In: Advances in Cryptology – CRYPTO'97, volume 1294 of Lecture Notes in Computer Sciences, pp. 410–424, Springer Verlag, 1997.

Efficient Group Signature Schemes for Large Groups

(Extended Abstract)

Jan Camenisch
Department of Computer Science
Haldeneggsteig 4
ETH Zurich
8092 Zurich, Switzerland
camenisch@inf.ethz.ch

Markus Stadler
Ubilab
Union Bank of Switzerland
Bahnhofstr. 45
8021 Zurich, Switzerland
Markus.Stadler@ubs.com

Abstract. A group signature scheme allows members of a group to sign messages on the group's behalf such that the resulting signature does not reveal their identity. Only a designated group manager is able to identify the group member who issued a given signature. Previously proposed realizations of group signature schemes have the undesirable property that the length of the public key is linear in the size of the group. In this paper we propose the first group signature scheme whose public key and signatures have length independent of the number of group members and which can therefore also be used for large groups. Furthermore, the scheme allows the group manager to add new members to the group without modifying the public key. The realization is based on methods for proving the knowledge of signatures.

1 Introduction

A group signature scheme allows members of a group to sign messages on behalf of the group. Signatures can be verified with respect to a single group public key, but they do not reveal the identity of the signer. Furthermore, it is not possible to decide whether two signatures have been issued by the same group member. However, there exists a designated *group manager* who can, in case of a later dispute, *open* signatures, i.e., reveal the identity of the signer.

Group signatures could for instance be used by a company for authenticating price lists, press releases, or digital contracts. The customers need to know only a single company public key to verify signatures. The company can hide any internal organizational structures and responsibilities, but still can find out which employee (i.e., group member) has signed a particular document.

The concept of group signatures was introduced by Chaum and van Heyst [11] and they also proposed the first realizations. Improved solutions were later presented by Chen and Pedersen [12], Camenisch [7], and Petersen [22]. However, all previously proposed solutions have the following undesirable properties:

- the length of the group's public key and/or the size of a signature depends on the size of the group. This is very problematic for large groups.
- to add new group members, it is necessary to modify at least the public key.

In this paper we present the first efficient group signature schemes which overcome these problems[1]. The lengths of the public key and of the signatures

[1] The only previously proposed schemes with fixed size public keys [21, 17] were broken.

are, as well as the computational effort for signing and verifying, *independent of the number of group members.* Furthermore, the public key *remains unchanged* if new members are added to the group. The schemes even conceal the size of the group.

For realizing such schemes we employ novel techniques of independent interest, such as efficient proofs of (or signatures of) knowledge of double discrete logarithms, of e-th roots of discrete logarithms, and of e-th roots of components of representations. Of particular interest is a method for proving the knowledge of a signature.

2 Group Signature Schemes

In this section we present the concept of a group signature scheme and explain the basic idea underlying our realizations.

2.1 The Concept of a Group Signature Scheme

A group signature scheme consists of the following four procedures:

`Setup:` a probabilistic interactive protocol between a designated group manager and the members of the group. Its result consists of the group's public key $\mathcal{Y}$, the individual secret keys x of the group members, and a secret administration key for the group manager.

`Sign:` a probabilistic algorithm which, on input a message m and a group member's secret key x, returns a signature s on m.

`Verify:` an algorithm which, on input a message m, a signature s, and the group's public key $\mathcal{Y}$, returns whether the signature is correct.

`Open:` on input a signature s and the group manager's secret administration key this algorithm returns the identity of the group member who issued the signature s together with a proof of this fact.

It is assumed that all communications between the group members and the group manager are secure. A group signature scheme must satisfy the following properties:

1. Only group members are able to correctly sign messages (unforgeability).
2. It is neither possible to find out which group member signed a message (anonymity) nor to decide whether two signatures have been issued by the same group member (unlinkability).
3. Group members can neither circumvent the opening of a signature nor sign on behalf of other group members; even the group manager cannot do so (security against framing attacks).

A consequence of the last property is that the group manager must not know the secret keys of the group members.

In an extended model it may be desirable to assign the different roles of the group manager, namely managing the membership list of the group and opening signatures, to different parties. Furthermore, these roles could be shared among several parties (i.e. among the group members) in order to increase the security against a cheating group manager.

With regard to the efficiency of a group signature scheme the following parameters are of particular interest:

- the size of the group public key $\mathcal{Y}$,
- the length of signatures,
- the efficiency of the algorithms `Sign` and `Verify`,
- and the efficiency of the protocols `Setup` and `Open`.

In all previously proposed schemes, the length of the public key is at least linear in the size of the group and therefore also the running time of the verification algorithm depends on the number of group members. In some schemes also the length of the signature and the running time of the signing algorithm depend on the group size. In Sections 4 and 6 we propose new group signature schemes which overcome these problems. Both solutions are based on the following idea.

2.2 Schemes with Fixed Size Public Key and Signatures

Using the techniques of Brassard et al. [6] or Boyar et al. [3] for proving the knowledge of a satisfying assignment of a boolean circuit, a group signature scheme with fixed length public key and signatures can be constructed as follows.

