In *Advances in Cryptology — EUROCRYPT '97*

Efficient and Generalized Group Signatures

Jan Camenisch

Department of Computer Science
ETH Zurich
CH-8092 Zurich, Switzerland
camenisch@inf.ethz.ch

Abstract. The concept of group signatures was introduced by Chaum et al. at Eurocrypt '91. It allows a member of a group to sign messages anonymously on behalf of the group. In case of a later dispute a designated group manager can revoke the anonymity and identify the originator of a signature. In this paper we propose a new efficient group signature scheme. Furthermore we present a model and the first realization of generalized group signatures. Such a scheme allows to define coalitions of group members that are able to sign on the group's behalf.

1 Introduction

In [6] Chaum and van Heyst proposed a new type of signature scheme for a group of entities, called *group signatures*. Such a scheme allows a group-member to sign a message on the group's behalf such that everybody can verify the signature but no one can find out which group member provided it. However, there is a trusted third party, called the group manager, who can in case of a later dispute reveal the identity of the originator of a signature. The group manager can either be a single entity or a number of coalitions of several entities (e.g. group members). This concept can be generalized to allow defined subsets of all group members to jointly sign a message on behalf of the group.

An application of group signature schemes is a company needing a corporate identity. Members of the company can sign contracts with customers such that a customer does not know who actually signed the contract. If a problem with a particular contract occurs later, the company can find out which employee is to be held responsible.

1.1 Related Work

There exist several other group-oriented concepts for signature schemes. The most important ones are multi-signatures [3,9,15] and proxy signatures [14]. Multi-signatures can be seen as generalized group signature without the ability of "opening" signatures, while proxy signatures are group signatures that do not provide anonymity.

Solutions for group signature schemes were first presented in [6] and later in [7]. We discuss these schemes briefly. In [6] four different schemes were proposed. Three of them require the group manager to contact each group member in order to find out who signed a message. These scheme provide computational anonymity, whereas the forth scheme provides information theoretical anonymity. For two of the schemes it is not possible to add a new member after the scheme is set up (including the scheme giving information theoretical anonymity). In none of the proposed schemes it is possible to distribute the functionality of the group manager efficiently.

Later, Chen and Pedersen proposed two new schemes in [7] providing information theoretical anonymity and computational anonymity, respectively. These schemes allow to add new members after the setup of the system and to distribute the functionality of the group manager. They are based on proofs of knowledge of one out of several discrete logarithms, each being the secret key of a group member. The proofs they apply have the special property that when knowing all secret keys, one can tell which one was used in the proof. To realize the group manager's ability to open signatures, two such proofs of knowledge must be used in parallel, where for one the manager is told the secret keys of all group members. However, this solution has the drawback that the group manager can falsely accuse a group member of having signed a message: she therefore computes one of the proofs of knowledge using the known secret key of the member she wants to accuse. This risk can be weakened, but not prevented, by sharing the functionality of the group manager. To solve this problem, some kind of disavowal protocol would be needed.

1.2 Our Results

In this paper we propose a group signature scheme where the manager cannot falsely accuse group members (even if she is also a group member) and which is also more efficient than all the previously proposed schemes. Furthermore, this scheme is extended to a generalized group signature scheme that is also presented. In both schemes, the functionality of the group manager can be shared such that the identity of a signer can still be revealed efficiently. Both schemes allow to add (or remove) group members after the initial setup. They provide computational anonymity which we believe is satisfactory because the security of the signature scheme itself is also computational (as is the case for all signature schemes).

The paper is structured as follows. In the next section we formalize the concept of (generalized) group signatures schemes. The preliminaries are given in Section 3, and in Section 4 we formalize different protocols for proving knowledge about discrete logarithms. This formalization allows a compact and comprehensive description of the new group signature schemes in Section 5. An example of a generalized group signature scheme is also given. In Section 6 we present extensions to the scheme, such as distributing of the functionality of the group manager.

2 Defining Group Signature Schemes

In this section we define the generalized concept of group signature schemes. Let $\mathcal{P} = \{P_1, ..., P_n\}$ be a set of *group members* and M be a designated entity, called *group manager*. The set of all authorized coalitions of group members $\Gamma \subseteq 2^{\mathcal{P}}$ is called *authority structure*. The structure must be *monotone*, i.e., for two sets $\mathcal{S}$ and $\mathcal{S}' \in 2^{\mathcal{P}}$, if $\mathcal{S} \in \Gamma$ and $\mathcal{S}' \supseteq \mathcal{S}$, then also $\mathcal{S}' \in \Gamma$. If $\Gamma = \{\{P_1\}, \{P_2\}, ..., \{P_n\}\}$, we call the group signature scheme *simple* (this is the only authority structure we do not require to be monotone).

A (generalized) group signature scheme for $\mathcal{P}$ and M with respect to Γ consists of four procedures:

`setup`**:** On input Γ this multi-party protocol between all members in $\mathcal{P}$ and M outputs the group public key $\mathcal{Y}$, to each group member $P_i \in \mathcal{P}$ a secret key x_i, and an *opening secret key* ω to the group manager M.

