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

Thomas Ristenpart and Scott Yilek

Dept. of Computer Science & Engineering 0404, University of California San Diego
9500 Gilman Drive, La Jolla, CA 92093-0404, USA
`{tristenp,syilek}@cs.ucsd.edu`
`http://www-cse.ucsd.edu/users/{tristenp,syilek}`

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

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, 38, 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 [40]), the LOSSW sequential aggregate signature scheme [28], and many designated-verifier signature schemes [23, 39, 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 [36], RFC 4210 [1], RFC 4211 [37]) 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. Here 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 [40]. (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 [35]. 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 [38].

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 [38] 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, 39, 27, 24].

2 Preliminaries

Basic notation. We denote string concatenation by $||$. Let S be any set. Then we define $S \overset{\cup}{\leftarrow} s$ for any appropriate value s as $S \leftarrow S \cup \{s\}$. For a multiset $\mathcal{S}$, let $\mathcal{S} - \{s\}$ denote the multiset $\mathcal{S}$ with one instance of element s removed. For multisets $\mathcal{S}$ and R, let $\mathcal{S} \backslash \mathcal{R}$ be the multiset formed by repeatedly executing $\mathcal{S} \leftarrow \mathcal{S} - \{r\}$ for each $r \in \mathcal{R}$ (including duplicates). We define $s \overset{\$}{\leftarrow} S$ as sampling uniformly from S and $s \overset{\$}{\leftarrow} A(x_1, x_2, \ldots)$ assigns to s the result of running A on fresh random coins and the inputs $x_1, x_2, \ldots$. For any string M, let $M[i]$ denote the i^{th} bit of M. For a table $\mathtt{H}$, let $\mathtt{H}[s]$ 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 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 $\mathbf{e}$: $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ 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: $\mathbb{G}_2 \rightarrow \mathbb{G}_1$. 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, 38] 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

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

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 $\mathsf{S} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Ver})$ consists of 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. 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

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

where the probability is over the coins used by Kg, Sign, and A.

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.

```
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
    $\mathsf{OMSign}(\mathcal{V}, M)$, 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.
    $\mathsf{OKReg}$: Simulate a new instance of algorithm $\mathsf{RegV}$, forwarding messages to
        and from $A$. If the instance's final message is $pk \neq \bot$, 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
```

Fig. 1. Multisignature security experiment in the registered key model.

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 central authority runs the (randomized) parameter generation algorithm MPg to create a parameter string par that is given to all parties and is an (usually implicit) input to the other three algorithms. The (randomized) 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 two rounds: 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.

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 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 $\mathsf{Reg} = \mathsf{Kosk}$.

In the case of two-round multisignature schemes, the multisignature oracle can be simplified: it just computes the honest parties' share of the multisignature and outputs it. Furthermore, we assume without loss of generality that adversaries never output a forgery on a message previously queried to their signing oracle and that they always output a forgery with $\mathcal{V}$ including the trusted party's public key.

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 $H\colon \{0,1\}^* \to \mathbb{G}_1$ be a random oracle. 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:

```
B-Kg:
    $sk \xleftarrow{\$} \mathbb{Z}_p$; $pk \leftarrow g^{sk}$
    Return $(pk, sk)$

B-Sign$^H(sk, M)$:
    Return $H(M)^{sk}$

B-Vf$^H(pk, M, \sigma)$:
    If $\mathbf{e}(H(M), pk) = \mathbf{e}(\sigma, g)$ then
        Return 1
    Return 0
```

The $\mathsf{BMS} = (\mathsf{B\text{-}MPg}, \mathsf{B\text{-}MKg}, \mathsf{B\text{-}MSign}, \mathsf{B\text{-}MVf})$ multisignature scheme [11] is a simple extension of BLS signatures. Parameter generation just selects the groups, generators, and pairings as described in Section 2. Key generation, using the global parameters, creates a key pair as in B-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 $\mathcal{V}, M, \sigma$ the verification algorithm B-MVf computes $PK = \prod_{i=1}^{v} pk_i$ and then runs $\mathsf{B\text{-}Vf}^H(PK, M, \sigma)$, 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 $H_{\mathrm{pop}}\colon \{0,1\}^* \to \mathbb{G}_1$ 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 $pk \,||\, \mathsf{B\text{-}Sign}^{H_{\mathrm{pop}}}(sk, \langle pk \rangle)$ and algorithm B-PopV, upon receiving (pk, π) computes $\mathsf{B\text{-}Vf}^{H_{\mathrm{pop}}}(pk, \langle pk \rangle, \pi)$ 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 and H_{pop} as long as domain separation is enforced. The following theorem captures the security of BMS with respect to this key registration protocol.

