Parts of this work previously appeared at ACM CCS 2006 [BN06] and CT-RSA 2007 [BN07]. This is the full version.

New Multi-Signature Schemes and a General Forking Lemma

Mihir Bellare[1] Gregory Neven[2]

July 2005

[1] Department of Computer Science & Engineering, University of California San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. Email: `mihir@cs.ucsd.edu`. URL: `http://www.cs.ucsd.edu/users/mihir`.
[2] Department of Electrical Engineering, Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, B-3001 Heverlee-Leuven, Belgium, and Département d'Informatique, Ecole normale supérieure, 45 rue d'Ulm, 75005 Paris, France. Email: `Gregory.Neven@esat.kuleuven.be`. URL: `http://www.neven.org`.

Abstract

A multi-signature scheme enables a group of signers to produce a compact, joint signature on a common document, and has many potential uses. However, existing schemes impose key setup or PKI requirements that make them impractical, such as requiring a dedicated, distributed key generation protocol amongst potential signers, or assuming strong, concurrent zero-knowledge proofs of knowledge of secret keys done to the CA at key registration. These requirements limit the use of the schemes. We provide a new scheme that is proven secure in the plain public-key model, meaning requires nothing more than that each signer has a (certified) public key. Furthermore, the important simplification in key management achieved is not at the cost of efficiency or assurance: our scheme matches or surpasses known ones in terms of signing time, verification time and signature size, and is proven secure in the random-oracle model under a standard (not bilinear map related) assumption. The proof is based on a simplified and general Forking Lemma that may be of independent interest. We also present a second, slightly less efficient scheme whose security is tightly related to the decisional Diffie-Hellman problem.

Since verification of a multi-signature still requires knowledge of all public keys, and since identity strings are likely to be much shorter than randomly generated public keys, the identity-based paradigm is particularly appealing for the case of multi-signatures. We present and prove secure an identity-based multi-signature scheme based on RSA, which in particular does not rely on (the rather new and untested) assumptions related to bilinear maps.

Contents

1 Introduction

Consider entities $1, \ldots, N$, each having a public key and corresponding secret key. A multi-signature (MS) scheme allows any subset $L \subseteq \{1, \ldots, N\}$ of them, at any time, to engage in an interactive protocol whose output is a joint signature on a message m of their choice. Verification can be done by any external party given just L, m, the purported multi-signature σ, and the public keys of all signers in L. Such a system could be useful for contract signing, co-signing, or distribution of a certificate authority.

A trivial way to implement a multi-signature scheme is to let the multi-signature σ of message m be the list $(\sigma_i : i \in L)$ where σ_i is i's signature on m. This multi-signature is however large, in particular of size proportional to the number $|L|$ of signers. There are several important practical reasons for which this is costly and undesirable. For example, on wireless devices such as PDAs, cell phones, RFID chips and sensors, battery life is the main limitation. Communicating even one bit of data uses significantly more power than executing one 32-bit instruction [BA03]. Reducing the number of bits to communicate saves power and is important to increase battery life. Also, in many settings, communication is not reliable, and so the fewer the number of bits one has to communicate, the better. For such reasons, we want multi-signature schemes that are non-trivial, meaning the size of the multi-signature is about the same as that of a single ordinary signature and in particular not proportional to the number $|L|$ of signers.

Rogue-key attacks. The early literature of MS schemes [IN83, Har94, LHL95, HMP95, Lan96, MH96, OO91, OO99] features numerous attacks breaking proposed schemes. In most cases, this was due to weaknesses related to key setup, in particular the ability to mount a *rogue-key attack*. In such an attack, an adversary who is a group member (insider) chooses its public key as a function of that of honest users in such a way that it can then easily forge multi-signatures. Although they might at first hearing sound far-fetched, rogue-key attacks are in fact possible to mount in practice and are a real threat. When, eventually, precise definitions [MOR01] and proven secure schemes [MOR01, Bol03, LOS$^+$06] emerged, they obviously paid a lot of attention to key setup. These schemes were, happily, proven secure against rogue-key attacks, but, unhappily, at the cost of complexity and expense in the scheme, or using unrealistic and burdensome assumptions on the public-key infrastructure (PKI), as we will now explain in detail.

Drawbacks of previous schemes. The MS-MOR [MOR01] scheme requires, as a pre-processing step, that the set of *potential* signers engages in an interactive key generation protocol that provides to each a public and secret key. The purpose of the protocol is to ensure that no dishonest player can choose its public key as a function of public keys of honest players. This type of *dedicated key generation* is however not practical for several reasons. First, it means the group of potential signers is restricted to being static: it must be decided and fixed before signing can start, and new signers cannot be added later. However, in practice the potential set of signers is dynamic and may not be known ahead of time, and we would like to be able to add potential signers at will. Second, the key generation protocol of [MOR01] is expensive and results in large, complex public keys. In particular, the public key of each signer has a size that depends on the number N of potential signers. (However, once keys are established, multi-signature generation and verification are actually attractively cheap.)

The drawback of the MS-Bo [Bol03] and MS-LOSSW [LOS$^+$06] MS schemes is that the security model makes the *knowledge of secret key* (KOSK) assumption. There is no dedicated key generation, but, when the adversary, mounting a rogue-key attack, provides a public key of its choice for a group member, it is required, in the model, to provide also a matching secret key. Of course "real" adversaries would not do any such thing, so what does this mean? It is explained by the authors as modeling the assumption that a user provides the certification authority (CA) with a proof of knowledge of its secret key before the CA certifies the corresponding public key. However, it is not that simple. The current implementation of these proofs is represented by standards PKCS#10 [PKC00] —used by VeriSign— and RFC 4210/11 [AFKM05, Sch05]. Here the proof is implemented by having the user send the CA a signature, under the public key it is attempting to get certified, of some message that includes the public key and the user identity. While such methods might intuitively seem to prove knowledge of the secret key, they do *not* suffice to realize the abstract model of [Bol03, LOS+06] in which the attacker actually hands the challenger the secret keys. In particular, not only does this type of proof of possession not suffice to prove secure the MS-Bo and MS-LOSSW schemes, but there are actually attacks against these schemes if such proofs of possession are used [BRY06]. (That is, if we attempt to drop the KOSK assumption and substitute these proofs of possession instead.) To obtain proofs of possession sufficient to implement the KOSK assumption, it appears one should use zero-knowledge (ZK) proofs of knowledge (POKs) that meet strong, formal extractability requirements [BG92]. However, that is not all. Since in practice we would expect that the users may register keys at any time, concurrently with other users or with executions of the multi-signature protocol, one would require ZK POKs extractable under such concurrent conditions. This eliminates many standard protocols, including standard POKs of discrete logarithms. Still, it is true that protocols with the desired properties do seem to exist. For example, in the random-oracle model one could use [Fis05] or, in the standard model, non-interactive ZK POKs [DP92]. But the first is not cheap and the second is prohibitive, and even if one were willing, adding them to the PKI requires modifying the client and CA functioning and software, which is difficult, costly and preferably avoided. Also, one is still left with the task of actually formally justifying a claim that this would implement the abstract KOSK model of [Bol03, LOS+06]. (We are not making such a claim here.) However, the main reason this route is impractical is simply that CAs do not right now implement such POKs. If, today, some corporation or person wishes to implement a MS scheme, it seems unlikely that VeriSign is going to oblige them by suddenly changing their offerings to include appropriate POKs of secret keys at registration.

Our MS-DL scheme. In summary, the most significant practical obstacle to multi-signatures at present is the key-setup requirements or assumptions of previous works. Our first contribution is to remove this obstacle by presenting a multi-signature scheme in the *plain public-key model.* This means that, with regard to key setup, nothing more is required than in *any* usage of public-key cryptography, namely that any potential signer has a public key. There is no dedicated key generation protocol. A signer is *not* assumed to have proved knowledge of its secret key to the CA, but only to have a standard certificate. Yet, security against rogue-key attacks is proved *without* the KOSK assumption.

To elaborate, in our setting, the group of potential signers is dynamic: anyone possessing a (certified) public key can join at any time. In our security model, the adversary can corrupt a signer and choose its public key as a function of those of other (honest) signers. It is not required to supply the challenger with a matching secret key, meaning we prove security even when the adversary does not know the secret key underlying a public key it makes for itself. The fact that we do not need to assume any kind of proof-of-knowledge of the secret key performed to the CA at the time a public key is registered and a certificate is obtained reduces the demands on the PKI and allows our protocols to be implemented within the current PKI. CAs need not take any special actions or change their functioning or software. Indeed, a CA does not need to even know that a key is to be used in our multi-signature scheme; it can be treated like any other key.

MS-DL is based on the Schnorr [Sch91] scheme and is proven secure in the random-oracle (RO) model [BR93] assuming hardness of the standard discrete logarithm problem. (MS-Bo and MS-MOR also use the RO model, but MS-LOSSW does not.) MS-DL allows secure concurrent executions of signing protocols by different subsets of the set of potential signers, which is important because applications on the Internet are inherently placed in a concurrent execution environment. Concurrent signing is explicitly disallowed in [MOR01]. Security of our scheme is proved even when the adversary can control the scheduling and mount rogue-key attacks, yet without the KOSK assumption.

A scheme with tight security. Being based on the Schnorr signature scheme, our MS-DL scheme unfortunately inherits its loose reduction to the discrete logarithm problem. We present a second scheme called MS-DDH based on a signature scheme of Katz and Wang [KW03] that has a *tight* security reduction to the DDH problem. Even though requiring slightly more computations than MS-DL, the MS-DDH could end up being more efficient because one can use smaller security parameters. Note that *none* of the previous MS schemes of [MOR01, Bol03, LOS$^+$06] had tight reductions to the underlying hard problem.

Identity-based multi-signatures. To verify the validity of a multi-signature, one still needs the public keys of all cosigners. In most applications these public keys will have to be transmitted along with the multi-signature, possibly with their associated certificate chains. This partially defeats the primary purpose of using a multi-signature scheme, namely to save on bandwidth. The inclusion of some information that uniquely identifies the cosigners is inevitable for verification, but usually signers can be identified more succinctly than by their public keys, e.g. by their user names or IP addresses.

Identity-based signatures. In an identity-based signature scheme [Sha85], the public key of a user is simply his identity, e.g. his name, email or IP address. A trusted key distribution center provides each signer with the secret signing key corresponding to his identity. When all signers have their secret keys issued by the same key distribution center, individual public keys become obsolete, removing the need for explicit certification and all associated costs. These features make the identity-based paradigm particularly appealing for use in conjunction with multi-signatures, leading to the concept of identity-based multi-signature (IBMS) schemes.

Generic constructions. In spite of their appeal with regard to applications, implementations of IBMS schemes are rather limited. As demonstrated in [DKXY03, BNN04], any standard signature scheme can be transformed into an identity-based one using the "certification paradigm". One can attempt to derive IBMS schemes from existing standard MS schemes via this approach [GHK06]. The problem is that the resulting multi-signature is not compact due to the need to include the certificates with each signature. Even if the signatures in the certificates can be aggregated [BGLS03], the public keys they contain cannot. In summary, unlike the case of standard signatures, there seems no trivial, general way to transform compact signature schemes into identity-based ones.

An existing construction. The only provably secure IBMS scheme known today is due to Gentry and Ramzan [GR06]. The scheme employs groups with bilinear maps (also known as pairings), which are usually implemented by modified Weil or Tate pairing over elliptic or hyperelliptic curves. To avoid putting all our eggs in the same basket, it is common practice in cryptography to try to find alternative constructions of a primitive based on different assumptions. While pairings have turned out extremely useful in the design of cryptographic protocols, they were only recently brought to the attention of cryptographers [Jou00], and hence did not yet enjoy the same exposure to cryptanalytic attacks by experts as other, older problems from number theory such as discrete logarithms, factoring and RSA. This exposure is necessary to build confidence in the hardness of the underlying problems; without it, their use in high-security applications may not be advisable.

Also, efficient implementations of RSA are ubiquitous, even in the public domain, while implementations of pairings are much harder to come by. Unlike RSA, even building an inefficient prototype implementation of pairings is far from straightforward for anyone but an expert, and even then it is often difficult or impossible to generate curves with the desired security parameters [GPS06]. Companies may have invested in expensive hardware or software implementations of RSA, and may

Scheme	**Sign**	**Verify**	$\|\mathbf{sig}\|$	$\|\mathbf{pk}\|$	$\|\mathbf{par}\|$	**Key setup**	**Assump**
MS-MOR	1 exp	1 exp	$2 \cdot 160$	$[3 + 2\lg(n_{\max})] \cdot 160$	160	dedicated key-reg	DL
MS-Bo	1 exp	2 pr	160	$6 \cdot 160$	$6 \cdot 160$	KOSK model	(co)CDH
MS-LOSSW	3 exp	2 pr	$7 \cdot 160$	$6 \cdot 160$	$162 \cdot 160$	KOSK model	(co)CDH
MS-DL	1 exp	1 exp	$2 \cdot 160$	160	160	plain pk model	DL
MS-DDH	2 exp	2 exp	$3 \cdot 160$	$2 \cdot 160$	160	plain pk model	DDH
IBMS-GR	2 exp	3 pr	$2 \cdot 160$	0	$6 \cdot 160$ **?**	ID-based	(co)CDH
IBMS-GQ	1 exp	1 exp	$1024 + 160$	0	$1024 + 160$	ID-based	RSA

Table 1: **MS scheme comparisons.** We compare the efficiency of our MS-DL and MS-DDH schemes against the MS-MOR [MOR01], MS-Bo [Bol03], and MS-LOSSW [LOS$^+$06] schemes, and that of our identity-based IBMS-GQ scheme against the IBMS-GR [GR06] scheme. For each scheme we show the computational cost of signing (per signer), the computational cost of verification of a multi-signature, the size of a multi-signature, the size of the public key of an individual signer, the size of the system parameters common to all signers, the type of key-setup, and the assumption used to prove security. All sizes are in bits. By "exp" we mean an exponentiation. (Some of the exponentiations are actually multi-exponentiations, but these have the same cost as single exponentiations.) $n_{\max}$ is the total number of signers in the system. By "pr" we mean a pairing, whose cost estimate is 6–20 exponentiations. We assume we work over a 160-bit elliptic-curve (EC) group for the DL-based schemes. For the coCDH-based schemes we assume an asymmetric pairing, that is, $\mathbf{e}\colon\ \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with $\mathbb{G}_1 \neq \mathbb{G}_2$ (this to make the signatures as short as possible) and an isomorphism $\psi\colon\ \mathbb{G}_2 \to \mathbb{G}_1$ (this to make the proofs go through) with group-element representation sizes in $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ being, respectively, 160-bits, $6 \cdot 160$ bits and $6 \cdot 160$ bits, which is what is needed, in this asymmetric-with-isomorphism setting, to provide the 1024-bit RSA level of security achieved by 160-bit EC groups [GPS06]. For RSA-based schemes, we assume a modulus N and public exponent e of sizes 1024 and 160 bits, respectively.

be reluctant to reinvest in new pairing implementations.

Efficiency and other scheme attributes. The significant gains in key setup achieved by our MS scheme are not at the cost of performance. As Table 1 indicates, our first scheme, denoted MS-DL, compares favorably with previous ones in terms of signing time, verifying time, signature size and other attributes. We now discuss the information in the table in a little more depth.

Unlike [Bol03, LOS$^+$06], we do not use pairings (bilinear maps), which not only results, for us, in greater efficiency and ease of implementation, but also means we do not rely on the relatively new and untested hardness assumptions related to pairing-based cryptography. Verification in MS-DL is cheaper than in MS-Bo, and both signing and verification are cheaper in MS-DL than in MS-LOSSW. We have included the system parameter size in the table mainly to note that this is very large (25,920 bits) for MS-LOSSW, unlike for any other scheme. Our signatures are 320 bits as opposed to the 160 of MS-Bo because the latter uses the pairing-based BSL [BSL01] short signature scheme, but our gain in verification time (by a factor of 12-40) more than compensates. The public keys in our scheme are shorter than in any other scheme, an important benefit in case they have to be transmitted with the multi-signature. Additionally, of course, MS-DL is in the plain public-key model. Our efficiency estimates do not take security into account, meaning that all schemes are not necessarily compared at the same level of security. We do this as we do not think our analyses are tight (that is, the real security is better) and thus comparing at the same level would be misleading for practice.

New Forking Lemma. Our proof of security of the MS-DL scheme relies on a generalization of the Forking Lemma of [PS00] that may be of independent interest. The original Forking Lemma of

Pointcheval and Stern [PS00] applies to signature schemes obtained from three-move identification schemes via the Fiat-Shamir [FS87] transform in the random oracle model. Roughly it says that in an *expected* $O(1/\epsilon)$ repeated executions of a forger A with success probability ϵ, one can find two accepting conversations that agree in the first prover move but not the verifier challenge, leading, via the special soundness property of Σ protocols, to recovery of the secret key and hence a proof of security of the signature scheme in the RO model. This lemma has been important in proving security of signature schemes via the rewinding technique. However, the lemma seems hard (if not impossible) to apply in situations like ours where we are not dealing with a regular signature scheme but a MS scheme. Indeed, in the past, variants of the lemma had to be formulated and proved for different types of signatures. (For example, a version for blind signatures is in [PS00], and one for ring signatures in [HS03]. Another variant is in [**?**].) The statement of our Forking Lemma (cf. Lemma 3.1), in contrast to previous ones, makes no mention of signatures or even, for that matter, random oracles. Rather it asserts a simple lower bound on the probability that two executions of an arbitrary algorithm on certain (related) inputs both accept. This statement, we feel, distills the probabilistic essence of the Forking Lemma and divorces it from any particular application context. (In our view, the Forking Lemma is something purely probabilistic, not about signatures. Previous Forking Lemmas mixed these things up.) In this form, it can be be applied not only to prove security of regular signature schemes but also, as we show, to prove security of schemes like ours where the setting is more complex. Our Forking Lemma also provides worst-case rather than expected-time guarantees on the constructed algorithm, in contrast to [PS00]. We feel this meshes better with standard assumptions. (In using the Lemma of [PS00], you need to assume, say, hardness of discrete-logarithm computations against expected-time adversaries. This assumption may be true, but is not the standard one.) Our Forking Lemma can be viewed as an extension of the Reset Lemma of [BP02], and our proof, which uses the techniques of the latter, is simpler than that of [PS00].

Relation to aggregate signatures. A natural thought is that multi-signatures are a special case of aggregate signatures, and we know the latter have been implemented without the KOSK assumption [BGLS03], so doesn't this yield multi-signatures in the plain public-key model? Let us explore this.

Suppose signer i has produced a BSL signature [BSL01] σ_i on a message m_i ($i = 1, \ldots, n$). The procedure of [BGLS03] aggregates $\sigma_1, \ldots, \sigma_n$ into a single, aggregate signature σ. Multi-signatures is simply the special case where $m_i = m$ is a common message for all $i \in L$. Now, [BGLS03] do prove security without the KOSK assumption, but for this *need to assume that the messages* $m_1, \ldots, m_n$ *are distinct.* So the multi-signature case is exactly the one where they do *not* have security without the KOSK assumption. In fact, in this case, there actually is a rogue-key attack on the scheme. Indeed, this MS scheme is exactly MS-Bo, which we know is not secure against rogue-key attack without the KOSK assumption.

