Distributed Authenticated Mappings
DINRG IETF-101

Sydney Li\textsuperscript{1}, Colin Man\textsuperscript{2}, Jean-Luc Watson\textsuperscript{2}

\textsuperscript{1}Electronic Frontier Foundation, \textsuperscript{2}Stanford University
I. Authenticated Mappings
What’s broken?

Problem
Private conversations over encrypted email

Secure internet service for small websites

Domain lookups

Verifying identity
<table>
<thead>
<tr>
<th>Problem</th>
<th>Point Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private conversations over encrypted email</td>
<td>Trusted keyservers</td>
</tr>
<tr>
<td>Secure internet service for small websites</td>
<td>HSTS preload lists</td>
</tr>
<tr>
<td>Domain lookups</td>
<td>DNS (+ DNSSEC)</td>
</tr>
<tr>
<td>Verifying identity</td>
<td>CA trust chains + CT</td>
</tr>
</tbody>
</table>
## What’s broken?

<table>
<thead>
<tr>
<th>Problem</th>
<th>Point Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private conversations over encrypted email</td>
<td>Trusted keyservers</td>
</tr>
<tr>
<td></td>
<td>// MITM</td>
</tr>
<tr>
<td>Secure internet service for small websites</td>
<td>HSTS preload lists</td>
</tr>
<tr>
<td></td>
<td>// Downgrade attacks</td>
</tr>
<tr>
<td>Domain lookups</td>
<td>DNS (+ DNSSEC)</td>
</tr>
<tr>
<td></td>
<td>// Poisoning; low adoption</td>
</tr>
<tr>
<td>Verifying identity</td>
<td>CA trust chains + CT</td>
</tr>
<tr>
<td></td>
<td>// Single point of failure</td>
</tr>
<tr>
<td>Problem</td>
<td>Point Solution</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>Private conversations over encrypted email</td>
<td>Trusted keyservers</td>
</tr>
<tr>
<td></td>
<td>// MITM</td>
</tr>
<tr>
<td>Secure internet service for small websites</td>
<td>HSTS preload lists</td>
</tr>
<tr>
<td></td>
<td>// Downgrade attacks</td>
</tr>
<tr>
<td>Domain lookups</td>
<td>DNS (+ DNSSEC)</td>
</tr>
<tr>
<td></td>
<td>// Poisoning; low adoption</td>
</tr>
<tr>
<td>Verifying identity</td>
<td>CA trust chains + CT</td>
</tr>
<tr>
<td></td>
<td>// Single point of failure</td>
</tr>
</tbody>
</table>
What’s broken?

<table>
<thead>
<tr>
<th>Problem</th>
<th>Point Solution</th>
<th>Need</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private conversations over encrypted email</td>
<td>Trusted keyservers</td>
<td>Public key mappings</td>
</tr>
<tr>
<td>Secure internet service for small websites</td>
<td>HSTS preload lists</td>
<td>Policy mappings</td>
</tr>
<tr>
<td>Domain lookups</td>
<td>DNS (+ DNSSEC)</td>
<td>Name mappings</td>
</tr>
<tr>
<td>Verifying identity</td>
<td>CA trust chains + CT</td>
<td>Certificate mappings</td>
</tr>
</tbody>
</table>

// MITM
// Downgrade attacks
// Poisoning; low adoption
// Single point of failure

Authenticated mappings!
Generalized Mappings

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

Idea: infrastructure for a global state database
- Append-only
Generalized Mappings

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

Idea: infrastructure for a global state database
- Append-only
- Well-formed transitions (more on this later)
Generalized Mappings

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

**Idea:** infrastructure for a global state database
- Append-only
- Well-formed transitions *(more on this later)*
- Transparent
Global State Database

(1) Bootstrap Certificate Transparency

Incentive and priority mismatch. Lack of knowledge to enforce domain specific semantics.
Global State Database

(1) Bootstrap Certificate Transparency
Incentive and priority mismatch. Lack of knowledge to enforce domain specific semantics.