Theorem 1. *Let $H, H_{\text{pop}}\colon \{0,1\}^* \to \mathbb{G}_1$ 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, H_{pop}, 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 $t' \in \mathcal{O}(t \log t + (q_h + q_{\text{pop}} + v)t_E + (q_k+1)t_{\mathsf{e}})$.

Proof. We wish to construct a co-CDH adversary B, which on input g, X, h utilizes an msuf-kr adversary A 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]. Without loss of generality, we assume that A always queries $H(M)$ before querying B-Sign(M). Likewise we assume that A always queries $H_{\text{pop}}(\langle pk\rangle)$ before querying OKReg(pk, π) for any π. Figure 2 details a sequence of four games. The game G0, which does not include the boxed statements, represents the core of our adversary B.

The execution G0(A) proceeds as follows. First Initialize is executed, which initializes several variables, including a co-CDH problem instance (g, X, h). Then A is run with input X. Oracle queries by A are handled as shown. Game G0 programs H (lazily built using an initially empty table $\mathtt{H}[\cdot]$) to sometimes return 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 query will correspond to the forgery message. G0 programs H_{pop} (lazily built using an initially empty table $\mathtt{H}_{\mathsf{p}}[\cdot]$) to 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 invoke an execution of B-PopV, utilizing the $\mathtt{H}_{\mathsf{p}}$ table for the algorithm's random oracle. Successful registrations have the POP signature stored in the table $\mathtt{P}$ (which is initially set everywhere to $\perp$). Once A halts with output a potential forgery the Finalize procedure is executed. We define the subroutine CheckForgery (not explicitly shown in the games for brevity) as follows. It checks that all keys in the multiset have an entry defined in $\mathtt{P}$ except the honest user's key (though if there are multiple copies of the honest user's key, then $\mathtt{P}[pk^*]$ must not be $\perp$). Then it checks if the multisignature verifies under the multiset of public keys given. If either check fails it returns zero, otherwise it returns one. Note that in the game x is not used beyond defining the co-CDH problem instance.

