A preliminary version of this paper appears in *Advances in Cryptology – EUROCRYPT 2007*, Lecture Notes in Computer Science Vol. 4515, pp. 228–245, M. Naor ed., Springer-Verlag, 2007. This is the full version.

The Power of Proofs-of-Possession: Securing Multiparty Signatures against Rogue-Key Attacks

Thomas Ristenpart* Scott Yilek†

Abstract

Multiparty signature protocols need protection against *rogue-key attacks*, made possible whenever an adversary can choose its public key(s) arbitrarily. For many schemes, provable security has only been established under the *knowledge of secret key* (KOSK) assumption where the adversary is required to reveal the secret keys it utilizes. In practice, certifying authorities rarely require the strong proofs of knowledge of secret keys required to substantiate the KOSK assumption. Instead, *proofs of possession* (POPs) are required and can be as simple as just a signature over the certificate request message. We propose a general *registered key* model, within which we can model both the KOSK assumption and in-use POP protocols. We show that simple POP protocols yield provable security of Boldyreva's multisignature scheme [11], the LOSSW multisignature scheme [28], and a 2-user ring signature scheme due to Bender, Katz, and Morselli [10]. Our results are the *first* to provide formal evidence that POPs can stop rogue-key attacks.

Keywords: Proofs of possession, PKI, multisignatures, ring signatures, bilinear maps

*Dept. of Computer Science & Engineering 0404, University of California San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0404, USA. Email: tristenp@cs.ucsd.edu. URL: http://www-cse.ucsd.edu/users/tristenp. Supported in part by NSF grant CNS-0524765.

†Dept. of Computer Science & Engineering 0404, University of California San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0404, USA. Email: syilek@cs.ucsd.edu. URL: http://www-cse.ucsd.edu/users/syilek. Supported in part by NSF grant CNS-0430595 and a Jacobs School Fellowship.

Contents

1 Introduction

We refer to any scheme that generates signatures bound to a group of parties as a *multiparty signature scheme*. We focus on schemes that are both adaptive and decentralized: the set of potential signers is dynamic and no group manager is directly involved in establishing eligibility of participants. Examples include multisignatures, ring signatures, designated-verifier signatures, and aggregate signatures. These schemes require special care against *rogue-key attacks*, which can be mounted whenever adversaries are allowed to choose their public keys arbitrarily. Typical attacks have the adversary use a public key that is a function of an honest user's key, allowing him to produce forgeries easily. Rogue-key attacks have plagued the development of multiparty signature schemes [26, 20, 22, 30, 32, 33, 25, 11, 28, 37, 31].

One method for preventing rogue-key attacks is to require, during public key registration with a certificate authority (CA), that a party proves knowledge of its secret key. This setting has typically been formalized as the *knowledge of secret key* (KOSK) assumption [11]: schemes are analyzed in a setting where adversaries must reveal their secret keys directly. This abstraction has lead to simple schemes and straightforward proofs of security. To name a few: Boldyreva's multisignature scheme [11] (we call it BMS), the LOSSW multisignature scheme [28] (we call it WMS for brevity and its basis on Waters signatures [39]), the LOSSW sequential aggregate signature scheme [28], and many designated-verifier signature schemes [23, 38, 27, 24]. Since simple rogue-key attacks against these schemes are known, it might appear that the security of these schemes actually depends on parties performing proofs of knowledge during registration.

Drawbacks of the KOSK assumption. Unfortunately, there are substantial drawbacks to using the KOSK assumption. Bellare and Neven discuss this in detail [7]; we briefly recall some of their discussion. First and foremost, the KOSK assumption is not realized by existing public key infrastructures (PKI). Registration protocols specified by the most widely used standards (RSA PKCS#10 [35], RFC 4210 [1], RFC 4211 [36]) do not specify that CA's should require proofs of knowledge. Thus, to use schemes proven secure under the KOSK assumption, one would be faced with the daunting task of upgrading existing (and already complex) PKI. This would likely require implementing clients and CA's that support zero-knowledge (ZK) proofs of knowledge that have extraction guarantees in fully concurrent settings [4]. Non-interactive ZK proofs of knowledge [17, 16, 19] could also be utilized, but these are more computationally expensive.

The plain setting. In the context of multisignatures, Bellare and Neven [7] show that it is possible to dispense with the KOSK assumption. They provide a multisignature scheme which is secure, even against rogue-key attacks, in the *plain public-key setting*, where registration with a CA ensures nothing about a party's possession or knowledge of a secret key. In this paper we are interested in something different, namely investigating the security of schemes (that are *not* secure in the plain setting) under more realistic key registration protocols, discussed next.

Proofs of possession. Although existing PKIs do not require proofs of knowledge, standards mandate the inclusion of a *proof of possession* (POP) during registration. A POP attests that a party has access to the secret key associated with his/her public key, which is typically accomplished using the functionality of the key pair's intended scheme. For signature schemes, the simplest POP has a party sign its certificate request message and send both the message and signature to the CA. The CA checks that the signature verifies under the public key being registered. In general, such proofs of possession (POPs) are clearly not sufficient for substantiating the KOSK assumption. In fact, POPs have not (previously) lead to any formal guarantees of security against rogue key attacks, even though intuitively they might appear to stop adversaries from picking arbitrary public keys. This logical gap has contributed to contention regarding the need for POPs in PKI standards [2].

Our contributions. We suggest analyzing the security of multiparty signature schemes in a registered key model, which allows modeling a variety of key registration assumptions including those based on POPs. Using the new model, we analyze the security of the BMS and WMS multisignature schemes under POP protocols. We show that, interestingly, requiring the in-use and standardized POP protocol described above *still* admits rogue-key attacks. This implies the intuition mentioned above is flawed. On the positive side, we show how a slight change to the standardized POP protocol admits proofs of security for these schemes. We also investigate the setting of ring signatures. We describe how the key registration model can be utilized to result in improved unforgeability guarantees. In particular we show that the Bender, Katz, and Morselli 2-user ring signature scheme based on Waters signatures [10] is secure against rogue-key attacks under a simple POP protocol. We now look at each of these contributions in more detail.

The registered key model. A key registration protocol is a pair of interactive algorithms (RegP, RegV), the former executed by a registrant and the latter executed by a certifying authority (CA). We lift security definitions to the registered key model by giving adversaries an additional key registration oracle, which, when invoked, executes a new instance of RegV. The security game can then restrict adversarial behavior based on whether successful registration has occurred. Security definitions in the registered key model are thus parameterized by a registration protocol. This approach allows us to straightforwardly model a variety of registration assumptions, including the KOSK assumption, the plain setting and POP-based protocols.

Multisignatures under POP. A multisignature scheme allows a set of parties to jointly generate a compact signature for some message. These schemes have numerous applications, e.g. contract signing, distribution of a certificate authority, or co-signing. The BMS and WMS schemes are simple multisignature schemes that are based directly on the short signature schemes of Boneh, Lynn, and Shacham (BLS) [14] and Waters [39]. (That is, a multisignature with group of size one is simply a BLS or Waters signature.) These schemes give short multisignatures (just 160 bits for BMS). Moreover, multisignature generation is straightforward: each party produces its BLS or Waters signature on the message, and the multisignature is just the (component-wise) product of these signatures. Both schemes fall prey to straightforward rogue-key attacks, but have proofs of security under the KOSK assumption [11, 28].

We analyze these schemes when key registration requires POPs. We show that the standardized POP mechanism described above, when applied to these schemes, *does not* lead to secure multisignatures. Both schemes fall to rogue-key attacks despite the use of the standardized POPs. We present a straightforward and natural fix for this problem: simply use separate hash functions for POPs and multisignatures. We prove the security of BMS and WMS multisignatures under such POP mechanisms, giving the first formal justification that these desirable schemes can be used in practice. Both proofs reduce to the same computational assumptions used in previous KOSK proofs and the reductions are just as tight.

Ring signatures under POP. Ring signatures allow a signer to choose a group of public keys and sign a message so that it is verifiable that some party in the group signed it, but no adversary can determine which party it was. The canonical application of ring signatures is leaking secrets [34]. Bender, Katz, and Morselli (BKM) have given a hierarchy of anonymity and unforgeability definitions for ring signature schemes [10]. For κ-user schemes, where only rings of size κ are allowed, we point out that the ability to mount rogue-key attacks (as opposed to the ability to corrupt honest parties) is a crucial distinguisher of the strength of unforgeability definitions. We introduce new security definitions that facilitate a formal analysis of this fact. BKM also propose two 2-user ring signature schemes that do not rely on random oracles, and prove them to meet the weaker unforgeability guarantee. As pointed out by Shacham and Waters, these schemes do not

meet the stronger definition due to rogue-key attacks [37].

We show that the KOSK assumption provably protects against rogue-key attacks for a natural class of ring signature schemes (both the BKM 2-user schemes fall into this class). We go on to prove the security of the BKM 2-user scheme based on Waters signatures under a simple POP-based registration protocol.

Schemes in the plain setting. We briefly overview some schemes built for the plain setting. The Micali, Ohta, and Reyzin multisignature scheme [31] was the first to be proven secure in the plain setting, but it requires a dedicated key setup phase after which the set of potential signers is necessarily static. The multisignature scheme of Bellare and Neven [7] does not require a key setup phase, and is proven secure in the plain setting. While computationally efficient, it requires several rounds of communication between all co-signers, which is more than the "non-interactive" BMS and WMS schemes.

Bender, Katz, and Morselli introduced the first ad-hoc ring signature scheme that provably resists rogue-key attacks [10]. Their scheme is not efficient, requiring semantically-secure encryption for each bit of a message. The ring signature scheme of Shacham and Waters [37] is more efficient but still not as efficient as the BKM schemes for rings of size two. Particularly, their ring signatures are at least three times as long as those given by the BKM scheme based on Waters signatures and they require more computational overhead. Of course, their solution works on rings with size greater than two.

Finally, aggregate signature schemes due to Boneh et al. [13] and Lysyanskaya et al. [29] are secure in the plain setting.

Related work and open problems. Boldyreva et al. [12] investigate certified encryption and signature schemes. They utilize a POP-based protocol to show the security of traditional certified signatures. They do not consider multiparty signatures. Many schemes beyond those treated here rely on the KOSK assumption and finding POP-based protocols for such schemes, if possible, constitutes an important set of open problems. A few examples are the LOSSW sequential aggregate signature scheme [28], the StKD encryption scheme due to Bellare, Kohno, and Shoup [5], and various designated-verifier signature schemes [23, 38, 27, 24].

2 Preliminaries

Basic notation. For a set S, we write $\mathbb{P}_\kappa(S)$ to mean the set of all subsets of size κ that exist in S. A multiset $\mathcal{S}$ is a set that allows multiple copies of elements. The size of a multiset is denoted $|\mathcal{S}|$. We write Multiset union $\mathcal{S}_1 \cup \mathcal{S}_2$ combines two multisets, including any duplicates. Thus, $|\mathcal{S}_1 \cup \mathcal{S}_2| = |\mathcal{S}_1| + |\mathcal{S}_2|$. We write $\mathcal{S} \xleftarrow{\cup} s$ to add (another copy of) s to $\mathcal{S}$, i.e. $\mathcal{S} \leftarrow \mathcal{S} \cup \{s\}$. Multiset subtraction $\mathcal{S} - \{s\}$ denotes removing one instance of s from $\mathcal{S}$. Multiset difference $\mathcal{S}_1 \backslash \mathcal{S}_2$ results in the multiset formed by repeatedly executing $\mathcal{S}_1 - \{s\}$ for each $s \in \mathcal{S}_2$ (note this is done once for each element, including duplicates). Multiset equality, containment, inequality, and intersection are defined in the natural ways. We abuse notation by using multiset notation also as boolean operators: "If $\mathcal{S}_1 \subseteq \mathcal{S}_2$ then" evaluates the statement following 'then' if $\mathcal{S}_1$ is a subset of $\mathcal{S}_2$ and does not otherwise. Other boolean multiset-operators are defined in the natural way.

We define $s \xleftarrow{\$} \mathcal{S}$ as sampling uniformly from $\mathcal{S}$ and $s \xleftarrow{\$} A(x_1, x_2, \ldots)$ assigns to s the result of running A on fresh random coins and the inputs $x_1, x_2, \ldots$. We denote concatenation of strings M and M' by $M \,||\, M'$. For any string M, let $M[i]$ denote the i^{th} bit of M. We often write $(x_i)_1^\eta$ to denote a vector of η values: $(x_1, \ldots, x_\eta)$. For a table $\mathtt{H_s}$, let $\mathtt{H_s}[s]$ denote the element associated with s. We write $\mathsf{Time}(A) = \max\{t_1, t_2, \ldots\}$ where $A = (A_1, A_2, \ldots)$ is a tuple of algorithms and

$t_1, t_2, \ldots$ are their respective worst case running times.

BILINEAR MAPS AND CO-CDH. The schemes we consider use bilinear maps. Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be groups, each of prime order p. Then $\mathbb{G}_1^*$, $\mathbb{G}_2^*$, and $\mathbb{G}_T^*$ represent the set of all generators of the groups (respectively). Let $\mathbf{e}$: $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be an efficiently computable bilinear map (also called a pairing). For the multisignature schemes we consider, we use the asymmetric setting [14, 13] where $\mathbb{G}_1 \neq \mathbb{G}_2$ and there exists an efficiently computable isomorphism ψ: $\mathbb{G}_2 \to \mathbb{G}_1$. The asymmetry allows for short signatures, while ψ is needed in the proofs. For the ring signature schemes we consider, we instead use the symmetric setting [10, 37] where $\mathbb{G}_1 = \mathbb{G}_2$. Let n represent the number of bits needed to encode an element of $\mathbb{G}_1$; for the asymmetric setting n is typically 160. Finally let g be a generator in $\mathbb{G}_2$. For the rest of the paper we treat $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathbf{e}$ as fixed, globally known parameters. Then we define the advantage of an algorithm A in solving the Computational co-Diffie-Hellman (co-CDH) problem in the groups $(\mathbb{G}_1, \mathbb{G}_2)$ as

$$\mathbf{Adv}^{\text{co-cdh}}_{(\mathbb{G}_1,\mathbb{G}_2)}(A) = \Pr\left[A(g, g^x, h) = h^x : \; x \xleftarrow{\$} \mathbb{Z}_p; h \xleftarrow{\$} \mathbb{G}_1\right]$$

where the probability is over the random choices of x and h and the coins used by A. Here $\mathbb{Z}_p$ is the set of integers modulo p. Note that in the symmetric setting this is just the CDH problem.

For a group element g, we write $\langle g \rangle$ to mean some canonical encoding of g as a bit string of the appropriate length. We write $\langle g \rangle_n$ to mean the first n bits of $\langle g \rangle$. We use the shorthand $\vec{u}$ (resp. $\vec{w}$) to mean a list of group elements $u_1, \ldots, u_n$ (resp. $w_1, \ldots, w_n$). Let t_E, t_ψ, and $t_{\mathbf{e}}$ be the maximum times to compute an exponentiation in $\mathbb{G}_1$, compute ψ on an element in $\mathbb{G}_2$, and compute the pairing.

SIGNATURE SCHEMES. A signature scheme $\mathsf{S} = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Ver})$ consists of an parameter generation algorithm, a key generation algorithm, a signing algorithm that outputs a signature given a secret key and a message, and a verification algorithm that outputs a bit given a public key, message, and signature. If Pg is omitted, this means there are no public parameters requiring trusted setup; in this case the implicit Pg algorithm just outputs an empty string. We require that $\mathsf{Ver}(pk, M, \mathsf{Sign}(sk, M)) = 1$ for all allowed M and valid pk, sk. Following [18], we define the advantage of an adversary A in forging against S in a chosen message attack as

$$\mathbf{Adv}^{\text{uf}}_{\mathsf{S}}(A) = \Pr\left[\mathsf{Ver}(pk, M, \sigma) = 1 : \begin{array}{l}(pk, sk) \xleftarrow{\$} \mathsf{Kg}; \\ (M, \sigma) \xleftarrow{\$} A^{\mathsf{Sign}(sk,\cdot)}(pk)\end{array}\right]$$

where the probability is over the coins used by Kg, Sign, and A. We only consider adversaries that output forgeries on a message M which was not queried to the oracle.

3 The Registered Key Model

KEY REGISTRATION PROTOCOLS. Let $\mathcal{P}$ and $\mathcal{S}$ be sets and $\mathcal{K} \subseteq \mathcal{P} \times \mathcal{S}$ be a relation on the sets (representing public keys, secret keys, and valid key pairs, respectively). A *key registration protocol* is a pair of interactive algorithms $(\mathsf{RegP}, \mathsf{RegV})$. A party registering a key runs RegP with inputs $pk \in \mathcal{P}$ and $sk \in \mathcal{S}$. A certifying authority (CA) runs RegV. We restrict our attention (without loss of generality) to protocols in which the last message is from RegV to RegP and contains either a $pk \in \mathcal{P}$ or a distinguished symbol $\perp$. We require that running $\mathsf{RegP}(pk, sk)$ with RegV results in RegV's final message being pk whenever $(pk, sk) \in \mathcal{K}$.

We give several examples of key registration protocols. The plain registration protocol $\mathsf{Plain} = (\mathsf{PlainP}, \mathsf{PlainV})$ has the registrant running $\mathsf{PlainP}(pk, sk)$ send pk to the CA. The CA running

PlainV, upon receiving a public key pk, simply replies with pk. This protocol will be used to capture the plain model, where no checks on public keys are performed by a CA. To model the KOSK assumption, we specify the registration protocol $\mathsf{Kosk} = (\mathsf{KoskP}, \mathsf{KoskV})$. Here $\mathsf{KoskP}(pk, sk)$ sends (pk, sk) to the CA. Upon receiving (pk, sk), the KoskV algorithm checks that $(pk, sk) \in \mathcal{K}$. (We assume that such a check is efficiently computable; this is the case for key pairs we consider.) If so, it replies with pk and otherwise with $\perp$.

We refer to registration protocols that utilize the key's intended functionality as proof-of-possession based. For example, let $\mathsf{S} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Ver})$ be a signature scheme. Define the registration protocol $\mathsf{S\text{-}Pop} = (\mathsf{PopP}, \mathsf{PopV})$ as follows. Running PopP on inputs pk, sk results in sending the message $pk \,||\, \mathsf{Sign}(sk, \langle pk \rangle)$ to the CA. Upon receiving message $pk \,||\, \sigma$, a CA running PopV replies with pk if $\mathsf{Ver}(pk, \langle pk \rangle, \sigma) = 1$ and otherwise replies with $\perp$. This corresponds to the simplest POPs for signature schemes specified in PKCS#10 and RFCs 4210/4211.

The registered key model. We consider security definitions that are captured by a game between an adversary and an environment. To lift such security definitions to the *registered key model*, we use the following general approach. Adversaries are given an additional key registration oracle **OKReg** that, once invoked, runs a new instance of RegV for some key registration protocol $(\mathsf{RegP}, \mathsf{RegV})$. If the last message from RegV is a public key pk, then pk is added to a table $\mathcal{R}$. This table can now be used to modify winning conditions or restrict which public keys are utilized by the adversary in interactions with the environment. Security of schemes under the new definition is therefore always with respect to some registration protocol.

The key registration protocols mentioned so far are *two round protocols*: the registrant sends a first message to the CA, which replies with a second message being either pk or $\perp$. For any two round protocol $\mathsf{Reg} = (\mathsf{RegP}, \mathsf{RegV})$, the **OKReg** oracle can be simplified as follows. An adversary queries with a first message, at which point RegP is immediately run and supplied with the message. The oracle halts RegP before it sends its reply message. The message is added to $\mathcal{R}$ if it is not $\perp$. The oracle finally returns pk or $\perp$ appropriately.