The group manager computes a key pair of an ordinary digital signature scheme, denoted (sig_M, ver_M), and a key pair of a probabilistic public-key encryption scheme, denoted $(encr_M, decr_M)$, and publishes the two public keys as the group public key. Alice can join the group in the following way: she chooses a random *secret key* x and computes a *membership key* $z = f(x)$, where f is a one-way function. She commits herself to z, e.g., by signing it, and then sends z to the group manager who returns to her the *membership certificate* $v = sig_M(z)$. Alice's group secret key consists of the triple (x, z, v).

To sign a message m on behalf of the group, Alice encrypts the pair (m, z) using the group manager's encryption key, i.e., $d = encr_M(r, (m, z))$, where r is a sufficiently large random string. She computes a non-interactive minimum-disclosure proof p that she knows values x', v', and r' satisfying the following equations:

$$d = encr_M(r', (m, f(x'))) \quad \text{and} \quad ver_M(v', f(x')) = \texttt{correct} \ .$$

The resulting signature on the message m consists of the pair (d, p) and can be verified by checking the proof p. To open this signature, the group manager decrypts the ciphertext d to obtain the membership key z which reveals Alice's identity. A proof of this fact consists of z, Alice's commitment to it, and a non-interactive proof that d encrypts (m, z).

It can easily be verified that all security properties hold:

1. Only group members who know a membership certificate can construct a valid proof p.
2. Because the proof p does not reveal information about x, z, or v, and because (m, z) is probabilistically encrypted, signatures are anonymous and unlinkable.

3. Group members cannot circumvent the opening of signatures because they prove that the value d contains their membership key.

Note that instead of encrypting the message m in d, one could instead make the proof p message-dependent (see Section 3).

The disadvantage of this solutions is that the general techniques for proving statements in minimum-disclosure make the resulting signatures very large and impractical. The rest of the paper describes techniques for the construction of more efficient scheme based on proofs (or signatures) about the knowledge of double discrete logarithms and about the knowledge of roots of logarithms.

3 Preliminaries and Techniques

After giving notational and number theoretic preliminaries, we present some well known techniques for proving knowledge of discrete logarithms and extend them to the building blocks for our group signature schemes.

3.1 Notations

The symbol $||$ denotes the concatenation of two (binary) strings (or of binary representations of integers and group elements) and ‘ ’ denotes the empty string. By $c[i]$ we denote the i-th rightmost bit of the string c. If A is a set, $a \in_R A$ means that a is chosen at random from A according to the uniform distribution. For an integer q, $\mathbb{Z}_q$ denotes the ring of integers modulo q and $\mathbb{Z}_q^*$ denotes the multiplicative group modulo q. Finally, we assume a collision resistant hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$ ($k \approx 160$).

3.2 Number Theoretic Preliminaries

Let $G = \langle g \rangle$ be a cyclic group of order n, and let a be an element of $\mathbb{Z}_n^*$. The discrete logarithm of $y \in G$ to the base g is the smallest positive integer x satisfying $g^x = y$. Similarly, the double discrete logarithm of $y \in G$ to the bases g and a is the smallest positive integer x satisfying

$$g^{(a^x)} = y \ ,$$

if such an x exists. In the sequel, the parameters n, G, g, and a should be chosen such that computing discrete logarithms in G to the base g and in $\mathbb{Z}_n^*$ to the base a is infeasible.

An e-th root of the discrete logarithm of $y \in G$ to the base g is an integer x satisfying

$$g^{(x^e)} = y \ ,$$

if such an x exists. Note that if the factorization of n is unknown, for instance if n is an RSA modulus (see [24]), computing e-th roots in $\mathbb{Z}_n^*$ is assumed to be infeasible.

3.3 Signature of Knowledge of Discrete Logarithms

Throughout this paper we make use of "proof systems" that allow one party to convince other parties about its knowledge of certain values, such that no useful information is leaked. Various such systems have been proposed, for instance minimum-disclosure proofs [6] and zero-knowledge proofs of knowledge [15]. We will make use of constructions based on the Schnorr signature scheme [25] to prove knowledge. However, to avoid confusions with the notion of proofs of knowledge of [15] and to point out that these proofs also serve as signatures, we call them *signatures of knowledge*. All these signatures of knowledge can be proved secure in the random oracle model [2, 15] and their interactive versions are zero-knowledge (given that several rounds with small challenges are used).

The first primitive we define is a signature of the knowledge of the discrete logarithm of y to the base g. It is basically a Schnorr signature [25] on a message m of the entity knowing the discrete logarithm of y.