The adversary B, when run on input (g, X, h), follows exactly the steps of G0(A), except that it uses its co-CDH problem instance to supply the appropriate values. We now must justify that $\mathbf{Adv}^{\text{co-cdh}}_{(\mathbb{G}_1,\mathbb{G}_2)}(B) = \Pr\left[\text{G0}(A) \not\Rightarrow \perp\right]$ where G0$(A) \not\Rightarrow \perp$ means that the output of G0's Finalize procedure is not $\perp$. By construction the behavior of G0(A) and B are equivalent, and thus all that remains to be shown is that if the variable G0(A) does not output $\perp$, then it outputs the co-CDH solution h^x. Let us fix some more 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 $\beta_i = \alpha - \gamma_i = \mathtt{B}[pk_i]$. Now,

```
G0 $\boxed{\text{G1}}$

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

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

On query B-Sign$(M)$:
Let $k$ be such that $M = M_k$
$S_k \leftarrow 1$
If $\delta_k = 1$ then $S_k \leftarrow X_1^{\alpha_k}$
Else $bad \leftarrow$ true $\boxed{; S_k \leftarrow \mathtt{H}[M]^x}$
Return $S_k$

On query $H_{\mathrm{pop}}(N)$:
$\mathtt{B}[N] \xleftarrow{\$} \mathbb{Z}_p$; Return $\mathtt{H_p}[N] \leftarrow h g_1^{\mathtt{B}[N]}$

On query OKReg$(pk, \pi)$:
If B-Vf$^{\mathtt{H_p}}(pk, \langle pk \rangle, \pi) = 1$ then
    $\mathtt{P}[pk] \leftarrow \pi$; Return $pk$
Return $\perp$

procedure Finalize$(\{X, pk_1, \ldots, pk_d\}, M, \sigma)$
$f \leftarrow$ CheckForgery$(\{X, pk_1, \ldots, pk_d\}, M, \sigma)$
If $f = 0$ then Return $\perp$
Let $k$ be such that $M = M_k$; $\alpha \leftarrow \alpha_k$
For each $i \in [1\,..\,d]$ do $\gamma_i \leftarrow \alpha - \mathtt{B}[\langle pk_i \rangle]$
$w \leftarrow \perp$
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 $bad \leftarrow$ true $\boxed{; w \leftarrow h^x}$
Return $w$
```

```
G2

procedure Initialize
$x \xleftarrow{\$} \mathbb{Z}_p$; $X \leftarrow g^x$; $h \xleftarrow{\$} \mathbb{G}_1^*$; $c \leftarrow 0$
Return $X$

On query $H(M)$:
$c \leftarrow c+1$; $M_c \leftarrow M$
$\alpha_c \xleftarrow{\$} \mathbb{Z}_p$; $\delta_c \xleftarrow{\delta} \{0,1\}$
Return $\mathtt{H}[M] \xleftarrow{\$} \mathbb{G}_1$

On query B-Sign$(M)$:
Let $k$ be such that $M = M_k$
if $\delta_k = 1$ then $S_k \leftarrow \mathtt{H}[M]^x$
Else $bad \leftarrow$ true; $S_k \leftarrow \mathtt{H}[M]^x$
Return $S_k$

On query $H_{\mathrm{pop}}(N)$:
Return $\mathtt{H_p}[N] \xleftarrow{\$} \mathbb{G}_1$

On query OKReg$(pk, \pi)$:
If B-Vf$^{\mathtt{H_p}}(pk, \langle pk \rangle, \pi) = 1$ then
    $\mathtt{P}[pk] \leftarrow \pi$; Return $pk$
Return $\perp$

procedure Finalize$(\{X, pk_1, \ldots, pk_d\}, M, \sigma)$
$f \leftarrow$ CheckForgery$(\{X, pk_1, \ldots, pk_d\}, M, \sigma)$
If $f = 0$ then Return $\perp$
Let $k$ be such that $M = M_k$; $\alpha \leftarrow \alpha_k$
If $\delta_k = 0$ then $w \leftarrow h^x$
Else $bad \leftarrow$ true; $w \leftarrow h^x$
Return $w$
```

```
G3

procedure Initialize
$x \xleftarrow{\$} \mathbb{Z}_p$; $X \leftarrow g^x$; $h \xleftarrow{\$} \mathbb{G}_1^*$; $c \leftarrow 0$
Return $X$

On query $H(M)$:
Return $\mathtt{H}[M] \xleftarrow{\$} \mathbb{G}_1$

On query B-Sign$(M)$:
$c \leftarrow c+1$; Return $H[M]^x$

On query $H_{\mathrm{pop}}(N)$:
Return $\mathtt{H_p}[N] \xleftarrow{\$} \mathbb{G}_1$

On query OKReg$(pk, \pi)$:
If B-Vf$^{\mathtt{H_p}}(pk, \langle pk \rangle, \pi) = 1$ then
    $\mathtt{P}[pk] \leftarrow \pi$; Return $pk$
Return $\perp$

procedure Finalize$(\{X, pk_1, \ldots, pk_d\}, M, \sigma)$
$f \leftarrow$ CheckForgery$(\{X, pk_1, \ldots, pk_d\}, M, \sigma)$
If $f = 0$ then Return $\perp$
For each $j \in [1\,..\,c]$ do
    $\delta_j \xleftarrow{\delta} \{0,1\}$; If $\delta_j = 0$ then $bad \leftarrow$ true
$\delta_{j+1} \xleftarrow{\delta} \{0,1\}$; If $\delta_{j+1} = 1$ then $bad \leftarrow$ true
Return $h^x$
```

Fig. 2. Games used in proof that BMS is secure using POPs.

because CheckForgery returns one if $\mathrm{G0}(A)$ does not output $\perp$, we necessarily have that $\mathbf{e}(\mathtt{H}[M], PK) = \mathbf{e}(\sigma, g)$ and that $\mathbf{e}(\mathtt{H}_{\mathsf{p}}[\langle pk_i \rangle], pk_i) = \mathbf{e}(\pi_i, g)$ for each $i \in [1\,..\,d]$. In turn this means that $\sigma = (hg_1^{\alpha})^{x+sk_1+\ldots+sk_d}$ and $\pi_i = (hg_1^{\beta_i})^{sk_i}$ for each $i \in [1\,..\,d]$. Thus, we can see that $w = h^x$:

$$w = \sigma X_1^{-\alpha} \prod_{i=1}^{d} \pi_i^{-1}\,\psi(pk_i)^{-\gamma_i} = \frac{(hg_1^{\alpha})^{x+sk_1+\ldots+sk_d}}{X_1^{\alpha} \cdot \prod (hg_1^{\beta_i})^{sk_i} (g_1^{sk_i})^{\alpha-\beta_i}} = h^x\,.$$

