A Usage for Shared Resources in RELOAD (ShaRe)

draft-knauf-p2psip-share-00

Alexander Knauf
Gabriel Hege
Thomas Schmidt
Matthias Wählisch

alexander.knauf@haw-hamburg.de, hege@fhtw-berlin.de, {t.schmidt,waehlisch}@ieee.org
Outline

1. Problem Statement and Objectives

2. Overview Shared Resources

3. Access Control

4. Variable Resource Names

5. Conclusion & Outlook
Problem Statement

Why do we need Shared Resources in RELOAD?

- Standard access control mechanisms are not sufficient for controlled write access by multiple peers
- Simplest way: USER-MATCH policy and certificate with same user name for all peers
  - Need to contact enrollment server → infeasible
  - Need to distribute private key/secrets/certificate
  - No individual revocation
- Use cases:
  - conference registration, message board, SSM source announcement, ...
Objectives

- Single resource to be writable by a well defined group of peers
  - Without contacting enrollment server
  - Allow revocation
- Optionally: more relaxed resource naming scheme
- Define some primitives for other Usages to build upon
Shared Resources - Overview

- RELOAD Resource (Kind) for which multiple peers have write access
- Resource Owner: has access by some (standard) policy (e.g., USER-MATCH)
- Resource Owner grants access using an Access Control List (ACL)
- ACL is stored under the same Resource-ID
  - on the same peer
- Write permission may be further delegated
  - Chain of delegations in ACL
Access Control Policies

• For the Owner:
  • Standard policy (e.g., USER-MATCH)
    – or relaxation thereof: USER-PATTERN-MATCH
  • Allows the Owner to store the ACL

• For other peers:
  • USER-CHAIN-ACL

• Enforced by the storing peer, but independently verifiable
Access Control List

- Stored under the same Resource Name as the Shared Resource
- Contains delegations from_user → to_user
- Users in the ACL may write the Shared Resource
- Chain of signed delegations may be independently verified

```
struct {
    opaque resource_name<0..2^16-1>;
    KindId kind;
    opaque from_user<0..2^16-1>;
    opaque to_user<0..2^16-1>;
    Boolean allow_delegation;
} AccessListData;
```
Revocation of Write Permission

Revocation is simple:

- Invalidate corresponding delegation in ACL
  - set exists=false
- Succeeding delegations also invalidated
- Owner can revoke the whole list by deleting the root entry
## Access Control List – Example

<table>
<thead>
<tr>
<th>#</th>
<th>Array Entries</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Kind:1234 from:Owner -&gt; to:Owner ad:1</td>
<td>signed by Owner</td>
</tr>
<tr>
<td>1</td>
<td>Kind:1234 from:Owner -&gt; to:Alice ad:1</td>
<td>signed by Owner</td>
</tr>
<tr>
<td>2</td>
<td>Kind:1234 from:Alice -&gt; to:Bob ad:0</td>
<td>signed by Alice</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>42</td>
<td>Kind:4321 from:Owner -&gt; to:Owner ad:1</td>
<td>signed by Owner</td>
</tr>
<tr>
<td>43</td>
<td>Kind:4321 from:Owner -&gt; to:Carol ad:0</td>
<td>signed by Owner</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Requirements for Using Shared Resources

- Separated Data Storage
  - Each element MUST be exclusively maintained by its creator
    → Kind MUST use a RELOAD data model consisting of individual objects (e.g. array or dictionary)

- Access Control Policy
  - Usage MUST permit the USER-CHAIN-ACL policy

- user_name field
  - Kind data structure MUST contain the user_name field
Variable Resource Names

• Extends the set of allowed Resource Names for a peer with a given user name
  → Relaxation of USER-MATCH policy
• Resource Names still closely related to Owner's user name
• Regular expression defines the allowed Resource Names for a Kind in the configuration document:

```
<variable-resource-names>
  <pattern kind="DISCO-REGISTRATION">
    .*-conf-\$USER@$DOMAIN
  </pattern>
</variable-resource-names>
```
Conclusion & Outlook

- Defined primitives to allow coordinated shared writing of a RELOAD resource
- Defined a relaxed resource naming scheme

- Now we need some drafts using these primitives ;-) (see draft-knauf-p2psip-disco-02)
- Use CGIs as an additional option for resource names
- Next version could use ECMAScript to define access policies as in draft-petithuguenin-p2psip-access-control
Thank you for your attention!

Any Questions?