WAVE: A Decentralized Authorization Framework with Transitive Delegation

Michael P Andersen, Sam Kumar, Hyung-Sin Kim, John Kolb, Kaifei Chen, Moustafa AbdelBaky, Gabe Fierro, David E. Culler, Raluca Ada Popa

This material is based on work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1752814. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.
Roadmap

1. The Problem
2. WAVE’s Approach
3. WAVE’s Storage Layer
4. WAVE’s Privacy Layer
5. Implementation and Evaluation
Authorization for IoT

“Set temperature to 80 F”
Authorization

1. How does the tenant receive permission to adjust the temperature?

2. How does the thermostat know that the request was sent by someone who has permission?
Authorization for IoT: Status Quo

- Shares identity, not just permission
- Not revocable
- Ad-hoc

Owner gives username:password to airbnb

Airbnb performs action when tenant requests

Tenant
The Problems

Existing authentication systems are centralized (often monolithic)

Transitive delegation is rare, leading to over-sharing

Attacks on centralized systems are common, and affect all the users

HACKER WARNING: Over a BILLION login and password details listed on the Dark Web

By ODIY DASSANAYAKE
PUBLISHED: 17/15, Tue, Dec 12, 2017 | UPDATED: 17/23, Tue, Dec 12, 2017

SHARE f TWEET G+ " ¥"
Roadmap

1. The Problem
2. WAVE’s Approach
3. WAVE’s Storage Layer
4. WAVE’s Privacy layer
5. Implementation and Evaluation
WAVE’s Approach

Make the flow of trust fine-grained.
WAVE Captures Trust Relations

- Allows delegation of a subset of permissions
- Prevents needing to share identity
- Cryptographically enforced

Attestation 1
Policy: “Airbnb can set Owner’s thermostat setpoint, and can delegate” *(Signed by Owner)*

Attestation 2
Policy: “Tenant can set Owner’s thermostat setpoint, for only the duration of their stay” *(Signed by Airbnb)*

PROOF

- Attestation 2
- Attestation 1
Global Permissions Graph

1. Entity receives permission via a chain of attestations

2. Entity presents a path through the graph as proof it is authorized

Authorization

1. How does one receive permission?

2. How does the device know that the request was sent by someone who has permission?
Two Technical Challenges in WAVE

1. How are attestations stored, disseminated, and discovered, without relying on a single trusted party?
   ◦ Storage layer

2. How to protect the privacy of attestations?
   ◦ Privacy layer
Roadmap

1. The Problem
2. WAVE’s Approach
3. WAVE’s Storage Layer
4. WAVE’s Privacy layer
5. Implementation and Evaluation
Storage Layer Goals

Storage provider is untrusted, so it must be verifiable that it is not:

- Hiding objects (such as revocation entries)
- Forging existence of non-existent objects

Blockchain is a natural solution but unfortunately doesn’t scale
Use log of operations backed by Merkle Tree [Certificate Transparency, Laurie et al. 2013]

How to make sure the server can’t hide objects?

◦ Server must be able to prove that an object doesn’t exist
◦ Not supported by Merkle Tree Log!
Use another Merkle tree to construct map of objects [Verifiable Log-Derived Map, Eijdenberg et al. 2015]

However, server could serve requests using an older version of the map
◦ How to fix this?
Use another log to store progression of map root hashes

WAVE’s Storage Layer (Final)

Merkle Tree Log of operations
Contains all the authorization objects
Can prove:
- Append-only
- Value exists in log

Merkle Tree Map of objects
Contains objects indexed by their hash
Can prove:
- Value does not exist
- Value exists

Merkle Tree Log of map roots
Contains all the root hashes of the map
Can prove:
- Append-only
- Value exists in log

Auditors make sure that each request is served using the latest map version
Roadmap

1. The Problem
2. WAVE’s Approach
3. WAVE’s Storage Layer
4. WAVE’s Privacy layer
5. Implementation and Evaluation
Private Attestations

With this storage model, global permissions graph is *publicly accessible*

- Leaks, e.g., who is renting which house on Airbnb

Storage is untrusted; can’t rely on it for access control

Instead we rely on cryptography

- Attestations are encrypted
- They can only be decrypted by an entity who can use them in a proof
Encrypt Attestations

Proving entity
Encrypt Attestations

Hidden attestations
Decryptable attestations

Proving entity
Encrypt Attestations

Hidden attestations

Decryptable attestations

Proving entity
Our Technique: Reverse-Discoverable Encryption (simplified)

Attestations are encrypted using recipient’s public key

Attestations include secret key of granter

◦ Allows decryption of upstream attestations

Owner signs statement saying “Airbnb has permission to set my thermostat setpoint, and can delegate”

Airbnb signs statement saying “Tenant has permission to adjust Owner’s thermostat setpoint”

We actually use policy-aware encryption to restrict access further.
Our Technique: Reverse-Discoverable Encryption (simplified)

Each entity has a keypair for encrypting attestations.

Attestations are encrypted using recipient’s public key.

Attestations include secret key of granter.
  ◦ Allows decryption of upstream attestations.

We actually use policy-aware encryption to restrict access further.
Reverse-Discoverable Encryption
Roadmap

1. The Problem
2. WAVE’s Approach
3. WAVE’s Storage Layer
4. WAVE’s Privacy layer
5. Implementation and Evaluation
# First Release of WAVE Version 3

<table>
<thead>
<tr>
<th>Feature</th>
<th>WAVE 2</th>
<th>WAVE 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delegation</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Decentralized</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Scalable</td>
<td>No (blockchain)</td>
<td>Yes</td>
</tr>
<tr>
<td>Encrypted Attestations</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Fully General</td>
<td>No (IoT pubsub)</td>
<td>Yes</td>
</tr>
<tr>
<td>Full Implementation</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

WAVE Version 2: [github.com/immesys/bw2](https://github.com/immesys/bw2)

WAVE Version 3: [github.com/immesys/wave](https://github.com/immesys/wave)
## Operation Times [ms]

<table>
<thead>
<tr>
<th>Operation</th>
<th>AMD64</th>
<th>ARMv8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create attestation</td>
<td>39.4</td>
<td>754</td>
</tr>
<tr>
<td>Create entity</td>
<td>25.7</td>
<td>344</td>
</tr>
<tr>
<td>Decrypt attestation as verifier</td>
<td>0.62</td>
<td>10.3</td>
</tr>
<tr>
<td>Decrypt attestation as subject</td>
<td>25.6</td>
<td>413</td>
</tr>
<tr>
<td>Decrypt delegated attestation</td>
<td>31.9</td>
<td>642</td>
</tr>
</tbody>
</table>

**Operations**
- Granting permissions
- Creating accounts
- Verifying proofs
- Discovering new attestations
1. Authenticate
   ◦ LDAP Bind

2. Check Auth Policy
   ◦ SQL Lookup

Total: 7.5 ms
Use Case Comparison (Critical Path)

1. Validate proof (yields policy)

Total: < 7 ms for common patterns
Conclusion

WAVE is an authentication/verification engine that makes trust relationships fine-grained

It can run at global scale without a central trusted party

It is a REAL artifact we have operated for 2 years, securing over 800 IoT devices in California!