Now we move through a sequence of games to lower bound the probability that $\mathrm{G0}(A)$ actually succeeds in terms of A's advantage. Let Good be the event that *bad* is never set to true. What we show is that

$$\begin{aligned}
\Pr[\mathrm{G0}(A) \not\Rightarrow \perp] \geq \Pr[\mathrm{G0}(A) \not\Rightarrow \perp \wedge \mathsf{Good}] &= \Pr[\mathrm{G1}(A) \not\Rightarrow \perp \wedge \mathsf{Good}] && (1)\\
&= \Pr[\mathrm{G2}(A) \not\Rightarrow \perp \wedge \mathsf{Good}] && (2)\\
&= \Pr[\mathrm{G3}(A) \not\Rightarrow \perp \wedge \mathsf{Good}] && (3)\\
&= \Pr[\mathrm{G3}(A) \not\Rightarrow \perp] \cdot \Pr[\mathsf{Good}] && (4)\\
&\geq \mathbf{Adv}^{\mathrm{msuf}}_{\mathsf{BMS}}(A) \cdot \frac{1}{e} \cdot \frac{1}{q_s+1} && (5)
\end{aligned}$$

which implies the theorem statement. Now to justify this sequence of equations. ► Game G0 and G1 are identical-until-*bad*. A variant [6] of the fundamental lemma of game-playing [9] justifies Equation 1. ► Game G2 simplifies game G1 by taking advantage of knowing x. Queries to H are always answered with values uniformly chosen from $\mathbb{G}_1$. Signature queries are always answered with $\mathtt{H}[M]^x$. The value h^x is always returned by Finalize. These changes mean we never need g_1 and X_1, so they are omitted. The only distinction between G2 and G1 then is how these values are computed; their distributions remain the same and we therefore have justified Equation 2. ► We now note that in game G2 the values chosen for the δ_c variables have no impact on any of the values returned by procedures in the game, and only affect the setting of *bad*. 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. With these facts in mind, we modify G2 to get game G3, in which we defer all possible settings of *bad* until the Finalize procedure. We only perform δ-biased coin tosses $c+1$ times: one for each signature query and one for the forgery. Equation 3 is justified by the fact that none of these changes affect the other variables in the game. (We also make some other cosmetic changes to simplify the games, but these do not modify distributions involved.) ► It is clear in game G3 that the event Good and "$\mathrm{G3}(A) \not\Rightarrow \perp$" are independent, justifying Equation 4. ► Lastly, we note that G3 now exactly represents the environment of $\mathbf{Exp}^{\mathrm{msuf\text{-}pop}}_{\mathsf{BMS}}(A)$ because if $\mathrm{G3}(A)$ does not output $\perp$ then A's output is a valid forgery. The lower bound $\Pr[\mathsf{Good}] \geq (e(q_s+1))^{-1}$ is standard (see, e.g. [6, 15]).

The adversary B runs A. Additionally B must perform an exponentiation for each H and H_{pop} query and one for each key in the forgery set $\mathcal{V}$. Finally B must perform a pairing for each OKReg query and to verify the forgery. Thus B runs in time $t' \in \mathcal{O}(t \log t + (q_h + q_{\mathrm{pop}} + v)t_E + (q_k+1)t_{\mathbf{e}})$ where $|\mathcal{V}| = v$. □

4.2 Multisignatures Based on Waters Signatures

Waters signatures and multisignatures. Let $H\colon \{0,1\}^n \to \mathbb{G}_1$ be a hash function and define the signature scheme $\mathsf{W} = (\mathsf{W\text{-}Kg}, \mathsf{W\text{-}Sign}, \mathsf{W\text{-}Vf})$ as shown below.

```
W-Kg:
    $\alpha \xleftarrow{\$} \mathbb{Z}_p$; $sk \leftarrow h^\alpha$
    $pk \leftarrow \mathbf{e}(h, g)^\alpha$
    Return $(pk, sk)$
```

```
W-Sign$^H(sk, M)$:
    $r \xleftarrow{\$} \mathbb{Z}_p$; $\rho \leftarrow g^r$
    $\sigma \leftarrow sk \cdot H(M)^r$
    Return $(\sigma, \rho)$
```

```
W-Vf$^H(pk, M, (\sigma, \rho))$:
    If $\mathbf{e}(\sigma, g) \cdot \mathbf{e}(H(M), \rho)^{-1} = pk$ then
        Return 1
    Return 0
```

Although one could use a random oracle for H, we can avoid the random oracle model by using the following hash function, as done in [28]. A trusted party, in addition to picking h, chooses $u, u_1, \ldots, u_n \xleftarrow{\$} \mathbb{G}_1$ and publishes them globally. Define $H_{u,\vec{u}}\colon \{0,1\}^n \to \mathbb{G}_1$ by $H_{u,\vec{u}}(M) = u \cdot \prod_{i=1}^n u_i^{M[i]}$. For simplicity we restrict ourselves to the message space $\{0,1\}^n$, but in practice we 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 chooses $h, u, \vec{u}$ as specified above in addition to fixing all the groups, generators, and pairings as per Section 2. Key generation, using the generated parameters, computes keys as in W-Kg. To generate a multisignature for multiset $\mathcal{V} = \{pk_1, \ldots, pk_v\}$, each participant i computes $(\sigma_i, \rho_i) \xleftarrow{\$} \mathsf{W\text{-}Sign}(sk_i, M)$ and broadcasts (σ_i, ρ_i). The multisignature is $(\prod_{i=1}^v \sigma_i, \prod_{i=1}^v \rho_i)$. To verify a signature (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 $\mathsf{W\text{-}Vf}(PK, M, (\sigma, \rho))$. This scheme was proven secure using the KOSK assumption in [28].