`sign`**:** On input a message m, the group public key $\mathcal{Y}$, the structure Γ , the coalition $\mathcal{S}$, and the corresponding secret keys x_i, this multi-party protocol between members in some $\mathcal{S} \in \Gamma$ outputs a signature s on m.

`verify`**:** On input a message m, the group public key $\mathcal{Y}$, the structure Γ, and a signature s, this algorithm outputs `yes` if and only if the signature is correct.

`open`**:** On input a message m, the group public key $\mathcal{Y}$, the structure Γ, a signature s and the opening secret key ω, the algorithm outputs $\mathcal{S} \in \Gamma$ (i.e., the set of group members that signed m) and a proof that $\mathcal{S}$ indeed signed m.

In the procedures being multi-party protocols the private inputs of the different parties must of course remain secret during and after the execution. The requirement that `open` also outputs a proof is often omitted but is essential if the trust to be put into the group manager is to be minimized.

The group publishes its public key $\mathcal{Y}$, the authority structure Γ, and some system parameters. A group signature scheme must satisfy the following properties:

1. Only authorized coalitions $\mathcal{S}$ of group members, i.e., $\mathcal{S} \in \Gamma$, can sign. The correctness of a signature can be publicly verified using $\mathcal{Y}$ and Γ.
2. It is not possible to find out which coalition $\mathcal{S} \in \Gamma$ signed a message (anonymity) or whether two different signature are signed by the same coalition (unlinkability).
3. In case of dispute, the group manager can open a signature, i.e., find out which coalition signed a message, by running the algorithm `open`
4. The group manager must only be involved in the procedures `setup` and `open`.

These properties are demanded in all previous papers and further properties follow from them, for instance the property that a coalition must not be able so sign in the name of another coalition. However, the following natural properties should also be satisfied by a group signature scheme. The property 5 was formulated as an open problem in [6] and achieved first in [7].

5. To decrease the trust to put in the group manager, it should be possible to distribute her role among a set of entities such as the members of the group.

6. The group manager is only trusted not to open signatures at will and is not trusted with regard to anything else.

When considering the efficiency of a scheme, the following parameters are of particular interest: the amount of computation in the algorithms `setup`, `sign`, `verify`, and `open`, the size of the group public key, and the length of signatures. The possibility of adding (or removing) new group members after the initial setup falls also in this category, namely in the efficiency of the algorithms `setup` (i.e., whether it is possible to run it incremental or not).

3 Preliminaries

In this section a variation of the ElGamal encryption scheme is described. This variation is used as a building block for both group signature schemes we present. We give a formal definition of secret sharing schemes and describe an example. Secret sharing is used for constructing the generalized group signature scheme.

3.1 ElGamal Encryption Variant

The original encryption scheme was proposed by ElGamal [10]. In this paper we interchange the role of the base and the public key and get the following scheme with the same security properties. Let G be a finite cyclic group of prime order q and let $g \in G$ be a generator of G such that computing discrete logarithms to the base g is infeasible. In order to encrypt a message m for an entity with public key $z = g^x$, one first chooses α randomly in $\mathbb{Z}_q$ and then encrypts m by computing the pair $(A, B) = (z^\alpha, g^\alpha m)$. The entity knowing the secret key x can decrypt the message m by calculating

$$\frac{B}{A^{x^{-1}}} = \frac{g^\alpha m}{g^{x\alpha x^{-1}}} = m \ .$$

3.2 Secret Sharing

A secret sharing scheme is a method for distributing a *secret* σ among a set of n participants $\mathcal{P} = \{P_1, ..., P_n\}$. Each participant P_i obtains a *share* ς_i of the secret σ such that every *qualified* subset $\mathcal{S}$ of $\mathcal{P}$ can reconstruct σ by using algorithm Υ_Γ. The following must hold:

$$\forall \mathcal{S} \in \Gamma : \quad \sigma = \Upsilon_\Gamma(\mathcal{S}, \{\varsigma_i | P_i \in \mathcal{S}\}) \ .$$

The union of all qualified subsets $\Gamma \subseteq 2^{\mathcal{P}}$ is called the *access structure* and is required to be monotone. A common special case is a *threshold* structure where for a threshold k the access structure Γ is defined as $\{\mathcal{S} \subseteq 2^{\mathcal{P}} \mid |\mathcal{S}| \geq k\}$. Every access structure Γ has a natural *dual* access structure Γ^*:

$$\mathcal{S} \in \Gamma^* \iff \bar{\mathcal{S}} \notin \Gamma ,$$

where $\bar{\mathcal{S}}$ denotes the complement of $\mathcal{S}$ in $\mathcal{P}$. If Γ is monotone, then Γ^* is also monotone and we have $(\Gamma^*)^* = \Gamma$. If Γ is a threshold structure, then so is Γ^*. A secret sharing scheme is called *perfect* if the participants forming a non-qualified subset of $\mathcal{P}$ are not able to obtain any information on σ. A secret sharing scheme is *ideal* if it is perfect and the secret and the shares are of the same length.

