WAVE: A Decentralized Authorization Framework with Transitive Delegation

Michael P Andersen, Sam Kumar, Moustafa AbdelBaky, Gabe Fierro, John Kolb, Hyung-Sin Kim, David E. Culler, Raluca Ada Popa
University of California, Berkeley
Representative authorization example

Building Owner  BLDG2/Floor3  Tenant Company CEO  BLDG2/Floor3/HVAC  BLDG2/Floor3/LIGHT  BLDG2/Floor3/DOORS  Employees
Traditional approach

- Building Owner
- Tenant Company CEO
- Employees

E.g. OAuth, LDAP

- BLDG2/Floor3
- BLDG2/Floor3/HVAC
- BLDG2/Floor3/LIGHT
- BLDG2/Floor3/DOORS
Traditional approach

Problems:
Central point of attack

Building Owner  BLDG2/Floor3  Tenant Company CEO  BLDG2/Floor3/HVAC  BLDG2/Floor3/LIGHT  BLDG2/Floor3/DOORS  Employees
Traditional approach

Building Owner

Tenant Company CEO

Employees

Problems:
Central point of attack
Can’t even trust operator
Traditional approach

Building Owner  Tenant Company  CEO  Employees

Problems:
Central point of attack
Can’t even trust operator

BLDG2/Floor3/ HVAC  BLDG2/Floor3/LIGHT  BLDG2/Floor3/DOORS
Traditional approach

Problems:
Central point of attack
Can’t even trust operator

Building Owner  Tenant Company  CEO  Employees

BLDG2/Floor3  BLDG2/Floor3/HVAC  BLDG2/Floor3/LIGHT  BLDG2/Floor3/DOORS
Traditional approach

Problems:
Central point of attack
Can’t even trust operator

Building Owner

Tenant Company CEO

Employees

BLDG2/Floor3
BLDG2/Floor3/HVAC
BLDG2/Floor3/LIGHT
BLDG2/Floor3/DOORS
Traditional approach

Problems:
Central point of attack
Can’t even trust operator

Building Owner

 Tenant Company
  CEO

 Employees

BLDG2/Floor3

BLDG2/Floor3/HVAC

BLDG2/Floor3/LIGHT

BLDG2/Floor3/DOORS
Traditional approach

Problems:
- Central point of attack
- Can’t even trust operator
- Sometimes delegation unsupported

Building Owner

Tenant Company CEO

Employees

BLDG2/Floor3

BLDG2/Floor3/HVAC

BLDG2/Floor3/LIGHT

BLDG2/Floor3/DOORS
Traditional approach

Problems:
- Central point of attack
- Can’t even trust operator
- Sometimes delegation unsupported
- When supported, not transitive

Building Owner → Tenant Company CEO → Employees

BLDG2/Floor3
BLDG2/Floor3/HVAC
BLDG2/Floor3/LIGHT
BLDG2/Floor3/DOORS
Lack of transitive delegation

Building Owner

BLDG2/Floor3

Tenant Company
CEO

BLDG2/Floor3/HVAC
BLDG2/Floor3/LIGHT
BLDG2/Floor3/DOORS

Employees
Lack of transitive delegation
Lack of transitive delegation

Building Owner

Tenant Company
CEO

Employees

BLDG2/Floor3
BLDG2/Floor3/HVAC
BLDG2/Floor3/LIGHT
BLDG2/Floor3/DOORS
What we want:

- Building Owner
- Tenant Company
- CEO
- Employees

BLDG2/Floor3
BLDG2/Floor3/HVAC
BLDG2/Floor3/LIGHT
BLDG2/Floor3/DOORS
Existing work lacks some important features

<table>
<thead>
<tr>
<th>System / Work</th>
<th>Avoid central authority</th>
<th>Transitive Delegation</th>
<th>Permission Discovery</th>
<th>No ordering constraints</th>
<th>Offline participants</th>
<th>Protected permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP, AD</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>OAuth2</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Macaroons</td>
<td>✗</td>
<td>✓</td>
<td>–</td>
<td>✗</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>SDSI/SPKI</td>
<td>✗</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Distributed TM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
Existing work lacks some important features

