Efficient Group Signatures without Trapdoors[⋆]

Giuseppe Ateniese and Breno de Medeiros

The Johns Hopkins University
Department of Computer Science
Baltimore, MD 21218, USA
ateniese@cs.jhu.edu, breno.demedeiros@acm.org

Abstract. Group signature schemes are fundamental cryptographic tools that enable unlinkably anonymous authentication, in the same fashion that digital signatures provide the basis for strong authentication protocols. In this paper we present the first group signature scheme with constant-size parameters that does not require any group member, including group managers, to know trapdoor secrets. This novel type of group signature scheme allows public parameters to be shared among organizations. Such sharing represents a highly desirable simplification over existing schemes, which require each organization to maintain a separate cryptographic domain.
Keywords: Group signatures, privacy and anonymity, cryptographic protocols.

1 Introduction

Group signatures allow group members to anonymously sign arbitrary messages on behalf of the group. In addition, signatures generated from the same signer are unlinkable, i.e., it is difficult to determine whether two or more signatures were generated by the same group member. In case of dispute, a group manager will be able to *open* a signature and incontestably show the identity of the signer. At the same time, no one (including the group manager) will be able to falsely accuse any other member of the group.

Group signatures were introduced by D. Chaum and E. van Heyst [16] in 1991. That was followed by several other works, but only relatively recent ones [3, 10, 11] have group public keys and group signatures with sizes that do not depend on the number of group members. (While in theory one always needs at least $\log n$ bits to uniquely identify n different users in any system, in practice $\log n$ is orders of magnitude smaller than the bit length of keys used in public key cryptography.) The scheme in [3] is the most efficient one and the only proven secure against an adaptive adversary. However, all the existing group signature schemes providing constant-size parameters require the group manager to know the factors of an RSA modulus. Sharing these factors among group managers of different organizations would compromise the security and/or the trust assumptions of the entire scheme. This paper provides the first, affirmative answer to the question of whether it is possible to design trapdoor-free group signature schemes with public parameters that do not increase linearly in size with the number of group members. We have an informal proof of security for the scheme (along the lines of the proof in [3]), and sketch some arguments that might lead to a formal proof in the sense of [5], in appendix §B.

[⋆] This work was partly funded by NSF.

1.1 Motivation

Our schemes are useful when several distinct groups or organizations must interact and exchange information about individuals while protecting their privacy. Credential transfer systems (CTS) [14, 15, 19, 17, 23, 9] are examples of such environments that can be built via group signature schemes [9]. Real-world scenarios for the use of CTS include the health-care industry, electronic voting, and transportation systems. In such cases, the added manageability and improved optimization opportunities permitted by the use of a single cryptographic domain for all participating organizations may outweigh other efficiency considerations. A CTS allows users to interact anonymously with several organizations so that it is possible to prove possession of a credential from one organization to another. Different transactions cannot be linked to real identities or even pseudonyms. It is then impossible to create profiles of users even if the organizations collude and, at the same time, users cannot falsely claim to possess credentials. Optionally, a privacy officer is able to retrieve user identities in case of disputes or emergencies. Users can thus authenticate themselves with anonymous credentials, protecting their privacy while exercising their right to vote, obtaining health services or renting a GPS-tracked automobile. The efficiency of a single signature generation or verification is measured in the human time scale. Consequently, theoretical computational advantages become less important, and instead the administrative complexity and related costs are likely to be the overwhelming concern of implementers. In these situations, a scheme with shareable parameters has a definite advantage since it eliminates the need for specialized techniques such as the ones employed in [9].

Recently in [5], it has been shown that group signatures can be built based on the assumption that trapdoor functions exist. It would be interesting to show the same but based on the existence of one-way functions. Our scheme is the first to be functionally trapdoor-free as no group member, nor even the group manager, needs to know the trapdoor information. Even though we use an RSA ring and we rely on the strong RSA assumption for security, the operation of the scheme exploits only the one-wayness of the RSA function, not its trapdoor properties.

Organization of this paper: The next section contains the definition of group signatures and the attending security requirements. In section §3 we give a high-level, intuitive description of our proposed scheme, and place it in the context of previous work. That section also introduces the cryptographic building blocks required for the scheme. The specific construction of our scheme takes all of section §4. A security analysis is provided in appendix §B.

2 Definition

In this section we present our characterization of group signature schemes. In general, a group signature scheme is defined by a family of procedures:

SETUP: A probabilistic algorithm that generates the group-specific parameters. The input to SETUP is the set of public parameters, which includes a security parameter, and its output are the group public key $\mathcal{P}$ and associated secret key $\mathcal{S}$.

`JOIN:` A prospective member executes this protocol (interacting with the group manager) to join the group. The new member's output is a membership certificate and the corresponding secret.

`SIGN:` A probabilistic algorithm that outputs a group signature when given as input a message, the group public key, a membership certificate, and the associated membership secret.

`VERIFY:` A boolean-valued algorithm used to test the authenticity of signatures generated by `SIGN`.

`OPEN:` An algorithm that given as input a message, a group signature on it, and the group secret key, extracts the membership certificate used to issue the signature, and a non-interactive proof of the signature's authorship.

2.1 Properties required

A group signature scheme must satisfy the following properties:

Correctness: A properly formed group signature must be accepted by the verification algorithm.

Unforgeability: Without possession of a membership certificate, and knowledge of associated secret, it is computationally infeasible to produce a signature that is accepted by the verification algorithm.

Anonymity/ Unlinkability: Given a group signature on a message, it is computationally infeasible to determine which member generated the signature. Moreover, given several group signatures on the same or different messages it is computationally infeasible to decide whether the signatures were issued by the same or by different group members.

Exculpability: A signature produced by a group member cannot be successfully attributed to another, and the group manager cannot generate signatures on behalf of other group members (non-framing).

Traceability: The group manager is "always" (with overwhelming probability) able to open a valid signature and determine which member signed it. Even if a coalition of group members collaborates to produce a signature on a message, possibly by combining their certificate secrets in some fashion, the group manager will succeed in attributing the signature to one of the colluding members (coalition-resistance) [3].

The requirements of unforgeability and coalition-resistance are equivalent to the requirements that group membership certificates be unforgeable under passive and active attacks, respectively, and only issuable by the group manager. In other words, a membership certificate should contain the equivalent of a digital signature by the group manager. Similarly, the requirements of traceability and exculpability imply that the group signature should hide a regular digital signature issued by the member.

These listed requirements are intuitive, but somewhat redundant: For instance, exculpability and traceability are clearly connected. In [5] the first formal model of group signature schemes was introduced, showing the relations between different requirements, and simplifying the task of proving the security of a group signature scheme. In that work, the authors claim that all security requirements of group signature schemes are derivable from two newly defined concepts: *full anonymity* and *full traceability*.

The new model introduces *two* independent group managers, one in charge of group membership management tasks, such as adding to or removing members from the group, and another responsible for opening group signatures – i.e., revealing the identity of the signer. The first manager provides *privacy* by enabling users to sign and authenticate themselves anonymously (or more properly, as arbitrary group members), while the second manager provides *accountability*, by tracing authorship of group signatures back to the issuer when required. Compromise of the first manager's secret key permits one to enroll arbitrary signing keys in the group and issue signatures on behalf of these non-entities. However it does not allow one to trace authorship of signatures. Compromise of the second manager's secret key allows one to trace authorship of signatures, but not to add new public keys to the group.

Definition 1. *Full anonymity (cf [5]): This is defined in terms of an adversarial game. The goal of the adversary is to defeat the anonymity by identifying the authorship of a group signature on a message. The game takes place in two stages. In the first (choose) stage, the adversary is given access to all members' secret keys. It also has access to an* OPEN *oracle, which it can query to find the authorship of various group signatures. The output of the first stage is two member identities i_0 and i_1, a message m and some state information S. These are given as input to the second (guess) stage, in which the adversary is also given a group signature σ on m, which is known to have been issued by either i_0 or i_1 with equal probability. The adversary can continue to query the* OPEN *oracle on signatures other than σ. The output of this stage is a guess i_b for the identity of the signer. The adversary is said to win this game if it can guess the correct signer with more than a negligible advantage over a random guess. The group signature scheme is fully anonymous if no efficient adversary can have a strategy for winning the game.*

Definition 2. *Full traceability (cf [5]): The game is played by an adversary, also in two stages. In the first (choose) stage the adversary is given access to the second group managers' secret key (the signature opening key) and can adaptively corrupt as many group members as it wishes. Let $\mathcal{C}$ be the set of corrupted members at the end of the first stage. State information (including the secret keys of the members of $\mathcal{C}$) is used as input to the guess stage, during which the adversary attempts to produce a message m and a valid group signature σ on m, such that if the (uncorrupted)* OPEN *protocol is invoked on (m, σ), it will fail to attribute σ to any group member in the set $\mathcal{C}$. (Either the* OPEN *protocol would fail to produce a valid group member identity, or it would produce the identity of a member that has not been corrupted by the adversary.) The group signature scheme is said to be fully traceable if no efficient adversary can succeed in this game with non-negligible probability.*

Remark 1. We also require that the compromise of either/both of the keys does not permit one to misattribute a signature issued by a legitimate group member. (Enrolled before the keys are compromised.) This means in particular that a group signature scheme is *not* a key escrow mechanism. This approaches differ from the one taken in [5]. There, it is the case that the first group manager escrows the users' secret keys – in particular users can be framed by compromising the first manager's secret key, which is equivalent to compromising *all* users' secret keys.

