the outcome of corrupting an "insider" (even though, technically speaking, there are not really any "insiders" in the context of ring signatures), we use this terminology.

Definition 8 [**Unforgeability w.r.t. insider corruption**] A ring signature scheme $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is *unforgeable w.r.t. insider corruption* if for any PPT adversary $\mathcal{A}$ and for any polynomial $n(\cdot)$, the probability that $\mathcal{A}$ succeeds in the following game is negligible:

1. Key pairs $\{(PK_i, SK_i)\}_{i=1}^{n(k)}$ are generated using $\mathsf{Gen}(1^k)$, and the set of public keys $S \overset{\text{def}}{=} \{PK_i\}_{i=1}^{n(k)}$ is given to $\mathcal{A}$.
2. $\mathcal{A}$ is given access to a *signing oracle* $\mathsf{OSign}(\cdot, \cdot, \cdot)$, where $\mathsf{OSign}(s, M, R)$ outputs $\mathsf{Sign}_{SK_s}(M, R)$ and we require that $PK_s \in R$.
3. $\mathcal{A}$ is also given access to a *corrupt oracle* $\mathsf{Corrupt}(\cdot)$, where $\mathsf{Corrupt}(i)$ outputs SK_i.
4. $\mathcal{A}$ outputs (R^*, M^*, σ^*) and succeeds if $\mathsf{Vrfy}_{R^*}(M^*, \sigma^*) = 1$, $\mathcal{A}$ never queried $(\star, M^*, R^*)$, and $R^* \subseteq S \setminus C$, where C is the set of corrupted users. $\diamondsuit$

Herranz [17] considers, albeit informally, a definition intermediate between our Definitions 7 and 8 in which corruptions of honest players are allowed but adversarially-chosen public keys are not explicitly mentioned.

4 Separations Between the Security Definitions

In the previous section, we presented various definitions of anonymity and unforgeability. Here, we show that these definitions are in fact distinct, in the sense that there exist (under certain assumptions) schemes satisfying a weaker definition but not a stronger one. First, we show separations for the definitions of anonymity, considering in each case a scheme simultaneously satisfying the strongest definition of unforgeability. (Proof sketches for the claims presented in this section are given in Appendix A.1.)

Claim 1 *If there exists a scheme that achieves* **basic** *anonymity and is unforgeable w.r.t. insider corruption, then there exists a scheme that achieves these same properties but is* **not** *anonymous w.r.t.* **adversarially-chosen keys**.

Claim 2 *If there exists a scheme that is anonymous w.r.t.* **adversarially-chosen keys** *and is unforgeable w.r.t. insider corruption, then there exists a scheme that achieves these same properties but is* **not** *anonymous against* **attribution attacks**.

Claim 3 *Assume the existence of a semantically-secure public-key encryption scheme. If there exists a scheme that is anonymous against* **attribution attacks** *and is unforgeable w.r.t. insider corruption, then there exists a scheme that achieves these same properties but is* **not** *anonymous against* **full key exposure**.

Looking ahead, the initial scheme we present in the following section also gives an example of a ring signature scheme having the properties described in Claim 3. However, we rely there on stronger assumptions than semantically-secure public-key encryption.

We also show separations for the definitions of unforgeability, considering now schemes which simultaneously achieve the strongest definition of anonymity:

Claim 4 *If there exists a scheme that is anonymous against full key exposure and unforgeable w.r.t. insider corruption, then there exists a scheme that is anonymous against full key exposure and unforgeable against* **fixed-ring attacks**, *but* **not** *unforgeable against* **chosen-subring attacks**.

We remark that the scheme of [18] serves as a *natural* example of a scheme that is unforgeable against fixed-ring attacks but **not** unforgeable against chosen-subring attacks (in the random oracle model); this was subsequently fixed in [17]. See Appendix A.2.

Claim 5 *If there exists a scheme that is anonymous against full key exposure and unforgeable against* **chosen-subring attacks**, *then there exists a scheme that achieves these same properties but is* **not** *unforgeable w.r.t.* **insider corruption**.

5 Ring Signatures Based on General Assumptions

We now describe our construction of a ring signature scheme that satisfies the strongest of our proposed definitions, and is based on general assumptions. Let $(\mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec})$ be a semantically-secure public-key encryption scheme, let $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ be a (standard) signature scheme, and let $(\ell, \mathcal{P}, \mathcal{V})$ be a ZAP (for an $\mathcal{NP}$-language that will become clear once we describe the scheme). We denote by $C^* \leftarrow \mathsf{Enc}^*_{R_E}(m)$ the probabilistic algorithm that takes as input a set of encryption public keys $R_E = \{pk_{E,1}, \ldots, pk_{E,n}\}$ and a message m, and does the following: it first chooses random $s_1, \ldots, s_{n-1} \in \{0,1\}^{|m|}$ and then outputs:

$$C^* = \left(\mathsf{Enc}_{pk_{E,1}}(s_1), \mathsf{Enc}_{pk_{E,2}}(s_2), \cdots, \mathsf{Enc}_{pk_{E,n-1}}(s_{n-1}), \mathsf{Enc}_{pk_{E,n}}(m \oplus \bigoplus_{j=1}^{n-1} s_j)\right).$$

Note that, informally, encryption using Enc^* is semantically secure as long as at least one of the corresponding secret keys is unknown.

The idea underlying our construction is the following. Each user has an encryption key pair (pk_E, sk_E) and a (standard) signature key pair (pk_S, sk_S). To generate a ring signature with respect to a ring R of n users, the signer first produces a standard signature σ' of $M \| R$ with her signing key. Next, the signer produces two ciphertexts C_0^*, C_1^* using Enc^* and the set R_E of all the *encryption* public keys in the ring; one of these ciphertexts will be an encryption of σ' and the other will be an encryption of "garbage". Finally, the signer produces a proof π (using the ZAP) that at least one of the ciphertexts is an encryption of a valid signature on the message with respect to the verification key of one of the ring members.

Toward a formal description, let L denote the $\mathcal{NP}$ language:

$$\left\{(R_S, M, R_E, C^*) : \exists pk_S, \sigma, \omega \text{ s.t. } pk_S \in R_S \bigwedge C^* = \mathsf{Enc}^*_{R_E}(\sigma; \omega) \bigwedge \mathsf{Vrfy}'_{pk_S}(M, \sigma) = 1\right\};$$

i.e., $(R_S, M, R_E, C^*) \in L$ iff C^* is an encryption (using $\mathsf{Enc}^*_{R_E}$) of a valid signature of M with respect to some verification key $pk_S \in R_S$. We now give the details of our construction, which is specified by the key-generation algorithm Gen, the ring signing algorithm Sign, and the ring verification algorithm Vrfy.

<u>$\mathsf{Gen}(1^k)$</u>:

1. Generate signing key pair $(pk_S, sk_S) \leftarrow \mathsf{Gen}'(1^k)$.
2. Generate encryption key pair $(pk_E, sk_E) \leftarrow \mathsf{EGen}(1^k)$ and erase sk_E.
3. Choose an initial ZAP message $r \leftarrow \{0,1\}^{\ell(k)}$.
4. Output the public key $PK := (pk_S, pk_E, r)$, and the secret key $SK := sk_S$.

