Towards a Security Model for Internet of Things (IoT)

Ankur Taly, Google Inc.

November 2015
Internet of Things (IoT)

Physical devices made accessible over the network.

Exciting new possibilities!!

img source: http://www.ti.com/lsds/media/images/wireless_connectivity/50BillionThings.png
Internet of Things Security!

Goldmine for the bad guys!!

Scary new possibilities!!
This is really scary!!

Live feed from an airplane hangar in Norway!!

Found using shodan.io --- a search engine for finding devices (IoT), e.g., routers, servers, cameras, SCADA systems, HVAC systems etc.

source: http://img.wonderhowto.com/img/original/32/45/63534020036048/0/635340200360483245.jpg
In popular press

WND EXCLUSIVE
'HACKERS REMOTELY KILL A JEEP ON THE HIGHWAY'

Several baby monitors vulnerable to hacking, cybersecurity firm warns

Refrigerator Busted Sending Spam Emails In Massive Cyberattack

Hackers Can Remotely Hack Self-Aiming Rifles to Change Its Target
Top IoT Vulnerabilities

HP Study Reveals 70 Percent of Internet of Things Devices Vulnerable to Attack

danielmiessler 07-29-2014 05:09 AM - edited 07-07-2015 12:33 PM

- Insufficient Authentication and Authorization (80%)
- Lack of Transport Encryption (70%)
- Insecure Web Interfaces (60%)
- Insecure Software Updates (60%)
- Insecure Defaults (70%)
Top IoT Vulnerabilities

HP Study Reveals 70 Percent of Internet of Things Devices Vulnerable to Attack

danielmiessler 07-29-2014 05:09 AM - edited 07-07-2015 12:33 PM

- Insufficient Authentication and Authorization (80%)
- Lack of Transport Encryption (70%)
- Insecure Web Interfaces (60%)
- Insecure Software Updates (60%)
- Insecure Defaults (70%)

**Security First:** Bake in security mechanisms from the ground up.
## IoT Security Requirements

<table>
<thead>
<tr>
<th>Identity and Authorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutual authorization</td>
</tr>
<tr>
<td>Channel integrity and confidentiality</td>
</tr>
<tr>
<td>Forward secrecy</td>
</tr>
<tr>
<td>Fine-grained sharing and delegation</td>
</tr>
<tr>
<td>Revocation and auditing</td>
</tr>
</tbody>
</table>
## IoT Security Requirements

### Identity and Authorization
- Mutual authorization
- Channel integrity and confidentiality
- Forward secrecy
- Fine-grained sharing and delegation
- Revocation and auditing

### Device Protection
- No remote code execution
- Automatic and secure updates
- Verified boot
Private Discovery

Nearby Devices

- Samsung TV: 4m
  Channel Guide | Setup
- Lighting Control: 4m
  Toggle | Brightness
- Nest Control: 6m
  Temperature | Settings
- Security System: 7m
  Status | Control
- Door Lock: 7m
  Status | Control

courtesy: physicalweb.org
Private Discovery: Devices must be discoverable only by authorized parties.
### IoT Security Requirements

<table>
<thead>
<tr>
<th>Identity and Authorization</th>
<th>Device Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutual authorization</td>
<td>No remote code execution</td>
</tr>
<tr>
<td>Channel integrity and confidentiality</td>
<td>Automatic and secure updates</td>
</tr>
<tr>
<td>Forward secrecy</td>
<td>Verified boot</td>
</tr>
<tr>
<td>Fine-grained sharing and delegation</td>
<td></td>
</tr>
<tr>
<td>Revocation and auditing</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Privacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private discovery</td>
</tr>
<tr>
<td>Anonymous communication</td>
</tr>
<tr>
<td>Transparency</td>
</tr>
<tr>
<td>Identity and Authorization</td>
</tr>
<tr>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>Mutual authorization</td>
</tr>
<tr>
<td>Channel integrity and confidentiality</td>
</tr>
<tr>
<td>Forward secrecy</td>
</tr>
<tr>
<td>Fine-grained sharing and delegation</td>
</tr>
<tr>
<td>Revocation and auditing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Device Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>No remote code execution</td>
</tr>
<tr>
<td>Automatic and secure updates</td>
</tr>
<tr>
<td>Verified boot</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Privacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private discovery</td>
</tr>
<tr>
<td>Anonymous communication</td>
</tr>
<tr>
<td>Transparency</td>
</tr>
</tbody>
</table>
Plan