3 Preliminaries

In the group authentication problem a holder U of a group certificate interacts with a verifier V to prove his status as a group member without revealing his certificate. If the interactive protocol can be made non-interactive through the Fiat-Shamir heuristic ([20]), then the resulting algorithm will be similar to the issuing of a group signature, except that U's identity may be unrecoverable from the signature alone. The issuing of a group signature requires, in addition to a proof of membership, that U *verifiably encrypts* some information about his certificate under the group manager's public key. U must provide the verifier with an encrypted token and prove to V that the group manager is able to decrypt the token to reveal U's authorship of the signature.

A group signature can be seen as a proof of knowledge of a group certificate which provides evidence of membership. The group certificate can be generated only by the group manager GM and should be difficult to forge. In other words, the group membership certificate has the effect of a signature issued by the group manager. In addition, it has to contain some secret information generated by the group member and unknown to GM to avoid framing attacks in which GM signs on behalf of other members.

3.1 Modified ElGamal signatures

Nyberg-Rueppel signatures [25] are ElGamal-type signature variants originally designed to provide message recovery. Instead of a one-way hash function, message-recovery schemes use a redundancy function. The redundancy function R is an one-to-one mapping of messages into a so-called message-signing space $\mathcal{M}_S$. The image of R, denoted $\mathcal{M}_R$, must be sparse within $\mathcal{M}_S$ i.e., given a random element of $\mathcal{M}_S$, there is a negligible probability of it being in $\mathcal{M}_R$. Otherwise, the message-recovery scheme is vulnerable to existential forgery attacks, as redundancy functions are, by definition, efficiently invertible. The following table assumes that $\mathcal{M}_S = \mathbf{Z}_p^*$. Again, the signature calls for a random input k, and the output is a pair (r, s), where $r = R(m)g^{-k} \mod p$, and s is computed as indicated in table 1.

If in the equations above, the redundancy function $R(\cdot)$ is replaced by an one-way function then the message-recovery property is lost. On the other hand, the requirement that the image of the function be sparse in the signing space may also be dropped. This modified Nyberg-Rueppel scheme, as a signature scheme of *short messages only*, is (loosely) reducible to the hardness of discrete logarithm computations in the standard model. Alternatively, it is (loosely) reducible to the discrete logarithm in the random oracle model if extended to arbitrarily long messages through the hash-and-sign paradigm. Moreover, the form of the modified verification equation – if the one-way function is suitably chosen – lends itself to the construction of proofs of knowledge of signatures that are more efficient. (When compared to similar proofs for unmodified ElGamal-type signature variants.)

Table 1. Nyberg-Rueppel signature variants.

Variant	Signing equation	Message recovery (verification)
I	$s = k^{-1}(1 + xr) \mod q$	$R(m) = ry^{rs^{-1}} g^{s^{-1}} \mod p$
II	$s = x^{-1}(-1 + kr) \mod q$	$R(m) = ry^{sr^{-1}} g^{r^{-1}} \mod p$
III	$s = -xr + k \mod q$	$R(m) = ry^r g^s \mod p$
IV	$s = -x + kr \mod q$	$R(m) = ry^{r^{-1}} g^{sr^{-1}} \mod p$
V	$s = x^{-1}(-r + k) \mod q$	$R(m) = ry^s g^r \mod p$
VI	$s = k^{-1}(x + r) \mod q$	$R(m) = ry^{s^{-1}} g^{s^{-1}r} \mod p$

We now describe the setting of our scheme. Let $\mathcal{G}$ be some arithmetic group. Not all groups $\mathcal{G}$ where Nyberg-Rueppel (or ElGamal) signatures make sense have the characteristics needed by our scheme. In section §4, we outline the specifics of the protocols in a suitable group, namely the subgroup of quadratic residues modulo a prime p, where p is simultaneously a *safe* prime, i.e, $p = 2q+1$, with q also prime, and a *Sophie Germain* prime, that is the number $\hat{p} = 2p + 1$ is prime. There are other choices for the group $\mathcal{G}$, see appendix §C for a simpler construction in certain RSA rings.

Let $\mathcal{G}$ be a suitable group. The order of $\mathcal{G}$ may be a known prime or unknown composite number. Let g and g_1 be fixed, public generators for $\mathcal{G}$; it is assumed that the discrete logarithm of g with respect to g_1 (and of g_1 w.r.t. g) is unknown to group members. Let $y = g^x$ be the public key of the signer GM, with associated secret x. (In the group signature scheme, y corresponds to the certificate issuing key.) Finally, this signature scheme defines the message space $\mathcal{M}$ as the set of integers modulo q in the case of known order, and the set of integers smaller than some upper bound otherwise. The signing space is $\mathcal{M}_S = \mathcal{G}$, and let the one-way function $h(\cdot) : \mathcal{M} \to \mathcal{M}_S$ be defined by $h(m) = g_1^m$. Clearly, $h(\cdot)$ satisfies the requirements of a secure one-way function: $h(\cdot)$ is pre-image resistant by the hardness of computing discrete logarithms in $\mathcal{G}$. In the case of known order, it is further one-to-one, hence trivially collision-resistant. In the case of unknown order, finding a collision would reveal the order of $\mathcal{G}$, i.e., it is equivalent to factorization.

The signing and verification algorithms of the modified Nyberg-Rueppel are as follows:

$$\begin{aligned} \text{Signing:} \quad & r = g_1^m g^{-k} \quad (\text{in } \mathcal{G}); && (1) \\ & s = -xr + k \quad (\text{mod } q); && (2) \\ \text{Verification:} \quad & g_1^m = ry^r g^s \quad (\text{in } \mathcal{G}). && (3) \end{aligned}$$

We have placed "mod q" within parenthesis as that reduction is only computed when the order of $\mathcal{G}$ is a known prime. These signatures are issuable only by the signer GM, who is privy to the secret key x associated to y. Indeed, such signatures are loosely reducible, through a standard forking lemma argument [26], to the discrete logarithm problem. Please refer to appendix §B.

3.2 High level description of the scheme

A prospective new member U who wishes to join the group must have first secured a digital signature certificate with some certification authority. U starts the join protocol by choosing a random, secret value u and computing $I_U = g_1^u$. More precisely, U and GM interact so that both contribute to the randomization of u, while its value remains secret from the GM. Then U constructs a zero-knowledge proof (of knowledge) of the discrete logarithm of the pseudonym I_U with respect to g_1. U signs the pseudonym and the proof of knowledge of the pseudonym secret, and sends it to the GM to request a group membership certificate.

GM verifies the signature against U's public certificate and the correctness of the zero-knowledge proof. If both are well-formed, GM responds with the signature pair (r, s) on I_U, which is technically GM's signature on an message u known only to U. This is safe from the GM's viewpoint because both GM and U contribute to the choice of the value u. It is imperative, however, that only U knows the value u, as it is in effect the secret key allowing U to use the membership certificate to issue signatures. The equations used by GM to generate (r, s) are:

$$r = I_U g^{-k} \quad (\text{in } \mathcal{G}); \; s = -xr + k \pmod{q}, \tag{4}$$

where k is a random parameter of GM's choice, and the reduction modulo q is applied only in the case of known order. U verifies the signature, checking that:

$$I_U = r y^r g^s \quad (\text{in } \mathcal{G}). \tag{5}$$

The scheme must permit U to prove knowledge of this certificate pair (r, s) without revealing any linkable function of r, s, or u. It must also allow GM to *open* the proof and show the identity of the group member. Both problems can be solved by employing a *verifiable encryption* of digital signature schemes. However, unlinkability between different protocol executions is not a requirement of verifiable encryption schemes, and indeed existing protocols for ElGamal-type signature schemes do not provide it. Hence, it would be possible to link two or more verifiable encryptions, which is equivalent to linking two or more group signatures from the same signer. This is because, in existing schemes, the first value r of the signature pair (r, s) is revealed and the actual protocol is applied only to the second value s, reducing then the problem of verifiable encryption of a digital signature to the simpler problem of verifiably encrypting a discrete logarithm (see [8, 1, 22, 2] for details).