To construct the shares for a given secret σ, we employ a nonstandard algorithm that, given the shares of a non-qualified set, outputs the shares for the remaining participants. Formally, the algorithm Ψ which takes as inputs the access structure Γ, a non-qualified set of participants $\mathcal{N} \notin \Gamma$, the set $\{\varsigma_i | P_i \in \mathcal{N}\}$ of their shares, and the secret σ and outputs the set $\{\varsigma_j | P_j \in \bar{\mathcal{N}}\}$, i.e.,

$$\Psi(\Gamma, \mathcal{N}, \{\varsigma_i | P_i \in \mathcal{N}\}, \sigma) = \{\varsigma_j | P_j \in \bar{\mathcal{N}}\} .$$

The algorithm relies on the fact that given the secret and the shares of a non-qualified set participants $\mathcal{N}$, it is possible to construct a complete set of shares.

As an example of a threshold secret sharing scheme with n participants and threshold k, we present Shamir's scheme [18]. A secret σ (an element of a finite field $GF(q)$, with $q > n$) is shared by randomly choosing the coefficients $\alpha_1, ..., \alpha_{k-1} \in GF(q)$ of the polynomial

$$f(X) = \alpha_{k-1} X^{k-1} + ... + \alpha_1 X + \sigma \pmod{q} .$$

The share for participant P_i is then calculated as $\varsigma_i = f(p_i)$, where p_i is a publicly known element of $GF(q)$ associated with participant P_i, e.g. $p_i = i$. Given k or more shares the function f and thus σ can be found by Lagrange interpolation on the points (p_i, ς_i). This scheme is ideal.

4 Proving Knowledge of Discrete Logarithms

In this section we define and formalize the building blocks for our scheme. They are based on different interactive proofs of knowledge of discrete logarithms that are made non-interactive using the techniques of [17]. To avoid confusion with the terminology of non-interactive proofs of knowledge, we call these building blocks *signatures of knowledge*.

The algebraic setting is as follows. Let G be a finite cyclic group of prime order q and let $g, g_1, ..., g_n \in G$ be generators of G such that computing discrete logarithms to any of the bases is infeasible. A *public key* y_i is constructed by computing $y_i = g^{x_i}$ with the *secret key* x_i chosen at random from $\mathbb{Z}_q$. The symbol $\|$ denotes the concatenation of two binary strings (or of the binary representation of group elements and integers). Finally, let $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\ell$ ($\ell \approx 128$) denote a one-way hash function.

The first building block we define is a signature of knowledge of the discrete logarithm of a public key y to the base g.

Definition 1. A pair (c, s) satisfying

$$c = \mathcal{H}(g \| y \| g^s y^c \| m)$$

is a signature of knowledge of the discrete logarithm of a group element y to the base g for the message m and is denoted by $SKDL(g, y, m)$.

Basically, such a signature of knowledge is a Schnorr signature (see [17]) with a slightly different argument to the hash function. A $SKDL$ can be computed only if the secret key x is known, by choosing r at random from $\mathbb{Z}_q$ and computing c and s according to

$$c = \mathcal{H}(g \| y \| g^r \| m)$$

and

$$s = r - cx \pmod{q}.$$

The values g^r, c, and s are often called *commitment*, *challenge*, and *response*, respectively, although the "proof" is non-interactive. If the context is clear then the hashing of bases and public keys could be omitted.

Another building block we use is a signature of knowledge of the discrete logarithm of one out of several public keys y_i without revealing which one. Such proof-systems were first introduced in [8].

Definition 2. A $2n$-tuple $(c_1, ..., c_n, s_1, ..., s_n)$ satisfying

$$\sum_{i=1}^{n} c_i = \mathcal{H}(g \| y_1 \| ... \| y_n \| g^{s_1} y_1^{c_1} \| ... \| g^{s_n} y_n^{c_n} \| m) \pmod{q}$$

is a signature of knowledge of the discrete logarithm of one group element out of the list $\{y_1, ..., y_n\}$ to the base g for the message m and is denoted by $SKDL\left[{n \atop 1}\right](g, y_1, ..., y_n, m)$.

A $SKDL\left[{n \atop 1}\right](g, y_1, ..., y_n, m)$ can only be given if at least one of the secret keys is known. We now show how to compute such a signature. Assume that the known secret key is x_1. The prover chooses $r, s_2 ..., s_n, c_2, ..., c_n$ randomly in $\mathbb{Z}_q$ and computes $t_1 = g^r$ and $t_i = g^{s_i} y_i^{c_i}$ for $i = 2, ..., n$. Then he computes c_1 and s_1 according to

$$c_1 = \mathcal{H}(g \| y_1 \| ... \| y_n \| t_1 \| ... \| t_n \| m) - \sum_{i=2}^{n} c_i \pmod{q}$$

and

$$s_1 = r - x_1 c_1 \pmod{q}.$$

The prover has thereby computed $SKDL\left[{n \atop 1}\right](g, y_1, ..., y_n, m) = (c_1, ..., c_n, s_1, ..., s_n)$.

The idea behind this is the fact that a *SKDL* can be forged if the challenge c is known before the computation of the commitment t. The verification condition of $SKDL\left[{n \atop 1}\right]$ is a linear equation over the c_i's and therefore all but one c_i can be chosen before computing the commitments. It follows that at least for one y_i the discrete logarithm must be known and one of the partial *SKDL*'s must be true.