<u>$\mathsf{Sign}_{i^*, SK_{i^*}}(M, (PK_1, \ldots, PK_n))$</u>:

1. Parse each PK_i as $(pk_{S,i}, pk_{E,i}, r_i)$, and parse SK_{i^*} as sk_{S,i^*}. Set $R_E := \{pk_{E,1}, \ldots, pk_{E,n}\}$ and $R_S := \{pk_{S,1}, \ldots, pk_{S,n}\}$.
2. Set $M^* := M \,|\, PK_1 \,|\, \cdots \,|\, PK_n$, where "$|$" denotes concatenation. Compute the signature $\sigma'_{i^*} \leftarrow \mathsf{Sign}'_{sk_{S,i^*}}(M^*)$.
3. Choose random coins ω_0, ω_1 and compute $C_0^* := \mathsf{Enc}^*_{R_E}(\sigma'_{i^*}; \omega_0)$ and[3] $C_1^* := \mathsf{Enc}^*_{R_E}(0^k; \omega_1)$.
4. For $j \in \{0,1\}$, let x_j denote the statement: "$(R_S, M^*, R_E, C_j^*) \in L$", and let $x := x_0 \bigvee x_1$. Compute the proof $\pi \leftarrow \mathcal{P}_{r_1}(x, (pk_{S,i^*}, \sigma'_{i^*}, \omega_0))$.
5. The signature is $\sigma := (C_0^*, C_1^*, \pi)$.

[3]We assume for simplicity that valid signatures w.r.t. the verification keys $\{pk_{S,i}\}$ always have length k. The construction can easily be adapted when this is not the case.

$\underline{\mathsf{Vrfy}_{\{PK_1,\ldots,PK_n\}}(M,\sigma)}$

1. Parse σ as (C_0^*, C_1^*, π). Parse each PK_i as $(pk_{S,i}, pk_{E,i}, r_i)$. Set $M^* := M \,|\, PK_1 \,|\, \cdots \,|\, PK_n$; set $R_E := \{pk_{E,1}, \ldots, pk_{E,n}\}$; and set $R_S := \{pk_{S,1}, \ldots, pk_{S,n}\}$.
2. For $j \in \{0,1\}$, let x_j denote the statement: "$(R_S, M^*, R_E, C_j^*) \in L$", and let $x := x_0 \bigvee x_1$.
3. Output $\mathcal{V}_{r_1}(x, \pi)$.

(In the conference version of this work [4], we presented a less efficient version of the scheme in which the signature contained n ciphertexts rather than two.) It is easy to see that the scheme above satisfies the functional definition of a ring signature scheme (recall that the $\{PK_i\}$ in a ring are always ordered lexicographically). We now prove that the scheme satisfies strong notions of anonymity and unforgeability (here we only claim anonymity against attribution attacks, but we discuss further below how it is possible to obtain anonymity against full key exposure):

Theorem 1 *If encryption scheme* $(\mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec})$ *is semantically secure, signature scheme* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *is existentially unforgeable under adaptive chosen-message attacks, and* $(\ell, \mathcal{P}, \mathcal{V})$ *is a ZAP for the language* $L' = \{(x_0, x_1) : x_0 \in L \bigvee x_1 \in L\}$*, then the above ring signature scheme is anonymous against attribution attacks and unforgeable w.r.t. insider corruption.*

Proof We prove each of the desired security properties in turn.

Anonymity. For simplicity of exposition, we consider Definition 5 with $n = 2$; i.e., we assume only two users numbers 0 and 1 for convenience. We also assume without loss of generality that $i_0 = 0$ and $i_1 = 1$, and that the adversary corrupts player 1. It is straightforward that this implies the general case. Given a PPT adversary $\mathcal{A}$, we consider a sequence of experiments E_0, Hybrid_0, Hybrid_1, E_1 such that E_0 (resp., E_1) corresponds to the experiment of Definition 5 with $b = 0$ (resp., $b = 1$), and such that each experiment is computationally indistinguishable from the one before it. This implies that $\mathcal{A}$ has negligible advantage in distinguishing E_0 from E_1, as desired.

For convenience, we review experiment E_0. Here, two key pairs $(PK_0 = (pk_{S,0}, pk_{E,0}, r_0), SK_0)$ and $(PK_1 = (pk_{S,1}, pk_{E,1}, r_1), SK_1)$ are generated and $\mathcal{A}$ is given PK_0 and the randomness used to generate (PK_1, SK_1) (recall our assumption that $\mathcal{A}$ corrupts player 1). The adversary is also given access to a signing oracle that can be used to obtain signatures computed using SK_0. The adversary $\mathcal{A}$ then outputs a message M along with a ring of public keys R containing both PK_0 and PK_1. Finally, $\mathcal{A}$ is given $\sigma \leftarrow \mathsf{Sign}_{SK_0}(M, R)$.

Experiment Hybrid_0 is the same as experiment E_0 except that we change how the signature σ is generated: in step 2 we compute $\sigma_0' \leftarrow \mathsf{Sign}'_{sk_{S,0}}(M^*)$ and C_0^* as before, but now additionally compute $\sigma_1' \leftarrow \mathsf{Sign}'_{sk_{S,1}}(M^*)$ and $C_1^* = \mathsf{Enc}^*_{R_E}(\sigma_1'; \omega_1)$.

It is not hard to see that experiment Hybrid_0 is computationally indistinguishable from experiment E_0, assuming semantic security of the encryption scheme $(\mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec})$. This follows from the observations that (1) adversary $\mathcal{A}$ is *not* given the random coins used in generating PK_0 and so, in particular, it is not given the coins used to generate $pk_{E,0}$; (2) (informally) semantic security of encryption under $\mathsf{Enc}_{pk_{E,0}}$ implies semantic security of encryption using $\mathsf{Enc}^*_{R_E}$ as long as $pk_{E,0} \in R_E$ (a formal proof is straightforward); and, finally, (3) the coins ω_1 used in generating C_1^* are not used in the remainder of the experiment.

Experiment Hybrid_1 is the same as Hybrid_0 except that we use a different witness when computing the proof π for the ZAP. In particular, instead of using witness $(pk_{S,0}, \sigma_0', \omega_0)$ we use the witness $(pk_{S,1}, \sigma_1', \omega_1)$. The remainder of the signing algorithm is unchanged.

It is relatively immediate that Hybrid_1 is computationally indistinguishable from Hybrid_0, assuming witness indistinguishability of the ZAP. (Use of a ZAP, rather than non-interactive zero-knowledge, is essential here since the adversary may choose the "random string" component of all the adversarially-chosen public keys any way it likes.) In more detail, we can construct the following malicious verifier algorithm $\mathcal{V}^*$ using $\mathcal{A}$: verifier $\mathcal{V}^*$ generates (PK_0, SK_0) and (PK_1, SK_1) exactly as in experiments Hybrid_0 and Hybrid_1, and gives these keys and the appropriate associated random coins to $\mathcal{A}$. Signing queries of $\mathcal{A}$ can be easily simulated by $\mathcal{V}^*$ in the obvious way. When $\mathcal{A}$ outputs M and R with $PK_0, PK_1 \in R$, the verifier $\mathcal{V}^*$ computes C_0^* and C_1^* exactly as in Hybrid_1 and then gives to the prover $\mathcal{P}$ the sets R_S and R_E, the message M^*, and the ciphertexts (C_0^*, C_1^*); this defines the $\mathcal{NP}$-statement x exactly as in step 4 of the ring signing algorithm. In addition, $\mathcal{V}^*$ gives the two witnesses $w_0 = (pk_{S,0}, \sigma_0', \omega_0)$ and $w_1 = (pk_{S,1}, \sigma_1', \omega_1)$ to $\mathcal{P}$. Finally, $\mathcal{V}^*$ sends as its first message the "random string" component r of the lexicographically-first public key in R (this r is the random string that would be used to generate the proof π in step 4 of the ring signing algorithm). The prover responds with a proof $\pi \leftarrow \mathcal{P}_r(x, w_b)$ (for some $b \in \{0,1\}$), and then $\mathcal{V}^*$ returns the signature (C_0^*, C_1^*, π) to $\mathcal{A}$.

