Identity-Based Multi-signatures from RSA

Mihir Bellare[1] and Gregory Neven[2]

[1] Department of Computer Science & Engineering
University of California San Diego
mihir@cs.ucsd.edu
http://www.cs.ucsd.edu/users/mihir
[2] Department of Electrical Engineering
Katholieke Universiteit Leuven
and Département d'Informatique
Ecole Normale Supérieure
Gregory.Neven@esat.kuleuven.be
http://www.neven.org

Abstract. Multi-signatures allow multiple signers to jointly authenticate a message using a single compact signature. Many applications however require the public keys of the signers to be sent along with the signature, partly defeating the effect of the compact signature. 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. In this paper, we present and prove secure an identity-based multi-signature (IBMS) scheme based on RSA, which in particular does not rely on (the rather new and untested) assumptions related to bilinear maps. We define an appropriate security notion for interactive IBMS schemes and prove the security of our scheme under the one-wayness of RSA in the random oracle model.

1 Introduction

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 [1], so it makes perfect sense to invest extra computation cycles to save on bandwidth. Also, communication

M. Abe (Ed.): CT-RSA 2007, LNCS 4377, pp. 145–162, 2007.

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 [20] 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 [28], 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 [12,2], 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 [16]. 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 [6], 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 [17]. 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 [21], 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 [15]. 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 [18], strengthened with techniques from [3] to provide security against concurrent attacks. Unstrengthened variants of our scheme were proposed before (but without security proofs) in [18,8]. The proof makes use of the general forking lemma of [3], which, unlike the original forking lemma by Pointcheval and Stern [27], 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 [17]. 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 [17]. 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 [17], 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 [24,3]. 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 [3], but unlike [24], 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 [17] 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 [6] 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 [17]; it is also based on pairings. IBAS schemes automatically give rise to IBMS schemes, but the scheme resulting from the only known IBAS instantiation [17] 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. [10] 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 [10,17] 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 [24,5,22,3]. The schemes of [5,22] are based on pairings, those of [24,3] on discrete logarithms. In a sequential aggregate signature (SAS) scheme [23], 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 [23,22]. The scheme of [23] 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 [22] uses pairings, and is the only one with security in the standard (i.e., non-random-oracle [4]) model.

2 Identity-Based Multi-signatures

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, $s \xleftarrow{\$} S$ 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 $y \xleftarrow{\$} \mathsf{A}(x_1, \ldots)$ means that we choose ρ at random and let $y = \mathsf{A}(x_1, \ldots; \rho)$.

GENERAL SETTING. We adapt definitions from [24,3] to the identity-based setting. Consider n different signers with identities $ID_1, \ldots, ID_n$ who collectively want to sign the same message m so that the resulting signature σ authenticates m under each of their identities. We consider schemes with interactive signing algorithms, meaning that all signers are simultaneously online and interact to produce the signature σ. We assume that signers interact in rounds, where at the beginning of each round each signer receives an incoming message from each of the other signers, performs some computation and sends an outgoing message to all other signers. We let the incoming message of the first round be the local input of the signing algorithm, consisting of the secret key, the list of co-signers, and the message m. The outgoing message of the last round is the final signature σ, or $\perp$ to indicate failure.

We assume that each signer has a direct connection to each of its co-signers. We do not assume these connections to be private or authenticated however, and neither do we assume the availability of an atomic broadcast primitive. When describing signing protocols, we let each signer refer to itself as signer 1 with identity ID_1, and let it assign an index $2, \ldots, n$ to each of its cosigners with identities $ID_2, \ldots, ID_n$. These indices serve merely as a local pointer for the signer to distinguish between its different cosigners and the connections over which it communicates with them. They have no global meaning outside this protocol instance: the signer that you refer to as signer 2 with identity ID_2 may very well be my signer 3 with identity ID_3, and in a later protocol instance I may very well refer to the same signer as signer 4 with identity ID_4. The indices have no certified relationship with identities, and are certainly not included in the identity strings.