4 Multisignatures using POPs

The goal of a multisignature scheme is for a group of parties, each with its own public and secret keys, to jointly create a compact signature on some message. Following the formulation in [7], a *multisignature scheme* is a tuple of algorithms $\mathsf{MS} = (\mathsf{MPg}, \mathsf{MKg}, \mathsf{MSign}, \mathsf{MVf})$. A trusted central authority runs the parameter generation algorithm MPg to create a parameter string *par* that is published globally. The key generation algorithm MKg, independently run by each party, outputs a key pair (pk, sk). The MSign interactive protocol is run by some group of players. Each party locally runs MSign on input being a secret key sk, a multiset of public keys $\mathcal{V}$, and a message M. It may consist of multiple rounds, though the protocols we consider here only require a single round: a request broadcast to all parties and the response(s). Finally, the verification algorithm MVf takes as input a tuple $(\mathcal{V}, M, \sigma)$, where $\mathcal{V}$ is a multiset of public keys, M is a message, and σ is a signature, and returns a bit. We require that $\mathsf{MVf}(\mathcal{V}, M, \mathsf{MSign}(sk, \mathcal{V}, M)) = 1$ for any M and where every participant correctly follows the algorithms. For schemes that do not require any trusted setup, we write the scheme as just a triple of algorithms $(\mathsf{MKg}, \mathsf{MSign}, \mathsf{MVf})$ where the now implicit MPg algorithm returns an empty string.

Multisignature security. Let $\mathsf{MS} = (\mathsf{MPg}, \mathsf{MKg}, \mathsf{MSign}, \mathsf{MVf})$ be a multisignature scheme, $\mathsf{Reg} = (\mathsf{RegP}, \mathsf{RegV})$ be a key registration protocol, and A be an adversary. Figure 1 displays the security game $\mathbf{Exp}^{\text{msuf-kr}}_{\mathsf{MS},\mathsf{Reg}}(A)$. The experiment simulates one honest player with public key pk^*. The goal of the adversary is to produce a *multisignature forgery*: a tuple $(\mathcal{V}, M, \sigma)$ that

Experiment $\mathbf{Exp}^{\text{msuf-kr}}_{\mathsf{MS},\mathsf{Reg}}(A)$

$par \xleftarrow{\$} \mathsf{MPg}$; $(pk^*, sk^*) \xleftarrow{\$} \mathsf{MKg}(par)$; $\mathcal{Q} \leftarrow \emptyset$; $\mathcal{R} \leftarrow \emptyset$

Run $A(par, pk^*)$ handling oracle queries as follows

OMSign$(\mathcal{V}, M)$, where $pk^* \in \mathcal{V}$: $\mathcal{Q} \xleftarrow{\cup} M$; Simulate a new instance of $\mathsf{MSign}(sk^*, \mathcal{V}, M)$, forwarding messages to and from A appropriately.

OKReg: Simulate a new instance of algorithm RegV, forwarding messages to and from A. If the instance's final message is $pk \neq \perp$, then $\mathcal{R} \xleftarrow{\cup} pk$.

A halts with output $(\mathcal{V}, M, \sigma)$

If $(\, pk^* \in \mathcal{V} \,) \wedge (\, M \notin \mathcal{Q} \,) \wedge (\, \mathsf{MVf}(\mathcal{V}, M, \sigma) = 1 \,) \wedge (\, (\mathcal{V} - \{pk^*\}) \setminus \mathcal{R} = \emptyset \,)$ then

  Return 1

Return 0

Figure 1: Multisignature security experiment in the registered key model.

satisfies the following four conditions. First, the honest public key pk^* is in the multiset $\mathcal{V}$ at least once. Second, the message M was not queried to the multisignature oracle. Third, the signature verifies. Fourth, each public key in $\mathcal{V} - \{pk^*\}$ must be in $\mathcal{R}$, where $\mathcal{V} - \{pk^*\}$ means the multiset $\mathcal{V}$ with *one* occurrence of the honest key removed. We define the msuf-kr-advantage of an adversary A against a multisignature scheme MS with respect to registration protocol Reg as $\mathbf{Adv}^{\text{msuf-kr}}_{\mathsf{MS},\mathsf{Reg}}(A) = \Pr\left[\mathbf{Exp}^{\text{msuf-kr}}_{\mathsf{MS},\mathsf{Reg}}(A) \Rightarrow 1\right]$. The probability is taken over the random coins used in the course of running the experiment, including those used by A. The definitions can be lifted to the random oracle model [8] in the natural way. It is easy to show that our definition is equivalent to the definition in [7] when $\mathsf{Reg} = \mathsf{Plain}$ and equivalent to the definition in [11] when $\mathsf{Reg} = \mathsf{Kosk}$.

An adversary A is *legitimate* if: (1) it only queries **OMSign** on a set of public keys $\mathcal{V}$ such that $pk^* \in \mathcal{V}$; and (2) outputs a forgery attempt $(\mathcal{V}, M, \sigma)$ for which $pk^* \in \mathcal{V}$ and $M \notin \mathcal{Q}$ and $(\mathcal{V} - \{pk^*\}) \setminus \mathcal{R} = \emptyset$ (all adversarially-chosen keys have been previously registered). Without loss of generality, we will from now on only consider legitimate adversaries.

We now prove the security of the BMS and WMS multisignature schemes relative to POP-based protocols that differ from current standards only by use of a distinct hash function. In Section 4.3 we discuss attacks against the schemes when standardized registration protocols are utilized.

4.1 Multisignatures Based on BLS Signatures

BLS SIGNATURES AND MULTISIGNATURES. Let H_s: $\{0,1\}^* \to \mathbb{G}_1$ be a random oracle and let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathbf{e}$ be pairing parameters. Boneh, Lynn, and Shacham [14] specify a signature scheme $\mathsf{BLS} = (\mathsf{B\text{-}Kg}, \mathsf{B\text{-}Sign}, \mathsf{B\text{-}Vf})$. The algorithms work as follows:

$\mathsf{B\text{-}Kg}$:	$\mathsf{B\text{-}Sign}^{H_s}(sk, M)$:	$\mathsf{B\text{-}Vf}^{H_s}(pk, M, \sigma)$:
$sk \xleftarrow{\$} \mathbb{Z}_p$; $pk \leftarrow g^{sk}$	Return $H_s(M)^{sk}$	If $\mathbf{e}(H_s(M), pk) = \mathbf{e}(\sigma, g)$ then
Return (pk, sk)		Return 1
		Return 0

The $\mathsf{BMS} = (\mathsf{B\text{-}MKg}, \mathsf{B\text{-}MSign}, \mathsf{B\text{-}MVf})$ multisignature scheme [11] is a simple extension of BLS signatures. Key generation is equivalent to $\mathsf{B\text{-}Kg}$. Multisignature generation for participants labeled $1, \ldots, v$, public keys $\mathcal{V} = \{pk_1, \ldots, pk_v\}$, and a message M proceeds as follows. Each participant i computes $\sigma_i \xleftarrow{\$} \mathsf{B\text{-}Sign}(sk_i, M)$ and broadcasts σ_i to all other participants. The multisignature is $\sigma \leftarrow \prod_{i=1}^{v} \sigma_i$. On input $\mathcal{V}, M, \sigma$ for $\mathcal{V} = \{pk_1, \ldots, pk_v\}$ the verification algorithm $\mathsf{B\text{-}MVf}$ computes

$PK = \prod_{i=1}^{v} pk_i$ and then runs $\mathsf{B\text{-}Vf}^{H_s}(PK, M, \sigma)$, returning its output. Boldyreva proved the scheme secure under the KOSK assumption [11].

THE B-Pop PROTOCOL. We now specify a POP-based key registration protocol under which we can prove BMS secure. Let H_p: $\{0,1\}^* \to \mathbb{G}_1$ be a random oracle. Then we define the $\mathsf{B\text{-}Pop} = (\mathsf{B\text{-}PopP}, \mathsf{B\text{-}PopV})$ protocol as follows. Algorithm $\mathsf{B\text{-}PopP}(pk, sk)$ sends $pk \,||\, \mathsf{B\text{-}Sign}^{H_p}(sk, \langle pk \rangle)$ and algorithm B-PopV, upon receiving (pk, π) computes $\mathsf{B\text{-}Vf}^{H_p}(pk, \langle pk \rangle, \pi)$ and if the result is 1 replies with pk and otherwise with $\perp$. We point out that one can use the same random oracle (and underlying instantiating hash function) for both H_s and H_p as long as domain separation is enforced. The following theorem captures the security of BMS with respect to this key registration protocol.

Theorem 4.1 *Let H_s, H_p: $\{0,1\}^* \to \mathbb{G}_1$ be random oracles. Let A be an* msuf-kr*-adversary, with respect to the* B-Pop *propocol, that runs in time t, makes q_h, q_{pop}, q_s, and q_k queries to H_s, H_p, the signing oracle, and the key registration oracle, and outputs a multisignature forgery on a group of size at most v. Then there exists an adversary B such that*

$$\mathbf{Adv}^{\text{msuf-kr}}_{\mathsf{BMS},\mathsf{B\text{-}Pop}}(A) \leq e(q_s + 1) \cdot \mathbf{Adv}^{\text{co-cdh}}_{(\mathbb{G}_1,\mathbb{G}_2)}(B)$$

where B runs in time at most $t + \mathcal{O}((q_h + q_{\text{pop}} + v)t_E + t_{\mathbf{e}})$.

Proof: We construct a co-CDH adversary B, which on input $X \in \mathbb{G}_2$ and $h \in \mathbb{G}_1$ utilizes a legitimate msuf-kr adversary A (with respect to the B-Pop protocol) to help it compute h^x where $x = \log_g X$. We adapt a game-playing [9] approach due to Bellare for proving the security of BLS signatures [3]. Recall that since BMS is a single-round protocol the signing oracle simply returns a BLS signature over the message. We make several simplifying assumptions about A that are without loss of generality. First, A always queries $H_s(M)$ before querying $\mathbf{OMSign}(M)$ for any message M. Second, A always queries $H_p(\langle pk \rangle)$ before querying $\mathbf{OKReg}(pk, \pi)$ for any π. Third, A only queries **OKReg** on pairs (pk, π) for which $\mathsf{B\text{-}Vf}^{H_p}(pk, \langle pk \rangle, \pi) = 1$.

Figure 2 details the adversary B. It first applies the isomorphism ψ to g and X and sets a counter c to zero. It runs A on input X, which is then the trusted public key of the BMS instance A is to attack. Adversary B uses four tables, $\mathtt{H_s}$, $\mathtt{H_p}$, $\mathtt{B}$, and $\mathtt{P}$. All four are initially everywhere undefined. Queries to H_s are sometimes responded to with values that include h and sometimes not, depending on a δ-biased coin (we notate flipping such a coin by $\delta_c \xleftarrow{\delta} \{0,1\}$). Intuitively, the δ_c values correspond to guessing which H_s query will correspond to the forgery message. Adversary B responds to H_p queries with values that always include h. This is so that the adversarially-supplied POPs can be used to help extract the co-CDH solution from a forgery. Queries to **OKReg** simply record the supplied POP in $\mathtt{P}$. Eventually A halts outputing a forgery attempt $(\mathcal{V}, M, \sigma)$. Adversary B checks if the forgery verifies and if so attempts to extract from it the value h^x. Here $(X, pk_1, \ldots, pk_d) \leftarrow \mathcal{V}$ means parse $\mathcal{V}$ into the sequence of public keys $X, pk_1, \ldots, pk_d$ which enumerates all those in $\mathcal{V}$. (Since A is legitimate X must be in $\mathcal{V}$.) Note that extraction of h^x is only possible if the forgery message had its H_s response programmed to include h.

We now proceed through a sequence of games to lower bound the probability that B succeeds in computing h^x. The first game is G0, shown in Figure 2 with boxed statements omitted. It is exactly like B except that it generates the co-CDH instance itself (first line of **Initialize**). The outputs $\text{G0}(A)$ and B are thus distributed identically. More-over, if their output is not $\perp$ then it is h^x. To see this, let us fix some notation related to the variables in the **Finalize** procedure of G0. Define $sk_i = \log_g pk_i$ and $\pi_i = \mathtt{P}[pk_i]$ for each $i \in [1\,..\,d]$. Then $\psi(pk_i) = g_1^{sk_i}$ holds for each i. Define

Adversary $B(X, h)$

$g_1 \leftarrow \psi(g)$; $X_1 \leftarrow \psi(X)$; $c \leftarrow 0$
Run $A(X)$, answering queries by

query $H_s(M)$
$c \leftarrow c + 1$; $M_c \leftarrow M$
$\alpha_c \xleftarrow{\$} \mathbb{Z}_p$; $\delta_c \xleftarrow{\delta} \{0, 1\}$
If $\delta_c = 1$ then $\mathtt{H_s}[M] \leftarrow g_1^{\alpha_c}$
Else $\mathtt{H_s}[M] \leftarrow h g_1^{\alpha_c}$
Ret $\mathtt{H_s}[M]$

query $H_p(N)$
$\mathtt{B}[N] \xleftarrow{\$} \mathbb{Z}_p$; Ret $\mathtt{H_p}[N] \leftarrow h g_1^{\mathtt{B}[N]}$

query OMSign(M)
Let k be s.t. $M = M_k$; $S_k \leftarrow \bot$
If $\delta_k = 1$ then $S_k \leftarrow X_1^{\alpha_k}$
Else $bad \leftarrow \mathsf{true}$
Ret S_k

query OKReg(pk, π)
$\mathtt{P}[pk] \leftarrow \pi$; Ret pk

A finishes, outputing $(\mathcal{V}, M, \sigma)$
If $\mathsf{B\text{-}MVf}(\mathcal{V}, M, \sigma) = 0$ then Ret $\bot$
$(X, pk_1, \ldots, pk_d) \leftarrow \mathcal{V}$
Let k be s.t. $M = M_k$; $\alpha \leftarrow \alpha_k$
For each $i \in [1 \,..\, d]$ do $\gamma_i \leftarrow \alpha - \mathtt{B}[\langle pk_i \rangle]$
$w \leftarrow \bot$
If $\delta_k = 0$ then
  $w \leftarrow \sigma X_1^{-\alpha} \prod_{i=1}^{d} (\mathtt{P}[pk_i]^{-1} \psi(pk_i)^{-\gamma_i})$
Else $bad \leftarrow \mathsf{true}$
Ret w

procedure Initialize G0 $\boxed{\text{G1}}$
$x \xleftarrow{\$} \mathbb{Z}_p$; $X \leftarrow g^x$; $h \xleftarrow{\$} \mathbb{G}_1$
$g_1 \leftarrow \psi(g)$; $X_1 \leftarrow \psi(X)$; $c \leftarrow 0$
Ret X

procedure $H_s(M)$
$c \leftarrow c + 1$; $M_c \leftarrow M$
$\alpha_c \xleftarrow{\$} \mathbb{Z}_p$; $\delta_c \xleftarrow{\delta} \{0, 1\}$
If $\delta_c = 1$ then $\mathtt{H_s}[M] \leftarrow g_1^{\alpha_c}$
Else $\mathtt{H_s}[M] \leftarrow h g_1^{\alpha_c}$
Ret $\mathtt{H_s}[M]$

procedure $H_p(N)$
$\mathtt{B}[N] \xleftarrow{\$} \mathbb{Z}_p$; Ret $\mathtt{H_p}[N] \leftarrow h g_1^{\mathtt{B}[N]}$

procedure OMSign(M)
Let k be s.t. $M = M_k$; $S_k \leftarrow \bot$
If $\delta_k = 1$ then $S_k \leftarrow X_1^{\alpha_k}$
Else $bad \leftarrow \mathsf{true}$ $\boxed{; S_k \leftarrow \mathtt{H_s}[M]^x}$
Ret S_k

procedure OKReg(pk, π)
$\mathtt{P}[pk] \leftarrow \pi$; Ret pk

procedure Finalize$(\mathcal{V}, M, \sigma)$
If $\mathsf{B\text{-}MVf}(\mathcal{V}, M, \sigma) = 0$ then Ret $\bot$
$(X, pk_1, \ldots, pk_d) \leftarrow \mathcal{V}$
Let k be s.t. $M = M_k$; $\alpha \leftarrow \alpha_k$
For each $i \in [1 \,..\, d]$ do $\gamma_i \leftarrow \alpha - \mathtt{B}[\langle pk_i \rangle]$
$w \leftarrow \bot$
If $\delta_k = 0$ then
  $w \leftarrow \sigma X_1^{-\alpha} \prod_{i=1}^{d} (\mathtt{P}[pk_i]^{-1} \psi(pk_i)^{-\gamma_i})$
Else $bad \leftarrow \mathsf{true}$ $\boxed{; w \leftarrow h^x}$
Ret w

Figure 2: Adversary B and games G0 and G1 used in the proof of Theorem 4.1. Game G0 omits the boxed statements while G1 includes them.

$\beta_i = \alpha - \gamma_i = \mathtt{B}[pk_i]$. Because A is legitimate we necessarily have that $\mathbf{e}(\mathtt{H_s}[M], PK) = \mathbf{e}(\sigma, g)$ and that $\mathbf{e}(\mathtt{H_p}[\langle pk_i \rangle], pk_i) = \mathbf{e}(\pi_i, g)$ for each $i \in [1 \,..\, d]$. Applying the properties of pairings to these equivalences gives that $\sigma = (h g_1^{\alpha})^{x + sk_1 + \ldots + sk_d}$ and $\pi_i = (h g_1^{\beta_i})^{sk_i}$ for each $i \in [1 \,..\, d]$. Then

$$w = \sigma X_1^{-\alpha} \prod_{i=1}^{d} \pi_i^{-1} \psi(pk_i)^{-\gamma_i} = \frac{(h g_1^{\alpha})^{x + sk_1 + \ldots + sk_d}}{X_1^{\alpha} \cdot \prod (h g_1^{\beta_i})^{sk_i} (g_1^{sk_i})^{\alpha - \beta_i}} = h^x \,. \tag{1}$$

Therefore we have that

$$\mathbf{Adv}^{\text{co-cdh}}_{(\mathbb{G}_1, \mathbb{G}_2)}(B) = \Pr\left[\text{G0}(A) \Rightarrow \bot\right] \tag{2}$$

The next game is G1, shown in Figure 2 with the boxed statements included. Games G0 and G1

procedure Initialize G2

$x \xleftarrow{\$} \mathbb{Z}_p$; $X \leftarrow g^x$; $h \xleftarrow{\$} \mathbb{G}_1$; $c \leftarrow 0$
Ret X

procedure $H_s(M)$

$c \leftarrow c+1$; $M_c \leftarrow M$
$\alpha_c \xleftarrow{\$} \mathbb{Z}_p$; $\delta_c \xleftarrow{\delta} \{0,1\}$
If $\delta_c = 1$ then $\mathtt{H_s}[M] \leftarrow g_1^{\alpha_c}$
Else $\mathtt{H_s}[M] \leftarrow h g_1^{\alpha_c}$
Ret $\mathtt{H_s}[M]$

procedure $H_p(N)$

$\mathtt{B}[N] \xleftarrow{\$} \mathbb{Z}_p$; Ret $\mathtt{H_p}[N] \leftarrow h g_1^{\mathtt{B}[N]}$

procedure OMSign(M)

Let k be s.t. $M = M_k$
If $\delta_k \neq 1$ then $bad \leftarrow \mathsf{true}$
Ret $\mathtt{H_s}[M]^x$

procedure OKReg(pk, π)

$\mathtt{P}[pk] \leftarrow \pi$; Ret pk

procedure Finalize$(\mathcal{V}, M, \sigma)$

If $\mathsf{B\text{-}MVf}(\mathcal{V}, M, \sigma) = 0$ then Ret $\perp$
Let k be s.t. $M = M_k$
If $\delta_k \neq 0$ then $bad \leftarrow \mathsf{true}$
Ret h^x

procedure Initialize G3

$x \xleftarrow{\$} \mathbb{Z}_p$; $X \leftarrow g^x$; $h \xleftarrow{\$} \mathbb{G}_1$; $c \leftarrow 0$
Ret X

procedure $H_s(M)$

$c \leftarrow c+1$; $M_c \leftarrow M$; $\delta_c \xleftarrow{\delta} \{0,1\}$
Ret $\mathtt{H_s}[M] \xleftarrow{\$} \mathbb{G}_1$

procedure $H_p(N)$

Ret $\mathtt{H_p}[N] \xleftarrow{\$} \mathbb{G}_1$

procedure OMSign(M)