Note that if the prover uses the first witness provided to it by $\mathcal{V}^*$ then the view of $\mathcal{A}$ is distributed exactly according to Hybrid_0, while if the prover uses the second witness provided to it by $\mathcal{V}^*$ then the view of $\mathcal{A}$ is distributed exactly according to Hybrid_1. Witness indistinguishability of the ZAP thus implies computational indistinguishability of Hybrid_0 and Hybrid_1.

We may now notice that Hybrid_1 is computationally indistinguishable from E_1 by exactly the same argument used to show the indistinguishability of Hybrid_0 and E_0. This completes the proof.

Unforgeability. Assume there exists a PPT adversary $\mathcal{A}$ that breaks the above ring signature scheme (in the sense of Definition 8) with non-negligible probability. We construct an adversary $\mathcal{A}'$ that breaks the underlying signature scheme $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ (in the standard sense of existential unforgeability) with non-negligible probability.

$\mathcal{A}'$ receives as input a public key pk_S. Let $n = n(k)$ be a bound on the number of (honest user) public keys that $\mathcal{A}$ expects to be generated. $\mathcal{A}'$ runs $\mathcal{A}$ with input public keys $S = \{PK_1, \ldots, PK_n\}$, that $\mathcal{A}'$ generates as follows. $\mathcal{A}'$ chooses $i^* \leftarrow \{1, \ldots, n\}$ and sets $pk_{S,i^*} = pk_S$. The remainder of public key PK_{i^*} is generated exactly as prescribed by the Gen algorithm, with the exception that the decryption key sk_{E,i^*} that is generated is *not* erased. Public keys PK_i for $i \neq i^*$ are also generated exactly as prescribed by the Gen algorithm, again with the exception that the decryption keys $\{sk_{E,i}\}$ are not erased.

$\mathcal{A}'$ then proceeds to simulate the oracle queries of $\mathcal{A}$ in the natural way:

1. When $\mathcal{A}$ requests a signature on message M, with respect to ring R (which may possibly contain some public keys generated in an arbitrary manner by $\mathcal{A}$), to be signed by user $i \neq i^*$, then $\mathcal{A}'$ can easily generate the response to this query by running the Sign algorithm completely honestly;

2. When $\mathcal{A}$ requests a signature on message M, with respect to ring R (which, again, may possibly contain some public keys generated in an arbitrary manner by $\mathcal{A}$) to be signed by user i^*, then $\mathcal{A}'$ cannot directly respond to this query since it does not know sk_{S,i^*}. Instead, $\mathcal{A}'$ submits the appropriate message M^* to its signing oracle, and obtains in return a signature σ_{i^*}'. It then computes the remainder of the ring signature by following the rest of the Sign algorithm; note, in particular, that sk_{S,i^*} is not needed for this;

3. Any corruption query made by $\mathcal{A}$ for a user $i \neq i^*$ can be faithfully answered by $\mathcal{A}'$. On the other hand, if $\mathcal{A}$ ever makes a corruption query for i^*, then $\mathcal{A}'$ simply aborts.

At some point, $\mathcal{A}$ outputs a forgery $\bar{\sigma} = (\bar{C}_0^*, \bar{C}_1^*, \bar{\pi})$ on a message $\bar{M}$ with respect to some ring $\bar{R} \subseteq S$, none of whose members is corrupted. If $PK_{i^*} \notin \bar{R}$, then $\mathcal{A}'$ aborts. Otherwise, since $\mathcal{A}'$ knows all relevant decryption keys, it can decrypt both $\bar{C}_0^*$ and $\bar{C}_1^*$ and obtain candidate signatures $\bar{\sigma}_{i^*,0}$ and $\bar{\sigma}_{i^*,1}$ respectively. Finally, $\mathcal{A}'$ sets $\bar{M}^* = \bar{M} \,|\, \overline{PK}_1 \,|\, \cdots \,|\, \overline{PK}_{n'}$ (where $\bar{R} = \{\overline{PK}_i\}$) and verifies whether either of $\sigma_{i^*,0}$ or $\bar{\sigma}_{i^*,1}$ is a valid signature for M^* with respect to $pk_S = pk_{S,i^*}$; if so, it outputs $\bar{M}^*$ along with the valid signature. Note that (by requirement) $\mathcal{A}$ never requested a signature on message $\bar{M}$ with respect to the ring $\bar{R}$, and so $\mathcal{A}'$ never requested a signature on message $\bar{M}^*$ from its own oracle.

We claim that if $\mathcal{A}$ forges a signature with non-negligible probability $\varepsilon = \varepsilon(k)$, then $\mathcal{A}'$ forges a signature with probability at least $\varepsilon' = \varepsilon/n - \mathsf{negl}(k)$. To see this, note first that if $\mathcal{A}$ outputs a valid forgery then with all but negligible probability (by adaptive soundness of the ZAP) it holds that $(\bar{R}_S, \bar{M}^*, \bar{R}_E, \bar{C}_j^*) \in L$ for some j (where $\bar{R}_S, \bar{R}_E$ are defined in the natural way based on the ring $\bar{R}$ and the public keys it contains). Assuming this to be the case, we know that there exist i, j, σ, and ω such that $C_j^* = \mathsf{Enc}^*_{R_E}(\sigma; \omega)$ and $\mathsf{Vrfy}'_{pk_{S,i}}(M^*, \sigma) = 1$ for some $pk_{S,i} \in \bar{R}$. Conditioned on this, with probability at least $1/n - \mathsf{negl}(k)$ it is the case that $\mathcal{A}'$ does not abort and can recover a valid signature on $\bar{M}^*$ with respect to $pk_S = pk_{S,i^*}$. (We rely here on the fact that with all but negligible probability over choice of encryption public keys, Enc^* has zero decryption error). Security of $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ thus implies that ε is negligible. ■

"Oblivious" users. The scheme above can also be used in a situation where some users in the ring have not generated a key pair according to Gen, as long as (1) every ring member has a public key both for encryption and for signing (these keys may be associated with different schemes), and (2) at least one of the members has included a sufficiently-long random string in his public key. Thus, a *single* user who establishes a public key for a ring signature scheme suffices to provide anonymity for everyone. This also provides a way to include "oblivious" users in the signing ring [1, 2].

Achieving a stronger anonymity guarantee. The above scheme is not secure against full key exposure, and it is essential to our proof of anonymity that the adversary not be given the random coins used to generate *all* (honest) ring signature keys.[4] (If the adversary gets all sets of random coins, it can decrypt ciphertexts encrypted using $\mathsf{Enc}^*_{R_E}$ for any ring of honest users R and thereby determine the true signer of a message.) It is possible to achieve anonymity against full key exposure using an enhanced form of encryption for which, informally, there exists an "oblivious" way to generate a public key without generating a corresponding secret key. This notion, introduced by Damgård and Nielsen [12], can be viewed as a generalization of *dense cryptosystems* in which the public key is required to be a uniformly distributed string (in particular, dense cryptosystems satisfy the definition below). We review the formal definition here.