○ Vanadium Security Model
  ○ Identity (Principals and Blessings)
  ○ Authentication and Authorization
○ Case Study: Smart Lock
○ Practicalities
○ Conclusions
Vanadium

Open source, cross-platform framework for building secure, multi-device applications

- Decentralized
  "Talk when possible" $\Rightarrow$ Peer-to-peer communication

- Secure
  Mutual authentication and authorization
  Fine-grained delegation & auditing
Mechanics

○ Go, Java and JavaScript libraries

○ Open-source

○ All we talk about in these slides is “code-complete”
Design Docs, Tutorials, Code

- Homepage: v.io
- Concepts: v.io/concepts
- Tutorials: v.io/tutorials
- Source: vanadium.googlesource.com
Vanadium Security Model

Principals & Blessings
principal

○ An entity that can communicate externally
  ○ **Fine-grained**: Each App/Process/Service is a different principal

○ Has a unique public/private key pair (P, S)
  ○ Private key is **never** shared over the network
  ○ Ideally held in a TPM on the device
Blessings

- Each principal has a set of **hierarchical human-readable strings** bound to it, called *blessings*

  Example: Alice’s television \( (P_t, S_t) \) may have blessings:
  - google/alice/hometv
  - samsung/products/tv/123

- Principals are authenticated and authorized based on their blessings. Example: Alice’s devices authorize all principals with blessing names matching **google/alice/***
Blessings

○ Each principal has a set of hierarchical human-readable strings bound to it, called blessings.
Example: Alice’s television (P_{tv}, S_{tv}) may have blessings:
  ○ google/alice/hometv
  ○ samsung/products/tv/123

○ Principals are authenticated and authorized based on their blessings. Example: Alice’s devices authorize all principals with blessing names matching google/alice/*

Simple Distributed Security Infrastructure (SDSI), Lampson and Rivest, 1996
Blessings

- Blessings are certificate chains bound to the principal’s public key
- Each certificate has a Name, PublicKey, Caveats and Signature

Very simple certificate format!
The Bless Operation

Extend one of your Blessings and bind it to another principal.

\[ (P_{\text{Alice}}, S_{\text{Alice}}) \] Bless \[ (P_{\text{TV}}, S_{\text{TV}}) \]

<table>
<thead>
<tr>
<th>google</th>
<th>alice</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_{\text{Google}} )</td>
<td>( P_{\text{Alice}} )</td>
</tr>
<tr>
<td><strong>Till 6/30/2016</strong></td>
<td><strong>Till 11/23/2015</strong></td>
</tr>
<tr>
<td>Signed by ( S_{\text{Google}} )</td>
<td>Signed by ( S_{\text{Google}} )</td>
</tr>
</tbody>
</table>
Extend one of your Blessings and bind it to another principal.

\[(P_{\text{Alice}}, S_{\text{Alice}}) \overset{\text{Bless}}{\longrightarrow} (P_{\text{TV}}, S_{\text{TV}})\]

- **google**
  - \(P_{\text{Google}}\)
  - Till 6/30/2016
  - Signed by \(S_{\text{Google}}\)

- **alice**
  - \(P_{\text{Alice}}\)
  - Till 11/23/2015
  - Signed by \(S_{\text{Google}}\)

- **hometv**
  - \(P_{\text{TV}}\)
  - Till 11/23/2015: 6PM
  - Signed by \(S_{\text{Alice}}\)
The Bless Operation

Extend one of your Blessings and bind it to another principal.

\((P_{\text{Alice}}, S_{\text{Alice}})\) Bless \((P_{\text{TV}}, S_{\text{TV}})\)

Dynamic Identity Creation OR Bound Capability Grant!
Blessings: Auditability and Binding

Blessings:
- Are bound to a private key that never leaves the device
- Can only be delegated by extending to other private keys
- Encapsulate an auditable delegation trail
- Examples:
  - google/alice (AccountManager app on Alice’s phone)
  - google/alice/hometv (DeviceManager app on Alice’s TV)
  - google/alice/hometv/youtube (Youtube app on Alice’s TV)
But Alice wants her TV to **only** access Youtube, NOT her Bank!