To solve this issue, it is necessary to ElGamal encrypt the value r as well, and prove in zero-knowledge that a Nyberg-Rueppel signature is known on a secret value u. More concretely, every time the group member must use the certificate, she encrypts the inverse of the value r, to get the ElGamal pair $(R_1, R_2) = (r^{-1} y_2^{\ell}, g_2^{\ell})$. This encryption is under the second public key $y_2 = g_2^z$ of the group manager, used for opening group member signatures, with associated secret z.

The group member also encrypts his pseudonym: $(Y_1, Y_2) = (I_U y_2^{\ell'}, g_2^{\ell'})$. Notice that the product cipher is:

$$(R_1 Y_1, R_2 Y_2) = (I_U r^{-1} y_2^{\ell+\ell'}, g_2^{\ell+\ell'}) = (y^r g^s y_2^{\ell+\ell'}, g_2^{\ell+\ell'}) \tag{6}$$

In order to prove knowledge of a membership certificate, the member U releases the above ElGamal encrypted pairs (R_1, R_2) and (Y_1, Y_2) and proves that the product cipher encrypts some information which the signer can write in two ways, i.e., as the product $I_U r^{-1}$ for pseudonym I_U (for which the signer knows the corresponding pseudonym secret) and value r, and also as $y^r g^s$, for the same value r and some s known to the signer. In other words, the signer shows that an equation like (6) holds for the product cipher.

To proceed, we must overcome a difficulty with equation (6): The value in the exponent is reduced modulo the order of the group $\mathcal{G}$, while the encrypted value r is an element of $\mathcal{G}$ itself. The reduction function does not preserve group operations, it is not multiplicative; and the method for proving equality between an ElGamal-encrypted value and a logarithm, due to Stadler [28], cannot be directly applied. The solution is to employ a technique due to Boudot [7] that permits efficient comparison between logarithms in different groups. So we use an auxiliary group $\mathcal{F}$ of order compatible with the operations in $\mathcal{G}$. We release a commitment to the value r as an exponent of an element of $\mathcal{F}$, and we show that it equals (up to modular reduction), the exponent of y in the representation with respect to the basis $\{y, g\}$ of the value ElGamal encrypted in the product cipher $(R_1 Y_1, R_2 Y_2)$. Next, we use Stadler's technique to prove the equality of the encrypted value r (in the pair R_1, R_2 of $\mathcal{G}$), with the value committed as an exponent in $\mathcal{F}$.

To complete the sign protocol, the signer proves knowledge of the discrete logarithm to basis g of the value I_U which is ElGamal encrypted in the pair (Y_1, Y_2). This shows that the group manager will be able to open the signature with just an ElGamal decryption operation.

Proofs of knowledge In this paper we make use of several types of proofs of knowledge about various relations between secrets. All these proofs of knowledge have been presented elsewhere. In order to harmonize the notation, which varies from author to author, and make the paper self-contained, we include an appendix (§A) in which we reproduce these various results.

4 The scheme

We now describe the scheme more concretely, starting with $\mathcal{T}$, the set of shared public parameters. $\mathcal{T}$ specifies security parameters δ, ϵ, σ, σ_2, and τ, and a secure hash function $\mathcal{H}$ that maps bit-strings of arbitrary length into bit-strings of fixed length τ. A typical set of choices would be $\delta = 40$, $\sigma = 40$, $\sigma_2 = 552$, $\tau = 160$, and $\mathcal{H}(\cdot) =$ SHA-1$(\cdot)$. The parameter ϵ should be larger than 1 by a non-negligible amount. These security parameters impact the security and efficiency of the various proofs of knowledge used in the scheme. (Notation as in appendix §A.) $\mathcal{T}$ also specifies an arithmetic group $\mathcal{G}$ and three generators g, g_1 and g_2 of $\mathcal{G}$.

In this section we assume that $\mathcal{G}$ is the quadratic residues subgroup of the multiplicative residues module p, where p is simultaneously a safe prime, i.e., and $p = 2q + 1$, with q also prime, and a Sophie Germain prime, i.e., the number $\hat{p} = 2p + 1$ is prime. Primes $\hat{p}$ such that $\hat{p} = 2p + 1$, and $p = 2q + 1$, with p and q also prime are called *strong* primes. (More generally, if $\hat{p} = mp + 1$ and $p = nq + 1$ with small m, and n, are also called strong primes, but $m = n = 2$ gives the most efficient scheme.) See [18, 21] for efficient methods to generate such primes. In order to choose g it is enough to pick a random element g' in $\mathbf{Z}_p^*$ and set $g \equiv g'^2 \mod p$, provided that $g \not\equiv 1 \mod p$. The same procedure should be used to obtain g_1 and g_2.

The scheme also requires an auxiliary group $\mathcal{F}$ of order p, which in this section will be chosen as the quadratic subgroup of the multiplicative residues modulo $\hat{p}$. Furthermore, the scheme requires a second auxiliary group $\mathcal{E}$ of unknown composite order $\hat{n}$. A trusted party generates a composite modulus n, plus a proof P that n is the product of two safe primes. The group $\mathcal{E}$ is defined as the quadratic residue subgroup of the multiplicative residues modulo n. The order of $\mathcal{E}$ is the universally unknown number $\phi(n)/4$. Group managers of competing organizations may all share the same modulus n, as the operation of the scheme does not require *anybody* to know the RSA trapdoor associated to n, and the trusted party may safely forget the factorization at its discretion.

Table 2. Shared and group specific parameters

Shared parameters
Security parameters: δ, ϵ, σ, σ_2, τ;
Secure hash function: $\mathcal{H}(\cdot) : \{0,1\}^* \longrightarrow \{0,1\}^\tau$;
$\hat{p}$, p, q, primes s.t. $\hat{p} = 2p + 1$ and $p = 2q + 1$;
$\mathcal{G} = \{x \in \mathbf{Z}_p^* : \exists\, a \in \mathbf{Z}_p^*$ s.t. $x \equiv a^2 \mod p\}$;
$\mathcal{F} = \{x \in \mathbf{Z}_{\hat{p}}^* : \exists\, a \in \mathbf{Z}_{\hat{p}}^*$ s.t. $x \equiv a^2 \mod \hat{p}\}$;
$\mathcal{E} = \{x \in \mathbf{Z}_n^* : \exists\, a \in \mathbf{Z}_n^*$ s.t. $x \equiv a^2 \mod n\}$;
g, g_1, and g_2, generators of $\mathcal{G}$.
Group-specific parameters
$\mathcal{S}$, a string including y and y_2;
CA's signature: CERT$_{CA}(\mathcal{S})$.

Table 3. The `JOIN` protocol

$U \longrightarrow GM : J_U = I^m \mod p$
$GM \longrightarrow U : a,\ b \mod q$
$U \longrightarrow GM : Sig_U(I_U = J_U^a g_1^b,\ PK[u : I_U = g_1^u])$
$GM \longrightarrow U : r = I_U g^{-k} \mod p,\ s = -xr + k \mod q$

The above public parameters can be further certified if so desired. A proof of primality can be provided for each of the primes; as for g, g_1 and g_2, anybody can verify their correct generation by testing that each is not congruent to 0 or 1 modulo p, and then verifying that each is a square, by computing the Legendre symbol and checking that: $\left(\frac{g}{p}\right) = \left(\frac{g_1}{p}\right) = \left(\frac{g_2}{p}\right) = 1$.

In order to setup a group using the shared parameters above, the group manager GM chooses x and z at random among the numbers $[1, q-1]$ and set the public keys $y = g^x$, and $y_2 = g_2^z$. The group manager should proceed to register these group-specific parameters with some certification authority. The GM would prepare a statement $\mathcal{S}$ containing (minimally) a description of the group signature algorithms, a reference to the shared parameters, GM's name, the group-specific parameters y, y_1, and y_2, and some timed information, such as start and expiration dates. The GM should obtain a certificate CERT$_{CA}(\mathcal{S})$ from the CA establishing the group-specific parameters.

Let $Sig_U(\cdot)$ denote U's signature algorithm. To join the group, a prospective member U chooses a random secret m in the interval $[1, q-1]$, computes $J_U = g_1^m$, and sends this value to GM, who responds with two values a, and b in $[1, q-1]$. U computes his pseudonym as $I_U = J_U^a g_1^b$, and its associated secret $u = am + b \mod q$. Next, U constructs a non-interactive proof of knowledge of the logarithm to basis g_1 of this pseudonym (see appendix A), and also his signature $S = Sig_U(I_U, PK)$ on both the pseudonym and the proof-of-knowledge just constructed. U forwards to the GM this signature S.