Definition 3.1. A pair $(c, s) \in \{0,1\}^k \times \mathbb{Z}_n^*$ satisfying $c = \mathcal{H}(\, m \,||\, y \,||\, g \,||\, g^s y^c)$ is a signature of knowledge of the discrete logarithm of the element $y \in G$ to the base g on the message m. □

Such a signature can be computed if the secret key $x = \log_g(y)$ is known, by choosing r at random from $\mathbb{Z}_n^*$ and computing c and s according to

$$c := \mathcal{H}(m \,||\, y \,||\, g \,||\, g^r) \quad \text{and} \quad s := r - cx \pmod{n}.$$

This technique for constructing a signature of the knowledge of a discrete logarithm can also be used to build signatures that involve more complex statements, such as the knowledge of a representation of y to the bases g and h, i.e., a pair (α, β) satisfying $y = g^\alpha h^\beta$ (see [4] for a discussion of the representation problem). Even signatures of knowledge of complex relationships among different representations are possible [5, 8, 16].

Before we define such signatures of knowledge let us explain our notation with the following example: a signature of knowledge, denoted

$$\mathit{SKREP}\left[(\alpha, \beta) : \; y = g^\alpha \;\wedge\; z = g^\beta h^\alpha\right](m),$$

is used for 'proving' the knowledge of the discrete logarithm of y to the base g and of a representation of z to the bases g and h, and in addition, that the h-part of this representation equals the discrete logarithm of y to the base g. This is equivalent to the knowledge of a pair (α, β) satisfying the equations on the right side of the colon. In the sequel, we use the convention that Greek letters denote the elements whose knowledge is proven and all other letters denote elements that are known to the verifier. We now generalize these types of signatures of knowledge.

Definition 3.2. A signature of the knowledge of representations of $y_1, \ldots, y_w$ with respect to the bases $g_1, \ldots, g_v$ on the message m is denoted as follows

$$\mathit{SKREP}\left[(\alpha_1, \ldots, \alpha_u) : \; \left(y_1 = \prod_{j=1}^{\ell_1} g_{b_{1j}}^{\alpha_{e_{1j}}}\right) \wedge \ldots \wedge \left(y_w = \prod_{j=1}^{\ell_w} g_{b_{wj}}^{\alpha_{e_{wj}}}\right)\right](m) \;,$$

where the indices $e_{ij} \in \{1, \ldots, u\}$ refer to the elements $\alpha_1, \ldots, \alpha_u$ and the indices $b_{ij} \in \{1, \ldots, v\}$ refer to the base elements $g_1, \ldots, g_v$. The signature consists of an $(u+1)$ tuple $(c, s_1, \ldots, s_u) \in \{0,1\}^k \times \mathbb{Z}_n^u$ satisfying the equation

$$c = \mathcal{H}\Big(m \| y_1 \| ... \| y_w \| g_1 \| ... \| g_v \| \{\{e_{ij}, b_{ij}\}_{j=1}^{\ell_i}\}_{i=1}^{w} \| y_1^c \prod_{j=1}^{\ell_1} g_{b_{1j}}^{s_{e_{1j}}} \| ... \| y_w^c \prod_{j=1}^{\ell_w} g_{b_{e_{wj}}}^{s_{e_{wj}}}\Big) \qquad \square$$

SKREP can be computed in the same way as the simple signature of knowledge of a discrete logarithm if a u-tuple $(\alpha_1, \ldots, \alpha_u)$ is known which satisfies the given equations. One first chooses $r_i \in_R \mathbb{Z}_n$ for $i = 1, \ldots, u$, computes c as

$$c := \mathcal{H}\Big(m \| y_1 \| \ldots \| \{\{e_{ij}, b_{ij}\}_{j=1}^{\ell_i}\}_{i=1}^{w} \| \prod_{j=1}^{\ell_1} g_{b_{1j}}^{r_{e_{1j}}} \| \ldots \| \prod_{j=1}^{\ell_w} g_{b_{e_{wj}}}^{r_{e_{wj}}}\Big) \ ,$$

and then sets $s_i := r_i - c\alpha_i \pmod{n}$ for $i = 1, \ldots, u$.

Signatures of knowledge of representations are a powerful tool for constructing various cryptographic systems, but we will also employ signatures of the knowledge of double discrete logarithms (see [26]) and of roots of logarithms.

Definition 3.3. Let $\ell \leq k$ be a security parameter. An $(\ell+1)$ tuple $(c, s_1, ..., s_\ell)$ $\in \{0,1\}^k \times \mathbb{Z}^\ell$ satisfying the equation

$$c = \mathcal{H}(\, m \,\|\, y \,\|\, g \,\|\, a \,\|\, t_1 \,\|\, \ldots \,\|\, t_\ell\,) \quad \text{with} \quad t_i = \begin{cases} g^{(a^{s_i})} & \text{if } c[i] \;=\; 0 \\ y^{(a^{s_i})} & \text{otherwise} \end{cases}$$

is a signature the knowledge of a double discrete logarithm of y to the bases g and a, and is denoted $\mathit{SKLOGLOG}\big[\alpha :\ y = g^{(a^\alpha)}\,\big](m)$. $\square$