<table>
<thead>
<tr>
<th>Caveats</th>
</tr>
</thead>
<tbody>
<tr>
<td>google</td>
</tr>
<tr>
<td>P\textsubscript{Google}</td>
</tr>
<tr>
<td><strong>Till 6/30/2016</strong></td>
</tr>
<tr>
<td>Signed by S\textsubscript{Google}</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>alice</td>
</tr>
<tr>
<td>P\textsubscript{Alice}</td>
</tr>
<tr>
<td><strong>Till 11/23/2015</strong></td>
</tr>
<tr>
<td>Signed by S\textsubscript{Google}</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>hometv</td>
</tr>
<tr>
<td>P\textsubscript{TV}</td>
</tr>
<tr>
<td><strong>Till 11/23/2015: 6PM</strong></td>
</tr>
<tr>
<td>Signed by S\textsubscript{Alice}</td>
</tr>
</tbody>
</table>
But Alice wants her TV to *only* access Youtube, NOT her Bank!

Specify arbitrary restrictions here
But Alice wants her TV to *only* access Youtube, NOT her Bank!

TV has the name `google/alice/hometv`
- *as long as the time is before 11/23/2015: 6PM*
- *as long as the service being accessed is google/youtube*
But Alice wants her TV to only access Youtube, NOT her Bank!

TV has the name google/alice/hometv

- as long as the time is before 11/23/2015: 6PM
- as long as the service being accessed is google/youtube

Macaroons: Cookies with Caveats for Decentralized Authorization, Politz et al., NDSS’14
Caveats are powerful

- Services can define their own caveats
  - Bless the Valet such that:
    - valet is only authorized to drive for < 5 miles
    - only for the next 3 hours
    - cannot access trunk or infotainment system
    - but can access GPS

- Validated by the target service (first-party) when the blessing is used to make a request (first-party caveats)
Third-party Caveats

- Caveats that must be validated by a specific third-party
- Target service (first-party) only expects a “discharge” (proof) that the caveat has been validated by the specific third-party
Third-party Caveats

- Caveats that must be validated by a specific third-party.
- Target service (first-party) only expects a “discharge” (proof) that the caveat has been validated by the specific third-party.

Third-party Caveat

ID: <content hash>

Restriction: within 10m proximity

Loc: google/alice/proximity

Verification Key: $P_{Proximity}$
Third-party Caveats

- Caveats that must be validated by a specific third-party
- Target service (first-party) only expects a “discharge” (proof) that the caveat has been validated by the specific third-party

**Third-party Caveat**

- **ID**: <content hash>
- **Restriction**: within 10m proximity
- **Loc**: google/alice/proximity
- **Verification Key**: $P_{\text{Proximity}}$

**Third-party Discharge**

- **ID**: <same as caveat.ID>
- **Caveat**: for next 1 minute
- **Signed by**: $S_{\text{Proximity}}$
Mechanics

(P_{Proximity}, S_{Proximity})

Alice’s proximity discharger

(P_{Bob}, S_{Bob})

google
... 
... 
... 

alice
... 
... 
... 

houseguest
... 
... 

(proximity caveat)
Mechanics

Alice’s proximity discharger

(P_{Proximity}, S_{Proximity})

(proximity caveat)

(P_{Bob}, S_{Bob})

(Alice’s proximity discharger)

(P_{TV}, S_{TV})

google

... alice

... houseguest

...
Mechanics

Perform proximity checks

(P_{Proximity}, S_{Proximity})

Alice’s proximity discharger

(P_{Bob}, S_{Bob})

(proximity caveat)

TV

(P_{TV}, S_{TV})

google
...  
...  
...  

...  
...  
...  

...  
...  
...  