The GM now verifies that the pseudonym incorporated his contribution, i.e., $I_U = J_U^a g_1^b$. This step is important because u is unknown to GM, who must sign it. Since the GM contributed to u's randomness, that does not constitute a threat to the GM's signature algorithm. The GM also verifies the correctness of the proof-of-knowledge and U's signature. If satisfied, the GM generates a random $k \mod q$, and computes $r = I_U g^{-k} \mod p$, checking that $r < c$, where c equals:

$$c = p - 2^{\sigma+\tau/2+2}\sqrt{p}, \tag{7}$$

and repeating the process of computing other random k and r until such an r is found. Note that $r < c$ with overwhelming probability in a single attempt, because since the quadratic residues are nearly uniformly distributed in the interval $[1, p-1]$, we have that $r < c$ with probability close to $1 - \frac{2^{\sigma+\tau/2+2}}{\sqrt{p}} > 1 - 2^{-645}$ if the security parameters have the typical values $\delta = 40$, $\tau = 160$ and p has at least 768 significant bits. This very minor restriction on the possible values of r reflects requirements of the proof of equality of discrete logarithms in distinct groups, as we shall see later. After a suitable r is found, U computes $s = k - xr \mod q$, and sends the certificate (r, s) to U. The GM also records the signature S, which ties U's identity to the certificate's pseudonym. U verifies that the certificate (r, s) satisfies the verification equation, and if so, accepts it as valid.

We now describe the protocol `SIGN`. One goal of this protocol is that U convince a verifier V of its knowledge of a membership certificate (r, s) as above. As in section §3, the signer chooses random ℓ, and ℓ', with $0 < \ell, \ell' < q$. U releases the ElGamal encrypted pairs:

$$(Y_1, Y_2) = (I_U y_2^{\ell'}, g_2^{\ell'}); \quad (R_1, R_2) = (r^{-1} y_2^{\ell}, g_2^{\ell});$$

Next, U demonstrates that the pseudonym I_U is encrypted by the pair (Y_1, Y_2), and proves knowledge of the pseudonym secret u, by executing $PK[u, \ell' : Y_1 = g_1^u y_2^{\ell'} \wedge Y_2 = g_2^{\ell'}]$. This step is crucial to prevent framing attacks against U, as not even the group manager can execute it without knowledge of u.

Continuing with the `SIGN` protocol, U generates a fresh, random generator χ of the group $\mathcal{F}$, and computes a (computationally zero-knowledge) commitment to the value r as $E_1 = E_1(r, 0) = \chi^r$. In the language of appendix §A, this is a (degenerate) commitment to the value r in the group $\mathcal{F}$, with respect to the generator χ.

U also generates a commitment to r in the auxiliary group $\mathcal{E}$ of unknown order. For that, U uses two generators β and γ of $\mathcal{E}$, where β and γ are provably randomly generated, so that U cannot know their relative discrete logarithm. For instance, γ and β can be generated as the squares of two consecutive values of a secure pseudo-random number generator *SPRNG*. The commitment is computed as $E_2 = E_2(r, s_2) = \gamma^r\beta^{s_2}$, where s_2 is a random parameter of U's choice: $s_2 \in [-2^{\kappa+\tau+1}, 2^{\kappa+\tau+1}]$, where $2^{\kappa-1} \leq |\mathcal{E}| < 2^{\kappa}$. Notice that the value $R_1Y_1 = I_U r^{-1} y_2^{\ell+\ell'} = y^r g^s y_2^{\ell+\ell'}$ is also a commitment to the value r in the group $\mathcal{G}$, with generators y, g, and y_2. Denote it by $E_3 = R_1Y_1$.

In the next step, U reveals the commitments E_1, E_2, and the respective generators γ, β, and χ. (In the case of γ and β, U must also reveal the seed of the *SPRNG* that leads to the computation of γ and β.) U then shows that E_1, E_2 and E_3 all are commitments to the same value r. (Notice that we are following the efficient construction found in [7], repeated in detail here for reasons of convenience.) U executes two proofs of equality of two committed values (def. 10). In the first proof U sends V a triple (c', D', D_1') satisfying: $c' = \mathcal{H}(\chi||\gamma||\beta||E_1||E_2||\chi^{D'}E_1^{-c'} \mod \hat{p}||\gamma^{D'}\beta^{D_1'}E_2^{-c'} \mod n)$. Again, refer to def. (10) for how to build these proofs. In agreement with the notation in appendix §A,we denote the above by $PK[r, s_2 : E_1 = E_1(r, 0) \wedge E_2 = E_2(r, s_2)]$. Then U sends V a quintuple (c, D, D_1, D_2, D_3) satisfying: $c = \mathcal{H}(\gamma||\beta||y||g||y_2||E_2||E_3||\gamma^D\beta^{D_1}E_2^{-c} \mod n||y^D g^{D_2} y_2^{D_3} E_3^{-c} \mod p||g_2^{D_3}(Y_2R_2)^{-c} \mod p)$. Denote that by $PK[r, s, s_2, t : E_2 = E_2(r, s_2) \;\wedge\; E_3 = E_3(r, s, t) \;\wedge\; Y_2R_2 = g_2^t]$.

If all of the commitments E_1, E_2, and E_3 took place within the same group the above would be a proof of equality of the committed exponent in each of the commitments. However, as the order of the groups differ, we have only proved knowledge of an integer value r which satisfies

$$r \equiv r_1 \mod p, \text{ and } r \equiv r_3 \mod q, \tag{8}$$

where r_1 and r_3 are, respectively, the exponents committed in E_1 and E_3, while r is the exponent committed in E_2. (As U does not know the order of $\mathcal{E}$, it cannot set up a modular equation that the exponent of E_2 should satisfy, and must use the full integer value r.) U could cheat and pass the "proof" above for any two different values r_1 and r_3, by setting r in E_2 to equal the solution, computed via the Chinese Remainder Theorem, to the pair of modular equations in (8). Thus, a non-member U' would be able to forge the proof of knowledge of a certificate, by choosing r_3 and s arbitrarily, computing the value r_1 that would make the certificate equation work, and then solving the pair of equations (8) for an r that reduces to $r_1 \mod p$ and $r_3 \mod q$, respectively. In the cheating case, however, because $r_1 \not\equiv r_3 \mod q$, U' computes a value $r > p$ as the solution of 8. Thus, if U' is required to prove that the value r_2 committed in E_2 is within an interval of width at most p, this forgery attack is prevented; and the commitments must all hide the same value. So to complete the "proof of equality of commitments in different groups," U must construct a proof that the value r is restricted to an interval of width at most p. For that, U uses the fact that $r < c$, and constructs

Table 4. The `SIGN` protocol

`Proof arguments:`
Y_1, Y_2, R_1, R_2, χ, γ, β, E_1, and E_2.
`Signature of knowledge:`
$SPK[u, \ell', \ell, r, s, s_2, t : Y_1 = g_1^u y_2^{\ell'} \;\wedge\; Y_2 = g_2^{\ell'}$
$\wedge\; E_1 = E_1(r, 0) = \chi^r \;\wedge\; R_1 = r^{-1} y_2^{\ell} \wedge R_2 = g_2^{\ell}$
$\wedge\; E_2 = E_2(r, s_2) = \gamma^r \beta^{s_2} \;\wedge\; r \in [-2^{\delta+\tau/2+1}\sqrt{c}, c + 2^{\delta+\tau/2+1}\sqrt{c}]$
$\wedge\; E_3 = E_3(r, s, t) = Y_1 R_1 = y^r g^s y_2^t \;\wedge\; Y_2 R_2 = g_2^t\,](\mathcal{M})$

the proof of knowledge that a committed value lies in a slightly larger interval, def. (13): $PK[r, s_2 : E_2 = E_2(r, s_2) \;\wedge\; r \in [-2^{\delta+\tau/2+1}\sqrt{c}, c + 2^{\delta+\tau/2+1}\sqrt{c}]]$. To observe that the interval in question has width smaller than p, notice that its width equals $c + 2^{\delta+\tau/2+2}\sqrt{c} < c + 2^{\delta+\tau/2+2}\sqrt{p} = p$, by choice of c (see equation 7).

Finally, U must show that the exponent committed in E_1 equals the value encrypted in the pair (R_1, R_2), by executing (definition 14): $PK[r, t : E_1 = \chi^r \;\wedge\; R_1 = r^{-1} y_2^t \wedge W_2 = g_2^t]$. The actual protocol `SIGN` combines all the proofs of knowledge into a single signature of knowledge. This is done by simultaneously committing to all the inputs of the proofs and using the resulting challenge in all the verification equations (à la Fiat-Shamir). In addition, the message $\mathcal{M}$ to be signed is used as an extra input of the hash function.

The protocol is summarized in table 4. Moreover, algorithm `VERIFY` can be derived immediately from the above formal description of `SIGN` as a proof of knowledge of a group certificate.

As for `OPEN`, it is enough that the group manager decrypts the pair (Y_1, Y_2) to obtain the value I_U and the corresponding group membership certificate. GM constructs a proof that I_U is indeed the value encrypted in (Y_1, Y_2) *without revealing the group secret* x: $PK[x : Y_1 I_U^{-1} = Y_2^x \wedge y_2 = g_2^x]$, a publicly verifiable *proof of authorship* of the signature.