An $\mathit{SKLOGLOG}\big[\alpha :\ y = g^{(a^\alpha)}\,\big]\,(m)$ can be computed only if the double discrete logarithm x of the group element y to the bases g and a is known. We assume that there is an upper bound λ on the length of x, i.e., $0 \leq x < 2^\lambda$ ($\lambda = |n|$ is an example, but for certain applications, smaller bounds can be used). Let $\epsilon > 1$ be a constant. One first computes the values

$$t_i^* := g^{(a^{r_i})}$$

for $i = 1, \ldots, \ell$ with randomly chosen $r_i \in \{0. \ldots, 2^{\epsilon\lambda} - 1\}$. Then, c is set to $\mathcal{H}(\, m \,\|\, y \,\|\, g \,\|\, a \,\|\, t_1^* \,\|\, \ldots \,\|\, t_\ell^*\,)$, and finally,

$$s_i := \begin{cases} r_i & \text{if } c[i] \;=\; 0, \\ r_i - x & \text{otherwise.} \end{cases}$$

for $i = 1, \ldots, \ell$. It can easily be verified that the resulting tuple $(c,\, s_1, \ldots, s_\ell)$ satisfies the verification equation. Note that if the order of $a \in \mathbb{Z}_n^*$ is known, the computations of the s_i can be "reduced" modulo this order.

Definition 3.4. An $(\ell+1)$ tuple $(c, s_1, \ldots, s_\ell) \in \{0,1\}^k \times \mathbb{Z}_n^{*\ell}$ satisfying the equation

$$c = \mathcal{H}(m \,\|\, y \,\|\, g \,\|\, e \,\|\, t_1 \,\|\, \ldots \,\|\, t_\ell) \quad \text{with} \quad t_i = \begin{cases} g^{(s_i^e)} & \text{if } c[i] \;=\; 0 \\ y^{(s_i^e)} & \text{otherwise} \end{cases}$$

is a signature of the knowledge of an e-th root of the discrete logarithm of y to the base g, and is denoted $SKROOTLOG\left[\alpha :\; y = g^{\alpha^e}\right](m)$. □

Note that the values $s_1, \ldots, s_\ell$ belong to $\mathbb{Z}_n^*$ and therefore must not be zero.

Such a signature can be computed if the e-th root x of the discrete logarithm of y to the base g is known. One first computes the values

$$t_i^* := g^{(r_i^e)}$$

for $i = 1, \ldots, \ell$ with randomly chosen $r_i \in \mathbb{Z}_n^*$. Then, c is set to $\mathcal{H}(\, m \,\|\, y \,\|\, g \,\|\, e \,\|\, t_1^* \,\|\, \ldots \,\|\, t_\ell^* \,)$, and finally,

$$s_i := \begin{cases} r_i & \text{if } c[i] \;=\; 0, \\ r_i/x \pmod{n} & \text{otherwise.} \end{cases}$$

for $i = 1, \ldots, \ell$. It can easily be seen that the resulting tuple $(c,\, s_1, \ldots, s_\ell)$ satisfies the verification equation.

4 The Basic Group Signature Scheme

In this section we propose a first realization of a group signature scheme based on the ideas presented in the end of Section 2. In this solution, the opening of signatures can even be realized in a simpler way.

4.1 System Setup

The group manager computes the following values:

- an RSA public key (n, e),
- a cyclic group $G = \langle g \rangle$ of order n in which computing discrete logarithms is infeasible (e.g. G could be a subgroup of $\mathbb{Z}_p^*$, for a prime p with $n|(p-1)$),
- an element $a \in \mathbb{Z}_n^*$ (a should be of large multiplicative order modulo both prime factors of n), and
- an upper bound λ on the length of the secret keys and a constant $\epsilon > 1$ (these parameters are required for the $SKLOGLOG$ signatures)

The group's public key is $\mathcal{Y} = (n, e, G, g, a, \lambda, \epsilon)$.

4.2 Generating Membership Keys and Certificates

When Alice is to join the group, she chooses her secret key $x \in_R \{0, \ldots, 2^\lambda - 1\}$ and computes the value $y := a^x \pmod{n}$ and her membership key $z = g^y$. She commits herself to y, for instance by signing it. She then sends y and z to the group manager and proves to him that she knows the discrete logarithm of y to the base a (this can be done with techniques similar to those for signatures of knowledge of a discrete logarithm, with the difference that the group order is unknown). When the group manager is convinced that Alice knows this logarithm, he returns to her the membership certificate

$$v \equiv (y+1)^{1/e} \pmod{n} \ .$$

It seems infeasible to construct such a triple (x, y, v) without the help of the group manager: on one hand, if y is correctly formed then it is infeasible to compute the e-th root of $y+1$ because the factorization of n is unknown. On the other hand, if $y+1$ is computed as w^e for some value w then it is infeasible to compute the discrete logarithm of $w^e - 1$ to the base a. Furthermore, even if several group members pool their values, they still seem unable to construct a new such triple.

4.3 Signing Messages

To sign a message m, Alice computes the following values:

- $\tilde{g} := g^r$ for $r \in_R \mathbb{Z}_n^*$
- $\tilde{z} := \tilde{g}^y$
- $V_1 := \mathit{SKLOGLOG}\big[\alpha : \ \tilde{z} = \tilde{g}^{a^\alpha}\big](m)$
- $V_2 := \mathit{SKROOTLOG}\big[\beta : \ \tilde{z}\tilde{g} = \tilde{g}^{\beta^e}\big](m)$

The resulting signature on the message m consists of $(\tilde{g},\ \tilde{z},\ V_1,\ V_2)$ and can be verified by checking the correctness of the signatures of knowledge V_1 and V_2.