Let k be s.t. $M = M_k$
If $\delta_k \neq 1$ then $bad \leftarrow \mathsf{true}$
Ret $\mathtt{H_s}[M]^x$

procedure OKReg(pk, π)

$\mathtt{P}[pk] \leftarrow \pi$; Ret pk

procedure Finalize$(\mathcal{V}, M, \sigma)$

If $\mathsf{B\text{-}MVf}(\mathcal{V}, M, \sigma) = 0$ then Ret $\perp$
Let k be s.t. $M = M_k$
If $\delta_k \neq 0$ then $bad \leftarrow \mathsf{true}$
Ret h^x

procedure Initialize G4

$x \xleftarrow{\$} \mathbb{Z}_p$; $X \leftarrow g^x$; $h \xleftarrow{\$} \mathbb{G}_1$; $c \leftarrow 0$
Ret X

procedure $H_s(M)$

Ret $\mathtt{H_s}[M] \xleftarrow{\$} \mathbb{G}_1$

procedure $H_p(N)$

Ret $\mathtt{H_p}[N] \xleftarrow{\$} \mathbb{G}_1$

procedure OMSign(M)

$c \leftarrow c+1$; Ret $\mathtt{H_s}[M]^x$

procedure OKReg(pk, π)

$\mathtt{P}[pk] \leftarrow \pi$; Ret pk

procedure Finalize$(\mathcal{V}, M, \sigma)$

If $\mathsf{B\text{-}MVf}(\mathcal{V}, M, \sigma) = 0$ then Ret $\perp$
For each $j \in [1\,..\,c]$ do
  $\delta_j \xleftarrow{\delta} \{0,1\}$; If $\delta_j = 0$ then $bad \leftarrow \mathsf{true}$
$\delta_{j+1} \xleftarrow{\delta} \{0,1\}$; If $\delta_{j+1} = 1$ then $bad \leftarrow \mathsf{true}$
Ret h^x

Figure 3: Games G2 and G3 used in the proof of Theorem 4.1.

are identical-until-bad. Let Good_i be the event that G$i(A)$ does not set bad. Then by a variant [6] of the fundamental lemma of game-playing [9] we have that

$$\Pr\left[\mathrm{G0}(A) \not\Rightarrow \bot\right] \geq \Pr\left[\mathrm{G0}(A) \not\Rightarrow \bot \wedge \mathsf{Good}_0\right] = \Pr\left[\mathrm{G1}(A) \not\Rightarrow \bot \wedge \mathsf{Good}_1\right]. \tag{3}$$

In game G1 note that signing queries always return $H_s[M]^x$. This is true because

$$X_1^{\alpha_k} = \psi(g^x)^{\alpha_k} = (g_1^x)^{\alpha_k} = (g_1^{\alpha_k})^x = H_s[M]^x. \tag{4}$$

Additionally **Finalize**, as per (1), always returns h^x if A's forgery verifies. Game G2 rewrites **OMSign** and **Finalize** to make these facts explicit, and also omits computing g_1 and X_1 which are no longer needed. We have that

$$\Pr\left[\mathrm{G1}(A) \not\Rightarrow \bot \wedge \mathsf{Good}_1\right] = \Pr\left[\mathrm{G2}(A) \not\Rightarrow \bot \wedge \mathsf{Good}_2\right]. \tag{5}$$

In game G2 all uses of hash range points are via $\mathtt{H_s}[M]$ and $\mathtt{H_p}[M]$ (i.e., we no longer utilize the special structure of the range points). Game G3 therefore dispenses with the special way in which H_s and H_p choose range points, now choosing them directly at random from $\mathbb{G}_1$. In both G2 and G3 the range points have the same (uniform) distribution, meaning that the code is equivalent and thus

$$\Pr\left[\mathrm{G2}(A) \not\Rightarrow \bot \wedge \mathsf{Good}_2\right] = \Pr\left[\mathrm{G3}(A) \not\Rightarrow \bot \wedge \mathsf{Good}_3\right]. \tag{6}$$

In G3 the setting of bad does not impact any other random variables in the game, and so in game G4 we defer its setting until **Finalize**. Furthermore, not all of the δ values can actually set bad: only those that end up being referenced during signing queries and the one extra for the forgery. Thus G4 moves incrementing the counter c to signing queries, and in **Finalize** only performs δ-biased coin tosses $c+1$ times: one for each signature query and one for the forgery. This justifies that

$$\Pr\left[\mathrm{G3}(A) \not\Rightarrow \bot \wedge \mathsf{Good}_3\right] = \Pr\left[\mathrm{G4}(A) \not\Rightarrow \bot \wedge \mathsf{Good}_4\right]. \tag{7}$$

In game G4, the event G4$(A) \not\Rightarrow \bot$ and the event Good_4 are independent. This is because the events do not rely on any shared random variables. The variable c is in fact not a random variable, because it will always equal q_s, the number of signing queries A uses. This justifies that

$$\Pr\left[\mathrm{G4}(A) \not\Rightarrow \bot \wedge \mathsf{Good}_4\right] = \Pr\left[\mathrm{G4}(A) \not\Rightarrow \bot\right] \cdot \Pr\left[\mathsf{Good}_4\right]. \tag{8}$$

where $\Pr\left[\mathsf{Good}_4\right] \geq (e(q_s+1))^{-1}$ (this is standard, see e.g. [6, 15]). Finally, G4 exactly simulates for A the ruf3-kr experiment, meaning

$$\Pr\left[\mathrm{G4}(A) \not\Rightarrow \bot\right] = \Pr\left[\mathbf{Exp}^{\text{ruf3-kr}}_{\mathsf{BMS},\mathsf{B\text{-}Pop}}(A) \Rightarrow 1\right] = \mathbf{Adv}^{\text{ruf3-kr}}_{\mathsf{BMS},\mathsf{B\text{-}Pop}}(A) \tag{9}$$

Combining (2), (3), (5), (6), (7), (8), (9), and the lower bound on $\Pr\left[\mathsf{Good}_4\right]$ yields

$$\mathbf{Adv}^{\text{co-cdh}}_{(\mathbb{G}_1,\mathbb{G}_2)}(B) \geq \frac{1}{e(q_s+1)} \mathbf{Adv}^{\text{ruf3-kr}}_{\mathsf{BMS},\mathsf{B\text{-}Pop}}(A)$$

which implies the theorem statement.

Adversary B runs A. Additionally B must perform an exponentiation for each H_s and H_p query and one for each key in the forgery set $\mathcal{V}$. Finally B must perform a pairing to verify the forgery. Thus B runs in time at most $t + \mathcal{O}((q_h + q_{\text{pop}} + v)t_E + t_{\mathbf{e}})$ where $|\mathcal{V}| = v$. ∎

4.2 Multisignatures Based on Waters Signatures

Waters signatures and multisignatures. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathbf{e}$ be pairing parameters. We define the signature scheme $\mathsf{W} = (\mathsf{W\text{-}Pg}, \mathsf{W\text{-}Kg}, \mathsf{W\text{-}Sign}, \mathsf{W\text{-}Vf})$, as given in [28] and which is a slight variant of the original Waters signature scheme in [39]. The parameter generation algorithm $\mathsf{W\text{-}Pg}$ selects values $h, u, u_1, \ldots, u_n \xleftarrow{\$} \mathbb{G}_1^{n+2}$ and publishes them globally. The other algorithms are defined below.

W-Kg	$\mathsf{W\text{-}Sign}^{H_{u,\vec{u}}}(sk, M)$	$\mathsf{W\text{-}Vf}^{H_{u,\vec{u}}}(pk, M, (\sigma, \rho))$
$\alpha \xleftarrow{\$} \mathbb{Z}_p$	$r \xleftarrow{\$} \mathbb{Z}_p$; $\rho \leftarrow g^r$	If $\mathbf{e}(h, pk) \cdot \mathbf{e}(H_{u,\vec{u}}(M), \rho) = \mathbf{e}(\sigma, g)$ then
$pk \leftarrow g^\alpha$; $sk \leftarrow h^\alpha$	$\sigma \leftarrow sk \cdot H_{u,\vec{u}}(M)^r$	Return 1
Return (pk, sk)	Return (σ, ρ)	Return 0

The function $H_{u,\vec{u}}$: $\{0,1\}^n \to \mathbb{G}_1$ is defined by $H_{u,\vec{u}}(M) = u \cdot \prod_{i=1}^n u_i^{M[i]}$ for any $u, \vec{u} \in \mathbb{G}_1^{n+1}$. For simplicity the message space as defined is restricted to $\{0,1\}^n$, but in practice one can use a collision-resistant hash function to expand the domain.

The $\mathsf{WMS} = (\mathsf{W\text{-}Pg}, \mathsf{W\text{-}MKg}, \mathsf{W\text{-}MSign}, \mathsf{W\text{-}MVf})$ multisignature scheme [28] is a straightforward extension of the Waters' signature scheme. Parameter generation and key generation are the same as in W. To generate a multisignature for multiset $\mathcal{V} = \{pk_1, \ldots, pk_v\}$, each participant i computes $(\sigma_i, \rho_i) \xleftarrow{\$} \mathsf{W\text{-}Sign}^{H_{u,\vec{u}}}(sk_i, M)$ and broadcasts (σ_i, ρ_i). The multisignature is $(\prod_{i=1}^v \sigma_i, \prod_{i=1}^v \rho_i)$. To verify a multisignature (σ, ρ) for a message M and public keys $\mathcal{V} = \{pk_1, \ldots, pk_v\}$, simply let $PK \leftarrow \prod_{i=1}^v pk_i$ and then return $\mathsf{W\text{-}Vf}^{H_{u,\vec{u}}}(PK, M, (\sigma, \rho))$. This scheme was proven secure using the KOSK assumption in [28].

THE $\mathsf{WM\text{-}Pop}$ PROTOCOL. Let $w, w_1, \ldots, w_n \xleftarrow{\$} \mathbb{G}_1^{n+1}$ be global parameters with associated hash function $H_{w,\vec{w}}$. These parameters require trusted setup, particularly because the CA should *not* know their discrete logs. (One might therefore have the trusted party that runs $\mathsf{W\text{-}Pg}$ also generate $w, \vec{w}$.) We define the following key registration protocol $\mathsf{WM\text{-}Pop} = (\mathsf{WM\text{-}PopP}, \mathsf{WM\text{-}PopV})$: Algorithm $\mathsf{WM\text{-}PopP}$ takes as input (pk, sk) and sends $pk \mid\mid (\pi, \varpi)$ where $(\pi, \varpi) = \mathsf{W\text{-}Sign}^{H_{w,\vec{w}}}(sk, \langle pk \rangle_n)$. Algorithm $\mathsf{WM\text{-}PopV}$ receives $pk \mid\mid (\pi, \varpi)$ and then runs $\mathsf{W\text{-}Vf}^{H_{w,\vec{w}}}(pk, \langle pk \rangle_n, (\pi, \varpi))$ and if the result is 1, replies with pk and else replies with $\perp$. The following theorem combined with [28, Theorem 2] (security of WMS under the KOSK assumption) establishes the security of WMS under $\mathsf{WM\text{-}Pop}$.

Theorem 4.2 *Let A be an* msuf-kr*-adversary, with respect to the* $\mathsf{WM\text{-}Pop}$ *protocol, that runs in time t, makes q_s multisignature queries, q_k registration queries, and outputs a forgery for a group of size at most v. Then there exists an adversary B such that*

$$\mathbf{Adv}^{\text{msuf-kr}}_{\mathsf{WMS},\mathsf{WM\text{-}Pop}}(A) \leq \mathbf{Adv}^{\text{msuf-kr}}_{\mathsf{WMS},\mathsf{Kosk}}(B)$$

and where B runs in time $t + \mathcal{O}(nt_E + (t_E + t_\psi)q_k)$ and makes q_s signature queries.

Proof: Let A be a legitimate adversary against WMS. Like in the proof for BMS, we note that for the scheme WMS, a multisignature session oracle is equivalent to a Waters signature signing oracle for the trusted party. We therefore deal with the latter, which is simpler.

We wish to construct an adversary B which uses A to create a forgery against the WMS multisignature scheme with the Kosk key registration protocol. Adversary B will receive as input system parameters $(h, g, u, u_1, \ldots, u_m)$ and target public key pk^*, where h is a generator of $\mathbb{G}_1$, $u, u_1, \ldots, u_m$ are all elements of $\mathbb{G}_1$, g is a generator of $\mathbb{G}_2$, and $pk^* = g^\alpha$ for some random $\alpha \in \mathbb{Z}_p$. The adversary is also given a signing oracle **OMSign-B** and a key registration oracle **OKReg-B**. Its goal is to produce a valid multisignature forgery which verifies under some set of keys that includes pk^*. Since A is an adversary against WMS with $\mathsf{WM\text{-}Pop}$ key registration, it must take additional input $w, \vec{w}$ (for key registration) and B must simulate for it a signing oracle **OMSign-A** and a key registration oracle **OKReg-A**. The adversary B operates as follows:

- Let $g_1 = \psi(g)$. Choose $z, z_1, \ldots, z_n \xleftarrow{\$} \mathbb{Z}_p^{n+1}$ and let $(w, w_1, \ldots, w_n) \leftarrow (g_1^z, g_1^{z_1}, \ldots, g_1^{z_n})$. Choose $s \xleftarrow{\$} \mathbb{Z}_p$. Run the WMS multisignature adversary A on input $pk = pk^*$, $par = (h, g, u, \vec{u}, w, \vec{w})$.

- On oracle query **OMSign-A**(M_i) from A, query **OMSign-B**(M_i). Upon receiving a signature (σ_i, ρ_i), forward it back to A.
- When A queries **OKReg-A**$(pk_i, (\pi_i, \varpi_i))$, let $sk_i \leftarrow \pi_i / \psi(\varpi_i)^{z+\sum_{j=1}^{n} z_j \cdot \langle pk_i \rangle[j]}$. Then query **OKReg-B**$(pk_i, sk_i)$.
- When A outputs an attempted forgery $(\mathcal{V}, M, (\sigma, \rho))$, output $(\mathcal{V}, M, (\sigma, \rho))$.

We now show that whenever A forges against the WMS multisignature scheme with WM-Pop key registration, B forges against the WMS mulsignature scheme with Kosk key registration (WMS was proven secure under the KOSK assumption in [28]). To do this we first claim that B perfectly simulates the environment for A. This is clear since the public key and system parameters given to A directly correspond to the public key and parameters given to B (namely $h, g, u, u_1, \ldots, u_n$) and thus the sign oracle provided by B is correct. Since B forwards all of A's sign oracle queries to the environment and B makes no other sign oracle queries, it is also clear that if A did not query its forgery message to the sign oracle provided by B, then B also did not query its sign oracle on that message. Lastly, we show that every time A sends a valid POP to the registration oracle provided by B, B is able to 'extract' the secret key to send to its own Kosk registration oracle. To see this recall that a valid POP (π_i, ϖ_i) for key pk_i is such that

$$\mathbf{e}(h, pk_i) \cdot \mathbf{e}(H_{w,\vec{w}}(\langle pk_i \rangle_n), \varpi_i) = \mathbf{e}(\pi_i, g). \tag{10}$$

Also, note that (pk_i, sk_i) is a valid key pair if and only if

$$\mathbf{e}(sk_i, g) = \mathbf{e}(h, pk_i). \tag{11}$$

Now, rearranging equation 10:

$$\begin{aligned}
\mathbf{e}(h, pk_i) &= \frac{\mathbf{e}(\pi_i, g)}{\mathbf{e}(H_{w,\vec{w}}(\langle pk_i \rangle_n), \varpi_i)} \\
&= \frac{\mathbf{e}(\pi_i, g)}{\mathbf{e}(g_1^{z+\sum_{j=1}^{n} z_j \langle pk_i \rangle[j]}, \varpi_i)} \\
&= \frac{\mathbf{e}(\pi_i, g)}{\mathbf{e}(\psi(\varpi_i)^{z+\sum_{j=1}^{n} z_j \langle pk_i \rangle[j]}, g)} \\
&= \mathbf{e}(\pi_i \cdot (\psi(\varpi_i)^{z+\sum_{j=1}^{n} z_j \langle pk_i \rangle[j]})^{-1}, g) \\
&= \mathbf{e}(sk_i, g)
\end{aligned}$$

which from equation 11 means that (pk_i, sk_i) is a valid key pair. This shows that each time adversary A is able to register a key, B is able to 'extract' the corresponding secret key and use it for KOSK key registration.

Now, whenever A forges, it must have properly registered the keys used in the forgery. Since we just showed that a proper key registration by A equates to a proper key registration for B, it follows that whenever A forges, B also forges. So,

$$\mathbf{Adv}^{\text{msuf-kr}}_{\mathsf{WMS},\mathsf{WM\text{-}Pop}}(A) \leq \mathbf{Adv}^{\text{msuf-kr}}_{\mathsf{WMS},\mathsf{Kosk}}(B).$$

In addition to running A, B must must perform one isomorphism and $n+1$ exponentiations before starting A, plus q_k isomorphisms and q_k exponentiations. Thus the running time of B is $t + \mathcal{O}(nt_E + (t_E + t_\psi) \cdot q_k)$ where t is the running time of A. ▌

4.3 Attacks against Standardized Key Registration Protocols

We show how the standardized proof-of-possession based key registration protocols (as per PKCS#10 [35] and RFCs 4210/4211 [1, 36]) fail to prevent rogue key attacks. Let $\mathsf{BadPop} = (\mathsf{BadP}, \mathsf{BadV})$ be the standardized key registration protocol for BMS and let the algorithms be as follows: Algorithm BadP, on input (pk, sk) sends $pk \,||\, \mathsf{B\text{-}Sign}^H(sk, \langle pk \rangle)$ and algorithm BadV, upon receiving (pk, π), runs $\mathsf{B\text{-}Vf}^H(pk, \langle pk \rangle, \pi)$ and replies with pk if the result is 1 and $\perp$ otherwise. Here H is the same hash function as used in B-MSign and B-MVf.

We define a simple msuf-kr adversary A that successfully mounts a rogue-key attack against BMS with respect to the BadPop registration protocol. Adversary A gets the honest party's public key pk^* which is equal to g^{sk^*}. It then chooses $s \xleftarrow{\$} \mathbb{Z}_p$. Its public key is set to $pk = g^s/pk^* = g^{s-sk^*}$. The forgery on any message M and multiset $\{pk^*, pk\}$ is simply $H(M)^s$, which clearly verifies under the two public keys given. Now to register its key, the adversary makes the query $\mathbf{OMSign}(\{pk^*\}, \langle pk \rangle)$, receiving $\sigma = H(\langle pk \rangle)^{sk^*}$. Then A sets $\pi \leftarrow H(\langle pk \rangle)^s/\sigma$ and registers with $pk \,||\, \pi$. It is easy to see that this verifies, and thus A can always output a multisignature forgery: its msuf-kr advantage is one.

An analogous key registration protocol could be defined for WMS, and again a simple attack shows its insecurity. Both approaches fall to attacks because the signatures used for key registration and normal multisignatures are calculated in the same manner. This motivated our simple deviations from standardized registration protocols for the B-Pop and WM-Pop protocols.

4.4 Other POP Variants

Another class of POP-based registration protocols for signature schemes has the CA send a random challenge to the registrant. The registrant must then supply a signature over the challenge message. Our results also apply to such protocols.

We first show that requiring a signature over a random challenge has the exact same pitfalls as requiring a signature of the key being registered. Let $\mathsf{BadRCPop} = (\mathsf{BadRCP}, \mathsf{BadRCV})$ be a random challenge key registration protocol for BMS with its algorithms as follows: algorithm BadRCP, on input (pk, sk) sends pk to the CA; algorithm BadRCV receives pk and replies with a random string $c \in \{0,1\}^n$; BadRCP receives c and replies with $\pi = \mathsf{B\text{-}Sign}^H(sk, c)$; finally BadRCV receives π and outputs pk if $\mathsf{B\text{-}Vf}^H(pk, c, \pi) = 1$ and $\perp$ otherwise. Hash function H is the same one used in B-MSign and B-MVf.