5 Conclusions

In this paper we introduced the first group signature scheme with constant-size parameters that does not require any group members, including group managers, to know trapdoor secrets. Our scheme is not bound to a specific setting but it can work in various groups where the Decision Diffie-Hellman assumption holds: The appendix §C contains a simpler construction in an RSA ring.

Our scheme is less efficient than the state-of-the-art scheme in [3]. However, the scheme in [3] requires the group manager to know trapdoor information which cannot be shared with other group managers, thus making it difficult to enable collaboration among distinct groups.

6 Acknowledgments

We are grateful to an anonymous referee who suggested changes to our security arguments that eventually led to a simplification of the scheme and its proof of security.

References

1. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In *Proc. of EUROCRYPT '98,* LNCS vol. 1403, Springer-Verlag, 1998.
2. G. Ateniese. Efficient Verifiable Encryption (and Fair Exchange) of Digital Signatures. In 6th ACM CCS, pp. 138-146, 1999.
3. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In M. Bellare, editor, *Proc. of CRYPTO 2000,* LNCS, Springer-Verlag, 2000.
4. G. Ateniese and G. Tsudik. Some open issues and directions in group signatures. In *Financial Cryptography* 1999. LNCS 1648, Springer-Verlag, 1999.
5. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction based on General Assumptions. In *Proc. of EUROCRYPT '03,* LNCS vol. 2656, Springer-Verlag, 2003.
6. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. of the* 1^{st} *ACM CCS*, 1993.
7. Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *Proc. of EUROCRYPT'00,* LNCS vol. 1807, Springer Verlag, 2000.
8. Jan Camenisch and Ivan Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In *Proc. of ASIACRYPT 2000,* LNCS vol. 1976, Springer Verlag.
9. Jan Camenisch and Anna Lysyanskaya. Efficient Non-transferable Anonymous Multi-show Credential System with Optional Anonymity Revocation. In *Proc. of EUROCRYPT'01,* Springer Verlag, 2001.
10. Jan Camenisch and Markus Michels. A group signature scheme with improved efficiency. In *Proc. of ASIACRYPT '98,* LNCS vol. 1514, Springer-Verlag, 1998.
11. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Proc. of CRYPTO'97,* LNCS vol. 1296, Springer-Verlag, 1997.
12. A. Chan, Y. Frankel, and Y. Tsiounis. Easy come – easy go divisible cash. In *Proc. of EUROCRYPT'98,* LNCS vol. 1403, Springer-Verlag, 1998.
13. A. Chan, Y. Frankel, and Y. Tsiounis. Easy come – easy go divisible cash. Updated version with corrections, GTE Technical Report. 1998. Available at `http://www.ccs.neu.edu/home/yiannis`.
14. D. Chaum, *Security Without Identification: Transactions Systems to Make Big Brother Obsolete, C*ACM Vol. 28, No. 10, October 1985.
15. D. Chaum and J. Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *Proc. of CRYPTO'86,* pp. 118-167, Springer-Verlag, 1986.
16. D. Chaum and E. van Heyst. Group signatures. In *Proc. of CRYPTO'91,* LNCS vol. 547, Springer-Verlag, 1991.
17. L. Chen. Access with pseudonyms. In *Cryptography: Policy and Algorithms,* pp. 232-243, Springer-Verlag, 1995.
18. Ronald Cramer and Victor Shoup. Signature schemes based on the strong RSA assumption. In *ACM Transactions on Information and System Security,* 2000.

19. I. Damgård. Payment systems and credential mechanisms with provable security against abuse by individuals. In *Proc. of CRYPTO '88,* Springer-Verlag, 1988.
20. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proc. of CRYPTO'86,* Springer Verlag, 1987.
21. Marc Joye, Pascal Paillier and Serge Vaudenay. Efficient generation of prime numbers. In *Cryptographic Hardware and Embedded Systems – CHES 2000,* LNCS vol. 1965, Springer Verlag, 2000.
22. J. Kilian and E. Petrank. Identity escrow. In CRYPTO '98, vol.1642 of LNCS, pp. 169-185, Berlin, 1998. Springer-Verlag.
23. A. Lysyanskaya, R. Rivest, A. Sahai, and S. Wolf. Pseudonym Systems. In Selected Areas in Cryptography. Springer-Verlag 1999.
24. Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* Chapter §11, note 11.83, pp. 461–462. CRC Press, 1996.
25. Kaisa Nyberg and Rainer A. Rueppel. Message recovery for signature schemes based on the discrete logarithm problem. In *Proc. of EUROCRYPT'94,* Springer Verlag, 1994.
26. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Proc. of EUROCRYPT'96*, Springer Verlag, newblock 1996.
27. Amit Sahai. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. In *Proc. of the* 40^{th} *FOCS Symposium.* 1999.
28. Markus Stadler. Publicly Verifiable Secret Sharing. In *Proc. of EUROCRYPT'96,* LNCS vol. 1070, Springer Verlag, 1996.

A Proofs of knowledge

All the proofs of knowledge listed in this section have been proved zero-knowledge in a statistical or computational sense within the random oracle model, under the Decisional Diffie-Hellman assumption, and the Strong RSA assumption, explained below.

Notation 1 (Groups and generators).

– $\mathcal{J}$ stands for an arithmetic group, such as an RSA ring with composite modulus n or the group $\mathbf{Z}_p^*$ of non-zero (multiplicative) residues modulo p.
– g stands for an element of $\mathcal{J}$ of unknown composite order or known prime order. Let q be the order of g.
– Let κ be the smallest integer such that 2^κ is larger than q. We assume that κ is known, even if q is not.
– g generates the subgroup $\mathcal{G}$ of $\mathcal{J}$.

Let $\mathcal{H}$ stand for a secure hash function which maps arbitrarily long bit-strings into bit-strings of fixed length τ. Let ϵ denote a second security parameter.

Definition 3 (Decisional Diffie-Hellman assumption (DDH)). *Let $\mathcal{J}$ be a group and g an element of known prime, or unknown composite, order q in $\mathcal{J}$. Let $\mathcal{G} = \langle g \rangle$ be the subgroup generated by g in $\mathcal{J}$. The DDH assumption for $\mathcal{G}$ is then there is no efficient (randomized, probabilistic) algorithm that can distinguish between the two following distributions in $\mathcal{G}$:*

$$\{(h, i, j), \text{ where } h, i, j \text{ are independently randomly distributed (i.r.d.) in } \mathcal{G}\}$$

and

$$\{(h', i', j'), \text{ where } h' = g^x, i' = g^y, j' = g^{xy} \text{ for i.r.d. } x, y \text{ with } 0 \le x, y < q\}$$

A triple of group elements such as (h', i', j') above is called a *Diffie-Hellman triple.* The DDH assumption is thus the statement that there is no efficient algorithm to distinguish between Diffie-Hellman triples and randomly generated triples.

Definition 4 (Strong RSA assumption (SRSA)). *Let $n = pq$ be a composite modulus, where p and q are two large primes. The strong RSA assumption states that there is no efficient (randomized, probabilistic) algorithm that, given as input n and an integer y, but not the factorization of n, can produce two other integers u and e, where $e > 1$ and $u^e \equiv y \mod n$.*

SRSA underlies the security of the proof of equality of logarithms in distinct groups (10).

Definition 5 (Proof of knowledge of a discrete logarithm). *U can prove to a verifier V his knowledge of an integer x in $\{0, \ldots, 2^\kappa - 1\}$, such that $h = g^x$, by releasing integers s and c, with s in $\{-2^{\epsilon(\tau+\kappa)+1}, \ldots, 2^{\epsilon(\tau+\kappa)+1} - 1\}$ and c in $\{0, \ldots, 2^\tau - 1\}$, s.t. $c = \mathcal{H}(g||h||g^s h^c)$, where the symbol $||$ denotes string concatenation.*

In order to compute the pair (s, c), U generates a random integer k in $\{-2^{\epsilon(\tau+\kappa)}, \ldots, 2^{\epsilon(\tau+\kappa)} - 1\}$ and sets $c = \mathcal{H}(g||h||g^k)$, and $s = k - cx$ (as integer). Denote it by (notation introduced in [11]): $PK[x \,:\, h = g^x]$.

This proof of knowledge can be transformed into a digital signature, with x being the secret key associated with public key h. To sign an arbitrary bitstring m, we instead compute c as: $c = \mathcal{H}(g||h||g^s h^c||m)$. Denote this *signature of knowledge* ([11]) by: $SPK[x \,:\, h = g^x](m)$.

Returning to the notation in definition (5), if the order q of the group $\mathcal{G}$ is known, then operations on the exponents should be computed modulo q, and some statements about the size of parameters can be simplified. In the above we would substitute:

$$x \in \{0, \ldots, 2^\kappa - 1\} \text{ by } x \in \{0, \ldots, q - 1\},$$
$$s \in \{-2^{\epsilon(\tau+\kappa)+1}, \ldots, 2^{\epsilon(\tau+\kappa)+1} - 1\} \text{ by } s \in \{0, \ldots, q - 1\}, \text{ and}$$
$$s = k - cx \text{ (in } \mathbf{Z}) \text{ by } s = k - cx \mod q.$$

In the following definitions we assume the group order q is unknown; as above, it is straightforward to adapt them to the case of known order.

