WAVE: A decentralised authorization system for IoT via blockchain smart contracts

Michael P Andersen, John Kolb, Kaifei Chen, Gabe Fierro, David E. Culler, Raluca Ada Popa
The problem

Authorization mechanisms tend to be centralized
The problem: put a slightly different way

If a given user wants to trust a given device, they are generally forced to trust some other party.
Examples:

AWS IoT

G Login with Google

Login with Twitter

Login with Digits

Login with Github

Login with Tumblr

Login with Facebook
This is problematic

In a global Internet of Things, who can be trusted to authorize the world?
This is problematic

In a global Internet of Things, who can be trusted to authorize the world?

Even if the company policy is “don’t be evil”, employees are fallible (or vulnerable to subpoena)

Symantec in yet-another dodgy digital certificate revocation

Security? We've heard of it, says security software company
Can we build a useful system offering

Democratized authentication and authorization

- Anyone can grant permissions
- Can do so without communicating with grantee or any authority
- Anyone can verify any permissions non-interactively

Decentralized, consistent, persistent and attack-resistant permission state

Fully authorized syndication with no trust of routers or brokers

Transparent (publicly auditable) permissions, but if desired also private
Breaking this up

**Authorization layer**
Entities, DoTs, Namespaces

**Syndication layer**
Resources, Publish / Subscribe

**Overlay layer**
Routers (brokers), agents

**Physical layer**
IoT devices, blockchain, servers
Authorization layer: Entity

$<E_{sk}, E_{vk}>$

A keypair for signing and verifying

- Identified by $E_{vk}$ e.g. l0hKkvaVyRDqf_lwt93WJC_a9Zu2F3l61Au6fZtIsCU=
- Optionally identified by a globally unique, immutable alias e.g. mike19
- Represents the holder of the signing key:
  - IoT device
  - Participant
  - Services
Authorization layer: Namespaces & Resources

A namespace is controlled by an entity $E_{ns}$ and is a collection of resources:

```
namespace/resource_path
```

- All resource URIs begin with the $E_{ns}$ of the namespace entity (or its alias)
- $E_{ns}$ has full permissions on all resources within the namespace

```
alicehome/hvac/thermostat/setpoint
alicehome/security/door/islocked
caiso/pricing/zone25/electricity
```
Authorization layer: Delegation of Trust

\(<E_{\text{from}}, E_{\text{to}}, URI_{\text{resource}}, \text{Permissions}, \text{Sig}_{E_{\text{from}}}>\)

For other entities to obtain permissions on a resource, they must receive them via a delegation of trust (DoT).

A DoT is *useful* if the granter $E_{\text{from}}$ itself has the permissions.

This object is public and discoverable (more on that later)

Can publish to door lock resource
Authorization layer: DoT graph

For $E_{src}$ to prove it has $P$ on a URI, it is sufficient to show there exists a chain of DoTs, end to end, from $E_{NS}$ to $E_{SRC}$ and the intersection of the permissions granted by DoTs on this chain is greater than or equal to $P$.
Syndication tier

- An entity can **subscribe** to a resource
- It will receive all messages **published** to that resource
- Same pattern as other pub/sub used in the IoT space
Syndication tier: messages

- A message consists of:
  - A Resource URI
  - The payload to publish to the URI
  - A chain of DoTs proving the message sender is authorized
  - A signature by the sender verifying the message has not been tampered or forged (authentication)
Overlay tier: Routers and Agents

Agent: acts on behalf of an Entity
- Builds proofs for entity
- Encodes and signs messages
- Validates incoming messages
- Participates in block chain

Router: routes messages for a namespace
- Verifies proofs
- Verifies messages
- Forwards published messages to subscribers
- Participates in block chain
So far, what do we have?