As before, there is a simple msuf-kr adversary A that can mount a rogue-key attack against BMS with this registration protocol. Adversary A gets the honest party's public key pk^* which is equal to g^{sk^*}. It then chooses $s \xleftarrow{\$} \mathbb{Z}_p$. Its public key is set to $pk = g^s/pk^* = g^{s-sk^*}$. The forgery on any message M and multiset $\{pk^*, pk\}$ is simply $H(M)^s$. To register its key, the adversary must sign some random challenge c with the rogue key. To do this, the adversary, upon receiving c, makes the query $\mathbf{OMSign}(\{pk^*\}, c)$ and receives $\sigma = H(c)^{sk^*}$. Then A sets $\pi \leftarrow H(c)^s/\sigma$ and sends π as its response.

To prevent the attack above, we again can use a distinct hash function for signing and key registration. To adapt the proof of Theorem 4.1 to the setting with random challenge key registration, the adversary B simply needs to compute $\gamma_i \leftarrow \alpha - \mathtt{B}[c_i]$, where c_i is the random challenge given when pk_i was registered. The rest of the proof is the same. The proof of Theorem 4.2 can be similarly modified. The reason that this works is that the proof of security does not depend on the choice of string signed in the proof of possession, as long as the string is known to the CA.

Experiment $\mathbf{Exp}^{\text{ranon-ind-b}}_{\mathsf{RS}}(A)$
$par \xleftarrow{\$} \mathsf{RPg}$
Run $A(par)$ handling oracle queries as follows
ORSignLR$(\mathcal{S}, \mathcal{V}, M)$, where $\mathcal{S} = (sk_0, sk_1)$; $\mathcal{V} = pk_0, \ldots, pk_{v-1}$; (pk_0, sk_0), (pk_1, sk_1) valid:
Ret $\mathsf{RSign}_{sk_b}(\mathcal{V}, M)$
A outputs b'
Ret b'

Figure 4: Ring signature anonymity experiment.

5 Ring Signatures in the Registered Key Model

A *ring signature scheme* $\mathsf{RS} = (\mathsf{RPg}, \mathsf{RKg}, \mathsf{RSign}, \mathsf{RVf})$ consists of four algorithms. The parameter generation algorithm generates a parameter string par, which is published globally. The key generation algorithm RKg outputs a key pair (pk, sk). The algorithm $\mathsf{RSign}_{sk}(\mathcal{V}, M) \equiv \mathsf{RSign}(sk, \mathcal{V}, M)$ generates a ring signature on input a secret key sk, a message M, and a multiset of public keys $\mathcal{V}$ such that there exists $pk \in \mathcal{V}$ for which (pk, sk) is a valid key pair. We further assume that $|\mathcal{V}| \geq 2$. Note that we do not require that all elements in $\mathcal{V}$ be distinct.[1] Lastly the verification algorithm $\mathsf{RVf}(\mathcal{V}, M, \sigma)$ outputs a bit. We require that $\mathsf{RVf}(\mathcal{V}, M, \mathsf{RSign}_{sk}(\mathcal{V}, M)) = 1$ for any message M, any valid multiset of public keys, and for any valid sk with a $pk \in \mathcal{V}$. Ring signatures that only allow rings of size exactly κ are called *κ-user ring signatures*.

5.1 Ring Signature Definitions

New anonymity definition. We propose a stronger definition of anonymity than those given by Bender et al. [10]. Intuitively, our definition requires that no adversary should be able to tell what secret key was used to generate a ring signature, even if the adversary itself chooses the secret keys involved. Formally, let A be an adversary and $\mathsf{RS} = (\mathsf{RPg}, \mathsf{RKg}, \mathsf{RSign}, \mathsf{RVf})$ be a ring signature scheme. Then we define the ranon-ind advantage of an adversary A by

$$\mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}}(A) = \Pr\left[\mathbf{Exp}^{\text{ranon-ind-0}}_{\mathsf{RS}}(A) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^{\text{ranon-ind-1}}_{\mathsf{RS}}(A) \Rightarrow 1\right]$$

where the experiment $\mathbf{Exp}^{\text{ranon-ind-b}}_{\mathsf{RS}}(A)$ is defined in Figure 4. We say a scheme is *perfectly r-anon-ind anonymous* if the advantage of any adversary is zero.

The ranon-ind definition is stronger than the strongest definition given in [10] (see Appendix A for details). Even so, it is easy to see that (for example) both of the 2-user ring signature schemes from [10] meet it, and are, in fact, perfectly ranon-ind anonymous.

Unforgeability definitions. We expand the unforgeability definitions given in [10], drawing a distinction between attacks where honest parties can be corrupted and rogue-key attacks (where the adversary can choose public keys). We also lift the strongest unforgeability definition to the registered key model. Fix some number η, representing the number of trusted potential honest signers. Let $\mathsf{RS} = (\mathsf{RPg}, \mathsf{RKg}, \mathsf{RSign}, \mathsf{RVf})$ be a ring signature scheme and $\mathsf{Reg} = (\mathsf{RegP}, \mathsf{RegV})$ be a registration protocol. Then we define the advantages

[1] This generalizes previous treatments, e.g. [10]. While a ring signature over a ring with duplicates is equivalent semantically to one over a ring with all duplicates removed, there is no intrinsic reason *not* to allow duplicates, and thus we would like definitions to capture this generalization. Of course a particular scheme could require distinct public keys, for example by removing duplicates before signing.

Experiment $\mathbf{Exp}^{\text{ruf3-kr}}_{\mathsf{RS},\mathsf{Reg}}(A)$

$par \xleftarrow{\$} \mathsf{RPg}$; For $i = 1$ to η do $(pk_i, sk_i) \xleftarrow{\$} \mathsf{RKg}(par)$

$\mathcal{Q} \leftarrow \mathcal{C} \leftarrow \mathcal{R} \leftarrow \emptyset$; $\mathcal{S} \leftarrow \{pk_i \mid i \in \{1, \ldots, \eta\}\}$

Run $A(par, (pk_i)_1^\eta)$ handling oracle queries as follows

ORSign$(s, \mathcal{V}, M)$, where $s \in [1 \,..\, \eta]$ and $pk_s \in \mathcal{V}$:

$\mathcal{Q} \xleftarrow{\cup} (\mathcal{V}, M)$; If $(\,\mathcal{V} \setminus \mathcal{S}\,) \setminus \mathcal{R} \neq \emptyset$ then Ret $\perp$

Ret $\mathsf{RSign}_{sk_s}(\mathcal{V}, M)$

OCorrupt(i), where $i \in [1 \,..\, \eta]$: $\mathcal{C} \xleftarrow{\cup} i$; Ret sk_i

OKReg: Simulate a new instance of algorithm RegV, forwarding messages to and from A. If the instance's last message is $pk \neq \perp$, then $\mathcal{R} \xleftarrow{\cup} pk$.

A outputs $(\mathcal{P}, M, \sigma)$

$\mathcal{V} = \{pk_i \mid i \in \mathcal{P}\}$

If $\mathsf{RVf}(\mathcal{V}, M, \sigma) = 1 \wedge (\,(\mathcal{V}, M) \notin \mathcal{Q}\,) \wedge (\,\mathcal{P} \cap \mathcal{C} = \emptyset\,) \wedge (\,\mathcal{P} \setminus \{1, \ldots, \eta\} = \emptyset\,)$ then Ret 1

Ret 0

Figure 5: Ring signature unforgeability experiment in the registered key model. The ruf2 experiment is derived by omitting the **OKReg** oracle. The ruf1 experiment is derived by omitting both the **OKReg** and **OCorrupt** oracles.

- $\mathbf{Adv}^{\text{ruf3-kr}}_{\mathsf{RS},\mathsf{Reg}}(A) = \Pr[\mathbf{Exp}^{\text{ruf3-kr}}_{\mathsf{RS},\mathsf{Reg}}(A) \Rightarrow 1]$ (rogue-key attacks, equivalent to Definition 7 in [10] when $\mathsf{Reg} = \mathsf{Plain}$)
- $\mathbf{Adv}^{\text{ruf2}}_{\mathsf{RS}}(A) = \Pr[\mathbf{Exp}^{\text{ruf2}}_{\mathsf{RS}}(A) \Rightarrow 1]$ (corruption attacks, similar to a definition in [21])
- $\mathbf{Adv}^{\text{ruf1}}_{\mathsf{RS}}(A) = \Pr[\mathbf{Exp}^{\text{ruf1}}_{\mathsf{RS}}(A) \Rightarrow 1]$ (chosen subring attacks, Definition 6 in [10])

where the $\mathbf{Exp}^{\text{ruf3-kr}}_{\mathsf{RS},\mathsf{Reg}}(A)$ experiment is defined in Figure 5 and the others are as follows. Experiment $\mathbf{Exp}^{\text{ruf2}}_{\mathsf{RS}}(A)$ is defined by the experiment in Figure 5 except with the **OKReg** oracle omitted. Experiment $\mathbf{Exp}^{\text{ruf1}}_{\mathsf{RS}}(A)$ is defined by the experiment in Figure 5 except both the **OCorrupt** and **OKReg** oracles are omitted. Note that we have not excluded the possibility that duplicate public keys are generated by the experiment[2], and in particular $\mathcal{V}$ and $\mathcal{S}$ might both be multisets in the experiments. (This better reflects the independent nature of key generation by users, who cannot necessarily be trusted to ensure all public keys generated are unique.) On the other hand, $\mathcal{P}$ must not have duplicates.

We also define a variant of the ruf3-kr security definition, which pertains to *fixed-ring* adversaries. These are adversaries which must forge against some ring of public keys selected by the environment. Let $\nu \geq 2$ and assume that RS supports rings of size ν. Define the experiment $\mathbf{Exp}^{\text{ruf3-kr-fr}}_{\mathsf{RS},\mathsf{Reg}}(A)$ by the code for $\mathbf{Exp}^{\text{ruf3-kr}}_{\mathsf{RS},\mathsf{Reg}}(A)$ in Figure 5 except with the following changes: η is everywhere replaced by ν; the forgery output $(\mathcal{P}, M, \sigma)$ must have $\mathcal{P} = \{1, \ldots, \nu\}$; and the (now useless) **OCorrupt** oracle is omitted. We define the ruf3-kr-fr advantage of an adversary A by

- $\mathbf{Adv}^{\text{ruf3-kr-fr}}_{\mathsf{RS},\mathsf{Reg}}(A) = \Pr\left[\mathbf{Exp}^{\text{ruf3-kr-fr}}_{\mathsf{RS},\mathsf{Reg}}(A) \Rightarrow 1\right]$

For κ-user schemes necessarily $\nu = \kappa$. Note that in [10] a similar fixed-ring definition is given, but it does not allow rogue-keys attacks.

An adversary A is *legitimate* if: (1) all queries to **ORSign**$(s, \mathcal{V}, M)$ are such that $s \in \{1, \ldots, \eta\}$; $pk_s \in \mathcal{V}$; and $(\mathcal{V} \setminus \mathcal{S}) \setminus \mathcal{R} = \emptyset$; (2) all queries to **OCorrupt**(i) are such that $i \in \{1, \ldots, \eta\}$ and A never

[2]This formalization implies that for a scheme to be secure against ruf2 and ruf3-kr adversaries, it must ensure that honestly generated keys are duplicates only with low probability. If there exists $pk_i = pk_j$ for $i \neq j$ then such an adversary can simply corrupt i and then forge against j and some other party using sk_i.

repeats a query i to **OCorrupt**; and (3) A only outputs tuples $(\mathcal{P}, M, \sigma)$ for which $\mathcal{P}\backslash\{1,\ldots,\eta\} = \emptyset$ (all indices in $\mathcal{P}$ are distinct and in the range $[1\,..\,\eta]$) and $\mathcal{P} \cap \mathcal{C} = \emptyset$ and $(\mathcal{V}, M) \notin \mathcal{Q}$. Without loss of generality, we will from now on only consider legitimate adversaries.

5.2 Unforgeability Implications for κ-user Schemes

In this section we focus on κ-user ring signature schemes, giving two results. The first is that for κ-user ring signatures meeting the ranon-ind anonymity definition, security against corruption attacks is implied by security against chosen subring attacks. This shows that (for these schemes) the ability to corrupt parties offers little benefit to adversaries. The second result shows that κ-user schemes secure against fixed-ring rogue-key attacks are also secure against general rogue-key attacks. Both results will be useful for establishing the security of schemes against rogue-key attacks when key registration protocols are utilized (as discussed in the next two sections). Additionally, the lemmas could be of independent interest.

Corruptions don't help. We now state and prove a lemma which states that the ability to corrupt parties is of limited usefulness to adversaries attacking κ-user ring signature schemes.

Lemma 5.1 *Let* $\mathsf{RS} = (\mathsf{RPg}, \mathsf{RKg}, \mathsf{RSign}, \mathsf{RVf})$ *be a* κ*-user ring signature scheme. Let* $\eta \geq \kappa$*. and let* A *be an* ruf2 *adversary that makes at most* q_s *signature queries, at most* q_c *corruption queries, and runs in time at most* t*. Then there exist adversaries* B_a *and* B_u *such that*

$$\mathbf{Adv}^{\text{ruf2}}_{\mathsf{RS}}(A) \leq \binom{\eta}{\kappa}\mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}}(B_a) + \binom{\eta}{\kappa}\mathbf{Adv}^{\text{ruf1}}_{\mathsf{RS}}(B_u)$$

where B_a *and* B_u *uses at most* q_s *queries and run in time at most* $t + \mathcal{O}(q_c + (q_s + \eta + 1)\mathsf{Time}(\mathsf{RS}))$.

Proof: Let A be a legitimate ruf2 adversary. We build a legitimate ruf1 adversary B_u that uses A. See Figure 6. The adversary B_u is given inputs a parameter string and set of honestly-generated public keys and access to a signing oracle denoted by **ORSign-1**. Recall that this oracle, which corresponds to the ruf1 experiment, does not allow queries on adversarially-generated public keys. Adversary B_u first selects a set of indices K that represent its guess of which parties will *not* be corrupted by A. Then, a new set of keys is generated by B_u. For every index in K the real key is used. For all others, B_u generates its own public key, secret key pair. Adversary A is run using the resulting multiset of public keys. (A is given a sequence of keys such that the ith key in the sequence corresponds to pk_i). Signing queries **ORSign-2**$(s, \mathcal{V}, M)$ from A are simulated in one of two ways. If the keys in $\mathcal{V}$ correspond to the guessed subset, then B_u utilizes its own oracle to respond. Otherwise, one of the keys that B_u generated itself is used to answer the query. The statement "Let $j \in [1\,..\,\eta]\backslash K$ be s.t. $pk_j \in \mathcal{V}$" means find a (e.g., the first) key pk_j that is both in $\mathcal{V}$ and in the set of public keys generated by B_u. For corruption queries, if the index queried is not in K then the appropriate secret key is returned (in this case B generated the key itself). Otherwise $\perp$ is returned. Note that by the legitimacy of A and the manner in which signing queries are handled B_u is also legitimate.

We now proceed through a game-playing sequence to lower bound the advantage of B_u in terms of the advantage of A and, as we'll see, an ranon-ind adversary B_a. Game G0 represents the first game, see Figure 6. It is equivalent to the adversary B_u except that it generates the parameter string and honest public keys itself and answers **ORSign** queries for the chosen set of public keys directly. Since the outputs of G0 and B_u are distributed equivalently and B_u forges anytime it doesn't output $\perp$, we have

$$\mathbf{Adv}^{\text{ruf1}}_{\mathsf{RS}}(B_u) = \Pr\left[\text{G0}(A) \not\Rightarrow \perp\right]. \tag{12}$$

Adversary $B_u(par, (pk_i)_1^\eta)$:

$K \xleftarrow{\$} \mathbb{P}_\kappa(\{1\,..\,\eta\})$
For $i = 1$ to η do
  If $i \notin K$ then $(pk'_i, sk'_i) \leftarrow \mathsf{RKg}(par)$
  Else $pk'_i \leftarrow pk_i$
$\mathcal{T} \leftarrow \{pk_i \mid i \in K\}$
Run $A(par, (pk'_i)_1^\eta)$, answering queries by:

  query ORSign-2$(s, \mathcal{V}, M)$
  If $\mathcal{V} = \mathcal{T}$ then Ret **ORSign-1**$(s, \mathcal{V}, M)$
  Let $j \in \{1\,..\,\eta\}\backslash K$ be s.t. $pk_j \in \mathcal{V}$
  Ret $\mathsf{RSign}_{sk'_j}(\mathcal{V}, M)$

  query OCorrupt(i)
  $sk \leftarrow \bot$
  If $i \notin K$ then $sk \leftarrow sk'_i$
  Else $bad \leftarrow \mathsf{true}$
  Ret sk

A finishes, outputing $(\mathcal{P}, M, \sigma)$
If $\mathcal{P} \neq K$ then $bad \leftarrow \mathsf{true}$
$\phi \leftarrow \bot$; $\mathcal{V} = \{pk_i \mid i \in \mathcal{P}\}$
If $\mathsf{RVf}(\mathcal{V}, M, \sigma) = 1$ then
  If $\mathsf{bad} \neq \mathsf{true}$ then $\phi \leftarrow (\mathcal{P}, M, \sigma)$
Ret ϕ

procedure Initialize G0
$par \xleftarrow{\$} \mathsf{RPg}$
For $i = 1$ to η do $(pk_i, sk_i) \xleftarrow{\$} \mathsf{RKg}(par)$
$K \xleftarrow{\$} \mathbb{P}_\kappa(\{1\,..\,\eta\})$
For $i = 1$ to η do
  If $i \notin K$ then $(pk'_i, sk'_i) \leftarrow \mathsf{RKg}(par)$
  Else $pk'_i \leftarrow pk_i$
$\mathcal{T} \leftarrow \{pk_i \mid i \in K\}$
Ret $(par, (pk'_i)_1^\eta)$

procedure ORSign-2$(s, \mathcal{V}, M)$
If $\mathcal{V} = \mathcal{T}$ then Ret $\mathsf{RSign}_{sk_s}(\mathcal{V}, M)$
Let $j \in \{1\,..\,\eta\}\backslash K$ be s.t. $pk_j \in \mathcal{V}$
Ret $\mathsf{RSign}_{sk'_j}(\mathcal{V}, M)$

procedure OCorrupt(i)
$sk \leftarrow \bot$
If $i \notin K$ then $sk \leftarrow sk'_i$
Else $bad \leftarrow \mathsf{true}$
Ret sk

procedure Finalize$(\mathcal{P}, M, \sigma)$
If $\mathcal{P} \neq K$ then $bad \leftarrow \mathsf{true}$
$\phi \leftarrow \bot$; $\mathcal{V} = \{pk_i \mid i \in \mathcal{P}\}$
If $\mathsf{RVf}(\mathcal{V}, M, \sigma) = 1$ then
  If $\mathsf{bad} \neq \mathsf{true}$ then $\phi \leftarrow (\mathcal{P}, M, \sigma)$
Ret ϕ

Figure 6: Adversary B_u and game G0 used in proof of Lemma 5.1.