SYNTAX OF IBMS SCHEMES. Formally an identity-based multi-signature scheme $\mathsf{IBMS} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Sign}, \mathsf{Vf})$ consists of four algorithms. A trusted key distribution center runs the 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 the message m and a multiset $L = \{ID_1, \ldots, ID_n\}$ containing the identities of all signers participating in the protocol. After a number of interactions, the Sign algorithm outputs the multi-signature σ, or $\perp$ to indicate failure. The verification algorithm Vf takes as input the master public key mpk, a multiset of identities $L = \{ID_1, \ldots, ID_n\}$, the message m and a candidate signature σ, and outputs 1 if σ is a valid signature on message m by all identities in L, or outputs 0 otherwise. (Because a cheating signer may try to impersonate an identity for which he does not have the key, we explicitly allow multiple occurrences of the same identity by modeling L as a multiset.)

In the random oracle model [4], the key derivation, signing and verification algorithms additionally have access to a random oracle $\mathrm{H}(\cdot) : \{0,1\}^* \to D$, where D depends on the scheme and possibly on the master public key mpk. If the scheme uses multiple random oracles, these can be derived from a single oracle H using techniques of [4].

Correctness requires that, whenever all signers correctly follow the Sign protocol using secret key $sk_{ID_i} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID_i)$, then with probability one they all end up with local output a signature σ such that $\mathsf{Vf}(mpk, L, m, \sigma) = 1$ for all positive integers n, all (mpk, msk) output by Setup, all $ID_1, \ldots, ID_n \in \{0,1\}^*$ and all messages $m \in \{0,1\}^*$.

3 Two Security Notions and Their Equivalence

In standard (i.e., non-identity-based) multi-signature schemes, security is commonly defined through an experiment with a single honest "target" signer, effectively viewing all other signers as corrupted [24,5,3]. Security requires that it be infeasible to forge a multi-signature involving the target signer. The adversary has access to a signing oracle that plays the role of the target signer, while the adversary plays the role of all other signers participating in the protocol. The logic underlying this simplified model is that any honestly generated public key is "as good" as any other one; no adversary is expected to perform significantly better in a model with multiple honest signers, as it could easily have simulated these other signers itself.

The same logic does *not* go through for IBMS schemes however. Any identity is not necessarily "as good" as any other one: the scheme may behave differently on different identities, or may even have "weak" identities for which forging signatures is easy. We therefore need to consider a stronger security notion where the adversary can adaptively decide to corrupt signers by submitting their identities to a key derivation oracle, resulting in it being given their secret signing keys.

The adversary is also given access to a signing oracle through which it can engage in any number of arbitrarily interleaved signing protocols with honest signers. Unlike the case of standard multi-signature schemes however, it is not immediately clear whether it is sufficient to let the oracle in each protocol instance play the role of a single honest signer, leaving it up to the adversary to

play the role of all other signers, or whether we should allow the oracle to play the role of multiple honest signers simultaneously. Indeed, an adversary in the former model could try to simulate the signing oracle of the latter model by corrupting all but one of the honest signers, but this precludes attacking any of the corrupted identities in the final forgery. Our goal is of course to achieve the strongest security notion possible, but at the same time we want to avoid making security proofs unnecessarily complicated. Since the relation between the two notions is not immediately clear, we present both in full detail below, and then prove that both notions are in fact equivalent (up to a factor that is the product of the number of the adversary's signature queries and the maximal number of participating signers in one protocol).

Our security notions do not cover *robustness* [7], meaning that we do not prevent malicious signers or network faults from disrupting the signing protocol or from making honest signers end up with invalid signatures. As argued in [24], the strong notion of unforgeability in an adversarially-controlled network strengthens the security guarantees offered by our scheme, but prevents robustness. Indeed, if a signer with identity ID refuses to cooperate or is unreachable due to network faults, then it should be impossible for the other signers to compute any signature involving ID, for otherwise the scheme would be forgeable.

Single-signer security. In somewhat more detail, we consider the following three-phase game associated to multi-signature scheme $\mathsf{IBMS} = (\mathsf{Setup}, \mathsf{KeyDer}, \mathsf{Sign}, \mathsf{Vf})$ and adversary (forger) F:

Setup: The game generates a master key pair $(mpk, msk) \xleftarrow{\$} \mathsf{Setup}$, and gives the master public key mpk as input to the forger.