In [8] such proof systems were generalized to proof systems for proving the knowledge of all discrete logarithms of one out of several defined subsets of the set of public keys $\mathcal{Y} = \{y_1, ..., y_n\}$ without revealing any further information. Formally, let Γ denote a monotone set of subsets of $\mathcal{Y}$, i.e., $\Gamma \subseteq 2^{\mathcal{Y}}$. By combining n signatures of knowledge $SKDL(g, y_i)$ and a secret sharing system with access structure Γ^*, it is possible to construct a system for proving the knowledge of the discrete logarithms of all $y_i \in \mathcal{S}$ for some $\mathcal{S} \in \Gamma$, without saying which subset $\mathcal{S}$.

Definition 3. A $2n$-tuple $(c_1, ..., c_n, s_1, ..., s_n)$ satisfying

$$\forall \mathcal{S}' \in \Gamma^* : \qquad \mathcal{H}(g\|y_1\|...\|y_n\|g^{s_1}y_1^{c_1}\|...\|g^{s_n}y_n^{c_n}\|m) = \Upsilon_{\Gamma^*}(\mathcal{S}', \{c_i \mid y_i \in \mathcal{S}'\})$$

is a signature of knowledge of the discrete logarithm of all $y_i \in \mathcal{S}\ \{y_1, ..., y_n\}$ to the base g for some $\mathcal{S} \in \Gamma$ for the message m. Such a signature is denoted by $SKDL[\Gamma](g, y_1, ..., y_n, m)$.

This signature system is similar to the one in Definition 2; here the secret sharing scheme implies conditions on the (partial) challenges c_i by interpreting them also as shares, whereas in Definition 2 we have only one condition (i.e., a linear equation) on the challenges. If the challenges and the shares do not have the same domain, a mapping must be introduced (for further technical details see [8]). Let us show how such a signature of knowledge $(c_1, ..., c_n, s_1, ..., s_n)$ can be computed. Assume that $x_1, ..., x_j$ are the known secret keys and that $\mathcal{S} = \{P_1, ..., P_j\} \in \Gamma$. The prover chooses $r_1, ..., r_j, s_{j+1} ..., s_n, c_{j+1}, ..., c_n$ randomly in $\mathbb{Z}_q$ and computes

$$\begin{aligned} \sigma &= \mathcal{H}(g\|y_1\|...\|y_n\|g^{r_1}\|...\|g^{r_j}\|g^{s_{j+1}}y_{j+1}^{c_{j+1}}\|...\|g^{s_n}y_n^{c_n}\|m) \ , \\ \{c_1, ..., c_j\} &= \Psi(\Gamma^*, \{P_{j+1}, ..., P_n\}, \{c_{j+1}, ..., c_n\}, \sigma) \ , \text{and} \\ s_k &= r_k - c_k x_k \pmod q \qquad \text{for } k = 1, ..., j \ . \end{aligned}$$

For the definition of the function Ψ see Section 3.2.

Another primitive often used in cryptography (e.g. [5]) is a signature that the logarithms of two group elements with respect to two different bases are the same. Such a signature also implies the knowledge of these logarithms.

Definition 4. A pair (c, s) satisfying

$$c = \mathcal{H}(h\|g\|z\|y\|h^s z^c\|g^s y^c\|m)$$

is signature of equality of the discrete logarithm of the group element z with respect to the base h and the discrete logarithm of the group element y with respect to the base g for the message m. It is denoted by $SEQDL(h, g, z, y, m)$.

This signature of equality can be seen as two parallel signatures of knowledge $SKDL(h, z, m)$ and $SKDL(g, y, m)$ where the exponent for the commitment, the challenges, and the responses are the same. By using several $SKEQ$ in parallel and implying conditions on their commitments (similar as in the Definitions 2 and 3), one obtains the signature systems $SEQDL[\binom{n}{1}](h, g, z_1, y_1, ..., z_n, y_n, m)$ and $SEQDL[\Gamma](h, g, z_1, y_1, ..., z_n, y_n, m)$, respectively.

Our last building block are signatures of knowledge of a representation. The respective proof systems were first introduced in [4]. Let $y = \prod_{i=1}^{n} g_i^{x_i}$ for some $x_1, ..., x_n \in \mathbb{Z}_q$.

Definition 5. A $(n+1)$-tuple $(c, s_1, ..., s_n)$ satisfying

$$c = \mathcal{H}(g_1 \| ... \| g_n \| y \| y^c \prod_{i=1}^{n} g_i^{s_i} \| m)$$

is a signature of knowledge of a representation of a group element y with respect to the bases $g_1, ..., g_n$ for the message m. It is denoted by $SKREP(g_1, ..., g_n, y, m)$.