Game G1 is shown in Figure 7. It is the same as G0 except that we simplify the initialization procedure to just pick one set of η key pairs. The view of the adversary is equivalent, since all keys are distributed as before. Thus

$$\Pr\left[\mathrm{G0}(A) \not\Rightarrow \bot\right] = \Pr\left[\mathrm{G1}(A) \not\Rightarrow \bot\right]. \tag{13}$$

Game G2 modifies the **ORSign** procedure of G1: in all cases the secret key associated with the index s is used to answer the response. Otherwise the games are identical. To bound the loss incurred by this game transition, we build an ranon-ind adversary B_a. This adversary will have advantage related to the ability of A to force the output of $\mathrm{G1}(A)$ to differ from the output of $\mathrm{G2}(A)$. Figure 7 depicts the adversary. It utilizes its **ORSignLR** oracle to help answer A's **ORSign-2** queries and returns a one whenever A's forgery is successful and bad is not set. We have that

$$\mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}}(B_a) = \Pr\left[\mathbf{Exp}^{\text{ranon-ind-1}}_{\mathsf{RS}}(B_a) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^{\text{ranon-ind-0}}_{\mathsf{RS}}(B_a) \Rightarrow 1\right]. \tag{14}$$

Note that if B_a is run in the ranon-ind-1 experiment, then the secret key sk_s is always utilized to answer B_a's oracle queries. Thus B_a returns a one whenever $\mathrm{G2}(A)$ does not output $\bot$, and so we have that $\Pr\left[\mathbf{Exp}^{\text{ranon-ind-1}}_{\mathsf{RS}}(B_a) \Rightarrow 1\right] = \Pr\left[\mathrm{G2}(A) \not\Rightarrow \bot\right]$. Similarly, if B_a is run in the ranon-ind-0 experiment, then the secret key sk_j is always used to answer B_a's queries. So $\Pr\left[\mathbf{Exp}^{\text{ranon-ind-0}}_{\mathsf{RS}}(B_a) \Rightarrow 1\right] = \Pr\left[\mathrm{G1}(A) \not\Rightarrow \bot\right]$. Substituting back into (14) and re-arranging we

procedure Initialize G1, G2

$par \xleftarrow{\$} \mathsf{RPg}$
For $i = 1$ to η do $(pk_i, sk_i) \xleftarrow{\$} \mathsf{RKg}(par)$
$K \xleftarrow{\$} \mathbb{P}_\kappa(\{1 \,..\, \eta\})$
$\mathcal{T} \leftarrow \{pk_i \mid i \in K\}$
Ret $(par, (pk_i)_1^\eta)$

procedure ORSign-2$(s, \mathcal{V}, M)$ G1

If $\mathcal{V} = \mathcal{T}$ then Ret $\mathsf{RSign}_{sk_s}(\mathcal{V}, M)$
Let $j \in \{1 \,..\, \eta\} \backslash K$ be s.t. $pk_j \in \mathcal{V}$
Ret $\mathsf{RSign}_{sk_j}(\mathcal{V}, M)$

procedure OCorrupt(i) G1, G2

$sk \leftarrow \bot$
If $i \notin K$ then $sk \leftarrow sk_i$
Else $bad \leftarrow \mathsf{true}$
Ret sk

procedure Finalize$(\mathcal{P}, M, \sigma)$ G1, G2

If $\mathcal{P} \neq K$ then $bad \leftarrow \mathsf{true}$
$\phi \leftarrow \bot$; $\mathcal{V} = \{pk_i \mid i \in \mathcal{P}\}$
If $\mathsf{RVf}(\mathcal{V}, M, \sigma) = 1$ then
  If $\mathsf{bad} \neq \mathsf{true}$ then $\phi \leftarrow (\mathcal{P}, M, \sigma)$
Ret ϕ

procedure ORSign-2$(s, \mathcal{V}, M)$ G2

If $\mathcal{V} = \mathcal{T}$ then Ret $\mathsf{RSign}_{sk_s}(\mathcal{V}, M)$
Let $j \in \{1 \,..\, \eta\} \backslash K$ be s.t. $pk_j \in \mathcal{V}$
Ret $\mathsf{RSign}_{sk_s}(\mathcal{V}, M)$

Adversary $B_a(par)$:

$par \xleftarrow{\$} \mathsf{RPg}$
For $i = 1$ to η do $(pk_i, sk_i) \xleftarrow{\$} \mathsf{RKg}(par)$
$K \xleftarrow{\$} \mathbb{P}_\kappa(\{[1 \,..\, \eta\})$
$\mathcal{T} \leftarrow \{pk_i \mid i \in K\}$
Run $A(par, (pk_i)_1^\eta)$, answering queries by:

  query ORSign-2$(s, \mathcal{V}, M)$
  If $\mathcal{V} = \mathcal{T}$ then Ret $\mathsf{RSign}_{sk_s}(\mathcal{V}, M)$
  Let $j \in \{1 \,..\, \eta\} \backslash K$ be s.t. $pk_j \in \mathcal{V}$
  Ret $\mathbf{ORSignLR}(\{sk_j, sk_s\}, \mathcal{V}, M)$

  query OCorrupt(i)
  $sk \leftarrow \bot$
  If $i \notin K$ then $sk \leftarrow sk_i$
  Else $bad \leftarrow \mathsf{true}$
  Ret sk

A halts with output $(\mathcal{P}, M, \sigma)$
If $\mathcal{P} \neq K$ then $bad \leftarrow \mathsf{true}$
$\phi \leftarrow \bot$; $\mathcal{V} = \{pk_i \mid i \in \mathcal{P}\}$
If $\mathsf{RVf}(\mathcal{V}, M, \sigma) = 1$ then
  If $\mathsf{bad} \neq \mathsf{true}$ then $\phi \leftarrow (\mathcal{P}, M, \sigma)$
If $\phi \neq \bot$ then Ret 1
Ret 0

Figure 7: Games G1 and G2 and adversary B_a used in the proof of Lemma 5.1. Game G2 utilizes the same **Initialize**, **OCorrupt**, and **Finalize** procedures as G1.

have that

$$\Pr\left[\mathrm{G1}(A) \not\Rightarrow \bot\right] = \Pr\left[\mathrm{G2}(A) \not\Rightarrow \bot\right] - \mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}}(B_a)\,. \tag{15}$$

Game G3 (Figure 8, boxed statements omitted) simplifies the **ORSign-2** oracle: the code is different but it implements the same functionality as in G2. As a result of this change, the game no longer requires $\mathcal{T}$, thus we omit it from **Initialize**.

$$\Pr\left[\mathrm{G2}(A) \not\Rightarrow \bot\right] = \Pr\left[\mathrm{G3}(A) \not\Rightarrow \bot\right]. \tag{16}$$

Let Good_i be the event that $\mathrm{G}i(A)$ does not set bad. Then we have by (16) that

$$\Pr\left[\mathrm{G2}(A) \not\Rightarrow \bot\right] = \Pr\left[\mathrm{G3}(A) \not\Rightarrow \bot\right] \geq \Pr\left[\mathrm{G3}(A) \not\Rightarrow \bot \wedge \mathsf{Good}_3\right].$$

Game G4 includes the boxed statements, which assist it in answering **ORSign-2** queries and ensure that G4 only outputs $\bot$ if A failed to produce a valid forgery. Games G3 and G4 are identical-until-bad. By a variant [6] of the fundamental lemma of game-playing [9] we have that

$$\Pr\left[\mathrm{G3}(A) \not\Rightarrow \bot \wedge \mathsf{Good}_3\right] = \Pr\left[\mathrm{G4}(A) \not\Rightarrow \bot \wedge \mathsf{Good}_4\right]. \tag{17}$$

procedure Initialize G3 $\boxed{\text{G4}}$

$par \xleftarrow{\$} \mathsf{RPg}$
For $i = 1$ to η do $(pk_i, sk_i) \xleftarrow{\$} \mathsf{RKg}(par)$
$K \xleftarrow{\$} \mathbb{P}_\kappa(\{1 \,..\, \eta\})$
Ret $(par, (pk_i)_1^\eta)$

procedure ORSign-2$(s, \mathcal{V}, M)$

Ret $\mathsf{RSign}_{sk_s}(\mathcal{V}, M)$

procedure OCorrupt(i)

$sk \leftarrow \bot$
If $i \notin K$ then $sk \leftarrow sk_i$
Else $bad \leftarrow \mathsf{true}$ $\boxed{;\ sk \leftarrow sk_i}$
Ret sk

procedure Finalize$(\mathcal{P}, M, \sigma)$

If $\mathcal{P} \neq K$ then $bad \leftarrow \mathsf{true}$
$\phi \leftarrow \bot$; $\mathcal{V} = \{pk_i \,|\, i \in \mathcal{P}\}$
If $\mathsf{RVf}(\mathcal{V}, M, \sigma) = 1$ then
  If $\mathsf{bad} \neq \mathsf{true}$ then $\phi \leftarrow (\mathcal{P}, M, \sigma)$
  $\boxed{\text{Else } \phi \leftarrow (\mathcal{P}, M, \sigma)}$
Ret ϕ

procedure Initialize G5

$par \xleftarrow{\$} \mathsf{RPg}$
For $i = 1$ to η do $(pk_i, sk_i) \xleftarrow{\$} \mathsf{RKg}(par)$
$K \xleftarrow{\$} \mathbb{P}_\kappa(\{1 \,..\, \eta\})$
Ret $(par, (pk_i)_1^\eta)$

procedure ORSign-2$(s, \mathcal{V}, M)$

Ret $\mathsf{RSign}_{sk_s}(\mathcal{V}, M)$

procedure OCorrupt(i)

If $i \in K$ then $bad \leftarrow \mathsf{true}$
Ret sk_i

procedure Finalize$(\mathcal{P}, M, \sigma)$

If $\mathcal{P} \neq K$ then $bad \leftarrow \mathsf{true}$
$\phi \leftarrow \bot$; $\mathcal{V} = \{pk_i \,|\, i \in \mathcal{P}\}$
If $\mathsf{RVf}(\mathcal{V}, M, \sigma) = 1$ then $\phi \leftarrow (\mathcal{P}, M, \sigma)$
Ret ϕ

procedure Initialize G6

$par \xleftarrow{\$} \mathsf{RPg}$
For $i = 1$ to η do $(pk_i, sk_i) \xleftarrow{\$} \mathsf{RKg}(par)$
Ret $(par, (pk_i)_1^\eta)$

procedure ORSign-2$(s, \mathcal{V}, M)$

Ret $\mathsf{RSign}_{sk_s}(\mathcal{V}, M)$

procedure OCorrupt(i)

Ret sk_i

procedure Finalize$(\mathcal{P}, M, \sigma)$

$K \xleftarrow{\$} \mathbb{P}_\kappa(\{1 \,..\, \eta\})$
If $\mathcal{P} \neq K$ then $bad \leftarrow \mathsf{true}$
$\phi \leftarrow \bot$; $\mathcal{V} \leftarrow \{pk_i \,|\, i \in \mathcal{P}\}$
If $\mathsf{RVf}(\mathcal{V}, M, \sigma) = 1$ then $\phi \leftarrow (\mathcal{P}, M, \sigma)$
Ret ϕ

Figure 8: Games G3, G4, G5, and G6 used in the proof of Lemma 5.1. Game G3 does not include the boxed statements, while G4 includes them.

Game G5 simplifies G4 in light of always answering corruption queries and always outputing a successful forgery by A. These changes do not change the implemented functionality of procedures **OCorrupt** and **Finalize**, so $\Pr[\mathrm{G4}(A) \not\Rightarrow \perp \wedge \mathsf{Good}_4] = \Pr[\mathrm{G5}(A) \not\Rightarrow \perp \wedge \mathsf{Good}_5]$. Note that in game G5 anytime bad is set within **OCorrupt**, necessarily $\mathcal{P} \neq K$ during **Finalize**. This is because A is legitimate, implying that the set $\mathcal{P}$ it outputs does not contain any indices previously queried to **OCorrupt**. Thus, if bad is set in **OCorrupt** for an index $i \in K$, then $i \notin \mathcal{P}$ and so $K \neq \mathcal{P}$. Game G6 therefore dispenses with the (now redundant) setting of bad in **OCorrupt** and additionally moves the choice of K to the **Finalize** procedure (they are no longer used elsewhere in the game). We have already justified that these changes are without loss, thus $\Pr[\mathrm{G5}(A) \not\Rightarrow \perp \wedge \mathsf{Good}_5] = \Pr[\mathrm{G6}(A) \not\Rightarrow \perp \wedge \mathsf{Good}_6]$.

In game G6 we note that, in fact, the events Good_6 and $\mathrm{G6}(A) \not\Rightarrow \perp$ are independent. Although the setting of bad and $\mathrm{G6}(A) \not\Rightarrow \perp$ both rely on the adversary's choice of $\mathcal{P}$, the probability that bad is set for *any* set $\mathcal{P}$ is the same, meaning that the event Good_6 is independent of a particular choice of $\mathcal{P}$, and hence of $\mathrm{G6}(A) \not\Rightarrow \perp$. Thus we have

$$\Pr[\mathrm{G6}(A) \not\Rightarrow \perp \wedge \mathsf{Good}_6] = \Pr[\mathrm{G6}(A) \not\Rightarrow \perp] \cdot \Pr[\mathsf{Good}_6] = \binom{\eta}{\kappa}^{-1} \cdot \Pr[\mathrm{G6}(A) \not\Rightarrow \perp]$$

where the last equation comes from noting that bad is not set if K is chosen to be the set of indices used in the forgery.

Finally we have that $\mathrm{G6}(A)$ is equivalent to $\mathbf{Exp}^{\mathrm{ruf2}}_{\mathsf{RS}}(A)$, giving that $\Pr[\mathrm{G6}(A) \not\Rightarrow \perp] = \mathbf{Adv}^{\mathrm{ruf2}}_{\mathsf{RS}}(A)$. Collecting all the previous game transitions we have that

$$\begin{aligned}
\Pr[\mathrm{G0}(A) \not\Rightarrow \perp] &= \Pr[\mathrm{G1}(A) \not\Rightarrow \perp] \\
&= \Pr[\mathrm{G2}(A) \not\Rightarrow \perp] - \mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}}(B_a) \\
&\geq \Pr[\mathrm{G3}(A) \not\Rightarrow \perp \wedge \mathsf{Good}] - \mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}}(B_a) \\
&= \Pr[\mathrm{G4}(A) \not\Rightarrow \perp \wedge \mathsf{Good}] - \mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}}(B_a) \\
&= \Pr[\mathrm{G5}(A) \not\Rightarrow \perp \wedge \mathsf{Good}] - \mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}}(B_a) \\
&= \Pr[\mathrm{G6}(A) \not\Rightarrow \perp \wedge \mathsf{Good}] - \mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}}(B_a) \\
&= \Pr[\mathsf{Good}] \cdot \Pr[\mathrm{G6}(A) \not\Rightarrow \perp] - \mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}}(B_a) \\
&= \binom{\eta}{\kappa}^{-1} \cdot \mathbf{Adv}^{\mathrm{ruf2}}_{\mathsf{RS}}(A) - \mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}}(B_a)
\end{aligned}$$

which, combined with (12), implies the lemma statement.

The running times of both B_u and B_a includes the time to run A plus the overhead associated with setting up the additional public keys and answering oracle queries. ∎

Fixed versus arbitrary ring security. The next lemma states that any κ-user ring signature scheme secure against rogue-key adversaries forging against a particular, fixed ring are also secure against the more general ruf3-kr adversaries. This lemma will be useful for simplifying the proof of Theorem 5.4, but we expect that it is also of independent interest.

Lemma 5.2 *Let* $\mathsf{RS} = (\mathsf{RPg}, \mathsf{RKg}, \mathsf{RSign}, \mathsf{RVf})$ *be a* κ*-user ring signature scheme and let* $\mathsf{Reg} = (\mathsf{RegP}, \mathsf{RegV})$ *be a registration protocol. Let* $\eta \geq \kappa$*. Let* A *be an* ruf3-kr *adversary making at most* q_s *signing queries, at most* q_c *corruption queries, at most* q_k *key registration queries, and running in time at most* t*. Then there exists an* ruf3-kr-fr *adversary* B *such that*

$$\mathbf{Adv}^{\text{ruf3-kr}}_{\mathsf{RS},\mathsf{Reg}}(A) \leq \binom{\eta}{\kappa} \mathbf{Adv}^{\text{ruf3-kr-fr}}_{\mathsf{RS},\mathsf{Reg}}(B)$$

Adversary $B(par, (pk_i)_1^\kappa))$:

$K \xleftarrow{\$} \mathbb{P}_\kappa(\{1\,..\,\eta\})$
$j \leftarrow 1$
For $i = 1$ to η do
  If $i \notin K$ then
    $(pk'_i, sk'_i) \leftarrow \mathsf{RKg}(par)$
    Register pk'_i using **OKReg-B**
  Else $pk'_i \leftarrow pk_j$; $\mathtt{I}[i] \leftarrow j$; $j \leftarrow j + 1$
Run $A(par, (pk'_i)_1^\eta)$, answering queries by:

  query ORSign-A$(s, \mathcal{V}, M)$
  If $s \in K$ then Ret **ORSign-B**$(\mathtt{I}[s], \mathcal{V}, M)$
  Ret $\mathsf{RSign}_{sk'_s}(\mathcal{V}, M)$

  query OCorrupt(i)
  $sk \leftarrow \bot$
  If $i \notin K$ then $sk \leftarrow sk'_i$
  Else $bad \leftarrow \mathsf{true}$
  Ret sk

A finishes, outputing $(\mathcal{P}, M, \sigma)$
If $\mathcal{P} \neq K$ then $bad \leftarrow \mathsf{true}$
$\phi \leftarrow \bot$; $\mathcal{V} = \{pk_i \mid i \in \mathcal{P}\}$
If $\mathsf{RVf}(\mathcal{V}, M, \sigma) = 1$ then
  If $\mathsf{bad} \neq \mathsf{true}$ then $\phi \leftarrow (\{1, \ldots, \kappa\}, M, \sigma)$
Ret ϕ

procedure Initialize G0
$par \xleftarrow{\$} \mathsf{RPg}$
For $i = 1$ to κ do $(pk_i, sk_i) \xleftarrow{\$} \mathsf{RKg}(par)$
$K \xleftarrow{\$} \mathbb{P}_\kappa(\{1\,..\,\eta\})$
For $i = 1$ to η do
  If $i \notin K$ then
    $(pk'_i, sk'_i) \leftarrow \mathsf{RKg}(par)$
  Else $pk'_i \leftarrow pk_j$; $\mathtt{I}[i] \leftarrow j$; $j \leftarrow j + 1$
Ret $(par, (pk'_i)_1^\eta)$

procedure ORSign-A$(s, \mathcal{V}, M)$
If $s \in K$ then $sk \leftarrow sk_{\mathtt{I}[s]}$; Ret $\mathsf{RSign}_{sk}(\mathcal{V}, M)$
Ret $\mathsf{RSign}_{sk'_s}(\mathcal{V}, M)$

procedure OCorrupt(i)
$sk \leftarrow \bot$
If $i \notin K$ then $sk \leftarrow sk'_i$
Else $bad \leftarrow \mathsf{true}$
Ret sk

procedure Finalize$(\mathcal{P}, M, \sigma)$
If $\mathcal{P} \neq K$ then $bad \leftarrow \mathsf{true}$
$\phi \leftarrow \bot$; $\mathcal{V} = \{pk_i \mid i \in \mathcal{P}\}$
If $\mathsf{RVf}(\mathcal{V}, M, \sigma) = 1$ then
  If $\mathsf{bad} \neq \mathsf{true}$ then $\phi \leftarrow (\{1, \ldots, \kappa\}, M, \sigma)$
Ret ϕ

Figure 9: Adversary B and game G0 used in proof of Lemma 5.2.

where B runs in time at most $t + \mathcal{O}(q_c + (\eta + q_s)\mathsf{Time}(\mathsf{RS}))$ and uses at most q_s signing queries and at most $q_k + \eta - \kappa$ registration queries.