- A way of implementing authorization and syndication at a high level
- Many properties are closely linked with how the objects making up the DOT graph are disseminated and stored

- Consistency: everyone sees the same global view
- Persistence (revocations are not forgotten)
- Attack resistance: spamming etc
Many ways this could be done

- Centralized
- DHT
- Synchronizing Key Servers (e.g. GPG)

These struggle with:

- Guaranteed persistence and dissemination of revocation
- Operating without trusting the “core”
How to solve this without an authority?
How to solve this without an authority?
### Four contracts

<table>
<thead>
<tr>
<th>WAVE object inspection:</th>
<th>Registry:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validate signatures</td>
<td>Store $E_{vk}$ + metadata</td>
</tr>
<tr>
<td>Decode packed objects</td>
<td>Store DoT</td>
</tr>
<tr>
<td>Precompiled for speed</td>
<td>Index objects for access</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Affinity:</th>
<th>Aliases:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store $E_{NS} \rightarrow E_{DR}$ map</td>
<td>Store $E_{vk} \leftrightarrow E_{alias}$ mappings</td>
</tr>
<tr>
<td>Store $E_{DR} \rightarrow$ IP address</td>
<td>Like DNS but immutable</td>
</tr>
<tr>
<td>Like DNS but no central authority</td>
<td></td>
</tr>
</tbody>
</table>
This solves ALMOST all the problems

Except privacy: permissions grant access to resources

Resource URIs contain potentially sensitive information

<alice, bob, /gndfloor/lock/samsunglock/openstate, sub>

<ed, phil, /17mlk/r&d/project_infinity/grav_feeds, pub>
Protected DoTs

DoTs contain URI patterns which identify resources and therefore devices, services, organizational structure etc. How can we hide this?

Much harder problem that you would initially suspect
Pfft that can’t be that hard, just use:

- Encrypted session: **Nope**, granter and grantee don’t communicate
- Asymmetric crypto, just encrypt under the recipient public key **Nope**, only lets recipient see your DoT, not the ones leading up to it
- Ok but also include copies of the OTHER dots they need **Nope**, DoTs granted out of order, those may not even exist when you grant
- They can contact a service that gives them the DoTs **Yes**, but not without compromising on everything we stated was important
- It’s not possible
- I agreed with you until recently
Identity based encryption primer

Master Public Key

Master Private Key
Identity based encryption primer

Master Public Key

Random String "identity"

Public key "identity"

Private key "identity"

Master Private Key
Identity based encryption primer

Random String “identity”

Master Public Key

Master Private Key

Public key “identity”

Public key “identity3”

Private key “identity”

Private key “identity2”

Private key “identity3”

NO WAY TO GET FROM ONE PRIVATE KEY TO ANOTHER
Protected DoTs
Encrypt this DoT in the recipient IBE system using the namespace and permissions as the “identity”
Encrypt this DoT in the recipient IBE system using the namespace and permissions as the “identity” but also include the PRIVATE KEY in the source IBE system generated with the SAME identity.
Wait, the *private* key?

Yes. The private key. Just watch
Protected DoTs

When it comes time to build a proof:

D can trivially decode the DoT from C to D because its under D’s key.
Protected DoTs

When it comes time to build a proof:

**D** can trivially decode the DoT from **C** to **D** because its under **D**’s key
Then **D** learns **C**’s private key, so can decode BC
When it comes time to build a proof:

D can trivially decode the DoT from C to D because its under D’s key
Then D learns C’s private key, so can decode BC
Etc etc
Well yeah, but you could have done that ANYWAY

- Yes, you could have just sent your private key and used normal encryption, but then the recipient has a powerful private key
- This IBE private key:
  - Is only used for encrypting DOTs
  - Only in a single namespace
  - Only granting a specific set of permissions
  - That you were granted access to
- So only a minimal set of information is revealed
definitely-not-useful paths are not revealed
Other parts not presented here

CPU, memory and bandwidth usage on a variety of platforms under different scenarios (idle, normal, attack) and different internet characteristics (latency, speed)

City-scale (millions of people) emulation drawn from public San Francisco data

Very robust DDOS protection due to rapid, accurate traffic identification coupled with sybil-proof identities

>400 days of deployment across a handful of namespaces, hundreds of devices and tens of thousands of resources
Questions?

Michael Andersen <m.andersen@berkeley.edu>