We now show how this signature of knowledge of a representation can be calculated from $x_1, ..., x_n$. The prover chooses $r_1, ..., r_n$ at random from $\mathbb{Z}_q$, computes $t = \prod_{i=1}^{n} g_i^{r_i}$,

$$c = \mathcal{H}(g_1 \| ... \| g_n \| y \| t \| m) \pmod{q},$$

and

$$s_i = r_i - x_i c \pmod{q} \qquad \text{for } i = 1, ..., n$$

and thus obtains an $SKREP(g_1, ..., g_n, y, m) = (c, s_1, ..., s_n)$. If the bases g_i are chosen in a random or pseudo-random manner, computation of another than the known representation is believed to be as hard as the discrete logarithm problem and is called the *representation problem*. For further discussion see [4].

5 Construction of a Group Signature Scheme

In this section an efficient simple group signature scheme and a generalized group signature scheme are proposed. They are based on the signature systems $SEQDL[\binom{n}{1}]$ and $SEQDL[\Gamma]$, respectively. These underlying systems already fulfill the properties of a group signature scheme except those related to the group manager's capability of "opening" a signature.

In the following we present efficient solutions to achieve the missing properties by using a variation of the ElGamal encryption scheme (see Section 3) and the techniques discussed in the previous section. The solutions further allow a simple way of distributing the functionality of the group manager, as will be shown in Section 6.

5.1 An Efficient Simple Group Signature Scheme

The algebraic setting is the same as in Section 4. In addition, let $z = g^{\omega}$ denote the public key of the group manager and ω her secret key. Each group member P_i chooses his secret key x_i randomly in $\mathbb{Z}_q$ and computes the public key $y_i = g^{x_i}$. The group's public key consists the list of all members' public keys $\mathcal{Y} = (y_1, ..., y_n)$ and is published together with the manager's public key and the system parameters.

The idea behind the scheme is that in order to sign a message, a group member encrypts one of the public keys of $\mathcal{Y} = \{y_1, ..., y_n\}$ with the public key of the group manager and proves that

– he encrypted one of the y_i's and that
– he actually knows the discrete logarithm of the encrypted key.

From this follows, that the group member must have encrypted his public key. More formally, to generate a signature of a message m, the group member P_j executes the following steps:

1. choose a randomly in $\mathbb{Z}_q$
2. encrypt y_j by computing $A = z^a$ and $B = y_j g^a$
3. calculate $(c_1, ..., c_n, s_1, ..., s_n) = \mathit{SEQDL}\binom{n}{1}(z, g, A, \frac{B}{y_1}, ..., A, \frac{B}{y_n}, m)$
4. calculate $(\tilde{c}, \tilde{s}) = \mathit{SKDL}(g, B, m)$

The computed group signature is the tuple $(A, B, c_1, ..., c_n, s_1, ..., s_n, \tilde{c}, \tilde{s})$ and can be verified by checking the correctness of $\mathit{SEQDL}\binom{n}{1}(z, g, A, \frac{B}{y_1}, ..., A, \frac{B}{y_n}, m)$ and $\mathit{SKDL}(g, B, m)$.

The first signature assures that (A, B) is the encryption of an element of the list $\mathcal{Y}$ and the second signature guarantees that the signer actually knows the discrete logarithm of the public key encrypted in (A, B). The signer thus proves indirectly his knowledge of the discrete logarithm of an element of $\mathcal{Y}$ and therefore that he is a member of the group $\mathcal{P}$. It can easily be seen that only group members can sign messages.

To open a valid signature the group manager decrypts (A, B) and immediately obtains the public key of the signer. Assume that the group member P_j has signed. By computing the signature of equality

$$\mathit{SEQDL}(g, z, B/(y_j), A, P_j)$$

the group manager can assure that she opened the signature correctly and that indeed P_j has issued this signature.

5.2 A generalized Group Signature Scheme

The system parameters are the same as for the simple group signature scheme. In addition to all public keys and to the system parameters, an authority structure Γ must be published.

The idea of the generalized scheme is similar to the one of the simple scheme. To sign a message m all members of an authorized coalition prove that each of them encrypted an element of $\mathcal{Y} = \{y_1, ...y_n\}$ and that they know the discrete logarithms of the encrypted values. Furthermore, they must also prove that the encrypted elements are all different. The problem with this approach is that the number of encryptions equals the size of the coalition, which should be kept secret. Therefore, the coalition must also encrypt some dummy values in order to provide n encryptions.

More formally, to generate a signature of a message m, the group members forming an authorized set $\mathcal{S} \in \Gamma$ execute together the following steps:

1. – choose $a_1, ..., a_n$, and b_i for all i with $y_i \notin \mathcal{S}$ randomly in $\mathbb{Z}_q$
   – for all $y_j \in \mathcal{S}$ encrypt y_j: $A_j = z^{a_j}$, $B_j = y_j g^{a_j}$
   – for all $y_i \notin \mathcal{S}$ encrypt g^{b_i}: $A_i = z^{a_i}$, $B_i = g^{b_i} g^{a_i}$
2. calculate $(c_1, ..., c_n, s_1, ..., s_n) = SEQDL[\Gamma](z, g, A_1, \frac{B_1}{y_1}, ..., A_n, \frac{B_n}{y_n}, m)$
3. calculate $(\tilde{c}_i, \tilde{s}_i) = SKDL(g, B_i, m \| c_1 \| ... \| c_n \| s_1 \| ... \| s_n)$ for $i = 1, ..., n$