Proof: The proof is similar in spirit to the proof of Lemma 5.1. Let A be a legitimate ruf3-kr adversary. We build a legitimate ruf3-kr-fr adversary B that uses A. See Figure 9. The adversary B is given inputs a parameter string and a ring of size κ that it must forge against. Adversary B has access to a signing oracle denoted by **ORSign-B** and a key registration oracle **OKReg**. Here the **ORSign-B** oracle allows queries on (registered) adversarially-chosen keys, as per the ruf3-kr experiment. Adversary B first selects a set of indices K that represent its guess of which parties will be forged against by A. Then, a new set of keys are generated by B. For every index in K one of the target keys is used. For all others, B generates its own public key, secret key pair and registers it as a rogue key using its **OKReg-B** oracle. Adversary A is run using the full set of η public keys. The table $\mathtt{I}$ is used to map between the indices in the set of public keys given to A and the indices in the set of public keys given to B. Signing queries **ORSign-A**$(s, \mathcal{V}, M)$ from A are simulated in one of two ways. If the signing key is one of the target keys, then B queries its **ORSign-B** oracle and returns the result. Otherwise the signing key must be one of those generated by B initially, and so B uses it to generate the requested ring signature. For corruption queries, if the index queried is not in K then the secret key is returned (in this case B knows the secret key). Otherwise $\bot$ is returned. Key registration invocations, which aren't explicitly shown in the figure

procedure Initialize G1

$par \xleftarrow{\$} \mathsf{RPg}$
For $i = 1$ to η do $(pk_i, sk_i) \xleftarrow{\$} \mathsf{RKg}(par)$
$K \xleftarrow{\$} \mathbb{P}_\kappa(\{1 \,..\, \eta\})$
Ret $(par, (pk'_i)_1^\eta)$

procedure ORSign-A$(s, \mathcal{V}, M)$

Ret $\mathsf{RSign}_{sk_s}(\mathcal{V}, M)$

procedure OCorrupt(i)

$sk \leftarrow \bot$
If $i \notin K$ then $sk \leftarrow sk_i$
Else $bad \leftarrow \mathsf{true}$
Ret sk

procedure Finalize$(\mathcal{P}, M, \sigma)$

If $\mathcal{P} \neq K$ then $bad \leftarrow \mathsf{true}$
$\phi \leftarrow \bot$; $\mathcal{V} = \{pk_i \mid i \in \mathcal{P}\}$
If $\mathsf{RVf}(\mathcal{V}, M, \sigma) = 1$ then
  If $\mathsf{bad} \neq \mathsf{true}$ then $\phi \leftarrow (\mathcal{P}, M, \sigma)$
Ret ϕ

Figure 10: Game G1 used in the proof of Lemma 5.2.

for brevity, are just forwarded to B's **OKReg-B** oracle.

We now proceed through a game-playing sequence to lower bound the advantage of B in terms of the advantage of A. Game G0 represents the first game, see Figure 6. It is equivalent to the adversary B except that it generates the parameter string and honest public keys itself and answers **ORSign-A** queries with $s \in K$ directly via computing RSign. It dispenses with the registration of the public keys it generates, which is no longer necessary since the game now also simulates the environment. Key registration queries made by A are handled by simulating a new instance of RegV (this is not shown in Figure 9 for brevity, nor will it be shown in the rest of the games as it never changes). Since the outputs of G0(A) and B are distributed equivalently and B forges any time it doesn't output $\bot$, we have

$$\mathbf{Adv}^{\text{ruf3-kr-fr}}_{\mathsf{RS}}(B) = \Pr\left[\text{G0}(A) \not\Rightarrow \bot\right]. \tag{18}$$

Game G1 (Figure 10, boxed statements omitted) simplifies the **Initialize** procedure by picking one set of η public keys. This means that the table `I` is no longer needed, and thus omitted. The **ORSign-A** oracle is thus written more simply; the new code implements the same functionality as the **ORSign-A** oracle of G0. The **Finalize** procedure outputs $(\mathcal{P}, M, \sigma)$ as opposed to $(\{1, \ldots, \kappa\}, M, \sigma)$ in G0, but this does not change the probability with which the game outputs $\bot$. We have that

$$\Pr\left[\text{G0}(A) \not\Rightarrow \bot\right] = \Pr\left[\text{G1}(A) \not\Rightarrow \bot\right]. \tag{19}$$

The rest of the game sequence (what would be games G1—G4) is similar to the sequence G3—G6 used in the proof of Lemma 5.1 (except for the implicit key registration oracle), see Figure 8. We therefore omit the next 4 games and their analysis, which can be easily adapted from the games

and analysis in the proof of Lemma 5.1. We conclude that

$$\Pr\left[\mathrm{G1}(A) \not\Rightarrow \perp\right] \geq \binom{\eta}{\kappa}^{-1} \cdot \mathbf{Adv}^{\text{ruf3-kr}}_{\mathsf{RS},\mathsf{Reg}}(A)$$

which combined with (18) and (19) implies the lemma statement.

The running time of B is equal to the running time of A plus the overhead of initializing the public keys and answering oracle queries, leading to the time bound given in the lemma statement. ∎

5.3 KOSK Improves Unforgeability Guarantees

Using the key registration protocol Kosk, any scheme that is unforgeable with respect to corruption attacks (ruf2) and meets our strong definition of anonymity is also secure against rogue-key attacks (ruf3-kr). Note that this result (unlike those in the last section) applies to any ring signature scheme, not just κ-user ring signature schemes.

Theorem 5.3 *Fix η and let $\mathsf{RS} = (\mathsf{RPg}, \mathsf{RKg}, \mathsf{RSign}, \mathsf{RVf})$ be a ring signature scheme. Let A be an* ruf3 *adversary with respect to the* Kosk *protocol that makes at most q_s signature queries, at most q_c corruption queries, at most q_k registration queries, and runs in time at most t. Then there exist adversaries B_a and B_u such that*

$$\mathbf{Adv}^{\text{ruf3-kr}}_{\mathsf{RS},\mathsf{Kosk}}(A) \leq \mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}}(B_a) + \mathbf{Adv}^{\text{ruf2}}_{\mathsf{RS}}(B_u)$$

where B_a and B_u both run in time at most $t + \mathcal{O}(q_k + q_c + (q_s + \eta + 1)\mathsf{Time}(\mathsf{RS}))$ and make at most q_s sign queries and at most q_c corrupt queries.

Before proceeding to the proof we point out that one can apply Lemma 5.1 and then Theorem 5.3 to the two 2-user ring signature schemes from Bender et al., rendering them secure against rogue-key attacks when Kosk is used for key registration. Now we prove Theorem 5.3.

Proof: Let A be a legitimate ruf3 adversary with respect to the Kosk key registration protocol. In this case the key registration oracle registers the public key of any valid (pk, sk) pair queried by A. We can assume without loss of generality that A never queries its registration oracle with a pair (pk, sk) that is *not* a valid key pair. We specify a legitimate ruf2 adversary B_u, shown in Figure 11, that takes input a parameter string and set of public keys and has access to a ruf2 signing oracle (denoted **ORSign-2**). It utilizes a table P to record the secret keys registered for adversarially-chosen public keys (initially, P is everywhere undefined). Adversary B_u runs A and responds to A's signing queries in one of two ways. If all the public keys in the query are ones generated by the environment, then B_u uses its **ORSign-2** oracle to generate the response. Otherwise, it uses the secret key registered for one of the adversarially-chosen public keys (say, the lexicographically first) to answer the query. Since A is legitimate, any adversarially-chosen public key used in a signing query must first have been registered. Adversary B_u is legitimate because A is legitimate and because B_u never queries its **ORSign-2** oracle with a set of public keys including rogue keys.

We specify a game G0 in Figure 11 which is just like B_u, except that it runs the parameter string and the public key, secret key generation algorithms itself and answers **ORSign-3** and **OCorrupt-3** queries directly. Its output is distributed identically to B_u and B_u forges anytime its output is not $\perp$. Thus

$$\mathbf{Adv}^{\text{ruf2}}_{\mathsf{RS}}(B_u) = \Pr\left[\mathrm{G0}(A) \not\Rightarrow \perp\right]. \tag{20}$$

Game G1 combines the **Initialize**, **OCorrupt**, **OKReg**, and **Finalize** procedures of G0 with the **ORSign-3** procedure (labeled G1) shown in Figure 11. In G1 sk_s is always utilized to generate

Adversary $B_u(par, (pk_i)_1^\eta)$
$\mathcal{S} \leftarrow \{pk_1, \ldots, pk_\eta\}$
Run $A(par, (pk_i)_1^\eta)$, answering queries by:

query ORSign-3$(s, \mathcal{V}, M)$
If $\mathcal{V} \subseteq \mathcal{S}$ then Ret **ORSign-2**$(s, \mathcal{V}, M)$
Let pk be s.t. $pk \in \mathcal{V} \backslash \mathcal{S}$
Ret $\mathsf{RSign}_{\mathtt{P}[pk]}(\mathcal{V}, M)$

query OCorrupt-3(i)
Ret **OCorrupt-2**(i)

query OKReg(pk, sk)
$\mathtt{P}[pk] \leftarrow sk$
Return pk

A finishes, outputing $(\mathcal{P}, M, \sigma)$
$\mathcal{V} \leftarrow \{pk_i \mid i \in \mathcal{P}\}$
If $\mathsf{RVf}(\mathcal{V}, M, \sigma) = 1$ then Ret $(\mathcal{P}, M, \sigma)$
Ret $\perp$

procedure ORSign-3$(s, \mathcal{V}, M)$ G1
If $\mathcal{V} \subseteq \mathcal{S}$ then Ret $\mathsf{RSign}_{sk_s}(\mathcal{V}, M)$
Let pk be s.t. $pk \in \mathcal{V} \backslash \mathcal{S}$
Ret $\mathsf{RSign}_{sk_s}(\mathcal{V}, M)$

Adversary $B_a(par, (pk_i)_1^\eta))$:
$\mathcal{S} \leftarrow \{pk_1, \ldots, pk_\eta\}$
Run $A(par, (pk_i)_1^\eta)$, answering queries by:

query ORSign-3$(s, \mathcal{V}, M)$
If $\mathcal{V} \subseteq \mathcal{S}$ then Ret $\mathsf{RSign}_{sk_s}(\mathcal{V}, M)$
Let pk be s.t. $pk \in \mathcal{V} \backslash \mathcal{S}$
Ret **ORSignLR**$(\{\mathtt{P}[pk], sk_s\}, \mathcal{V}, M)$

query OCorrupt-3(i)
Ret sk_i

query OKReg(pk, sk)
$\mathtt{P}[pk] \leftarrow sk$
Return pk

A finishes, outputing $(\mathcal{P}, M, \sigma)$
$\mathcal{V} \leftarrow \{pk_i \mid i \in \mathcal{P}\}$
If $\mathsf{RVf}(\mathcal{V}, M, \sigma) = 1$ then Ret 1
Ret 0

procedure Initialize: G0, G1
$par \xleftarrow{\$} \mathsf{RPg}$
For $i = 1$ to η do $(pk_i, sk_i) \xleftarrow{\$} \mathsf{RKg}(par)$
$\mathcal{S} \leftarrow \{pk_1, \ldots, pk_\eta\}$
Run $A(par, (pk_i)_1^\eta)$, answering queries by:

procedure ORSign-3$(s, \mathcal{V}, M)$ G0
If $\mathcal{V} \subseteq \mathcal{S}$ then Ret $\mathsf{RSign}_{sk_s}(\mathcal{V}, M)$
Let pk be s.t. $pk \in \mathcal{V} \backslash \mathcal{S}$
Ret $\mathsf{RSign}_{\mathtt{P}[pk]}(\mathcal{V}, M)$

procedure OCorrupt-3(i) G0, G1
Ret sk_i

procedure OKReg(pk, sk) G0, G1
$\mathtt{P}[pk] \leftarrow sk$
Return pk

procedure Finalize$(\mathcal{P}, M, \sigma)$ G0, G1
$\mathcal{V} \leftarrow \{pk_i \mid i \in \mathcal{P}\}$
If $\mathsf{RVf}(\mathcal{V}, M, \sigma) = 1$ then Ret $(\mathcal{P}, M, \sigma)$
Ret $\perp$

procedure Initialize: G2
$par \xleftarrow{\$} \mathsf{RPg}$
For $i = 1$ to η do $(pk_i, sk_i) \xleftarrow{\$} \mathsf{RKg}(par)$
$\mathcal{S} \leftarrow \{pk_1, \ldots, pk_\eta\}$
Run $A(par, (pk_i)_1^\eta)$, answering queries by:

procedure ORSign-3$(s, \mathcal{V}, M)$
Ret $\mathsf{RSign}_{sk_s}(\mathcal{V}, M)$

procedure OCorrupt-3(i)
Ret sk_i

procedure OKReg(pk, sk)
$\mathtt{P}[pk] \leftarrow sk$
Return pk

procedure Finalize$(\mathcal{P}, M, \sigma)$
$\mathcal{V} \leftarrow \{pk_i \mid i \in \mathcal{P}\}$
If $\mathsf{RVf}(\mathcal{V}, M, \sigma) = 1$ then Ret $(\mathcal{P}, M, \sigma)$
Ret $\perp$

Figure 11: Games and adversaries used in the proof of Theorem 5.3. Game G1 (resp. G2) is fully specified by replacing the **ORSign-3** procedure of G0 with the **ORSign-3** labeled with G1 (resp. G2).

responses to signing queries. To bound the loss due to this change, we build an ranon-ind adversary B_a and will relate the difference in the output distributions of G0(A) and G1(A) to the advantage of B_a. Figure 11 shows the adversary. It behaves exactly like G0 and G1 except that it utilizes its **ORSignLR** oracle to respond to some signing queries. By the definition of ranon-ind advantage we have that

$$\mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}}(B_a) = \Pr\left[\mathbf{Exp}^{\text{ranon-ind-1}}_{\mathsf{RS}}(B_a) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^{\text{ranon-ind-0}}_{\mathsf{RS}}(B_a) \Rightarrow 1\right]. \tag{21}$$

Note that if B_a is run in the ranon-ind-1 experiment, then the secret key sk_s is always utilized to answer B_a's oracle queries. Thus B_a returns a one whenever G1(A) would not output $\perp$, and so we have that $\Pr\left[\mathbf{Exp}^{\text{ranon-ind-1}}_{\mathsf{RS}}(B_a) \Rightarrow 1\right] = \Pr\left[\text{G1}(A) \not\Rightarrow \perp\right]$. Similarly, if B_a is run in the ranon-ind-0 experiment, then the secret key $\mathsf{P}[pk]$ is always used to answer B_a's queries. So $\Pr\left[\mathbf{Exp}^{\text{ranon-ind-0}}_{\mathsf{RS}}(B_a) \Rightarrow 1\right] = \Pr\left[\text{G0}(A) \not\Rightarrow \perp\right]$. Substituting back into (21) and re-arranging we have that

$$\Pr\left[\text{G0}(A) \not\Rightarrow \perp\right] = \Pr\left[\text{G1}(A) \not\Rightarrow \perp\right] - \mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}}(B_a). \tag{22}$$

The final game is G2, which just simplifies the **ORSign-3** oracle from game G1. The change does not change the functionality of the oracle, and thus $\Pr\left[\text{G1}(A) \not\Rightarrow \perp\right] = \Pr\left[\text{G2}(A) \not\Rightarrow \perp\right]$. Now it is apparent that G2 simulates for A exactly the experiment $\mathbf{Exp}^{\text{ruf2}}_{\mathsf{RS}}(A)$ and so we have that $\Pr\left[\text{G2}(A) \not\Rightarrow \perp\right] = \mathbf{Adv}^{\text{ruf2}}_{\mathsf{RS}}(A)$. Combining all of the above, we have that

$$\begin{aligned}
\mathbf{Adv}^{\text{ruf3}}_{\mathsf{RS},\mathsf{Kosk}}(B_u) &= \Pr\left[\text{G0}(A) \not\Rightarrow \perp\right] \\
&= \Pr\left[\text{G1}(A) \not\Rightarrow \perp\right] - \mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}}(B_a) \\
&= \Pr\left[\text{G2}(A) \not\Rightarrow \perp\right] - \mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}}(B_a) \\
&= \mathbf{Adv}^{\text{ruf2}}_{\mathsf{RS}}(A) - \mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}}(B_a).
\end{aligned}$$

The running time of B_u and B_a are (each) the running time of A plus the overhead needed to respond to A's oracle queries, leading to the time bounds given in the theorem statement. ∎

5.4 Ring Signatures with POPs

For all the reasons already described, we'd like to avoid the KOSK assumption wherever possible. Thus, we give a proof-of-possession based registration protocol for the 2-user scheme based on Waters signatures from Bender et al. [10]. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g, \mathbf{e}$ be pairing parameters with $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$. Let $\mathsf{WRS} = (\mathsf{W\text{-}RKg}, \mathsf{W\text{-}RSign}, \mathsf{W\text{-}RVf})$ be the ring signature scheme defined as below.

$\underline{\mathsf{W\text{-}RKg}}$	$\underline{\mathsf{W\text{-}RSign}(sk, \{pk, pk'\}, M)}$	$\underline{\mathsf{W\text{-}RVf}(\{pk, pk'\}, M, (\sigma, \rho))}$
$sk \xleftarrow{\$} \mathbb{Z}_p$; $g_1 \leftarrow g^{sk}$	$(g_1, u, \vec{u}) \leftarrow pk$	$(g_1, u, \vec{u}) \leftarrow pk$
$u, u_1, \ldots, u_n \xleftarrow{\$} \mathbb{G}^{n+1}$	$(g_1', u', \vec{u}') \leftarrow pk'$	$(g_1', u', \vec{u}') \leftarrow pk'$
$pk \leftarrow (g_1, u, \vec{u})$	$r \xleftarrow{\$} \mathbb{Z}_p$; $\rho \leftarrow g^r$	$Y \leftarrow H_{u,\vec{u}}(M) \cdot H_{u',\vec{u}'}(M)$
Return (pk, sk)	$Y \leftarrow H_{u,\vec{u}}(M) \cdot H_{u',\vec{u}'}(M)$	If $\mathbf{e}(g_1, g_1') \cdot \mathbf{e}(Y, \rho) = \mathbf{e}(\sigma, g)$ then
	$\sigma \leftarrow (g_1')^{sk} \cdot Y^r$	Return 1
	Return (σ, ρ)	Return 0

When we write $(g_1, u, \vec{u}) \leftarrow pk$ (resp. $(g_1', u', \vec{u}') \leftarrow pk'$), we mean that pk (resp. pk') is parsed into its constituent parts. In [10] the scheme is proven secure against ruf1 adversaries and we can apply Theorem 5.1 to show security against ruf2 adversaries. Additionally it is easy to see that

the scheme is perfectly ranon-ind. However, as shown in [37] the scheme is *not* secure against rogue-key attacks (in the plain setting). We now show a simple proof-of-possession based registration protocol under which provable security against rogue-key attacks can be achieved. Choose global parameters $(h_0, h_1, w, w_1, \ldots, w_n) \xleftarrow{\$} (\mathbb{G}_1)^{n+3}$ (this will require trusted setup, since the CA should not know the discrete logs of these values). Then we specify the registration protocol $\mathsf{WR\text{-}Pop} = (\mathsf{WR\text{-}PopP}, \mathsf{WR\text{-}PopV})$. Algorithm $\mathsf{WR\text{-}PopP}$ takes as input (pk, sk) and sends $pk \,||\, (\pi_0, \varpi_0, \pi_1, \varpi_1)$ which is computed by generating the two signatures $\mathsf{W\text{-}Sign}^{H_{w,\vec{w}}}(h_0^{sk}, \langle pk \rangle_n)$ and $\mathsf{W\text{-}Sign}^{H_{w,\vec{w}}}(h_1^{sk}, \langle pk \rangle_n)$. Algorithm $\mathsf{WR\text{-}PopV}$, upon receiving the message, verifies the signatures in the natural way: get g_1 from pk and check that both $\mathbf{e}(g_1, h_0) \cdot \mathbf{e}(H_{w,\vec{w}}(\langle pk \rangle_n), \varpi_0) = \mathbf{e}(\pi_0, g)$ and $\mathbf{e}(g_1, h_1) \cdot \mathbf{e}(H_{w,\vec{w}}(\langle pk \rangle_n), \varpi_1) = \mathbf{e}(\pi_1, g)$. If both signatures verify, the algorithm replies with pk and otherwise $\perp$. The following theorem captures the security of WRS with respect to the $\mathsf{WR\text{-}Pop}$ registration protocol.

Theorem 5.4 *Fix $\eta \geq 2$. Let A be an* ruf3-kr *adversary with respect to the* $\mathsf{WR\text{-}Pop}$ *protocol that makes at most q_s signature queries, q_c corruption queries, q_k key registration queries, and runs in time at most t. Then there exists an adversary B such that*

$$\mathbf{Adv}^{\text{ruf3-kr}}_{\mathsf{WRS},\mathsf{WR\text{-}Pop}}(A) \leq \eta^2 \mathbf{Adv}^{\text{uf}}_{\mathsf{W}}(B)$$

where B makes at most q_s signing queries and runs in time at most $t + \mathcal{O}(q_c + (n + q_s + 1)\mathsf{Time}(\mathsf{WRS}))$.