Attack: F can decide to corrupt a signer by querying a key derivation oracle with any identity ID, which returns the secret key for that identity $sk_{ID} \xleftarrow{\$} \mathsf{KeyDer}(msk, ID)$. In the random oracle model [4], it also has access to a random oracle $\mathrm{H}(\cdot)$. The forger can engage in an instance of the signature protocol with any honest signer ID that it chooses, while F itself plays the role of all other signers. It does so by submitting the identity $ID \in \{0,1\}^*$, a multiset of identities L and a message m to a signing oracle. The multiset L contains ID at least once. The oracle plays the role of ID as dictated by the Sign algorithm; the role of the other (possibly cheating) signers in L is played by F. Note that the identities in $L \setminus \{ID\}$ need not all be corrupted; the forger is free to try to simulate them without their secret key. The forger can schedule an arbitrary number of protocol instances concurrently, interacting with "clones" of the same honest signer, where each clone maintains its own state and uses its own coins. When the honest signer terminates a signing protocol, its local output (whether $\perp$ or a compact signature σ) is returned to F.

Forgery: At the end of its execution, F outputs a multiset $L = \{ID_1, \ldots, ID_n\}$ of identities, a message m and a forged signature σ. The forger is said to win the game if $\mathsf{Vf}(mpk, L, m, \sigma) = 1$, if L contains at least one uncorrupted identity, and if the forger never submitted a query (ID, L, m) to the signing oracle for any $ID \in L$.

The advantage of F in breaking the single-signer unforgeability of IBMS 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 signers, and the setup phase. We say that a forger F $(t, q_{\mathrm{K}}, q_{\mathrm{S}}, q_{\mathrm{H}}, n_{\max}, \epsilon)$-breaks the single-signer security of IBMS if it runs in time at most t; performs at most q_{K} key derivation queries and at most q_{H} random oracle queries; engages in at most q_{S} signature interactions with up to $n_{\max}$ signers; and has advantage at least ϵ. (If there is more than one random oracle, q_{H} denotes the sum of the number of queries to all random oracles.) We say that IBMS is $(t, q_{\mathrm{K}}, q_{\mathrm{S}}, q_{\mathrm{H}}, n_{\max}, \epsilon)$ single-signer secure in the random oracle model if no forger F $(t, q_{\mathrm{K}}, q_{\mathrm{S}}, q_{\mathrm{H}}, n_{\max}, \epsilon)$-breaks it.

Multi-signer security. Alternatively, we define the notion of *multi-signer security*. The game is similar to the one described above, except that the signing oracle in each protocol instance can play the role of multiple honest signers simultaneously. In particular, it performs signature queries 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 identities of honest signers, whose role will be played by the oracle as dictated by the Sign algorithm. The forger plays the role of the (possibly) cheating signers contained in L_{c}.

For the communication between signers, we consider the strongest possible notion: the adversary completely controls all network traffic, even between honest signers. We model this by letting honest signers, when they want to send a message to another signer, hand the message to the adversary for delivery. The adversary can then choose to inspect, modify and whether or not to deliver the message at will. We do not assume the availability of a trusted broadcast primitive, so the forger can cause different honest signers to have a different view of the protocol by providing them with different messages. Note that this is a situation that in particular cannot arise in the single-signer notion.

The forger F is said to win the game if eventually F outputs a forgery (L, m, σ) such that $\mathsf{Vf}(mpk, L, m, \sigma) = 1$, L contains at least one uncorrupted identity, and the forger never performed a signature query $(L_{\mathrm{h}}, L_{\mathrm{c}}, m)$ where $L_{\mathrm{h}} \cup L_{\mathrm{c}} = L$. The advantage of F in breaking the multi-signer security of IBMS is defined as the probability that it wins the above game. We say that a forger F $(t, q_{\mathrm{K}}, q_{\mathrm{S}}, q_{\mathrm{H}}, n_{\max}, \epsilon)$-breaks the multi-signer security of IBMS if has advantage at least ϵ and adheres to the resource restrictions as explained above. We say that IBMS is $(t, q_{\mathrm{K}}, q_{\mathrm{S}}, q_{\mathrm{H}}, n_{\max}, \epsilon)$ multi-signer secure if no forger F $(t, q_{\mathrm{K}}, q_{\mathrm{S}}, q_{\mathrm{H}}, n_{\max}, \epsilon)$-breaks it.

Single-signer implies multi-signer security. It is obvious that the notion of multi-signer security is at least as strong as that of single-signer security, since the latter can be viewed as a special case of the former where each of the sets L_{h} is restricted to be a singleton. The following theorem states the less obvious direction that single-signer security also implies multi-signer security, at the loss of a factor $n_{\max} q_{\mathrm{H}}$ in the reduction.