Member P_j must calculate the signature $SKDL(g, B_j, m)$ and also parts of the signature in Step 2 alone in order to hide his secret key from the other members. All other computations should be performed by all group members on their own in order to assure themselves of the correctness of the outcome. The random choices in these common computations must be agreed upon by the group members in advance, for instance by choosing a random string each, committing to the string by hashing it, exchanging these commitments, then exchanging the random strings, and finally taking the XOR of all these random strings. The resulting group signature is the tuple $(A_1, B_1, ..., A_n, B_n, c_1, ..., c_n, s_1, ..., s_n, \tilde{c}, \tilde{s})$ and can be verified by checking the correctness of the signatures of knowledge $SEQDL[\Gamma](z, g, A_1, \frac{B_1}{y_1}, ..., A_n, \frac{B_n}{y_n}, m)$ and $SKDL(g, B_i, m)$ for all i.

The first signature assures that the list $((A_1, B_1), ..., (A_n, B_n))$ contains the encryptions of some $y_j \in \mathcal{Y}$ such that the corresponding P_j's form an authorized coalition. The signatures generated in Step 3 assure that the authorized coalition was really involved, i.e., that the discrete logarithms of the encrypted y_j's are known. Here, the signature of Step 2 is appended to the message in order to bind the two steps together. This prevents the reuse of a *SKDL* in another run of the scheme.

Again, it is easy to see that the group manager can find out which coalition provided the signature by checking the validity of the signature and decrypting all pairs (A_j, B_j). Note that a coalition cannot encrypt a public key of a member $P_i \notin \mathcal{S}$ not participating in the signing because then they could not provide the corresponding signature in Step 3 and therefore the group signature would not be valid. By computing the signatures of equality

$$SEQDL(g, z, B/(y_j), A, P_j)$$

for all P_j having participated in the signing, the group manager can assure that she opened the signature correctly.

Remark. The signature can be made shorter if all $\tilde{c}_i$ are the same, i.e., all signatures $SKDL(g, B_i, m)$ are merged and are verified simultaneously by checking the equation

$$\tilde{c} = \mathcal{H}(g \| B_1 \| ... \| B_n \| g^{\tilde{s}_1} B_1^{\tilde{c}} \| ... \| g^{\tilde{s}_n} B_n^{\tilde{c}} \| m) \ .$$

Of course, the signatures must then be computed in parallel and $\tilde{c}$ calculated accordingly. This choice also binds Steps 2 and 3 together, i.e., the concatenation of the first signature to the message is not needed in this case. This is applied in the following example.

5.3 An Example for a Threshold Group Signature Scheme

In this section we give an example for a generalized group signature scheme with a threshold authority structure. Let k be the minimum number of members that must cooperate in order to sign and let $f(x) = \sum_{i=0}^{k-1} \alpha_i x^i$ denote the polynomial of a secret sharing scheme with threshold k as described in Section 3.2. To generate a signature of a message m, the group members forming an authorized set $\mathcal{S}$, i.e., $|\mathcal{S}| \geq k$, execute the steps below. In Step 2 it is indicated when the calculations must be performed by a specific member of the coalition, whereas in Step 3, all calculations for a specific j must be performed by member P_j for $P_j \in \mathcal{S}$. All other computations should by done by the coalition members on their own using the agreed-on random string.

1. – choose $a_1, ..., a_n$, and b_i for all i with $y_i \notin \mathcal{S}$ randomly in $\mathbb{Z}_q$
   - for all $y_j \in \mathcal{S}$, member P_j encrypts y_j: $A_j = z^{a_j}$, $B_j = y_j g^{a_j}$
   - for all $y_i \notin \mathcal{S}$ encrypt g^{b_i}: $A_i = z^{a_i}$, $B_i = g^{b_i} g^{a_i}$
2. compute $SEQDL[\Gamma](z, g, A_1, \frac{B_1}{y_1}, ..., A_n, \frac{B_n}{y_n}) = (\alpha_0, ..., \alpha_{k-1}, s_1, ..., s_n, m)$:
   - for all $y_j \in \mathcal{S}$, member P_j chooses r_j randomly in $\mathbb{Z}_q$ and calculates $t_{z,j} = z^{r_j}$ and $t_{g,j} = g^{r_j}$
   - for all $y_i \notin \mathcal{S}$ choose r_i and c_i randomly in $\mathbb{Z}_q$ and compute $t_{z,i} = z^{r_i} A_i^{c_i}$ and $t_{g,i} = g^{r_i} (\frac{B_i}{y_i})^{c_i}$
   - $c = \mathcal{H}\big(z \big\| g \big\| A_1 \big\| \frac{B_1}{y_1} \big\| ... \big\| A_n \big\| \frac{B_n}{y_n} \big\| t_{z,1} \big\| t_{g,1} \big\| ... \big\| t_{z,n} \big\| t_{g,n} \big\| m\big)$
   - choose $\alpha_0, ..., \alpha_{k-1}$ such that $f(i) = c_i \pmod q$ for all $i | y_i \notin \mathcal{S}$ and $f(0) = c \pmod q$
   - for all $y_j \in \mathcal{S}$, member P_j computes $s_j = r_j - f(j) a_j \pmod q$
   - for all $y_i \notin \mathcal{S}$ set $s_i = r_i$
3. calculate the combined signatures $SKDL(g, B_i, m) = (\tilde{c}, \tilde{s}_1, ..., \tilde{s}_n)$:
   - for $i = 1, ..., n$ choose $\tilde{r}_i$ randomly in $\mathbb{Z}_q$
   - for $i = 1, ..., n$ compute $\tilde{t}_i = g^{\tilde{r}_i}$
   - $\tilde{c} = \mathcal{H}(g \| B_1 \| ... \| B_n \| \tilde{t}_1 \| ... \| \tilde{t}_n \| m)$

