ROUGH DRAFT
A Rough Sketch of an Enhanced Group Membership System

Chris Laas

June 42, 2000

`$Id: enhanced-group.tex,v 1.5 2000/07/14 22:32:29 golem Exp $`

1 Introduction

2 System initialization

Before system initialization time, the following algorithms are agreed upon:

- A signature scheme $(Sign_{sk}, Verify_{pk})$ for the CA (e.g. DSA)

At system initialization time, the CA generates the following values:

- A key pair (sk, pk) for the aforementioned signature scheme
- Large primes p, q such that q divides $p-1$. Also, $q-1$ must have a "fairly large" prime factor π. (π need only be significantly greater than the maximum expected lifetime of any group; in practice, choosing $\pi > 2^{20}$ should be more than sufficient to last for the next millenium, even if new "day" certificates are obtained every hour.)
- A generator $g \in \mathbb{Z}_p^*$ of order q (i.e. $g^q = 1 \mod p$)
- Random exponents $e_1, e_2 \in_{\mathcal{R}} \mathbb{Z}_q^*$ not equal to 1. These are secret keys of the CA.
- Generators $g_1 = g^{e_1}$ and $g_2 = g^{e_2}$.

Programmed into each device at initialization are the following parameters:

- $Verify_{pk}$ and the CA's public key pk (which may be updated by a key management protocol external to this document)
- p, q, g, g_1, g_2

In addition, registration of individuals with the CA is assumed to be an external operation and is not addressed here, although it may become part of the system in further drafts. It is assumed that individuals can prove their identity and that each has a unique identifier I.

3 Group creation and destruction

To create a group, a party submits a signed request to the CA containing the following information:

- Proof of authorization to create a group (the form of which is external to this specification)
- The group's owner (which may be an individual, another group, or an external party)
- A list of initial members (individuals or groups)
- A list of verifiers ("interested parties", such as door locks)
- Any auxiliary information

The CA checks the signature and the authorization status of the signator. If it accepts, the CA generates a unique group identifier G, which it stores along with the creation request. It then generates a random value $z \in_{\mathcal{R}} \mathbb{Z}_q^*$ with order π in $\mathbb{Z}_q^*$, and stores it in the database along with G. (z is a CA secret key.) The value $h_1 = g^z \mod p$ is published as the initial public key for the group: it might be printed in a newspaper, or submitted to a third-party timestamping service, etc.

To destroy a group, the owner submits a signed destruction request, and the CA marks the group as destroyed. The CA may remove the group from the active database and put the records in cold storage, but the group identifier G and secret key z should never be reused.

4 Addition of members

To add a member, the group owner simply submits a signed request to the CA, which updates the database. The affected individual may be notified by the group owner, the CA, or by some other out-of-band means. No further action need by taken by the CA until the individual requests certificates.

5 Group membership certificate issuing protocol

To obtain certificates for day k, an individual Alice establishes an encrypted, authenticated channel with the CA Carol, and requests certificates for a group G. The protocol proceeds as follows:

1. After checking that Alice (identified by I) is in G, Carol generates the following values:
   - $h_k = g^{z^k \mod q} \mod p$ (or $h_k = h_{k-1}^z \mod p$ if h_{k-1} is stored)
   - $\sigma_k = Sign_{sk}(G; k; h_k)$
   - $x_k = e_2^{-1}(z^k - Ie_1) \mod q$

   Carol sends $(h_k; \sigma_k; x_k)$ to Alice.

2. Alice checks the following relations:
   - $Verify_{pk}(\sigma_k, (G; k; h_k))$
   - $h_k = g_1^I g_2^{x_k} \mod p$

3. Carol engages in an interactive zero-knowledge proof of knowledge of a value $\zeta \in \mathbb{Z}_q$ such that $h_1 = g^\zeta \mod p$ and $h_k = h_{k-1}^\zeta \mod p$.

Alice accepts if Alice accepts Carol's proof of knowledge and if both of the verification relations hold.

The zero-knowledge proof ensures that the CA could not attempt to make members of the group distinguishable by giving them different public keys. (For this to work, members of the group must compare public keys at some point to ensure that the CA is not cheating. One way to accomplish this might be for a new member to obtain an h_k from an old member, but this is not very practical; this part is a flaw in the system, and needs further work.) The first check verifies the CA's signature on the certificate's public key h_k. The second check ensures that the certificate's secret key does in fact correspond to the certificate's public key.

Alice stores $(G; k; h_k; \sigma_k; x_k)$ as the certificate for the day.

6 Revocation of membership

To remove a member, the group owner simply submits a signed request to the CA, which updates the database. The affected individual may be notified by the group owner, the CA, or by some other out-of-band means.

If the revocation must take effect before the current batch of "day" certificates expires, a CRL mechanism must be employed. If the identity I is to be removed from group G, then the pair (G, I) is signed and distributed to all verifiers who might be interested in revocations from group G.

7 Interactive proof of membership

The proof considered here amounts to "I am a member of group G, but I am NOT the person identified by $\hat{I}$." In more detail, one first shows the certificate $(G; k; h_k; \sigma_k)$ as proof that h_k is the day's public key for G; then, one wants to prove that one knows a representation (I, x_k) of h_k with respect to (g_1, g_2) such that $I \neq \hat{I}$. This is easily accomplished by engaging in a zero-knowledge proof of knowledge of a representation of g_2 with respect to $(g_2^{\hat{I}} h_k^{-1}, g_1)$. That's it.

8 Flaws in this system

- It's not clear how to establish veracity of the public-key 0-token h_1.
- A flaw: if two members of a group collude, they can always prove "I am not the person identified by I"; extending the proof mode to "I am a member of group G, but I am NOT the person identified by I_1 nor I_2 nor I_3, etc." does not help, because a collusion of two members of the group can generate representations with any value of I.
- A greater flaw: if three members of a group collude, they can reconstruct e_1, e_2, and z. This is a total break; I do not know yet how to fix this, as it would seem to be a fundamental flaw with using the representation problem as a basis. One solution would be to grow the size of the representation with the number of members, but this would require the secret and public keys to have size linear in the number of members; this is clearly unacceptable for large groups.
- e_1 and e_2 are long-term valuable secrets and potential breakage points (as opposed to only valuable for a day)

- I is used "bare" in the representation. Perhaps the representation value corresponding to I should be changed daily. This would allow membership to be revoked using a CRL without revealing that Mallory is the person begin revoked (and hence was once a member).

9 Possible modifications

- Put some subset of p, q, g, g_1, g_2 in the short-lived "day" certificate.

  Pros: additional flexibility, especially if those values are compromised (e.g. someone finds a non-trivial representation of 1 with respect to (g_1, g_2)). Possibly, a solution to the problem of collusion to fake identities.

  Cons: larger certificates; the protocols become more complex. Also, this is tricky because it must be possible for group members to verify that each member gets the same values.