Definition 9 An *oblivious key generator* for the public-key encryption scheme (EGen, Enc, Dec) is a pair of PPT algorithms (OblEGen, OblRand) such that:

- OblEGen, on input 1^k and random coins $\omega \in \{0,1\}^{n(k)}$, outputs a key pk;
- OblRand, on input a key pk, outputs a string ω;

and the following distribution ensembles are computationally indistinguishable:

$$\left\{ \omega \leftarrow \{0,1\}^{n(k)} : (\omega, \mathsf{OblEGen}(1^k; \omega)) \right\}$$

[4]We remark that anonymity still holds if the adversary is given all *secret keys* (but not the randomness used to generate all secret keys). This is because the decryption key sk_E is erased, and not included in SK.

and

$$\left\{(pk, sk) \leftarrow \mathsf{EGen}(1^k); \omega \leftarrow \mathsf{OblRand}(pk) : (\omega, pk)\right\} .$$

$\diamondsuit$

Note that if (EGen, Enc, Dec) is semantically secure, then (informally speaking) it is also semantically secure to encrypt messages using a public key *pk* generated by OblEGen, even if the adversary has the random coins used by OblEGen in generating *pk*. We remark for completeness that the El Gamal encryption scheme (over the group of quadratic residues modulo a prime) is an example of a scheme having an oblivious key generator.

Given the above, we adapt our construction in the natural way: specifically, the Gen algorithm is changed so that instead of generating pk_E using EGen (and then erasing the secret key sk_E and the random coins used), we now generate pk_E using OblEGen. Adapting the proof of Theorem 1, we can easily show:

Theorem 2 *Under the assumptions of Theorem 1 and assuming* (EGen, Enc, Dec) *has an oblivious key generator, the modified ring signature scheme described above is anonymous against full key exposure, and unforgeable w.r.t. insider corruption.*

Proof The proof of unforgeability follows immediately from Theorem 1 since, by Definition 9, the adversary cannot distinguish between the original scheme (in which the encryption key is generated using EGen) and the modified scheme (in which the encryption key is generated using OblEGen).

We now argue that the modified scheme achieves anonymity against full key exposure. First note that anonymity against attribution attacks claimed in Theorem 1 holds even when the adversary is given all random coins used to generate (PK_0, SK_0) *except* for those coins used to generate $pk_{E,0}$ (using EGen). Now, if there exists a PPT adversary $\mathcal{A}$ that breaks anonymity of the modified scheme in the sense of full key exposure, we can use it to construct a PPT adversary $\mathcal{A}'$ that breaks anonymity of the original scheme against attribution attacks. $\mathcal{A}'$ receives PK_0, the random coins $\omega_{S,1}, \omega_{E,1}$ used to generate (PK_1, SK_1), and the random coins $\omega_{S,0}$ used to generate $pk_{S,0}$ (i.e., $\mathcal{A}$ is *not* given the coins used to generate $pk_{E,0}$). Next, $\mathcal{A}'$ runs $\omega'_{E,0} \leftarrow \mathsf{OblRand}(pk_{E,0})$ and $\omega'_{E,1} \leftarrow \mathsf{OblRand}(pk_{E,1})$ and gives to $\mathcal{A}$ the public key PK_0 it received as well as the random coins $\omega_{S,0}, \omega'_{E,0}, \omega_{S,1}, \omega'_{E,1}$. The remainder of $\mathcal{A}$'s execution is simulated in the natural way by $\mathcal{A}'$.

Definition 9 implies that the advantage of $\mathcal{A}$ in the above is negligibly close to the advantage of $\mathcal{A}$ in attacking the modified scheme in the sense of full key exposure. But the advantage of $\mathcal{A}$ in the above is exactly the advantage of $\mathcal{A}'$ in attacking the original scheme via key attribution attack. Since we have already proved that the original scheme is anonymous against attribution attacks (cf. Theorem 1), the modified scheme is anonymous against full key exposure. ■

6 Efficient Two-User Ring Signature Schemes

In this section, we present more efficient constructions of two-user ring signature schemes based on specific assumptions. Our first scheme is based on the (standard) signature scheme constructed by Waters [23], whereas the second is based on the Camenisch-Lysyanskaya signature scheme [7].

6.1 The Waters Scheme

We briefly review the Waters signature scheme. Let $\mathbb{G}, \mathbb{G}_1$ be groups of prime order q such that there exists an efficiently computable bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$. We assume that $q, \mathbb{G}, \mathbb{G}_1, \hat{e}$, and a generator $g \in \mathbb{G}$ are publicly known. The Waters signature scheme for messages of length n is defined as follows:

Key Generation. Choose $\alpha \leftarrow \mathbb{Z}_q$ and set $g_1 = g^\alpha$. Additionally choose random elements $h, u', u_1, \ldots, u_n \leftarrow \mathbb{G}$. The public key is $(g_1, h, u', u_1, \ldots, u_n)$ and the secret key is h^α.

Signing. To sign the n-bit message M, first compute $w = u' \cdot \prod_{i:M_i=1} u_i$. Then choose random $r \leftarrow \mathbb{Z}_q$ and output the signature $\sigma = (h^\alpha \cdot w^r,\ g^r)$.

Verification. To verify the signature (A, B) on message M with respect to public key $(g_1,\ h,\ u',\ u_1,\ \ldots,\ u_n)$, compute $w = u' \cdot \prod_{i:M_i=1} u_i$ and then check whether $\hat{e}(g_1, h) \cdot \hat{e}(B, w) \overset{?}{=} \hat{e}(A, g)$.

6.2 A 2-User Ring Signature Scheme

The main observation we make with regard to the above scheme is the following: element h is arbitrary, and only knowledge of h^α is needed to sign. So we can dispense altogether with including h in the public key; instead, a user U with secret α and the value $g_1 = g^\alpha$ in his public key will use as his "h-value" the value $\bar{g}_1$ contained in the public key of a *second* user $\bar{U}$. This provides anonymity since $\bar{U}$ could *also* have computed the same value $(\bar{g}_1)^\alpha$ using the secret value $\bar{\alpha} = \log_g \bar{g}_1$ known to him (because $\bar{g}_1^\alpha = g_1^{\bar{\alpha}}$). We now proceed with the details.

Key Generation. Choose $\alpha \leftarrow \mathbb{Z}_q$ and set $g_1 = g^\alpha$. Additionally choose random elements $u', u_1, \ldots, u_n \leftarrow \mathbb{G}$. The public key is $(g_1,\ u',\ u_1,\ \ldots,\ u_n)$ and the secret key is α. (We again assume that $q, \mathbb{G}, \mathbb{G}_1, \hat{e}$, and g are system-wide parameters.)

