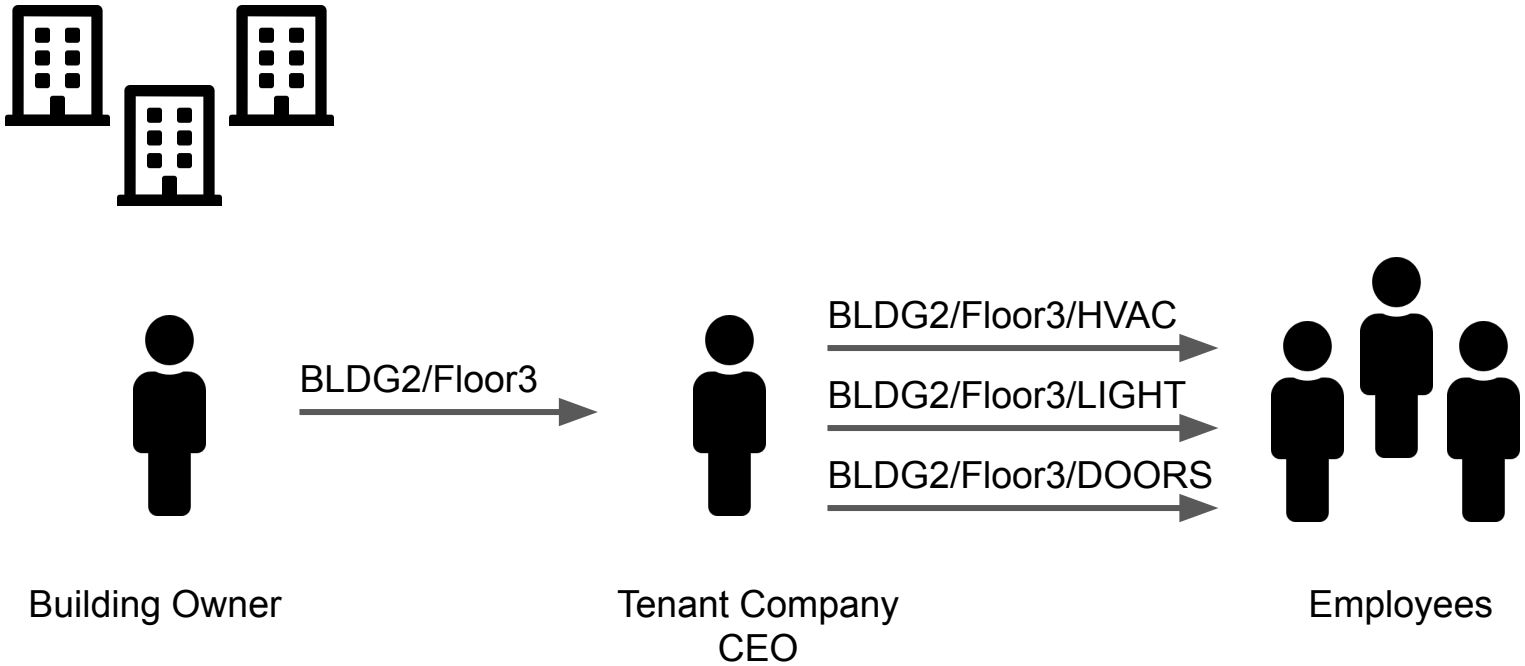
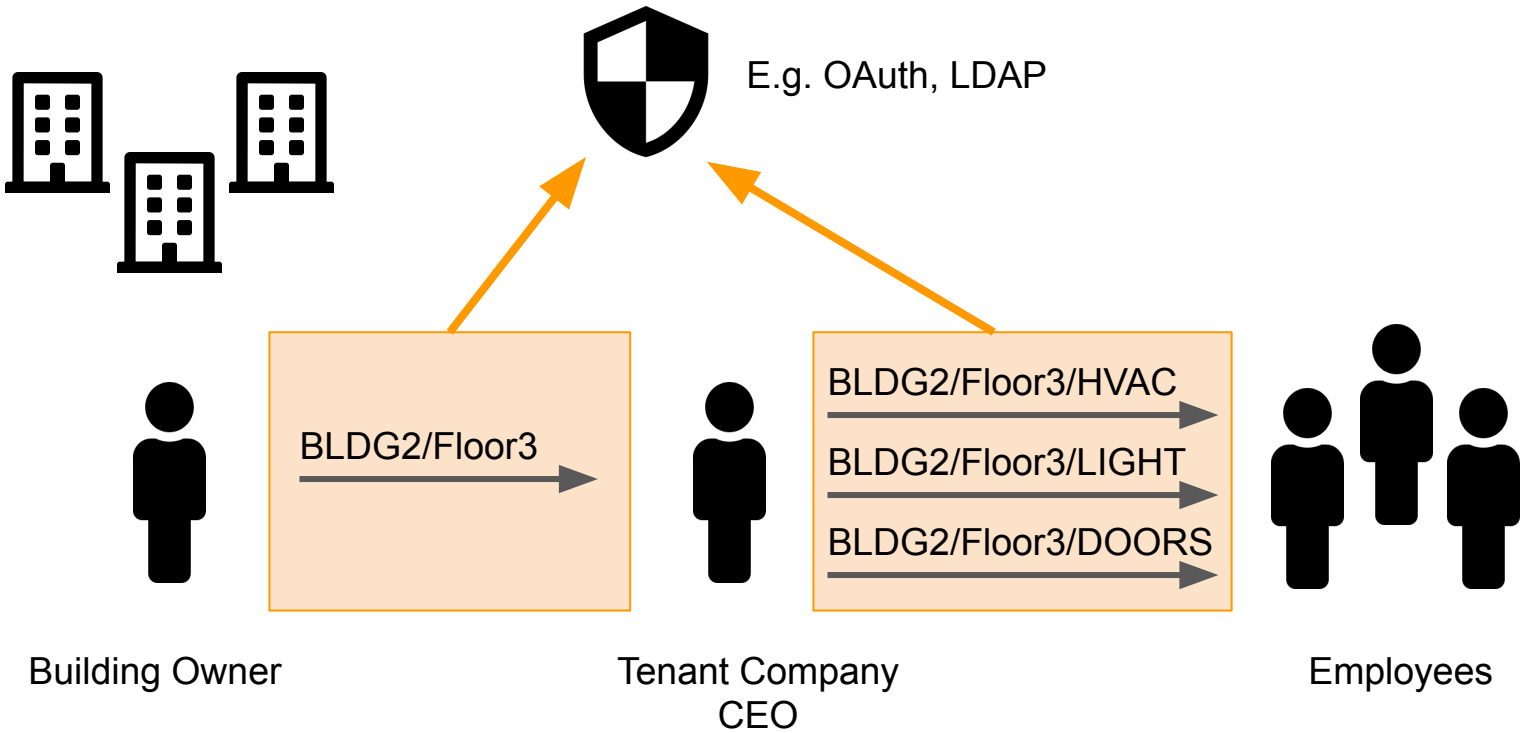
WAVE: A Decentralized Authorization Framework with Transitive Delegation

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

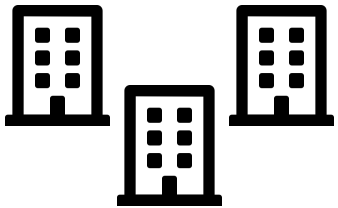
Representative authorization example

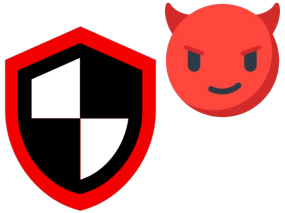


Traditional approach



Traditional approach





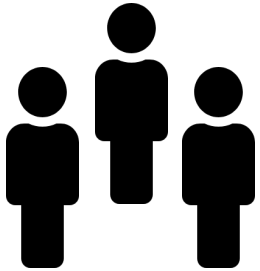
<u>Problems:</u>
Central point of attack

BLDG2/Floor3 →

BLDG2/Floor3/HVAC →

BLDG2/Floor3/LIGHT →

BLDG2/Floor3/DOORS →



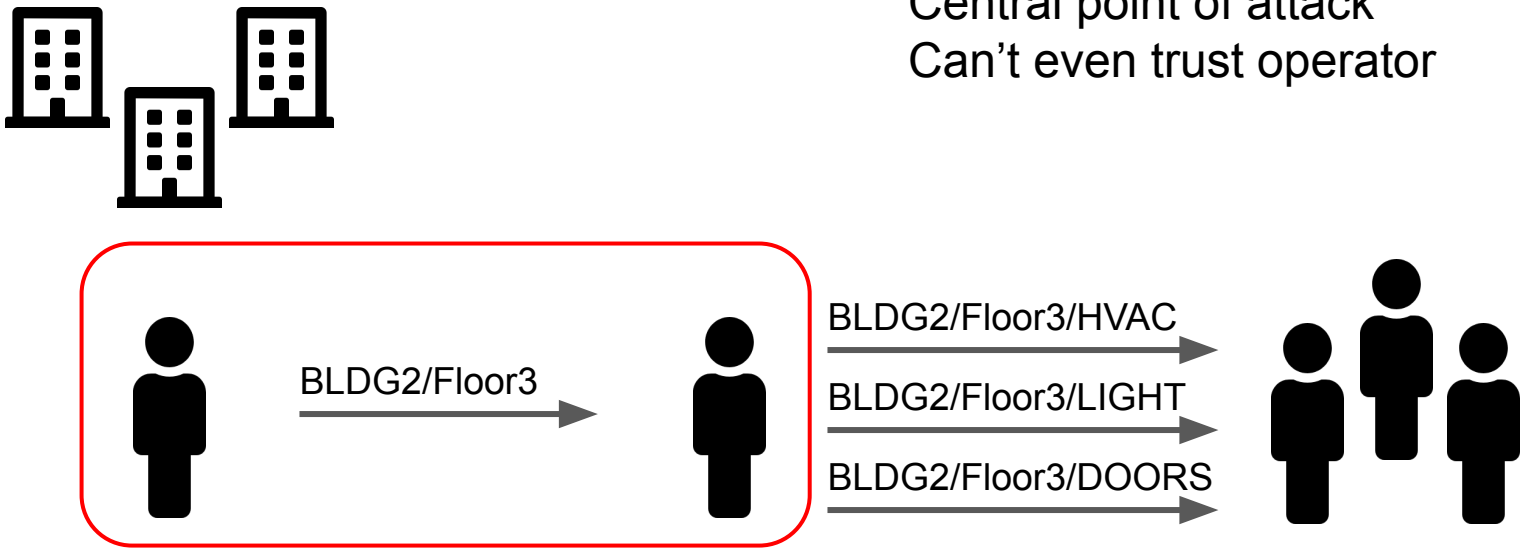
Building Owner

Tenant Company
CEO

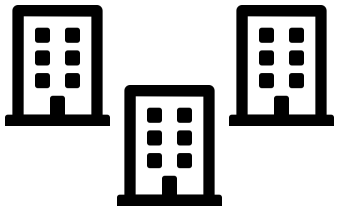
Employees

Traditional approach

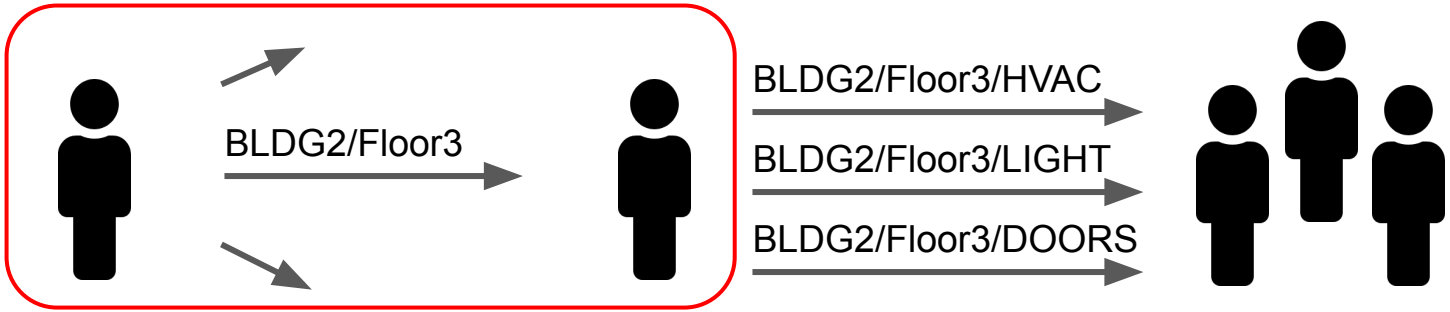
<u>Problems:</u>
Central point of attack
Can’t even trust operator



Traditional approach

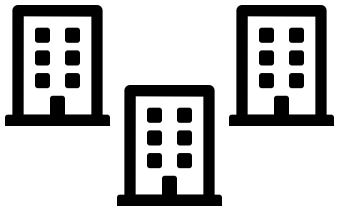


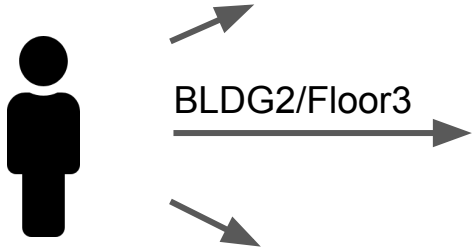
<u>Problems:</u>
Central point of attack
Can’t even trust operator

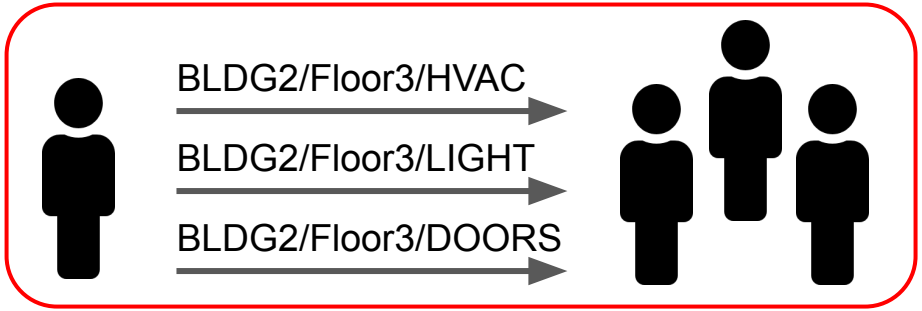


Traditional approach

<u>Problems:</u>
Central point of attack
Can’t even trust operator





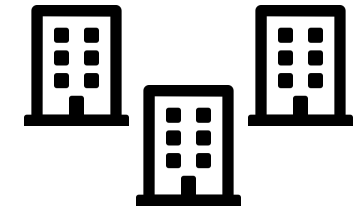


Building Owner

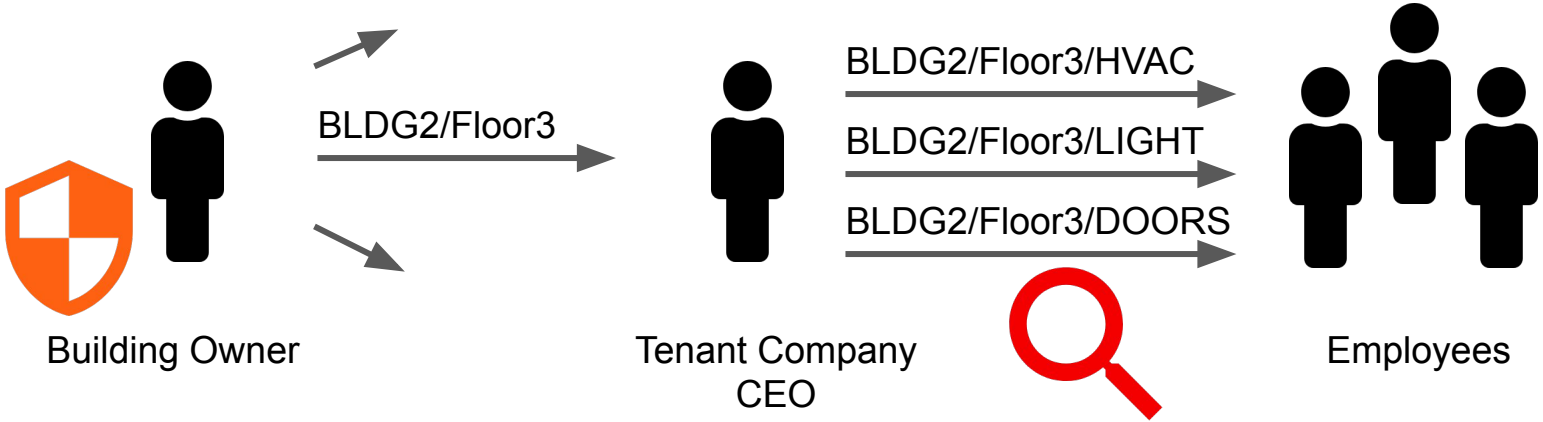
Tenant Company
CEO

Employees

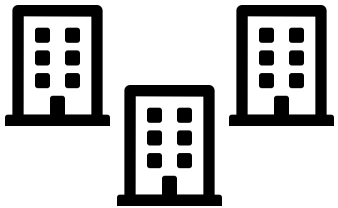
Traditional approach



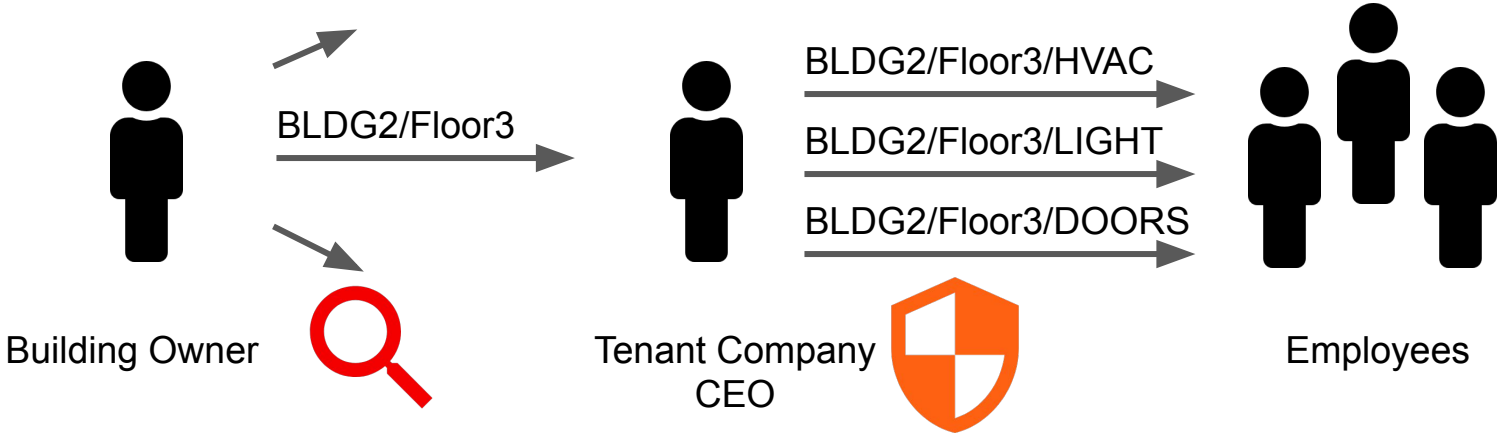
<u>Problems:</u>
Central point of attack
Can’t even trust operator