The WM-Pop protocol. Let $w, w_1, \ldots, w_n \xleftarrow{\$} \mathbb{G}_1$ 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 W-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 WM-PopP takes as input (pk, sk) and sends $pk \,||\, (\pi, \varpi)$ where $(\pi, \varpi) = \mathsf{W\text{-}Sign}^{H_{w,\vec{w}}}(sk, \langle pk \rangle_n)$. Algorithm WM-PopV receives $pk \,||\, (\pi, \varpi)$ 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, proof of which is given in the full version of the paper [34], and Theorem 2 in [28] (security of WMS under the KOSK assumption) establish the security of WMS under WM-Pop.

Theorem 2. *Let A be an* msuf-kr*-adversary, with respect to the* WM-Pop *protocol, that runs in time t, makes q_s signing 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*

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

and where B runs in time $t' \in \mathcal{O}(t \log t + n t_E + (t_E + t_\psi + t_{\mathbf{e}}) q_k)$ and makes q_s signature queries.

4.3 Attacks against Standardized Key Registration Protocols

We show how the standardized proof-of-possession based key registration protocols (as per PKCS#10 [36] and RFCs 4210/4211 [1,37]) 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 $pk \mid\mid \mathsf{B\text{-}Sign}^H(sk, \langle pk \rangle)$ 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 $\mathsf{B\text{-}MSign}$ and $\mathsf{B\text{-}MVf}$.

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 $\mathsf{OMSign}(\{pk^*\}, \langle pk \rangle)$, receiving $\sigma = H(\langle pk \rangle)^{sk^*}$. Then A sets $\pi \leftarrow H(\langle pk \rangle)^s/\sigma$ and registers with $pk \mid\mid \pi$. 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 $\mathsf{B\text{-}Pop}$ and $\mathsf{WM\text{-}Pop}$ 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 apply to such protocols, also, see the full version for details.

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 string *par* given to all parties and (often implicitly) input to the other three algorithms. 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 set 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 $|\mathcal{V}| \geq 2$ and all keys in $\mathcal{V}$ are distinct. It outputs a ring signature. 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 set of public keys, and for any valid sk with a $pk \in \mathcal{V}$. Ring signatures that only allow rings of size κ are called *κ-user ring signatures.*

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 the experiment $\mathbf{Exp}_{\mathsf{RS}}^{\text{r-anon-ind-b}}(A)$ works as follows: it runs $par \xleftarrow{\$} \mathsf{RPg}$ and then runs $A(par)$, giving it a left-or-right oracle $\mathsf{ORSignLR}(\cdot,\cdot,\cdot)$. The oracle takes queries of the form $\mathsf{ORSignLR}(\mathcal{S}, \mathcal{V}, M)$ where $\mathcal{S} = (sk_0, sk_1)$ and $\mathcal{V} = pk_0, \ldots, pk_{v-1}$ is a set of public keys such that (pk_0, sk_0) and (pk_1, sk_1) are valid key pairs. The oracle returns $\mathsf{RSign}_{sk_b}(\mathcal{V}, M)$. Finally A outputs a bit b', and wins if $b = b'$. The r-anon-ind advantage of A is

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

We say a scheme is *perfectly r-anon-ind anonymous* if the advantage of any adversary is zero.

The r-anon-ind definition is stronger than the strongest definition given in [10] (see the full version for details). Even so, both of the BKM 2-user ring signature schemes meet it, and are, in fact, perfectly r-anon-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. Figure 3 gives the security experiment for the strongest definition of security lifted to the registered key model, r-uf3-kr, which represents resistance to rogue-key attacks. A weaker definition, r-uf2, is obtained by defining an experiment $\mathbf{Exp}_{\mathsf{RS}}^{\text{r-uf2}}(A)$ that is the same as $\mathbf{Exp}_{\mathsf{RS},\mathsf{Reg}}^{\text{r-uf3-kr}}(A)$ except we do not allow the adversary to choose its own public keys. We also omit the key registration oracle and remove the requirement in ORSign that all adversarily chosen keys must be in $\mathcal{R}$. Lastly, we weaken this definition one step further by defining $\mathbf{Exp}_{\mathsf{RS}}^{\text{r-uf1}}(A)$, which disallows corruption queries. We thus define the following advantages:

- $\mathbf{Adv}_{\mathsf{RS},\mathsf{Reg}}^{\text{r-uf3-kr}}(A) = \Pr[\mathbf{Exp}_{\mathsf{RS},\mathsf{Reg}}^{\text{r-uf3-kr}}(A) \Rightarrow 1]$ (rogue-key attacks, equivalent to Definition 7 in [10] when $\mathsf{Reg} = \mathsf{Plain}$)
- $\mathbf{Adv}_{\mathsf{RS}}^{\text{r-uf2}}(A) = \Pr[\mathbf{Exp}_{\mathsf{RS}}^{\text{r-uf2}}(A) \Rightarrow 1]$ (corruption attacks, similar to a definition in [21])
- $\mathbf{Adv}_{\mathsf{RS}}^{\text{r-uf1}}(A) = \Pr[\mathbf{Exp}_{\mathsf{RS}}^{\text{r-uf1}}(A) \Rightarrow 1]$ (chosen subring attacks, Definition 6 in [10])

For κ-user ring signatures that meet the strongest anonymity definition, we have that security against corruption attacks is actually implied by security against chosen subring attacks. The reduction is tighter for small κ. This stems from having to guess a particular ring out of the η potential participants in the proof. The proof is given in the full version.

```
Experiment $\mathbf{Exp}^{\text{r-uf3-kr}}_{\mathsf{RS},\mathsf{Reg}}(A)$
    $par \xleftarrow{\$} \mathsf{RPg}$; $(pk_i, sk_i) \xleftarrow{\$} \mathsf{RKg}(par)$ for $i \in [1\,..\,\eta]$;  $\mathcal{S} \leftarrow \{pk_1, \ldots, pk_\eta\}$
    $\mathcal{Q} \leftarrow \mathcal{C} \leftarrow \mathcal{R} \leftarrow \emptyset$
    Run $A(par, \mathcal{S})$ handling oracle queries as follows
    $\mathsf{ORSign}(s, \mathcal{V}, M)$, 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 Return $\perp$
        Return $\mathsf{RSign}_{sk_s}(\mathcal{V}, M)$
    $\mathsf{OCorrupt}(i)$, where $i \in [1\,..\,\eta]$: $\mathcal{C} \xleftarrow{\cup} pk_i$;  Return $sk_i$
    $\mathsf{OKReg}$: Simulate a new instance of algorithm $\mathsf{RegV}$, forwarding messages to
        and from $A$. If the instance's last message is $pk \neq \perp$, then $\mathcal{R} \xleftarrow{\cup} pk$.
    $A$ outputs $(\mathcal{V}, M, \sigma)$
    If $\mathsf{RVf}(\mathcal{V}, M, \sigma) = 1 \wedge (\,(\mathcal{V}, M) \notin \mathcal{Q}\,) \wedge (\,\mathcal{V} \subseteq \mathcal{S} \setminus \mathcal{C}\,)$ then Return 1
    Return 0
```

Fig. 3. Ring signature unforgeability experiment in the registered key model.

Theorem 3. *Let* RS *be a* κ*-user ring signature scheme. Let* $\eta \geq \kappa$ *be some number and let* A *be an* r-uf2 *adversary that makes at most* q_s *signature queries,* q_c *corruption queries, and runs in time at most* t*. Then there exists adversaries* B_a *and* B_u *such that*

$$\mathbf{Adv}^{\text{r-uf2}}_{\mathsf{RS}}(A) \leq \binom{\eta}{\kappa} \mathbf{Adv}^{\text{r-anon-ind}}_{\mathsf{RS}}(B_a) + \binom{\eta}{\kappa} \mathbf{Adv}^{\text{r-uf1}}_{\mathsf{RS}}(B_u)$$

where B_a *uses* q_s *queries and runs in time* $t_a \in \mathcal{O}(t \log t + (\eta+1)\mathsf{Time}(\mathsf{RS}))$ *and* B_u *uses* q_s *queries and runs in time* $t_u \in \mathcal{O}(t \log t + (\eta + 1 + q_s)\mathsf{Time}(\mathsf{RS}))$*.*

Using KOSK. Using the key registration protocol Kosk, any scheme that is unforgeable with respect to corruption attacks (r-uf2) and meets our strong definition of anonymity is also secure against rogue-key attacks (r-uf3-kr). Note that this result (unlike the last) applies to any ring signature scheme, not just κ-user ring signature schemes.