Theorem 1. *If the IBMS scheme* IBMS *is* $(t', q'_{\mathrm{K}}, q_{\mathrm{S}}, q_{\mathrm{H}}, n_{\max}, \epsilon')$ *single-signer secure, then it is also* $(t, q_{\mathrm{K}}, q_{\mathrm{S}}, q_{\mathrm{H}}, n_{\max}, \epsilon)$ *multi-signer secure for*

$\epsilon \geq n_{\max}(q_S + 1) \cdot \epsilon'$, $t \leq t' - n_{\max} q_S \cdot t_{\mathsf{Sign}}$ *and* $q_K \leq q'_K - n_{\max} q_S$, *where* t_{Sign} *is the running time of an execution of the signing protocol.*

Proof. Let F be a forging algorithm that $(t, q_K, q_S, q_H, n_{\max}, \epsilon)$-breaks the multi-signer security of IBMS. Consider the single-signer forging algorithm F' that, on input mpk, guesses indices $i \xleftarrow{\$} \{0, \ldots, q_S\}$ and $j \xleftarrow{\$} \{1, \ldots, n_{\max}\}$ and runs $\mathsf{F}(mpk)$. F' maintains a counter ctr, initially zero, and an identity string ID^*, initially $\perp$. It responds to F's signature queries (L_h, L_c, m) as follows. It first increases ctr; if $ctr = i$, it sets ID^* to the j-th identity in L_h, or aborts if L_h contains less than j elements. If F' already corrupted ID^* before, it also aborts. To simulate the signing protocol, F' uses its own signing oracle on input $(ID^*, L_h \cup L_c, m)$ to simulate ID^*, and corrupts all other identities in L_h so that it can correctly simulate them using their secret keys.

When F queries for the secret key of ID^*, F' gives up; otherwise, it forwards the response from its own key derivation oracle. Eventually, F outputs its forgery (L, m, σ). To be a valid forgery, L must contain at least one identity ID that F never corrupted. If $ID = ID^*$, then (L, m, σ) is also a valid forgery for F'. Likewise, if $i = 0$ and F never performed any signature queries involving ID, then (L, m, σ) is a valid forgery for F'. In all other cases, F' aborts.

It is easy to see that F' succeeds with probability $\epsilon' \geq \epsilon/(n_{\max}(q_S + 1))$, that its running time is at most that of F plus the time of $n_{\max} q_S$ signing protocols, and that it performs at most q_S signature queries, $q_K + n_{\max} q_S$ key derivation queries, and q_H random oracle queries. □

Viewing that both security notions are essentially equivalent and that the notion of single-signer security is much easier to work with, we will stick to the latter throughout the rest of the paper. When we talk about the advantage of a forger or the security of an IBMS scheme, we implicitly mean the advantage and security under the single-signer notion.

4 Our Scheme

In this section, we present an IBMS scheme based on the GQ identity-based signature scheme [18]. To give some intuition into our scheme, we briefly recall the GQ scheme here. The key distribution center generates an RSA modulus N and exponents e, d such that $ed \equiv 1 \bmod \varphi(N)$. The master public key is the pair (N, e), while d is the master secret key. The signature on a message m by identity ID is a pair (R, s) such that $s^e \equiv R \cdot \mathrm{H}_2(ID)^c \bmod N$ where $c = \mathrm{H}_1(R \| m)$ and $\mathrm{H}_1, \mathrm{H}_2$ are random oracles. As pointed out by [17], to make an aggregate variant of a randomized signature scheme, one must find a way to "aggregate the randomness" in the different signatures. In the case of GQ signatures, this can be achieved by multiplying elements together, so that a signature by identities $ID_1, \ldots, ID_n$ is a pair (R, s) such that

$$s^e \equiv R \cdot \prod_{i=1}^{n} \mathrm{H}_2(ID_i)^c \bmod N ,$$

where R and s are the product of the R_i and s_i generated by all signers, respectively, and $c = \mathrm{H}_1(R\|m)$. Note that the combined randomness R is needed in order to compute c, which is the reason why the scheme has an interactive signing protocol.