However, [BGLS03] also suggest a workaround to the message distinctness assumption. Have each signer prepend its public key to its message before signing, so that the individual signatures now are (in the MS case) on the *enhanced messages* $pk_1 \,\|\, m, \ldots, pk_l \,\|\, m$. Now, security is guaranteed as long as the enhanced messages are distinct. However, this is not enough for security against rogue key attack in our plain public-key model. If an attacker, playing the role of signer 2, sets its public key pk_2 to equal the public key pk_1 of honest user 1 (an easy task) and outputs some forgery on some message m for some group including signers $1, 2$, then we have a situation where two enhanced messages are the same, so the result of [BGLS03] does not apply. To fix this one can use the analysis of [**?**] that shows the scheme is secure even if enhanced messages are not distinct, and then we do obtain a secure MS scheme. However, verification of a multi-signature for n signers costs $n + 1$ pairings, making it substantially less efficient than all the other schemes we have discussed, where verification time does not depend on the number of signers in the group.

Another potential route to multi-signatures is via sequential aggregate signatures [LMRS04]. These can be built from trapdoor permutation families in which there is a single domain underlying the entire family [LMRS04], but in fact there seems to be no example of such a family. RSA does not have the desired property. The authors build some RSA-based schemes directly, another one can be constructed using techniques from [HOT04]. Some limitations of the schemes of [LMRS04] are lifted in [**?**]. However, a sequential-aggregate based MS scheme will require a number of communication rounds proportional to the number n of signers involved in a signature, as well as n applications of the trapdoor function to verify, while all previous protocols, including ours, are constant-round and have constant verification cost.

With the increased adoption of small, energy-restricted devices such as laptops, cell phones, PDAs and sensors, battery life has become a crucial bottleneck in the usage of these devices — and an important distinguishing factor in their sales. Fast progress is being made in the development of lighter and higher-capacity batteries, but at the same time the demand for energy-preserving technology is more pressing than ever. Much effort is being put in the design of low-power microprocessors, but also the software running on these processors is being optimized for energy consumption, rather than for speed or portability.

In accordance with their wireless nature, communication on these portable devices often takes place over wireless channels such as Bluetooth and WiFi. Unfortunately, these communication mechanisms are rather expensive in terms of energy consumption. Reducing the number of bits to communicate is crucial to increase battery life: communicating a single bit of data requires significantly more power than executing a 32-bit instruction [BA03], so it makes perfect sense to invest extra computation cycles to save on bandwidth. Also, communication is often not reliable, so the fewer the number of bits one has to communicate, the better. To make things worse, wireless channels are inherently vulnerable to eavesdropping and tampering attacks by outsiders. Strong cryptography is needed to protect the communication, adding even more overhead to the communication. It is our challenge as designers of cryptographic primitives to limit this overhead to a minimum.

Multi-signature schemes. A multi-signature (MS) scheme [IN83] allows n different signers with public keys $pk_1, \ldots, pk_n$ to collectively sign a message m, yielding a multi-signature σ of roughly the same size as a standard signature, yet that certifies m under all public keys $pk_1, \ldots, pk_n$ simultaneously. By transmitting σ instead of n individual signatures, multi-signature schemes can help greatly to save on communication costs.

However, one still needs the public keys of all cosigners in order to verify the validity of such a multi-signature. In most applications these public keys will have to be transmitted along with the multi-signature, which partially defeats the primary purpose of using a multi-signature scheme, namely to save on bandwidth. The inclusion of some information that uniquely identifies the cosigners seems inevitable for verification, but often this information can be represented more succinctly than by means of randomly generated public keys. For example, the signers' user names or IP addresses could suffice for this purpose; this information may even already be present in package headers. Moreover, each public key may come with an associated certificate containing a signature from a certification authority (CA) and the CA's public key, which on its turn may come with a chain of certificates leading to the root CA. Altogether, this sums up to many more bits being transmitted than strictly necessary to authenticate the message.

Identity-based signatures. In an identity-based signature scheme [Sha85], the public key of a user is simply his identity, e.g. his name, email or IP address. A trusted key distribution center provides each signer with the secret signing key corresponding to his identity. When all signers have their secret keys issued by the same key distribution center, individual public keys become obsolete, removing the need for explicit certification and all associated costs. These features make the identity-based paradigm particularly appealing for use in conjunction with multi-signatures, leading to the concept of identity-based multi-signature (IBMS) schemes.

Generic constructions. In spite of their appeal with regard to applications, implementations of IBMS schemes are rather limited. As demonstrated in [DKXY03, BNN04], any standard signature scheme can be transformed into an identity-based one using the "certification paradigm". One can attempt to derive IBMS schemes from existing standard MS schemes via this approach [GHK06]. The problem is that the resulting multi-signature is not compact due to the need to include the certificates with each signature. Even if the signatures in the certificates can be aggregated [BGLS03], the public keys they contain cannot. In summary, unlike the case of standard signatures, there seems no trivial, general way to transform compact signature schemes into identity-based ones.

An existing construction. The only provably secure IBMS scheme known today is due to Gentry and Ramzan [GR06]. The scheme employs groups with bilinear maps (also known as pairings), which are usually implemented by modified Weil or Tate pairing over elliptic or hyperelliptic curves. To avoid putting all our eggs in the same basket, it is common practice in cryptography to try to find alternative constructions of a primitive based on different assumptions. While pairings have turned out extremely useful in the design of cryptographic protocols, they were only recently brought to the attention of cryptographers [Jou00], and hence did not yet enjoy the same exposure to cryptanalytic attacks by experts as other, older problems from number theory such as discrete logarithms, factoring and RSA. This exposure is necessary to build confidence in the hardness of the underlying problems; without it, their use in high-security applications may not be advisable.

Also, efficient implementations of RSA are ubiquitous, even in the public domain, while implementations of pairings are much harder to come by. Unlike RSA, even building an inefficient prototype implementation of pairings is far from straightforward for anyone but an expert, and even then it is often difficult or impossible to generate curves with the desired security parameters [GPS06]. Companies may have invested in expensive hardware or software implementations of RSA, and may be reluctant to reinvest in new pairing implementations.

Our contributions. We present an efficient and provably secure IBMS scheme based on RSA, which is thereby the first provably secure IBMS scheme not relying on the use of pairings. Our scheme is essentially a multi-signature variant of the Guilliou-Quisquater (GQ) identity-based signature scheme [GQ90], strengthened with techniques from [BN06] to provide security against concurrent attacks. Unstrengthened variants of our scheme were proposed before (but without security proofs) in [GQ90, CJKT06]. The proof makes use of the general forking lemma of [BN06], which, unlike the original forking lemma by Pointcheval and Stern [PS00], applies to more general contexts than generic standard signature schemes. Signatures under our scheme are 1184 bits long for typical values of the security parameters, which is longer than the 320-bit signatures of the scheme of [GR06]. Verification on the other hand is considerably cheaper: our scheme needs only a single (multi-) exponentiation in an RSA group, as opposed to three pairing computations for the scheme of [GR06]. The cost of one pairing computation is roughly that of 6–20 exponentiations.

We prove our scheme secure in the random oracle model under the one-wayness of RSA. Unlike the scheme of [GR06], our scheme requires the signers to interact to generate a signature, so we had to extend their security notion to model this interaction in the presence of an adversary, taking inspiration from the (non-identity-based) notions of [MOR01, BN06]. We consider the strongest possible setting, namely with insecure and unauthenticated communication links controlled by the adversary, without assuming the availability of a trusted broadcast primitive. In fact, we distinguish between two different notions, called *single-signer* and *multi-signer* security, based on the number of signers whose role can be played by the signing oracle. While not obvious at first, we prove that these notions are in fact equivalent, so that we can prove our scheme secure under the simpler single-signer notion. As in [BN06], but unlike [MOR01], we allow the adversary to concurrently engage in as many arbitrarily interleaved signature protocols as it wants.

Interactive vs. non-interactive schemes. As noted above, our scheme requires the signers to interact in order to generate a signature. The IBMS scheme of [GR06] is non-interactive, meaning that each signer independently computes its share to the signature, and anyone can combine these shares into a compact signature. The requirement of interaction may seem to conflict with our goal of saving on bandwidth. We argue that this is not always the case. Consider for example a wired network of sensors in a very remote location (e.g. in a desert, or in space) that needs to report back to a far-away base station through wireless communication. The sensors can use the cheap wired network to execute joint signing protocols and send the resulting compact signatures over the expensive wireless channel. A non-interactive scheme does not offer any real advantage here. In particular, it does not remove the need for local communication: the sensors still need a round of interaction to exchange signature shares. In general, the added cost of interaction depends highly on the network topology.

Aggregate vs. multi-signatures. Aggregate signature (AS) schemes [BGLS03] can be seen as a generalization of multi-signatures where each signer i signs a different message m_i. Only a single identity-based aggregate signature (IBAS) scheme is known [GR06]; it is also based on pairings. IBAS schemes automatically give rise to IBMS schemes, but the scheme resulting from the only known IBAS instantiation [GR06] is less efficient than their direct IBMS construction. We note that the distinction between aggregate and multi-signatures becomes irrelevant for interactive schemes. Indeed, one can easily construct an interactive aggregate signature scheme from a multi-signature scheme by letting the signers, in a first round of communication, inform each other about the messages m_i they are about to sign. The common message m can then be taken to be the concatenation of (ID_i, m_i) tuples. Hence, the single-message restriction of multi-signature schemes is not really limiting in the case of interactive schemes.

Other related work. Cheng et al. [CLW05] recently proposed another interactive IBMS scheme based on pairings, but proved it secure only under a weak variant of selective-ID security. To the best of our knowledge, the schemes of [CLW05, GR06] are the only instantiations of IBMS in the literature.

There is more work on compact signature schemes in the non-identity-based setting. There is a vast literature on MS schemes, but the only provably secure schemes are those of [MOR01, Bol03, LOS$^+$06, BN06]. The schemes of [Bol03, LOS$^+$06] are based on pairings, those of [MOR01, BN06] on discrete logarithms. In a sequential aggregate signature (SAS) scheme [LMRS04], aggregation cannot be performed by an outsider; instead, the signers cooperate, each in turn aggregating their signature into the current aggregate using their secret key. The only known instantiations of SAS schemes are due to [LMRS04, LOS$^+$06]. The scheme of [LMRS04] is based on families of certified trapdoor permutations, of which strictly speaking no instantiations exist, but the authors discuss how to instantiate their scheme with RSA. The scheme of [LOS$^+$06] uses pairings, and is the only one with security in the standard (i.e., non-random-oracle [BR93]) model.

2 Notation and Standard Definitions

Notation. Let $\mathbb{N} = \{1, 2, 3, \ldots\}$. A string means a binary one. The empty string is denoted ε. If x, y are strings, then $|x|$ is the length of x. If $x_1, x_2, \ldots$ are objects then $x_1 \| x_2 \| \ldots$ denotes an encoding of them as strings from which the constituent objects are easily recoverable. If S is a (multi)set, then $|S|$ is its cardinality, $s \overset{\$}{\leftarrow} S$ denotes the operation of assigning to s an element of S chosen at random, and $\langle S \rangle$ is a unique encoding of S as a string. If A is a randomized algorithm, then $\mathsf{A}(x_1, \ldots; \rho)$ denotes its output on inputs $x_1, \ldots$ and coins ρ, while $y \overset{\$}{\leftarrow} \mathsf{A}(x_1, \ldots)$ means that we choose ρ at random and let $y = \mathsf{A}(x_1, \ldots; \rho)$.

Standard digital signatures. We recall the standard definition of unforgeability under chosen-message attack following [GMR88]. A digital signature scheme $\mathsf{DS} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ is a triple of algorithms. The signer generates a key pair $(pk, sk) \overset{\$}{\leftarrow} \mathsf{Kg}$ and signs a message m via $\sigma \leftarrow \mathsf{Sign}(sk, m)$. Verification is done by checking that $\mathsf{Vf}(pk, m, \sigma)$ returns 1, indicating acceptance. The advantage of a forging algorithm F in breaking the existential unforgeability under chosen-message attack (uf-cma) of DS is defined as

$$\mathbf{Adv}^{\text{uf-cma}}_{\mathsf{DS}}(\mathsf{F}) = \Pr\left[\begin{array}{c} \mathsf{Vf}(pk, m, \sigma) = 1 \text{ and} \\ \mathsf{F} \text{ did not query } \mathsf{Sign}(sk, m) \end{array} : (pk, sk) \overset{\$}{\leftarrow} \mathsf{Kg}\ ;\ (m, \sigma) \overset{\$}{\leftarrow} \mathsf{F}^{\mathsf{Sign}(sk, \cdot)}(pk) \right]$$

where $\mathsf{Sign}(sk, \cdot)$ is a signing oracle under sk. In the random oracle model [BR93], the Sign and Vf algorithms, as well as the forger, additionally have access to a random oracle H. We say that forger F $(t, q_{\mathrm{S}}, q_{\mathrm{H}}, \epsilon)$-breaks DS if it runs in time at most t, makes at most q_{S} queries to the signing oracle and q_{H} queries to the random oracle, and has advantage $\mathbf{Adv}^{\text{uf-cma}}_{\mathsf{DS}}(\mathsf{F}) \geq \epsilon$. The scheme DS is said to be $(t, q_{\mathrm{S}}, q_{\mathrm{H}}, \epsilon)$-secure if there exists no forger F who $(t, q_{\mathrm{S}}, q_{\mathrm{H}}, \epsilon)$-breaks it.

3 A Simple and General Forking Lemma

We state our Forking Lemma here. We will use this lemma to prove the security of our multi-signature scheme in the next section, and then come back to prove the Forking Lemma itself in Section 3. Our Forking Lemma, unlike that of Pointcheval and Stern [PS00], makes no mention of signature schemes or random oracles, but rather concentrates on the output behavior of an algorithm when run twice on related inputs. This makes it easily applicable in contexts other than standard signature schemes, and separates the probabilistic analysis of the rewinding from the actual simulation in the security proof, allowing for more modular (and hence easier to verify) proofs. In the following, think of x as a public key and $h_1, \ldots, h_q$ as replies to queries to a random oracle.

Lemma 3.1 [General Forking Lemma]Fix an integer $q \geq 1$ and a set H of size $h \geq 2$. Let A be a randomized algorithm that on input $x, h_1, \ldots, h_q$ returns a pair, the first element of which is an integer in the range $0, \ldots, q$ and the second element of which we refer to as a *side output*. Let IG be a randomized algorithm that we call the input generator. The *accepting probability* of A, denoted acc, is defined as the probability that $J \geq 1$ in the experiment

$$x \overset{\$}{\leftarrow} \mathsf{IG}\ ;\ h_1, \ldots, h_q \overset{\$}{\leftarrow} H\ ;\ (J, \sigma) \overset{\$}{\leftarrow} \mathsf{A}(x, h_1, \ldots, h_q)\ .$$

The *forking algorithm* $\mathsf{F_A}$ associated to A is the randomized algorithm that takes input x proceeds as follows:

```
Algorithm $\mathsf{F_A}(x)$
    Pick coins $\rho$ for $\mathsf{A}$ at random
    $h_1, \ldots, h_q \overset{\$}{\leftarrow} H$
    $(I, \sigma) \leftarrow \mathsf{A}(x, h_1, \ldots, h_q; \rho)$
    If $I = 0$ then return $(0, \varepsilon, \varepsilon)$
    $h'_I, \ldots, h'_q \overset{\$}{\leftarrow} H$
    $(I', \sigma') \leftarrow \mathsf{A}(x, h_1, \ldots, h_{I-1}, h'_I, \ldots, h'_q; \rho)$
    If $(I = I'$ and $h_I \neq h'_I)$ then return $(1, \sigma, \sigma')$
    Else return $(0, \varepsilon, \varepsilon)$.
```

Let

$$\text{frk} \;=\; \Pr\left[\, b=1 \;:\; x \xleftarrow{\$} \mathsf{IG}\,;\; (b,\sigma,\sigma') \xleftarrow{\$} \mathsf{F}_{\mathsf{A}}(x) \,\right] .$$

Then

$$\text{frk} \;\geq\; \text{acc} \cdot \left(\frac{\text{acc}}{q} - \frac{1}{h} \right) . \tag{1}$$

Alternatively,

$$\text{acc} \;\leq\; \frac{q}{h} + \sqrt{q \cdot \text{frk}}\,. \quad \blacksquare \tag{2}$$

When using this lemma in a security proof, we will typically take an adversary attacking our scheme and build from it an algorithm A that fits the assumptions of the lemma above. Intuitively, $h_1, \ldots, h_q$ will be the replies to random oracle queries made by the original adversary, who will now be executed by A. The forking adversary implements the rewinding. It is important above that the two executions of A performed by F_{A} use the same coins ρ. We have left to prove our Forking Lemma. Before doing so, we recall two sublemmas that we will use in the proof. The first is a standard fact. One can derive it from Jensen's inequality, or as a consequence of the fact that the variance of any random variable is non-negative. For the sake of self-containment we provide a direct proof based on the last approach.

Lemma 3.2 Let X be a real-valued random variable. Then $\mathbf{E}\left[X^2\right] \geq \mathbf{E}\left[X\right]^2$.

Proof of Lemma 3.2: Let $\mu = \mathbf{E}\left[X\right]$. The random variable $(X-\mu)^2$ is non-negative. Thus

$$\begin{aligned} 0 \;&\leq\; \mathbf{E}\left[(X-\mu)^2\right] \;=\; \mathbf{E}\left[X^2 - 2\mu X + \mu^2\right] \;=\; \mathbf{E}\left[X^2\right] - 2\mu \mathbf{E}\left[X\right] + \mu^2 \\ &=\; \mathbf{E}\left[X^2\right] - 2\mu^2 + \mu^2 \;=\; \mathbf{E}\left[X^2\right] - \mu^2 , \end{aligned}$$

and thus $\mathbf{E}\left[X^2\right] \geq \mu^2$. $\blacksquare$

We will use the above directly, and also use the following corollary. (It is used also in [AR00] where it is proved differently.)

Lemma 3.3 Suppose $q \geq 1$ is an integer, and $x_1, \ldots, x_q \geq 0$ are real numbers. Then

$$\sum_{i=1}^{q} x_i^2 \;\geq\; \frac{1}{q} \cdot \left(\sum_{i=1}^{q} x_i \right)^2 .$$

Proof: Let X be the random variable that takes value x_i with probability $1/q$ for each $1 \leq i \leq q$. Then

$$\mathbf{E}\left[X^2\right] \;=\; \frac{1}{q} \cdot \sum_{i=1}^{q} x_i^2 \qquad \text{and} \qquad \mathbf{E}\left[X\right]^2 \;=\; \frac{1}{q^2} \cdot \left(\sum_{i=1}^{q} x_i \right)^2 .$$

Now apply Lemma 3.2. $\blacksquare$

We are now ready to prove our Forking Lemma.