Theorem 4. *Fix* η *and let* RS *be a ring signature scheme for which* $t_{\mathcal{K}}$ *is the maximal time needed to validate a key pair. Let* A *be an* r-uf3-kosk *adversary that makes at most* (q_s, q_c, q_k) *signature queries, corruption queries, and registration queries, and runs in time at most* t*. Then there exists adversaries* B_a *and* B_u *such that*

$$\mathbf{Adv}^{\text{r-uf3-kr}}_{\mathsf{RS},\mathsf{Kosk}}(A) \leq \mathbf{Adv}^{\text{r-anon-ind}}_{\mathsf{RS}}(B_a) + \mathbf{Adv}^{\text{r-uf2}}_{\mathsf{RS}}(B_u)$$

where B_a *runs in time* $t_a \in \mathcal{O}(t \log t + (\eta + 1)\mathsf{Time}(\mathsf{RS}))$*, using at most* q_s *sign queries, and* B_u *runs in time* $t_u \in \mathcal{O}(t \log t + q_k t_{\mathcal{K}} + (\eta + 1 + q_s)\mathsf{Time}(\mathsf{RS}))$*, using at most* q_s *sign queries and* q_c *corrupt queries.*

The proof is given in the full version. We can apply Theorem 3 and then Theorem 4 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.

USING 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 $\mathsf{WRS} = (\mathsf{W\text{-}RPg}, \mathsf{W\text{-}RKg}, \mathsf{W\text{-}RSign}, \mathsf{W\text{-}RVf})$. The parameter generation selects groups, generators, and a pairing in the symmetric setting as per Section 2. The key generation algorithm $\mathsf{W\text{-}RKg}$ chooses $\alpha \xleftarrow{\$} \mathbb{Z}_q$, sets $g_1 \leftarrow g^\alpha$, and chooses random elements $u, u_1, \ldots, u_n \xleftarrow{\$} \mathbb{G}_1^*$. Finally it outputs $pk \leftarrow g_1, u, u_1, \ldots, u_n$ and $sk \leftarrow \alpha$. Define $\mathsf{W\text{-}RSign}_{sk}(\{pk, pk'\}, M)$ as follows. (Without loss we assume sk corresponds to pk.) Parse pk as $g_1, u, \vec{u}$ and pk' as $g_1', u', \vec{u}'$ and let $H'(M) = H_{u,\vec{u}}(M) \cdot H_{u',\vec{u}'}(M)$. Finally, return $\mathsf{W\text{-}Sign}^{H'}(g_1'^{sk}, M)$. The verification algorithm $\mathsf{W\text{-}RVf}(\{pk, pk'\}, M, (\sigma, \rho))$ first parses pk as $g_1, u, u_1, \ldots, u_n$ and pk' as $g_1', u', u_1', \ldots, u_m'$ and defines H' as in signature generation. Then it outputs one if $\mathbf{e}(g_1, g_1') \cdot \mathbf{e}(H'(M), \rho) = \mathbf{e}(\sigma, g)$.

In [10] the scheme is proven secure against r-uf1 adversaries, but as shown in [38] the scheme is *not* secure against rogue-key attacks without key registration. We now show a simple proof-of-possession based registration protocol to render the scheme secure. Choose global parameters $h_0, h_1, w, w_1, \ldots, w_n \xleftarrow{\$} \mathbb{G}_1$ (this will require trusted setup, and could be accomplished with $\mathsf{W\text{-}RPg}$). 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 (sk, pk) and sends $pk \,||\, (\pi_0, \varpi_0, \pi_1, \varpi_1)$ which is simply 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 security of WRS with respect to the $\mathsf{WR\text{-}Pop}$ registration protocol. The proof is given in the full version.

Theorem 5. *Fix η. Let A be an* r-uf3-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*

$$\mathbf{Adv}^{\text{r-uf3-kr}}_{\mathsf{WRS},\mathsf{WR\text{-}Pop}}(A) \leq \eta^2 \mathbf{Adv}^{\text{uf}}_{\mathsf{W}}(B)$$

where B makes at most q_s signing queries and runs in time $t_B \in \mathcal{O}(t \log t + \eta t_E + q_s t_E + (q_k + 1) t_{\mathbf{e}})$.

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. The first author is supported by NSF grant CNS–0524765. The second author is supported by NSF grant CNS–0430595 and a Jacobs School Fellowship.