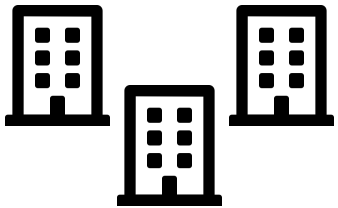
Traditional approach



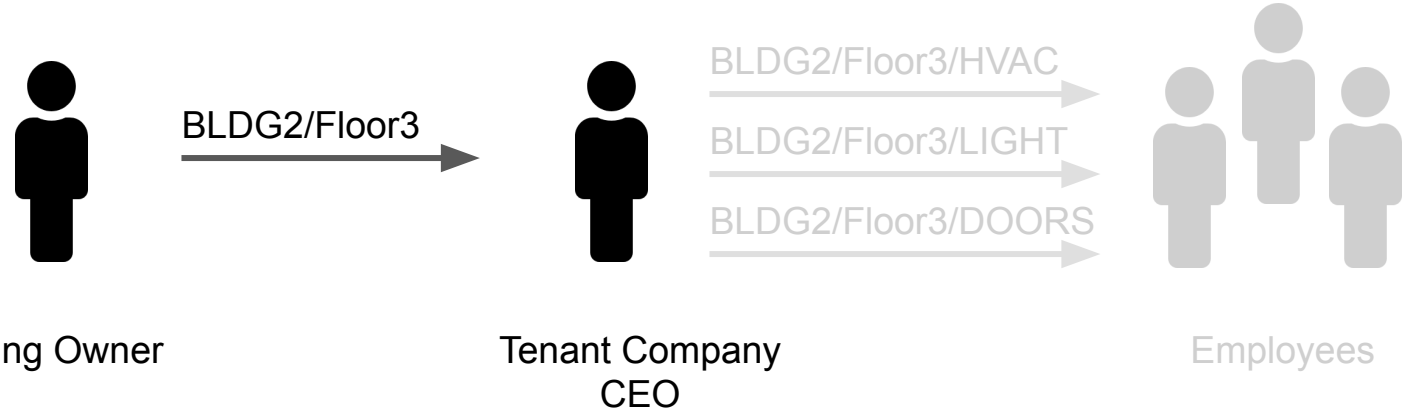
<u>Problems:</u>
Central point of attack
Can’t even trust operator



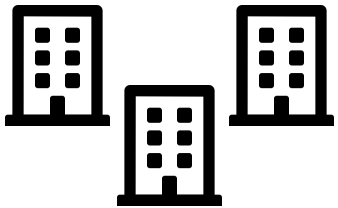
Traditional approach



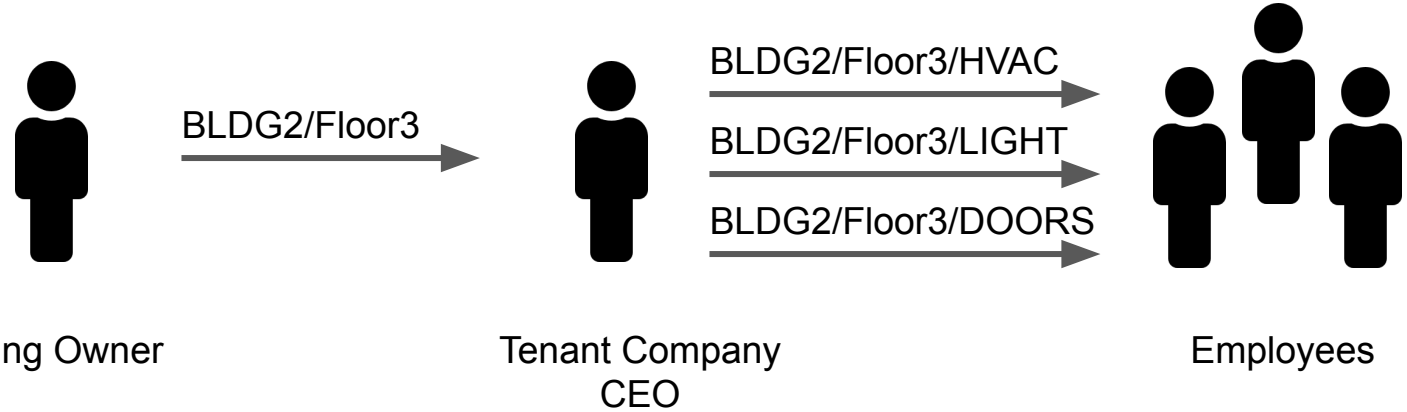
<u>Problems:</u>
Central point of attack
Can’t even trust operator
Sometimes delegation unsupported



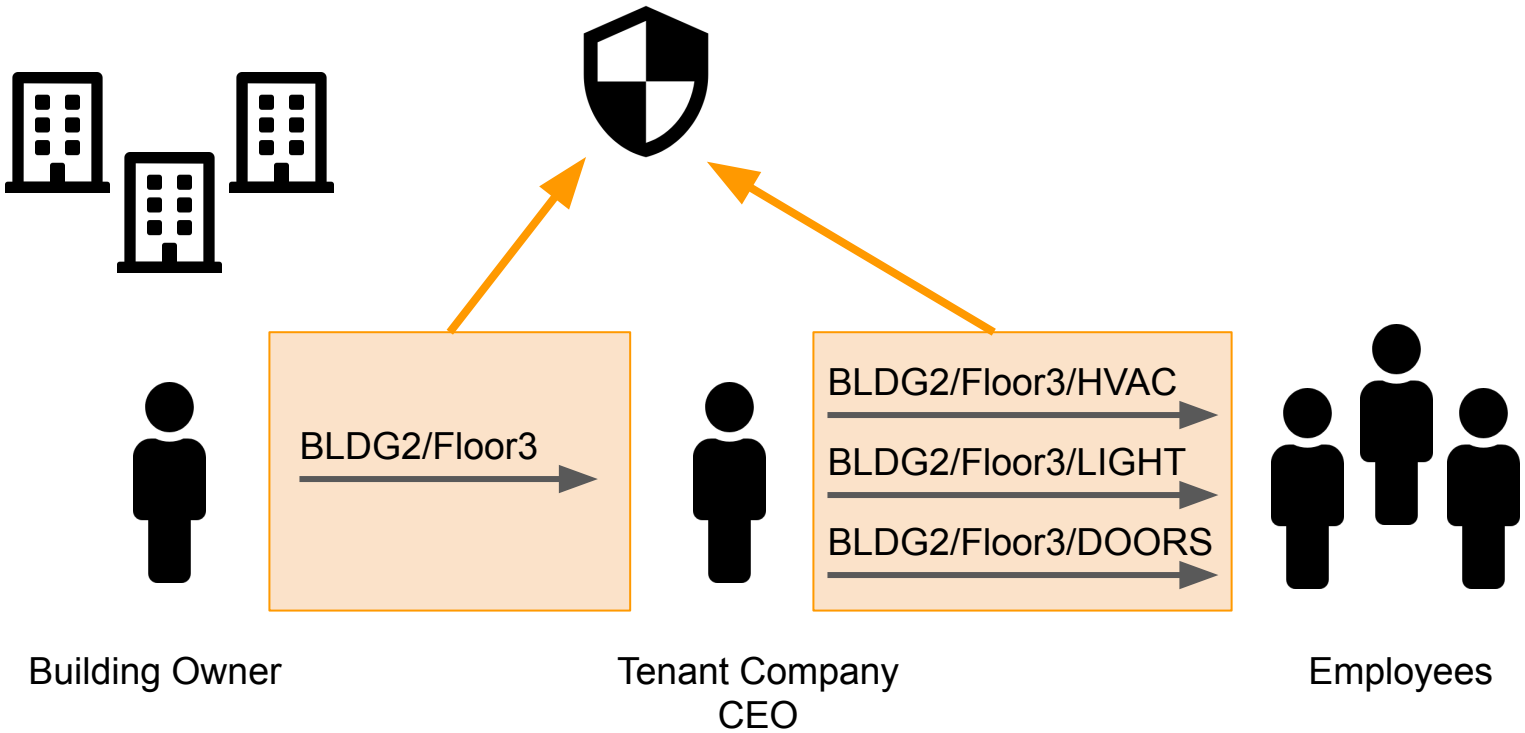
Traditional approach



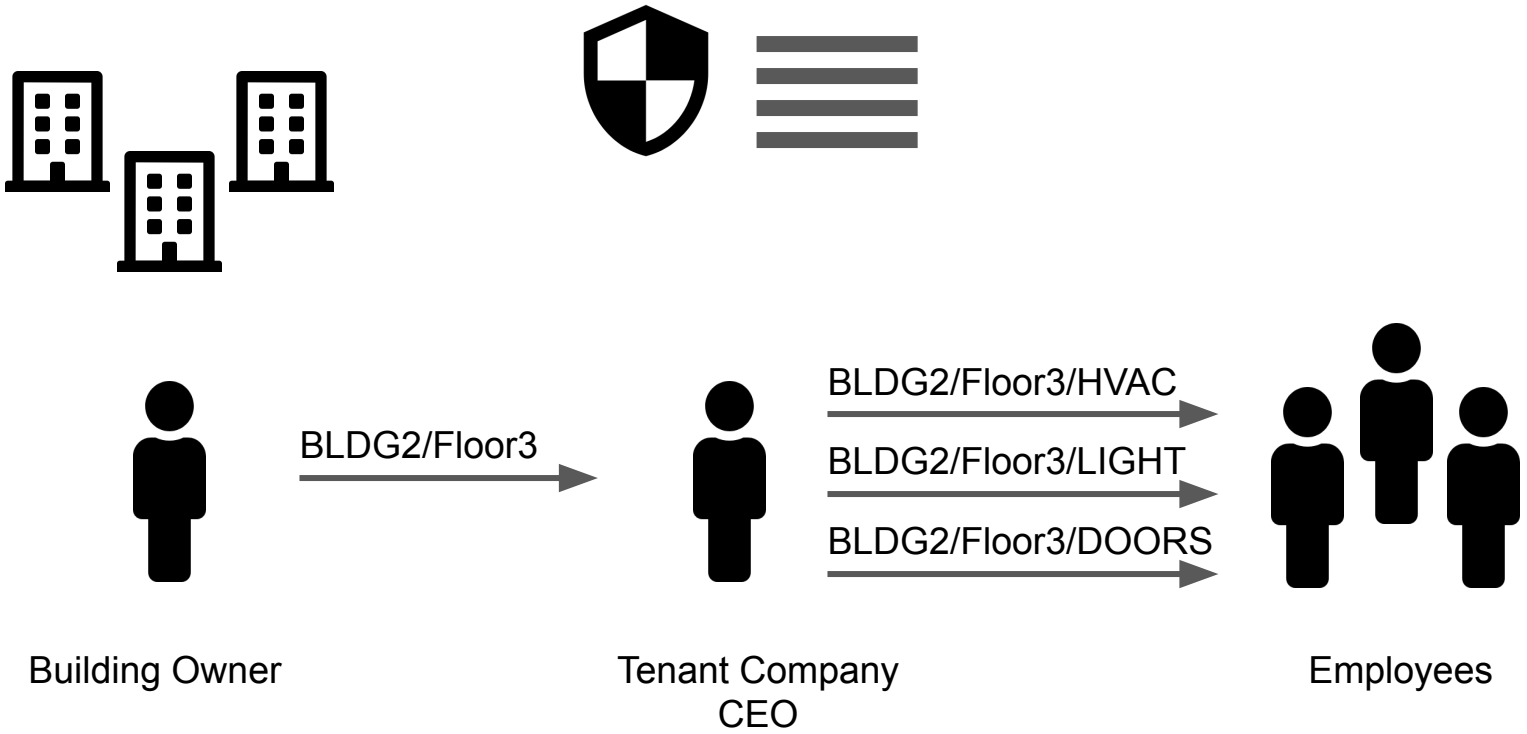
<u>Problems:</u>
Central point of attack
Can’t even trust operator
Sometimes delegation unsupported
When supported, not transitive



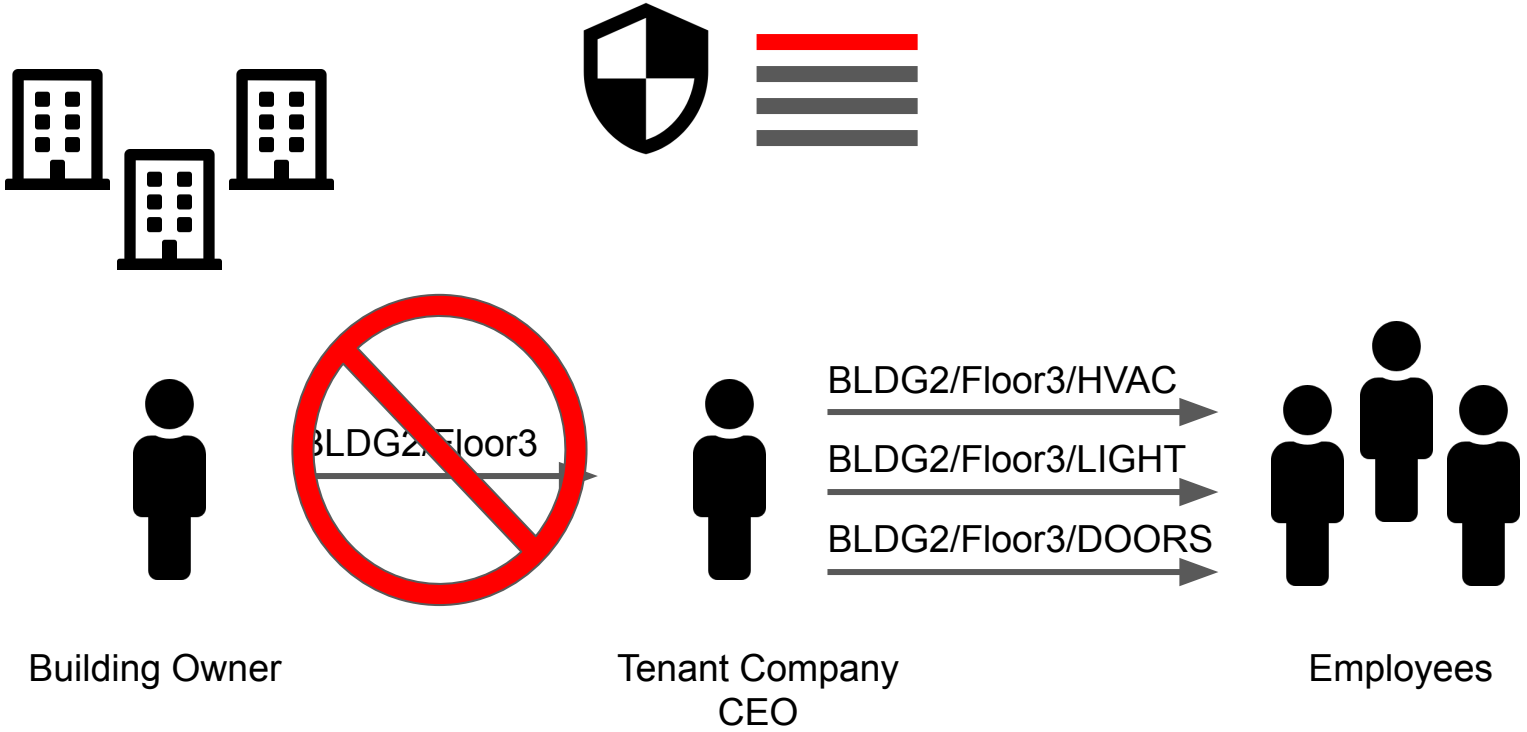
Lack of transitive delegation



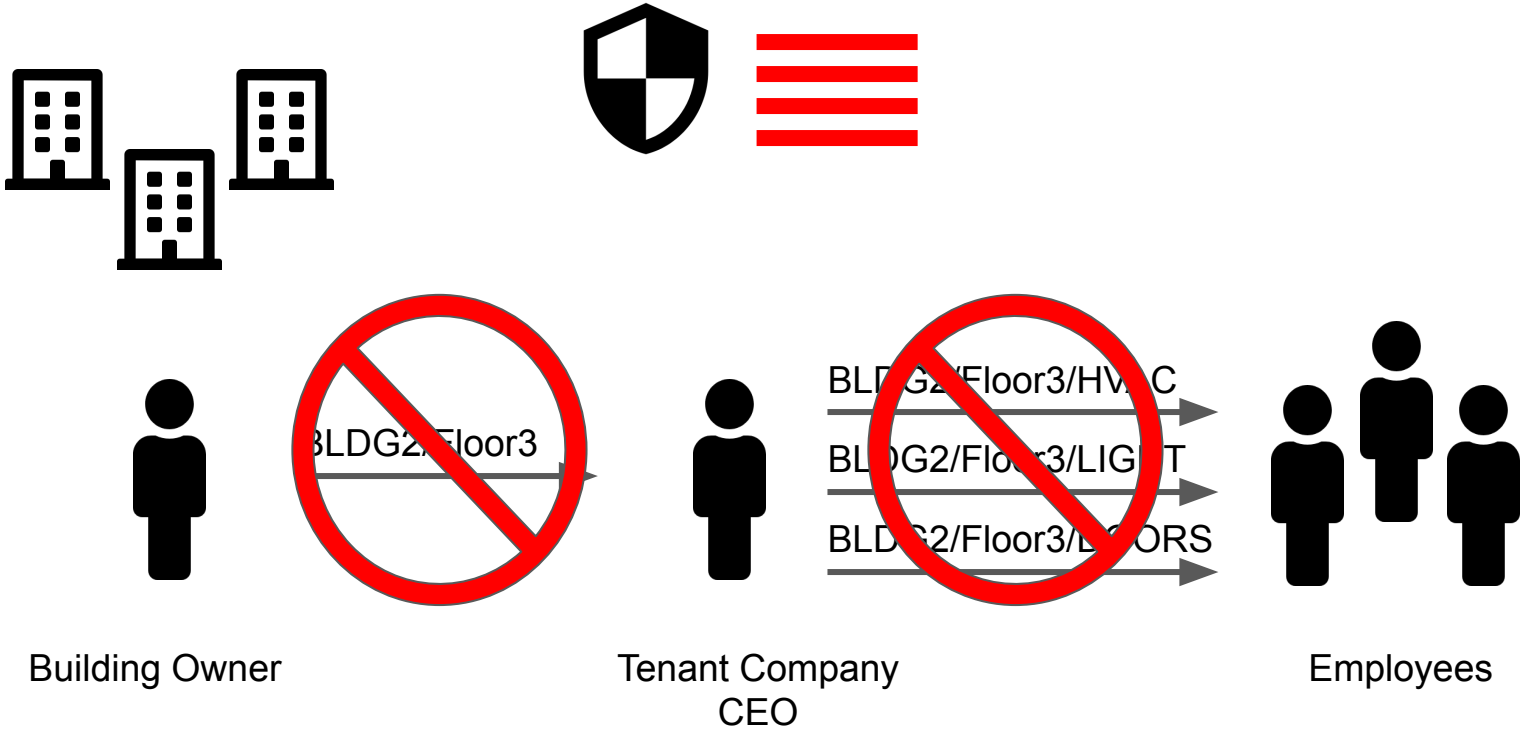
Lack of transitive delegation



Lack of transitive delegation



What we want:


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

Existing work lacks some important features

System / Work	Avoid central authority	Transitive Delegation	Permission Discovery	No ordering constraints	Offline participants	Protected permissions
LDAP, AD	✗	✓	✓	✓	✓	✗
OAuth2	✗	✓	✓	✓	✓	✗
Macaroons	✗	✓	–	✗	–	–
SDSI/SPKI	✗	✓	–	✓	–	–
Distributed TM	✓	✓	✓	✓	✗	✗

Existing work lacks some important features

System / Work	Avoid central authority	Transitive Delegation	Permission Discovery	No ordering constraints	Offline participants	Protected permissions
LDAP, AD	✗	✓	✓	✓	✓	✗
OAuth2	✗	✓	✓	✓	✓	✗
Macaroons	✗	✓	–	✗	–	–
SDSI/SPKI	✗	✓	–	✓	–	–
Distributed TM	✓	✓	✓	✓	✗	✗

Existing work lacks some important features

System / Work	Avoid central authority	Transitive Delegation	Permission Discovery	No ordering constraints	Offline participants	Protected permissions
LDAP, AD	✗	✓	✓	✓	✓	✗
OAuth2	✗	✓	✓	✓	✓	✗
Macaroons	✗	✓	–	✗	–	–
SDSI/SPKI	✗	✓	–	✓	–	–
Distributed TM	✓	✓	✓	✓	✗	✗

Existing work lacks some important features

System / Work	Avoid central authority	Transitive Delegation	Permission Discovery	No ordering constraints	Offline participants	Protected permissions
LDAP, AD	✗	✓	✓	✓	✓	✗
OAuth2	✗	✓	✓	✓	✓	✗
Macaroons	✗	✓	–	✗	–	–
SDSI/SPKI	✗	✓	–	✓	–	–
Distributed TM	✓	✓	✓	✓	✗	✗

WAVE is designed to provide these

System / Work	Avoid central authority	Transitive Delegation	Permission Discovery	No ordering constraints	Offline participants	Protected permissions
LDAP, AD	✗	✓	✓	✓	✓	✗
OAuth2	✗	✓	✓	✓	✓	✗
Macaroons	✗	✓	–	✗	–	–
SDSI/SPKI	✗	✓	–	✓	–	–
Distributed TM	✓	✓	✓	✓	✗	✗
WAVE	✓	✓	✓	✓	✓	✓

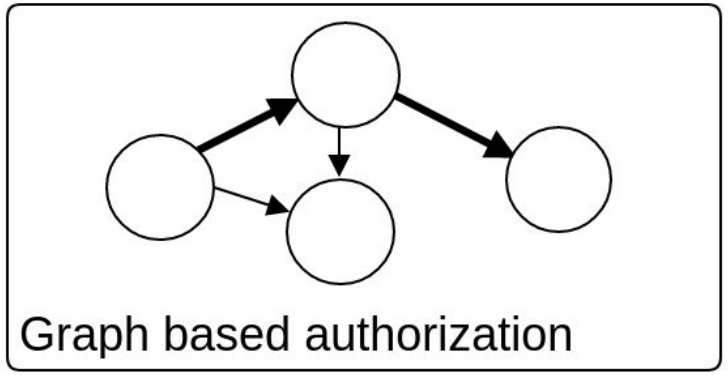
What is WAVE

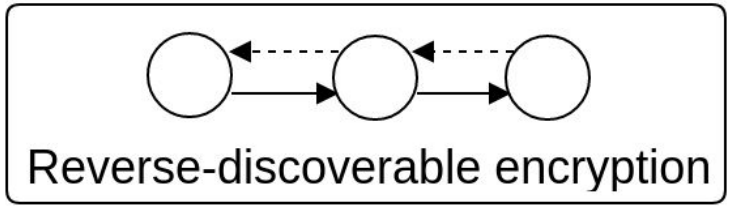
System / Work	No central authority	Transitive Delegation	Permission Discovery	No ordering constraints	Offline participants	Protected permissions
WAVE	✓	✓	✓	✓	✓	✓

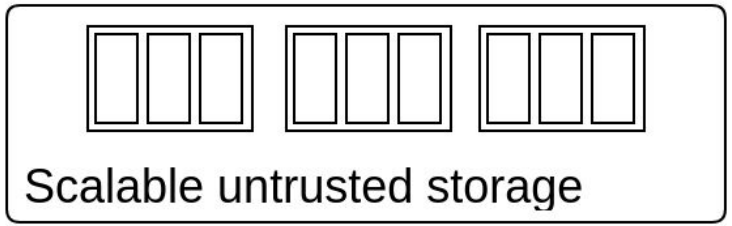
WAVE is a cryptographically enforced decentralized authorization system

- 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:

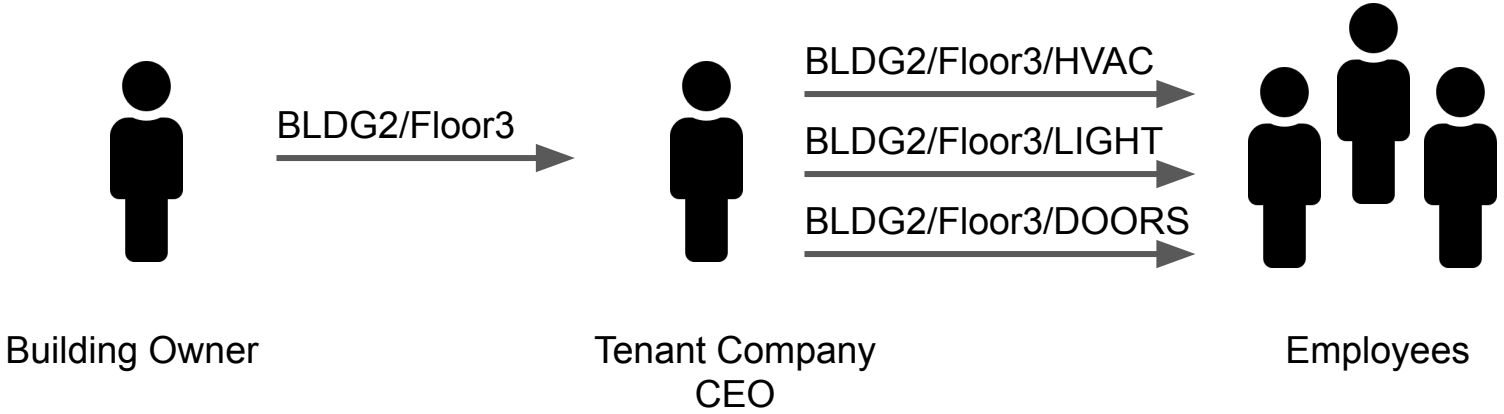




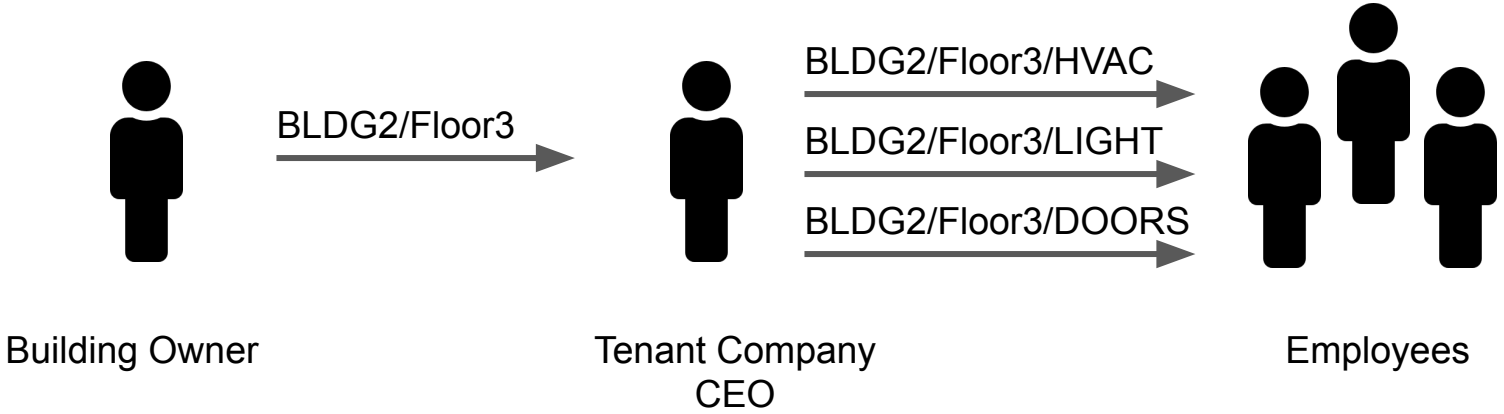


Graph Based Authorization

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

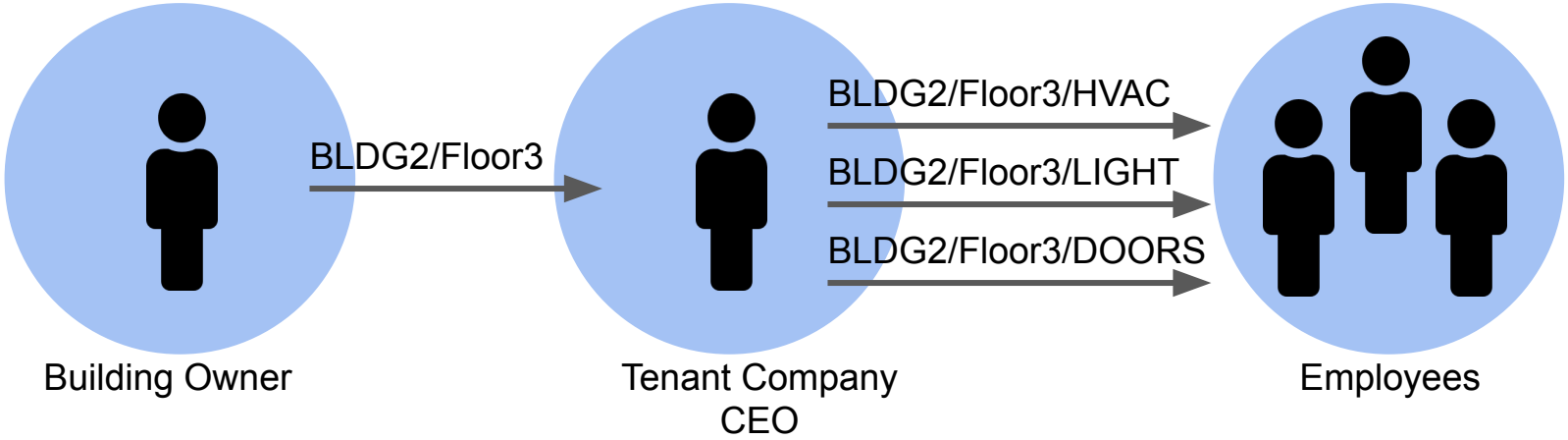


Graph Based Authorization



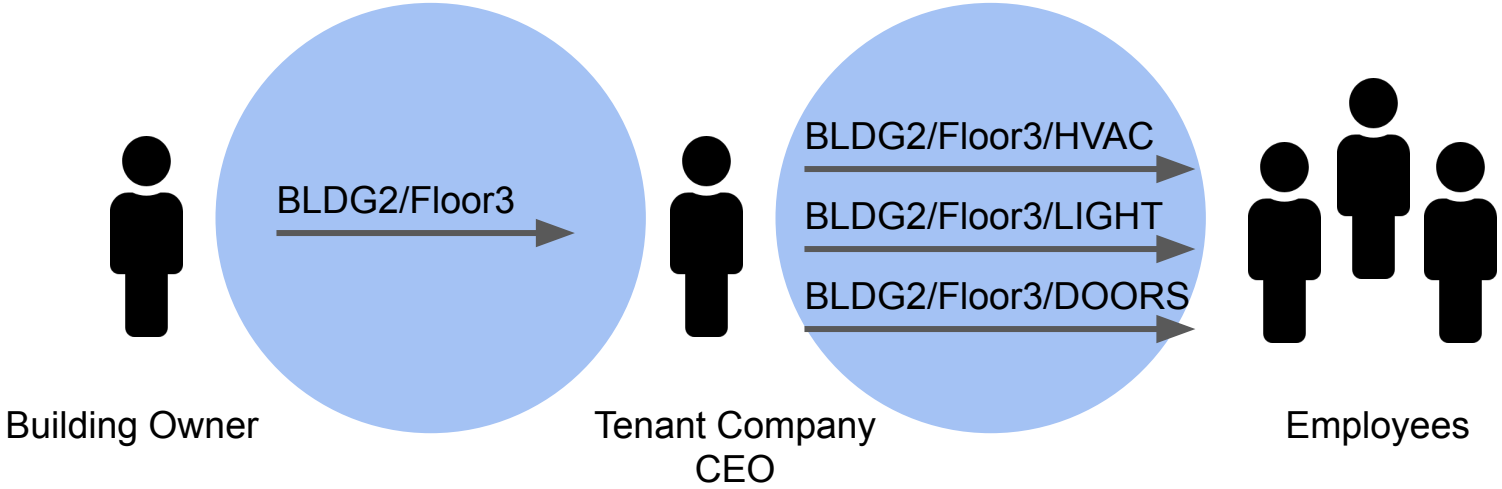
Graph Based Authorization

Participants: Entities
Collections of cryptographic keys



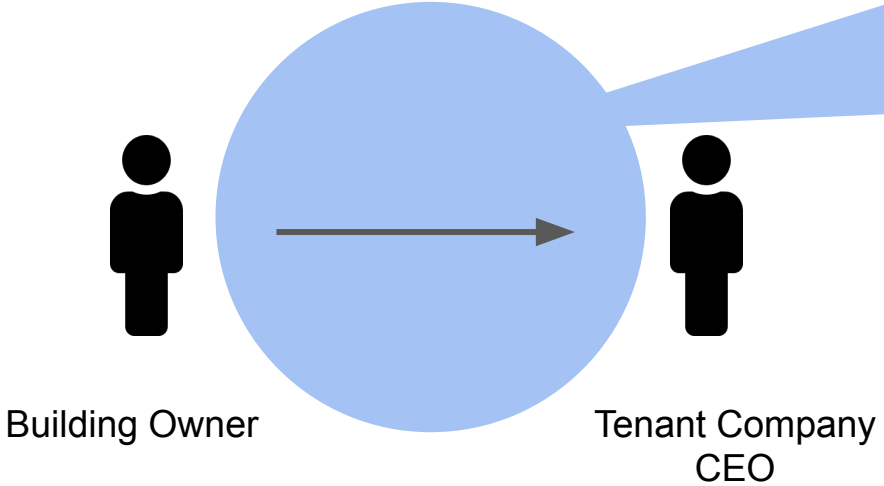
Graph Based Authorization

Grants of permissions: Attestations
Signed certificates created by Entities



Graph Based Authorization

Attestations grant permissions on a **resource**



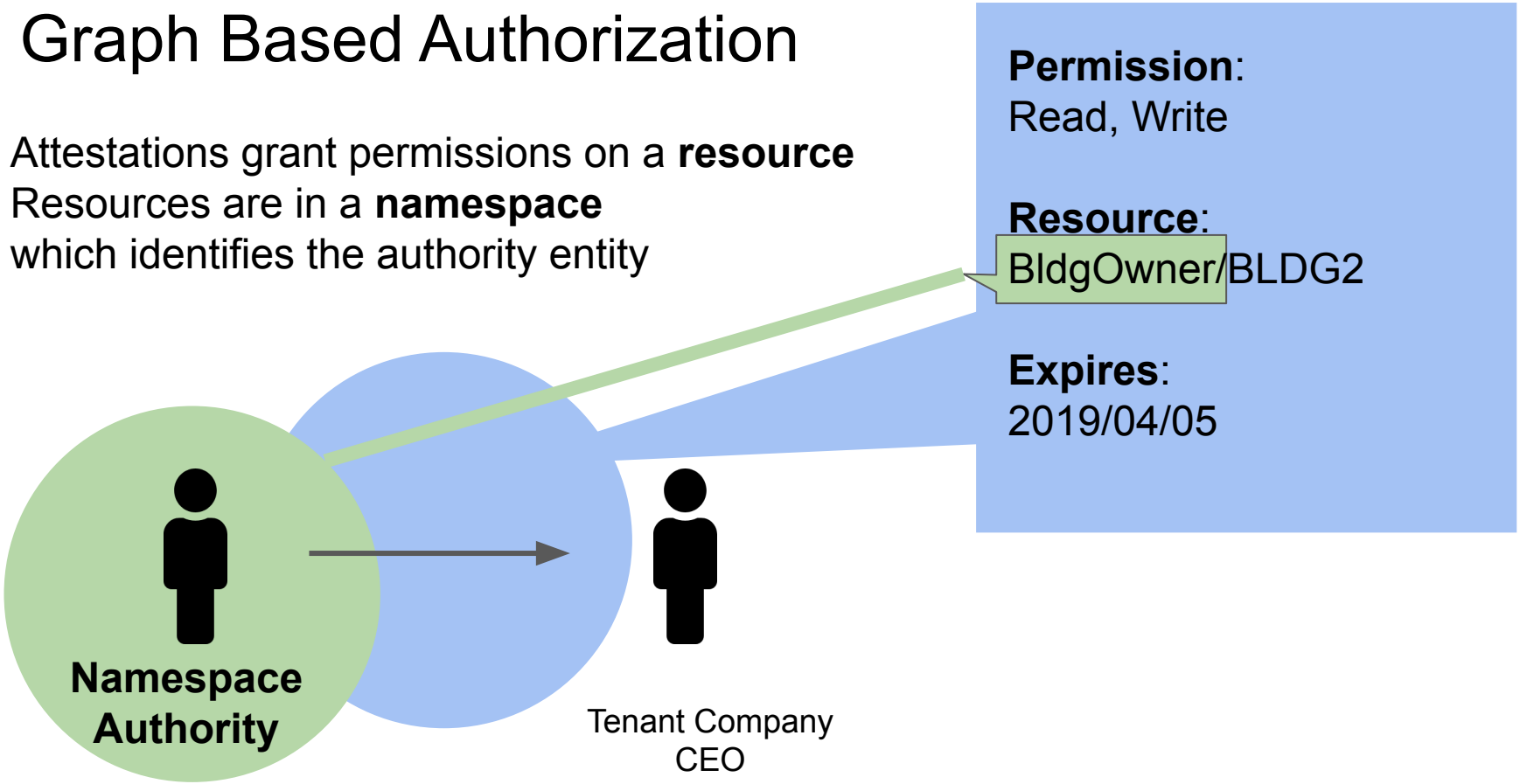
Permission:
Read, Write

Resource:
BldgOwner/BLDG2

Expires:
2019/04/05

Graph Based Authorization

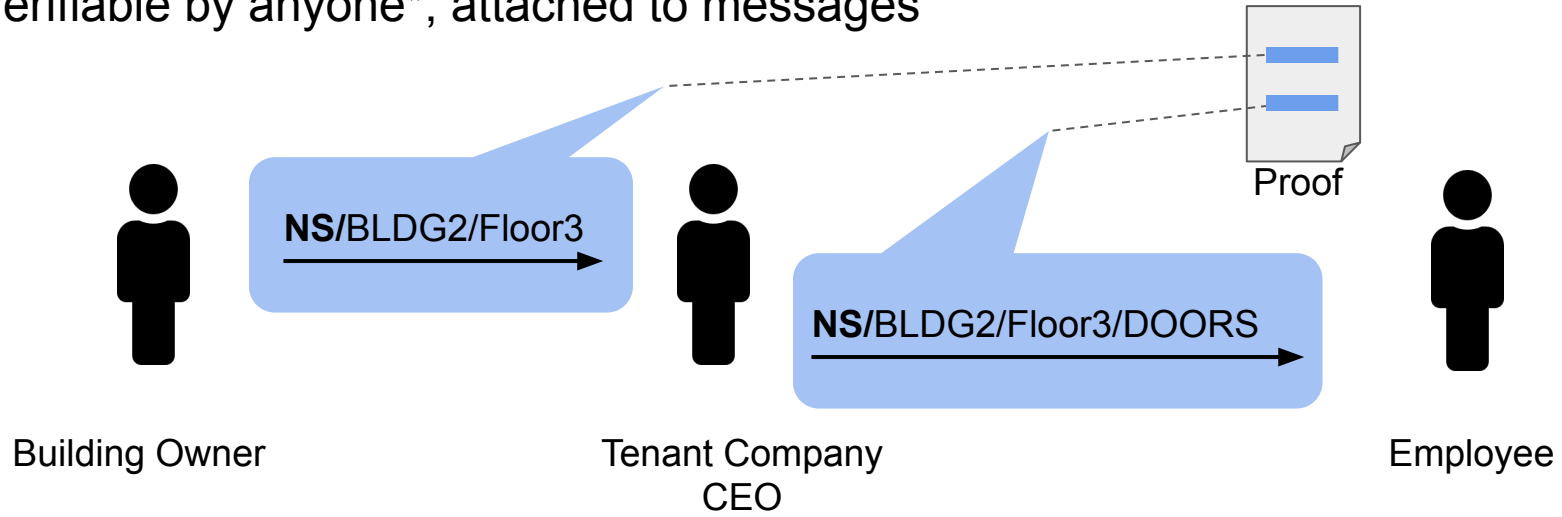
Attestations grant permissions on a **resource**
Resources are in a **namespace**
which identifies the authority entity



Graph Based Authorization

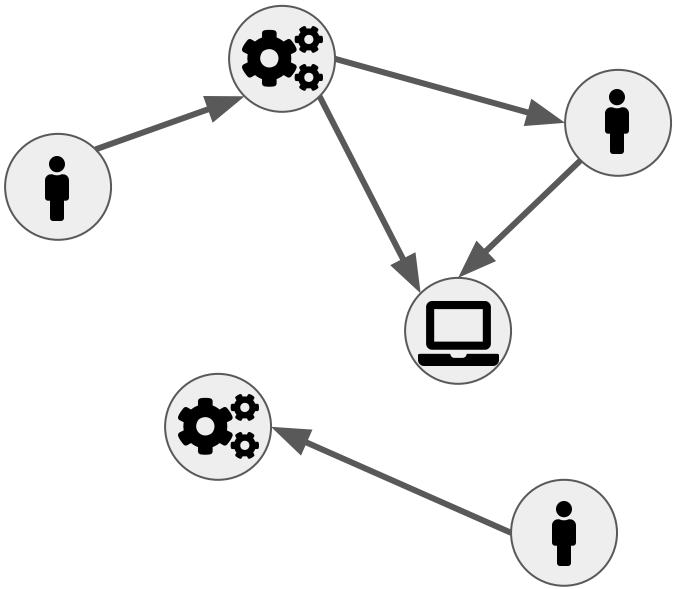
Proof of permissions: A path through the graph from Namespace Authority to the prover

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

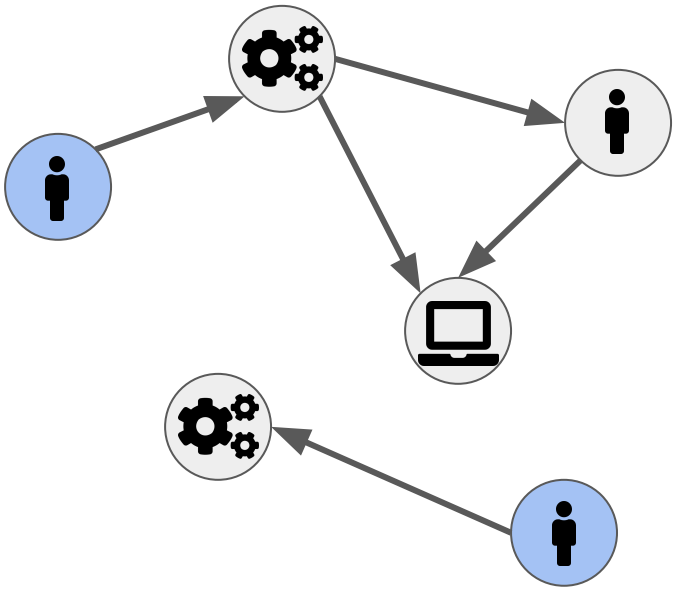


* In WAVE, not SDSI/SPKI

This forms a single global graph

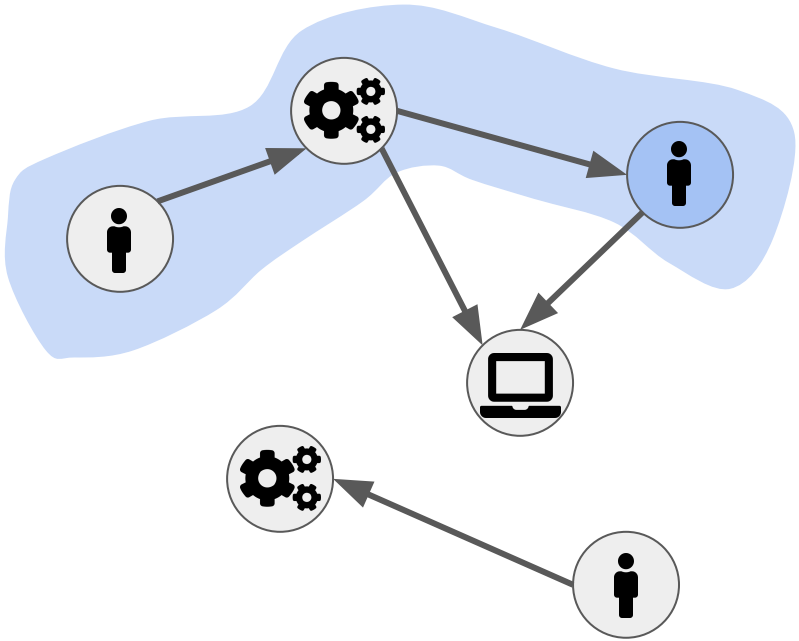


This forms a single global graph



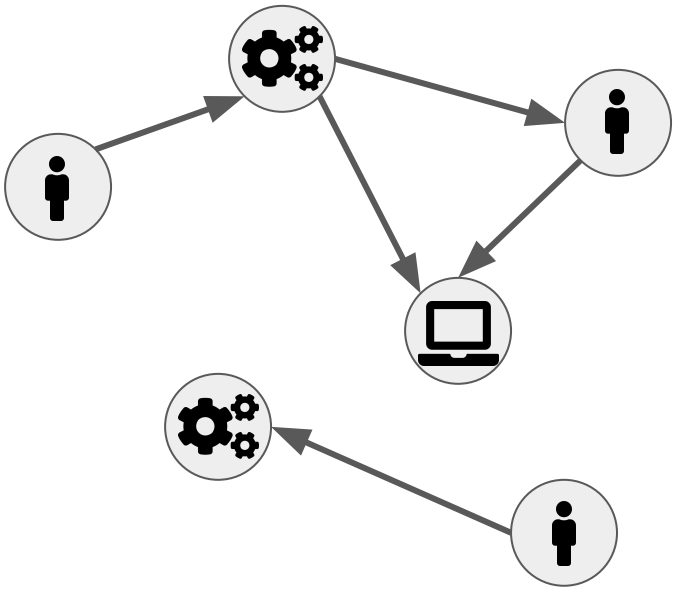
- Multiple namespace authorities in the graph

This forms a single global graph



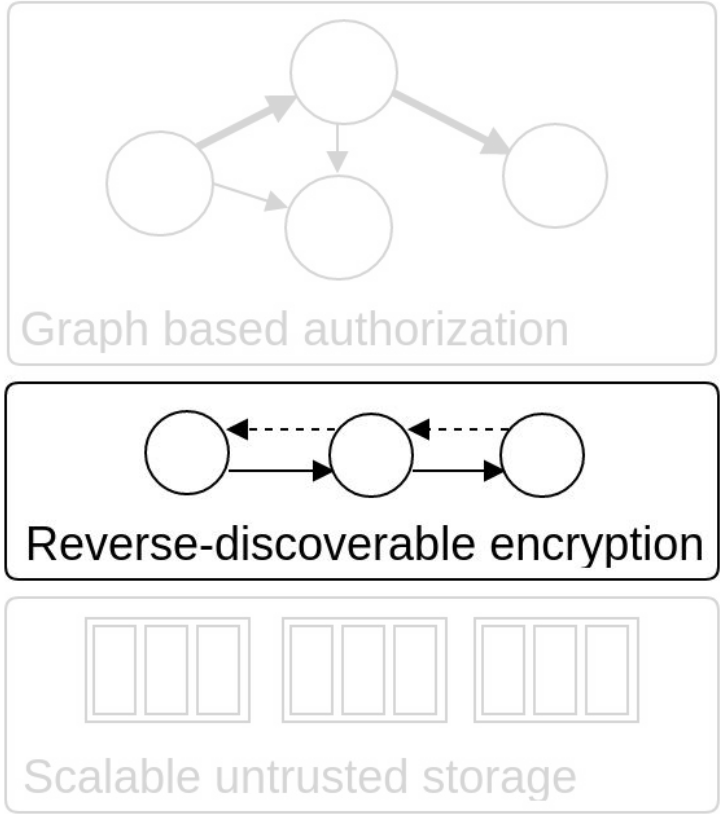
- Multiple namespace authorities in the graph
- Different entities will only see portions of the graph

This forms a single global graph

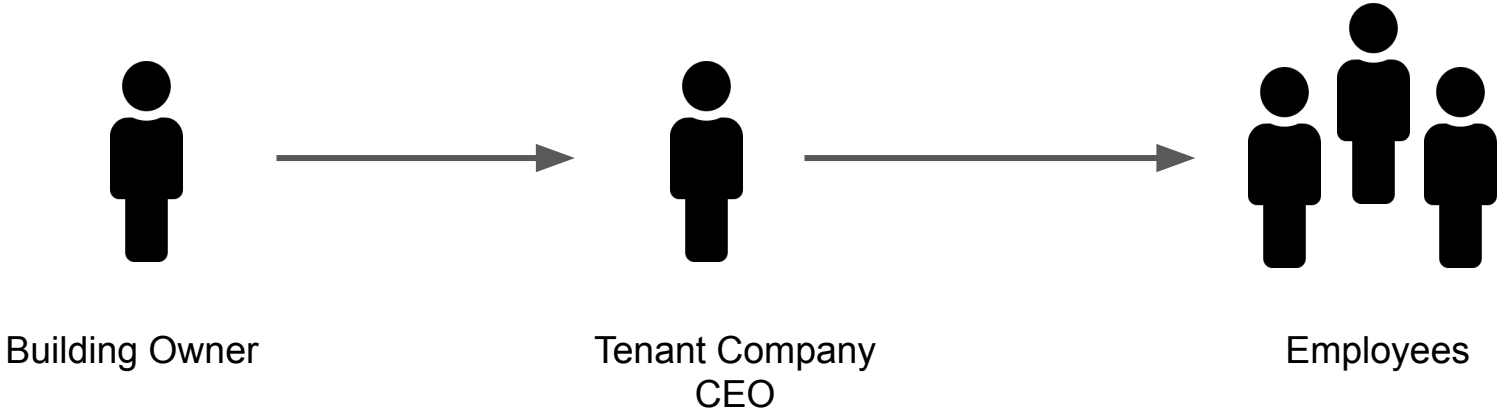


- Multiple namespace authorities in the graph
- Different entities will only see portions of the graph
- The graph is publicly accessible

