Capabilities for Distributed Authorization

INTERPEER

@IETF118 // ACEWG

Introduction

Interpeer Project does R&D in “internet technology”;

This talk is about authorization in distributed systems.

Work done under a grant from ISOC foundation.

https://www.isocfoundation.org/

Text

- Datatracker

  https://datatracker.ietf.org/doc/draft-jfinkhaeuser-caps-for-distributed-auth/

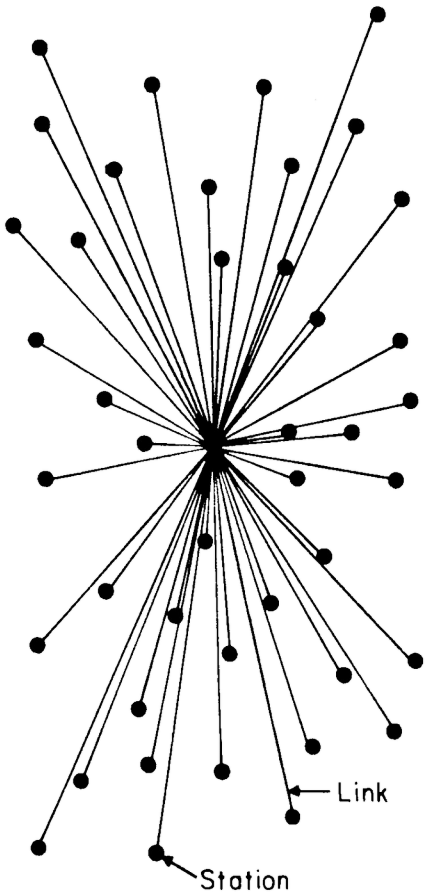
- Latest:

  https://specs.interpeer.io/draft-jfinkhaeuser-caps-for-distributed-auth/
  (includes revisions not yet on datatracker)

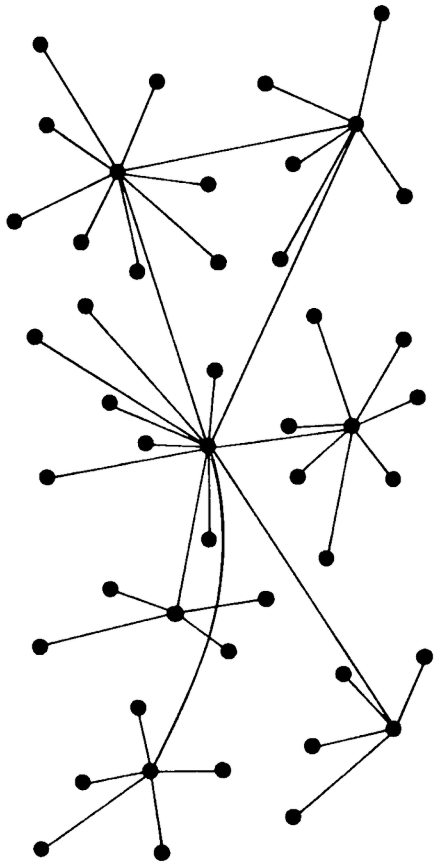
- Repo:

  https://codeberg.org/interpeer/specs/

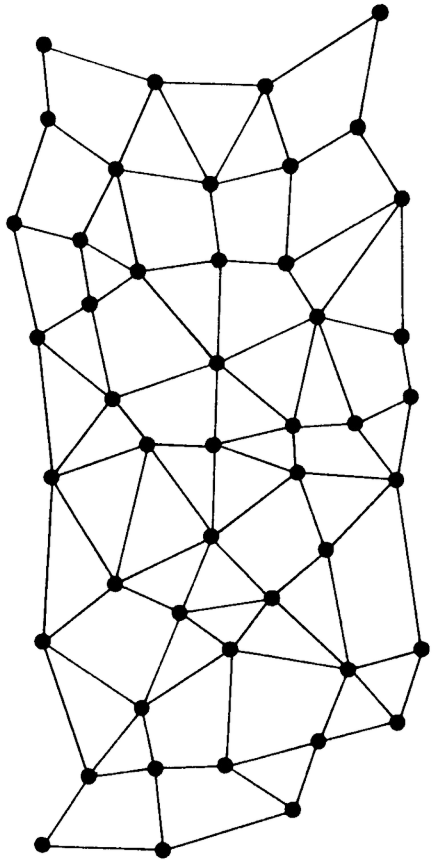
Use Cases



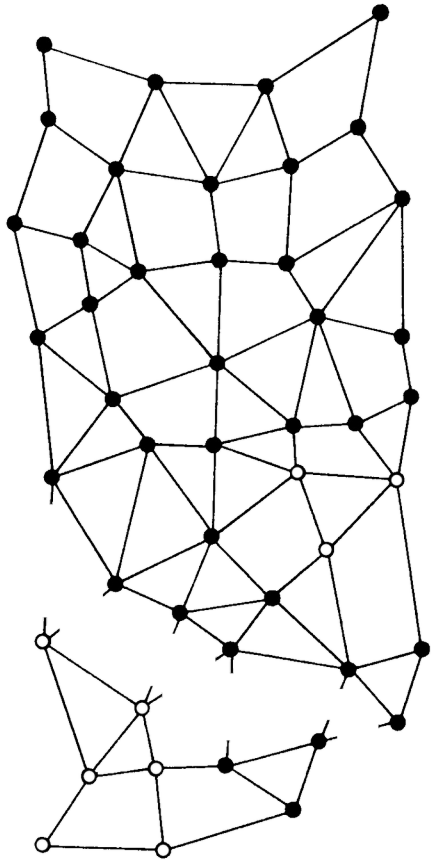
CENTRALIZED
(A)