Proof of Lemma 3.1: We first establish Equation (1) and then show that it implies Equation (2).For any input x let

$$\begin{aligned} \text{acc}(x) \;&=\; \Pr\left[\, J \geq 1 \;:\; h_1, \ldots, h_q \xleftarrow{\$} H\,;\; (J,\sigma) \xleftarrow{\$} \mathsf{A}(x, h_1, \ldots, h_q) \,\right] \\ \text{frk}(x) \;&=\; \Pr\left[\, b=1 \;:\; (b,\sigma,\sigma') \xleftarrow{\$} \mathsf{F}_{\mathsf{A}}(x) \,\right] . \end{aligned}$$

We claim that for all x,

$$\mathrm{frk}(x) \;\geq\; \mathrm{acc}(x) \cdot \left(\frac{\mathrm{acc}(x)}{q} - \frac{1}{h} \right) . \tag{3}$$

Then, with the expectation taken over $x \xleftarrow{\$} \mathsf{IG}$, we have

$$\begin{aligned} \mathrm{frk} &= \mathbf{E}\left[\mathrm{frk}(x)\right] \;\geq\; \mathbf{E}\left[\mathrm{acc}(x) \cdot \left(\frac{\mathrm{acc}(x)}{q} - \frac{1}{h} \right) \right] \\ &= \frac{\mathbf{E}\left[\mathrm{acc}(x)^2\right]}{q} - \frac{\mathbf{E}\left[\mathrm{acc}(x)\right]}{h} \;\geq\; \frac{\mathbf{E}\left[\mathrm{acc}(x)\right]^2}{q} - \frac{\mathbf{E}\left[\mathrm{acc}(x)\right]}{h} \;=\; \mathrm{acc} \cdot \left(\frac{\mathrm{acc}}{q} - \frac{1}{h} \right) . \end{aligned}$$

This establishes Equation (1). Above, we used Equation (3), Lemma 3.2 and also the fact that $\mathbf{E}\left[\mathrm{acc}(x)\right] = \mathrm{acc}$.

We proceed to the proof of Equation (3). For any input x, with probabilities taken over the coin tosses of $\mathsf{F_A}$ we have

$$\begin{aligned} \mathrm{frk}(x) &= \Pr\left[\, I = I' \wedge I \geq 1 \wedge h_I \neq h'_I \,\right] \;\geq\; \Pr\left[\, I = I' \wedge I \geq 1 \,\right] - \Pr\left[\, I \geq 1 \wedge h_I = h'_I \,\right] \\ &= \Pr\left[\, I = I' \wedge I \geq 1 \,\right] - \frac{\Pr\left[\, I \geq 1 \,\right]}{h} \;=\; \Pr\left[\, I = I' \wedge I \geq 1 \,\right] - \frac{\mathrm{acc}(x)}{h} . \end{aligned}$$

It remains to show that $\Pr\left[\, I = I' \wedge I \geq 1 \,\right] \geq \mathrm{acc}(x)^2/q$. Let $\mathcal{R}$ denote the set from which A draws its coins at random. For each $i \in \{1, \ldots, q\}$ let $X_i \colon \mathcal{R} \times H^{i-1} \to [0,1]$ be defined via

$$X_i(\rho, h_1, \ldots, h_{i-1}) \;=\; \Pr\left[\; J = i \;:\; h_i, \ldots, h_q \xleftarrow{\$} H \;;\; (J, \sigma) \leftarrow \mathsf{A}(x, h_1, \ldots, h_q; \rho) \;\right]$$

for all $\rho \in \mathcal{R}$ and $h_1, \ldots, h_{i-1} \in H$. Regard X_i as a random variable over the uniform distribution on its domain. Then

$$\begin{aligned} \Pr\left[\, I = I' \wedge I \geq 1 \,\right] &= \sum_{i=1}^{q} \Pr\left[\, I = i \wedge I' = i \,\right] \;=\; \sum_{i=1}^{q} \Pr\left[\, I = i \,\right] \cdot \Pr\left[\, I' = i \;:\; I = i \,\right] \\ &= \sum_{i=1}^{q} \sum_{\rho, h_1, \ldots, h_{i-1}} X_i(\rho, h_1, \ldots, h_{i-1}) \cdot X_i(\rho, h_1, \ldots, h_{i-1}) \cdot \frac{1}{|\mathcal{R}| \cdot |H|^{i-1}} \\ &= \sum_{i=1}^{q} \mathbf{E}\left[X_i^2\right] \;\geq\; \sum_{i=1}^{q} \mathbf{E}\left[X_i\right]^2 , \end{aligned}$$

where in the last step we used Lemma 3.2. Now let $x_i = \mathbf{E}\left[X_i\right]$ for $i \in \{1, \ldots, q\}$, and apply Lemma 3.3. We get

$$\sum_{i=1}^{q} \mathbf{E}\left[X_i\right]^2 \;\geq\; \frac{1}{q} \cdot \left(\sum_{i=1}^{q} \mathbf{E}\left[X_i\right] \right)^2 \;=\; \frac{1}{q} \cdot \mathrm{acc}(x)^2 .$$

This completes the proof of Equation (3) and thus of Equation (1). We now show how to obtain Equation (2). Using Equation (1) we have

$$\left(\mathrm{acc} - \frac{q}{2h} \right)^2 \;=\; \mathrm{acc}^2 - \frac{q}{h} \cdot \mathrm{acc} + \frac{q^2}{4h^2} \;\leq\; q \cdot \mathrm{frk} + \frac{q^2}{4h^2} .$$

```
Algorithm Kg
    $g \xleftarrow{\$} \mathbb{G}^* \,;\, x \xleftarrow{\$} \mathbb{Z}_p \,;\, X \leftarrow g^x$
    $pk \leftarrow (\mathbb{G}, p, g, X)$
    $sk \leftarrow (\mathbb{G}, p, g, x)$
    Return $(pk, sk)$

Algorithm $\mathsf{Sign}^{\mathrm{H}}(sk, m)$
    Parse $sk$ as $(\mathbb{G}, p, g, x)$
    $r \xleftarrow{\$} \mathbb{Z}_p \,;\, R \leftarrow g^r$
    $s \leftarrow r + x \cdot \mathrm{H}(R \,\|\, m) \bmod p$
    Return $(R, s)$

Algorithm $\mathsf{Vf}^{\mathrm{H}}(pk, m, \sigma)$
    Parse $pk$ as $(\mathbb{G}, p, g, X)$
    Parse $\sigma$ as $(R, s)$
    If $R \notin \mathbb{G}$ then return 0
    If $g^s = RX^{\mathrm{H}(R \,\|\, m)}$
    then return 1 else return 0
```

Figure 1: The Schnorr signature scheme $\mathsf{Sch} = (\mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$, associated to group $\mathbb{G}$ of prime order p and to integer $\ell \geq 1$. Here H: $\{0,1\}^* \to \{0,1\}^\ell$ is a random oracle.

Taking the square root of both sides, and using the fact that $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$ for any real numbers $a, b \geq 0$, we get

$$\mathrm{acc} - \frac{q}{2h} \quad \leq \quad \sqrt{q \cdot \mathrm{frk}} + \sqrt{\frac{q^2}{4h^2}} \quad = \quad \frac{q}{2h} + \sqrt{q \cdot \mathrm{frk}} \,.$$

Re-arranging terms yields Equation (2). ▌

4 Example Application to Schnorr Signatures

Let us do an example application to the Schnorr signature scheme [Sch90]. We begin by recalling some necessary definitions. The scheme itself is depicted in Figure 1.

THE DISCRETE LOGARITHM ASSUMPTION. Let $\mathbb{G}$ be a multiplicative group of prime order p, and let $\mathbb{G}^* = \mathbb{G} \setminus \{1\}$. The advantage of algorithm A in solving the discrete logarithm problem in $\mathbb{G}$ is defined as

$$\mathbf{Adv}^{\mathrm{dlog}}_{\mathbb{G}}(\mathsf{A}) = \Pr\left[\, g^x = y \;:\; g \xleftarrow{\$} \mathbb{G}^* \,;\, y \xleftarrow{\$} \mathbb{G} \,;\, x \xleftarrow{\$} \mathsf{A}(y) \,\right] \,.$$

We say that A (t, ϵ)-solves the discrete logarithm problem in $\mathbb{G}$ if it runs in time at most t and has advantage $\mathbf{Adv}^{\mathrm{dlog}}_{\mathbb{G}}(\mathsf{A}) \geq \epsilon$, and we say that the discrete logarithm problem in $\mathbb{G}$ is (t, ϵ)-hard if no algorithm A (t, ϵ)-solves it.

APPLICATION TO SCHNORR SIGNATURES. The following theorem states the security of the Sch signature scheme. The result is known [PS00], but we present a simpler proof using our generalized Forking Lemma.

Theorem 4.1 If the discrete logarithm problem in $\mathbb{G}$ is (t', ϵ')-hard, then the Sch signature scheme associated to $\mathbb{G}$ and $\ell \in \mathbb{N}$ is $(t, q_{\mathrm{S}}, q_{\mathrm{H}}, \epsilon)$-secure whenever

$$\epsilon' \;\leq\; \frac{\epsilon^2}{q_{\mathrm{H}} + q_{\mathrm{S}} + 1} - \frac{2q_{\mathrm{S}}}{2^k} - \frac{1}{2^\ell} \quad \text{and} \quad t' \;\geq\; 2t + 2q_{\mathrm{S}} t_{\mathrm{exp}} + O(q_{\mathrm{H}} + q_{\mathrm{S}} + 1) \,.$$

Here, $k = \lfloor \log_2 p \rfloor$ and t_{exp} is the time needed to compute an exponentiation in $\mathbb{G}$.

Proof of Theorem 4.1: We prove the theorem by contradiction: given a forging algorithm B that $(t, q_{\mathrm{S}}, q_{\mathrm{H}}, \epsilon)$-breaks the Sch signature scheme, we build an algorithm D that (t', ϵ')-solves the discrete logarithm problem in $\mathbb{G}$ with

$$\epsilon' \;\geq\; \frac{\epsilon^2}{q_{\mathrm{H}} + q_{\mathrm{S}} + 1} - \frac{2q_{\mathrm{S}}}{2^k} - \frac{1}{2^\ell} \quad \text{and} \quad t' = 2t + 2q_{\mathrm{S}} t_{\mathrm{exp}} + O(q_{\mathrm{H}} + q_{\mathrm{S}} + 1) \,.$$

First consider algorithm A that takes input a public key $(\mathbb{G}, p, g, X)$ and $h_1, \ldots, h_{q_{\mathrm{H}}+q_{\mathrm{S}}+1} \in H$:

```
Algorithm $\mathsf{A}((\mathbb{G}, p, g, X), h_1, \ldots, h_{q_\mathrm{H}+q_\mathrm{S}+1})$
    $ctr \leftarrow 0$ ; $S \leftarrow \emptyset$ ; $bad \leftarrow \mathtt{false}$
    Run $\mathsf{B}$ on input $(\mathbb{G}, p, g, X)$, answering its oracle queries as follows:
        On query $R \,\|\, m$ to the hash oracle:
            If $T[R \,\|\, m] = \perp$ then $ctr \leftarrow ctr + 1$ ; $R_{ctr} \,\|\, m_{ctr} \leftarrow R \,\|\, m$ ; $T[R \,\|\, m] \leftarrow h_{ctr}$
            Return $T[R \,\|\, m]$ to $\mathsf{B}$
        On query $m$ to the sign oracle:
            $ctr \leftarrow ctr + 1$ ; $s \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ ; $R \leftarrow g^s X^{-h_{ctr}}$ ; $S \leftarrow S \cup \{m\}$
            If $T[R \,\|\, m] \neq \perp$ then $bad \leftarrow \mathtt{true}$
            $T[R \,\|\, m] \leftarrow h_{ctr}$ ; Return $(R, s)$
    Until $\mathsf{B}$ outputs $(m, (R, s))$
    If $T[R \,\|\, m] = \perp$ then $ctr \leftarrow ctr + 1$ ; $R_{ctr} \,\|\, m_{ctr} \leftarrow R \,\|\, m$ ; $T[R \,\|\, m] \leftarrow h_{ctr}$
    If ($\mathsf{Vf}(pk, m, (R, s)) = 0$ or $m \in S$ or $bad = \mathtt{true}$) then return $(0, \varepsilon)$
    Let $i$ be such that $R_i \,\|\, m_i = R \,\|\, m$
    Return $(i, (s, h_i))$
```

Let IG be the algorithm that runs Kg to get (pk, sk) and outputs pk. Let acc be defined as in Lemma 3.1. Note that

$$\mathrm{acc} \;\geq\; \epsilon - \Pr\left[\, bad = \mathtt{true} \,\right] \;\geq\; \epsilon - \frac{q_\mathrm{S}(q_\mathrm{H} + q_\mathrm{S} + 1)}{2^k} \,.$$

Let F_A denote the forking algorithm associated to A as per Lemma 3.1. (It takes input $x = pk$ output by IG.) Then we define our discrete logarithm finding algorithm as follows:

```
Algorithm $\mathsf{D}(\mathbb{G}, p, g, X)$
    $pk \leftarrow (\mathbb{G}, p, g, X)$ ; $(b, \sigma, \sigma') \stackrel{\$}{\leftarrow} \mathsf{F}_\mathsf{A}(pk)$
    If $b = 0$ then return 0 and halt
    Parse $\sigma$ as $(s, h)$ and parse $\sigma'$ as $(s', h')$
    // We know that there exists $R \in \mathbb{G}$ such that $g^s = RX^h$ and $g^{s'} = RX^{h'}$, and $h \neq h'$
    Return $(s - s')(h - h')^{-1} \bmod p$
```

We will justify the comment in the code above later. Note that assuming it is true, the output of D is the discrete logarithm (to base g) of X. Let frk be defined as in Lemma 3.1 based on F_A. Then, applying the lemma, we have that the advantage ϵ' of D in solving the discrete logarithm problem in $\mathbb{G}$ is

$$\begin{aligned}
\epsilon' &\geq \mathrm{frk} \\
&\geq \mathrm{acc} \cdot \left(\frac{\mathrm{acc}}{q_\mathrm{H} + q_\mathrm{S} + 1} - \frac{1}{2^\ell} \right) \\
&\geq \frac{\epsilon^2}{q_\mathrm{H} + q_\mathrm{S} + 1} - \frac{2q_\mathrm{S}}{2^k} - \frac{1}{2^\ell}
\end{aligned}$$

It remains to justify the comment in the code of D. The definition of F_A, in combination with that of A, tells us that if $b = 1$ then there are coins ρ for A, $i \geq 1$, and $h_1, \ldots, h_{q_\mathrm{H}+q_\mathrm{S}+1}, h'_i, \ldots, h'_{q_\mathrm{H}+q_\mathrm{S}+1} \in H$ with $h = h_i \neq h'_i = h'$ such that, in the execution of $\mathsf{A}((\mathbb{G}, p, g, X), h_1, \ldots, h_{q_\mathrm{H}+q_\mathrm{S}+1}; \rho)$, B outputs a valid forgery $(m, (R, s))$ with $R \,\|\, m = R_i \,\|\, m_i$ and $T[R \,\|\, m] = h_i$, and also, in the execution of $\mathsf{A}((\mathbb{G}, p, g, X), h_1, \ldots, h_{i-1}, h'_i, \ldots, h'_{q_\mathrm{H}+q_\mathrm{S}+1}; \rho)$, B outputs a valid forgery $(m', (R', s'))$ where $R' \,\|\, m' = R'_i \,\|\, m'_i$ and $T[R \,\|\, m] = h'_i$. Since the two executions of A are identical until the i-th random oracle query, the arguments of the i-th random oracle query in both executions $R_i \| m_i$ and $R'_i \| m'_i$ are identical as well. It follows that $R = R'$ and $m' = m$ and $g^s = RX^h$ and $g^{s'} = RX^{h'}$.

For the running time, we assume that an exponentiation in $\mathbb{G}$ takes time $t_{\exp}$ and all other operations take unit time. The running time t' of D is twice that of A plus $O(q_{\mathrm{H}} + q_{\mathrm{S}} + 1)$. The running time of A is $t + q_{\mathrm{S}} t_{\exp} + O(q_{\mathrm{H}} + q_{\mathrm{S}} + 1)$. ▌

5 Multi-Signatures

Here we provide our definitions for multi-signatures with security in the plain public-key model.

The model. Consider a group of signers signing the same message m, each having as input its own public and secret key as well as a list of the public keys of the other signers. The signers want to interact in a protocol which eventually outputs a compact signature σ that represents the signature of each individual signer on the message m. We assume the signers are connected to each other via point-to-point links over which they can send messages. We do not assume these links are secure (that is, they are neither private nor authenticated) and we do not assume a broadcast primitive. The signers interact for some number of rounds. In each round, view a signer as receiving a (but not necessarily the same) message from every other signer, performing some computation, and then sending a message to every other signer, except that in the first round the "received message" is the party's input (and so is not really received) and in the last round the "sent message" is a local output (and so is not really sent). The local output is either $\perp$ to indicate failure or is the compact signature σ. Instances of the protocol may be executed concurrently, with one signer possibly participating in several concurrent instances at the same time.

In describing protocols, we will have each signer assign indices $1, \ldots, n$ to the signers, with itself being signer 1. We clarify that these are local references to the cosigners participating in this protocol instance. (That is, each signer in this protocol instance chooses its own indexing, so that signer 3 on my list and your list may not be the same. Think of the index a signer gives to its cosigners as locally identifying the link over which they communicate.) These indices have no meaning outside this protocol instance and have no certified relationship with the public keys. In particular, they are not identities.

Formally a multi-signature scheme $\mathsf{MS} = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ consists of four algorithms. A central authority runs the parameter generation algorithm Pg to generate the system-wide parameters par. Each signer independently generates its own public and private key pair via $(pk, sk) \overset{\$}{\leftarrow} \mathsf{Kg}(par)$. We stress that this is a non-interactive process that can be performed by any signer at any given time. New signers can join the system at will, and need not engage in expensive protocols with a CA or with other signers to prove knowledge of the corresponding secret key before participating in signing protocols. The Sign algorithm represents the signing protocol as indicated above. The verification algorithm Vf takes as input a multiset of public keys $L = \{pk_1, \ldots, pk_n\}$, the message m and a candidate signature σ, and outputs 1 if σ is a valid signature for L and m, or outputs 0 otherwise. (Because users may let their keys depend on those of other users, we explicitly allow them to be the same by modeling L as a multiset.) We have the obvious correctness requirement, namely that if a group of signers begin their interaction with public keys $L = \{pk_1, \ldots, pk_n\}$ and message m, and all signers follow the protocol (meaning, perform their computations according to Sign) then all have local output σ such that Vf returns 1 on input L, σ.

In describing protocols we will not specify the signing algorithm directly but instead describe protocols informally by saying what parties receive, compute and send in each round.

Security. The notion of security requires that it be infeasible to forge multi-signatures involving at least one honest signer. As in previous works [MOR01, BGLS03, LOS+06] we can in fact assume there is a single honest signer. Our adversary will be viewed as having effectively corrupted all other signers. It can choose their public keys as it likes, even as a function of the public key of the honest signer, and can interact arbitrarily with the honest signer in any number of concurrent instances before outputting its forgery attempt. In somewhat more detail, we consider the following three-phase game associated to multi-signature scheme $\mathsf{MS} = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ and adversary (forger) F:

Setup: The game chooses the system-wide parameters $par \overset{\$}{\leftarrow} \mathsf{Pg}$ and a key-pair $(pk^*, sk^*) \overset{\$}{\leftarrow} \mathsf{Kg}(par)$ for the for the "target" (honest) signer. The target public key pk^* is given as input to the forger F.