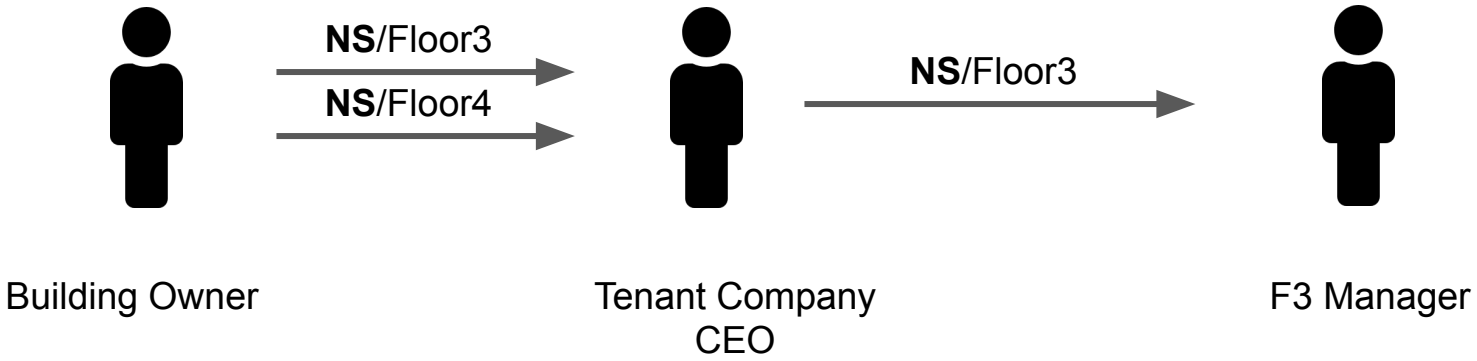
We need to hide portions of the graph



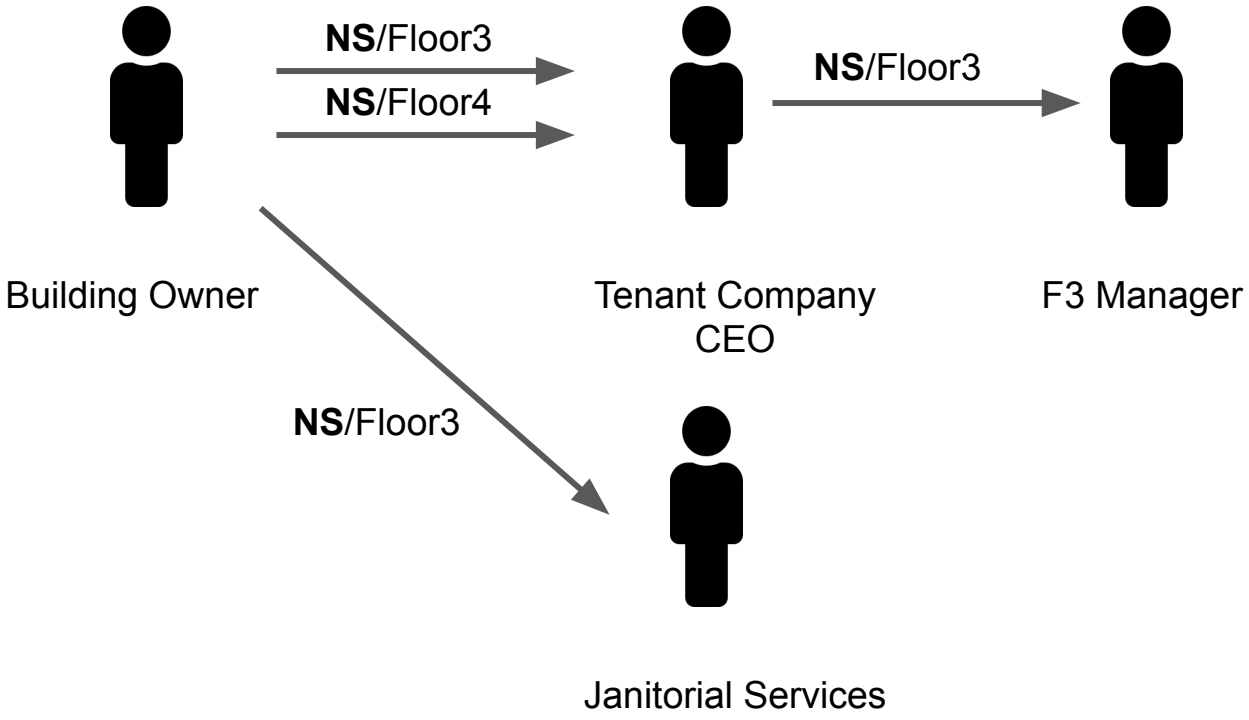
Reverse Discoverable Encryption



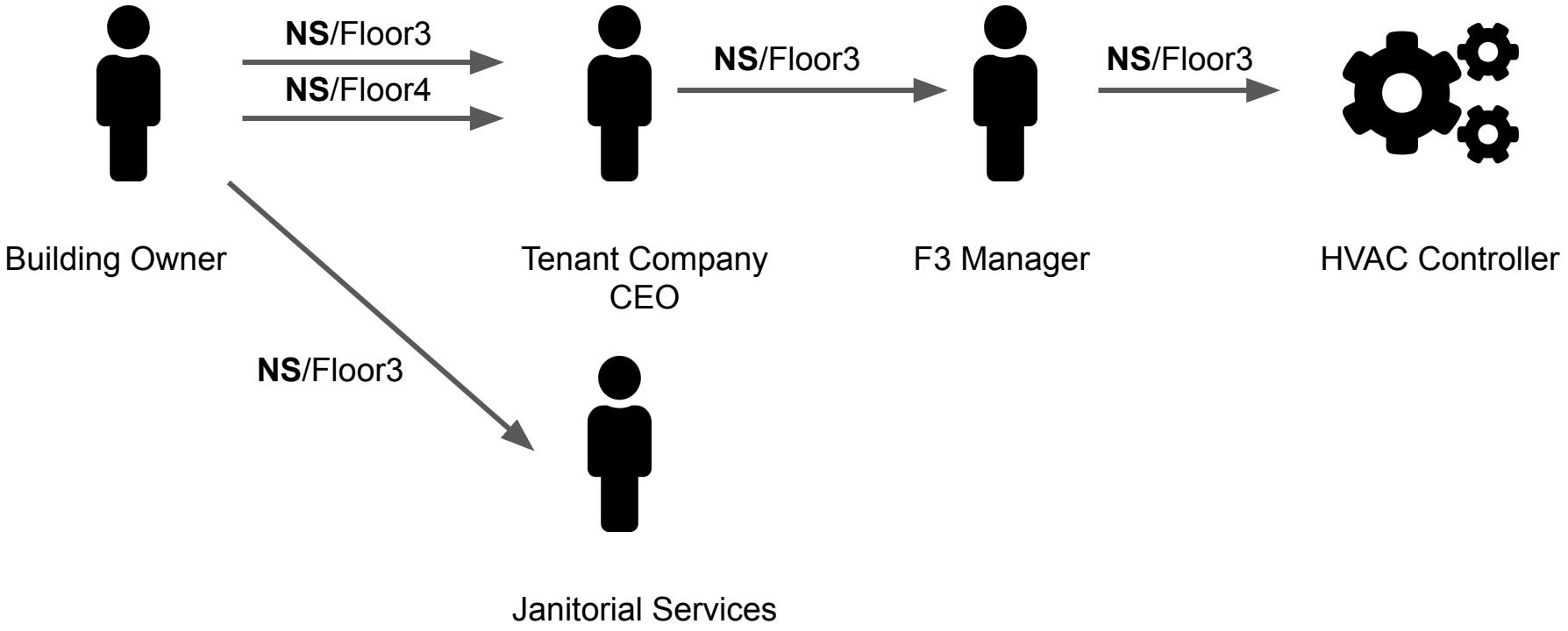
Reverse Discoverable Encryption



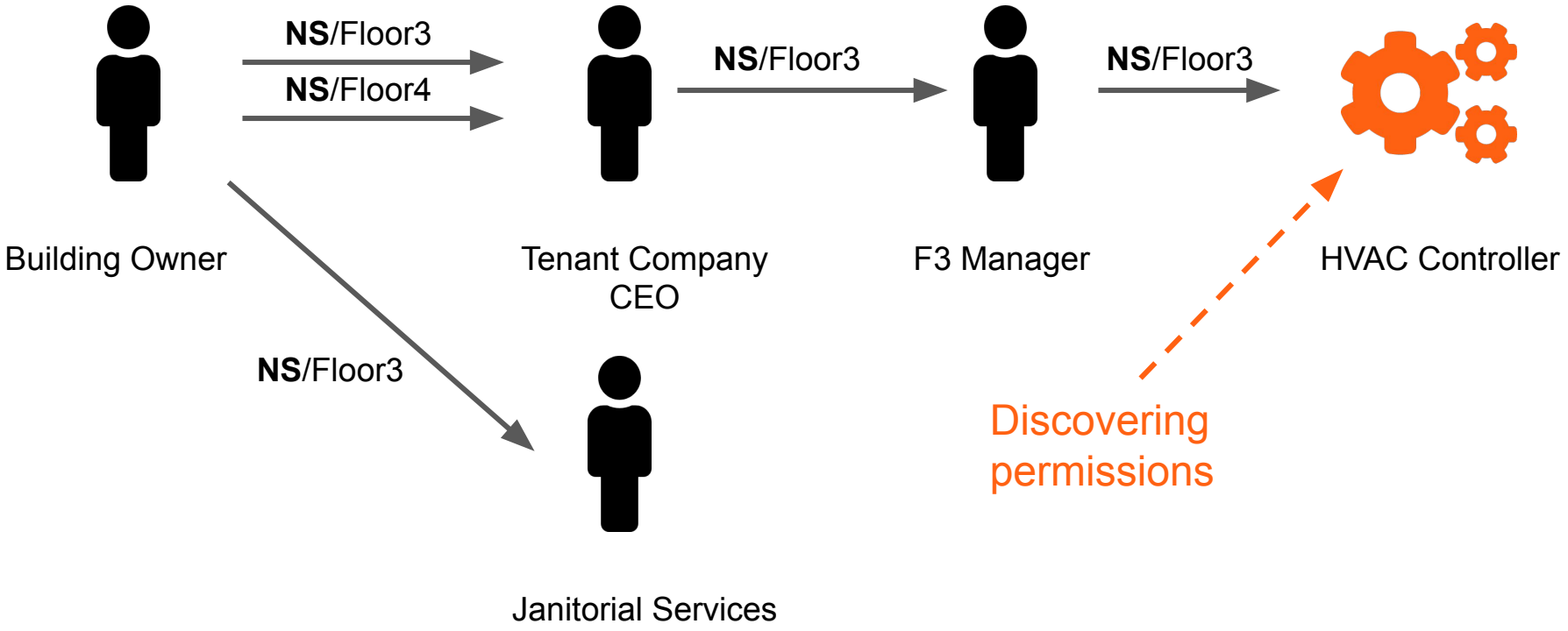
Reverse Discoverable Encryption



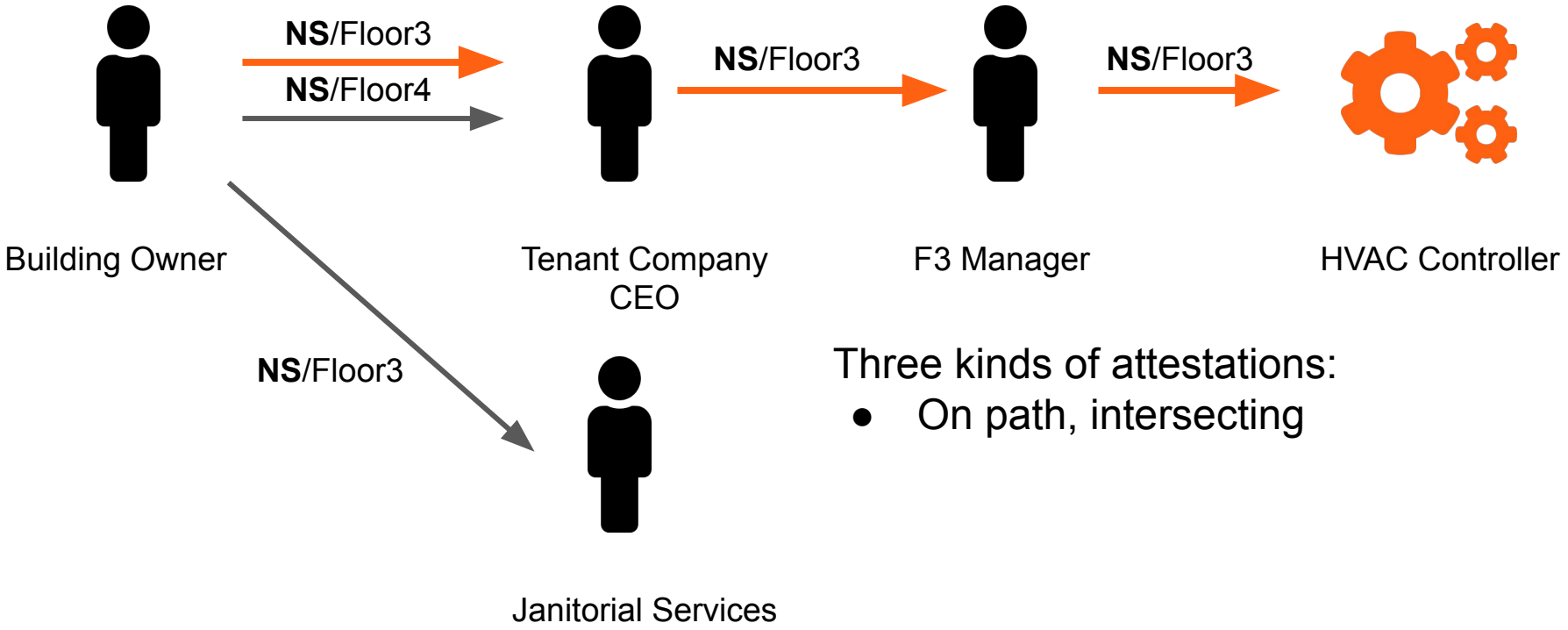
Reverse Discoverable Encryption



Reverse Discoverable Encryption



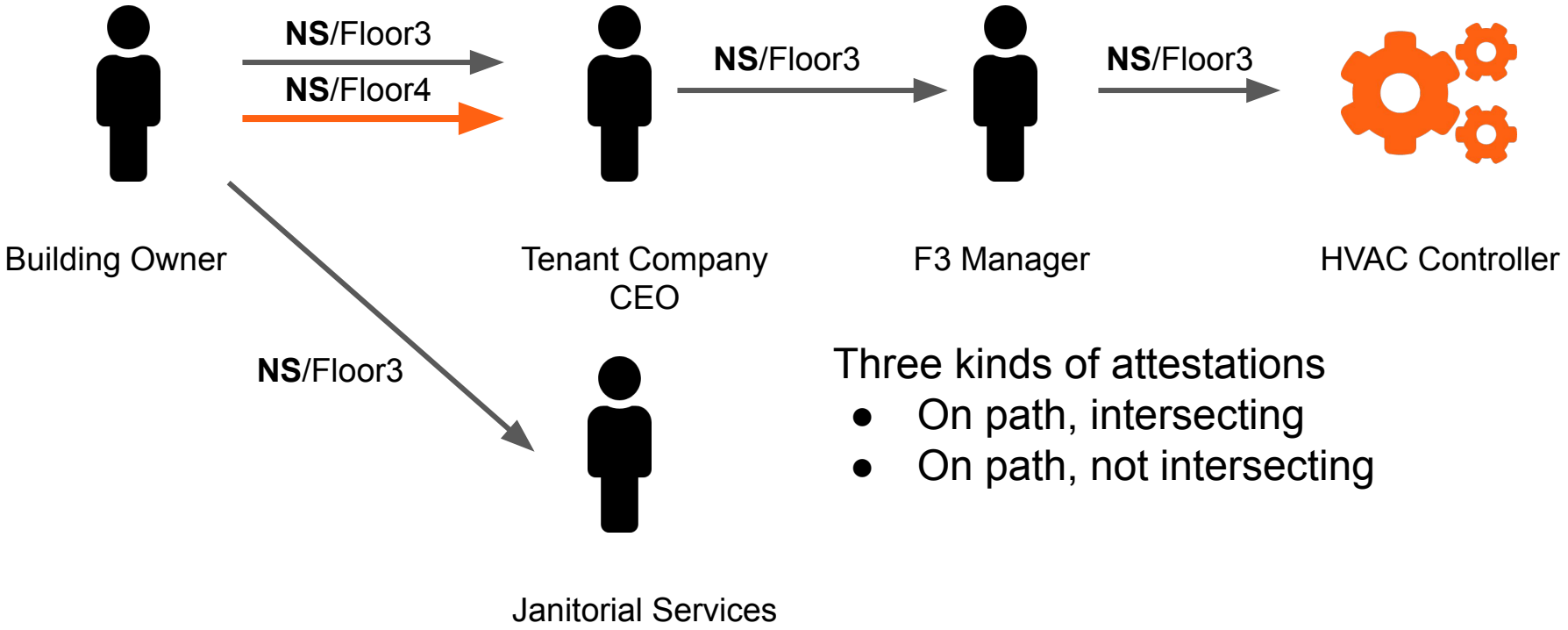
Reverse Discoverable Encryption



Three kinds of attestations:

- On path, intersecting

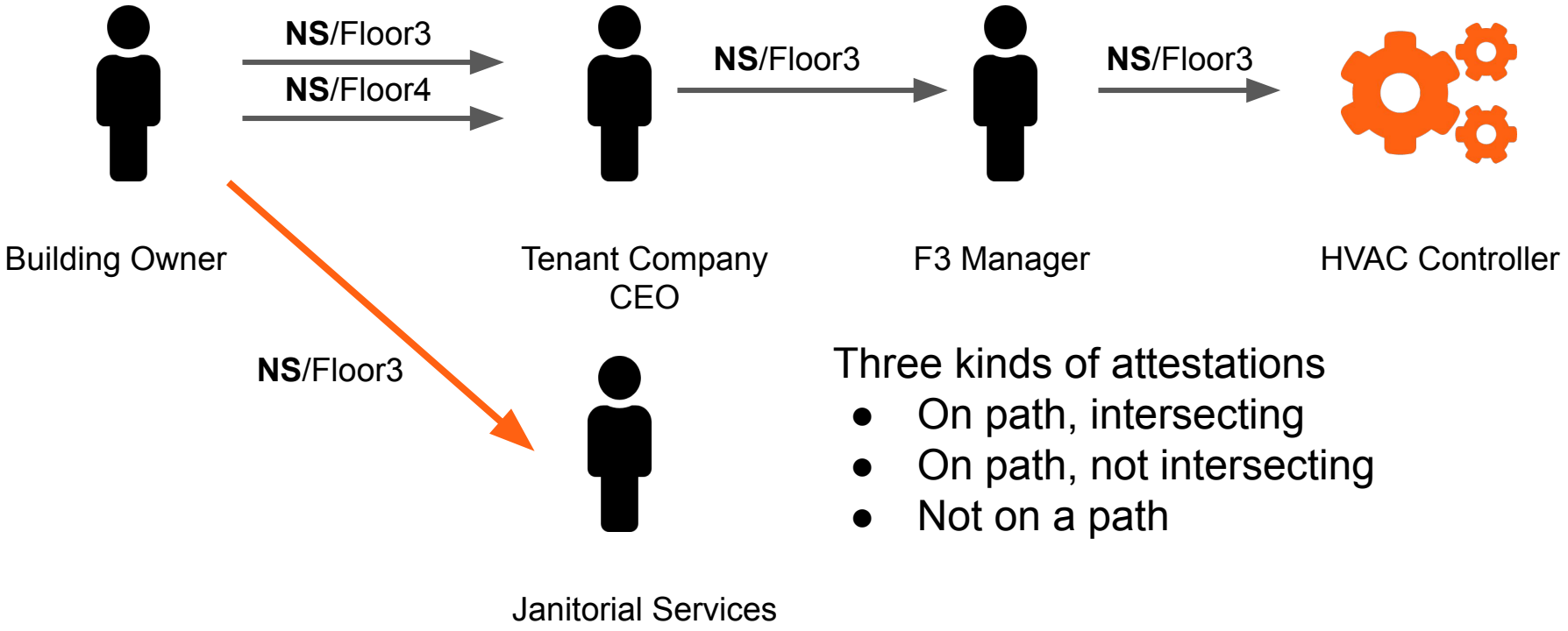
Reverse Discoverable Encryption



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

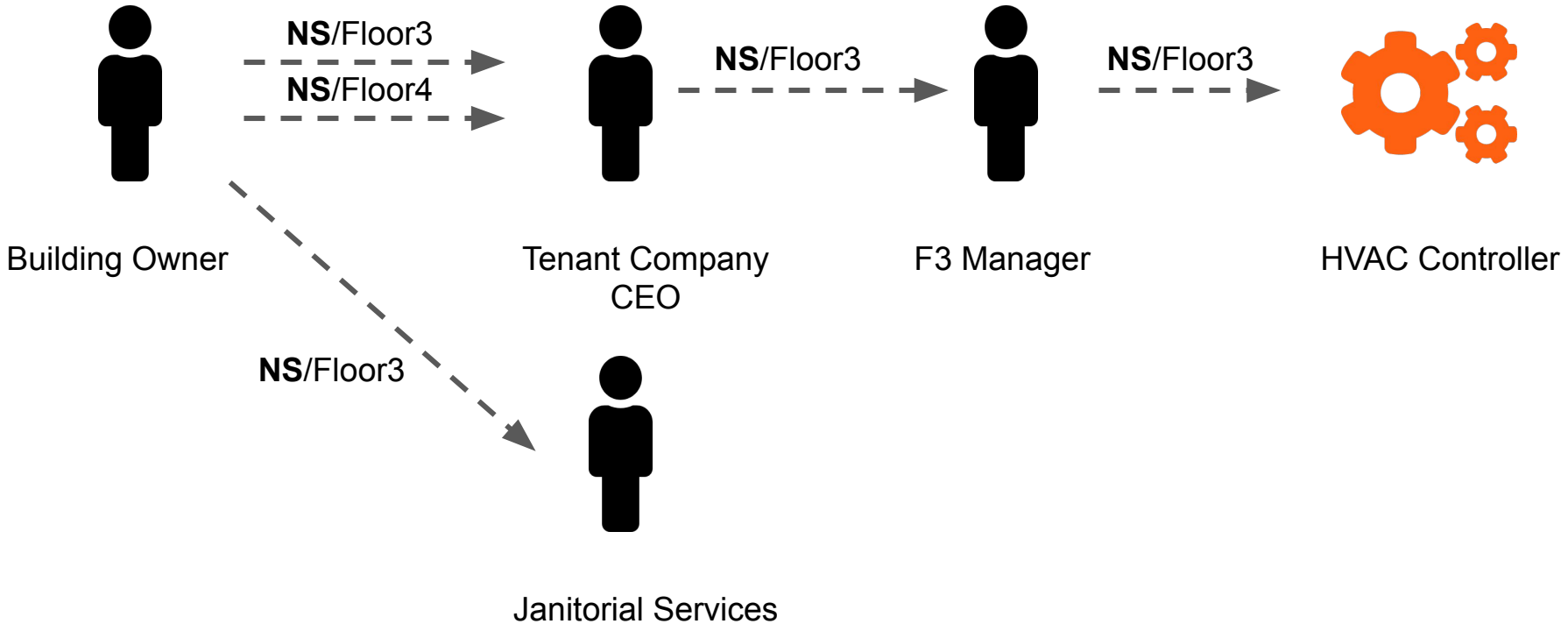
Technique in a nutshell

Encrypt attestations

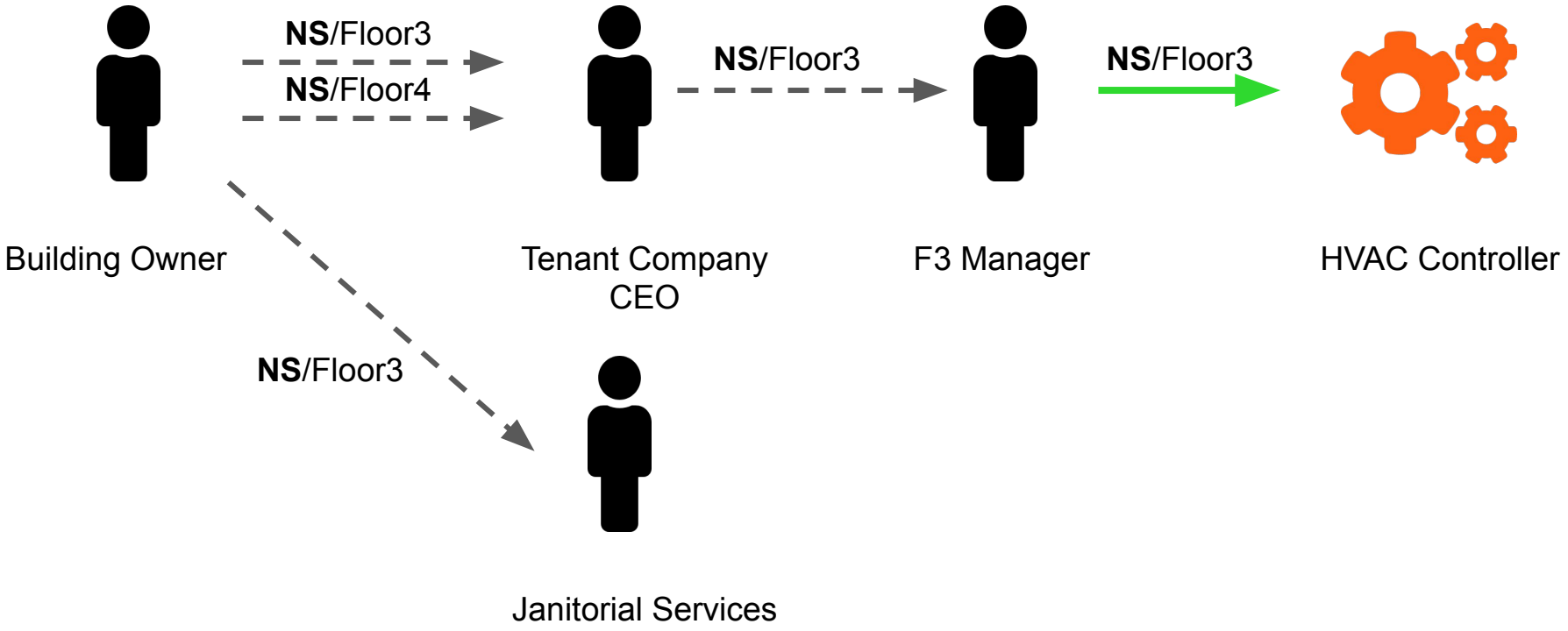
In each attestation, include a secret that allows you to decrypt upstream attestations that have intersecting permissions

(on path, intersecting)