The basic multi-signature scheme is not new: it was already present in [18] and was recently strengthened to provide robustness in [8], but was never proved secure as such. If one were to attempt such a proof, it would be complicated by the fact that, just like for other signature scheme following the Fiat-Shamir paradigm [14], simulation of signatures relies on the unpredictability of the randomness R used in the signature. In particular, for the simulator to be able to program the random oracle H_1 to simulate signature queries, the adversary's view needs to be independent of R. One way to overcome this problem [24] is by guessing, for each of the q_S signature queries, the index of a random oracle query that contains the correct R, and rewinding the adversary if the guess is incorrect. To avoid an exponential blowup of the running time, one has to restrict the multi-signature scheme to forbid concurrent signing sessions. This may be a limiting restriction viewing the inherently concurrent execution setting on the Internet. Instead, we apply a recent technique of [3] to regain provable security for concurrent protocol executions. The trick consists of letting signers commit to their randomness share R_i through a random oracle in the first round of the protocol, so that the simulator, who sees all random oracle queries, can extract these values and correctly program the random oracle before the final value of R is known to the forger.

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 present our scheme in more detail. 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 multi-signature scheme IBMS-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.)

While largely based on a combination of existing schemes and techniques, we believe the above scheme is important viewing the practical attractiveness of the identity-based setting in combination with compact signatures. Our scheme points out that provably secure IBMS can be achieved without the use of pairings, which is an interesting observation in its own right. Pairings have only recently been introduced to cryptography, and may therefore be at a greater risk of novel security breaches than the better-tested assumptions relating to RSA. Moreover, hardware and software implementations of RSA are ubiquitous, even in the public domain, while good pairing implementations are much harder

to find. Many companies have invested in efficient and secure implementations of RSA, and may prefer to recycle these investments in their future products.

5 Security of Our Scheme

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 [24] because keys are generated by the trusted center instead of by the signers themselves. Also, unlike the scheme of [24], we do allow concurrent signing sessions by reusing techniques of [3].

Theorem 2. *If the RSA function associated to* $\mathsf{Kg}_{\mathrm{rsa}}$ *is* (t', ϵ')*-one-way, then the* IBMS-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*

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

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

To prove the above theorem, we use a variant of the Forking Lemma of Pointcheval and Stern [27] that was presented in [3]. Unlike the original Forking Lemma, this variant is easily applicable to settings other than standard signature schemes. We recall the lemma here.

Lemma 3. *Let* $q \geq 1$ *be an integer and* H *a set 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 by*

$$\mathrm{acc} \;=\; \Pr\left[\, J \geq 1 \;:\; x \xleftarrow{\$} \mathsf{IG}\,;\; h_1, \ldots, h_q \xleftarrow{\$} H\,;\; (J, \sigma) \xleftarrow{\$} \mathsf{A}(x, h_1, \ldots, h_q) \,\right] .$$

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

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

Let

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

Then

$$\mathrm{frk} \;\geq\; \mathrm{acc} \cdot \left(\frac{\mathrm{acc}}{q} - \frac{1}{h} \right) . \qquad (2)$$

We are now ready to prove the security of our IBMS scheme.

Proof (Theorem 2). 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 $T_0[R] \xleftarrow{\$} \{0,1\}^{l_0}$. 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 [11] 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 $x \xleftarrow{\$} \mathbb{Z}_N^*$ 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 $t_1 \xleftarrow{\$} \{0,1\}^{l_0}$ 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 $c \xleftarrow{\$} \{0,1\}^{l_1}$, $s_1 \xleftarrow{\$} \mathbb{Z}_N^*$ 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 $2 \le i \le n$, 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 $bad_2 \leftarrow \mathtt{true}$ and halts returning $(0, \varepsilon)$. If for some i no such value was found then it sets $alert \leftarrow \mathtt{true}$; 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 $bad_3 \leftarrow \mathtt{true}$ 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 $2 \le i \le n$. If not, it ends this signing protocol with local output $\perp$. If $R_i \neq R'_i$ for some i, then A sets $bad_2 \leftarrow \mathtt{true}$ and halts with output $(0, \varepsilon)$. If $alert = \mathtt{true}$ then it sets $bad_4 \leftarrow \mathtt{true}$ 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 $1 \le i \le n$, 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 $1 \le i \le n$. 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 $bad_0 \leftarrow \mathtt{true}$ 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 with respect to $H = \{0,1\}^{l_1}$ and an input generator IG that returns triples (N, e, y) such that $(N, e, d) \xleftarrow{\$} \mathsf{Kg}_{\mathrm{rsa}}$ and $y \xleftarrow{\$} \mathbb{Z}_N^*$. We overload our notation to let bad_i denote the event that the flag bad_i gets set to $\mathtt{true}$ during the execution of A. We can lower-bound the accepting probability of A probability by:

$$\mathrm{acc} \;\ge\; \epsilon \cdot \Pr[\,\neg bad_0\,] - \Pr[\,bad_1\,] - \Pr[\,bad_2\,] - \Pr[\,bad_3\,] - \Pr[\,bad_4\,] \;. \quad (3)$$