<table>
<thead>
<tr>
<th>System / Work</th>
<th>Avoid central authority</th>
<th>Transitive Delegation</th>
<th>Permission Discovery</th>
<th>No ordering constraints</th>
<th>Offline participants</th>
<th>Protected permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP, AD</td>
<td>✗</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✗</td>
</tr>
<tr>
<td>OAuth2</td>
<td>✗</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✗</td>
</tr>
<tr>
<td>Macaroons</td>
<td>✗</td>
<td>✅</td>
<td>-</td>
<td>✗</td>
<td>✅</td>
<td>-</td>
</tr>
<tr>
<td>SDSI/SPKI</td>
<td>✗</td>
<td>✅</td>
<td>-</td>
<td>✅</td>
<td>✅</td>
<td>-</td>
</tr>
<tr>
<td>Distributed TM</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
Existing work lacks some important features

<table>
<thead>
<tr>
<th>System / Work</th>
<th>Avoid central authority</th>
<th>Transitive Delegation</th>
<th>Permission Discovery</th>
<th>No ordering constraints</th>
<th>Offline participants</th>
<th>Protected permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP, AD</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>OAuth2</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Macaroons</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SDSI/SPKI</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Distributed TM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
### Existing work lacks some important features

<table>
<thead>
<tr>
<th>System / Work</th>
<th>Avoid central authority</th>
<th>Transitive Delegation</th>
<th>Permission Discovery</th>
<th>No ordering constraints</th>
<th>Offline participants</th>
<th>Protected permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP, AD</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>OAuth2</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Macaroons</td>
<td>✗</td>
<td>✓</td>
<td>−</td>
<td>✗</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>SDSI/SPKI</td>
<td>✗</td>
<td>✓</td>
<td>−</td>
<td>✓</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Distributed TM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>
WAVE is designed to provide these

<table>
<thead>
<tr>
<th>System / Work</th>
<th>Avoid central authority</th>
<th>Transitive Delegation</th>
<th>Permission Discovery</th>
<th>No ordering constraints</th>
<th>Offline participants</th>
<th>Protected permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAP, AD</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>OAuth2</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Macaroons</td>
<td>✗</td>
<td>✓</td>
<td>-</td>
<td>✗</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SDSI/SPKI</td>
<td>✗</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Distributed TM</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>WAVE</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
What is WAVE

<table>
<thead>
<tr>
<th>System / Work</th>
<th>No central authority</th>
<th>Transitive Delegation</th>
<th>Permission Discovery</th>
<th>No ordering constraints</th>
<th>Offline participants</th>
<th>Protected permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAVE</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

WAVE is a cryptographically enforced decentralized authorization system

- It can be used in place of most mainstream authorization systems
- Anyone can delegate permissions or revoke permissions they have delegated
- Anyone can discover their permissions and form a proof of authorization
- Anyone (even devices) can verify proofs of authorization
WAVE achieves this with three techniques:

- Graph based authorization
- Reverse-discoverable encryption
- Scalable untrusted storage
Graph Based Authorization

- Popularized by SDSI/SPKI \cite{Rivest, Lampson, 1996}
- Represents permissions as a graph, rather than an ACL table
- Naturally represents transitive delegation

Building Owner → Tenant Company CEO → Employees

BLDG2/Floor3  →  BLDG2/Floor3/HVAC  →  BLDG2/Floor3/LIGHT  →  BLDG2/Floor3/DOORS
Graph Based Authorization

Building Owner

Tenant Company
CEO

Employees

BLDG2/Floor3

BLDG2/Floor3/HVAC

BLDG2/Floor3/LIGHT

BLDG2/Floor3/DOORS
Graph Based Authorization

Participants: Entities
Collections of cryptographic keys

Building Owner

Tenant Company CEO

BLDG2/Floor3
BLDG2/Floor3/HVAC
BLDG2/Floor3/LIGHT
BLDG2/Floor3/DOORS

Employees
Graph Based Authorization

Grants of permissions: Attestations
Signed certificates created by Entities
Graph Based Authorization

Attestations grant permissions on a resource

Permission: Read, Write

Resource: BldgOwner/BLDG2

Expires: 2019/04/05
Graph Based Authorization

Attestations grant permissions on a **resource**. Resources are in a **namespace** which identifies the authority entity.