Ring Signing. To sign message $M \in \{0,1\}^n$ with respect to the ring $R = \{PK, \overline{PK}\}$ using secret key α (where we assume without loss of generality that α is the secret corresponding to PK), proceed as follows: parse PK as $(g_1,\ u',\ u_1,\ \ldots,\ u_n)$ and $\overline{PK}$ as $(\bar{g}_1,\ \bar{u}',\ \bar{u}_1,\ \ldots,\ \bar{u}_n)$, and compute $w := u' \cdot \prod_{i:M_i=1} u_i$ and $\bar{w} := \bar{u}' \cdot \prod_{i:M_i=1} \bar{u}_i$. Then choose random $r \leftarrow \mathbb{Z}_q$ and output the signature

$$\sigma = (\bar{g}_1^\alpha \cdot (w\bar{w})^r,\ g^r)\ .$$

Ring Verification. To verify the signature (A, B) on message M with respect to the ring $R = \{PK, \overline{PK}\}$ (parsed as above), compute $w := u' \cdot \prod_{i:M_i=1} u_i$ and $\bar{w} := \bar{u}' \cdot \prod_{i:M_i=1} \bar{u}_i$ and then check whether $\hat{e}(g_1, \bar{g}_1) \cdot \hat{e}(B, (w\bar{w})) \overset{?}{=} \hat{e}(A, g)$.

It is not hard to see that correctness holds. We prove the following regarding the above scheme:

Theorem 3 *Assume the Waters signature scheme is existentially unforgeable*[5] *under adaptive chosen message attack. Then the 2-user ring signature scheme described above is unconditionally anonymous against full key exposure, and unforgeable against chosen-subring attacks.*

Proof Unconditional anonymity against full key exposure follows easily from the observation made earlier: namely, that only the value $\bar{g}_1^\alpha = g_1^{\bar{\alpha}}$ (where $\bar{\alpha} \overset{\text{def}}{=} \log_g \bar{g}_1$) is needed to sign, and either of the two (honest) parties can compute this value.

We now prove that the scheme satisfies Definition 7. We do this by showing how an adversary $\mathcal{A}$ that forges a signature with respect to the ring signature scheme with non-negligible probability can be used to construct an adversary $\hat{\mathcal{A}}$ that forges a signature with respect to the Waters signature scheme (in the standard sense) with the same probability. For simplicity in the proof, we assume

[5]This holds [23] under the computational Diffie-Hellman assumption in $\mathbb{G}$.

that $\mathcal{A}$ only ever sees the public keys of two users, requests all signatures to be signed with respect to the ring R containing these two users, and forges a signature with respect to that same ring R. By a hybrid argument, it can be shown that (for this scheme) this is equivalent to the more general case when $\mathcal{A}$ may see multiple public keys, request signatures with respect to various (different) 2-user subsets, and then output a forgery with respect to any 2-user subset of its choice.

Construct $\hat{\mathcal{A}}$ as follows: $\hat{\mathcal{A}}$ is given the public key $(\hat{g}_1,\ \hat{h},\ \hat{u}',\ \hat{u}_1,\ \ldots,\ \hat{u}_n)$ of an instance of the Waters scheme. $\hat{\mathcal{A}}$ constructs two user public keys as follows: first, it sets $g_1 = \hat{g}_1$ and $\bar{g}_1 = \hat{h}$. Then, it chooses random $u', u_1, \ldots, u_n \leftarrow \mathbb{G}$ and sets $\bar{u}' = \hat{u}'/u'$ and $\bar{u}_i = \hat{u}_i/u_i$ for all i. It gives to $\mathcal{A}$ the public keys $(g_1,\ u',\ u_1,\ \ldots,\ u_n)$ and $(\bar{g}_1,\ \bar{u}',\ \bar{u}_1,\ \ldots,\ \bar{u}_n)$. Note that both public keys have the appropriate distribution. When $\mathcal{A}$ requests a ring signature on a message M with respect to the ring R containing these two public keys, $\hat{\mathcal{A}}$ requests a signature on M from its signing oracle, obtains in return a signature (A, B), and gives this signature to $\mathcal{A}$. This is a perfect simulation, since

$$\left(\hat{h}^{\log_g \hat{g}_1} \cdot \left(\hat{u}' \prod_{i:M_i=1} \hat{u}_i\right)^r,\ g^r\right) = \left(\bar{g}_1^{\log_g g_1} \cdot \left(u'\bar{u}' \prod_{i:M_i=1} u_i\bar{u}_i\right)^r,\ g^r\right),$$

which is an appropriately-distributed ring signature with respect to the public keys given to $\mathcal{A}$.

When $\mathcal{A}$ outputs a forgery (A^*, B^*) on a message M^*, this same forgery is output by $\hat{\mathcal{A}}$. Note that $\hat{\mathcal{A}}$ outputs a valid forgery whenever $\mathcal{A}$ does, since

$$\hat{e}(g_1, \bar{g}_1) \cdot \hat{e}\left(B^*, (u'\bar{u}' \textstyle\prod_{i:M_i^*=1} u_i\bar{u}_i)\right) = \hat{e}(A^*, g)$$

implies

$$\hat{e}(\hat{g}_1, \hat{h}) \cdot \hat{e}\left(B^*, (\hat{u}' \textstyle\prod_{i:M_i^*=1} \hat{u}_i)\right) = \hat{e}(A^*, g).$$

We conclude that $\hat{\mathcal{A}}$ outputs a forgery with the same probability as $\mathcal{A}$. Since, by assumption, the Waters scheme is secure, this completes the proof. ■

An efficiency improvement. A (slightly) more efficient variant of the above scheme is also possible. Key generation is the same as before, except that an additional, random *identifier* $I \in \{0,1\}^k$ is also chosen and included in the public key. Let $<_{\text{lex}}$ denote lexicographic order. To sign message $M \in \{0,1\}^n$ with respect to the ring $R = \{PK, \overline{PK}\}$, first parse PK as $(I,\ g_1,\ u',\ u_1,\ \ldots,\ u_n)$ and $\overline{PK}$ as $(\bar{I},\ \bar{g}_1,\ \bar{u}',\ \bar{u}_1,\ \ldots,\ \bar{u}_n)$. Choose random $r \leftarrow \mathbb{Z}_q$. If $I \leq_{\text{lex}} \bar{I}$, compute $w = u' \cdot \prod_{i:M_i=1} u_i$ and the signature

$$\sigma = (s \cdot w^r,\ g^r)\ ;$$

if $\bar{I} <_{\text{lex}} I$, compute $\bar{w} = \bar{u}' \cdot \prod_{i:M_i=1} \bar{u}_i$ and the signature

$$\sigma = (s \cdot \bar{w}^r,\ g^r)\ ,$$

where, in each case, $s = \bar{g}_1^{\alpha} = g_1^{\bar{\alpha}}$ is computed using whichever secret key is known to the signer. Verification is changed in the obvious way. A proof similar to the above shows that this scheme satisfies the same security properties as in Theorem 3.

6.3 A Construction Based on the Camenisch-Lysyanskaya Scheme

A second ring signature scheme based on similar ideas can be derived from the signature scheme of Camenisch and Lysyanskaya (scheme A in [7]), which we briefly review. Let $\mathbb{G}, \mathbb{G}_1, q, \hat{e}, g$ be as above (we again assume that these are publicly known). The Camenisch-Lysyanskaya signature scheme for messages in $\mathbb{Z}_q$ is defined as follows:

Key Generation. Choose $x, y \leftarrow \mathbb{Z}_q$ and set $X = g^x$ and $Y = g^y$. The public key is (X, Y) and the secret key is (x, y).

Signing. To sign the message $m \in \mathbb{Z}_q$, choose a random value $a \in \mathbb{G}$ and output the signature (a, a^y, a^{x+mxy}).

