ROUGH DRAFT
A Rough Sketch of a Simple Group Membership System

Chris Laas

June 42, 2000

`$Id: group.tex,v 1.8 2000/07/14 22:29:39 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)
- A collision-intractable hash function $\mathcal{H} : \mathbb{Z}_q^* \to \mathbb{Z}_q^*$ (e.g. SHA)

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 $p - 1$.
- A generator $g \in \mathbb{Z}_p^*$ of order q (i.e. $g^q = 1 \mod p$)

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)
- $\mathcal{H}$
- p, q, g

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)
- A "key resynchronization interval" ℓ, which should be greater than the number of days expected until a new group public key sequence is started. (This may be a system parameter. Currently I'm unsure how to bootstrap the public key sequences other than by e.g. publishing a hash fingerprint in a newspaper, which is silly.)
- 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 $x_\ell \in_{\mathcal{R}} \mathbb{Z}_q^*$ not equal to 1, and stores those in the database along with G. (x_ℓ is a CA secret key.) The value $x_0 = \mathcal{H}^\ell(x_\ell)$ is computed, and the value $h_0 = g^{x_0} \bmod p$ 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. (Needs further work.)

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 should never be reused.

4 Addition and removal of members

To add or 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. No further action need by taken by the CA until the individual requests certificates.

There is no CRL mechanism to allow revocations to take effect before the current batch of "day" certificates expires. (This is the goal of the "enhanced" mechanism I'm working on.)

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:

   - $x_k = \mathcal{H}^{\ell-k}(x_\ell)$
   - $h_k = g^{x_k} \mod p$
   - $\sigma_k = \mathit{Sign}_{sk}(G; k; h_k)$

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

2. Alice checks the following relations:

   - $x_{k-1} = \mathcal{H}(x_k)$, or, if x_{k-1} is not known, $h_0 = g^{\mathcal{H}^k(x_k)}$
   - $h_k = g^{x_k} \mod p$
   - $\mathit{Verify}_{pk}(\sigma_k, (G; k; h_k))$

   Alice accepts if all three are true.

The first and second checks ensure 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 a x_k from an old member, but this is not very practical; this part is a flaw in the system, and needs further work.) The third check verifies the CA's signature on the certificate's public key h_k.

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

6 Interactive proof of membership

To prove membership in the group G, Alice first shows the certificate $(G; k; h_k; \sigma_k)$ as proof that h_k is the day's public key for G; then, Alice proves knowledge of a value x_k such that $h_k = g^{x_k} \mod p$.

7 Flaws in this system

- It's not clear how to establish veracity of public-key 0-token h_0.
- Revocation has to wait till certificates expire. Especially, in case of key compromise, there is no way to selectively revoke the certificates of individuals.

8 Possible modifications

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

  Pros: additional flexibility, especially if those values are compromised (turn out to be weak).

  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.