...  
...  
...  

houseguest

(proximity caveat)
Mechanics

Alice’s proximity discharger

(P_{Proximity}, S_{Proximity})

(P_{Bob}, S_{Bob})

(P_{TV}, S_{TV})

google

alice

houseguest

proximity caveat

proximity discharge
Mechanics

Alice’s proximity discharger

($P_{Proximity}$, $S_{Proximity}$)

($P_{Bob}$, $S_{Bob}$)

($P_{TV}$, $S_{TV}$)

google
alice
houseguest

proximity caveat

proximity discharger

proximity caveat

proximity discharge

+
Third-party Caveat Examples

○ Revocation
  ○ Third-party caveats provide a natural refresh mechanism.
  ○ Subsumes *Online Certificate Status Protocol (OCSP)*

○ Social network
  ○ G+ must assert membership in “work” circle

○ Parental controls
  ○ Kids can watch TV only if Mom approves
Validating Blessings

How does the TV validate Bob’s blessings?

- google
  - $P_{Google}$
  - Till 6/30/2016
  - Signed by S_{Google}

- alice
  - $P_{Alice}$
  - Till 6/30/2016
  - Signed by S_{Google}

- houseguest
  - $P_{Bob}$
  - Till 7/4/2015
  - TPCaveat: $P_{Proximity}$
  - Signed by S_{Alice}

+ TPDischarge
Validating Blessings

How does the TV validate Bob’s blessings?

1. Verify Certificate Signatures

<table>
<thead>
<tr>
<th>google</th>
<th>alice</th>
<th>houseguest</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_{Google}</td>
<td>P_{Alice}</td>
<td>P_{Bob}</td>
</tr>
<tr>
<td>Till 6/30/2016</td>
<td>Till 6/30/2016</td>
<td>Till 7/4/2015</td>
</tr>
<tr>
<td>Signed by S_{Google}</td>
<td>Signed by S_{Google}</td>
<td>Signed by S_{Alice}</td>
</tr>
<tr>
<td>+ TP\text{Discharge}\</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TP\text{Caveat: } P_{Proximity}
Validating Blessings

How does the TV validate Bob’s blessings?

1. Verify Certificate Signatures
2. Validate all first-party and third-party caveats
Validating Blessings

How does the TV validate Bob’s blessings?

1. Verify Certificate Signatures
2. Validate all first-party and third-party caveats
3. Verify that the blessing root is recognized
Validating Blessings

How does the TV validate Bob’s blessings?

1. Verify Certificate Signatures
2. Validate all first-party and third-party caveats
3. Verify that the blessing root is recognized

Bob can be recognized as google/alice/houseguest
All communication must be encrypted, mutually authenticated and authorized.
Authentication and Authorization

Client: Initiator of request
Server: Responder of request

Mutual Authentication
○ Each end *learns* the other end’s blessings and is convinced that the other end possesses the corresponding private key

Mutual Authorization
○ Each end validates the other end’s blessings and evaluates the blessing names against an authorization policy
Mutual Authentication Protocol

SIGMA: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman, Krawczyk et al., CRYPTO'03

Client

Diffie-Hellman (DH) Exchange

Server

\[ g^x \]

\[ g^y \]

Derive (authenticated-encryption) key k from DH secret
Mutual Authentication Protocol

**SIGMA**: The 'SIgn-and-MAc' Approach to Authenticated Diffie-Hellman, Krawczyk et al., CRYPTO'03

- **Client**: Bob learns Blessings\textsubscript{TV} and authorizes them.
- **Server**: Derive (authenticated-encryption) key $k$ from DH secret.

**Diffie-Hellman (DH) Exchange**

- $g^x$ sent from Client to Server
- $g^y$ returned from Server to Client

**Key Derivation**

- { Blessings\textsubscript{TV}, $\text{Sign}_{TV}(\langle"s","g^x, g^y\rangle)$ }\textsubscript{k}
**Mutual Authentication Protocol**

**SIGMA**: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman, Krawczyk et al., CRYPTO'03

---

**Diffie-Hellman (DH) Exchange**

Client

\[ g^x \]

\[ g^y \]