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 $\Pr[\,\neg bad_0\,] \ge \delta^{q_\mathrm{K}}(1-\delta)$. 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

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

from which we can conclude that

$$\Pr[\,\neg bad_0\,] \;\ge\; \frac{1}{4q_\mathrm{K}} \;, \quad (4)$$

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

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 $q_\mathrm{H} + n_{\max} q_\mathrm{S}$ entries defined in table T_0, the probability that this happens is at most

$$\Pr[\,bad_1\,] \;\le\; \frac{q_\mathrm{S} \cdot (q_\mathrm{H} + n_{\max} q_\mathrm{S})}{2^{l_N}} \;. \quad (5)$$

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 $q_\mathrm{H} + n_{\max} q_\mathrm{S}$ values that are all chosen uniformly at random from $\{0,1\}^{l_0}$ this happens with probability at most

$$\Pr[\, bad_2 \,] \;\leq\; \frac{(q_\mathrm{H} + n_{\max} q_\mathrm{S})^2}{2^{l_N+1}} \;. \tag{6}$$

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

$$\Pr[\, bad_3 \,] \;\leq\; q_\mathrm{S} \cdot \left(\frac{q_\mathrm{H} + q_\mathrm{S}}{2^{l_N}} + \frac{q_\mathrm{H}}{2^{l_N}} + \frac{q_\mathrm{S}}{2^{l_N}} \right) \;=\; \frac{2q_\mathrm{S}(q_\mathrm{H} + q_\mathrm{S})}{2^{l_N}} \;. \tag{7}$$

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

$$\Pr[\, bad_4 \,] \;\leq\; \frac{n_{\max} q_\mathrm{S}}{2^{l_0}} \;. \tag{8}$$

Combining Equations (3–8) and using $n_{\max} > 0$ gives

$$\begin{aligned}
\mathrm{acc} \;&\geq\; \frac{\epsilon}{4q_\mathrm{K}} - \frac{q_\mathrm{S}(q_\mathrm{H} + n_{\max} q_\mathrm{S})}{2^{l_N}} - \frac{(q_\mathrm{H} + n_{\max} q_\mathrm{S})^2}{2^{l_N+1}} - \frac{2q_\mathrm{S}(q_\mathrm{H} + q_\mathrm{S})}{2^{l_N}} - \frac{n_{\max} q_\mathrm{S}}{2^{l_0}} \\
&\geq\; \frac{\epsilon}{4q_\mathrm{K}} - \frac{3q_\mathrm{S}(q_\mathrm{H} + n_{\max} q_\mathrm{S})}{2^{l_N}} - \frac{q_\mathrm{H}^2 + 2n_{\max} q_\mathrm{S} q_\mathrm{H} + n_{\max}^2 q_\mathrm{S}^2}{2^{l_N+1}} - \frac{n_{\max} q_\mathrm{S}}{2^{l_0}} \\
&\geq\; \frac{\epsilon}{4q_\mathrm{K}} - \frac{q_\mathrm{H}^2 + 4n_{\max} q_\mathrm{S} q_\mathrm{H} + 4n_{\max}^2 q_\mathrm{S}^2}{2^{l_N}} - \frac{n_{\max} q_\mathrm{S}}{2^{l_0}} \;.
\end{aligned} \tag{9}$$

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

$$s^e \equiv R x^{ec} y^{cn_1} \bmod N \quad \text{and} \quad s'^e \equiv R' x'^{ec'} y^{c'n_1'} \bmod N \;.$$

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

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

where in the last step we use Equation (9) and the facts that $(a-b)^2 \geq a^2 - 2ab$ and that $0 \leq \epsilon/4q_{\mathrm{K}} \leq 1$. 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 $t' = 2t + 2(n_{\max} + 2 + q_{\mathrm{H}} + q_{\mathrm{K}} + q_{\mathrm{S}}(n_{\max}+1)) \cdot t_{\mathrm{exp}}$. □

6 Alternative Implementations

Our scheme is based on the GQ signature scheme [18], but our techniques can be applied to other identity-based signature schemes following the Fiat-Shamir paradigm as well. In particular, one can obtain efficient IBMS schemes based on RSA from [28], based on factoring from [14,13,25,26], and based on pairings from [19,9,29]. An extensive overview of the security properties of these schemes as identity-based signature schemes can be found in [2].