Attack: F can start a protocol instance with the honest signer by providing the latter with a message m and a multiset $L = \{pk_1, \ldots, pk_n\}$ of purported cosigners, where pk^* occurs in L at least once. It can choose these public keys as it wishes, including as a function of pk^* and previous protocol flows. In interacting with the honest signer, F will play the role of all signers in L other than one instance of pk^*, sending messages to, and receiving messages from, the honest signer. The forger can schedule an arbitrary number of protocol instances concurrently, interacting with "clones" of the honest signer, where each clone maintains its own state and uses its own coins but all use the keys pk^*, sk^* and follow the protocol (meaning use algorithm Sign) to compute their responses to received messages. When the honest signer terminates then its local output (whether $\perp$ or a compact signature) is returned to F.

Forgery: At the end of its execution, F outputs a multiset $L = \{pk_1, \ldots, pk_n\}$, a message m and a forged signature σ. The forger is said to win the game if $\mathsf{Vf}(L, m, \sigma) = 1$, $pk^* \in L$ and F never initiated a signing protocol with L, m.

The advantage of F in breaking MS, denoted as $\mathbf{Adv}^{\text{uf-cma}}_{\mathsf{MS}}(\mathsf{F})$, is defined as the probability that F wins the above game, where the probability is taken over the coin tosses of the forger, the honest signer, and the setup phase. We say that a forger F $(t, q_{\mathrm{S}}, n_{\max}, \epsilon)$-breaks MS if F runs in time at most t, F initiates at most q_{S} signing protocols with the honest signer, the number of public keys in the multiset L involved in any signing query or in the forgery is at most $n_{\max}$, and $\mathbf{Adv}^{\text{uf-cma}}_{\mathsf{MS}}(\mathsf{F}) \geq \epsilon$. The scheme MS is said to be $(t, q_{\mathrm{S}}, n_{\max}, \epsilon)$-secure if no forger $(t, q_{\mathrm{S}}, n_{\max}, \epsilon)$-breaks it. In the random oracle model, the Sign and Vf algorithms, as well as the adversary, additionally have access to a random oracle $\mathrm{H} : \{0,1\}^* \to D$, where D is a set possibly depending on the system parameters. The additional parameter q_{H} denotes the maximum number of F's random oracle queries. (If there is more than one random oracle, we mean the total number of queries to all random oracles.) We say that F $(t, q_{\mathrm{S}}, q_{\mathrm{H}}, n_{\max}, \epsilon)$-breaks MS in the random oracle model, and that MS is $(t, q_{\mathrm{S}}, q_{\mathrm{H}}, n_{\max}, \epsilon)$-secure in the random oracle model.

Discussion. Note that the game described above does not require the adversary to fix the public keys of all signers in the system at the beginning of the game (as is required by the notions of [MOR01, Bol03]), or to submit the secret keys of all corrupt signers to a special certification oracle (as is required by the notions of [Bol03, LOS+06]). Rather, our model allows the adversary to dynamically add new signers to the system, using arbitrary public keys that may depend on the target public key or on previous signing interactions. It thereby avoids the KOSK assumption and does not presume expensive proof of knowledge protocols to be performed with the CA. This security notion reflects a real-world system with the desirable features that new signers can join on-the-fly using self-generated keys, and that existing (external) CA infrastructure can be reused for the certification of these keys.

6 A Multi-Signature Scheme based on Discrete Logarithms

Intuition. Our first construction is based on the Schnorr signature scheme. A first idea to aggregate signatures may be to let a signature under keys $L = \{X_1, \ldots, X_n\}$ of message m be a pair (R, s) such that $g^s = R \prod_{i=1}^{n} X_i^c$, where $c = \mathrm{H}(R\|\langle L\rangle\|m)$ is determined by a random oracle. Without restrictions on key generation however, this type of scheme is vulnerable to a well-known attack [HMP95, Lan96, MH96, MOR01] where a corrupt signer chooses $x_1 \xleftarrow{\$} \mathbb{Z}_p$ and sets its public key $X_1 \leftarrow g^{x_1} \cdot \prod_{i=2}^{n} X_i^{-1}$. This way, x_1 essentially becomes the "secret key" for the entire group of signers $L = \{X_1, \ldots, X_n\}$: signer 1 can by himself sign any message m in name of the entire group L by choosing $r \xleftarrow{\$} \mathbb{Z}_p$ and computing $(R = g^r\ ,\ s = cx_1 + r \bmod p)$ where $c = \mathrm{H}(R\|\langle L\rangle\|m)$.

We counteract this attack by using a different value c_i in the exponent of each public key X_i, so that the verification equation becomes $g^s = R \prod_{i=1}^{n} X_i^{c_i}$, where the values for c_i are determined by independent random oracle queries $c_i = \mathrm{H}(X_i\|R\|\langle L\rangle\|m)$. With the help of our generalized Forking Lemma (see Lemma 3.1), we succeed in extracting from any forger the discrete logarithm of the target public key. The way we apply the Forking Lemma is particularly interesting because we need certain random oracle responses to be the same in both executions of the adversary, even though the corresponding queries may not occur until *after* the fork.

A second problem is that, in order to respond to the forger's signature queries, the simulator needs to know the value of R before the forger does, so that it can program the random oracle. The value of R is typically computed as the product of individual shares of R chosen by each signer. Unless the target signer is the last to reveal his share (which we cannot assume to always be the case), the forger knows R before the simulator does, enabling the forger to perform a random oracle query involving R and thereby to prevent the simulator from programming this entry later on. We overcome this problem by letting signers first "commit" to their share of R through an additional random oracle query. The simulator, who sees all random oracle queries, can therefore look up the individual shares of R before the forger can, and can thus correctly program the random oracle.

THE SCHEME. Let $k = \lfloor \log_2 p \rfloor$, let $l_0, l_1 \in \mathbb{N}$, and let $\mathrm{H}_0 : \{0,1\}^* \to \{0,1\}^{l_0}$ and $\mathrm{H}_1 : \{0,1\}^* \to \{0,1\}^{l_1}$ be random oracles. To these, we associate the multi-signature scheme $\mathsf{MS\text{-}DL} = (\mathsf{Pg}, \mathsf{Kg}, \mathsf{Sign}, \mathsf{Vf})$ as follows:

Parameter generation. A trusted center chooses a random generator $g \xleftarrow{\$} \mathbb{G}^*$ and publishes $(\mathbb{G}, p, g)$ as system-wide parameters.

Key generation. Each signer runs the Kg algorithm to generate a random secret key $x \xleftarrow{\$} \mathbb{Z}_p$ and the corresponding public key $X \leftarrow g^x$.

Signing. Let X_1 and x_1 be the public and private key of a signer, let m be the message to be signed, and let $X_2, \ldots, X_n$ be the public keys of all other cosigners. We recall that the indices $1, \ldots, n$ are merely local references to cosigners, defined by one signer within one protocol instance. These indices are not tied to public keys in a global way, and in particular are not unique identities of signers. The communication proceeds in a number of rounds, where in each round each signer receives a message from every other signer, performs some local computation, and sends a message to every other signer.

Round 1:

- Local input: x_1, $L = \{X_1, \ldots, X_n\}$, m
- Computation: Choose $r_1 \xleftarrow{\$} \mathbb{Z}_p$, compute $R_1 \leftarrow g^{r_1}$ and query $t_1 \leftarrow \mathrm{H}_0(R_1)$.
- Send to signer i: t_1

Round 2:

- Receive from signer i: t_i
- Send to signer i: R_1

Round 3:

– Receive from signer i: R_i

– Computation: For all $2 \leq i \leq n$, check that $t_i = \mathrm{H}_0(R_i)$. Abort the protocol with local output $\perp$ if this is not the case; otherwise, compute $R \leftarrow \prod_{i=1}^{n} R_i$, query $c_1 \leftarrow \mathrm{H}_1(X_1 \| R \| \langle L \rangle \| m)$ where $\langle L \rangle$ is a unique encoding of L (e.g. the sequence of keys in lexicographic order), and compute $s_1 \leftarrow x_1 c_1 + r_1 \bmod p$.

– Send to signer i: s_1

Round 4:

– Receive from signer i: s_i

– Computation: $s \leftarrow \sum_{i=1}^{n} s_i \bmod p$

– Local output: the signature $\sigma = (R, s)$

Verification. Given a multiset of public keys $L = \{X_1, \ldots, X_n\}$, message m and signature $\sigma = (R, s)$, the verifier computes $c_i \leftarrow \mathrm{H}_1(X_i \| R \| \langle L \rangle \| m)$ for all $1 \leq i \leq n$. He accepts the signature if $g^s = R \prod_{i=1}^{n} X_i^{c_i}$, and rejects otherwise.

EFFICIENCY. An overview comparing the efficiency of our scheme to that of other (provably secure) multi-signature schemes is given in Table 1. Of course, one could argue whether a fair comparison between schemes achieving different security notions (KOSK assumption or not, random oracle or not) and relying on different computational assumptions (CDH versus DL, pairings or not) is possible at all. With that caveat in mind, we stress that the stronger security achieved by the MS-DL scheme does not come at the cost of efficiency. Compared to the MS-MOR scheme [MOR01], it avoids the expensive and impractical interactive key generation protocol, while offering considerably shorter public keys and maintaining the same signature length and signing/verification costs. Moreover, our scheme allows for concurrent signing sessions at the cost of one extra round of interaction (and the computation of some hash values). Compared to the MS-Bo scheme [Bol03], our scheme has about twice the signature size, yet offers faster verification. The MS-DL beats the MS-LOSSW scheme [LOS+06] in all costs, especially in the common parameter size that is particularly large for MS-LOSSW. Moreover, our scheme has the advantage over MS-Bo and MS-LOSSW of not relying on pairings to be defined over the underlying group.

We state the security of the MS-DL scheme that we presented above in the plain public-key model (i.e., without the KOSK assumption) in the theorem below. The proof makes use of the simplified Forking Lemma (Lemma 3.1) that we presented in Section 3.

Theorem 6.1 If there exists a $(t, q_{\mathrm{S}}, q_{\mathrm{H}}, n_{\max}, \epsilon)$-forger F in the random oracle model against the multi-signature scheme MS-DL described above, then there exists an algorithm B that (t', ϵ')-breaks the discrete logarithm problem in $\mathbb{G}$, where

$$\begin{aligned} \epsilon' &\geq \frac{\epsilon^2}{q_{\mathrm{H}} + q_{\mathrm{S}}} - \frac{2q_{\mathrm{H}} + 16n_{\max}^2 q_{\mathrm{S}}}{2^{l_0}} - \frac{8n_{\max}q_{\mathrm{S}}}{2^k} - \frac{1}{2^{l_1}} , \qquad (4) \\ t' &= 2t + q_{\mathrm{S}} t_{\exp} + O((q_{\mathrm{S}} + q_{\mathrm{H}})(1 + q_{\mathrm{H}} + n_{\max} q_{\mathrm{S}})) \end{aligned}$$

and $t_{\exp}$ is the time of an exponentiation in $\mathbb{G}$.

Proof: The idea of the proof is to use our Forking Lemma to obtain from the forger F two forgeries (R, s) and (R', s') satisfying

$$g^s = R \prod_{i=1}^{n} X_i^{c_i} \quad \text{and} \quad g^{s'} = R \prod_{i=1}^{n} X_i^{c'_i} ,$$

such that $c_i = c'_i$ if X_i is the target public key X^*, and $c_i \neq c'_i$ for all other keys. We can then extract the discrete logarithm of X^* by dividing the two equations above. Special care must be taken however in responding F's random oracle queries so that the above relations between c_i and c'_i are ensured. In particular, F may not ask the queries defining c_i and c'_i until *after* the fork, where the two executions of F have already diverged. We overcome this by fixing the response values to these queries *before* the fork, and by recognizing the queries when they actually occur after the fork. It is mainly due to the modularity of our simplified Forking Lemma that the complexity of the proof is kept manageable.

We are now ready to present the actual proof. Given a $(t, q_S, q_H, n_{\max}, \epsilon)$-forger F, consider the following algorithm A. On inputs $g \in \mathbb{G}^*$, $X^* \in \mathbb{G}$ and $h_1, \ldots, h_{q_H+q_S} \in \{0,1\}^{l_1}$, algorithm A runs the forger F on input system parameters $par = (\mathbb{G}, p, g)$ and target public key $pk^* = X^*$. Algorithm A initializes counters ctr_1, ctr_2 to zero, and maintains initially empty associative arrays $T_0[\cdot]$, $T_1[\cdot,\cdot]$, $T_2[\cdot]$. It assigns $T_2[X^*] \leftarrow 0$ and answers F's oracle queries as follows. Tables T_0 and T_1 are used to simulate random oracles H_0 and H_1, respectively, while T_2 assigns a unique index $1 \le i \le q_H + n_{\max} q_S$ to each public key X occurring either as a cosigner's public key in one of F's signature queries, or as the first item in the argument of one of F's queries to H_1. Algorithm A assigns index 0 to the target public key X^* by setting $T_2[X^*] \leftarrow 0$. It responds to F's oracle queries as follows:

- $\mathrm{H}_0(R)$: If $T_0[R]$ is undefined, then A chooses $T_0[R] \xleftarrow{\$} \{0,1\}^{l_0}$. It returns $T_0[R]$ to F.

- $\mathrm{H}_1(X\|Q)$: If $T_2[X]$ is undefined then A increases ctr_2 and sets $T_2[X] \leftarrow ctr_2$. Let $i = T_2[X]$. If $T_1[i, Q]$ has not yet been defined, then A immediately assigns random values to *all* entries $T_1[j, Q]$ for $1 \le j \le q_H + n_{\max} q_S$, increases ctr_1 and assigns $T_0[0, Q] \leftarrow h_{ctr_1}$. (If the argument of the query cannot be parsed as $X\|Q$, then A simply returns a random element of $\{0,1\}^{l_1}$, preserving consistency if the same query is asked again.)

- Signing query with public keys L and message m: If $X^* \notin L$ then algorithm A returns $\perp$ to F; otherwise, it parses the elements of L as $\{X_1 = X^*, X_2, \ldots, X_n\}$ and continues as follows. First, it checks for all $2 \le i \le n$ whether $T_2[X_i]$ has already been defined; it increases ctr_2 and sets $T_2[X_i] \leftarrow ctr_2$ if not. Then, A increases counter ctr_1 and sets $c_1 \leftarrow h_{ctr_1}$. It chooses $s_1 \xleftarrow{\$} \mathbb{Z}_p$, computes $R_1 \leftarrow g^{s_1} X_1^{-c_1}$ and sends $t_1 = \mathrm{H}_0(R_1)$ to all cosigners.

  After receiving $t_2, \ldots, t_n$ from F (who's playing the role of the cosigners), A searches in table T_0 for the values R_i so that $t_i = T_0[R_i]$. If no such R_i can be found for some $2 \le i \le n$, then A sets a flag $alert \leftarrow \texttt{true}$ and sends R_1 to all cosigners. If more than one such value is found for some R_i, then it sets $bad_1 \leftarrow \texttt{true}$, aborts the execution of F and halts with output $(0, \varepsilon)$. Otherwise, A computes $R \leftarrow \prod_{i=1}^{n} R_i$ and checks whether $T_1[0, Q]$ has already been defined for $Q = R\|\langle L\rangle\|m$. If so, it sets $bad_2 \leftarrow \texttt{true}$, aborts the execution of F and halts with output $(0, \varepsilon)$. If not, it sets $T_1[0, Q] \leftarrow c_1$, chooses $T_1[i, Q] \xleftarrow{\$} \{0,1\}^{l_1}$ for all $1 \le i \le q_H + n_{\max} q_S$, and sends R_1 to all cosigners.

  If, after receiving $R_2, \ldots, R_n$, there exists an index $1 \le i \le n$ such that $\mathrm{H}_0(R_i) \ne t_i$, then A stops the signing protocol returning $\perp$. If $alert = \texttt{true}$ while $\mathrm{H}_0(R_i) = t_i$ for all $1 \le i \le n$, then it sets $bad_3 \leftarrow \texttt{true}$, aborts the execution of F and halts outputting $(0, \varepsilon)$. Otherwise, it sends s_1 to all cosigners.

  After receiving $s_2, \ldots, s_n$, A computes $s \leftarrow \sum_{i=1}^{n} s_i \bmod p$ and returns (R, s) as the signature.

Eventually, F outputs a forged signature (R, s) together with multiset $L = \{X_1, \ldots, X_n\}$ and message m. Algorithm A first performs additional random oracle queries $\mathrm{H}_1(X_i\|R\|\langle L\rangle\|m)$ for $1 \le i \le n$, thereby making sure that $T_2[X_i]$ is defined. Let $1 \le J \le q_H + q_S$ be the index such that

$T_1[0, R\|\langle L\rangle\|m] = h_J$, and let n^* be the number of times that X^* occurs in L. If F's forgery is valid, algorithm A halts returning $(J, (R, h_J, s, n^*))$; if not, it halts returning $(0, \varepsilon)$.

Consider set $H = \{0,1\}^{l_1}$ and input generator IG that returns random elements $g, X^* \xleftarrow{\$} \mathbb{G}$. We bound the accepting probability acc of A with respect to these, as defined in Lemma 3.1, as follows:

$$\begin{aligned}
\mathrm{acc} \;&\geq\; \epsilon - \Pr[\, bad_1 = \mathtt{true}\,] - \sum_{i=1}^{q_\mathrm{S}} \Big(\Pr[\, bad_2 = \mathtt{true} \text{ during } i\text{-th } \textsc{Sign} \text{ query}\,] \\
&\qquad + \Pr[\, bad_3 = \mathtt{true} \text{ during } i\text{-th } \textsc{Sign} \text{ query}\,] \Big) \\
&\geq\; \epsilon - \frac{(q_\mathrm{H} + n_\mathrm{max} q_\mathrm{S} + 1)^2}{2^{l_0+1}} - \sum_{i=1}^{q_\mathrm{S}} \left(\frac{q_\mathrm{H} + n_\mathrm{max} q_\mathrm{S}}{2^k} + \frac{q_\mathrm{H} + q_\mathrm{S}}{2^k} + \frac{n_\mathrm{max}}{2^{l_0}} \right) \\
&\geq\; \epsilon - \frac{(q_\mathrm{H} + n_\mathrm{max} q_\mathrm{S} + 1)^2}{2^{l_0}} - \frac{2q_\mathrm{S}(q_\mathrm{H} + n_\mathrm{max} q_\mathrm{S})}{2^k}\ .
\end{aligned}$$

We clarify how the bounds in the second inequality were obtained. If at some point in the execution of F two values $R_i \neq R'_i$ are found such that $t_i = \mathrm{H}_0(R_i) = \mathrm{H}_0(R'_i)$, then clearly at least one collision must have occurred in H_0. However, all response values of H_0 are chosen uniformly at random from $\{0,1\}^{l_0}$, and since there are at most $q_\mathrm{H} + n_\mathrm{max} q_\mathrm{S}$ queries to H_0, the probability that at least one collision occurs is at most $((q_\mathrm{H} + n_\mathrm{max} q_\mathrm{S})(q_\mathrm{H} + n_\mathrm{max} q_\mathrm{S} + 1)/2)/2^{l_0} \leq (q_\mathrm{H} + n_\mathrm{max} q_\mathrm{S} + 1)^2/2^{l_0}$.