Server

Derive (authenticated-encryption) key \( k \) from DH secret

{ Blessings_{TV}, \text{Sign}_{TV}(\langle "s", g^x, g^y \rangle) \}_k

Bob learns Blessings_{TV} and authorizes them

{ Blessings_{Bob}, \text{Sign}_{Bob}(\langle "c", g^x, g^y \rangle) \}_k

TV learns Blessings_{Bob} and authorizes them
Mutual Authentication Protocol

**SIGMA**: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman, Krawczyk et al., CRYPTO'03

---

**Client**

Bob learns Blessings\textsubscript{TV} and authorizes them

---

**Server**

TV learns Blessings\textsubscript{Bob} and authorizes them

---

**Diffie-Hellman (DH) Exchange**

\[ g^x \]
\[ g^y \]

Derive (authenticated-encryption) key k from DH secret

\[ \{ \text{Blessings}_{\text{TV}}, \text{Sign}\textsubscript{TV}(<"s", g^x, g^y>) \}_k \]

\[ \{ \text{Blessings}_{\text{Bob}}, \text{Sign}\textsubscript{Bob}(<"c", g^x, g^y>) \}_k \]

---

**Server presents its blessings before the client**
Mutual Authentication Protocol

**SIGMA**: The 'SIGn-and-MAc' Approach to Authenticated Diffie-Hellman, Krawczyk et al., CRYPTO'03

Client

```
Bob learns Blessings_{TV} and authorizes them
```

Server

```
TV learns Blessings_{Bob} and authorizes them
```

Diffie-Hellman (DH) Exchange

```
\{ Blessings_{TV}, Sign_{TV}(\langle s, g^x, g^y \rangle) \}_k
```

```
\{ Blessings_{Bob}, Sign_{Bob}(\langle c, g^x, g^y \rangle) \}_k
```

```
Formally verified in ProVerif
```

Server presents its blessings before the client
Private Mutual Authentication*

- Neither the server nor the client wants to present its blessings first.
  - I will only reveal my name to google/alice/hometv
  - I only reveal my name to delegates of google/alice

- How do we resolve this deadlock?
Private Mutual Authentication*

- Neither the server nor the client wants to present its blessings first.
  - I will only reveal my name to google/alice/hometv
  - I only reveal my name to delegates of google/alice
- How do we resolve this deadlock?

Relevant cryptography research (carried out in the context of secret agents)

- Oblivious Signature-Based Envelope - Li et al., PODC‘03
- Secret Handshakes from Pairing-Based Key Agreements - Balfanz et al., S&P‘03
Authorization policies are based on blessing names.
Authorization policies are based on blessing names.

1) Verify certificate signatures
2) Validate caveats
3) Verify blessing roots
Authorization policies are based on blessing names.

1) Verify certificate signatures
2) Validate caveats
3) Verify blessing roots
Authorization policies are based on blessing names.

1) Verify certificate signatures
2) Validate caveats
3) Verify blessing roots

Verify that blessing name satisfies the authorization policy (e.g., ACL)

PASS

FAIL
Access Control Lists (ACL)

Set of authorized blessings specified using explicit ACLs.

Alice’s TV

<table>
<thead>
<tr>
<th>Label</th>
<th>Blessing Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin</td>
<td><code>in: google/alice</code></td>
</tr>
<tr>
<td>Photos</td>
<td><code>in: google/alice/*</code></td>
</tr>
<tr>
<td>Movies</td>
<td><code>in: google/alice/spouse</code></td>
</tr>
</tbody>
</table>
Set of authorized blessings specified using explicit ACLs.

**Alice’s TV**

<table>
<thead>
<tr>
<th>Label</th>
<th>Blessing Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin</td>
<td><code>In: google/alice</code></td>
</tr>
<tr>
<td>Photos</td>
<td><code>In: google/alice/*</code></td>
</tr>
<tr>
<td>Movies</td>
<td><code>In: google/alice/spouse</code></td>
</tr>
</tbody>
</table>

*Note: google/alice/houseguest can browse photos but cannot watch movies.*
## Access Control Lists (ACL)

Set of authorized blessings specified using explicit ACLs.