Verification. To verify the signature (a, b, c) on message m with respect to public key (X, Y), check that $\hat{e}(a, Y) \overset{?}{=} \hat{e}(g, b)$ and $\hat{e}(X, a) \cdot \hat{e}(X, b)^m \overset{?}{=} \hat{e}(g, c)$.

The reader is referred to [7] for details regarding the assumption under which the above scheme can be proven secure. As for adapting the above to a two-user ring signature scheme, our key observation is that knowledge of *either* (x, Y) *or* (X, y) is sufficient to generate a signature. In more detail:

- Using (x, Y), a signature on m may be computed as follows: choose random $r \in \mathbb{Z}_q$ and set $a = g^r$. Then output the signature $(a, Y^r, a^x Y^{mxr})$.
- Using (X, y), a signature on m may be computed as follows: choose random $r \in \mathbb{Z}_q$ and set $a = g^r$. Then output the signature (a, a^y, X^{r+mry}).

This suggests the following ring signature scheme: to generate a public key, choose $x \leftarrow \mathbb{Z}_q$ and a random identifier $I \in \{0,1\}^k$; the public key is $(I, X = g^x)$ and the secret key is x. To sign m with respect to ring $\big\{(I, X), (\bar{I}, \bar{X})\big\}$, proceed as follows: if $I \leq_{\rm lex} \bar{I}$, compute a Camenisch-Lysyanskaya signature (as described above) for the "public key" $(X, \bar{X})$; if $\bar{I} <_{\rm lex} I$, compute a Camenisch-Lysyanskaya signature for the "public key" $(\bar{X}, X)$. Verification is done in the obvious way.

Theorem 4 *Assume the Camenisch-Lysyanskaya signature scheme is existentially unforgeable under adaptive chosen message attack. Then the 2-user ring signature scheme described above is unconditionally anonymous against full key exposure, and unforgeable against chosen-subring attacks.*

Proof Unconditional anonymity against full key exposure is immediate. Unforgeability against chosen-subring attacks (assuming security of the Camenisch-Lysyanskaya scheme) can be easily proven as in Theorem 3. Specifically, we show how an adversary $\mathcal{A}$ that forges a signature with respect to the ring signature scheme with non-negligible probability can be used to construct an adversary $\hat{\mathcal{A}}$ that forges a signature with respect to the Camenisch-Lysyanskaya signature scheme (in the standard sense) with the same probability. For simplicity in the proof, we assume that $\mathcal{A}$ only ever sees the public keys of two users, requests all signatures to be signed with respect to the ring R containing these two users, and forges a signature with respect to that same ring R. As before, this implies the more general case.

Construct $\hat{\mathcal{A}}$ as follows: $\hat{\mathcal{A}}$ is given the public key (X, Y) of an instance of the Waters scheme. $\hat{\mathcal{A}}$ first chooses random $I, \bar{I} \in \{0,1\}^k$; assume without loss of generality that $I \leq_{\rm lex} \bar{I}$. Then $\hat{\mathcal{A}}$ gives $\mathcal{A}$ the public keys (I, X) and $(\bar{I}, Y)$. Note that both public keys have the appropriate distribution. When $\mathcal{A}$ requests a ring signature on a message M with respect to the ring R containing these two public keys, $\hat{\mathcal{A}}$ requests a signature on M from its signing oracle, obtains in return a signature (a, b, c), and gives this signature to $\mathcal{A}$. It is trivial to see that this is a perfect simulation.

When $\mathcal{A}$ outputs a forgery (a^*, b^*, c^*) on a message M^*, this same forgery is output by $\hat{\mathcal{A}}$. It is again trivial to see that this is a valid forgery for $\hat{\mathcal{A}}$, and so we conclude that $\hat{\mathcal{A}}$ outputs a forgery with the same probability as $\mathcal{A}$. Since, by assumption, the Camenisch-Lysyanskaya scheme is secure, this completes the proof. ■

Acknowledgments

We thank Giuseppe Ateniese for referring us to [5, 7], and Javier Herranz for informing us about [17].

References

[1] M. Abe, M. Ohkubo, and K. Suzuki. 1-out-of-n signatures from a variety of keys. In *Advances in Cryptology — Asiacrypt 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 415–432. Springer, 2002.

[2] B. Adida, S. Hohenberger, and R.L. Rivest. Ad-hoc-group signatures from hijacked keypairs. A preliminary version was presented at the *DIMACS Workshop on Theft in E-Commerce*, 2005. Manuscript available at `http://theory.lcs.mit.edu/~rivest/publications.html`.

[3] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology — Eurocrypt 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, 2003.

[4] A. Bender, J. Katz, and R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *Third Theory of Cryptography Conference, TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 60–79. Springer, 2006.

[5] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology — Eurocrypt 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.

[6] E. Bresson, J. Stern, and M. Szydlo. Threshold ring signatures and applications to ad-hoc groups. In *Advances in Cryptology — Crypto 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 465–480. Springer, 2002.

[7] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology — Crypto 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, 2004.

[8] D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology — Eurocrypt '91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, 1991.

[9] L. Chen, C. Kudla, and K.G. Paterson. Concurrent signatures. In *Advances in Cryptology — Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 287–305. Springer, 2004.

[10] S.S.M. Chow, V.K.-W. Wei, J.K. Liu, and T. H. Yuen. Ring signatures without random oracles. In *Proc. ACM Symposium on Information, Computer and Communications Security (ASIACCS) 2006*, pages 297–302. ACM, 2006.

[11] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology — Crypto '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.

[12] I. Damgård and J.B. Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *Advances in Cryptology — Crypto 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 432–450. Springer, 2000.

[13] Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup. Anonymous identification in ad-hoc groups. In *Advances in Cryptology — Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 609–626. Springer, 2004.

[14] C. Dwork and M. Naor. Zaps and their applications. In *Proc. 41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 283–293. IEEE, 2000.

[15] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero knowledge proofs under general assumptions. *SIAM J. Computing*, 29(1):1–28, 1999.

[16] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology — Crypto '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1987.

[17] J. Herranz. *Some digital signature schemes with collective signers.* PhD thesis, Universitat Politècnica de Catalunya, Barcelona, April 2005. Available at `http://www.lix.polytechnique.fr/~herranz/thesis.htm`.

[18] J. Herranz and G. Sáez. Forking lemmas for ring signature schemes. In *Progress in Cryptology — Indocrypt 2003*, volume 2904 of *Lecture Notes in Computer Science*, pages 266–279. Springer, 2003.

[19] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *Advances in Cryptology — Eurocrypt '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154. Springer, 1996.

[20] J.K. Liu, V.K. Wei, and D.S. Wong. Linkable spontaneous anonymous group signatures for ad hoc groups. In *Information Security and Privacy: 9th Australasian Conference (ACISP 2004)*, volume 3108 of *Lecture Notes in Computer Science*, pages 325–335. Springer, 2004.

[21] M. Naor. Deniable ring authentication. In *Advances in Cryptology — Crypto 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 481–498. Springer, 2002.

[22] R.L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret: Theory and applications of ring signatures. In *Theoretical Computer Science, Essays in Memory of Shimon Even*, volume 3895 of *Lecture Notes in Computer Science*, pages 164–186. Springer, 2006. Preliminary version in Asiacrypt 2001.

[23] B. Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology — Eurocrypt 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.

[24] J. Xu, Z. Zhang, and D. Feng. A ring signature scheme using bilinear pairings. In *Workshop on Information Security Applications (WISA)*, volume 3325 of *Lecture Notes in Computer Science*, pages 160–169. Springer, 2004.