Acknowledgments

Mihir Bellare was supported by NSF grant CNS-0524765, a gift from Intel Corporation, and NSF CyberTrust project "CT-ISG: Cryptography for Computational Grids". 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

1. K. Barr and K. Asanovic. Energy aware lossless data compression. In *MobiSys 2003*, pages 231–244. ACM Press, 2003.
2. M. Bellare, C. Namprempre, and G. Neven. Security proofs for identity-based identification and signature schemes. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 268–286. Springer-Verlag, 2004.
3. M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *ACM CCS 06*. ACM Press, 2006.
4. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*, pages 62–73. ACM Press, 1993.
5. A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Y. Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer-Verlag, 2003.
6. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–432. Springer-Verlag, 2003.
7. C. Castelluccia, S. Jarecki, J. Kim, and G. Tsudik. A robust multisignatures scheme with applications to acknowledgment aggregation. In C. Blundo and S. Cimato, editors, *SCN 2004*, volume 3352 of *LNCS*, pages 193–207. Springer-Verlag, 2005.
8. C. Castelluccia, S. Jarecki, J. Kim, and G. Tsudik. Secure acknowledgment aggregation and multisignatures with limited robustness. *Computer Networks*, 50(10):1639–1652, 2006.
9. J. C. Cha and J. H. Cheon. An identity-based signature from gap Diffie-Hellman groups. In Y. Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 18–30. Springer-Verlag, 2003.
10. X. Cheng, J. Liu, and X. Wang. Identity-based aggregate and verifiably encrypted signatures from bilinear pairing. In O. Gervasi, M. L. Gavrilova, V. Kumar, A. Laganà, H. P. Lee, Y. Mun, D. Taniar, and C. J. K. Tan, editors, *Computational Science and Its Applications ICCSA 2005*, volume 3483 of *LNCS*, pages 1046–1054. Springer-Verlag, 2005.
11. J.-S. Coron. On the exact security of full domain hash. In M. Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer-Verlag, 2000.
12. Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong key-insulated signature schemes. In Y. Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 130–144. Springer-Verlag, 2003.
13. U. Feige, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988.

14. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer-Verlag, 1987.
15. S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. `http://eprint.iacr.org/`, 2006.
16. D. Galindo, J. Herranz, and E. Kiltz. On the generic construction of identity-based signatures with additional properties. To appear in *ASIACRYPT 2006*, *LNCS*. Springer-Verlag, 2006.
17. C. Gentry and Z. Ramzan. Identity-based aggregate signatures. In M. Yung, editor, *PKC 2006*, volume 3958 of *LNCS*, pages 257–273. Springer-Verlag, 2006.
18. L. C. Guillou and J.-J. Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In S. Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 216–231. Springer-Verlag, 1990.
19. F. Hess. Efficient identity based signature schemes based on pairings. In K. Nyberg and H. M. Heys, editors, *SAC 2002*, volume 2595 of *LNCS*, pages 310–324. Springer-Verlag, 2003.
20. K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, 71:1–8, 1983.
21. A. Joux. A one round protocol for tripartite Diffie-Hellman. In *Algorithmic Number Theory Symposium – ANTS IV*, volume 1838 of *LNCS*, pages 385–394. Springer-Verlag, 2000.
22. S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures and multisignatures without random oracles. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*. Springer-Verlag, 2006.
23. A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 74–90. Springer-Verlag, 2004.
24. S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures. In *ACM CCS 01*, pages 245–254. ACM Press, 2001.
25. K. Ohta and T. Okamoto. A modification of the Fiat-Shamir scheme. In S. Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 232–243. Springer-Verlag, 1990.
26. H. Ong and C.-P. Schnorr. Fast signature generation with a Fiat Shamir–like scheme. In I. Damgård, editor, *EUROCRYPT'90*, volume 473 of *LNCS*, pages 432–440. Springer-Verlag, 1990.
27. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
28. A. Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and D. Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer-Verlag, 1985.
29. X. Yi. An identity-based signature scheme from the Weil pairing. *IEEE Communications Letters*, 7(2):76–78, Feb. 2003.