### Alice’s TV

<table>
<thead>
<tr>
<th>Label</th>
<th>Blessing Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin</td>
<td><strong>In: google/alice</strong></td>
</tr>
<tr>
<td>Photos</td>
<td><strong>In: google/alice/</strong>*&lt;br&gt;<strong>NotIn: google/alice/housemaid</strong></td>
</tr>
<tr>
<td>Movies</td>
<td><strong>In: google/alice/spouse</strong></td>
</tr>
</tbody>
</table>

*google/alice/houseguest* can browse photos but cannot watch movies.
Distributed Groups

ACLs can also contain groups.

<table>
<thead>
<tr>
<th>Label</th>
<th>Blessing Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin</td>
<td>In: alice</td>
</tr>
<tr>
<td>Photos</td>
<td>In: g&lt;Carol Devices&gt;</td>
</tr>
<tr>
<td>Movies</td>
<td>In: alice/devices/*</td>
</tr>
</tbody>
</table>
ACLs can also contain groups.

<table>
<thead>
<tr>
<th>Label</th>
<th>Blessing Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin</td>
<td>In: alice</td>
</tr>
<tr>
<td>Photos</td>
<td>In: g&lt;Carol Devices&gt;</td>
</tr>
<tr>
<td>Movies</td>
<td>In: alice/devices/*</td>
</tr>
</tbody>
</table>

Reference Monitor

Group
Carol Devices
g<Carol Work Devices>
google/carol/devices/phone
google/carol/devices/tablet
ACLs can also contain groups.

<table>
<thead>
<tr>
<th>Label</th>
<th>Blessing Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin</td>
<td>In: alice</td>
</tr>
<tr>
<td>Photos</td>
<td>In: g&lt;Carol Devices&gt;</td>
</tr>
<tr>
<td>Movies</td>
<td>In: alice/devices/*</td>
</tr>
</tbody>
</table>

Distributed Authorization with Distributed Grammars, Abadi et al. PLABS’15
Case Study: Smart Lock
Why Smart Locks?

- Remote locking/unlocking.
- Keyless proximity-based access.
- Maintain an audit log of who got in.
- Mint new (virtual) keys and share with others.
- Some also have a camera that will take the visitors picture.
Lock Setup

Blessings
google/alice

Blessings
lockcorp/1234
Lock Setup

Blessings
google/alice

Out of Band
<token>, <LockWiFi>

Blessings
lockcorp/1234
Lock Setup

Blessings

| google/alice |

I am lockcorp/1234

Blessings

| lockcorp/1234 |

Authorization Policy

| Claim: Allow Everyone |
Lock Setup

Authorization Policy
Claim: Allow Everyone

Claim("alice-door", <token>, <Wifi>)

Blessings
google/alice

Blessings
lockcorp/1234
Lock Setup

Claim ("alice-door", <token>, <Wifi>)

Authorization Policy
Claim: Allow Everyone

Blessings
- google/alice
- lockcorp/1234

Create self-signed blessing with name alice-door
Lock Setup

Blessings

| google/alice |

Claim

Claim("alice-door", <token>, <Wifi>)

Authorization Policy

| Claim: Allow Everyone |

Blessings

| lockcorp/1234 alice-door |

Now I am alice-door
Lock Setup

Blessings
- google/alice
- alice-door/key

Claim: Allow Everyone

Blessings
- lockcorp/1234
- alice-door

Claim("alice-door", <token>, <Wifi>)
Lock Setup

**Claim**

Claim(“alice-door”, <token>, <Wifi>)

**Authorization Policy**

Lock: alice-door/key/*
Unlock: alice-door/key/*
Lock Setup

Claim ("alice-door", <token>, <Wifi>)

Authorization Policy
Lock: alice-door/key/*
Unlock: alice-door/key/*
Lock Setup

Claim

Authorization Policy

Lock: alice-door/key/*
Unlock: alice-door/key/*

Blessings

google/alice
alice-door/key

Blessings

lockcorp/1234
alice-door

Blessings

google/bob

Lock using alice-door/key
Unlock using alice-door/key

Claim("alice-door", <token>, <Wifi>)
Lock Setup

Blessings
- google/alice
  - alice-door/key

Claim("alice-door", <token>, <Wifi>)