Proof: In fact we consider an ruf3-kr-fr adversary A (with respect to the $\mathsf{WR\text{-}Pop}$ protocol), and will later apply Lemma 5.2 to prove the theorem. We use A to build a uf adversary B against the Waters signature scheme; the adversary B is shown in Figure 12. Adversary B is given input a parameter string and Waters' public key and has access to a Waters' signing oracle **OSign**. First it parses the parameter string and Waters' public key pk^*, notated by the code $(g_1, u, u_1, \ldots, u_n) \leftarrow pk^*$. It generates two WRS public keys. The first, pk_0^*, is composed of g_1 and a set of randomly chosen group elements $a, \vec{a}$. The second, pk_1^*, is composed of the Waters' parameter h and a set of group elements $b, \vec{b}$ chosen to be the Waters' elements $u, \vec{u}$ component-wise divided by the values $a, \vec{a}$. Lastly it generates the public parameters for the $\mathsf{WR\text{-}Pop}$ protocol, programming the values h_0 and h_1 to include the group elements g_1 and h, respectively. The values $w, \vec{w}$ are chosen so that the environment knows their discrete logs (base g). The adversary handles A's queries as follows. For key registration, the POP is simply recorded in a table $\mathtt{P}$, which is initially undefined everywhere. Without loss of generality we assume A only sends valid POPs. For signing queries, A will either query on the set of public keys $\mathcal{V}$ including both pk_0^* and pk_1^* or at least one of the two (recall that because A is legitimate, it will never query on two keys it chose, and all adversarially-chosen keys queried to **ORSign** will have been previously registered). If the former, then B uses its signing oracle to generate a response. Otherwise, it utilizes the POP of the adversarially-chosen public key to assist it in computing a response. The code $(pk_s^*, pk) \leftarrow \mathcal{V}$ means: (1) parse $\mathcal{V}$ as two public keys pk' and pk and (2) let $s \in \{0, 1\}$ be such that $pk' = pk_s^*$. Finally when A finishes, B outputs A's forgery in the case that it verifies and otherwise outputs $\perp$.

We now show that if A outputs a successful forgery, then B does also. The forgery is of the form $(\{pk_0^*, pk_1^*\}, M, (\sigma, \rho))$ and if $\mathsf{W\text{-}RVf}$ verifies then we know that $\mathbf{e}(g_1, h) \cdot \mathbf{e}(Y, \rho) = \mathbf{e}(\sigma, g)$ where $Y = H_{a,\vec{a}}(M) \cdot H_{b,\vec{b}}(M)$. But we have that for any $M \in \{0,1\}^n$ that

$$H_{a,\vec{a}}(M) \cdot H_{b,\vec{b}}(M) = a \cdot b \cdot \prod_{i=1}^{n} (a_i b_i)^{M[i]} = a \cdot \frac{u}{a} \cdot \prod_{i=1}^{n} \left(a_i \cdot \frac{u_i}{a_i} \right)^{M[i]} = u \cdot \prod_{i=1}^{n} (u_i)^{M[i]} = H_{u,\vec{u}}(M)\,. \quad (23)$$

Thus σ, ρ are such that $\mathsf{W\text{-}Vf}^{H_{u,\vec{u}}}(pk^*, M, (\sigma, \rho)) = 1$.

Adversary $B(par, g_1)$
$(h, u, u_1, \ldots, u_n) \leftarrow par$
$a, a_1, \ldots, a_n \xleftarrow{\$} \mathbb{G}^{n+1}$
$pk_0^* \leftarrow (g_1, a, a_1, \ldots, a_n)$
$b \leftarrow u/a$; For $j = 1$ to n do $b_j \leftarrow u_j/a_j$
$pk_1^* \leftarrow (h, b, b_1, \ldots, b_n)$
$\beta_0, \beta_1, x, x_1, \ldots, x_n \xleftarrow{\$} \mathbb{Z}_p^{n+3}$
$h_0 \leftarrow g_1 g^{\beta_0}$; $h_1 \leftarrow h g^{\beta_1}$
$w, \vec{w} \leftarrow g^x, g^{x_1}, \ldots, g^{x_n}$
Run $A(h_0, h_1, w, \vec{w}, (pk_0^*, pk_1^*))$, answering queries by

query ORSign$(s, \mathcal{V}, M)$
If $\mathcal{V} = \{pk_0^*, pk_1^*\}$ then Ret **OSign**(M)
$(pk_s^*, pk) \leftarrow \mathcal{V}$; $(\hat{g}, z, z_1, \ldots, z_n) \leftarrow pk$
$(\pi_0, \varpi_0), (\pi_1, \varpi_1) \leftarrow \mathtt{P}[pk]$
$r \xleftarrow{\$} \mathbb{Z}_p$; $P \leftarrow \varpi_s^{x + \Sigma_{j=1}^n x_j pk[j]}$
If $s = 0$ then $Y \leftarrow H_{z,\vec{z}}(M) \cdot H_{a,\vec{a}}(M)$
Else $Y \leftarrow H_{z,\vec{z}}(M) \cdot H_{b,\vec{b}}(M)$
$\sigma \leftarrow \pi_s \cdot Y^r \cdot \hat{g}^{-\beta_s} \cdot P^{-1}$
Ret (σ, g^r)

query OKReg$(pk, (\pi_0, \varpi_0, \pi_1, \varpi_1))$
$\mathtt{P}[pk] \leftarrow (\pi_0, \varpi_0, \pi_1, \varpi_1)$; Ret pk

A finishes, outputting $(\mathcal{P}, M, (\sigma, \rho))$
$\phi \leftarrow \perp$
If $\mathsf{RVf}(\{pk_0^*, pk_1^*\}, M, (\sigma, \rho)) = 1$ then
  $\phi \leftarrow (M, (\sigma, \rho))$
Ret ϕ

procedure Initialize G0
$u, \vec{u} \xleftarrow{\$} \mathbb{G}^{n+1}$; $\alpha, \delta \xleftarrow{\$} \mathbb{Z}_p$; $h \leftarrow g^\delta$; $g_1 \leftarrow g^\alpha$
$a, a_1, \ldots, a_n \xleftarrow{\$} \mathbb{G}^{n+1}$
$pk_0^* \leftarrow (g_1, a, a_1, \ldots, a_n)$
$b \leftarrow u/a$; For $j = 1$ to n do $b_j \leftarrow u_j/a_j$
$pk_1^* \leftarrow (h, b, b_1, \ldots, b_n)$
$\beta_0, \beta_1, x, x_1, \ldots, x_n \xleftarrow{\$} \mathbb{Z}_p^{n+3}$
$h_0 \leftarrow g_1 g^{\beta_0}$; $h_1 \leftarrow h g^{\beta_1}$
$w, \vec{w} \leftarrow g^x, g^{x_1}, \ldots, g^{x_n}$
Ret $(h_0, h_1, w, \vec{w}, (pk_0^*, pk_1^*))$

procedure ORSign$(s, \mathcal{V}, M)$
If $\mathcal{V} = \{pk_0^*, pk_1^*\}$ then
  Ret $\mathsf{W\text{-}Sign}^{H_{u,\vec{u}}}(h^\alpha, M)$
$(pk_s^*, pk) \leftarrow \mathcal{V}$; $(\hat{g}, z, z_1, \ldots, z_n) \leftarrow pk$
$(\pi_0, \varpi_0), (\pi_1, \varpi_1) \leftarrow \mathtt{P}[pk]$
$r \xleftarrow{\$} \mathbb{Z}_p$; $P \leftarrow \varpi_s^{x + \Sigma_{j=1}^n x_j pk[j]}$
If $s = 0$ then $Y \leftarrow H_{z,\vec{z}}(M) \cdot H_{a,\vec{a}}(M)$
Else $Y \leftarrow H_{z,\vec{z}}(M) \cdot H_{b,\vec{b}}(M)$
$\sigma \leftarrow \pi_s \cdot Y^r \cdot \hat{g}^{-\beta_s} \cdot P^{-1}$
Ret (σ, g^r)

procedure OKReg$(pk, (\pi_0, \varpi_0, \pi_1, \varpi_1))$
$\mathtt{P}[pk] \leftarrow (\pi_0, \varpi_0, \pi_1, \varpi_1)$; Ret pk

procedure Finalize$(\mathcal{P}, M, (\sigma, \rho))$
$\phi \leftarrow \perp$
If $\mathsf{RVf}(\{pk_0^*, pk_1^*\}, M, (\sigma, \rho)) = 1$ then
  $\phi \leftarrow (M, (\sigma, \rho))$
Ret ϕ

Figure 12: Adversary B and game G0 used in the proof of Theorem 5.4.

We proceed through a sequence of games to lower bound B's advantage. The first game is G0, shown in Figure 12. It is exactly like adversary B except that it produces the Waters' scheme parameters and keys itself. Instead of querying an **OSign** oracle it runs W-Sign directly. The output of B and G0 are the same, and since B forges against W (as argued above) any time $\perp$ is not output we have

$$\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{W}}(B) = \Pr\left[\mathrm{G0}(A) \not\Rightarrow \perp\right]. \tag{24}$$

The next game is G1, shown in Figure 13. Instead of computing $\mathsf{W\text{-}Sign}^{H_{u,\vec{u}}}(h^\alpha, M)$ it computes $\mathsf{W\text{-}RSign}(\alpha, \mathcal{V}, M)$. Here α corresponds to the secret key of pk_0^*. In fact this change is without loss because they compute the same values. The former generates $\sigma = h^\alpha \cdot H_{u,\vec{u}}(M)^r$ and $\rho = g^r$ for $r \xleftarrow{\$} \mathbb{Z}_p$. The latter generates $\sigma' = h^\alpha \cdot \Big(H_{a,\vec{a}}(M) \cdot H_{b,\vec{b}}(M)\Big)^r$ and $\rho' = g^r$ for $r \xleftarrow{\$} \mathbb{Z}_p$. Applying (23) to σ' we have that $\sigma' = h^\alpha \cdot H_{u,\vec{u}}(M)^r = \sigma$ and so the computations are equivalent. We have just

procedure Initialize G1

$u, \vec{u} \xleftarrow{\$} \mathbb{G}^{n+1}$; $\alpha, \delta \xleftarrow{\$} \mathbb{Z}_p$; $h \leftarrow g^{\delta}$; $g_1 \leftarrow g^{\alpha}$
$a, a_1, \ldots, a_n \xleftarrow{\$} \mathbb{G}^{n+1}$
$pk_0^* \leftarrow (g_1, a, a_1, \ldots, a_n)$
$b \leftarrow u/a$; For $j = 1$ to n do $b_j \leftarrow u_j/a_j$
$pk_1^* \leftarrow (h, b, b_1, \ldots, b_n)$
$\beta_0, \beta_1, x, x_1, \ldots, x_n \xleftarrow{\$} \mathbb{Z}_p^{n+3}$
$h_0 \leftarrow g_1 g^{\beta_0}$; $h_1 \leftarrow h g^{\beta_1}$
$w, \vec{w} \leftarrow g^x, g^{x_1}, \ldots, g^{x_n}$
Ret $(h_0, h_1, w, \vec{w}, (pk_0^*, pk_1^*))$

procedure ORSign$(s, \mathcal{V}, M)$
If $\mathcal{V} = \{pk_0^*, pk_1^*\}$ then
  Ret $\mathsf{W\text{-}RSign}(\alpha, \mathcal{V}, M)$
$(pk_s^*, pk) \leftarrow \mathcal{V}$; $(\hat{g}, z, z_1, \ldots, z_n) \leftarrow pk$
$(\pi_0, \varpi_0), (\pi_1, \varpi_1) \leftarrow \mathsf{P}[pk]$
$r \xleftarrow{\$} \mathbb{Z}_p$; $P \leftarrow \varpi_s^{x + \sum_{j=1}^n x_j pk[j]}$
If $s = 0$ then $Y \leftarrow H_{z,\vec{z}}(M) \cdot H_{a,\vec{a}}(M)$
Else $Y \leftarrow H_{z,\vec{z}}(M) \cdot H_{b,\vec{b}}(M)$
$\sigma \leftarrow \pi_s \cdot Y^r \cdot \hat{g}^{-\beta_s} \cdot P^{-1}$
Ret (σ, g^r)

procedure OKReg$(pk, (\pi_0, \varpi_0, \pi_1, \varpi_1))$
$\mathsf{P}[pk] \leftarrow (\pi_0, \varpi_0, \pi_1, \varpi_1)$; Ret pk

procedure Finalize$(\mathcal{P}, M, (\sigma, \rho))$
$\phi \leftarrow \perp$
If $\mathsf{RVf}(\{pk_0^*, pk_1^*\}, M, (\sigma, \rho)) = 1$ then
  $\phi \leftarrow (M, (\sigma, \rho))$
Ret ϕ

procedure Initialize G2

$u, \vec{u} \xleftarrow{\$} \mathbb{G}^{n+1}$; $\alpha, \delta \xleftarrow{\$} \mathbb{Z}_p$; $h \leftarrow g^{\delta}$; $g_1 \leftarrow g^{\alpha}$
$a, a_1, \ldots, a_n \xleftarrow{\$} \mathbb{G}^{n+1}$
$pk_0^* \leftarrow (g_1, a, a_1, \ldots, a_n)$
$b \leftarrow u/a$; For $j = 1$ to n do $b_j \leftarrow u_j/a_j$
$pk_1^* \leftarrow (h, b, b_1, \ldots, b_n)$
$\beta_0, \beta_1, x, x_1, \ldots, x_n \xleftarrow{\$} \mathbb{Z}_p^{n+3}$
$h_0 \leftarrow g_1 g^{\beta_0}$; $h_1 \leftarrow h g^{\beta_1}$
$w, \vec{w} \leftarrow g^x, g^{x_1}, \ldots, g^{x_n}$
Ret $(h_0, h_1, w, \vec{w}, (pk_0^*, pk_1^*))$

procedure ORSign$(s, \mathcal{V}, M)$
If $\mathcal{V} = \{pk_0^*, pk_1^*\}$ then
  Ret $\mathsf{W\text{-}RSign}(\alpha, \mathcal{V}, M)$
If $s = 0$ then Ret $\mathsf{W\text{-}RSign}(\alpha, \mathcal{V}, M)$
Ret $\mathsf{W\text{-}RSign}(\delta, \mathcal{V}, M)$

procedure OKReg$(pk, (\pi_0, \varpi_0, \pi_1, \varpi_1))$
$\mathsf{P}[pk] \leftarrow (\pi_0, \varpi_0, \pi_1, \varpi_1)$; Ret pk

procedure Finalize$(\mathcal{P}, M, (\sigma, \rho))$
$\phi \leftarrow \perp$
If $\mathsf{RVf}(\{pk_0^*, pk_1^*\}, M, (\sigma, \rho)) = 1$ then
  $\phi \leftarrow (M, (\sigma, \rho))$
Ret ϕ

Figure 13: Games G1 and G2 used in proof of Theorem 5.4.

argued that

$$\Pr\left[\mathrm{G0}(A) \not\Rightarrow \perp\right] = \Pr\left[\mathrm{G1}(A) \not\Rightarrow \perp\right]. \tag{25}$$

Game G2 simplifies handling of signing queries further. Queries on $\mathcal{V} = \{pk_0^*, pk_1^*\}$ are handled as in game G1. There are two other cases. First if $\mathcal{V} = \{pk_0^*, pk\}$, the response is computed via $\mathsf{W\text{-}RSign}(\alpha, \mathcal{V}, M)$. Second if $\mathcal{V} = \{pk_1^*, pk\}$, the response is computed via $\mathsf{W\text{-}RSign}(\delta, \mathcal{V}, M)$. (Note that δ is the discrete log of h base g, and plays the role of sk_1^*.) We focus on the first case, where $s = 0$, showing that G1 and G2 implement the same signing functionality when queried by A on $\mathcal{V} = \{pk_0^*, pk\}$ and a message M for some adversarially-chosen public key $pk = (\hat{g}, z, \vec{z})$. (The other case is argued similarly.) In G1 such a query is answered by

$$\sigma = \pi_0 \cdot Y^r \cdot \hat{g}^{-\beta_0} \cdot \left(\varpi_0^{x + \sum_{j=1}^n x_j pk[j]}\right)^{-1} \tag{26}$$

and a value g^r where $r \xleftarrow{\$} \mathbb{Z}_p$. Since A is legitimate, the POP tuple (π_0, ϖ_0) verifies under

WR-PopV, meaning that

$$\begin{aligned}\mathbf{e}(\pi_0, g) &= \mathbf{e}(\hat{g}, h_0) \cdot \mathbf{e}(H_{w,\vec{w}}(pk), \varpi_0) \\ &= \mathbf{e}(\hat{g}, g_1 g^{\beta_0}) \cdot \mathbf{e}(H_{w,\vec{w}}(pk), \varpi_0) \\ &= \mathbf{e}(\hat{g}, g_1) \cdot \mathbf{e}(\hat{g}, g^{\beta_0}) \cdot \mathbf{e}(H_{w,\vec{w}}(pk), \varpi_0) \end{aligned} \tag{27}$$

and since $g_1 = g^\alpha$ the properties of $\mathbf{e}$ give that $\mathbf{e}(\hat{g}, g_1) = \mathbf{e}(\hat{g}^\alpha, g)$. Substituting this into (27) and rearranging gives

$$\begin{aligned}\mathbf{e}(\hat{g}^\alpha, g) &= \frac{\mathbf{e}(\pi_0, g)}{\mathbf{e}(\hat{g}, g^{\beta_0}) \cdot \mathbf{e}(H_{w,\vec{w}}(pk), \varpi_0)} \\ &= \frac{\mathbf{e}(\pi_0, g)}{\mathbf{e}(\hat{g}, g^{\beta_0}) \cdot \mathbf{e}(g^{x+\sum_{i=1}^n x_i pk[i]}, \varpi_0)} \\ &= \frac{\mathbf{e}(\pi_0, g)}{\mathbf{e}(\hat{g}^{\beta_0}, g) \cdot \mathbf{e}\left(\varpi_0^{x+\sum_{i=1}^n x_i pk[i]}, g\right)} \\ &= \frac{\mathbf{e}(\pi_0, g)}{\mathbf{e}\left(\hat{g}^{\beta_0} \cdot \varpi_0^{x+\sum_{i=1}^n x_i pk[i]}, g\right)} \\ &= \mathbf{e}\left(\pi_0 \cdot \hat{g}^{-\beta_0} \cdot \left(\varpi_0^{x+\sum_{i=1}^n x_i pk[i]}\right)^{-1}, g\right). \end{aligned} \tag{28}$$

Above we have utilized the properties of $\mathbf{e}$ to derive the chain of equalities. Equation (28) gives that

$$\hat{g}^\alpha = \pi_0 \cdot \hat{g}^{-\beta_0} \cdot \left(\varpi_0^{x+\sum_{i=1}^n x_i pk[i]}\right)^{-1}. \tag{29}$$

and substituting (29) into (26) we see that $\sigma = \hat{g}^\alpha \cdot Y^r$. Investigating the definition of W-RSign, we see that the output of $\mathsf{W\text{-}RSign}(\alpha, \mathcal{V}, M)$ is $(\hat{g}^\alpha \cdot Y^r, g^r)$ when $\mathcal{V} = \{pk_0^*, pk\}$ and where $r \xleftarrow{\$} \mathbb{Z}_p$. Thus the handling of this query in G1 results in the same response as the code handling this query in G2. The other case can be argued similarly, giving that

$$\Pr\left[\mathrm{G1}(A) \not\Rightarrow \perp\right] = \Pr\left[\mathrm{G2}(A) \not\Rightarrow \perp\right]. \tag{30}$$

In game G3, we further simplify the **ORSign** procedure, always returning exactly the ring signature requested by A. Since WRS is perfectly ranon-ind anonymous there is no distinction between the handling of signing queries in G2 and in G3. Thus,

$$\Pr\left[\mathrm{G2}(A) \not\Rightarrow \perp\right] = \Pr\left[\mathrm{G3}(A) \not\Rightarrow \perp\right]. \tag{31}$$

In the final game G4 we re-write the **Initialize** procedure, computing values as shown in Figure 14. The distributions of the variables returned by the procedure remain unchanged, giving that

$$\Pr\left[\mathrm{G3}(A) \not\Rightarrow \perp\right] = \Pr\left[\mathrm{G4}(A) \not\Rightarrow \perp\right]. \tag{32}$$

Now however we note that G4 exactly implements the ruf3-kr-fr experiment with respect to the WR-Pop protocol. Thus we have that $\Pr\left[\mathrm{G4}(A) \not\Rightarrow \perp\right] = \mathbf{Adv}^{\text{ruf3-kr-fr}}_{\mathsf{WRS},\mathsf{WR\text{-}Pop}}(A)$ and combining all the above game transitions we have that $\mathbf{Adv}^{\text{ruf3-kr-fr}}_{\mathsf{WRS},\mathsf{WR\text{-}Pop}}(A) \leq \mathbf{Adv}^{\text{uf}}_{\mathsf{W}}(B)$.