We now explain briefly why this signature really proves that Alice belongs to the group. On one hand, because of V_1 the value $\tilde{z}\tilde{g}$ must be of the form

$$\tilde{z}\tilde{g} = \tilde{g}^{a^\alpha + 1}$$

for an integer α Alice knows. On the other hand, V_2 proves that Alice knows an e-th root of $(a^\alpha + 1)$, which means that Alice knows the secret key and a membership certificate of her membership key.

4.4 Opening Signatures

Linking two signatures $(\tilde{g}, \tilde{z}, V_1, V_2)$ and $(\tilde{g}', \tilde{z}', V_1', V_2')$, i.e., deciding whether these signatures have been issued by the same group member or not, is only possible by deciding whether $\log_{\tilde{g}} \tilde{z} = \log_{\tilde{g}'} \tilde{z}'$. Generally, solving this problem is infeasible and therefore the signatures of the group members are anonymous and unlinkable. However, the group manager has an advantage: he knows the relatively few possible values of $\log_{\tilde{g}} \tilde{z}$, namely the discrete logarithms (to the base g) of the membership keys of the group members, and can therefore perform this test. Given only a signature $(\tilde{g}, \tilde{z}, V_1, V_2)$ for a message m, the group manager can find the group member who issued this signature by testing

$$\tilde{g}^{y_P} \overset{?}{=} \tilde{z}$$

for all group members P (where y_P denotes discrete logarithm of P's membership key z_P to the base g). A proof of this fact consists of the signer's membership key z_P, his commitment to this key, and a non-interactive proof of the equality of $\log_g z$ and $\log_{\tilde{g}} \tilde{z}$. Unfortunately, this method is impractical for very large groups. In Section 6 we present an extension that makes it possible to identify group members directly.

4.5 Security and Efficiency Considerations

The security of the basic group signature scheme presented in this section is based on the difficulty of the discrete logarithm problem and on the security of the Schnorr [25] and of the RSA [24] signature schemes. It is also based on the additional assumption that computing membership certificates of valid membership keys is infeasible if the factorization of the modulus n is unknown. With regard to the anonymity of group members, linking two signatures is as hard as deciding whether two discrete logarithms are equal (for instance, undeniable signatures [10] make also use of this assumption).

With the following values of the system parameters

$$k = 160,\ \ell = 64,\ \lambda = 170,\ \epsilon = 4/3,\ |n| = 600, \text{ and } e = 3,$$

a signature is less than 7 KBytes long and the operations for signing messages and for verifying signatures require the computation of approximately 140'000 modular multiplications with a 600 bit modulus (this corresponds to about 155 exponentiations with full 600 bit exponents).

5 Efficient *SKROOTLOG*

A disadvantage of the scheme presented in the previous section is that the signatures *SKROOTLOG* and *SKLOGLOG* are quite inefficient. In this section we show how an efficient *SKROOTLOG* can be realized when the exponent e is small.

5.1 A Simple Observation

If e is small then it is possible to efficiently convince somebody about one's knowledge of the e-th root of the discrete logarithm of $z = g^{x^e}$ to the base g by computing the following $e - 1$ values:

$$z_1 := g^x,\ z_2 := g^{x^2}, \ldots,\ z_{e-1} := g^{x^{e-1}}$$

and showing with a signature of knowledge

$$U := \mathit{SKREP}\left[\alpha :\ z_1 = g^\alpha\ \wedge\ z_2 = z_1^\alpha\ \wedge \ldots \wedge\ z = z_{e-1}^\alpha\right]$$

that the discrete logarithms 'between' two subsequent values in the list g, z_1, $\ldots$, z_{e-1}, z are all equal and known. Therefore the following equations

$$z = z_{e-1}^\alpha = z_{e-2}^{\alpha^2} = \ldots = z_1^{\alpha^{e-1}} = g^{\alpha^e}$$

must hold and the knowledge of an e-th root of z to the base g is assured. More generally, one could use any addition chain for the integer e, but we restrict ourselves to this simple case for the rest of the paper.

However, the problem of this approach is that the values $z_1, \dots, z_{e-1}$ leak additional information. In the next section we show how these values can be randomized. This leads to an efficient *SKROOTLOG* presented in the next but one section.

5.2 Efficient Signatures Proving the Knowledge of Roots of Representations

From now on we assume that an element $h \in G$ is available whose discrete logarithm to the base g is unknown (for instance, h could be computed according to a suitable pseudo-random process with g as seed). The element h is now used to randomize (or blind) z and the z_i's of the previous section, i.e., v becomes $h^r z$ for some random r and one wants to 'prove' the knowledge of a pair (α, β) for which $v = h^\alpha g^{\beta^e}$ holds. Such a signature can be given efficiently by applying the method described above.