Definition 6 (Proof of knowledge of a common discrete logarithm). *U can prove to a verifier V his knowledge of an x (with $0 \le x < 2^\kappa$) s.t. two lists $g_1, g_2, \ldots, g_\ell$ and $h_1, h_2, \ldots, h_\ell$ (of elements of $\mathcal{G}$) satisfy $h_i = g_i^x, i = 1 \ldots \ell$, by releasing s and c ($-2^{\epsilon(\tau+\kappa)+1} \le s < 2^{\epsilon(\tau+\kappa)+1}$ and $0 \le c < 2^\tau$) s.t.*
$$c = \mathcal{H}(g_1|| \ldots ||g_\ell||h_1|| \ldots ||h_\ell||(g_1 \ldots g_\ell)^s (h_1 \ldots h_\ell)^c).$$

U computes $c = \mathcal{H}(g_1|| \ldots ||g_\ell||h_1|| \ldots ||h_\ell||(g_1 \ldots g_\ell)^k)$ for a randomly chosen k ($-2^{\epsilon(\tau+\kappa)} \le k < 2^{\epsilon(\tau+\kappa)}$), and sets $s = k - cx$. Denote it by: $PK[x \,:\, h_1 = g_1^x \wedge \cdots \wedge h_\ell = g_\ell^x]$.

Definition 7 (Proof of knowledge of a representation). *U can prove his knowledge of elements $x_1, \ldots, x_\ell$ (with $0 \le x_i < 2^\kappa$) s.t. a given element A satisfies $A = g_1^{x_1} \cdots g_\ell^{x_\ell}$, by releasing s_i and c ($-2^{\epsilon(\tau+\kappa)+1} \le s_i < 2^{\epsilon(\tau+\kappa)+1}$; $0 \le c < 2^\tau$) s.t. $c = \mathcal{H}(g_1|| \ldots ||g_\ell||A||g_1^{s_1} \ldots g_\ell^{s_\ell} A^c)$.*

Again, U computes $c = \mathcal{H}(g_1|| \ldots ||g_\ell||A||g_1^{k_1} \ldots g_\ell^{k_\ell})$ for randomly chosen k_i ($-2^{\epsilon(\tau+\kappa)} \le k_i < 2^{\epsilon(\tau+\kappa)}$), and sets $s_i = k_i - cx_i$. Denote it by: $PK[x_1, \ldots, x_\ell \,:\, A = g_1^{x_1} \cdots g_\ell^{x_\ell}]$.

The next two proofs of knowledge assert that a committed value lies in an interval. The first one was introduced in [12], and corrected in [13]. The second one, which uses the first as building block, was introduced in [7], and is used in our scheme.

Let g, h be two elements of $\mathcal{G}$. Assume that g and h are constructed in a provably random way, for instance as consecutive images of a secure pseudo-random generator. Generating g and h in such a way ensures that no one knows the discrete logarithm of g to basis h, or that of h to basis g.

Definition 8 (Commitment to a secret value). *Let x be a secret value held by U. Let g and h be two provably random generators of $\mathcal{G}$. We say that $E = E(x, r) = g^x h^r$ is a commitment to the value x in $\mathcal{G}$, where r is a randomly generated value, $0 < r < q$.*

If q is unknown, then one must choose r in a larger interval, say $-2^{\kappa+\tau+1} < r < 2^{\kappa+\tau+1}$, to ensure that all elements in the interval $[0, q-1]$ are sampled nearly uniformly. The commitment reveals nothing about r in a statistical sense.

Let $\mathcal{E}$ be a distinct arithmetic group of unknown composite order n. For instance, $\mathcal{E}$ can be chosen as the subgroup of quadratic residues in an RSA ring. Let $g = g_1$, g_2, $h = h_1$, and h_2 be provably random generators of $\mathcal{E}$. We assume that the smallest integer λ s.t. $2^\lambda > n$ is known. Assume U has published two commitments, $E = E_1(x, r) = g_1^x h_1^{r_1}$ in $\mathcal{G}$, and a second commitment $E_2(x, r_2) = g_2^x h_2^{r_2}$.

Let δ, σ and σ_2 be other security parameters. Assume further that $x < b$.

Definition 9 (Proof of knowledge of a committed value). *U can prove in* ZK *to a verifier V knowledge of a number x committed through $E = E(x, r) = g^x h^r$, by sending V a triple (c, D, D_1) satisfying: $c = \mathcal{H}(g||h||E||g^D h^{D_1} E^{-c} \mod n)$.*

U generates random $t \in [1, 2^{\delta+\tau/2} b + 1]$ and $s \in [1, 2^{\delta+\tau/2+\sigma} n - 1]$; computes $W = g^t h^s \mod n$; computes $c = \mathcal{H}(g||h||E||W)$; and finally computes $D = t + cx$, $D_1 = s + cr$ (in **Z**).

Definition 10 (Proof of equality of two committed values). *U can prove in* ZK *to a verifier V that two commitments $E_1 = E_1(x, r_1)$ and $E_2 = E_2(x, r_2)$ hide the same exponent x, by sending V a quadruple (c, D, D_1, D_2) satisfying: $c = \mathcal{H}(g_1||h_1||g_2||h_2||E_1||E_2||g_1^D h_1^{D_1} E_1^{-c} \mod n||g_2^D h_2^{D_2} E_2^{-c} \mod n)$.*

U generates the random values $t \in [1, 2^{\delta+\tau/2} b + 1]$, $s_1 \in [1, 2^{\delta+\tau/2+\sigma} n - 1]$, and $s_2 \in [1, 2^{\delta+\tau/2+\sigma_2} n - 1]$. Next, U computes $W_1 = g_1^t h_1^{s_1} \mod n$, $W_2 = g_2^t h_2^{s_2} \mod n$; and sets $c = \mathcal{H}(g_1||h_1||g_2||h_2||E_1||W_1||W_2)$. Finally, U computes $D = t + cx$, $D_1 = s_1 + cr_1$, $D_2 = s_2 + cr_2$ (in **Z**). Denote this by $PK[x, r_1, r_2 : E_1 = E_1(x, r_1) \wedge E_2 = E_2(x, r_2)]$.

Definition 11 (Proof that a committed number is a square). *U can convince a verifier V that the commitment $E = E(x^2, r_1) = g^{x^2} h^{r_1} \mod n$ ($r_1 \in [-2^\sigma n + 1, 2^\sigma n - 1]$) contains the square of a number known to U, by sending V the quintuple (F, c, D, D_1, D_2), where $c = \mathcal{H}(g||h||E||F||F^D h^{D_1} E^{-c} \mod n||g^D h^{D_2} F^{-c} \mod n)$.*

Indeed, U generates a random r_2 in $[-2^\sigma n + 1, 2^\sigma n - 1]$, and sets $F = g^x h^{r_2}$. Notice now that U can rewrite E in the basis $\{F, h\}$ as $E(x, r^3) = F^x h^{r_3} \mod n$, where $r_3 = r_1 - r_2 x$, and $r_3 \in [-2^\sigma bn + 1, 2^\sigma bn - 1]$. It is enough then for U to use the previous proof of equality of the exponent x committed though $E_1 = F = E(x, r_2)$ and $E_2 = E = E(x, r_3)$, i.e., execute $PK[x, r_2, r_3 : F = g^x h^{r_2} \wedge E = F^x h^{r_3}]$. Denote this by $PK[x, r_1 : E = E(x^2, r_1)]$.

Definition 12 (Proof that a committed number lies in a larger interval). *A prover U can convince a verifier V that a number $x \in [0, b]$ which is committed in $E = E(x, r) = g^x h^r \mod n$ ($r \in [-2^\sigma n + 1, 2^\sigma n - 1]$), lies in the much larger interval $[-2^{\sigma+\tau/2} b, 2^{\sigma+\tau/2} b]$, by sending V the triple (C, D_1, D_2), where $D_1 \in [cb, 2^{\delta+\tau/2} b - 1]$, and $C = \mathcal{H}(g||h||E||g^{D_1} h^{D_2} E^{-c})$; $c = C \mod 2^{\tau/2}$.*

U generates randoms $s \in [0, 2^{\delta+\tau/2} b - 1]$, $t \in [-2^{\delta+\tau/2+\sigma} n + 1, 2^{\delta+\tau/2+\sigma} n - 1]$; computes $W = g^s h^t \mod n$; computes $C = \mathcal{H}(g||h||E||W)$, and $c = C \mod 2^{\tau/2}$; and sets $D_1 = s + cx$, $D_2 = t + cr$, repeating the procedure from the beginning if $D_1 \notin [cb, 2^{\delta+\tau/2} b - 1]$. We denote the above by $PK_{CFT}[x, r : E = E(x, r) \wedge x \in [-2^{\delta+\tau/2} b, 2^{\delta+\tau/2} b]]$.