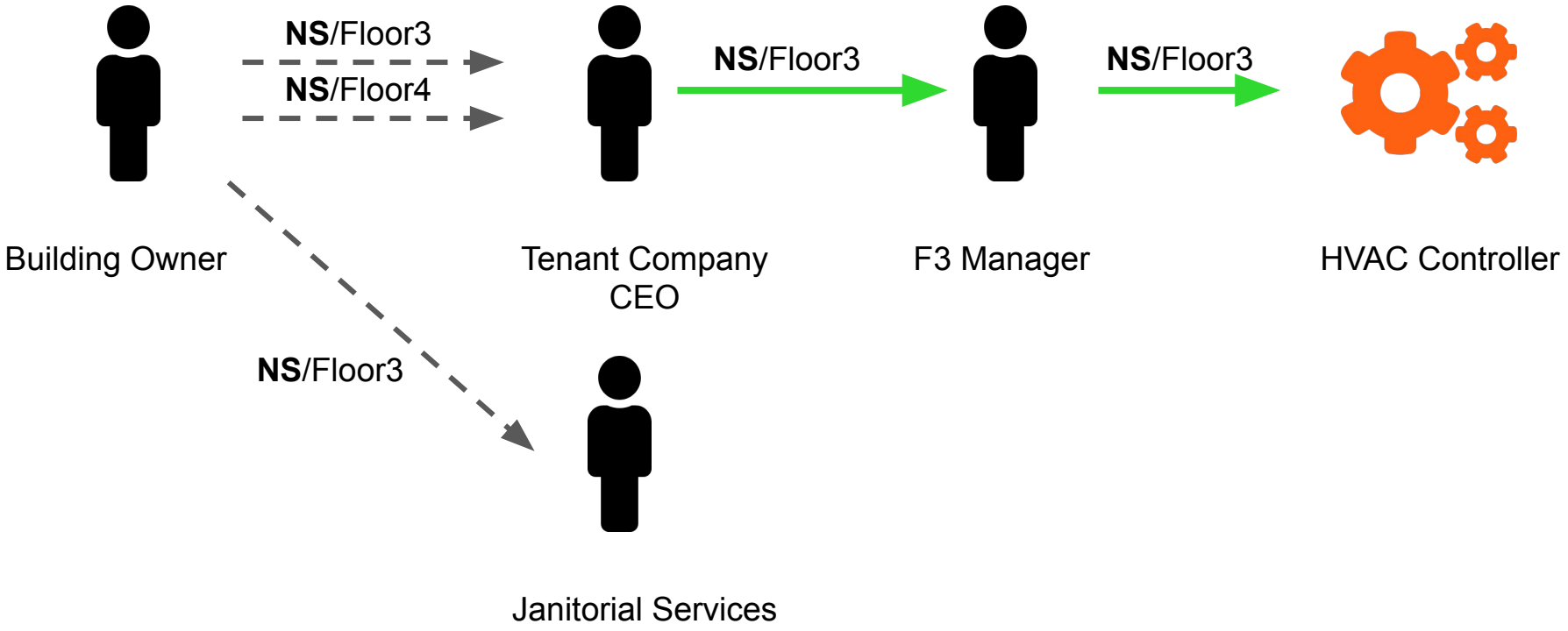
Reverse Discoverable Encryption



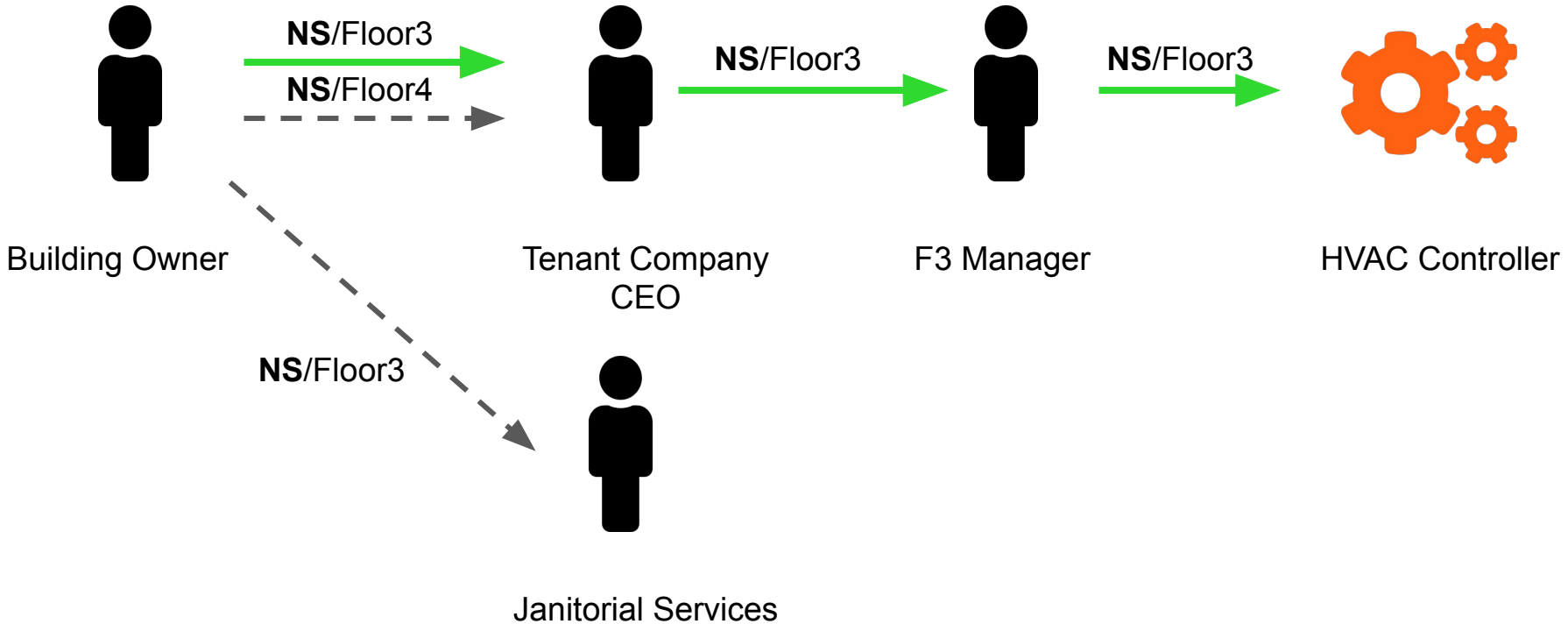
Reverse Discoverable Encryption



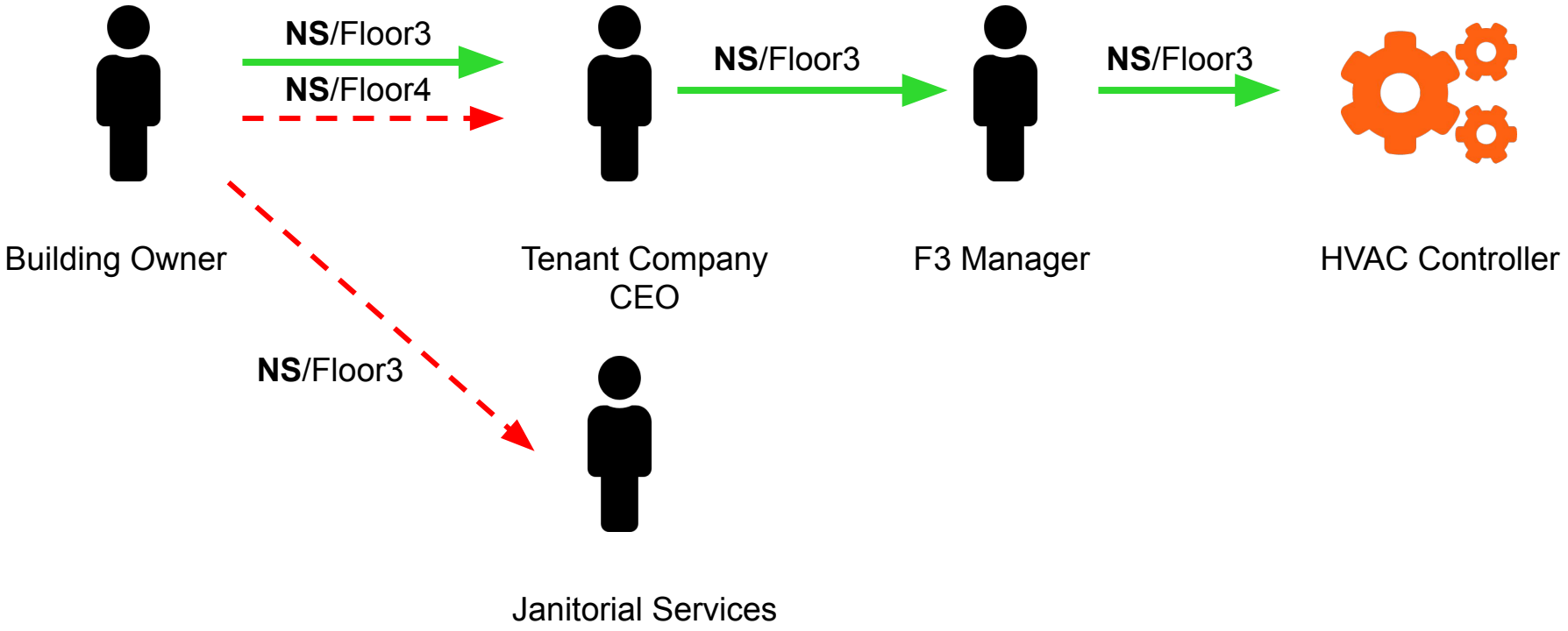
Reverse Discoverable Encryption



Reverse Discoverable Encryption



Reverse Discoverable Encryption

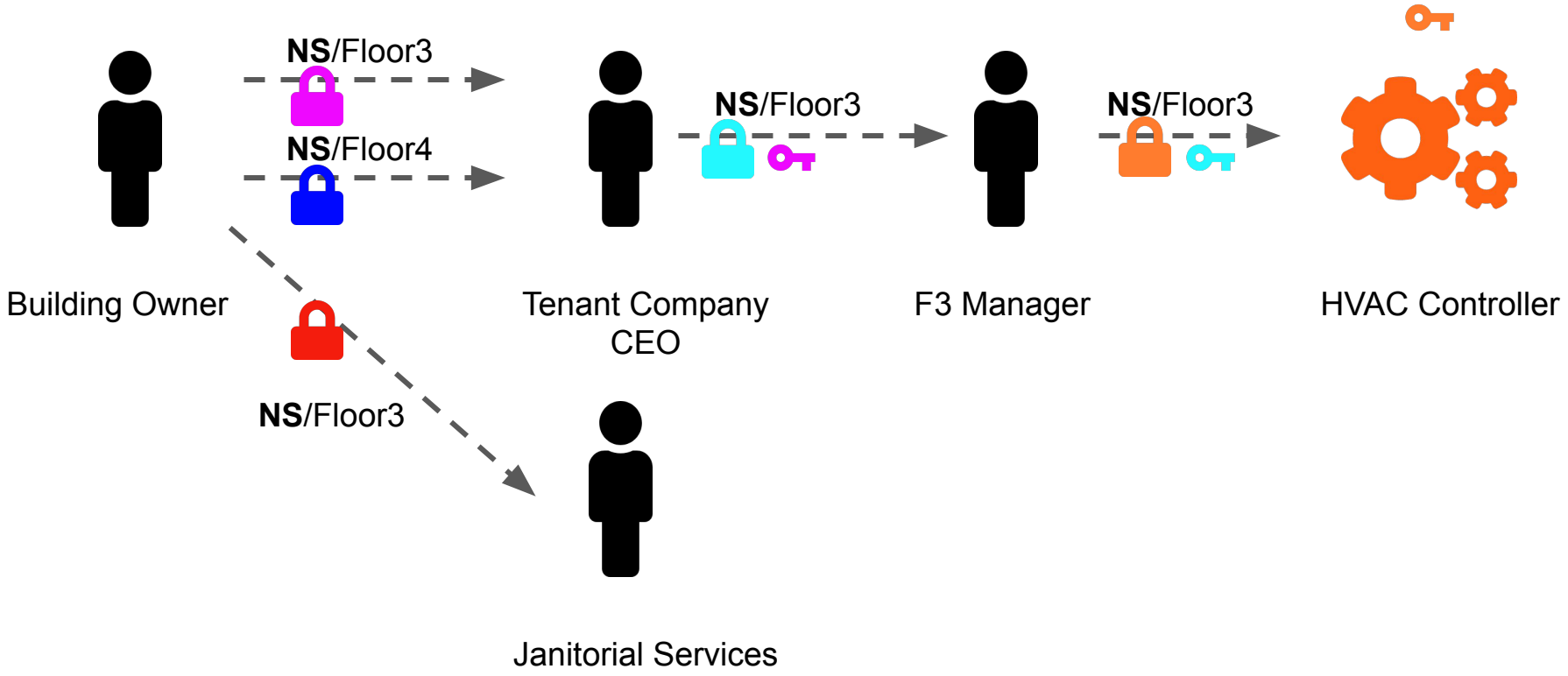


The encryption & secret must capture the permissions

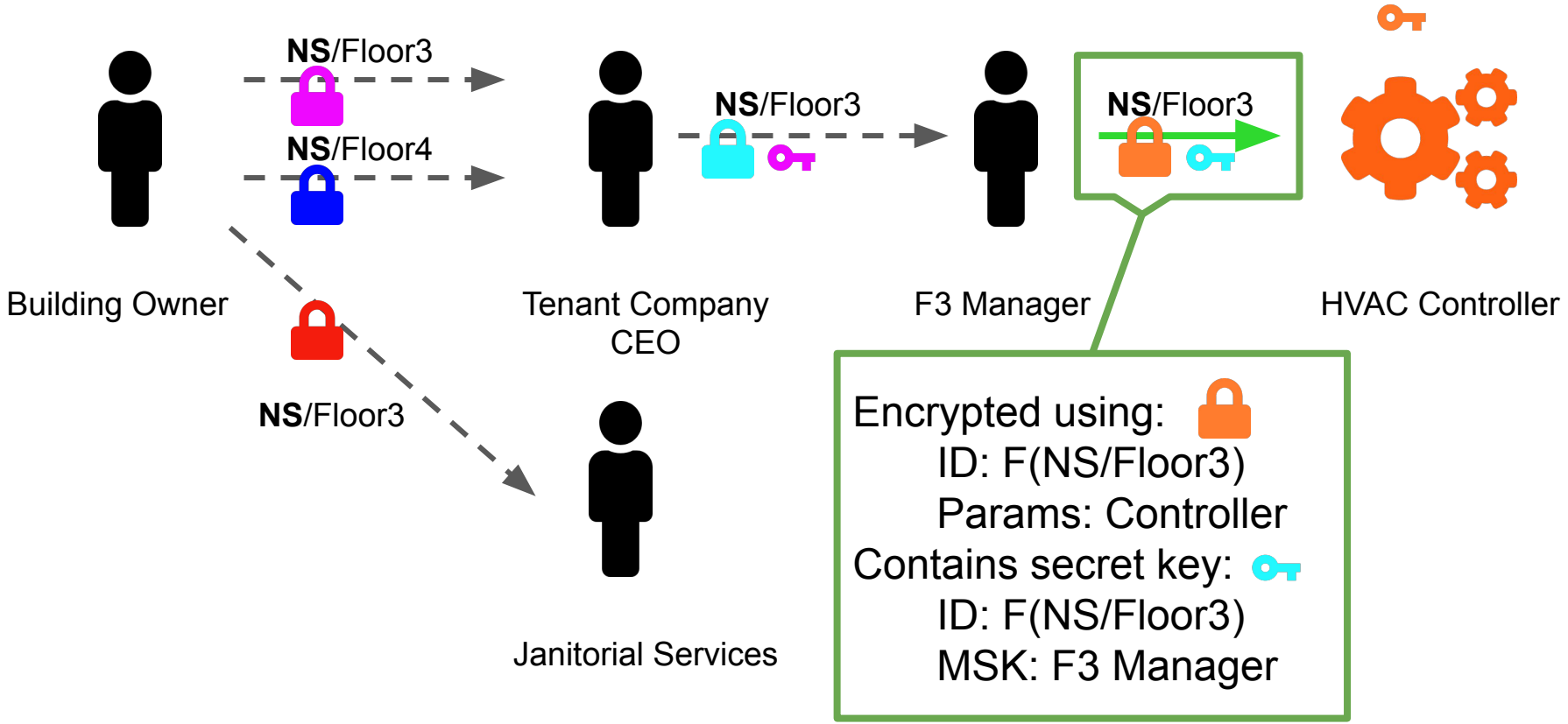
- We use Wildcard Identity Based Encryption (WIBE) [Abdalla, 2006]
- Every entity has a WIBE master key
  - No PKG, **every entity has their own system**
  - Used just for RDE, nothing else
- When you create attestation (grant permissions)
  - Form WIBE ID = F(permissions)
  - Generate private key for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