Similar techniques have already been used in [13, 19] for the purpose of proving properties of bit commitments.

Definition 5.1. An efficient signature of the knowledge of the e-th root of the g-part of a representation of v to the bases h and g, denoted

$$\textit{E-SKROOTREP}\left[(\alpha,\beta):\ v = h^\alpha g^{\beta^e}\right](m)\ ,$$

consists of an $(e-1)$-tuple $(v_1, \dots, v_{e-1}) \in G^{e-1}$ and of a signature of knowledge

$$U = \textit{SKREP}\Big[(\gamma_1,\dots,\gamma_e,\delta):\ v_1 = h^{\gamma_1} g^{\delta}\ \wedge\ v_2 = h^{\gamma_2} v_1^{\delta}\ \wedge \dots$$
$$\dots \wedge\ v_{e-1} = h^{\gamma_{e-1}} v_{e-2}^{\delta}\ \wedge\ v = h^{\gamma_e} v_{e-1}^{\delta}\Big](m)\ .$$

The signature of knowledge can be verified by checking the correctness of U. □

The following equation explains why a verifier will be convinced of the prover's knowledge of (α, β):

$$v = h^{\gamma_e}\Big(h^{\gamma_{e-1}}\big(\dots h^{\gamma_2}(h^{\gamma_1} g^{\delta})^{\delta}\dots\big)^{\delta}\Big)^{\delta} = h^{\gamma_e + \gamma_{e-1}\delta + \dots + \gamma_2\delta^{e-2} + \gamma_1\delta^{e-1}} g^{\delta^e} =: h^\alpha g^{\beta^e}.$$

Such a signature can be computed if values r and x in $\mathbb{Z}_n$ are known for which $v = h^r g^{x^e}$: one first computes the values $v_i := h^{r_i} g^{x^i}$ for $i = 1, \dots, e-1$ with randomly chosen $r_i \in \mathbb{Z}_n$. Then the signature of knowledge U is computed. Note that the elements v_i are truly random group elements and so do not leak any information.

5.3 Efficient Signatures proving the Knowledge of Roots of Logarithms

Based on the *E-SKROOTREP*'s, it is now easy to construct an efficient and secure *E-SKROOTLOG*, by showing that z itself is not blinded:

Definition 5.2. A efficient signature on the knowledge of the e-th root of the discrete logarithm of z to the base g, denoted

$$\textit{E-SKROOTLOG}\Big[\delta : \ z = g^{\delta^e}\Big](m)$$

consists of the two signatures

$$\textit{E-SKROOTREP}\Big[(\alpha, \beta) : \ z = h^{\alpha} g^{\beta^e}\Big](m) \quad \text{and} \quad \textit{SKREP}\big[\gamma : \ z = g^{\gamma}\big](m)$$

where the discrete logarithm of h the base g must be unknown. $\square$

Since one can know only one representation of z to the bases h and g, it follows that $\alpha \equiv 0 \pmod{n}$ and $\gamma \equiv \beta^e \pmod{n}$ and that the prover knows the e-th root of the discrete logarithm of z to the base g.

6 A More Efficient Variant

Because similar improvements as for *SKROOTLOG* signatures seem not be possible for *SKLOGLOG* signature, an evident solution to design a more efficient group signature scheme is to modify it in a way that allows to replace the *SKLOGLOG* by an *SKROOTLOG* signature. This is indeed possible if the membership key is computed using $y = x^e \pmod{n}$ instead of $y = h^x \pmod{n}$.

As an immediate consequence, the group manager must be prevented from learning the value y (otherwise he could compute an e-th root of y and sign on behalf of group members). This problem can be solved by sending the group manager only g^y and adapting the protocol for issuing the membership certificates accordingly. Furthermore, because the group manager no longer knows y, the method for opening signatures as described in Section 2.2 must be realized.

6.1 System Setup

The group manager computes the following values:

- an RSA modulus n and two public exponents e_1, $e_2 > 1$, such that e_2 is relatively prime to $\varphi(n)$,
- two integers $f_1, f_2 > 1$ whose e_1-th roots and e_2-th roots cannot be computed without knowing the factorization of n,
- a cyclic group $G = \langle g \rangle$ of order n in which computing discrete logarithms is infeasible,
- an element $h \in G$ whose discrete logarithm to the base g must not be known,
- his public key $y_R = h^{\rho}$ for a randomly chosen value $\rho \in \mathbb{Z}_n$.

The group's public key consists of $\mathcal{Y} = (n, e_1, e_2, f_1, f_2, G, g, h, y_R)$, whereas ρ and the factorization of n remain the group manager's secret key. Possible choices for the parameters e_1, e_2, f_1, and f_2 are discussed in section 6.5.