To cause $bad_2 = \mathtt{true}$ in the i-th signing query, we distinguish between the case that $\mathrm{H}_0(R_1)$ was previously queried by the forger, and the case that it wasn't. In the first case, F probably knows R and may have deliberately queried $\mathrm{H}_1(X\|R\|\langle L\rangle\|m)$ for some X. But since R_1 was chosen by A independently from F's view at the beginning of the signing protocol, the probability that F queried $\mathrm{H}_0(R_1)$ is at most $(q_\mathrm{H} + n_\mathrm{max} q_\mathrm{S})/p \leq (q_\mathrm{H} + n_\mathrm{max} q_\mathrm{S})/2^k$. In the second case, F's view is completely independent of R_1, and hence of R. The probability that R occurred by chance in a previous query to H_1 or was set by A in one of the $i-1$ previous signature simulations is at most $(q_\mathrm{H} + q_\mathrm{S})/p \leq (q_\mathrm{H} + q_\mathrm{S})/2^k$.

Lastly, in order to set $bad_3 = \mathtt{true}$, F must have predicted the value of $\mathrm{H}_0(R_i)$ for at least one $1 \leq i \leq n$, which it can do with probability at most $n_\mathrm{max}/2^{l_0}$. The third inequality follows from simple rearranging of terms after assuming (without loss of generality) that $q_\mathrm{H}, q_\mathrm{S}, n_\mathrm{max} > 0$.

Now consider an algorithm B that on input X^* runs the forking algorithm $\mathsf{F}_\mathsf{A}(X^*)$, which with probability frk returns $(1, (R, h, s, n^*), (R', h', s', n'^*))$ with $h \neq h'$. These forgeries are such that

$$g^s \;=\; R\prod_{i=1}^{n} X_i^{c_i} \qquad \text{and} \qquad g^{s'} \;=\; R'\prod_{i=1}^{n'} X_i'^{c'_i}$$

where $L = \{X_1, \ldots, X_n\}$ and m are the public keys and the message involved in F's first forgery, and $c_i = \mathrm{H}_1(X_i\|R\|\langle L\rangle\|m)$ are the relevant random oracle responses from the first run. Let $I^* \subseteq \{1, \ldots, n\}$ be the set of indices such that $X_i = X^*$. Variables $L', X'_1, \ldots, X'_{n'}, m', c'_1, \ldots, c'_{n'}, I'^*$ are defined analogously for the second run of F. We will show later that, due to the way that A simulates F's environment, it must hold that $n = n'$, that $L = L'$, that $I^* = I'^*$, that $n^* = n'^*$, that $c_i = c'_i$ for $i \notin I^*$, and that $c_i = h$ and $c'_i = h'$ for $i \in I^*$. Dividing the two equations above then gives

$$g^{s-s'} = \prod_{i\in I^*} (X^*)^{h-h'} = (X^*)^{n^*(h-h')}\ ,$$

so that B can compute the discrete logarithm of X^* as $(s - s')/(n^*(h - h'))$ mod p. The probability that algorithm B succeeds in doing so is given by

$$\begin{aligned}
\epsilon' &\geq \mathrm{frk} \\
&\geq \frac{\mathrm{acc}^2}{q_\mathrm{H}+q_\mathrm{S}} - \frac{1}{2^{l_1}} \\
&\geq \frac{\epsilon^2}{q_\mathrm{H}+q_\mathrm{S}} - \frac{2(q_\mathrm{H}+n_\mathrm{max}q_\mathrm{S}+1)^2}{(q_\mathrm{H}+q_\mathrm{S})\cdot 2^{l_0}} - \frac{4q_\mathrm{S}(q_\mathrm{H}+n_\mathrm{max}q_\mathrm{S})}{(q_\mathrm{H}+q_\mathrm{S})\cdot 2^k} - \frac{1}{2^{l_1}} \\
&\geq \frac{\epsilon^2}{q_\mathrm{H}+q_\mathrm{S}} - \frac{2q_\mathrm{H}+16n_\mathrm{max}^2q_\mathrm{S}}{2^{l_0}} - \frac{8n_\mathrm{max}q_\mathrm{S}}{2^k} - \frac{1}{2^{l_1}} ,
\end{aligned}$$

where again we assume without loss of generality that $q_\mathrm{H}, q_\mathrm{S}, n_\mathrm{max} > 0$. The theorem follows.

We still have to argue why the equalities between all the variables in both runs of A hold. In the case that F returned $(1,(R,h,s,n^*),(R,h',s',n'^*))$, let J be the index that A returned after both executions by F_A. In A's first execution, $h_J = h$ is assigned to $T_1[0, R\|\langle L\rangle\|m] = \mathrm{H}_1(X^*\|R\|\langle L\rangle\|m)$ at the moment when F makes its first query $\mathrm{H}_1(X\|R\|\langle L\rangle\|m)$ for *some* public key X (so not necessarily X^*), where $L = \{X_1,\ldots,X_n\}$. Likewise, in the second run, $h'_J = h'$ is assigned to $T_1[0, R'\|\langle L'\rangle\|m']$ when F queries $\mathrm{H}_1(X'\|R'\|\langle L'\rangle\|m')$ for some public key X', where $L' = \{X'_1,\ldots,X'\}$. Up to the point of this hash query, however, the environments of F provided by A in the first and the second run are identical, because A uses the same inputs, random tape and values $h_1,\ldots,h_{J-1}$ to generate F's inputs, random tape and oracle responses. Therefore, the two executions of F are identical up to this point, and in particular the arguments of both hash queries must be the same, implying that $R = R'$, $L = L'$, $n = n'$, $X_i = X'_i$ and $m_i = m'_i$ for $1 \leq i \leq n$. Moreover, the entries for $X_1,\ldots,X_n$ in T_2 are assigned *at the latest* when parsing the arguments of this hash query, causing the values of $T_2[X_i]$ to be the same in both runs as well. The forger's other queries $\mathrm{H}_1(X_i\|R\|\langle L\rangle\|m)$ may not occur until much later, but the response values $T_1[T_2[X_i], R\|\langle L\rangle\|m]$ for these queries are chosen *before* the fork, and hence are the same in both runs as well. Therefore, it holds that $c_i = c'_i$ for all $1 \leq i \leq n$, that $c_i = h_J = h$ for $i \in I$, and that $c'_i = h'_J = h'$ for $i \in I$, which concludes the proof.

The running time t' of B is twice that of A plus the time needed to compute the discrete logarithm $(s-s')/(n^*(h-h')) \bmod p$. The running time of A is the running time t of F plus the time needed to answer $q_\mathrm{H}+n_\mathrm{max}q_\mathrm{S}$ random oracle queries and q_S signature queries. We assume that exponentiations in $\mathbb{G}$ take time t_exp, and all other operations take unit time. Each random oracle query may cause A to perform $O(1+q_\mathrm{H}+n_\mathrm{max}q_\mathrm{S})$ unit-time operations. Each signature query involves one multi-exponentiation in $\mathbb{G}$ and $O(1+q_\mathrm{H}+n_\mathrm{max}q_\mathrm{S})$ unit-time operations. Therefore, we have $t' = 2t + q_\mathrm{S}t_\mathrm{exp} + O((q_\mathrm{S}+q_\mathrm{H})(1+q_\mathrm{H}+n_\mathrm{max}q_\mathrm{S}))$. ▮

Reduction tightness. The reduction of the $\mathsf{MS\text{-}DL}$ scheme presented above is tighter than that of the $\mathsf{MS\text{-}MOR}$ scheme [MOR01], but is still not tight, as can be seen from Equation (4). In comparison, the security proof of the $\mathsf{MS\text{-}MOR}$ scheme requires two applications of the forking technique (once to extract the secret keys of corrupt players, and once to obtain two forgeries) and $q_\mathrm{H}\cdot q_\mathrm{S}$ rewindings (to simulate signing protocols) of the forger, yielding a considerable loss in the tightness of the security reduction. The pairing-based $\mathsf{MS\text{-}Bo}$ and $\mathsf{MS\text{-}LOSSW}$ schemes do not have tight security reductions either.

7 A Scheme with Tight Reduction to DDH

The security reduction of the $\mathsf{MS\text{-}DL}$ scheme presented in the previous section is tighter than that of the $\mathsf{MS\text{-}MOR}$ scheme [MOR01], but is still not tight, as can be seen from Equation (4). In the following, we present a scheme with a tight security reduction to the decisional Diffie-Hellman problem. The scheme is based on the Katz-Wang standard signature scheme [KW03], which on its turn was based on a previously proposed scheme [CV90, GJ03].

THE DDH ASSUMPTION. Let $\mathbb{G}$ be a multiplicative group of prime order p, and let $\mathbb{G}^* = \mathbb{G} \setminus \{1\}$. The advantage of an algorithm A in solving the decisional Diffie-Hellman (DDH) problem in $\mathbb{G}$ is defined as

$$\begin{aligned}\mathbf{Adv}^{\mathrm{ddh}}_{\mathbb{G}}(\mathsf{A}) \;=\; \Big| &\Pr\Big[\,\mathsf{A}(g,h,g^x,h^x)=1 \;:\; g,h \xleftarrow{\$} \mathbb{G}^*\;;\; x \xleftarrow{\$} \mathbb{Z}_p\,\Big] \\ &- \Pr\Big[\,\mathsf{A}(g,h,g^x,h^y)=1 \;:\; g,h \xleftarrow{\$} \mathbb{G}^*\;;\; x,y \xleftarrow{\$} \mathbb{Z}_p\,\Big]\Big|\end{aligned}$$

We say that A (t,ϵ)-solves the DDH problem in $\mathbb{G}$ if it runs in time at most t and has advantage $\mathbf{Adv}^{\mathrm{ddh}}_{\mathbb{G}}(\mathsf{A}) \geq \epsilon$, and we say that the DDH problem in $\mathbb{G}$ is (t,ϵ)-hard if no algorithm A (t,ϵ)-solves it.

THE SCHEME. Let $k = \lfloor \log_2 p \rfloor$ and let $l_0 \in \mathbb{N}$. Let $\mathrm{H}_0 : \{0,1\}^* \to \{0,1\}^{l_0}$ and $\mathrm{H}_1 : \{0,1\}^* \to \mathbb{Z}_p$ be random oracles, where the range of H_1 depends on the common parameters. To these, we associate the following multisignature scheme MS-DDH:

Parameter generation. The trusted parameter generator chooses random elements $g,h \xleftarrow{\$} \mathbb{G}^*$ and publishes $(\mathbb{G},p,g,h)$ as the system-wide parameters.

Key generation. On input parameters $(\mathbb{G},p,g,h)$, each signer runs the Kg algorithm to generate a secret key $x \xleftarrow{\$} \mathbb{Z}_p$ and computes the corresponding public key consisting of $X \leftarrow g^x$ and $Y \leftarrow h^x$.

Signing. On input secret key x_1, corresponding public key $(X_1 = g^{x_1}, Y_1 = h^{x_1})$, message m and the cosigner's public keys $(X_2,Y_2),\ldots,(X_n,Y_n)$, a signer proceeds as follows.

Round 1:

- Local input: x_1, $L = \{(X_1,Y_1),\ldots,(X_n,Y_n)\}$, m
- Computation: Choose $r_1 \xleftarrow{\$} \mathbb{Z}_p$, compute $A_1 \leftarrow g^{r_1}$ and $B_1 \leftarrow h^{r_1}$, and query $t_1 \leftarrow \mathrm{H}_0(A_1 \| B_1)$.
- Send to signer i: t_1

Round 2:

- Receive from signer i: t_i
- Send to signer i: A_1, B_1

Round 3:

- Receive from signer i: A_i, B_i
- Computation: Check that $t_i = \mathrm{H}_0(A_i \| B_i)$ for all $2 \leq i \leq n$. Abort the protocol with local output $\perp$ if this is not the case. Otherwise, compute $A \leftarrow \prod_{i=1}^n A_i$, $B \leftarrow \prod_{i=1}^n B_i$, $c_1 \leftarrow \mathrm{H}_1(X_1 \| Y_1 \| A \| B \| \langle L \rangle \| m)$, and $s_1 \leftarrow x_1 c_1 + r_1 \bmod p$.
- Send to signer i: s_1

Round 4:

- Receive from signer i: s_i
- Computation: $s \leftarrow \sum_{i=1}^n s_i \bmod p$
- Local output: the signature $\sigma = (A,B,s)$

Verification. On input a multiset of public keys $L = \{(X_1,Y_1),\ldots,(X_n,Y_n)\}$, message m and candidate signature $\sigma = (A,B,s)$, the verifier computes $c_i \leftarrow \mathrm{H}_1(X_i \| Y_i \| A \| B \| \langle L \rangle \| m)$ for all $1 \leq i \leq n$. He accepts the signature if and only if $g^s = A \prod_{i=1}^n X_i^{c_i}$ and $h^s = B \prod_{i=1}^n Y_i^{c_i}$.

Theorem 7.1 If the DDH problem in $\mathbb{G}$ is (t', ϵ')-hard, then the $\mathsf{MS\text{-}DDH}$ scheme is $(t, q_S, q_H, n_{\max}, \epsilon)$-secure whenever $t' \geq t + (q_S + 2)t_{\exp} + O(q_H + (n_{\max} + 1)q_S + 1)$ and

$$\epsilon' \;\leq\; \epsilon - \frac{(q_H + n_{\max}q_S + 1)^2}{2^{l_0}} - \frac{2q_S(q_H + n_{\max}q_S + 2)}{2^k} \;. \blacksquare$$

Proof: Given forger F, consider the following algorithm A solving the DDH problem. On inputs (h, X^*, Y^*), A has to decide whether $\log_g X^* = \log_h Y^*$. Algorithm A maintains initially empty associative arrays $T_0[\cdot]$, $T_1[\cdot]$, and runs F on inputs $par = (\mathbb{G}, p, g, h)$ and $pk^* = (X^*, Y^*)$, answering F's oracle queries as follows:

- $H_0(\cdot)$ and $H_1(\cdot)$: Algorithm A returns random strings chosen from $\{0,1\}^{l_0}$ and $\mathbb{Z}_p$, respectively, maintaining consistency of responses for repeating queries as usual using associative arrays $T_0[\cdot]$ and $T_1[\cdot]$.

- Signing query with public keys L and message m: If $(X^*, Y^*) \notin L$, then A returns $\perp$ to F, else let $L = \{(X_1, Y_1) = (X^*, Y^*), (X_2, Y_2), \ldots, (X_n, Y_n)\}$. Algorithm A chooses $c_1, s_1 \overset{\$}{\leftarrow} \mathbb{Z}_p$, computes $A_1 \leftarrow g^{s_1} X_1^{-c_1}$, $B_1 \leftarrow h^{s_1} Y_1^{-c_1}$, $t_1 \leftarrow H_0(A_1 \| B_1)$, and sends t_1 to all cosigners.

  After having received $t_2, \ldots, t_n$ from F, A looks up A_i, B_i such that $T_0[A_i \| B_i] = t_i$. If for some $2 \leq i \leq n$ no such values A_i, B_i exist, A sets a flag $alert \leftarrow \texttt{true}$, and sends A_1, B_1 to all cosigners. If multiple possibilities A_i, B_i are found for some i, then A aborts the execution of F and outputs 0. Otherwise, A computes $A \leftarrow \prod_{i=1}^n A_i$, $B \leftarrow \prod_{i=1}^n B_i$ and sets $T_1[X^* \| Y^* \| A \| B \| \langle L \rangle \| m] \leftarrow c_1$, or it aborts outputting 0 if this entry in T_1 was already defined. A sends A_1, B_1 to all other cosigners.

  After having received A_i, B_i for all $2 \leq i \leq n$, A checks that $t_i = H_0(A_i \| B_i)$ for $1 \leq i \leq n$. If one of these tests fails, A halts this signing protocol returning $\perp$ to F. If $alert = \texttt{true}$ yet all tests succeeded, then A aborts the execution of F and outputs 0. Otherwise, it sends s_1 to all cosigners.

  After having received s_i for all $2 \leq i \leq n$, A computes $s \leftarrow \sum_{i=1}^n s_i \bmod p$, and returns (A, B, s) to F as the signature.

When F outputs its forged signature (A, B, s) for public keys L and message m, A performs one additional hash query $H_1(X^* \| Y^* \| A \| B \| \langle L \rangle \| m)$. If the forgery is valid, meaning that $\mathsf{Vf}(L, m, (A, B, s)) = 1$, $(X^*, Y^*) \in L$ and F never queried (L, m) to the signing oracle, then algorithm A outputs 1, otherwise it outputs 0.

If (h, X^*, Y^*) is a Diffie-Hellman tuple, then the environment provided to F by A is a perfect simulation of a real attack environment, unless A aborts the execution of F prematurely. We analyze the probability that A aborts. If A halts during some signing query due to multiple values A_i, B_i being found such that $T_0[A_i \| B_i] = t_i$, then there must be at least two different entries in T_0 that were assigned the same value. Since all $q_H + n_{\max}q_S$ values in $T_0[\cdot]$ are chosen uniformly at random from $\{0,1\}^{l_0}$, this happens with probability at most $((q_H + n_{\max}q_S + 1)^2)/2^{l_0}$.