- **Permission**: Read, Write
- **Resource**: BldgOwner/BLDG2
- **Expires**: 2019/04/05
Graph Based Authorization

Proof of permissions: A path through the graph from Namespace Authority to the prover

Proof grants the intersection of the permissions of each attestation
Verifiable by anyone*, attached to messages

* In WAVE, not SDSI/SPKI
This forms a single global graph
This forms a single global graph

- Multiple namespace authorities in the graph
This forms a single global graph

- Multiple namespace authorities in the graph
- Different entities will only see portions of the graph
This forms a single global graph

- Multiple namespace authorities in the graph
- Different entities will only see portions of the graph
- The graph is publicly accessible
We need to hide portions of the graph
Reverse Discoverable Encryption

Building Owner → Tenant Company CEO → Employees
Reverse Discoverable Encryption

Building Owner \(\xrightarrow{\text{NS/Floor3}}\) Tenant Company CEO \(\xrightarrow{\text{NS/Floor4}}\) F3 Manager
Reverse Discoverable Encryption

Building Owner

Tenant Company CEO

F3 Manager

Janitorial Services

NS/Floor3

NS/Floor4

NS/Floor3
Reverse Discoverable Encryption

Building Owner → Tenant Company CEO → F3 Manager → HVAC Controller

NS/Floor3

Janitorial Services

NS/Floor4
Reverse Discoverable Encryption

- Building Owner
- Tenant Company CEO
- F3 Manager
- HVAC Controller
- Janitorial Services

Discovering permissions
Reverse Discoverable Encryption

Three kinds of attestations:
- On path, intersecting
Reverse Discoverable Encryption

- **Building Owner**
- **Tenant Company CEO**
- **F3 Manager**
- **Janitorial Services**
- **HVAC Controller**

Three kinds of attestations:
- On path, intersecting
- On path, not intersecting
Reverse Discoverable Encryption

Three kinds of attestations
- On path, intersecting
- On path, not intersecting
- Not on a path
Technique in a nutshell

Encrypt attestations

In each attestation, include a secret that allows you to decrypt upstream attestations that have intersecting permissions

(on path, intersecting)
Reverse Discoverable Encryption

Building Owner

Tenent Company CEO

F3 Manager

HVAC Controller

Janitorial Services
Reverse Discoverable Encryption

Building Owner  NS/Floor3  Tenant Company CEO  F3 Manager  HVAC Controller

Janitorial Services  NS/Floor3  NS/Floor3  NS/Floor3
Reverse Discoverable Encryption

Building Owner → NS/Floor3 → NS/Floor4 → Tenant Company CEO → NS/Floor3 → F3 Manager → NS/Floor3 → HVAC Controller

Janitorial Services → NS/Floor3 → Tenant Company CEO
Reverse Discoverable Encryption

Building Owner

Tenant Company CEO

F3 Manager

HVAC Controller

Janitorial Services
Reverse Discoverable Encryption

Building Owner

Tenant Company CEO

F3 Manager

HVAC Controller

Janitorial Services

NS/Floor3

NS/Floor3

NS/Floor3

NS/Floor3

NS/Floor3

NS/Floor3

NS/Floor3

NS/Floor3
The encryption & secret must capture the permissions

- We use Wildcard Identity Based Encryption (WIBE) [Abdalla, 2006]
- Every entity has a WIBE master key
  - No PKG, every entity has their own system
  - Used just for RDE, nothing else
- When you create attestation (grant permissions)
  - Form WIBE ID = F(permissions)
  - Generate private key for that ID using granting entity master key
  - Include in attestation
  - Encrypt attestation using WIBE params for recipient using same ID

This is simplified, please see paper for more details
Reverse Discoverable Encryption

Building Owner → NS/Floor3
NS/Floor4 → Tenant Company CEO
NS/Floor3 → F3 Manager
NS/Floor3 → HVAC Controller
NS/Floor3 → Janitorial Services
Reverse Discoverable Encryption

Encrypted using:  
ID: F(NS/Floor3)  
Params: Controller  
Contains secret key:  
ID: F(NS/Floor3)  
MSK: F3 Manager
Reverse Discoverable Encryption