6.2 Membership Keys and Blind Issuing of Membership Certificates

To become a group member, Alice computes her membership key as follows:

- $y := x^{e_1} \pmod{n}$ for $x \in_R \mathbb{Z}_n^*$ (see also discussion in Section 6.5)
- $z := g^y$

A certificate in this scheme is of the form

$$v = (f_1 y + f_2)^{1/e_2} \pmod{n} \ .$$

To prevent the group manager from learning y, this certificate must be issued using the blind RSA-signature scheme of Chaum [9]. Additionally, Alice must send z to the group manager and convince him that the discrete logarithm of z to the base g is a valid membership key and is contained in the blinded certificate. More formally, Alice computes

- $\tilde{y} := r^{e_2}(f_1 y + f_2) \pmod{n}$ for $r \in_R \mathbb{Z}_n^*$
- $U := \textit{E-SKROOTLOG}\big[\alpha : \ z = g^{\alpha^{e_1}}\big](\text{` '})$
- $V := \textit{E-SKROOTLOG}\big[\beta : \ g^{\tilde{y}} = (z^{f_1} g^{f_2})^{\beta^{e_2}}\big](\text{` '})$

and sends $\tilde{y}, z, U$, and V to the group manager. If U and V are correct, the group manager sends Alice the blinded certificate

$$\tilde{v} = \tilde{y}^{1/e_2} \pmod{n},$$

which Alice unblinds and thereby obtains her membership certificate

$$v = \tilde{v}/r = (f_1 y + f_2)^{1/e_2} \pmod{n}.$$

Let us now explain what the signatures of knowledge U and V actually mean. The signature U shows that the element z is of the form $g^{\alpha^{e_1}}$ for some α Alice knows. The signature V assures that $\tilde{y} = \beta^{e_2}(f_1 \alpha^{e_1} + f_2) \pmod{n}$ holds for some β Alice knows, and therefore the group manager can conclude that $\tilde{y}$ is a correctly blinded membership key.

6.3 Signing Messages

To sign a message m on behalf of the group, Alice performs the following computations:

- $\tilde{z} := h^r g^y$ for $r \in_R \mathbb{Z}_n^*$
- $d := y_R^r$
- $V_1 := \textit{E-SKROOTREP}\big[(\alpha, \beta) : \ \tilde{z} = h^{\alpha} g^{\beta^{e_1}}\big](m)$
- $V_2 := \textit{E-SKROOTREP}\big[(\gamma, \delta) : \ \tilde{z}^{f_1} g^{f_2} = h^{\gamma} g^{\delta^{e_2}}\big](m)$
- $V_3 := \textit{SKREP}\big[(\varepsilon, \zeta) : \ d = y_R^{\varepsilon} \ \wedge \ \tilde{z} = h^{\varepsilon} g^{\zeta}\big](m)$

The resulting signature on the message m consists of $(\tilde{z}, d, V_1, V_2, V_3)$ and is valid if the three signatures of knowledge V_1, V_2, and V_3 are correct.

The following explains briefly why such a signature convinces a verifier that the signer knows the secret key of a certified membership key. Consider the signature V_1: it 'proves' that the signer knows a representation (α, β^{e_1}) of $\tilde{z}$ to the bases h and g and that she knows the e_1-th root of the g-part of this representation, i.e., β. The signature V_2 'proves' the signer's knowledge of a representation (γ, δ^{e_2}) of $\tilde{z}^{f_1} g^{f_2}$ to the bases h and g and her knowledge of the e_2-th root of δ^{e_2}. As the signer can know *at most one* representation of $\tilde{z}^{f_1} g^{f_2}$ to the bases h and g it follows that

$$\gamma \equiv \alpha \pmod{n} \quad \text{and} \quad \delta^{e_2} \equiv f_1 \beta^{e_1} + f_2 \pmod{n}.$$

The fact that the signer knows an e_1-th root of β^{e_1} and an e_2-th root of δ^{e_2} means that she knows a membership certificate and the secret key of the corresponding membership key.

Finally, consider the element d and the signature V_3. The pair $(d, \tilde{z})$ is a (modified) ElGamal encryption [14] of g^y encrypted under the group manager's public key y_R and enables the group manager to open signatures. The signature V_3 guarantees that this encryption is formed correctly.

6.4 Opening Signatures

When the group manager wants to open a signature $(\tilde{z},\ d,\ V_1,\ V_2,\ V_3)$ on the message m, he computes $\hat{z} := \tilde{z}/d^{1/\rho}$ which corresponds to the signer's membership key z. To prove that z is indeed encrypted in $\tilde{z}$ and d, the group manager computes

$$\mathit{SKREP}[\alpha :\ \tilde{z} = z d^{\alpha}\ \wedge\ h = y_R^{\alpha}](\text{' '}),$$

which he can do because $1/\alpha \pmod{n}$ corresponds to his administration key.

6.5 Security and Efficiency Considerations

The security of the group signature scheme presented in this section is based on the difficulty of the discrete logarithm problem and on the security of the RSA and Schnorr signature schemes. The security of the scheme relies also on the difficulty of computing certificates when the factorization of n is unknown. The latter depends on the choices for the values e_1, e_2, f_1, and f_2. For instance, the choice $e_1 = 2$, $e_2 = 2$, and $f_1 = 1$, related to the Ong-Schnorr-Shamir signature scheme [20], is not secure for any value of f_2 as is shown in [1, 23]. Generally, it is regarded open problem to determine which types of polynomial congruences with composite moduli are hard to solve [18]. Furthermore, it is also important that when given several solutions of such a polynomial congruence it remains hard to compute other ones.