– for $i = 1, ..., n$ compute $\tilde{s}_i = \begin{cases} \tilde{r}_i - \tilde{c}(x_i + a_i) \pmod q & \text{if } y_i \in \mathcal{S} \\ \tilde{r}_i - \tilde{c}(b_i + a_i) \pmod q & \text{if } y_i \notin \mathcal{S} \end{cases}$

The group signature of m is the tuple $(\alpha_0, ..., \alpha_{k-1}, s_1, ..., s_n, \tilde{c}, \tilde{s}_1, ..., \tilde{s}_n)$. Note that instead of all c_i's, the values $\alpha_0, ..., \alpha_{k-1}$ are included in the signature. This makes the signature shorter but not less secure because c and all c_i's are uniquely determined by $\alpha_0, ..., \alpha_{k-1}$.

The group signature can be verified by checking the following equations:

$$\alpha_0 = \mathcal{H}\Big(z \Big\| g \Big\| A_1 \Big\| \frac{B_1}{y_1} \Big\| ... \Big\| A_n \Big\| \frac{B_n}{y_n} \Big\| z^{s_1} A_1^{f(1)} \Big\| g^{s_1} \left(\frac{B_1}{y_1}\right)^{f(1)} \Big\| ... \\ ... \Big\| z^{s_n} A_n^{f(n)} \Big\| g^{s_n} \left(\frac{B_n}{y_n}\right)^{f(n)} \Big\| m\Big)$$

and

$$\tilde{c} = \mathcal{H}(g \| B_1 \| ... \| B_n \| g^{\tilde{s}_1} B_1^{\tilde{c}} \| ... \| g^{\tilde{s}_n} B_n^{\tilde{c}} \| m)$$

where

$$f(x) = \alpha_0 + \alpha_1 x + ... + \alpha_{k-1} x^{k-1} \pmod q.$$

5.4 Security and Efficiency Considerations

Let us shortly discuss the security properties of the generalized group signature scheme (which hold also for the simple scheme).

Non-members cannot sign: If a non-member would be able to forge a group signature, he would also be able to forge Schnorr signature.

Signatures are unlinkable and anonymous: Unlinkability follows from the properties of $SEQDL[\Gamma]$ and from the fact that the y_i's are randomly encrypted, which also guarantees anonymity.

Authorized coalitions cannot sign on behalf of another coalition: Clearly, a coalition cannot sign on behalf of a coalition that includes members that are not included in itself. If a coalition contains an true authorized subset, some members try to make it appear as if they were not involved in the signing. This attack is prevented by the mutually agreed random string.

The group manager cannot falsely accuse members: This is assured by the proof the group manager must provide as evidence in the procedure `open`.

With regard to efficiency, all algorithms except `open` have efficiency linear in the number of group members. The size of the group's public key and the length of signatures are also linear in the number of group members. The algorithm `open` is independent of the group's size (however, finding the identity of a signer given his key requires a look up in a database).

Comparing the second scheme of [7] and our simple group signature scheme, it turns out, that our scheme is approximately four times more efficient in terms of computations of the signer and signatures are about the same ratio shorter. Furthermore, in [7] the algorithm `open` has an efficiency that is linear in the group's size.

6 Extensions

In this section we show how the functionality of the group manager can be shared among several parties (e.g. among the group members) and present a method for reducing the size of the group's public key.

6.1 Sharing the Functionality of the Group Manager

To obtain higher security against fraudulent opening of signatures, the capability of the group manager can be shared among several managers according to an access structure such that only predefined subsets of the managers are able to cooperatively open a signature.

To achieve this, the group manager's secret key ω must be shared among the managers and exponentiation with ω^{-1} must be possible in a distributed manner without leaking information about the shares.

For an access structure with threshold t and k managers, a realization is based on Shamir's secret sharing scheme [18] and Feldman's verifiable secret sharing scheme [11]. A solution to powering with ω^{-1} is described in [12] for the case $t < k/2$ if all managers are honest and for the case $t < k/3$ if up to t of the managers may be actively cheating.

More general access structures are possible if exponentiation with ω^{-1} is avoided, i.e., if signatures are opened as follows. Compute B^{ω}/A and then compare the result with the list $\{y_1^{\omega}, ..., y_n^{\omega}\}$. This list can be (pre-)computed (without revealing ω) during the setup of the system[1]. Then, for instance the monotone circuit construction of Benaloh and Leichter [1] can be applied over $GF(q)$ and powering B with ω can be achieved by multiplying all B^{ω_j}, where ω_j denotes the share of a manager in a qualified set.