We analyze the probability that A aborts during the i-th signing query due to an entry of T_1 already being defined when it wants to program $T_1[X^* \| Y^* \| A \| B \| \langle L \rangle \| m] \leftarrow c_1$. We distinguish between the case that F previously queried $H_0(A_1 \| B_1)$, and the case that it did not. In the first case, we must assume that A knows A, B, but since A_1 was chosen at random and independent of F's view at the beginning of the protocol, the probability that F submitted this query to H_0 is at most $((q_H + n_{\max}q_S + 1)^2/2)/2^k$. In the second case, F's view is completely independent of A, B, and the probability that $A\|B$ appeared as part of a previous query to H_1 is at most $(q_{\mathrm{H}}+q_{\mathrm{S}})/2^k$. Finally, in order to make A abort because *alert* is set while $\mathrm{H}_0(A_i\|B_i)=t_i$ for all $1\le i\le n$, F must have predicted at least one response of H_0, which it can do with probability at most $n_{\max}/2^{l_0}$. Therefore, we have that

$$\begin{aligned}
&\Pr\Big[\,\mathsf{A}(h,g^x,h^x)=1\ :\ h\xleftarrow{\$}\mathbb{G}\,;\,x\xleftarrow{\$}\mathbb{Z}_p\,\Big]\\
&\geq\ \epsilon-\frac{(q_{\mathrm{H}}+n_{\max}q_{\mathrm{S}}+1)^2}{2^{l_0+1}}-\sum_{i=1}^{q_{\mathrm{S}}}\left(\frac{q_{\mathrm{H}}+n_{\max}q_{\mathrm{S}}+1}{2^k}+\frac{q_{\mathrm{H}}+q_{\mathrm{S}}}{2^k}+\frac{n_{\max}}{2^{l_0}}\right)\\
&\geq\ \epsilon-\frac{(q_{\mathrm{H}}+n_{\max}q_{\mathrm{S}}+1)^2}{2^{l_0}}-\frac{2q_{\mathrm{S}}(q_{\mathrm{H}}+n_{\max}q_{\mathrm{S}}+1)}{2^k}\ ,
\end{aligned}$$

where in the last inequality we assume without loss of generality that $n_{\max}>0$. If $(h,X^*=g^x,Y^*=g^y)$ is a random tuple, then $x\neq y$ with probability $1-1/p\geq 1-1/2^k$. In the case that $x\neq y$, we show that it is information-theoretically impossible for F to output a valid forgery. Let $(A=g^a,B=g^b,s)$ be a valid forgery for public keys $L=\{(X_1=g^{x_1},Y_1=g^{y_1}),\ldots,(X_n=g^{x_n},Y_n=h^{y_n})\}$ and message m. Then we have

$$s\ =\ a+\sum_{i=1}^{n}c_ix_i\ =\ b+\sum_{i=1}^{n}c_iy_i \bmod p$$

where $c_i=\mathrm{H}_1(X_i\|Y_i\|A\|B\|\langle L\rangle\|m)$ for $1\le i\le n$. Consider the partition $\{I_1,\ldots,I_{n'}\}$ of the indices $\{1..n\}$ so that two indices i_1,i_2 belong to the same component I_j if $(X_{i_1},Y_{i_1})=(X_{i_2},Y_{i_2})$. Moreover, define $(\overline{X}_j,\overline{Y}_j,\overline{x}_j,\overline{y}_j,\overline{c}_j)$ to be (X_i,Y_i,x_i,y_i,c_i) for any $i\in I_j$. (Note that X_i,Y_i,x_i,y_i,c_i are the same for all indices i belonging to the same component I_j.) Let I_1 be the component of the target public key, i.e. $(\overline{X}_1,\overline{Y}_1)=(X^*,Y^*)$. Then the equation above can be rewritten as the following linear equation in unknowns $\overline{c}_j$, $j=1..n'$:

$$|I_1|\cdot(\overline{x}_1-\overline{y}_1)\cdot\overline{c}_1+\sum_{j=2}^{n'}|I_j|\cdot(\overline{x}_j-\overline{y}_j)\cdot\overline{c}_j+(a-b)=0 \bmod p\ . \tag{5}$$

The argument of F's random oracle query defining $\overline{c}_j$ contains A, B, $\overline{X}_j$ and $\overline{Y}_j$ for $j=1..n'$, so the forger is forced to fix values for a, b, $\overline{x}_j$ and $\overline{y}_j$ for all $j=1..n'$ before seeing the value of $\overline{c}_j$ for any $1\le j\le n'$. Hence, the game can be seen as if the forger first chooses $\overline{x}_j,\overline{y}_j$ for $j=2..n'$ in Equation (5), and then the random oracle chooses a random vector $(\overline{c}_j)_{1\le j\le n'}\xleftarrow{\$}\mathbb{Z}_p^n$. We want to bound the probability that the vector chosen by the random oracle is a solution of Equation (5). From our hypothesis that $x\neq y$ and $|I_1|>0$ follows that the coefficient of $\overline{c}_1$ is certainly non-zero, so the equation has $p^{n'-1}$ solutions in $\mathbb{Z}_p^{n'}$. The probability that a randomly chosen vector $(\overline{c}_j)_{1\le j\le n}$ is a solution is at most $p^{n'-1}/p^{n'}\le 1/2^k$. Therefore, we have that

$$\Pr\Big[\,\mathsf{A}(h,g^x,h^y)=1\ :\ h\xleftarrow{\$}\mathbb{G}\,;\,x,y\xleftarrow{\$}\mathbb{Z}_p\,\Big]\ \leq\ \frac{1}{2^k}+\frac{1}{2^k}=\frac{2}{2^k}\ ,$$

so that the success probability of A in solving the DDH problem in $\mathbb{G}$ is at least

$$\epsilon'\ \geq\ \epsilon-\frac{(q_{\mathrm{H}}+n_{\max}q_{\mathrm{S}}+1)^2}{2^{l_0}}-\frac{2q_{\mathrm{S}}(q_{\mathrm{H}}+n_{\max}q_{\mathrm{S}}+2)}{2^k}\ ,$$

assuming without loss of generality that $q_{\mathrm{S}}>0$. The running time t' of A is the running time of F plus the time needed to verify the forgery and to answer $q_{\mathrm{H}}+n_{\max}q_{\mathrm{S}}$ random oracle queries and

q_S signature queries. We assume that (multi-)exponentiations in $\mathbb{G}$ take time $t_{\exp}$ and all other operations take unit time. Then each random oracle takes $O(1)$ unit-time operations, each signature query takes two exponentiations and $O(n_{\max} + 1)$ unit-time operations, and the verification takes two multi-exponentiations. Hence we have that $t' = t + (q_S + 2)t_{\exp} + O(q_H + n_{\max}q_S + q_S + 1)$. ▮

8 An Identity-Based Scheme from RSA

The primary purpose of multi-signatures is to save on communication costs. To verify the signatures, however, one still needs the public keys of all cosigners, each of which may come with an associated certificate. Transmitting, storing and verifying these n public keys and certificates partly defeats the cost-saving purpose of multi-signatures. One cannot always avoid the need to include some information identifying the cosigners, but usually this can be done more compactly than through their public keys, e.g. through their user names or IP addresses.

In an identity-based signature scheme [Sha85], the public key of a user is simply his identity, eliminating the need for explicit certification and all associated costs. Moreover, an identity may very well be shorter than a randomly generated public key, further saving on communication and storage costs. This makes the identity-based setting particularly appealing for use in conjunction with multi-signatures, yielding the concept of *identity-based multi-signature* (IBMS) schemes.

Existing implementations. Even though IBMS schemes are compelling with regard to applications, their implementations are limited. As shown in [DKXY03, BNN04], any standard signature scheme can be transformed into an identity-based one using the "certification paradigm". One can attempt to derive IBMS schemes from existing standard MS schemes via this approach. The problem is that the resulting multi-signature is not compact due to the need to include the certificates with each signature. Even if the signatures in the certificates can be aggregated [BGLS03], the public keys they contain cannot. In summary, unlike the case of standard signatures, there seems no trivial, general way to transform compact signature schemes into identity-based ones.

The first fully secure IBMS scheme was recently proposed by Gentry and Ramzan [GR06]. The scheme by Cheng et al. [CLW05] was only proved secure under a weak variant of selective-ID security. Both schemes employ groups with bilinear maps (also known as pairings). It is common practice in cryptography to try to find alternative constructions of a primitive based on different assumptions. While pairings turned out to be extremely useful in the design of cryptographic protocols, they were only recently brought to the attention of cryptographers [Jou00], and hence did not yet enjoy the same exposure to cryptanalytic attacks by experts as other, older problems from number theory like discrete logarithms, factoring and RSA. Moreover, companies may have invested in expensive hardware or software implementations of RSA, and may be reluctant to dump these in favor of new implementations of pairings.

Contributions of our IBMS scheme. In this section, we present an efficient and provably secure IBMS scheme based on RSA, which is thereby the first IBMS scheme not relying on the use of pairings. Our scheme is essentially a multi-signature variant of the Guilliou-Quisquater [GQ90] identity-based signature scheme. For 1024-bit RSA moduli, signatures are typically 1184 bits long. This is longer than the 320-bit signatures of the scheme of [GR06], but verification is considerably cheaper: our scheme needs only a single (multi-)exponentiation in an RSA group, as opposed to three pairing computations for the scheme of [GR06]. The cost of one pairing computation is roughly that of 6–20 exponentiations.

Syntax and security. We extend the syntax and security notion of interactive multi-signature schemes to the identity-based setting as follows. A trusted key distribution center runs a Setup algorithm to generate a master public key mpk and corresponding master secret key msk. To a signer with identity $ID \in \{0,1\}^*$, it provides a secret key derived via $sk_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$. The signer can use this secret key to participate in signing protocols as prescribed by the Sign algorithm, which takes as additional input a multiset $L = \{ID_1, \ldots, ID_n\}$ containing the identities of all cosigners and the message m. Verification is done by running the Vf algorithm on input mpk, a multiset of identities L, a message m and a candidate signature σ. Correctness requires that for all $n \in \mathbb{N}$, for all (mpk, msk) output by Setup, for all $ID_1, \ldots, ID_n \in \{0,1\}^*$ and for all messages $m \in \{0,1\}^*$, each signer i outputs a signature σ_i such that $\mathsf{Vf}(mpk, L, m, \sigma_i) = 1$ with probability one whenever all signers follow the signing protocol using secret key $sk_i \xleftarrow{\$} \mathsf{KeyDer}(msk, ID_i)$.

Our security notion is more involved than that of [GR06] because we need to model interactive signing protocols. (The scheme of [GR06] is non-interactive.) As for standard multi-signatures, security requires that it be infeasible to forge a multi-signature involving at least one honest signer. We consider a very strong security notion where the adversary can adaptively decide to corrupt honest signers, resulting in it being given their secret keys, and can engage in any number of arbitrarily interleaved signing protocols. In each protocol instance, the adversary can choose to interact with any number of honest signers, while itself plays the role of all other signers involved in the protocol. It can do so with or without knowing their secret keys (i.e., the adversary need not have corrupted a signer prior to playing its role in a protocol) and can deviate from the rules prescribed by the Sign algorithm.

We stress that the capability to interact with *multiple* honest signers in a single protocol instance is important in the identity-based setting. In the public-key setting, we could assume without loss of generality that all signers are corrupted except for one "target" signer. This does not hold in the identity-based setting however. The adversary may want to let its choice which identities to corrupt and under which identity to forge depend on previous signature transcripts. Requiring the adversary to corrupt all but one of the participants before engaging in a signing protocol precludes such attacks. (The adversary can always try to play the role of a signer without knowing its secret key, but this protocol will not result in a valid signature unless the scheme is forgeable.)

More precisely, the forger is given the master public key mpk as input at the beginning of the game, and has access to a key derivation oracle $\mathsf{KeyDer}(msk, \cdot)$ which it can query up to q_{K} times. It can engage in up to q_{S} arbitrarily interleaved signing protocols with honest signers by submitting two multisets of identities $L_{\mathrm{h}}, L_{\mathrm{c}}$ and a message m to the signing oracle. The multiset L_{h} contains the honest signers, whose role will be played by the challenger according to the Sign algorithm. The forger plays the role of the (possibly) cheating signers contained in L_{c}.

We again assume that the adversary controls all network traffic, even between honest signers. To model this, each honest signer hands its outgoing messages to the adversary, who can then choose to read or modify the message, and even whether to deliver it or not. Since we do not assume the availability of a secure broadcast primitive, our security proof has to take into account that the adversary can set different honest signers up with a different view of the protocol by tampering with the messages sent to them.

Eventually, the forger outputs a signature σ, a multiset of identities L and a message m. The forger wins the game if $\mathsf{Vf}(mpk, L, m, \sigma) = 1$, if there exists an identity $ID \in L$ such that F never queried the key derivation oracle on ID, and if the F never performed a signature query $(L_{\mathrm{h}}, L_{\mathrm{c}}, m')$ so that $L = L_{\mathrm{h}} \cup L_{\mathrm{c}}$ and $m = m'$. The scheme is said to be $(t, q_{\mathrm{K}}, q_{\mathrm{S}}, n_{\max}, \epsilon)$-secure if no forger F running in time at most t, performing at most q_{K} key derivation and q_{S} signature queries with up to $n_{\max}$ signers, has probability greater than ϵ to win the above game. In the random oracle model, the KeyDer, Sign and Vf algorithms, as well as the adversary, additionally have access to one or more random oracles, which the adversary can query up to q_{H} times.

THE RSA ASSUMPTION. An *RSA key generator* $\mathsf{Kg}_{\mathrm{rsa}}$ is an algorithm that generates triplets (N, e, d) such that N is the product of two large primes and $ed \equiv 1 \bmod \varphi(N)$. The advantage of A in breaking the one-wayness of RSA related to $\mathsf{Kg}_{\mathrm{rsa}}$ is defined as

$$\mathbf{Adv}^{\text{ow-rsa}}_{\mathsf{Kg}_{\mathrm{rsa}}}(\mathsf{A}) = \Pr\left[x^e \equiv y \bmod N \; ; \; \begin{array}{c} (N, e, d) \xleftarrow{\$} \mathsf{Kg}_{\mathrm{rsa}} \; ; \; y \xleftarrow{\$} \mathbb{Z}_N^* \; ; \\ x \xleftarrow{\$} \mathsf{A}(N, e, y) \end{array} \right] .$$

We say that A (t, ϵ)-breaks the one-wayness of RSA with respect to $\mathsf{Kg}_{\mathrm{rsa}}$ if it runs in time at most t and has advantage $\mathbf{Adv}^{\text{ow-rsa}}_{\mathsf{Kg}_{\mathrm{rsa}}}(\mathsf{A}) \geq \epsilon$, and we say that the RSA function associated to $\mathsf{Kg}_{\mathrm{rsa}}$ is (t, ϵ)-one-way if no algorithm A (t, ϵ)-breaks it.

THE SCHEME. We now propose an identity-based multisignature scheme $\mathsf{IBMS\text{-}GQ}$ based on GQ signatures [GQ90] that we prove secure under the one-wayness of RSA. Let $l_0, l_1, l_N, \in \mathbb{N}$, and let $\mathrm{H}_0 : \{0,1\}^* \to \{0,1\}^{l_0}$, $\mathrm{H}_1 : \{0,1\}^* \to \{0,1\}^{l_1}$ and $\mathrm{H}_2 : \{0,1\}^* \to \mathbb{Z}_N^*$ be random oracles, where H_2 depends on the master public key of the scheme. Let $\mathsf{Kg}_{\mathrm{rsa}}$ be an RSA key pair generator that outputs triplets (N, e, d) such that $\varphi(N) > 2^{l_N}$ and with prime encryption exponents e of length strictly greater than $l_1 + \log_2 n_{\max}$ bits. To these, we associate the following identity-based multisignature scheme $\mathsf{IBMS\text{-}GQ}$.

Setup. The key distribution center runs $\mathsf{Kg}_{\mathrm{rsa}}$ to generate RSA parameters (N, e, d). It publishes $mpk = (N, e)$ as the master public key, and keeps the master secret key d secret.

Key derivation. On input master secret key d and signer identity ID, the key distribution center computes $x \leftarrow \mathrm{H}_2(ID)^d \bmod N$, and sends the user secret key x over a secure and authenticated channel to the signer with identity ID.

Signing. On input user secret key x_1 for identity ID_1, message m and cosigner identities $ID_2, \ldots, ID_n$, a signer proceeds as follows.

Round 1:

- Local input: x_1, $L = \{ID_1, \ldots, ID_n\}$, m
- Computation: Choose $r_1 \xleftarrow{\$} \mathbb{Z}_N^*$, compute $R_1 \leftarrow r_1^e \bmod N$ and $t_1 \leftarrow \mathrm{H}_0(R_1)$.
- Send to signer i: t_1

Round 2:

- Receive from signer i: t_i
- Send to signer i: R_1

Round 3:

- Receive from signer i: R_i
- Computation: Check that $t_i = \mathrm{H}_0(R_i)$ for all $2 \leq i \leq n$, and halt the protocol with local output $\perp$ if one of these tests fails. Otherwise, compute $R \leftarrow \prod_{i=1}^n R_i \bmod N$, $c \leftarrow \mathrm{H}_1(R \| \langle L \rangle \| m)$ and $s_1 \leftarrow r_1 x_1^c \bmod N$.
- Send to signer i: s_1

Round 4:

- Receive from signer i: s_i
- Computation: $s \leftarrow \prod_{i=1}^n s_i \bmod N$
- Local output: the signature $\sigma = (c, s)$

Verification. On input the master public key (N, e), a multiset of signer identities $L = \{ID_1, \ldots, ID_n\}$, a message m and a candidate signature (c, s), the verifier recomputes $R \leftarrow s^e \big(\prod_{i=1}^n \mathrm{H}_2(ID_i) \big)^{-c} \bmod N$. He accepts the signature as valid if $c = \mathrm{H}_1(R \| \langle L \rangle \| m)$, and rejects otherwise.

The length of a multi-signature is $l_1 + l_N$ bits, or about $160 + 1024 = 1184$ bits for typical values of the security parameter. Signing takes two exponentiations in $\mathbb{Z}_N^*$ for each signer, and verification takes a single (multi-)exponentiation, independent of the value of n. (Note that verification time is not completely independent of n due to the computation of $\prod_{i=1}^{n} \mathrm{H}_2(ID_i)$, but this is fast.)

The following theorem relates the unforgeability of our IBMS scheme to the one-wayness of the RSA problem associated to $\mathsf{Kg}_{\mathrm{rsa}}$. The proof is given below. We stress that we do not run into the key generation issues of non-identity-based schemes because keys are generated by the trusted center instead of by the signers themselves.

Theorem 8.1 If the RSA function associated to $\mathsf{Kg}_{\mathrm{rsa}}$ is (t', ϵ')-one-way, then the $\mathsf{IBMS\text{-}GQ}$ scheme is $(t, q_{\mathrm{K}}, q_{\mathrm{S}}, q_{\mathrm{H}}, n_{\max}, \epsilon)$-secure whenever $t' \geq 2t + (2q_{\mathrm{H}} + 2q_{\mathrm{K}} + 2q_{\mathrm{S}}(n_{\max}+1) + 2n_{\max} + 4) \cdot t_{\exp}$ and

$$\epsilon' \;\leq\; \frac{\epsilon^2}{16 q_{\mathrm{K}}^2 (q_{\mathrm{H}}+1)} \;-\; \frac{2q_{\mathrm{H}}^2 + 8n_{\max} q_{\mathrm{S}} q_{\mathrm{H}} + 8n_{\max}^2 q_{\mathrm{S}}^2}{2^{l_N}} - \frac{n_{\max} q_{\mathrm{S}}}{2^{l_0}} - \frac{1}{2^{l_1}} \;, \tag{6}$$

where $t_{\exp}$ is the time of an exponentiation in $\mathbb{Z}_N^*$.

Before proving the above theorem, we remark that our techniques can be applied to other identity-based signature schemes than the GQ scheme as well. In particular, one can obtain efficient IBMS schemes based on RSA from [Sha85], based on factoring from [FS87, FFS88, OO90, OS90], and based on pairings from [Hes03, CC03, Yi03]. An extensive overview of the security properties of these schemes as identity-based signature schemes can be found in [BNN04].

Proof: Given a forger F, consider the following algorithm A. On inputs $(N, e, y), h_1, \ldots, h_{q_{\mathrm{H}}+1}$, algorithm A runs F on inputs $mpk = (N, e)$.

Algorithm A maintains a counter ctr_1 with initial value 0 and initially empty associative arrays $T_0[\cdot]$, $T_1[\cdot,\cdot]$, $T_2[\cdot]$. It runs F on input $mpk = (N, e)$ and answers F's oracle queries as follows.