In order to make it harder to forge membership certificates, it is possible to modify the group signature scheme such that only solutions of the polynomial equation are accepted that meet additional requirements. For instance, by modifying the *E-SKROOTLOG* signature V_1, one can efficiently prove that the secret key x is smaller than $\sqrt{n}$ (the techniques are similar to those used for the *SKLOGLOG* signatures). As a challenge, we propose to use this approach with the following parameters:

$$e_1 = 5, e_2 = 3, f_1 = 1, \text{ and, } f_2 \text{ such that its 3rd root is hard to compute.}$$

For this choice and with $k = 160$ and $|n| = 600$, a signature is about 1.4 KByte long and the operations for signing for verifying signatures require the computation of approximately 18'000 modular multiplications with a 600 bit modulus (this corresponds to about 20 exponentiations with full 600 bit exponents).

7 Extensions

An obvious (and for the second scheme simple) extension would be to assign the different roles of the group manager to different entities, i.e., to a membership manager, who is responsible for adding new members to the group, and to a revocation manager, who is responsible for opening signatures. The functionality of these managers can also be shared among several entities. The realization is straightforward.

Acknowledgments

Many thanks to M. Abe, D. Bleichenbacher, C. Cachin, R. Cramer, U. Maurer, M. Michels, S. Wolf, and the anonymous referee for helpful comments and discussions, and to K. McCurley, D. Coppersmith, and J. Benaloh for pointing us to useful literature. The first author is supported by the Swiss Commission for Technology and Innovation (KTI) and by the Union Bank of Switzerland.

References

1. L. M. Adleman, D. R. Estes, and K. S. McCurley. Solving bivariate quadratic congruences in random polynomial time. *Mathematics of Computation*, 43(177):17–28, Jan. 1987.
2. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communication Security*, pages 62–73. Association for Computing Machinery, 1993.
3. J. Boyar and R. Peralta. Short discreet proofs. In *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 131–142. Springer Verlag, 1996.
4. S. Brands. An efficient off-line electronic cash system based on the representation problem. Technical Report CS-R9323, CWI, Apr. 1993.
5. S. Brands. Rapid demonstration of linear relations connected by boolean operators. In *In Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 318–333. Springer Verlag, 1997.
6. G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, Oct. 1988.
7. J. Camenisch. Efficient and generalized group signatures. In *In Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 465–479. Springer Verlag, 1997.

8. J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. Technical Report TR 260, Institute for Theoretical Computer Science, ETH Zürich, Mar. 1997.
9. D. Chaum. Blind signature systems. In *Advances in Cryptology — CRYPTO '83*, page 153. Plenum Press, 1984.
10. D. Chaum and H. van Antwerpen. Undeniable signatures. In *Advances in Cryptology - CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216. Springer Verlag, 1990.
11. D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology — EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer-Verlag, 1991.
12. L. Chen and T. P. Pedersen. New group signature schemes. In *Advances in Cryptology — EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 171–181. Springer-Verlag, 1995.
13. I. B. Damgård. Practical and provable secure release of a secret and exchange of signature. In *Advances in Cryptology — EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 200–217. Springer-Verlag, 1994.
14. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology — CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer Verlag, 1985.
15. U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1:77–94, 1988.
16. E. Fujisaki and T. Okamoto. Witness hiding protocols to confirm modular polynomial relations. In *The 1997 Symposium on Cryptograpy and Information Security*, Fukuoka, Japan, Jan. 1997. The Institute of Electronics, Information and Communcation Engineers. SCSI97-33D.
17. S. J. Kim, S. J. Park, and D. H. Won. Convertible group signatures. In *Advances in Cryptology — ASIACRYPT '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 311–321. Springer Verlag, 1996.
18. K. McCurley. Odds and ends from cryptology and computational number theory. In *Cryptology and computational number theory*, volume 42 of *Proceedings of Symposia in Applied Mathematics*, pages 145–166. American Mathematical Society, 1990.
19. T. Okamoto. Threshold key-recovery systems for RSA. In *Security Protocols Workshop*, Paris, 1997.
20. H. Ong, C. P. Schnorr, and A. Shamir. Efficient signature schemes based on polymonial equations. In *Advances in Cryptology — CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 37–46. Springer Verlag, 1984.
21. 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.
22. H. Petersen. How to convert any digital signature scheme into a group signature scheme. In *Security Protocols Workshop*, Paris, 1997.
23. J. M. Pollard and C. P. Schnorr. An efficient solution of the congruence $x^2 + ky^2 = m \pmod{n}$. *IEEE Transactions on Information Theory*, 33(5):702–709, September 1987.
24. R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Feb. 1978.
25. C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.
26. M. Stadler. Publicly verifiable secret sharing. In *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 191–199. Springer Verlag, 1996.