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 *CRYPTO '92.* Volume 740 of *LNCS*, Springer (1993) 390–420
5. M. Bellare, T. Kohno, V. Shoup. Stateful public-key cryptosystems: how to encrypt with one 160-bit exponentiation. ACM Conference on Computer and Communications Security. (2006) 380–389
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 *ACM Conference on Computer and Communications Security.* (2006) 390–399
8. M. Bellare, P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security.* (1993) 62–73
9. M. Bellare, P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In *EUROCRYPT '06.* Volume 4004 of *LNCS*, Springer (2006) 409–426
10. A. Bender, J. Katz, R. Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In *TCC '06.* Volumne 3876 of *LNCS*, Springer (2006) 60–79
11. A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *PKC '03.* Volume 2567 of *LNCS*, Springer (2002) 31–46
12. A. Boldyreva, M. Fischlin, A. Palacio, B. Warinschi. A closer look at PKI: security and efficiency. Public Key Cryptography (2007), to appear.
13. D. Boneh, C. Gentry, B. Lynn, H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *EUROCRYPT '03.* Volume 2656 of *LNCS*, Springer (2003) 416–432
14. D. Boneh, B. Lynn, H. Shacham. Short signatures from the weil pairing. In *ASIACRYPT '01.* Volume 2248 *LNCS*, Springer (2001) 514–532
15. J.S. Coron. On the exact security of full domain hash. In *CRYPTO '00.* Volume 1880 of *LNCS*, Springer (2000) 229–235
16. A. De Santis, G. Persiano. Zero-knowledge proofs of knowledge without interaction (extended abstract). In *FOCS '92.* IEEE (1992) 427–436
17. M. Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In *CRYPTO '05.* Volume 3621 of *LNCS*, Springer (2005) 152–168
18. S. Goldwasser, S. Micali, R. Rivest. A Paradoxical Solution to the Signature Problem. In *FOCS '84.* IEEE (1984) 441–449.
19. J. Groth, R. Ostrovsky, A. Sahai. Perfect non-interactive zero knowledge for NP. In *EUROCRYPT '06.* Volume 4004 of *LNCS*, Springer (2006) 339–358
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 *EUROCRYPT '96.* Volume 1070 of *LNCS*, Springer (1996) 143–154
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 *LNCS*, Springer (2005) 105–119
25. S.K. Langford. Weakness in some threshold cryptosystems. In *CRYPTO '96.* Volume 1109 of *LNCS*, Springer (1996) 74–82
26. C.M. Li, T. Hwang, N.Y. Lee. Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders. In *EUROCRYPT '94.* Volume 950 of *LNCS*, Springer (1995) 194–204
27. H. Lipmaa, G. Wang, F. Bao. Designated verifier signature schemes: attacks, new security notions and a new construction. In *ICALP 2005.* Volume 3580 *LNCS*, Springer (2005) 459–471
28. S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, B. Waters. Sequential aggregate signatures and multisignatures without random oracles. In *EUROCRYPT '06.* Volume 4004 of *LNCS*, Springer (2006) 465–485
29. A. Lysyanskaya, S. Micali, L. Reyzin, H. Shacham. Sequential Aggregate Signatures from Trapdoor Permutations. In *EUROCRPYT '04.* Volume 3027 of *LNCS*, Springer (2004) 74–90
30. M. Michels, P. Horster. On the risk of disruption in several multiparty signature schemes. In *ASIACRYPT '96.* Volume 1163 of *LNCS*, Springer (1996) 334–345
31. S. Micali, K. Ohta, L. Reyzin. Accountable-subgroup multisignatures. In *ACM Conference on Computer and Communications Security.* (2001) 245–254
32. K. Ohta, T. Okamoto. A digital multisignature scheme based on the Fiat-Shamir scheme. In *ASIACRYPT '91.* Volume 739 of *LNCS*, Springer (1993) 139–148
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. T. Ristenpart, S. Yilek. The power of proofs-of-possession: securing multiparty signatures against rogue-key attacks. Full version of current paper. `http://www.cse.ucsd.edu/users/tristenp/`
35. R.L. Rivest, A. Shamir, Y. Tauman. How to leak a secret. In *ASIACRYPT '01.* Volume 2248 of *LNCS*, Springer (2001) 552–565
36. RSA Laboratories: RSA PKCS #10 v1.7: Certification Request Syntax Standard `ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-10/pkcs-10v1_7.pdf`.
37. J. Schaad. Internet X.509 public key infrastructure certificate request message format (CRMF). Request for Comments (RFC) 4211, Internet Engineering Task Force (September 2005)
38. H. Shacham, B. Waters. Efficient ring signatures without random oracles. Public Key Cryptography (2007), to appear.
39. R. Steinfeld, L. Bull, H. Wang, J. Pieprzyk. Universal designated-verifier signatures. In *ASIACRYPT '03.* Volume 2894 of *LNCS*, Springer (2003) 523–542
40. B. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT '05.* Volume 3494 of *LNCS*, Springer (2005) 114–127