- $\mathrm{H}_0(R)$: If $T_0[R]$ is undefined, then A chooses $T_0[R] \stackrel{\$}{\leftarrow} \{0,1\}^{l_0}$. It returns $T_0[R]$ to F.
- $\mathrm{H}_1(Q)$: A returns $T_1[Q]$, increasing ctr_1 and setting $T_1[Q] \leftarrow h_{ctr_1}$ if this entry is not yet defined.
- $\mathrm{H}_2(ID)$: We use Coron's technique [Cor00] when simulating H_2 to obtain a tighter security bound. If $T_2[ID] = (b, x, X)$ then A returns X. If this entry is not yet defined, it chooses $x \stackrel{\$}{\leftarrow} \mathbb{Z}_N^*$ and tosses a biased coin b so that $b = 0$ with probability δ and $b = 1$ with probability $1 - \delta$. If $b = 0$, then A sets $X \leftarrow x^e \bmod N$; if $b = 1$, it sets $X \leftarrow x^e y \bmod N$. It stores $T_2[ID] \leftarrow (b, x, X)$ and returns X to F.
- Key derivation query for ID: Algorithm A looks up $T_2[ID] = (b, x, X)$, performing an additional query $\mathrm{H}_2(ID)$ if this entry is not yet defined. If $b = 0$, then A returns x; otherwise, it sets $bad_0 \leftarrow \mathtt{true}$ and aborts the execution of F returning $(0, \varepsilon)$.
- Signature query for identity ID_1, multiset of cosigners $L = \{ID_1, \ldots, ID_n\}$ and message m: Algorithm A first performs a query $\mathrm{H}_2(ID_1)$ and looks up $T_2[ID_1] = (b_1, x_1, X_1)$. If $b_1 = 0$, then A simulates signer ID_1 following the real $\mathsf{Sign}(x_1, L, m)$ algorithm using x_1 as a secret key. If $b_1 = 1$, it simulates the signing protocol as follows.

  It first chooses $t_1 \stackrel{\$}{\leftarrow} \{0,1\}^{l_0}$ and sends t_1 to all other cosigners. After having received $t_2, \ldots, t_n$ from all other cosigners (whose role is played by F), it chooses $c \stackrel{\$}{\leftarrow} \{0,1\}^{l_1}$, $s_1 \stackrel{\$}{\leftarrow} \mathbb{Z}_N^*$ and computes $R_1 \leftarrow s_1^e X_1^{-c} \bmod N$. If $T_0[R_1]$ has already been defined, then A sets $bad_1 \leftarrow \mathtt{true}$

and halts returning $(0, \varepsilon)$; otherwise, it sets $T_0[R_1] \leftarrow t_1$. For all $2 \leq i \leq n$, A looks up values R_i such that $T_0[R_i] = t_i$. If for some i multiple such values are found, A sets $bad_2 \leftarrow \texttt{true}$ and halts returning $(0, \varepsilon)$. If for some i no such value was found then it sets $alert \leftarrow \texttt{true}$; otherwise, it computes $R \leftarrow \prod_{i=1}^{n} R_i \bmod N$ and sets $T_1[R\|\langle L\rangle\|m] \leftarrow c$, or sets $bad_3 \leftarrow \texttt{true}$ and halts with output $(0, \varepsilon)$ if this entry was already defined. It sends R_1 to all other cosigners.

After having received $R'_2, \ldots, R'_n$ from the cosigners, A verifies that $\mathrm{H}_0(R'_i) = t_i$ for all $2 \leq i \leq n$. If not, it ends this signing protocol with local output $\perp$. If $R_i \neq R'_i$ for some i, then A sets $bad_2 \leftarrow \texttt{true}$ and halts with output $(0, \varepsilon)$. If $alert = \texttt{true}$ then it sets $bad_4 \leftarrow \texttt{true}$ and halts with output $(0, \varepsilon)$. Otherwise, it sends s_i to all cosigners.

After having received $s_2, \ldots, s_n$ from the cosigners, A computes $s \leftarrow \prod_{i=1}^{n} s_i \bmod N$ and returns the signature (c, s) to F.

Eventually, F outputs a forged signature (c, s) together with multiset of identities $L = \{ID_1, \ldots, ID_n\}$ and message m. Algorithm A computes performs additional random oracle queries $\mathrm{H}_2(ID_i)$ for $1 \leq i \leq n$, computes $R \leftarrow s^e \prod_{i=1}^{n} \mathrm{H}_2(ID_i)^{-c}$ and performs another random oracle query $\mathrm{H}_1(R\|\langle L\rangle\|m)$.

Let $U \subseteq \{ID_1, \ldots, ID_n\}$ be the uncorrupted identities in L, meaning those for which F never submitted a key derivation query. If the forgery is invalid, meaning that $\mathsf{Vf}(mpk, L, m, (R, s)) = 0$, $U = \emptyset$, or F previously made a signing query (ID, L, m), then A returns $(0, \varepsilon)$. Otherwise, algorithm A looks up $T_2[ID_i] = (b_i, x_i, X_i)$ for $1 \leq i \leq n$. Let $L_0 = \{ID_i : b_i = 0\}$ and $L_1 = \{ID_i : b_i = 1\}$. Since the forgery is valid, we have that

$$s^e \;\equiv\; R \cdot \prod_{i=1}^{n} X_i^c \;\equiv\; R \cdot \prod_{i=1}^{n} x_i^{ec} \cdot \prod_{i \in L_1} y^c \bmod N \;.$$

Let J be the index such that $h_J = c = T_1[R\|\langle L\rangle\|m]$. If $L_1 = \emptyset$ then A sets $bad_0 \leftarrow \texttt{true}$ and halts with output $(0, \varepsilon)$. Otherwise, it lets $x \leftarrow \prod_{i=1}^{n} x_i$, $n_1 \leftarrow |L_1|$, and halts with output $(J, (x, c, s, n_1))$.

We want to lower-bound the probability that A produces a "useful" output, i.e. an output other than $(0, \varepsilon)$. This is exactly the accepting probability acc as defined in Lemma 3.1 with respect to $H = \{0,1\}^{l_1}$ and an input generator IG that returns triples (N, e, y) such that $(N, e, d) \stackrel{\$}{\leftarrow} \mathsf{Kg}_{\mathrm{rsa}}$ and $y \stackrel{\$}{\leftarrow} \mathbb{Z}_N^*$. We overload our notation to let bad_i denote the event that the flag bad_i gets set to $\texttt{true}$ during the execution of A. We can lower-bound the accepting probability of A probability by:

$$\mathrm{acc} \;\geq\; \epsilon \cdot \Pr[\neg bad_0] - \Pr[bad_1] - \Pr[bad_2] - \Pr[bad_3] - \Pr[bad_4] \;. \tag{7}$$

First, let's take look at the factor $\Pr[\neg bad_0]$. The flag bad_0 gets raised whenever F makes a key derivation query for an identity for which $b = 1$, and if the final forgery does not contain any identities for which $b = 1$. Since the set L in the forgery must contain at least one uncorrupted identity, we have that $\Pr[\neg bad_0] \geq \delta^{q_{\mathrm{K}}}(1 - \delta)$. This function reaches a maximum for $\delta = q_{\mathrm{K}}/(q_{\mathrm{K}} + 1)$; filling in this value of δ in the above expression gives

$$\Pr[\neg bad_0] \;\geq\; \left(\frac{q_{\mathrm{K}}}{q_{\mathrm{K}} + 1}\right)^{q_{\mathrm{K}}} \cdot \frac{1}{q_{\mathrm{K}} + 1} \;=\; \frac{1}{q_{\mathrm{K}}} \cdot \left(1 - \frac{1}{q_{\mathrm{K}} + 1}\right)^{q_{\mathrm{K}} + 1}$$

from which we can conclude that

$$\Pr[\neg bad_0] \;\geq\; \frac{1}{4 q_{\mathrm{K}}} \;, \tag{8}$$

because $\Pr[\neg bad_0] = 1$ if $q_{\mathrm{K}} = 0$, because $\Pr[\neg bad_0] \geq 1/(4q_{\mathrm{K}})$ for $q_{\mathrm{K}} = 1$, and because $(1 - 1/(q_{\mathrm{K}} + 1))^{q_{\mathrm{K}}+1}$ is a monotonically increasing sequence for $q_{\mathrm{K}} \geq 1$.

The flag bad_1 gets raised during one of the q_S signature queries when $T_0[\cdot]$ is defined for an argument that is uniformly distributed over $\mathbb{Z}_N^*$ and that is independent from F's view. Since at any moment there are at most $q_\mathrm{H} + n_\mathrm{max} q_\mathrm{S}$ entries defined in table T_0, the probability that this happens is at most

$$\Pr[\, bad_1 \,] \;\leq\; \frac{q_\mathrm{S} \cdot (q_\mathrm{H} + n_\mathrm{max} q_\mathrm{S})}{2^{l_N}} \;. \tag{9}$$

The flag bad_2 only gets raised when two different entries in T_0 have the same value assigned to them. Since T_0 contains at most $q_\mathrm{H} + n_\mathrm{max} q_\mathrm{S}$ values that are all chosen uniformly at random from $\{0,1\}^{l_0}$ this happens with probability at most

$$\Pr[\, bad_2 \,] \;\leq\; \frac{(q_\mathrm{H} + n_\mathrm{max} q_\mathrm{S})^2}{2^{l_N+1}} \;. \tag{10}$$

To bound the probability that bad_3 is raised during the i-th signing query, we distinguish between the case that F "knows" R_1, meaning that it either queried $\mathrm{H}_0(R_1)$ directly, or saw R_1 as the honest signer's randomness in a previous signature query, and the case that it doesn't "know" R_1. In the latter case, F's view is independent of R, so the probability that this happens is simply given by the number of defined entries in T_1, which is at most $q_\mathrm{H} + q_\mathrm{S}$, divided by 2^{l_N}. In the former case, we cannot say that F's view is independent of R, so F may have queried $\mathrm{H}_1(R, \langle L \rangle, m)$ on purpose. Suppose F previously made a query $\mathrm{H}_0(R_1)$. Until right before this query, F's view was independent of R_1, so it had probability at most $q_\mathrm{H}/2^{l_N}$ to guess it correctly during any of its q_H queries. Likewise, the probability that A previously used the same randomness R_1 in a signature simulation is at most $q_\mathrm{S}/2^{l_N}$. In total, we have that

$$\Pr[\, bad_3 \,] \;\leq\; q_\mathrm{S} \cdot \left(\frac{q_\mathrm{H} + q_\mathrm{S}}{2^{l_N}} + \frac{q_\mathrm{H}}{2^{l_N}} + \frac{q_\mathrm{S}}{2^{l_N}} \right) \;=\; \frac{2 q_\mathrm{S} (q_\mathrm{H} + q_\mathrm{S})}{2^{l_N}} \;. \tag{11}$$

Lastly, the probability that bad_4 gets set is bounded by the probability that F managed to "predict" the value of $\mathrm{H}_0(R_i)$ during one of the q_S signature protocols and for one of the at most n_max signers, which is

$$\Pr[\, bad_4 \,] \;\leq\; \frac{n_\mathrm{max} q_\mathrm{S}}{2^{l_0}} \;. \tag{12}$$

Combining Equations (7–12) and using $n_\mathrm{max} > 0$ gives

$$\begin{aligned}
\mathrm{acc} \;&\geq\; \frac{\epsilon}{4 q_\mathrm{K}} - \frac{q_\mathrm{S}(q_\mathrm{H} + n_\mathrm{max} q_\mathrm{S})}{2^{l_N}} - \frac{(q_\mathrm{H} + n_\mathrm{max} q_\mathrm{S})^2}{2^{l_N+1}} - \frac{2 q_\mathrm{S}(q_\mathrm{H} + q_\mathrm{S})}{2^{l_N}} - \frac{n_\mathrm{max} q_\mathrm{S}}{2^{l_0}} \\
&\geq\; \frac{\epsilon}{4 q_\mathrm{K}} - \frac{3 q_\mathrm{S}(q_\mathrm{H} + n_\mathrm{max} q_\mathrm{S})}{2^{l_N}} - \frac{q_\mathrm{H}^2 + 2 n_\mathrm{max} q_\mathrm{S} q_\mathrm{H} + n_\mathrm{max}^2 q_\mathrm{S}^2}{2^{l_N+1}} - \frac{n_\mathrm{max} q_\mathrm{S}}{2^{l_0}} \\
&\geq\; \frac{\epsilon}{4 q_\mathrm{K}} - \frac{q_\mathrm{H}^2 + 4 n_\mathrm{max} q_\mathrm{S} q_\mathrm{H} + 4 n_\mathrm{max}^2 q_\mathrm{S}^2}{2^{l_N}} - \frac{n_\mathrm{max} q_\mathrm{S}}{2^{l_0}} \;.
\end{aligned} \tag{13}$$

Now consider an algorithm B that on input (N, e, y) runs the forking algorithm $\mathsf{F}_\mathsf{A}((N, e, y))$, which with probability frk returns a tuple $(1, (x, c, s, n_1), (x', c', s', n_1'))$ with $c \neq c'$. Since these originate from valid forgeries, their values are such that

$$s^e \equiv R x^{ec} y^{c n_1} \bmod N \quad \text{ and } \quad s'^e \equiv R' x'^{ec'} y^{c' n_1'} \bmod N \;.$$

The two executions of A when run by F_A are identical up to the "crucial" random oracle queries $\mathrm{H}_1(R \| \langle L \rangle \| m)$ and $\mathrm{H}_1(R' \| \langle L' \rangle \| m')$, where R, L, m and R', L', m' are the randomness, identity sets and messages that F used in its first and second forgeries, respectively. By the construction of A, we know that the two executions of F are identical up to this query (because it was provided with the exact same input, random tape and oracle responses), so in particular we have that $R = R'$, $L = L'$ and $m = m'$. Since the entries $T_2[ID_i] = (b_i, x_i, X_i)$ for $ID_i \in L = L'$ are chosen by A at the latest at the time of the crucial hash query, we also have that $x = x'$ and $n_1 = n_1'$. Dividing and reorganizing the two equations above gives

$$(x^{c-c'} s/s')^e \equiv y^{(c-c')n_1} \bmod N .$$

Since $c \neq c' \in \{0,1\}^{l_1}$, $n_1 \leq n_{\max}$, and e is a prime of length strictly greater than $l_1 + \log_2(n_{\max})$, we have that $e > (c-c')n_1$ and therefore that $\gcd(e, (c-c')n_1) = 1$. Using the extended Euclidean algorithm, one can find $a, b \in \mathbb{Z}$ such that $ae + b(c-c')n_1 = 1$. We then have that

$$y \equiv y^{ae+b(c-c')n_1} \equiv \left(y^a \cdot (x^{c-c'} s/s')^b\right)^e \bmod N .$$

Algorithm B can therefore output $y^a \cdot (x^{c-c'} s/s')^b$ as the RSA inversion of y. The probability that algorithm B succeeds in doing so is given by

$$\begin{aligned} \epsilon' &\geq \text{frk} \\ &\geq \frac{\text{acc}^2}{q_{\rm H}+1} - \frac{1}{2^{l_1}} \\ &\geq \frac{\epsilon^2}{16 q_{\rm K}^2 (q_{\rm H}+1)} - 2 \cdot \left(\frac{q_{\rm H}^2 + 4 n_{\max} q_{\rm S} q_{\rm H} + 4 n_{\max}^2 q_{\rm S}^2}{2^{l_N}} - \frac{n_{\max} q_{\rm S}}{2^{l_0}} \right) - \frac{1}{2^{l_1}} \end{aligned}$$

where in the last step we use Equation (13) and the facts that $(a-b)^2 \geq a^2 - 2ab$ and that $0 \leq \epsilon/4q_{\rm K} \leq 1$. The theorem follows.

We have left to show the bound for the running time t' of B. We permit ourselves to assume that (multi-)exponentiations in $\mathbb{Z}_N^*$ take time $t_{\exp}$ while all other operations take zero time. The running time of B is twice that of A, plus one multi-exponentiation mod N. The running time of A is that of the forger F plus one at most $n_{\max} + 1$ multi-exponentiations plus the time needed to answer F's oracle queries. Each random oracle or key derivation query takes at most one exponentiation. A signature simulation takes at most $n_{\max} + 1$ exponentiations. We therefore have that $t' = 2t + 2(n_{\max} + 2 + q_{\rm H} + q_{\rm K} + q_{\rm S}(n_{\max}+1)) \cdot t_{\exp}$. ▌

Acknowledgments

Mihir Bellare was supported by NSF grant CNS-0524765 and a gift from Intel Corporation. Gregory Neven is a Postdoctoral Fellow of the Flemish Research Foundation (FWO – Vlaanderen), and was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and in part by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

References

[AFKM05] Carlisle Adams, Stephen Farrell, Tomi Kause, and Tero Mononen. Internet X.509 public key infrastructure certificate management protocol (CMP). Internet Engineering Task Force RFC 4210, 2005. (Cited on page 2.)

[AR00] Michel Abdalla and Leonid Reyzin. A new forward-secure digital signature scheme. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 116–129, Kyoto, Japan, December 3–7, 2000. Springer-Verlag, Berlin, Germany. (Cited on page 10.)

[BA03] Kenneth Barr and Krste Asanovic. Energy aware lossless data compression. In *Proceedings of the First International Conference on Mobile Systems, Applications, and Services (MobiSys 2003)*, pages 231–244. ACM Press, 2003. (Cited on pages 1 and 6.)

[BG92] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 390–420, Santa Barbara, CA, USA, August 16–20, 1992. Springer-Verlag, Berlin, Germany. (Cited on page 2.)

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432, Warsaw, Poland, May 4–8, 2003. Springer-Verlag, Berlin, Germany. (Cited on pages 3, 5, 7, 8, 15, 24 and 37.)

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *ACM CCS 06: 13th Conference on Computer and Communications Security*, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press. (Cited on pages i, 7 and 8.)

[BN07] Mihir Bellare and Gregory Neven. Identity-based multi-signatures from RSA. In Masayuki Abe, editor, *Topics in Cryptology – CT-RSA 2007*, volume 4377 of *Lecture Notes in Computer Science*, pages ?–?, San Francisco, CA, USA, February 5–9, 2007. Springer-Verlag, Berlin, Germany. (Cited on page i.)

[BNN04] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 268–286, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany. (Cited on pages 3, 7, 24 and 27.)

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46, Miami, USA, January 6–8, 2003. Springer-Verlag, Berlin, Germany. (Cited on pages 1, 2, 3, 4, 8, 15, 17 and 37.)

[BP02] Mihir Bellare and Adriana Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 162–177, Santa Barbara, CA, USA, August 18–22, 2002. Springer-Verlag, Berlin, Germany. (Cited on page 5.)

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press. (Cited on pages 2, 8 and 9.)

[BRY06] Mihir Bellare, Thomas Ristenpart, and Scott Yilek. Work in progress. 2006. (Cited on page 2.)

[BSL01] Dan Boneh, Hovav Shacham, and Ben Lynn. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532, Gold Coast, Australia, December 9–13, 2001.

Springer-Verlag, Berlin, Germany. (Cited on pages 4 and 5.)

[CC03] Jae Choon Cha and Jung Hee Cheon. An identity-based signature from gap Diffie-Hellman groups. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30, Miami, USA, January 6–8, 2003. Springer-Verlag, Berlin, Germany. (Cited on page 27.)