[25] F. Zhang and K. Kim. ID-based blind signature and ring signature from pairings. In *Advances in Cryptology — Asiacrypt 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 533–547. Springer, 2002.

A Separation Results

A.1 Proofs of Claims 1–5

Proof of Claim 1: Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a ring signature scheme satisfying the conditions stated in the claim. Construct the following scheme Π': the key generation algorithm $\mathsf{Gen}'(1^k)$ computes $(PK, SK) \leftarrow \mathsf{Gen}(1^k)$ and outputs $PK' = 0|PK$ and $SK' = SK$. The signing algorithm $\mathsf{Sign}'_{s,SK_s}(M, R)$ checks whether all public keys in R begin with a "0": if so, it outputs $\sigma \leftarrow \mathsf{Sign}_{s,SK_s}(M, \bar{R})$ (where $\bar{R}$ contains the same keys as R, but with the leading bit of each key removed); otherwise, it outputs s. $\mathsf{Vrfy}'_R(M, \sigma)$ similarly checks whether all public keys in R begin with a "0": if so, it outputs $\mathsf{Vrfy}_{\bar{R}}(M, \sigma)$ (with $\bar{R}$ as above); otherwise, it outputs 0. (Note that correctness is only required to hold for rings containing honestly-generated public keys.)

Clearly, the above scheme does not achieve anonymity w.r.t. adversarially-chosen keys. On the other hand, it clearly still achieves basic anonymity. It is also not difficult to see that it remains unforgeable w.r.t. insider corruption.

Proof of Claim 2: Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a ring signature scheme satisfying the conditions stated in the claim. The existence of such a scheme implies the existence of a standard signature scheme, which in turn implies the existence of one-way functions, which in turn implies the existence of a symmetric-key encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ that is semantically-secure under chosen-plaintext attacks. Construct scheme Π' as follows: the key generation algorithm $\mathsf{Gen}'(1^k)$ computes $(PK, SK) \leftarrow \mathsf{Gen}(1^k)$ but additionally chooses $\kappa \leftarrow \{0,1\}^k$; it outputs $PK' = PK$ and $SK' = SK|\kappa$. The signing algorithm $\mathsf{Sign}'_{s,SK_s|\kappa}(M, R)$ computes $\sigma \leftarrow \mathsf{Sign}_{s,SK_s}(M, R)$ and $C \leftarrow \mathsf{Enc}_\kappa(0^k)$; it outputs the signature (σ, C). Verification is changed in the obvious way, simply ignoring the ciphertext included as part of the signature.

The scheme does not achieve anonymity under attribution attacks since, given a signature computed by a user with secret key $SK|\kappa$ with respect to any ring, as long as the adversary has all-but-one of the secret keys of the members of the ring (and, in particular, has the $\{\kappa_i\}$ values for all-but-one of the members), it can determine the correct signer with all but negligible probability. On the other hand, it is not hard to show that the scheme remains anonymous w.r.t. adversarially-chosen keys and also remains unforgeable w.r.t. insider corruption.

We remark that although the modified scheme, above, does not satisfy our formal definition of anonymity against attribution attacks, it does not quite allow an adversary to unambiguously prove to a third party that some user was the signer. (The issue is that the adversary can output whatever $\{\kappa_i\}$ it likes, and not the "actual" values it chose at the time of key generation.) This can be prevented, however, if we additionally require users to include a commitment to κ as part of their public key, and to include the corresponding decommitment as part of their secret key.

Proof of Claim 3: Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a ring signature scheme satisfying the conditions stated in the claim, and let $(\mathsf{EGen}, \mathsf{Enc}, \mathsf{Dec})$ be a semantically-secure public-key encryption scheme. We denote by $C^* \leftarrow \mathsf{Enc}^*_{R_E}(m)$ the probabilistic algorithm that takes as input a set of encryption public keys $R_E = \{pk_1, \ldots, pk_n\}$ and a message m, and does the following: it first chooses random

$s_1, \ldots, s_{n-1} \in \{0,1\}^{|m|}$ and then outputs:

$$C^* = \left(\mathsf{Enc}_{pk_1}(s_1), \mathsf{Enc}_{pk_2}(s_2), \cdots, \mathsf{Enc}_{pk_{n-1}}(s_{n-1}), \mathsf{Enc}_{pk_n}(m \oplus \bigoplus_{j=1}^{n-1} s_j) \right).$$

Informally, encryption using Enc^* is semantically secure as long as at least one of the corresponding secret keys is unknown. (We will use the same encryption scheme in Section 5.)

Construct $\Pi' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ as follows: $\mathsf{Gen}'(1^k)$ computes $(PK, SK) \leftarrow \mathsf{Gen}(1^k)$ and $(pk, sk) \leftarrow \mathsf{EGen}(1^k)$, and then sets the public key equal to $PK|pk$ and the secret key equal to SK. The signing algorithm $\mathsf{Sign}'_{s,SK_s}(M, R)$ parses each public key in R as $PK_i|pk_i$ and sets $R' = \{PK_i\}$ and $R_E = \{pk_i\}$. Next, it computes $\sigma \leftarrow \mathsf{Sign}_{s,SK_s}(M, R')$ and $C^* \leftarrow \mathsf{Enc}^*_{R_E}(s)$ and outputs (σ, C^*) as the signature. Verification simply ignores the second component.

It is immediate that Π' is not anonymous against full key exposure (since C^* can be decrypted given the decryption keys $\{sk_i\}$ of the entire ring). On the other hand, it is not difficult to see that Π' remains anonymous against attribution attacks (since C^* cannot be decrypted if one of the secret keys is missing), and unforgeable w.r.t. insider corruption.

Note that our construction of Section 5 is anonymous against attribution attacks and unforgeable w.r.t. insider corruption, but is easily seen to be insecure against full key exposure. (However, that construction also assumes the existence of a ZAP.)

Proof of Claim 4: Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a ring signature scheme satisfying the conditions of the claim. Construct $\Pi' = (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ as follows. The key generation algorithm Gen' is the same as Gen. The signing algorithm $\mathsf{Sign}'_{s,SK_s}(M, R)$ sets $R' = R \cup \{M\}$ (where M is treated as a public key) and computes $\sigma_1 \leftarrow \mathsf{Sign}_{s,SK_s}(M, R)$ and $\sigma_2 \leftarrow \mathsf{Sign}_{s,SK_s}(0^k, R')$. The output is the signature (σ_1, σ_2). To verify a signature (σ_1, σ_2) (using Vrfy'), simply verify that signature σ_1 is correct (using Vrfy).

It is easy to see that the scheme is insecure against chosen-subring attacks. Specifically, consider the adversary $\mathcal{A}$ who receives the set of public keys (PK_1, PK_2, PK_3) and then requests a signature on the message $M = PK_3$ with respect to the ring $R = (PK_1, PK_2)$. Let (σ_1, σ_2) be the response of the signing oracle. $\mathcal{A}$ outputs $(0^k, (\sigma_2, \sigma_2), (PK_1, PK_2, PK_3))$ and terminates. Note that (σ_2, σ_2) is a valid ring signature (with respect to the scheme Π') for the message 0^k with respect to the ring (PK_1, PK_2, PK_3). Also note that $\mathcal{A}$ never requested a signature for such a message/ring pair. We therefore conclude that $\mathcal{A}$ succeeds in producing a valid forgery with probability 1.