(2) Byzantine Fault Tolerant Cluster
Limited participation. Uniform set of incentives undermines security.

KeyNet (interim meeting)
Distributed OpenPGP key store for encrypted email
## Global State Database

<table>
<thead>
<tr>
<th>(1) Bootstrap Certificate Transparency</th>
<th>(2) Byzantine Fault Tolerant Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incentive and priority mismatch.</td>
<td>Limited participation.</td>
</tr>
<tr>
<td>Lack of knowledge to enforce domain</td>
<td>Uniform set of incentives undermines</td>
</tr>
<tr>
<td>specific semantics.</td>
<td>security.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(3) Proof-of-{Work, Stake}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open membership w/out accountability.</td>
</tr>
<tr>
<td>Trust is tied to hashing power or</td>
</tr>
<tr>
<td>available resources.</td>
</tr>
</tbody>
</table>
Global State Database

(1) Bootstrap Certificate Transparency
Incentive and priority mismatch.
Lack of knowledge to enforce domain specific semantics.

(2) Byzantine Fault Tolerant Cluster
Limited participation.
Uniform set of incentives undermines security.

(3) Proof-of-{Work, Stake}
Open membership w/out accountability.
Trust is tied to hashing power or available resources.

(4) Federated Byzantine Agreement
Variety of well-known stakeholders.
Trust in network is tied to real-world trust relationships.
II. Well-formed Transitions
Example 1: PGP Keys

We might want to securely map *aliases* to *public keys*. 

On creation of an entry, we can check that a domain authority verifies their identity.

Every time an entry is updated, we should verify:

1. the previous public key has signed the update.

   OR

2. $n$ of $m$ trusted parties have signed the update.
Example 2: Binary Hashes

We might want to securely map download URLs to binary hashes.

On creation of an entry, we should check that the domain hosting the URL has signed the entry.

Every time an entry is updated, we should maintain

1. the same domain has signed the update.
Observations

A mapping abstraction allows for shared components

- Entry **create** and **update** validation based on local state
- External authentication
- Ownership
- Recovery/threshold cryptography
Mapping: specification for key-value mapping with **validators** to ensure well-formed, secure entry **creation** and **updates**.

- All creation validators must succeed to allow a new entry
- All update validators must succeed to allow a transition/change
Mapping Example: PGP Keys

```python
mapping = Mapping {
    create_validator = [ "bootstrap_validator": Validator {...} ],
    update_validator = [ "identity_validator": Validator {...} ],
    key_type = ALIAS,
    value_type = PUBLIC_KEY
}
```
Validators: collections of operations enforced on entry creation/update
- At least one must succeed for validation to pass
- create and update validators defined at the mapping level

Validator Example: PGP Keys

"identity_validator": Validator {
  operation = [
    // require existing signature for updates
    "owner": ...
    // allow threshold encryption for recovery
    "multisig": ...
  ]
}
Operations: validation rules enforced on each entry in a mapping

- Allowed operations in **Validators** are specified at **mapping** level
- Individual entries can customize operation parameters
- Example Operations
  - OpCASignature, OpOwnerSignature, OpNofMSignatures

**Operation Example: PGP Keys**

Validator { operations = [
  "owner": OpOwnerSignature { }
  "multisig": OpNOfMSignatures {
    alias = ["eff.org", "mozilla.org", "ietf.org"]
    required_number = 2
  }
]}

EntryUpdates: changes to a mapping entry
- All validators are evaluated and must pass for update to succeed

Entry Update Example: PGP Keys
EntryUpdate {
  mapping_id = "keynet",
  key = Alias { Email {
    address = "colinman@stanford.edu"
    domain = "stanford.edu"
  } }
  value = "{new public key here}"
  update_operations = {optional parameters}
}
Mapping Abstraction vs Smart Contracts

- Easy to implement on top of consensus layer
- Easy to use (operations already defined)
- Less error-prone (Parity, Dao, etc.)
- Designed to be bootstrapped off existing trust infrastructure
  - Exploring options to use Stellar Consensus Protocol
Questions?

https://github.com/colinman/keynet