[CJKT06] Claude Castelluccia, Stanislaw Jarecki, Jihye Kim, and Gene Tsudik. Secure acknowledgment aggregation and multisignatures with limited robustness. *Computer Networks*, 50(10):1639–1652, 2006. (Cited on page 7.)

[CLW05] Xiangguo Cheng, Jingmei Liu, and Xinmei Wang. Identity-based aggregate and verifiably encrypted signatures from bilinear pairing. In Osvaldo Gervasi, Marina L. Gavrilova, Vipin Kumar, Antonio Laganà, Heow Pueh Lee, Youngsong Mun, David Taniar, and Chih Jeng Kenneth Tan, editors, *Computational Science and Its Applications ICCSA 2005*, volume 3483 of *Lecture Notes in Computer Science*, pages 1046–1054. Springer-Verlag, Berlin, Germany, 2005. (Cited on pages 8 and 24.)

[Cor00] Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235, Santa Barbara, CA, USA, August 20–24, 2000. Springer-Verlag, Berlin, Germany. (Cited on page 27.)

[CV90] David Chaum and Hans Van Antwerpen. Undeniable signatures. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216, Santa Barbara, CA, USA, August 20–24, 1990. Springer-Verlag, Berlin, Germany. (Cited on page 21.)

[DKXY03] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Strong key-insulated signature schemes. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 130–144, Miami, USA, January 6–8, 2003. Springer-Verlag, Berlin, Germany. (Cited on pages 3, 7 and 24.)

[DP92] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction. In *33rd Annual Symposium on Foundations of Computer Science*, pages 427–436, Pittsburgh, Pennsylvania, October 24–27, 1992. IEEE Computer Society Press. (Cited on page 2.)

[FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. Zero knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988. (Cited on page 27.)

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 152–168, Santa Barbara, CA, USA, August 14–18, 2005. Springer-Verlag, Berlin, Germany. (Cited on page 2.)

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer-Verlag, Berlin, Germany. (Cited on pages 5 and 27.)

[GHK06] David Galindo, Javier Herranz, and Eike Kiltz. On the generic construction of identity-based signatures with additional properties. In Xuejia Lai, editor, *Advances in Cryptology – ASIACRYPT 2006*, Lecture Notes in Computer Science, Shanghai, China, December 3–7, 2006. Springer-Verlag, Berlin, Germany. (Cited on pages 3 and 7.)

[GJ03] Eu-Jin Goh and Stanislaw Jarecki. A signature scheme as secure as the Diffie-Hellman problem. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 401–415, Warsaw, Poland, May 4–8, 2003. Springer-Verlag, Berlin, Germany. (Cited on page 21.)

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988. (Cited on page 9.)

[GPS06] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. `http://eprint.iacr.org/`, 2006. (Cited on pages 3, 4 and 7.)

[GQ90] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231, Santa Barbara, CA, USA, August 21–25, 1990. Springer-Verlag, Berlin, Germany. (Cited on pages 7, 24 and 26.)

[GR06] Craig Gentry and Zulfikar Ramzan. Identity-based aggregate signatures. In Moti Yung, editor, *PKC 2006: 9th International Workshop on Theory and Practice in Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 257–273, New York, NY, USA, April 24–26, 2006. Springer-Verlag, Berlin, Germany. (Cited on pages 3, 4, 7, 8, 24 and 25.)

[Har94] Lein Harn. Group-oriented (t, n) threshold digital signature scheme and digital multisignature. *IEE Proceedings – Computers and Digital Techniques*, 141(5):307–313, September 1994. (Cited on page 1.)

[Hes03] Florian Hess. Efficient identity based signature schemes based on pairings. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002: 9th Annual International Workshop on Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 310–324, St. John's, Newfoundland, Canada, August 15–16, 2003. Springer-Verlag, Berlin, Germany. (Cited on page 27.)

[HMP95] Patrick Horster, Markus Michels, and Holger Petersen. Meta-multisignatures schemes based on the discrete logarithm problem. In *IFIP TC11 Eleventh International Conference on Information Security (IFIP/SEC 1995)*, pages 128–141. Chapman & Hall, 1995. (Cited on pages 1 and 16.)

[HOT04] Ryotaro Hayashi, Tatsuaki Okamoto, and Keisuke Tanaka. An RSA family of trapdoor permutations with a common domain and its applications. In Feng Bao, Robert Deng, and Jianying Zhou, editors, *PKC 2004: 7th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2947 of *Lecture Notes in Computer Science*, pages 291–304, Singapore, March 1–4, 2004. Springer-Verlag, Berlin, Germany. (Cited on page 6.)

[HS03] Javier Herranz and Germán Sáez. Forking lemmas for ring signature schemes. In Thomas Johansson and Subhamoy Maitra, editors, *Progress in Cryptology – INDOCRYPT 2003*, volume 2904 of *Lecture Notes in Computer Science*, pages 266–279. Springer-Verlag, Berlin, Germany, 2003. (Cited on page 5.)

[IN83] K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, 71:1–8, 1983. (Cited on pages 1 and 6.)

[Jou00] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. In *Algorithmic Number Theory Symposium – ANTS IV*, volume 1838 of *Lecture Notes in Computer Science*, pages 385–394. Springer-Verlag, Berlin, Germany, 2000. (Cited on pages 3, 7 and 24.)

[KW03] Jonathan Katz and Nán Wáng. Efficiency improvements for signature schemes with tight security reductions. In *ACM CCS 03: 10th Conference on Computer and Communications Security*, pages 155–164, Washington D.C., USA, October 27–30, 2003. ACM Press.

(Cited on pages 3 and 21.)

[Lan96] Susan K. Langford. Weakness in some threshold cryptosystems. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 74–82. Springer-Verlag, Berlin, Germany, 1996. (Cited on pages 1 and 16.)

[LHL95] Chuan-Ming Li, Tzonelih Hwang, and Narn-Yih Lee. Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 194–204. Springer-Verlag, Berlin, Germany, 1995. (Cited on page 1.)

[LMRS04] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 74–90, Interlaken, Switzerland, May 2–6, 2004. Springer-Verlag, Berlin, Germany. (Cited on pages 6, 8 and 37.)

[LOS$^+$06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, St. Petersburg, Russia, May 29 –June 1, 2006. Springer-Verlag, Berlin, Germany. Available as Cryptology ePrint Report 2006/096. (Cited on pages 1, 2, 3, 4, 8, 15 and 17.)

[LWH01] Chih-Yin Lin, Tzong-Chen Wu, and Jing-Jang Hwang. ID-based structured mulitsignature schemes. In Bart De Decker, Frank Piessens, Jan Smits, and Els Van Herreweghen, editors, *Advances in Network and Distributed Systems Security, IFIP TC11 WG11.4 First Annual Working Conference on Network Security*, volume 206 of *IFIP Conference Proceedings*, pages 45–60. Kluwer, 2001. (Cited on page 37.)

[MH96] Markus Michels and Patrick Horster. On the risk of disruption in several multiparty signature schemes. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology – ASIACRYPT'96*, volume 1163 of *Lecture Notes in Computer Science*, pages 334–345. Springer-Verlag, Berlin, Germany, 1996. (Cited on pages 1 and 16.)

[Mit01] Chris J. Mitchell. An attack on an ID-based multisignature scheme. Technical Report RHUL-MA-2001-9, Royal Holloway University of London, 2001. (Cited on page 37.)

[MOR01] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures. In *ACM CCS 01: 8th Conference on Computer and Communications Security*, pages 245–254, Philadelphia, PA, USA, November 5–8, 2001. ACM Press. (Cited on pages 1, 3, 4, 7, 8, 15, 16, 17, 20 and 37.)

[OO90] Kazuo Ohta and Tatsuaki Okamoto. A modification of the Fiat-Shamir scheme. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 232–243, Santa Barbara, CA, USA, August 21–25, 1990. Springer-Verlag, Berlin, Germany. (Cited on page 27.)

[OO91] Kazuo Ohta and Tatsuaki Okamoto. A digital multisignature scheme based on the Fiat-Shamir scheme. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *Advances in Cryptology – ASIACRYPT'91*, volume 739 of *Lecture Notes in Computer Science*, pages 139–148. Springer-Verlag, Berlin, Germany, 1991. (Cited on page 1.)

[OO99] Kazuo Ohta and Tatsuaki Okamoto. Multi-signature schemes secure against active insider attacks. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, E82-A(1):21–31, 1999. (Cited on page 1.)

[OS90] H. Ong and Claus-Peter Schnorr. Fast signature generation with a Fiat Shamir–like scheme. In Ivan Damgård, editor, *Advances in Cryptology – EUROCRYPT'90*, volume

473 of *Lecture Notes in Computer Science*, pages 432–440, Aarhus, Denmark, May 21–24, 1990. Springer-Verlag, Berlin, Germany. (Cited on page 27.)

[PKC00] PKCS #10: Certification request syntax standard. RSA Data Security, Inc., 2000. (Cited on page 2.)

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000. (Cited on pages 4, 5, 7, 9 and 12.)

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smartcards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252, Santa Barbara, CA, USA, August 20–24, 1990. Springer-Verlag, Berlin, Germany. (Cited on page 12.)

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991. (Cited on page 2.)

[Sch05] Jim Schaad. Internet X.509 public key infrastructure certificate request message format. Internet Engineering Task Force RFC 4211, 2005. (Cited on page 2.)

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53, Santa Barbara, CA, USA, August 19–23, 1985. Springer-Verlag, Berlin, Germany. (Cited on pages 3, 6, 24 and 27.)

[WH02] Tzong-Sun Wu and Chien-Lung Hsu. ID-based multisignatures with distinguished signing authorities for sequential and broadcasting architectures. *Applied Mathematics and Computation*, 131(2-3):349–356, 2002. (Cited on page 37.)

[Yi03] Xun Yi. An identity-based signature scheme from the Weil pairing. *IEEE Communications Letters*, 7(2):76–78, February 2003. (Cited on page 27.)

[YLL05] W.C. Yang, C.S. Laih, and C.C. Lin. Known signature attack on Wu-Hsu's ID-based multi-signature schemes. Manuscript available from `http://crypto.ee.ncku.edu.tw/~wcyang/`, 2005. (Cited on page 37.)

A Expected-Time Forking Lemma

Lemma A.1 [Simple Expected-Time Forking Lemma] Let $q \geq 1$ be an integer, H a set of size $h \geq 2$, and A a randomized algorithm that on inputs $x, h_1, \ldots, h_q$ returns an integer in the range $0, \ldots, q$ after at most $\mathbf{T}_{\mathsf{A}}(|x|)$ time steps. Let B be a *brute-force algorithm* that on input x returns ε after at most $\mathbf{T}_{\mathsf{B}}(|x|)$ time steps. The *accepting probability* $\mathrm{acc}(\mathsf{A}, x)$ is defined to be the probability that the following experiment returns 1:

```
Pick coins $\rho$ for $\mathsf{A}$ at random
$h_1, \ldots, h_q \xleftarrow{\$} H$ ; $i \leftarrow \mathsf{A}(x, h_1, \ldots, h_q; \rho)$
If $i \geq 1$ then return 1 else return 0.
```

Then the *forking algorithm* $\mathsf{F}_{\mathsf{A}}(x)$:

```
Pick coins $\rho$ for $\mathsf{A}$ at random
$h_1, \ldots, h_q \xleftarrow{\$} H$ ; $i \leftarrow \mathsf{A}(x, h_1, \ldots, h_q; \rho)$
If $i = 0$ then return 0
Repeat, in parallel with $\mathsf{B}(x)$,
    $h'_i, \ldots, h'_q \xleftarrow{\$} H$
    $j \leftarrow \mathsf{A}(x, h_1, \ldots, h_{i-1}, h'_i, \ldots, h'_q; \rho)$
Until ($i = j$ and $h_i \neq h'_i$) or $\mathsf{B}$ has halted
Return 1
```

returns 1 with probability $\mathrm{acc}(\mathsf{A}, x)$ in expected time

$$\mathbf{T}_{\mathsf{F}_\mathsf{A}}(|x|) \;\leq\; (4q+1)\mathbf{T}_\mathsf{A}(|x|) + \frac{4q}{h} \cdot \mathbf{T}_\mathsf{B}(|x|) \ . \tag{14}$$

Proof of Lemma A.1: It is clear that the forking algorithm will eventually return 1 whenever $i > 0$ after the first run of A, which happens with probability $\mathrm{acc}(\mathsf{A}, x)$.

Let X_i be defined as in the proof of Lemma 3.1. Assuming that processor time is divided equally between the two algorithms running in parallel, the expected running time of the forking algorithm F_A on input x is

$$\begin{aligned}
\mathbf{T}_{\mathsf{F}_\mathsf{A}}(|x|) &= \mathbf{T}_\mathsf{A}(|x|) + \mathbf{E}\left[\sum_{i=1}^{q} X_i \cdot \min\Big(2\mathbf{T}_\mathsf{A}(|x|) \cdot \Pr\big[\, i = j \,\wedge\, h_i \neq h'_i \,\big]^{-1},\, 2\mathbf{T}_\mathsf{B}(|x|)\Big)\right] \\
&\leq \mathbf{T}_\mathsf{A}(|x|) + \mathbf{E}\left[\sum_{i=1}^{q} \min\Big(2X_i \cdot \mathbf{T}_\mathsf{A}(|x|) \cdot \Pr\big[\, i = j \,\wedge\, h_i \neq h'_i \,\big]^{-1},\, 2X_i \cdot \mathbf{T}_\mathsf{B}(|x|)\Big)\right] \\
&\leq \mathbf{T}_\mathsf{A}(|x|) + \sum_{i=1}^{q} \Bigg(2\mathbf{T}_\mathsf{A}(|x|) \cdot \mathbf{E}\left[X_i \cdot \Pr\big[\, i = j \,\wedge\, h_i \neq h'_i \,\big]^{-1} \;\middle|\; X_i \geq \frac{2}{h}\right] \cdot \Pr\left[X_i \geq \frac{2}{h}\right] \\
&\qquad + 2\mathbf{T}_\mathsf{B}(|x|) \cdot \mathbf{E}\left[X_i \;\middle|\; X_i < \frac{2}{h}\right] \cdot \Pr\left[X_i < \frac{2}{h}\right]\Bigg) \\
&\leq \mathbf{T}_\mathsf{A}(|x|) + \sum_{i=1}^{q} \left(2\mathbf{T}_\mathsf{A}(|x|) \cdot \mathbf{E}\left[\frac{X_i}{X_i - h^{-1}} \;\middle|\; X_i \geq \frac{2}{h}\right] \cdot + \frac{4}{h} \cdot \mathbf{T}_\mathsf{B}(|x|)\right) \\
&\leq \mathbf{T}_\mathsf{A}(|x|) + \sum_{i=1}^{q} \left(4\mathbf{T}_\mathsf{A}(|x|) + \frac{4}{h} \cdot \mathbf{T}_\mathsf{B}(|x|)\right) \\
&= (4q+1)\mathbf{T}_\mathsf{A}(|x|) + \frac{4q}{h} \cdot \mathbf{T}_\mathsf{B}(|x|) \ ,
\end{aligned}$$

yielding Equation (14) as required. The fourth line above is true because

$$\begin{aligned}
\Pr[\, i = j \,] &= X_i \\
&= \Pr\big[\, i = j \,\wedge\, h_i = h'_i \,\big] + \Pr\big[\, i = j \,\wedge\, h_i \neq h'_i \,\big] \\
&\leq \frac{1}{h} + \Pr\big[\, i = j \,\wedge\, h_i \neq h'_i \,\big]
\end{aligned}$$

and since the expectation is conditioned on $X_i \geq \frac{2}{h} > \frac{1}{h}$ we have that

$$\Pr\big[\, i = j \,\wedge\, h_i \neq h'_i \,\big]^{-1} \leq \frac{1}{X_i - h^{-1}} \ . \tag{15}$$

$\blacksquare$

B Generic Constructions

We clarify the relation between the different types aggregate signatures and multisignatures by presenting generic IAS constructions from all of these.

FROM GAS TO IAS. A GAS scheme allows anyone (so not just signers) to combine a possibly already aggregated signature σ_n authenticating messages $m_1, \ldots, m_n$ under public keys $pk_1, \ldots, pk_n$ with a signature σ_{n+1} on message m_{n+1} under public key pk_{n+1} into a new compact signature σ_{n+1} authenticating $m_1, \ldots, m_{n+1}$ under keys $pk_1, \ldots, pk_{n+1}$.

Any GAS scheme naturally gives rise to an IAS scheme in the following way. The signers first communicate to each other their public keys $pk_1, \ldots, pk_n$ and the messages $m_1, \ldots, m_n$ that they will be authenticating. Next, all signers create a signature under their own public key for message $m = \langle\{(pk_1, m_1), \ldots, (pk_n, m_n)\}\rangle$, and send this signature to all cosigners. Each signer computes the combined signature by aggregating his own signature with all received signatures. Alternatively, the signers could decide on one cosigner to do the aggregation and broadcast the result.

It is not hard to see that any GAS scheme secure under the notion of [BGLS03] yields a secure IAS scheme under our notion. Note that the construction where each signer signs his own message m_i rather than m is not secure, because then the partial signatures can be reused in other aggregations, which is not allowed under our notion. The only known implementation of GAS schemes is due to [BGLS03] and is based on pairings.

FROM SAS TO IAS. Similarly, any SAS scheme directly gives rise to an IAS scheme. Recall that any signer in a SAS scheme can use his secret key to add a signature on his own message m_{n+1} to a possibly already aggregated signature σ_n for messages $m_1, \ldots, m_n$ under public keys $pk_1, \ldots, pk_n$.

First, the signers decide on an order among them, and communicate to each other their messages $m_1, \ldots, m_n$ and public keys $pk_1, \ldots, pk_n$. The first signer creates a signature for message $m = \langle\{(pk_1, m_1), \ldots, (pk_n, m_n)\}\rangle$ using his own secret key, and passes this signature on to the second signer. The second signer uses his secret key to add his own signature for m onto the signature that he received, and sends the combined signature to the next signer in the chain. This process continues until the last signer adds his signature on m, and sends the resulting signature to all cosigners.

Again, it is not hard to see that any scheme secure under the notion of [LMRS04] yields a secure IAS scheme. The only practical SAS scheme known today is based on RSA [LMRS04], but suffers from a couple of drawbacks as mentioned in the introduction.

FROM IAS TO MULTISIGNATURES AND BACK. It is clear that signers in an IAS scheme can always choose to sign the same message m, so any IAS scheme is at the same time a multisignature scheme. Our results therefore contribute directly to the state of the art in multisignature schemes as well. In particular, our work gives rise to the first multisignature schemes free of assumptions on key generation, the first scheme with a tight security reduction, the first provably secure identity-based multisignature scheme (a number of proposed schemes [LWH01, WH02] were later found to be flawed [Mit01, YLL05]), and the first scheme with a security proof in the standard model.

Vice versa, a multisignature scheme can be converted into an IAS scheme by letting all signers sign $m = \langle\{(pk_1, m_1), \ldots, (pk_n, m_n)\}\rangle$. Disregarding the difference in assumptions on key setup, any multisignature scheme secure in the notion of [MOR01] yields a secure IAS scheme in our notion. The only provably secure multisignature schemes are due to [MOR01, Bol03] and are based on discrete logarithms and pairings, respectively.