Building Owner

Tenant Company
CEO

F3 Manager

Janitorial Services

Encrypted using:
ID: F(NS/Floor3)
Params: F3 Manager
Contains secret key:
ID: F(NS/Floor3)
MSK: CEO
Reverse Discoverable Encryption

Encrypted using:  
ID: F(NS/Floor3)  
Params: CEO  
Contains secret key:  
Not necessary
Reverse Discoverable Encryption

Encrypted using: 
ID: F(NS/Floor4) 
Params: CEO

Cannot be decrypted: wrong resource
Reverse Discoverable Encryption

Encrypted using: ID: F(NS/Floor3)
Params: Janitorial Services

Cannot be decrypted:
No key from Janitorial Services
Reverse Discoverable Encryption Summary

- Allows entities to decrypt attestations that they can use in a proof
- Does not require out of band communication
- Works when attestations are granted in any order

Full version (in paper) supports expiry of attestations
We need a place to store the encrypted attestations.

[Diagram:
- Graph based authorization
- Reverse-discoverable encryption
- Scalable untrusted storage]
A blockchain nearly works

- Our earlier work used a blockchain
  - Cryptographically proven integrity
  - No central authorities

- Unfortunately it didn’t scale well
  - Blockchains don’t really go past a few tens of transactions per second
  - Especially if transactions are large (attribution objects)
Unequivocable Log Derived Map

We designed the Unequivocable Log Derived Map to provide similar guarantees to a blockchain, when only storing objects.

Horizontally scalable public ledger with cryptographic integrity proofs similar to Certificate Transparency or Key Transparency, except:

1) It supports proof of non-existence, which allows revocation
2) It has efficient auditing
   - Clients only rarely communicate with auditors
   - Auditing load scales with number of additions to storage, not size of storage
High Level Overview

Storage servers

Clients
High Level Overview

Storage servers

Auditors

Clients
Constructed using three Merkle trees

Operation Log
0: Attestation
1: Attestation
2: Entity
3: Revocation
...

Merkle Tree Log
Can prove:
- Append-only
- Value exists in log

Object Map
Hash -> Attestation
Hash -> Entity
Hash -> Revocation
...

Merkle Tree Map
Can prove:
- Value does not exist
- Value exists

Map Root Log
0: H(Map₀)
1: H(Map₁)
2: H(Map₂)
3: H(Map₃)
...

Merkle Tree Log
Can prove:
- Append-only
- Value exists in log
Auditor replays operation log to construct replica

Operation Log

0: Attestation
1: Attestation
2: Entity
3: Revocation

Object Map

Hash -> Attestation
Hash -> Attestation
Hash -> Entity
Hash -> Revocation

Map Root Log

0: H(Map_0)
1: H(Map_1)
2: H(Map_2)
3: H(Map_3)

Ensures Object Map is properly derived from operation log

Clients send Root Hash of Map Root Log to auditors periodically (daily)
- Ensures every client is seeing the same data structure
Unequivocable Log Derived Map Summary

- Stores encrypted attestations, public entity objects, revocations
- Uses cryptographic proofs of integrity
- Forces operators to be honest, or be detected as dishonest
- Auditing requires infrequent communication between clients and auditors

Graph based authorization

Reverse-discoverable encryption

Scalable untrusted storage
WAVE is fully implemented

It’s written in Go, with some crypto in C++

[github.com/immesys/wave](https://github.com/immesys/wave)

We’ve used various versions of WAVE over the course of three years:

>200 devices, 20 buildings, multiple namespaces and organizations
It’s pretty fast

- Graph-changing operations - very fast by UI standards:
  - Creating an entity takes 9ms
  - Creating an attestation takes 43 ms
  - Decrypting an attestation takes 6ms

- Proof building / verification:
Conclusion

WAVE is a decentralized authorization system that offers transitive delegation by using graph based authorization

- Stores the graph in global storage with cryptographically enforced integrity
- Encrypts attestations, hiding the graph
- It can be used in place of most traditional authorization systems
Thank you & Questions

Michael Andersen
m.andersen@berkeley.edu