Motivation
State of Delegated Mappings

Current delegated mapping systems have:

- **Centralized trust**
  - Often, with many trusted authorities that can maliciously rebind mappings.

- **Unilateral revocation**
  - Mappings can be revoked or overwritten by authorities without consent of the delegee.

- **No common interface**
  - Multiple systems exist to solve the problem of having authenticated delegated mappings in domain-specific ways.
## Delegated Mapping Use Cases

<table>
<thead>
<tr>
<th>System</th>
<th>Mapping</th>
<th>Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain Mappings (DNS/DNSSEC)</td>
<td>Domain → Zone</td>
<td><strong>Domain roots are trusted</strong> and can remap existing entries.</td>
</tr>
<tr>
<td>Public Key Infrastructure (CA trust chains)</td>
<td>Domain → Certificate</td>
<td><strong>CAs are trusted</strong> and can issue malicious certificates.</td>
</tr>
<tr>
<td>Web Security Policy (HSTS preload list)</td>
<td>Domain → Policy</td>
<td><strong>Not scalable.</strong> Large lists. Policy tied to browser versions.</td>
</tr>
<tr>
<td>E2E encryption (Privately managed,</td>
<td>User → Key</td>
<td><strong>Provider is trusted</strong> and can remap public key directories. Offline verification required to be secure.</td>
</tr>
<tr>
<td>non-transparent directories)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP Address Allocation</td>
<td>IP Address → Key</td>
<td><strong>AS trust each other.</strong> Malicious AS can announce route that it does not own.</td>
</tr>
</tbody>
</table>
Generalized Mappings

Can we derive a scalable solution that will work for any mapping?

Generic mapping interface with built-in authentication and delegation

Public, append-only log with enforced well-formed transitions.

<table>
<thead>
<tr>
<th>DNS</th>
<th>PKI</th>
<th>E2E Directories</th>
<th>IP Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delegated Distributed Mappings</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Consensus Layer (e.g. SCP)
Delegated Mappings
Building Blocks

- **Cell** - basic record of delegation

  - `create_time`
  - `revision_time`
  - `commitment_time`

  Guarantee from *delegator* to *deleee* that the mapping will remain valid until the `commitment_time`.

  Protection from arbitrary delegator actions.
Building Blocks

- **ValueCell** - resolves lookups for individual values

  ValueCell
  - create_time
  - revision_time
  - commitment_time

  Value
  - owner_key
  - sig
  - value

  Ties namespace value to owner public key.

  Signature of cell owner

  Application-specific content:
  - email encryption key
  - domain zone file
  - binary hash
**Building Blocks**

- **DelegateCell** - authorizes delegee to manage namespace

  - create_time
  - revision_time
  - commitment_time

  **Delegation**
  - delegee_key
  - authority_sig
  - namespace

Links management of namespace *range* to delegee key.

Delegator signature

Specific namespace range:
- 192.0.2.0/24
- .com
Building Blocks

- **Table** - maps lookup keys in a namespace to individual cells

Delegee key controls delegations (i.e. table modifications) within the namespace range it controls

Determines delegation rules: **PREFIX, SUFFIX**

Keys must be part of the table namespace
Building Blocks

- Root Key Listing - stores root key for each namespace

<table>
<thead>
<tr>
<th>roots</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace_root_key1</td>
</tr>
<tr>
<td>.namespace_root_keyn</td>
</tr>
</tbody>
</table>

Key and app_id uniquely identify root table for specific namespace.
Structure

- The chain of delegated keys for a namespace links each table

```
.. namespace_key ..
```

- **Table1**
  - `type`
  - `entries`
    - `key1`

- **Table2**
  - `type`
  - `entries`
    - `key2`

- **Table3**
  - `type`
  - `entries`
    - `key3`
Structure

- Domain Delegation

```
.. dns_key ..
```

```
Root
  SUFFIX
  entries
    .com -> Registrar1

Registrar1
  SUFFIX
  entries
    .a.com -> A & Co.

A & Co.
  SUFFIX
  entries
    x.a.com -> <zone file>
```
Structure

- IP Prefix Delegation

```
.. | ip_del_key | ..
----|-------------|
```

```
Root
  PREFIX
  entries
    1.0.0.0 /8
    | ARIN

ARIN
  PREFIX
  entries
    1.9.0.0 /16
    | Stanford
    1.9.5.1 /32
      | Sydney's key

Stanford
  PREFIX
  entries
    1.9.5.1 /32
```
Common Operations

- **Creating a delegation**
  - The table authority inserts a new `DelegateCell` for the delegee in its table and creates a new table for the delegee
  - Proof of the additions is passed to the consensus algorithm

- **Key rotation**

- **Lookup**
Common Operations

- Recording a delegation
- Key rotation
  - The owner rotates their own key in the cell and signs with the old key
- Lookup
Common Operations

- Recording a delegation
- Key rotation
- **Lookup**
  - Read root key from Root Key Listing and find root table
  - Access table for delegation with the prefix/suffix match recursively
  - Return the contents of the desired ValueCell or NULL
Consensus

- Safety and consistency of the delegation tables must be provided by a consensus algorithm
  - Enforcement of table and delegation semantics
- Can use any consensus protocol (but not all algorithms are created equal)
- SCP
  - Proofs of table updates are batched and agreed to between SCP nodes
  - Authorized/correct modifications are determined by a distributed-mapping-specific \textit{validity function} on each node.
Considerations
Considerations

Who controls the root namespace?

- DNSSEC?
  - IANA root keys
- A set of n authorities, or k of n authorities (threshold)?
- Vote amongst existing root authorities?
Considerations

How do we prevent spam?

- Explicit delegation quotas in state machine
- Each delegator responsible for direct sub-delegates
Considerations

How do we scale?

● Who needs to run full consensus validation nodes?
● Scaling to per-user magnitude?