6.2 Reducing the Size of the Group's Public Key

The size of the group's public key can be reduced using a technique proposed by Blom for public key distribution [2]. Let Φ be a publicly known generator matrix of an (n, k) MDS code over $\mathbb{Z}_q$. The group's public key now becomes $\{y_1, ..., y_k\}$. The public key of member P_j is then computed as

$$\tilde{y}_j = \prod_{i=1}^{k} y_i^{\phi_{ij}},$$

where ϕ_{ij} denotes the element of Φ in row i and column j. These public keys are then used in Step 2 of the signature generating procedure. The secret keys

[1] The computation of such a list can be avoided if normal ElGamal encryption is used in our group signature scheme. Then the signature systems in Step 2 and 3 must be adjusted: in Step 2 the A_i's instead of the B_i's must be divided by the respective y_i's. and in Step 3 the signatures $\mathrm{SKDL}(g, B_i, m)$ must be replaced by signatures $\mathrm{SKREP}(g, z, B_i, m)$. This change would make the signatures somewhat longer, but the public key of the signer could be computed directly as A_i/B_i^{ω}.

of the individual group members are computed similarly. This method has the disadvantages that a trusted third party is needed to compute the group's public and secret keys, and that if more than k group members collude, they can find out all secret keys and therefore sign on behalf of any authorized set. Hence there exists a trade-off between the size of the group's public key and the security.

7 Open Problems

In all previously proposed schemes, as well as in our scheme, the size of the group's public key is linear in the number of group members. It is an open problem to construct a group signature scheme where the size of the public key and the amount of computation for signing and verifying does not depend on the size of the group (the only proposed schemes [13,16] with fixed size public keys were broken).

Acknowledgments

It is has been a pleasure to discuss group signatures and the results of this paper with Christian Cachin, Ronald Cramer, Ueli Maurer, and Markus Stadler. These discussions greatly improved the paper. The comments of the anonymous referees were also welcomed.

The author is supported by the Swiss Commission for Technology and Innovation (KTI) and by the Union Bank of Switzerland.

References

1. J. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. In S. Goldwasser, editor, *Advances in Cryptology — CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 27–35. Springer-Verlag, 1990.
2. R. Blom. An optimal class of symmetric key generation systems. *Proc. EUROCRYPT'84, Lecture Notes in Comp. Sc., vol. 209, New York, NY: Springer Verlag*, pages 335–338, 1985.
3. C. Boyd. Digital multisignatures. In H. J. Beker and F. Piper, editors, *Cryptography and Coding*, pages 241–246. The Institute of Mathematics and its Applications Conference Series, Oxford Science Publications, 1989.
4. S. Brands. An efficient off-line electronic cash system based on the representation problem. Technical Report CS-R9323, CWI, Apr. 1993.
5. D. Chaum and T. Pedersen. Wallet databases with observers. In E. F. Brickell, editor, *Advances in Cryptology — CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105. Springer-Verlag, 1993.
6. D. Chaum and E. van Heyst. Group signatures. In D. W. Davies, editor, *Advances in Cryptology — EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer-Verlag, 1991.
7. L. Chen and T. P. Pedersen. New group signature schemes. In A. D. Santis, editor, *Advances in Cryptology — EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 171–181. Springer-Verlag, 1995.

8. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. G. Desmedt, editor, *Advances in Cryptology – CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer Verlag, 1994.
9. R. Croft and S. Harris. Public key cryptography and re-usable shared secrets. In H. J. Beker and F. Piper, editors, *Cryptography and Coding*, pages 189–201. The Institute of Mathematics and its Applications Conference Series, Oxford Science Publications, 1989.
10. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology – CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer Verlag, 1985.
11. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proc. 28th IEEE Symp. Found. Comp. Sc.*, pages 427–437, 1987.
12. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In U. Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 354–371. Springer Verlag, 1996.
13. S. J. Kim, S. J. Park, and D. H. Won. Convertible group signatures. In K. Kim and T. Matsumoto, editors, *Advances in Cryptology — ASIACRYPT '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 311–321. Springer Verlag, 1996.
14. M. Mambo, K. Usuda, and E. Okamoto. Proxy signatures for delegating signing operation. In *3rd ACM Conference on Computer and Communicatons Security*, pages 48–57, New Delhi, Mar. 1996. acm press.
15. K. Ohta and T. Okamoto. A digital multisignature scheme based on the Fiat-Shamir scheme. In H. Imai, R. L. Rivest, and T. Matsumoto, editors, *Advances in Cryptology — ASIACRYPT '91*, volume 739 of *Lecture Notes in Computer Science*, pages 139–148. Springer-Verlag, 1993.
16. S. J. Park, I. S. Lee, and D. H. Won. A practical group signature. In *Proceedings of the 1995 Japan-Korea Workshop on Information Security and Cryptography*, pages 127–133, Jan. 1995.
17. C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.
18. A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.