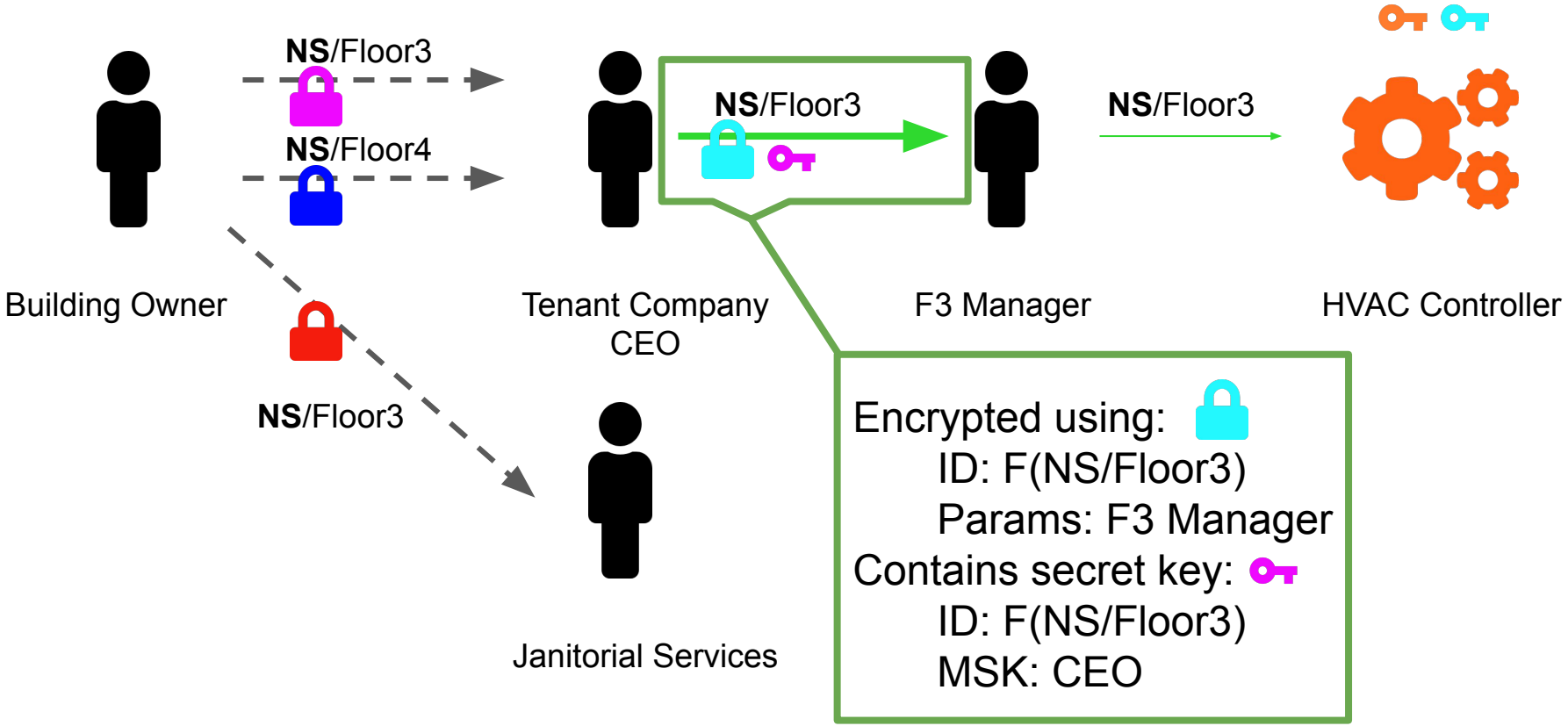
Reverse Discoverable Encryption



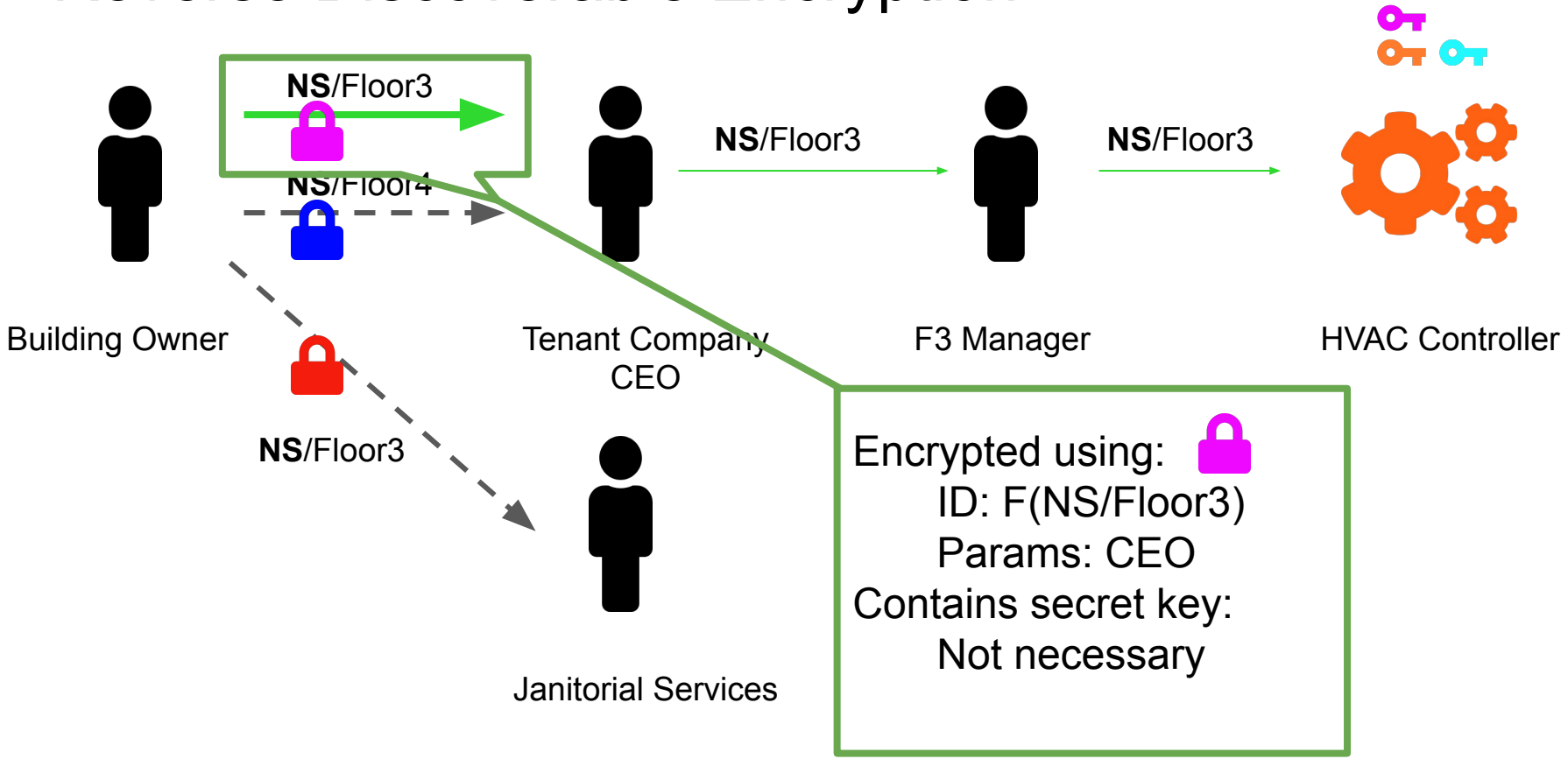
Reverse Discoverable Encryption



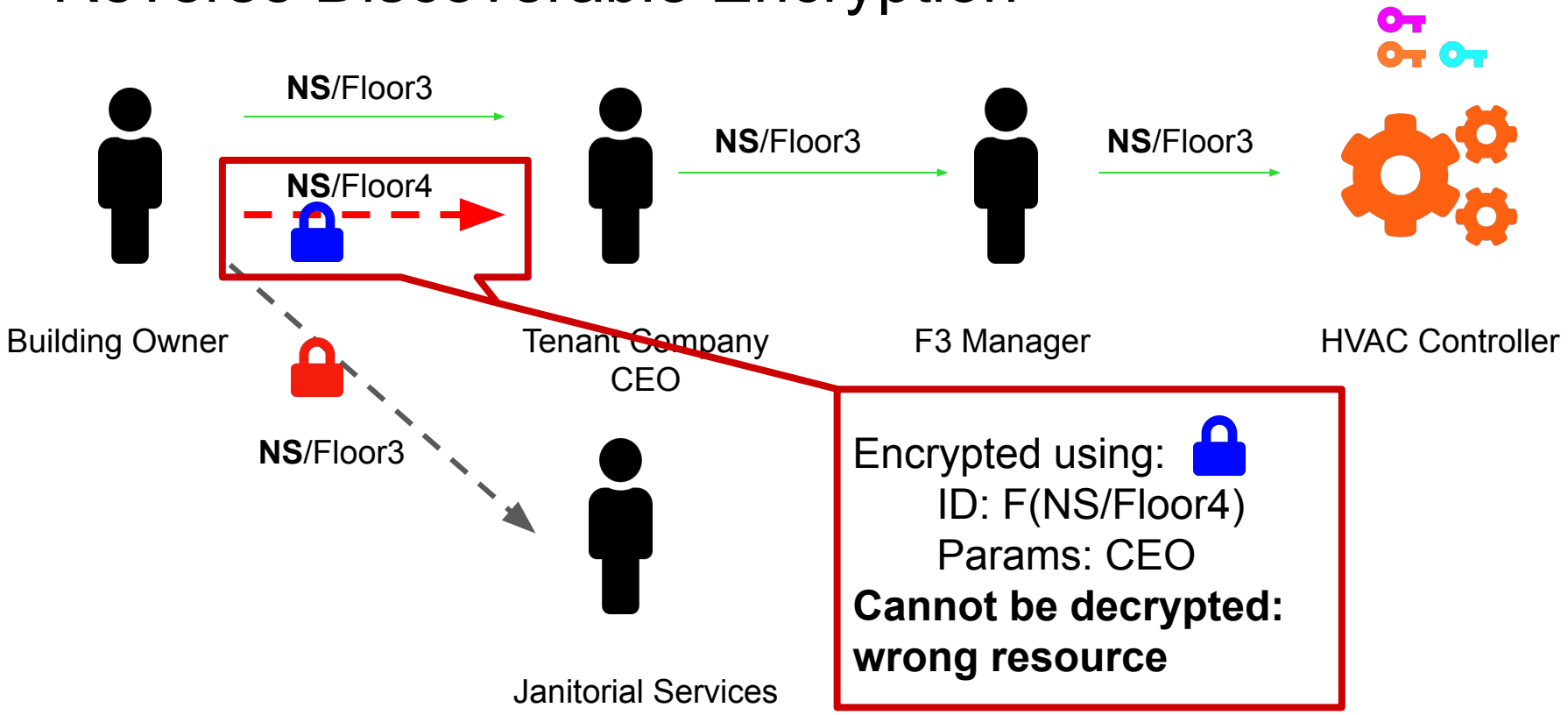
Reverse Discoverable Encryption



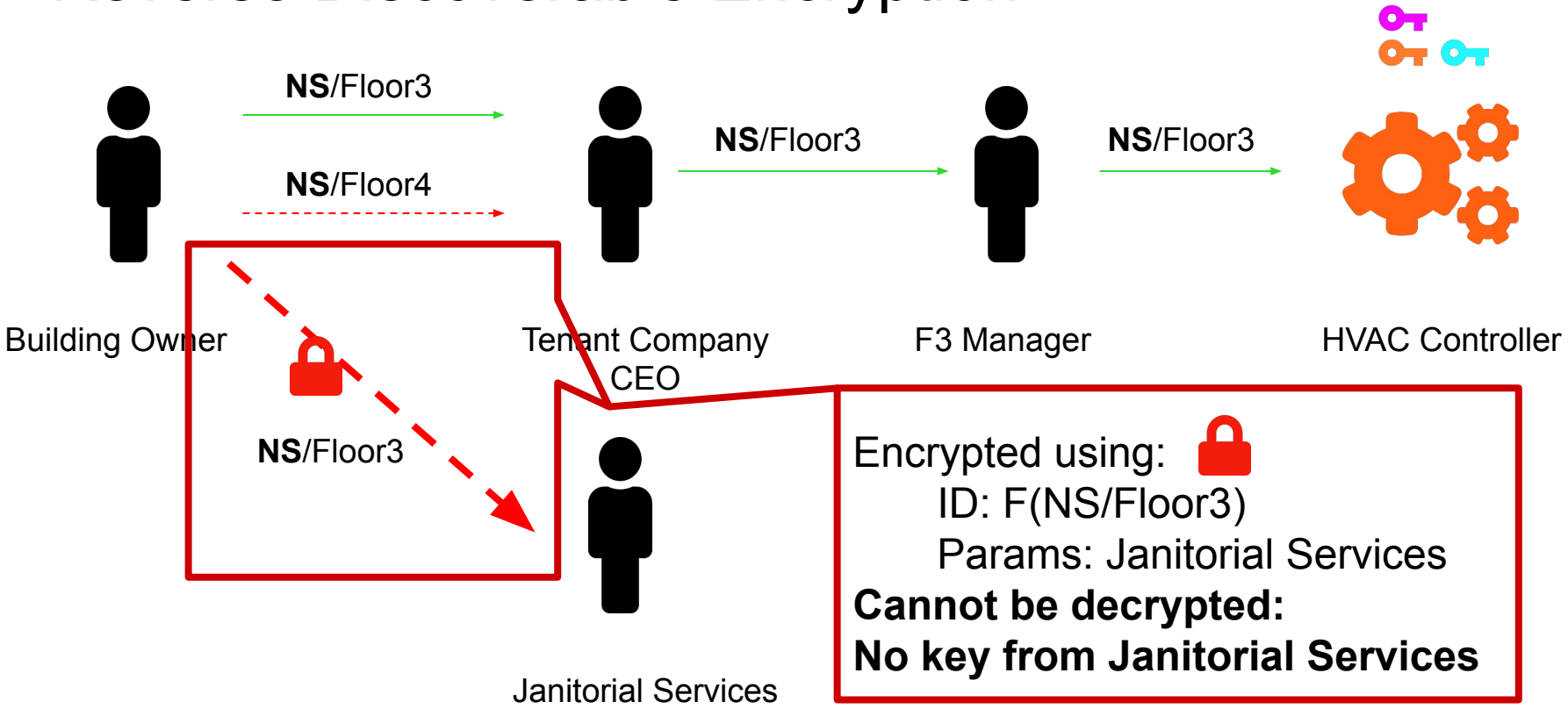
Reverse Discoverable Encryption



Reverse Discoverable Encryption



Reverse Discoverable Encryption

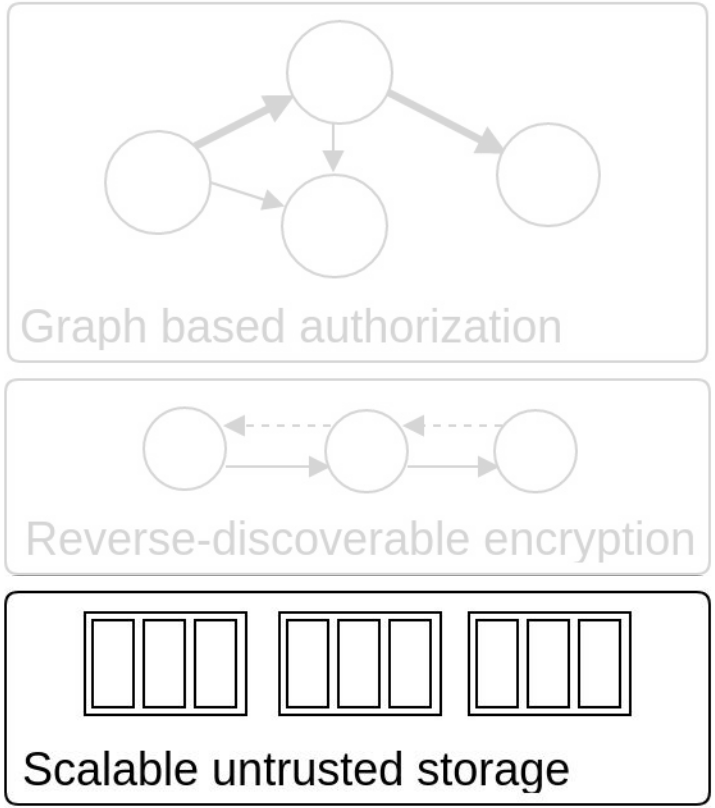


Reverse Discoverable Encryption Summary

- 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



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 (attestation objects)

Unequivocable Log Derived Map

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

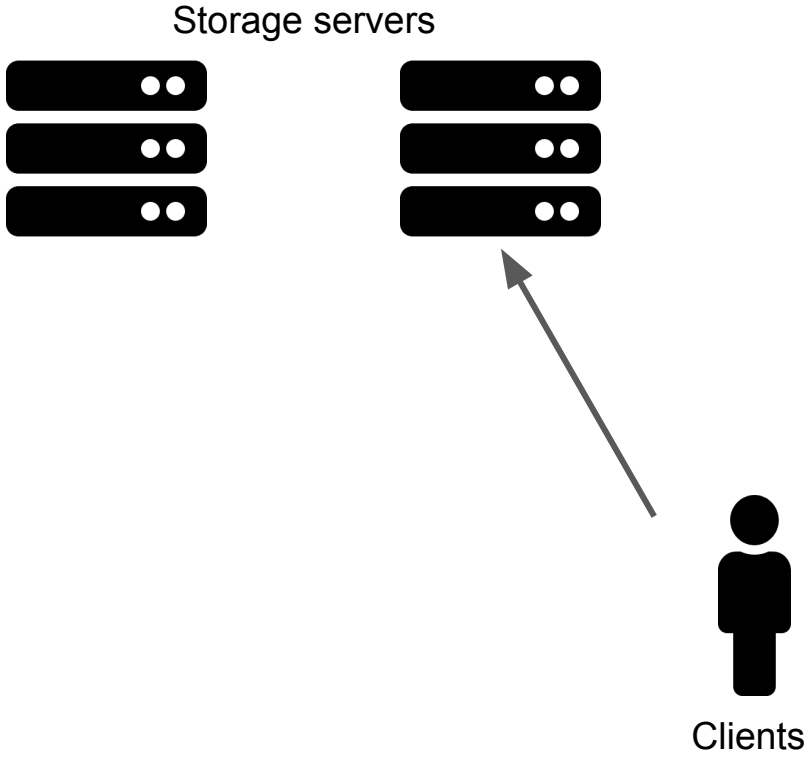
Horizontally scalable public ledger with cryptographic integrity proofs

similar to Certificate Transparency or Key Transparency, except:

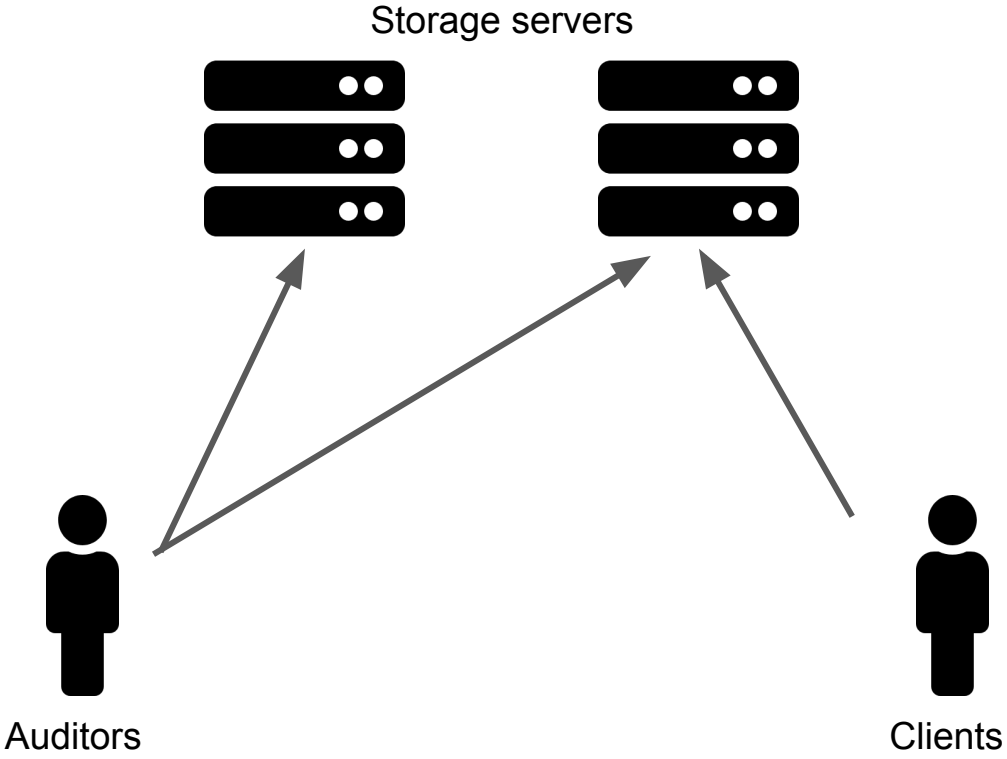
1) It supports proof of non-existence, which allows revocation
2) It has efficient auditing

- Clients only rarely communicate with auditors
- Auditing load scales with number of additions to storage, not size of storage

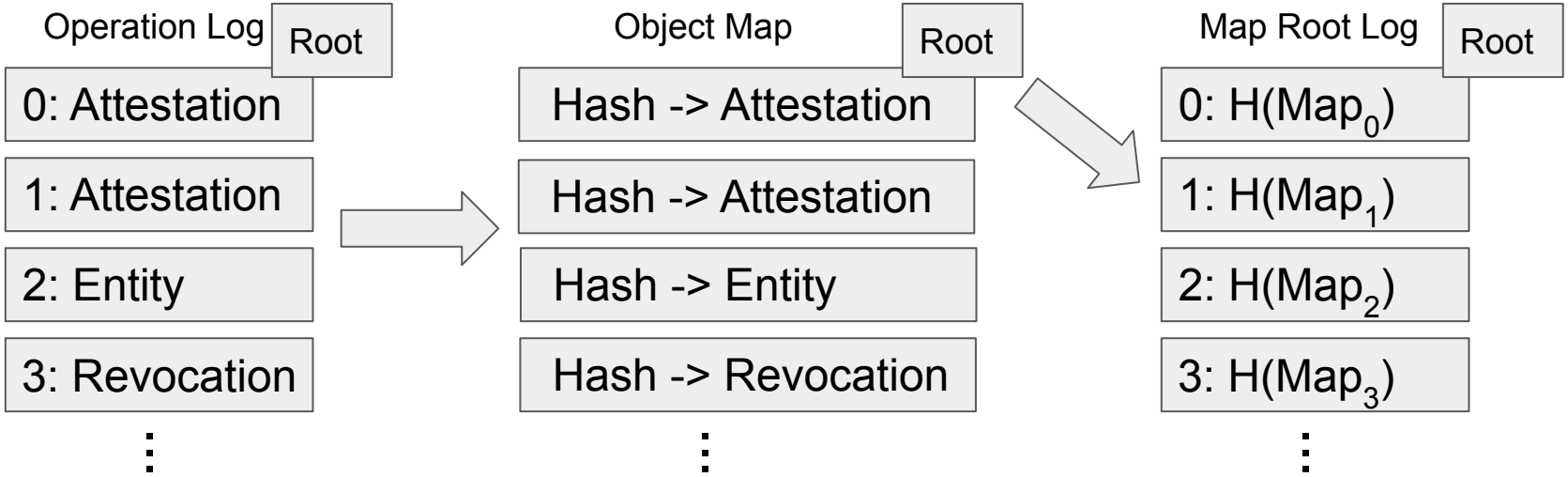
High Level Overview



High Level Overview



Constructed using three Merkle trees

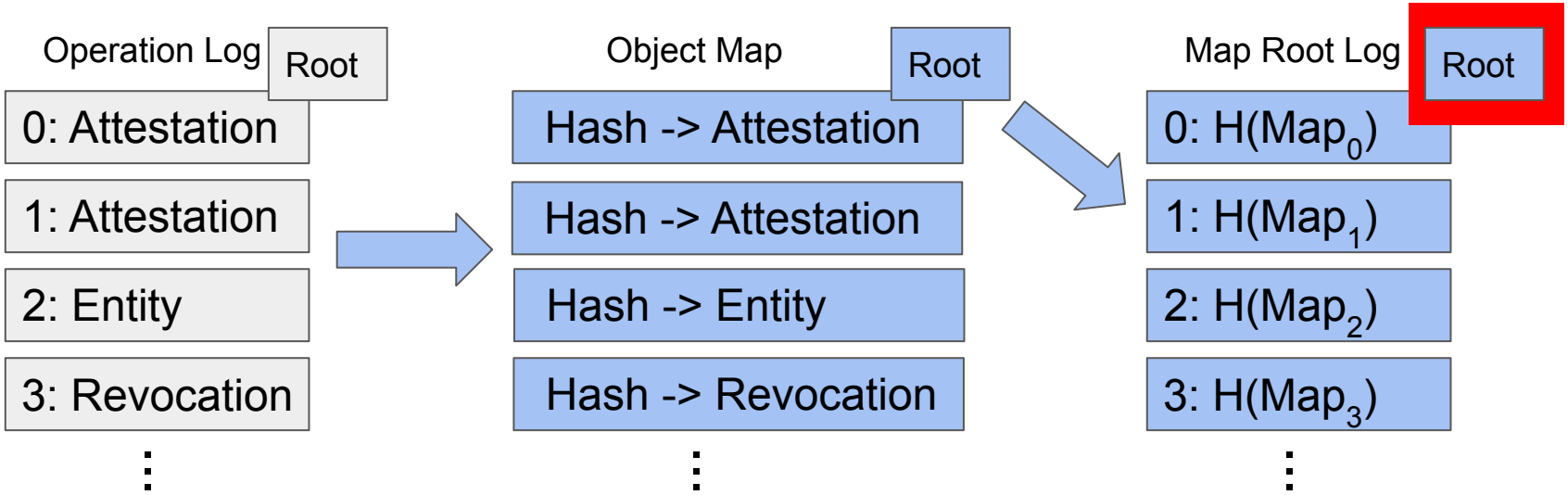


Merkle Tree Log
Can prove:
- Append-only
- Value exists in log

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

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

Auditor replays operation log to construct replica

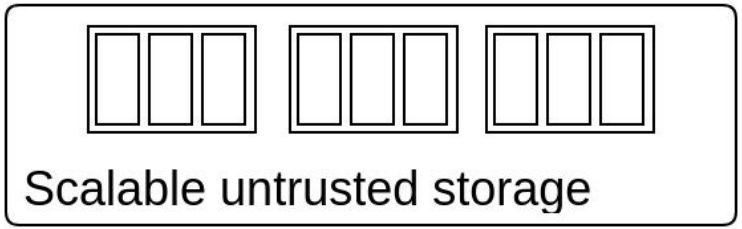


Ensures Object Map is properly derived from operation log

Clients send Root Hash of Map Root Log to auditors periodically (daily)

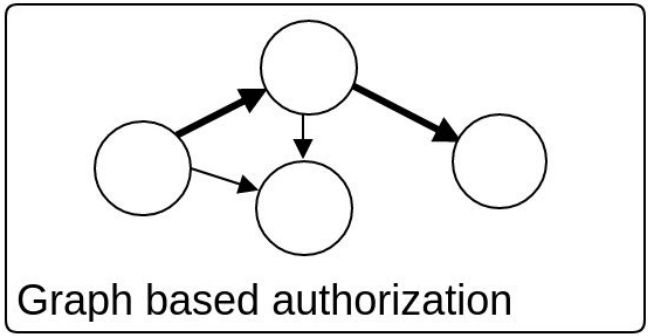
- Ensures every client is seeing the same data structure

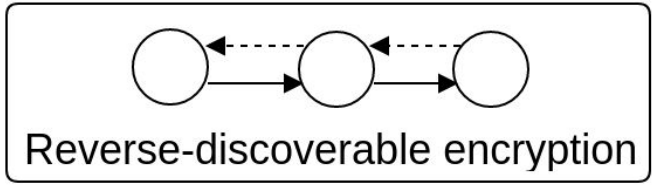
Unequivocable Log Derived Map Summary

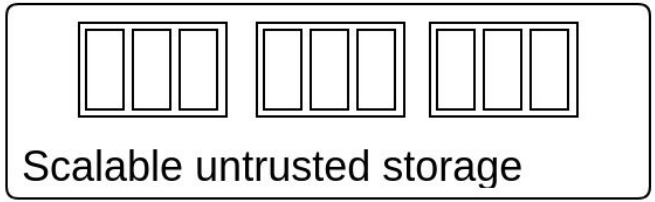


- Stores encrypted attestations, public entity objects, revocations
- Uses cryptographic proofs of integrity
- Forces operators to be honest, or be detected as dishonest
- Auditing requires infrequent communication between clients and auditors

WAVE is fully implemented







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

github.com/immesys/wave

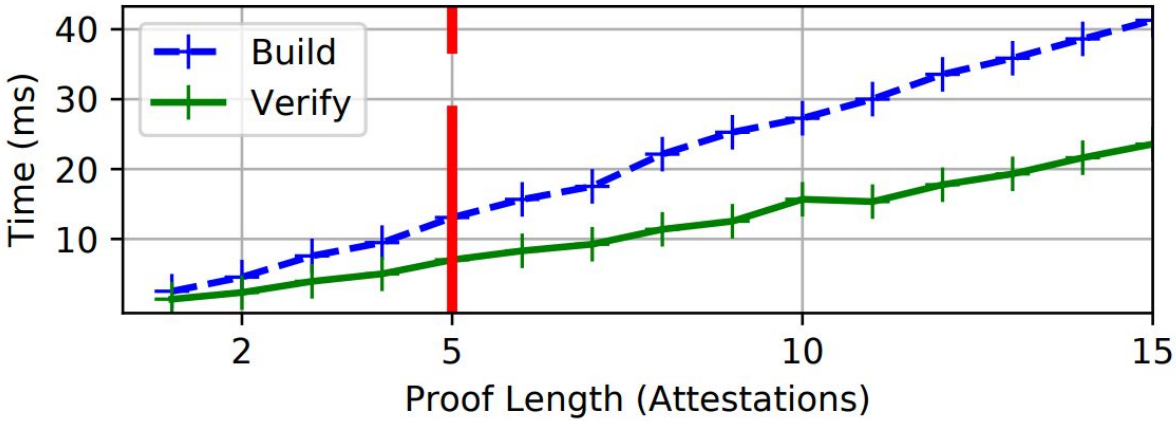
We've used various versions of WAVE over the course of three years:

>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