It is quite obvious that Π' remains anonymous against full key exposure. Π' is also unforgeable against fixed-ring attacks. We prove this by contradiction. Let $\mathcal{A}'$ be an adversary that breaks the unforgeability of Π' against fixed-ring attacks. We construct an adversary $\mathcal{A}$ that breaks the unforgeability of Π w.r.t. insider corruption. (We remark that we do not use $\mathcal{A}$'s ability to corrupt users here, but only its ability to request signatures with respect to rings containing adversarially-generated keys.) $\mathcal{A}$ takes as input a ring $R = (PK_1, \ldots, PK_n)$ and feeds it to $\mathcal{A}'$. When $\mathcal{A}'$ requests a signature (under Sign') on the message M with respect to the fixed ring R, $\mathcal{A}$ uses its signing oracle to obtain the two components σ_1 and σ_2. Note that $\mathcal{A}$ can obtain σ_2, because it can request a signature on a ring that contains public keys of its choice (M in this case). When $\mathcal{A}'$ outputs a candidate forgery $(M, (\sigma_1, \sigma_2))$ with respect to the fixed ring R, then $\mathcal{A}$ outputs σ_1 as a candidate forgery for message M with respect to the ring R. If the output of $\mathcal{A}'$ is a valid signature with respect to Π', then the output of $\mathcal{A}$ is a valid signature with respect to Π. Also, if $\mathcal{A}'$ never requested a signature on M, then $\mathcal{A}$ never requested a signature on M with respect to the ring R. We conclude that $\mathcal{A}$ outputs a valid forgery whenever $\mathcal{A}'$ does.

Proof of Claim 5: Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a scheme satisfying the conditions of the claim. We construct the scheme Π' as follows. The key generation algorithm Gen' runs Gen to obtain (PK, SK), then outputs $PK' = 0|PK$ and $SK' = SK$. We will say that a public key is "good" if it begins with a zero and that it is "bad" if it begins with a one. Note that all public keys generated by Gen' are "good."

The signing algorithm $\mathsf{Sign}'_{s,SK_s}(M, R)$ proceeds as follows: let R' be the ring consisting of only the "good" public keys from R, with the initial bit stripped. Then compute $\sigma \leftarrow \mathsf{Sign}_{s,SK_s}(M, R')$ and output this as the signature. The verification algorithm is modified in the appropriate way.

Π' is not unforgeable w.r.t. insider corruption. To see this, consider the adversary $\mathcal{A}$ who receives public keys (PK'_1, PK'_2). Next, $\mathcal{A}$ generates an arbitrary "bad" public key $PK' = 1|PK''$. The adversary then requests a signature on an arbitrary message M with respect to the ring (PK'_1, PK'_2, PK') on behalf of the signer holding PK'_1. The signing oracle returns a signature σ that is a valid signature for message M respect to the ring (PK'_1, PK'_2) (recall that PK' is ignored, since it is "bad"). But now $\mathcal{A}$ can output the forgery $(M, \sigma, (PK'_1, PK'_2))$ and succeed with probability 1.

It is not hard to see that Π' remains unforgeable against chosen-subring attacks (since, in such attacks, the adversary can only request signatures with respect to rings that consist entirely of "good" public keys). One can also easily show that Π' remains anonymous w.r.t. key exposures.

A.2 The Herranz-Sáez Ring Signature Scheme

In the proof of Claim 4 (above), we presented an "artificial" ring signature scheme that is unforgeable against fixed-ring attacks, but not against chosen-subring attacks. We now show a "natural" scheme, the Herranz-Sáez ring signature scheme [18], that illustrates the same separation (albeit under less general assumptions).

We first review the Herranz-Sáez ring signature scheme. Let $\mathbb{G}$ be a group of prime order q, such that, given a bit string y it is possible to efficiently verify whether $y \in \mathbb{G}$. Let $H : \{0,1\}^* \to \mathbb{Z}_q$ be a hash function modeled as a random oracle. We assume that H, q, $\mathbb{G}$, and a generator $g \in \mathbb{G}$ are publicly known. The scheme is defined as follows:

Key Generation. Choose $x \leftarrow \mathbb{Z}_q$ and set $y = g^x$. The public key is y and the secret key is x.

Ring Signing. To sign message M with respect to the ring $R = \{y_1, \ldots, y_n\}$ (where $y_i \in \mathbb{G}$ for all i) using secret key x_s, proceed as follows:

1. For $i = 1, \ldots, n$, $i \neq s$, choose random $a_i \leftarrow \mathbb{Z}_q$ and set $C_i = g^{a_i}$;
2. Choose random $a_s \leftarrow \mathbb{Z}_q$;
3. Compute C_s and b as follows:

$$\begin{aligned} C_s &= g^{a_s} \prod_{i \neq s} y_i^{-H(M, C_i)} \\ b &= a_s + x_s H(M, C_s) + \sum_{i \neq s} a_i; \end{aligned}$$

4. In the unlikely event that the C_i are not all distinct, restart from the beginning;
5. Output the signature $\sigma = (b, C_1, \ldots, C_n)$.

Ring Verification. To verify the signature $(b, C_1, \ldots, C_n)$ on message M with respect to the ring $R = \{y_1, \ldots, y_n\}$ (where $y_i, C_i \in \mathbb{G}$ for all i), check that the C_i are all distinct and that:

$$g^b \stackrel{?}{=} \prod_{i=1}^{n} C_i \cdot y_i^{H(M,R_i)}.$$

It is not hard to see that the scheme above is unconditionally anonymous against full key exposure, even in the standard model. This is because a ring signature on message M with respect to a ring R is a uniformly random sample from the set of tuples $(b, C_1, \ldots, C_n)$ that satisfy the ring verification condition, and this distribution is independent of the index s of the signing key used. Additionally, Herranz and Sáez [18] prove that this scheme is unforgeable against fixed-ring attacks under the discrete logarithm assumption in the random oracle model. (The authors do not formally define unforgeability, but an inspection of their proof of security reveals that their notion of unforgeability matches our Definition 6.)

However, the Herranz-Sáez scheme is *not* unforgeable against chosen-subring attacks. Consider an adversary that requests two signatures on the same arbitrary message M with respect to the *disjoint* rings $R = (y_1, \ldots, y_n)$ and $R' = (y'_1, \ldots, y'_m)$, obtaining signature $\sigma = (b, C_1, \ldots, C_n)$ in the first case and $\sigma' = (b', C'_1, \ldots, C'_m)$ in the second. The adversary then outputs the forged signature

$$\sigma^* = (b + b', C_1, \ldots, C_n, C'_1, \ldots, C'_m)$$

on M with respect to the ring $R \cup R' = (y_1, \ldots, y_n, y'_1, \ldots, y'_m)$. Applying the ring verification algorithm shows that this is indeed a valid forgery, except in the unlikely case that $C_i = C'_j$ for some i, j.

The above attack was, in fact, addressed in subsequent work of Herranz [17], where it is shown that a simple modification of the scheme (in which the ring R is included as an additional input to the hash function) is unforgeable against chosen-subring attacks. (Examination of the proof shows that the modified scheme is also secure with respect to our Definition 8, although adversarially-chosen keys are not explicitly addressed in [17].) Nevertheless, the attack on the original scheme that we have demonstrated shows that security against chosen-subring attacks is strictly stronger than security against fixed-ring attacks, and illustrates yet again the importance of rigorously formalizing desired notions of security.