Definition 13 (Proof that a committed number lies in a slightly larger interval). *A prover U can convince a verifier V that a number $x \in [a, b]$, committed in $E = E(x, r) = g^x h^r \mod n$ ($r \in [-2^\sigma n + 1, 2^\sigma n - 1]$) lies in the slightly larger interval $[a - \alpha, b + \alpha]$, where $\alpha = 2^{\delta+\tau/2+1}\sqrt{b-a}$, by releasing $\tilde{E}_1, \bar{E}_1$, and proving: $PK[x, r : E = E(x, r)]$, $PK[\tilde{x}_1, \tilde{r}_1 : \tilde{E}_1 = E(\tilde{x}_1^2, \tilde{r}_1)]$, $PK[\bar{x}_1, \bar{r}_1 : \bar{E}_1 = E(\bar{x}_1^2, \bar{r}_1)]$, $PK_{CFT}[\tilde{x}_2, \tilde{r}_2 : \tilde{E}_2 = E(\tilde{x}_2, \tilde{r}_2) \wedge \tilde{x}_2 \in [-\alpha, \alpha]]$, where $\tilde{E}_2 = \frac{E}{g^a \tilde{E}_1} \mod n$, $PK_{CFT}[\bar{x}_2, \bar{r}_2 : \bar{E}_2 = E(\bar{x}_2, \bar{r}_2) \wedge \bar{x}_2 \in [-\alpha, \alpha]]$, where $\bar{E}_2 = \frac{g^b}{E\bar{E}_1} \mod n$.*

U computes $\tilde{E} = E/g^a \mod n$, $\bar{E} = g^b/E \mod n$; sets $\tilde{x} = x - a$ and $\bar{x} = b - x$; computes $\tilde{x}_1 = \lfloor\sqrt{x-a}\rfloor$, $\tilde{x}_2 = \tilde{x} - \tilde{x}_1^2$, $\bar{x}_1 = \lfloor\sqrt{b-x}\rfloor$, $\bar{x}_2 = \bar{x} - \bar{x}_1^2$; generates random $\tilde{r}_1$ and $\tilde{r}_2$ in $[-2^\sigma n + 1, 2^\sigma n - 1]$ s.t. $\tilde{r}_1 + \tilde{r}_2 = r$, and similarly $\bar{r}_1$, $\bar{r}_2$ s.t. $\bar{r}_1 + \bar{r}_2 = -r$; computes the commitments $\tilde{E}_1 = E(\tilde{x}_1^2, \tilde{r}_1)$, $\tilde{E}_2 = E(\tilde{x}_2, \tilde{r}_2)$, $\bar{E}_1 = E(\bar{x}_1^2, \bar{r}_1)$, and $\bar{E}_2 = E(\bar{x}_2, \bar{r}_2)$; and executes the proofs of knowledge listed in the above definition. We denote the above proof of knowledge by $PK[x, r : E = E(x, r) \wedge x \in [a - \alpha, b + \alpha]$.

The last cryptographic building block we need is the verifiable ElGamal encryption of an exponent.

Definition 14 (Verifiable ElGamal encryption of an exponent). *Assume U holds a secret r, and has published the value $\omega = \chi^r$. Here χ is a generator of a group $\mathcal{F}$ of order n, where n may be prime or composite, and $0 < r < n$. We assume that the DDH assumption holds in $\mathcal{F}$. It is possible for U to prove in zero-knowledge that a pair $(A = r^{-1}y^a, B = g^a) \mod n$, is an ElGamal encryption under public key y of the exponent of ω to basis χ.*

We denote it by: $PK[r : \omega = \chi^r \wedge A = r^{-1}y^a \wedge B = g^a]$. The proof can be found in [28], and we repeat it here for convenience. For i in $\{1, \ldots, \nu\}$, U generates random t_i, and computes $g_i = g^{t_i}$, $y_i = y^{t_i}$, and $\omega_i = \chi^{y_i}$. Next, U computes

$$c = \mathcal{H}(\chi \,||\, \omega \,||\, A \,||\, B \,||\, g_1 \,||\, \omega_1 \,||\, \cdots \,||\, g_\nu \,||\, \omega_\nu). \tag{9}$$

Next, U computes $s_i = t_i - c_i a$, where c_i stand for the i^{th}-bit of c. The proof consists of c and s_i, $i = 1, \ldots, \nu$. In order to verify, V recomputes $g_i = g^{s_i} B^{c_i}$, $y'_i = y^{s_i} A^{c_i}$, and $\omega_i = \omega^{y'_i}$, and checks that (9) holds. The rationale for the proof is that, when $c_i = 0$, the verifier checks that g_i and ω_i are correctly constructed; when $c_i = 1$, the verifier checks that (A, B) is the ElGamal Encryption of the discrete logarithm of ω to basis χ, provided that g_i and ω_i are constructed correctly. If the statement were false, U could pass only one of the verification equations, for each i. In the random oracle model, the probability of U successfully proving a false statement is $2^{-\nu}$.

B Security analysis

Before the introduction of a formal model of security of group signature schemes [5], it was common practice to prove the security of a scheme by showing that it would satisfy the various informal requirements listed in section §2. Of course, it is impossible to be sure that any such list is complete, and in fact early schemes failed to identify the need for resistance against coalition/collusion attacks (see [4] for a discussion about this issue).

Thanks to the formal model, a clearer picture about the complete security requirements of group signatures has now emerged; a scheme proven to satisfy "full anonymity" and "full traceability" can be trusted to provide security – at least as long as the particular computational assumptions underlying the cryptographic primitives (digital signatures, encryption, proofs-of-knowledge) used in the scheme hold up. Unfortunately it is challenging to provide a proof in the new model. The only example of such a proof is for the general construction given in [5] itself. While that construction shares similar design principles with ours, their proof works in a different model of computation. In particular, security conditions for the proofs-of-knowledge are defined in the Common Reference String model. On the other hand, the primitives used in our scheme are provably secure only in the Random Oracle Model (ROM). Indeed, ALL primitives based on discrete logarithms (which we must use if the scheme is to be functionally trapdoor-free) are only proven secure in the ROM model. Thus, in order to provide a formal security proof, we would have to adapt the framework of [5] to the ROM setting. We plan to pursue this direction in a future journal publication of this work. In this section we will give some arguments on how such a formal proof would work for our scheme. Before we proceed, however, we would like to remark that it is simple to prove the security of our scheme by going over each property in §2. In fact, the only requirement that is not clear from the construction is security against coalition attacks. Equivalently, it is not obvious whether group membership certificates are unforgeable even if some (or all) the group members conspire to share their secrets, because our scheme uses a new, modified Nyberg-Rueppel signature for certificate issuance. Indeed, certificate unforgeability is equivalent to the property that this signature be existentially unforgeable under active attacks. We now prove the security of the modified Nyberg-Rueppel.

Proposition 1 (Forking lemma for modified Nyberg-Rueppel). *Let A be an adversary which attempts to forge modified Nyberg-Rueppel signatures on messages issued under the public key $y = g^x$. Assume A has a non-negligible probability of success, as computed over the sample space of messages m, random tapes r and random bases g_1. Then A has a non-negligible probability of success of computing relative discrete logarithms in the group $\mathcal{G}$.*

Proof. Since A has non-negligible success probability over sample triples (m, r, g_1), a standard product sample argument can be used to show that for a non-negligible set of choices of values for the first two components, (i.e., values for the message m and random tape r) the algorithm has a non-negligible probability of success over choices for the remaining component (the basis g_1 in $\mathcal{G}$). Now consider the following reduction to the relative discrete logarithm problem. Given two arbitrary values g_2 and g_3 in $\mathcal{G}$, choose (with non-negligible probability of success) values m and r such that A can forge signatures on message m with random tape r for a non-negligible subset of bases g_1 in $\mathcal{G}$. Then, with non-negligible probability, both g_2 and g_3 will belong to that subset. But this implies that A can compute a pair (m, r) and values s and s' such that $g_2^m = ry^r g^s$ and $g_3^m = ry^r g^{s'}$. Dividing the equations, we get $\left(\frac{g_2}{g_3}\right)^m = g^{s-s'}$, which implies $\mathrm{dlog}_{g_3}(g_2) = \frac{s-s'}{m}$.

Proposition 2. *The modified Nyberg-Rueppel signature scheme, as a signature scheme on short messages, is existentially unforgeable under chosen message attacks, if the discrete logarithm problem is hard in $\mathcal{G}$.*

Proof. Since we are considering short messages only, there is no need to use the random oracle model. The previous proposition reduces such forgeries to the hardness of discrete logarithm computations. Of course the reduction is “loose” by a factor of 2: If you can forge signatures with probability at least p, the probability of successful computation of discrete logarithms is at least p^2.