The adversary B runs in time at most the running time of A plus the overhead of the simulation. This includes $\mathcal{O}(n \cdot \mathsf{Time}(\mathsf{WRS}))$ operations, which is the time to initialize the public keys; $\mathcal{O}(q_s \mathsf{Time}(\mathsf{WRS}))$ operations, for simulating signing queries; $\mathcal{O}(q_c)$ operations, for handling registration queries; and a single $\mathsf{Time}(\mathsf{WRS})$ for the verification procedure. Thus the total running time is at most $t + \mathcal{O}(q_c + (n + q_s + 1)\mathsf{Time}(\mathsf{WRS}))$.

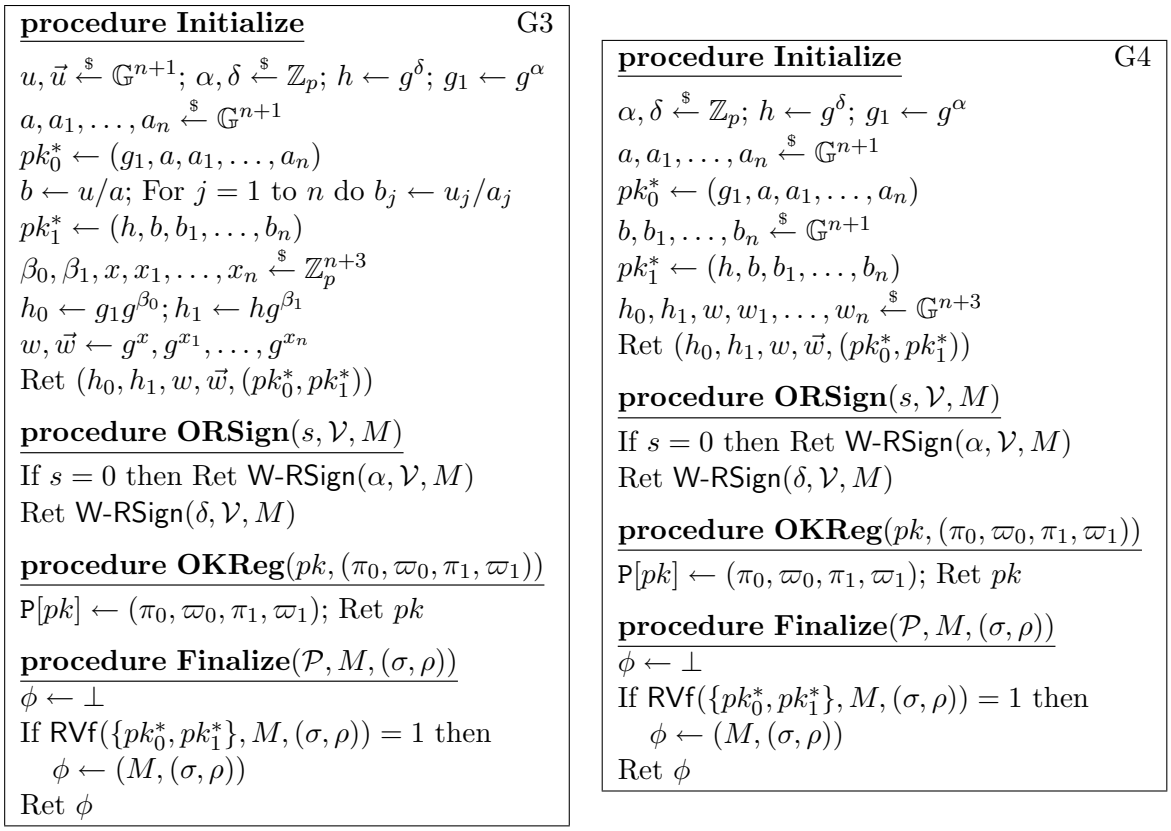

Figure 14: Games G3 and G4 used in proof of Theorem 5.4.

We can now apply Lemma 5.2, which adds a factor ν^2 but does not require additional time over that already used by adversary B. ∎

Acknowledgements

The authors thank Mihir Bellare for suggesting that they investigate the security of multisignature schemes when the proof of knowledge is replaced by a proof of possession akin to ones currently used in PKIs, and for many useful discussions. The authors thank the anonymous reviewers for their helpful comments.

References

[1] C. Adams, S. Farrell, T. Kause, T. Mononen. Internet X.509 public key infrastructure certificate management protocols (CMP). Request for Comments (RFC) 4210, Internet Engineering Task Force (September 2005)

[2] N. Asokan, V. Niemi, P. Laitinen. On the usefulness of proof-of-possession. In *Proceedings of the 2nd Annual PKI Research Workshop.* (2003) 122–127

[3] M. Bellare. CSE 208: Advanced Cryptography. UCSD course (Spring 2006).

[4] M. Bellare, O. Goldreich. On Defining Proofs of Knowledge. In *Advances in Cryptology – CRYPTO '92.* Volume 740 of *Lecture Notes in Computer Science*, pp. 390–420, E. Brickell ed., Springer-Verlag, 1993

[5] M. Bellare, T. Kohno, V. Shoup. Stateful public-key cryptosystems: how to encrypt with one 160-bit exponentiation. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, pp. 380–389, ACM, 2006

[6] M. Bellare, C. Namprempre, G. Neven. Unrestricted aggregate signatures. Cryptology ePrint Archive, Report 2006/285 (2006) `http://eprint.iacr.org/`.

[7] M. Bellare, G. Neven. Multi-signatures in the plain public-key model and a generalized forking lemma. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS),* pp. 390–399, ACM, 2006

[8] M. Bellare, P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS),* pp. 62–73, ACM, 1993

[9] M. Bellare, P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *Advances in Cryptology – EUROCRYPT '06.* Volume 4004 of *Lecture Notes in Computer Science*, pp. 409–426, S. Vaudenay ed., Springer-Verlag, 2006

[10] A. Bender, J. Katz, R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *Third Theory of Cryptography Conference – TCC '06.* Volume 3876 of *Lecture Notes in Computer Science*, pp. 60–79, Springer-Verlag, 2006

[11] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography – PKC '03.* Volume 2567 of *Lecture Notes in Computer Science*, pp. 31–46, Springer-Verlag, 2003

[12] A. Boldyreva, M. Fischlin, A. Palacio, B. Warinschi. A closer look at PKI: security and efficiency. In *Proceedings of the 10th International Conference on Practice and Theory in Public Key Cryptography – PKC '07,* Volume 4450 of *Lecture Notes in Computer Science*, T. Okamoto, X. Wang eds., Springer-Verlag, 2007

[13] D. Boneh, C. Gentry, B. Lynn, H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology – EUROCRYPT '03.* Volume 2656 of *Lecture Notes in Computer Science*, pp. 416–432, E. Biham ed., Springer-Verlag, 2003

[14] D. Boneh, B. Lynn, H. Shacham. Short signatures from the weil pairing. In *Advances in Cryptology – ASIACRYPT '01.* Volume 2248 *Lecture Notes in Computer Science*, pp. 514–532, C. Boyd ed., Springer-Verlag, 2001

[15] J.S. Coron. On the exact security of full domain hash. In *Advances in Cryptology – CRYPTO '00.* Volume 1880 of *Lecture Notes in Computer Science*, pp. 229–235, M. Bellare ed., Springer-Verlag, 2000

[16] A. De Santis, G. Persiano. Zero-knowledge proofs of knowledge without interaction (extended abstract). In *33rd Annual Symposium on Foundations of Computer Science – FOCS '92,* pp. 427–436, IEEE, 1992

[17] M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *Advances in Cryptology – CRYPTO '05.* Volume 3621 of *Lecture Notes in Computer Science*, pp. 152–168, V. Shoup ed., Springer-Verlag, 2005

[18] S. Goldwasser, S. Micali, R. Rivest. A Paradoxical Solution to the Signature Problem. In *25th Annual Symposium on Foundations of Computer Science – FOCS '84,* pp. 441–449, IEEE, 1984

[19] J. Groth, R. Ostrovsky, A. Sahai. Perfect non-interactive zero knowledge for NP. In *Advances in Cryptology – EUROCRYPT '06.* Volume 4004 of *Lecture Notes in Computer Science*, pp. 339–358, C. Dwork ed., Springer-Verlag, 2006

[20] L. Harn. Group-oriented (t,n) threshold digital signature scheme and digital multisignature. Computers and Digital Techniques, IEEE Proceedings **141**(5) (1994) 307–313

[21] J. Herranz. *Some digital signature schemes with collective signers.* Ph.D. Thesis, Universitat Politècnica De Catalunya, Barcelona. April, 2005.

[22] P. Horster, M. Michels, H. Petersen. Meta signature schemes based on the discrete logarithm problem. In *IFIP TC11 Eleventh International Conference on Information Security (IFIP/SEC 1995)* (1995) 128–141

[23] M. Jakobsson, K. Sako, R. Impagliazzo. Designated verifier proofs and their applications. In *Advances in Cryptology – EUROCRYPT '96.* Volume 1070 of *Lecture Notes in Computer Science*, pp. 143–154, U. Maurer ed., Springer-Verlag, 1996

[24] F. Laguillaumie, D. Vergnaud. Designated verifier signatures: anonymity and efficient construction from any bilinear map. In *Security in Communication Networks, 4th International Conference, SCN 2004.* Volume 3352 of *Lecture Notes in Computer Science*, pp. 105–119, Springer, 2005

[25] S.K. Langford. Weakness in some threshold cryptosystems. In *Advances in Cryptology – CRYPTO '96.* Volume 1109 of *Lecture Notes in Computer Science*, pp. 74–82, N. Koblitz ed., Springer-Verlag, 1996

[26] C.M. Li, T. Hwang, N.Y. Lee. Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders. In *Advances in Cryptology – EUROCRYPT '94.* Volume 950 of *Lecture Notes in Computer Science*, pp. 194–204, A. DeSantis ed., Springer-Verlag, 1995

[27] H. Lipmaa, G. Wang, F. Bao. Designated verifier signature schemes: attacks, new security notions and a new construction. In *32nd International Colloquium on Automata, Languages and Programming – ICALP 2005.* Volume 3580 *Lecture Notes in Computer Science*, pp. 459–471, Springer-Verlag, 2005

[28] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, B. Waters. Sequential aggregate signatures and multisignatures without random oracles. In *Advances in Cryptology – EUROCRYPT '06.* Volume 4004 of *Lecture Notes in Computer Science*, pp. 465–485, S. Vaudenay ed., Springer-Verlag, 2006

[29] A. Lysyanskaya, S. Micali, L. Reyzin, H. Shacham. Sequential Aggregate Signatures from Trapdoor Permutations. In *Advances in Cryptology – EUROCRYPT '04.* Volume 3027 of *Lecture Notes in Computer Science*, pp. 74–90, C. Cachin, J. Camenisch eds., Springer-Verlag, 2004

[30] M. Michels, P. Horster. On the risk of disruption in several multiparty signature schemes. In *Advances in Cryptology – ASIACRYPT '96.* Volume 1163 of *Lecture Notes in Computer Science*, pp. 334–345, K. Kim, T. Matsumoto eds., Springer-Verlag, 1996

[31] S. Micali, K. Ohta, L. Reyzin. Accountable-subgroup multisignatures. In *Proceedings of the 8th ACM Conference on Computer and Communications Security – CCS '01,* pp. 245–254, ACM, 2001

[32] K. Ohta, T. Okamoto. A digital multisignature scheme based on the Fiat-Shamir scheme. In *Advances in Cryptology – ASIACRYPT '91.* Volume 739 of *Lecture Notes in Computer Science*, pp. 139–148, H. Imai, R. Rivest, T. Matsumoto eds., Springer-Verlag, 1993

[33] K. Ohta, T. Okamoto. Multi-signature schemes secure against active insider attacks. IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences E82-A(1) (1999) 21–31

[34] R.L. Rivest, A. Shamir, Y. Tauman. How to leak a secret. In *Advances in Cryptology – ASIACRYPT '01.* Volume 2248 of *Lecture Notes in Computer Science*, pp. 552–565, C. Boyd ed., Springer-Verlag, 2001

[35] RSA Laboratories: RSA PKCS #10 v1.7: Certification Request Syntax Standard `ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-10/pkcs-10v1_7.pdf`.

[36] J. Schaad. Internet X.509 public key infrastructure certificate request message format (CRMF). Request for Comments (RFC) 4211, Internet Engineering Task Force (September 2005)

[37] H. Shacham, B. Waters. Efficient ring signatures without random oracles. In *Proceedings of the 10th International Conference on Practice and Theory in Public Key Cryptography – PKC '07,* Volume 4450 of *Lecture Notes in Computer Science*, T. Okamoto, X. Wang eds., Springer-Verlag, 2007

[38] R. Steinfeld, L. Bull, H. Wang, J. Pieprzyk. Universal designated-verifier signatures. In *Advances in Cryptology – ASIACRYPT '03.* Volume 2894 of *Lecture Notes in Computer Science*, pp. 523–542, C.S. Laih ed., Springer-Verlag, 2003

[39] B. Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology – EUROCRYPT '05.* Volume 3494 of *Lecture Notes in Computer Science*, pp. 114–127, R. Cramer ed., Springer-Verlag, 2005

A Ring Signature Anonymity Definitions

We recall the strongest definition from [10], anonymity with respect to full key exposure. Let $\mathsf{RS} = (\mathsf{RPg}, \mathsf{RKg}, \mathsf{RSign}, \mathsf{RVf})$ be a ring signature scheme and A be an adversary. The experiment $\mathbf{Exp}^{\text{ranon-fke-b}}_{\mathsf{RS}}(A)$ is defined in Figure 15. Note that we only allow adversaries which make one query to **ORSignLR**. We define the ranon-fke advantage of A by

$$\mathbf{Adv}^{\text{ranon-fke}}_{\mathsf{RS}}(A) = \Pr\left[\mathbf{Exp}^{\text{ranon-fke-0}}_{\mathsf{RS}}(A) \Rightarrow 1\right] - \Pr\left[\mathbf{Exp}^{\text{ranon-fke-1}}_{\mathsf{RS}}(A) \Rightarrow 1\right]$$

We compare this definition to the ranon-ind anonymity definition given in Section 5. Recall that, intuitively, the ranon-ind definition requires that no adversary can determine which secret key was used to sign, even if the adversary itself chooses all the keys. First we show a separation, that there exist schemes which are secure in the ranon-fke sense but not in the ranon-ind sense.

Theorem A.1 *Let* $\mathsf{RS} = (\mathsf{RPg}, \mathsf{RKg}, \mathsf{RSign}, \mathsf{RVf})$ *be a ring signature scheme and let A, B, and C be* ranon-ind*,* ranon-fke*, and* ruf3-kr *adversaries, respectively. Then there exists a ring signature scheme* RS' *such that*

$$\begin{aligned}
\mathbf{Adv}^{\text{ranon-ind}}_{\mathsf{RS}'}(A) &= 1 \\
\mathbf{Adv}^{\text{ranon-fke}}_{\mathsf{RS}'}(B) &= \mathbf{Adv}^{\text{ranon-fke}}_{\mathsf{RS}}(B) \\
\mathbf{Adv}^{\text{ruf3-kr}}_{\mathsf{RS}'}(C) &= \mathbf{Adv}^{\text{ruf3-kr}}_{\mathsf{RS}}(C)
\end{aligned}$$

Proof: Let $\mathsf{RS} = (\mathsf{RPg}, \mathsf{RKg}, \mathsf{RSign}, \mathsf{RVf})$ be a ring signature scheme. Define a new ring signature scheme $\mathsf{RS}' = (\mathsf{RPg}, \mathsf{RKg}', \mathsf{RSign}', \mathsf{RVf}')$ as follows. Parameter generation is equivalent. Key generation first runs $(pk, sk) \xleftarrow{\$} \mathsf{RKg}$ and then returns $(pk, sk \,||\, 0)$. Signature generation $\mathsf{RSign}_{sk \,||\, d}(\mathcal{V}, M)$ works by first running $\sigma \xleftarrow{\$} \mathsf{RSign}_{sk}(\mathcal{V}, M)$ and then outputing $\sigma \,||\, d$. Verification of $(\mathcal{V}, M, \sigma')$ works by first dropping the last bit of σ' to get σ, and then returning $\mathsf{RVf}(\mathcal{V}, M, \sigma)$.

Now we show that there exists a ranon-ind adversary A that has advantage one against RS'. The adversary runs $(pk_0, sk_0), (pk_1, sk_1) \xleftarrow{\$} \mathsf{RKg}$ and then lets $sk'_0 = sk_0 \,||\, 0$ and $sk'_1 = sk_0 \,||\, 1$. It then queries $\mathbf{ORSignLR}(\{sk_0, sk_1\}, \{pk_0, pk_1\}, M)$ for any message M. The adversary outputs the last bit of the returned signature. Clearly its advantage is one.

It is easy to verify that any ranon-fke adversary B has equivalent advantage against either RS or RS' and similarly for any ruf3-kr adversary C (relative to any key registration protocol). ∎

Experiment $\mathbf{Exp}^{\text{ranon-fke}}_{\mathsf{RS}}(A)$
$b \xleftarrow{\$} \{0,1\}$; $par \xleftarrow{\$} \mathsf{RPg}$
For $i = 1$ to η do $(pk_i, sk_i) \xleftarrow{\$} \mathsf{RKg}(par)$
$\mathcal{K} = \{sk_i \mid i \in [1\,..\,\eta]\}$
Run $A(par, (pk_i)_1^\eta))$ handling oracle queries as follows
ORSign$(s, \mathcal{V}, M)$, where $s \in [1\,..\,\eta]$; $pk_s \in \mathcal{V}$
  Ret $\mathsf{RSign}_{sk_s}(\mathcal{V}, M)$
ORSignLR$(\{i_0, i_1\}, \mathcal{V}, M)$, where $i_0, i_1 \in [1\,..\,\eta]$; $i_0 \neq i_1$; $pk_{i_0}, pk_{i_1} \in \mathcal{V}$
  Ret $(\mathcal{K}, \mathsf{RSign}_{sk_{i_b}}(\mathcal{V}, M))$
A outputs b'
Ret $b' = b$

Figure 15: Ring signature full-key exposure anonymity experiment.

On the other hand, it is straightforward to see that security in the ranon-ind sense implies security in the ranon-fke sense. Briefly, assume we have an adversary A against the anonymity of a ring signature scheme in the full key exposure experiment. We can build an adversary B that uses A to distinguish in the ranon-ind game. The adversary B picks η key pairs pk_i, sk_i and give them to A. Then, it simulates, using these secret keys, adversary A's signing and corruption oracles in the natural way. Finally A outputs a pair of indices i_0, i_1, a message M, and a target group $\mathcal{V}$. In turn, B queries its left-or-right oracle on $(sk_{i_0}, sk_{i_1}, \mathcal{V}, M)$ and forwards the response to A. When A outputs a bit, the adversary B returns it.