Lock using alice-door/key

Unlock using alice-door/key

Authorization Policy

Lock: alice-door/key/*
Unlock: alice-door/key/*

Blessings
- lockcorp/1234
  - alice-door

Blessings
- google/bob
  - alice-door/key/bob
Lock Setup

Blessings
- google/alice
  alice-door/key

Authorization Policy
- Lock: alice-door/key/*
- Unlock: alice-door/key/*

Blessings
- google/bob
  alice-door/key/bob

Claim("alice-door", <token>, <Wifi>)

Lock using alice-door/key

Unlock using alice-door/key

Unlock using alice-door/key/bob

Blessings
- lockcorp/1234
  alice-door
Lock Setup

- **Works Offline**
  No internet access required to interact with the lock
- **Fully Decentralized**
  No cloud server controls access to all locks
- **Fine-grained Auditing**
  Each lock device can keep track of who accessed it (including delegation trail)
- **No bearer tokens involved**
Practicalities
Vanadium Identity Provider

○ We run our own identity provider that grants blessings by authenticating users using Google OAuth2.
  ○ Blessings are of the form dev.v.io/u/alice@gmail.com.
  ○ They carry third-party revocation caveats.

○ Other organizations (e.g., Facebook, Stanford University) can also become Identity Providers.
Blessings Management

- Devices and apps would accumulate multiple blessings over time.
- How should users visualize and grant blessings?
Blessings Management

- Devices and apps would accumulate multiple blessings over time.
- How should users visualize and grant blessings?

Vanadium Blessings Manager App
- UI for visualizing blessings
- Grant blessings over NFC, Bluetooth

Future work: Blessing mailbox in the cloud
Private Key Management

- Securely storing private keys on device.
- Many different hardware architectures and operating systems.
- Multiple private keys per device (one for each app).
Private Key Management

- Securely storing private keys on device.
- Many different hardware architectures and operating systems.
- Multiple private keys per device (one for each app).

An Approach: Use a security agent (e.g., Plan9’s factotum)
- Special process that holds private keys and performs crypto.
- May store private keys in a TPM, if available.
- May adjust itself based on the hardware.
Conclusions
Principal and Blessings
A principal is a unique public/private key pair with human-readable names bound to it

All communication is encrypted & mutually authenticated
Forward-secrecy safe protocol, client and service identity privacy

Blessing names based Authorization
Principals authenticated and authorized based on their blessing names

Fine-grained Delegation and Audit
Bind an extension of a blessing to another principal under caveats
## What goals have we met?

### Identity and Authorization
- Mutual authorization
- Channel integrity and confidentiality
- Forward secrecy
- Fine-grained sharing and delegation
- Revocation and auditing

### Device Protection
- No remote code execution
- Automatic and secure updates
- Verified boot

### Privacy
- Private discovery
- Anonymous communication
- Transparency
What goals have we met?

<table>
<thead>
<tr>
<th>Identity and Authorization</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutual authorization</td>
<td>✓</td>
</tr>
<tr>
<td>Channel integrity and confidentiality</td>
<td>✓</td>
</tr>
<tr>
<td>Forward secrecy</td>
<td>✓</td>
</tr>
<tr>
<td>Fine-grained sharing and delegation</td>
<td>✓</td>
</tr>
<tr>
<td>Revocation and auditing</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Device Protection</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No remote code execution</td>
<td>🚫</td>
</tr>
<tr>
<td>Automatic and secure updates</td>
<td>🚫</td>
</tr>
<tr>
<td>Verified boot</td>
<td>🚫</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Privacy</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Private discovery</td>
<td>🚫</td>
</tr>
<tr>
<td>Anonymous communication</td>
<td>🚫</td>
</tr>
<tr>
<td>Transparency</td>
<td>X</td>
</tr>
</tbody>
</table>
Thank You!
Users, Devices, Applications

○ The actors in our model are apps/processes

○ Each app on each device is a separate principal
  ○ It may be acting-on-behalf-of of a user depending on its blessings

○ Examples:
  ○ google/alice (AccountManager app on Alice’s phone)
  ○ google/alice/hometv (DeviceManager app on Alice’s TV)
  ○ google/alice/hometv/youtube (Youtube app on Alice’s TV)