Notice that the `SIGN` protocol is a Schnorr-type signature scheme, in the sense that it binds all the signature parameters in a single hash computation, and the signer’s secret is a discrete logarithm. In fact, the signature itself includes a proof of knowledge of discrete logarithm of the signer’s public key with respect to a fixed basis (also tied in the hash computation). Such constructions can be proven secure in the random oracle model [26]. In other words, individual group member signatures are secure against existential forgery by adaptively chosen message attacks.

Consider now the anonymity game. The attacker has corrupted all secret keys of all group members. It is allowed to query an `OPEN` oracle for opening arbitrary valid signatures. After possibly some interaction with the oracle it can choose two identities i_0 and i_1 and a message m. The adversary challenge σ is then a valid group signature on m that is known to have been issued by either i_0 or i_1 with equal probability. The adversary is allowed to further interact with the `OPEN` oracle, but is now restricted not to query the oracle with the challenge (m, σ).

Claim (Reduction to passive attacks). Assume that the group member signature is secure against existential forgery by adaptively chosen message attacks, and that it implements a sound zero-knowledge proof of knowledge of a certificate on a pseudonym and its associated secret. If there is an efficient attacker that, upon interacting with an `OPEN` oracle, can guess the identity of the signer on the challenge with non-negligible advantage over a random guess, then there is an efficient attacker *without access to an* `OPEN` *oracle* that can similarly guess the identity of the signer with non-negligible advantage over a random guess.

Argument. The idea for the proof is as follows: Let A_0 be an attacker with access to the oracle, and A_1 an attacker that has full access to ALL the group members for all time – i.e., it is able to see the internal state of the group members that lead to computation of group signatures (except that he cannot see the computation of the challenge). However, A_1 is not given access to the oracle. Let Q be some query made by A_0 to the oracle. If the oracle accepts and decrypts the message, then it means that either the query included a valid group member signature or that the proof of knowledge was forged. Since we assume the proof of knowledge is sound, this second case can only happen with negligible probability. Therefore, with overwhelming probability the adversary either submitted a signature previously computed by some group member, or A_0 constructed a new signature using his knowledge of one of the group member's secret key. In the latter case, A_0 already knew what the response of the oracle would be and could have continued the computation without need of the query Q. In the former case, A_0 does acquire knowledge through the interaction, but this knowledge is available to A_1 through its access to the internal state of all group members through time. So with overwhelming probability we can reduce a computation of A_0 to one of A_1.

Claim (Full anonymity). Under the assumptions of the previous proposition, and assuming further that the signature of knowledge composes well with ElGamal encryption, our group signature scheme provides full anonymity.

Argument. Since the identity of the signer is encrypted using ElGamal, which is semantically secure, it is safe against passive attacks on the encryption scheme, as long as the proofs of knowledge compose well with it. But from the previous proposition, we know that an adversary does not gain any significant advantage from accessing the `OPEN` oracle, i.e., from staging active attacks against the encryption scheme.

Remark 2. Such a result may sound surprising, specially in view of the proof in [5], which implies that in order for a group signature scheme to be secure in the formal model it is *required* that the cipher used be secure against chosen ciphertext attacks, whereas our scheme uses ElGamal, which is only semantically secure. Still, in light of results such as [27], it is at least conceivable that semantic security is sufficient if the proofs of knowledge are *non-malleable*.

Moreover, our scheme can be easily modified to use Cramer-Shoup encryption instead of ElGamal. This will only require adding the authenticating tags to each of the two ElGamal encrypted pairs (Y_1, Y_2) and (R_1, R_2) and verifying such tags during signature verification as well as before decrypting within the signature opening algorithm. (Notice that the authenticating tags can be shown well-constructed without requiring knowledge of the Cramer-Shoup scheme's private keys.)

The second property we should prove is the full traceability.

Claim (Full traceability). Under the assumptions of the previous claims, and using the fact that the modified Nyberg-Rueppel signature is unforgeable under chosen message attacks, our group signature scheme is fully traceable.

Argument. To prove such a claim one must show the impossibility of an adversary to produce a signature that, when opened, reveals either an invalid pseudonym or a valid pseudonym whose secret is unknown to the attacker. In each case, the attacker must either be capable of forging the proof of knowledge of a certificate on a pseudonym and associate secret, or must be able to produce certificates for new, invalid users. (Forging a new certificate for a valid, uncompromised user would NOT suffice, for the adversary would still have to prove knowledge of the pseudonym secret.) The latter case is not possible because the modified Nyberg-Rueppel is existentially unforgeable under chosen message attacks. The former case would violate the assumption that the Schnorr signature implements sound proofs-of-knowledge.

C An alternative construction in the RSA ring

In this appendix we briefly describe another possible realization of the scheme. Much of the notation and procedures are the same as in section 4. The shared parameters are chosen differently. We define $\mathcal{G}$ to be the group of quadratic residues in the RSA ring generated by a composite modulus which is a product of safe primes. Namely, a trusted party generates two safe primes p, q, and publishes $n = pq$. After constructing a proof that n is formed correctly, the third party may forget its factorization, as it is not needed for the scheme. The group $\mathcal{F}$ is chosen as a group of order n. For that, one searches for a prime $\hat{p}$ so that $\hat{p} = mn + 1$, where m is a small number. One then sets $\mathcal{F}$ to be the subgroup of m-powers in the group $\mathbf{Z}^*_{\hat{p}}$. The group-specific parameters are the same.

The `JOIN` protocol is little changed. There are no restrictions on the value of $r = I_U g^{-k} \mod n$, where k is chosen in the interval $[-2^{\tau+2\kappa}, 2^{\tau+2\kappa} - 1]$; as before, κ stands for the bitlength of $|\mathcal{G}|$. The terms a, b, and s cannot be reduced modulo the unknown order of $\mathcal{G}$, which is unknown.

The `SIGN` protocol can be considerably simplified. There is no need for an extra commitment in a group of unknown order, as the order of the group $\mathcal{G}$ is itself unknown. Moreover, there is no need to prove that the r in the commitment E_1 is bounded in a certain interval, as a cheating U could not find a value that reduces to different values r_1 mod n and $r_2 \mod \phi(n)$ while satisfying the signature equation, because $\phi(n)$ is unknown to U.

Protocol `OPEN` is unchanged from the previous case.

Table 5. Shared and group specific parameters.

Shared parameters

Security parameters δ, ϵ, σ_1, σ_2, τ (integers);

Secure hash function $\mathcal{H}(\cdot) : \{0,1\}^* \longrightarrow \{0,1\}^\tau$;

n, a composite integer, the product of safe primes;

$\hat{p}$, a prime satisfying $\hat{p} = mn + 1$, where m is small;

$\mathcal{G} = \{x \in \mathbf{Z}_n^* : \exists\, a \in \mathbf{Z}_n^*$ s.t. $x \equiv a^2 \mod n\}$;

$\mathcal{F} = \{x \in \mathbf{Z}_{\hat{p}}^* : \exists\, a \in \mathbf{Z}_{\hat{p}}^*$ s.t. $x \equiv a^m \mod \hat{p}\}$;

P, an (optional) proof that n is a product of safe primes;

g, g_1, and g_2, generators of $\mathcal{G}$;

P', an (optional) proof that g, g_1, and g_2 are quadratic residues.

Group-specific parameters

$\mathcal{S}$, a string including y and y_2;

CA's signature CERT$_{CA}(\mathcal{S})$.

Table 6. The `JOIN` protocol.

$U \longrightarrow GM : J_U = I^m \mod n$

$GM \longrightarrow U : a,\ b \in [-2^{\tau/2+\kappa}, 2^{\tau/2+\kappa} - 1]$

$U \longrightarrow GM : Sig_U(I_U = J_U^a g_1^b \mod n,\ PK[u : I_U = g_1^u])$

$GM \longrightarrow U : r = I_U g^{-k} \mod n,$

$\qquad s = -xr + k \in [-2^{2\kappa+\tau+1}, 2^{2\kappa+\tau+1} - 1]$

Table 7. The `SIGN` protocol.

Proof arguments:

Y_1, Y_2, R_1, R_2, χ, E_1.

Signature of knowledge:

$$SPK[u, \ell', \ell, r, s, t : Y_1 = g_1^u g^{\ell'} \ \wedge\ Y_2 = g_2^{\ell'}$$
$$\wedge\ E_1 = E_1(r, 0) = \chi^r \ \wedge\ R_1 = r^{-1} y_2^{\ell} \ \wedge\ R_2 = g_2^{\ell}$$
$$\wedge\ E_2 = Y_1 R_1 = E_2(r, s, t) = y^r g^s y_2^t \ \wedge\ Y_2 R_2 = g_2^t](\mathcal{M})$$