DECENTRALIZED
(B)



DISTRIBUTED
(C)



DISRUPTED
(D)

Capabilities are not new, so what?

Prior work falls into at least one of the following categories:

- Tied to a specific use case (overly concrete)
- Tied to a specific technology (overly concrete)
- Complex by trying to capture everything
- Complex and abstract

e.g. RFC2693 “SPKI Certificate Theory”, though excellent, is **both** complex and abstract, and overly concrete by being tied to X.509.

Goals & Strategy

- Generic enough for wide applicability.

- Simple enough for implementation.

→ Focus on terminology, basic mechanisms over encoding, etc.

- No (hidden) single point of failure (at use).

- Small enough for 0-RTT authorization.

→ Focus on minimum components over completeness.

Authorization

Distinguish between Auth Management, Query and Access Granting.

Auth Management assigns privileges to identifiers applied to objects (by whichever method; aside on attributes in a few slides).

Auth Query presents an query to access some resource, which is resolved into a boolean accept/deny resolution.

Access Granting grants or denies access to a resource based on the above response.

Capabilities vs. “Traditional”

Auth Management is always in some first phase.

Traditionally, a request to a resource is comprised of Auth Query and Access Granting, yielding either an error or the resource.

Capabilities:

- perform the Auth Query in first phase

- encode the result in a signed bearer token in first phase

- At use (resource request) Access Granting can occur based on valid signature in second phase

Aside: Attributes

Attribute based authorization does not assign privileges to identifiers, and so needs no prior authentication.

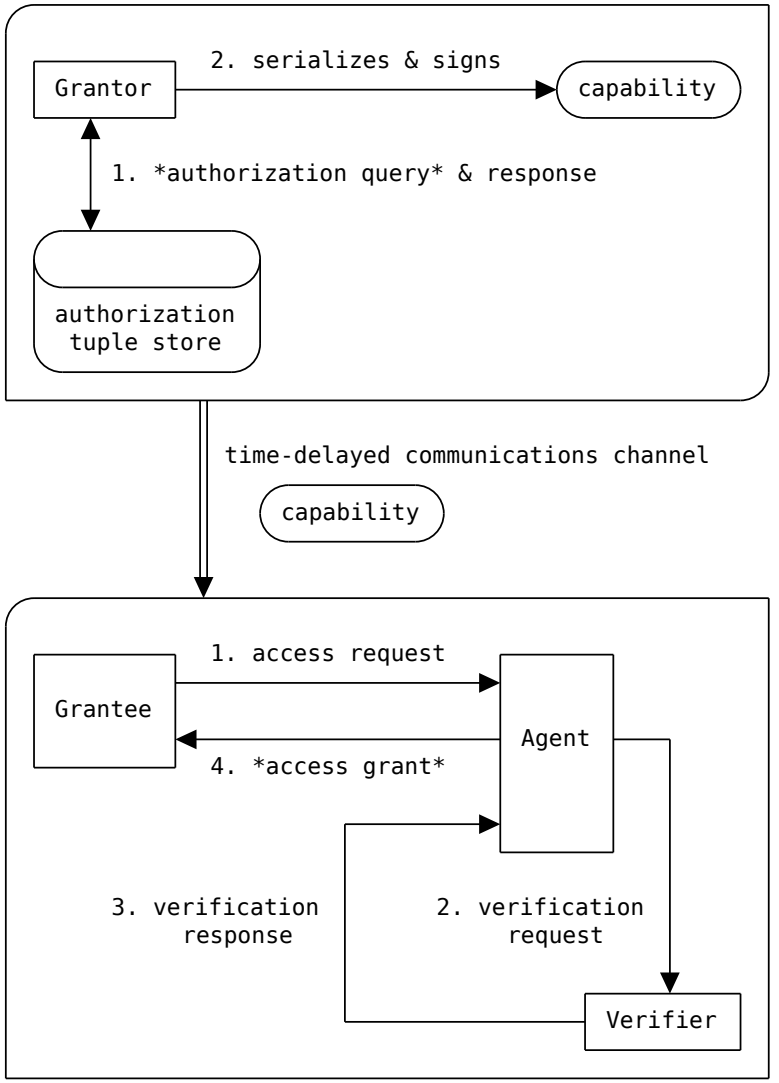
Assigns privileges to (essentially) a set of attributes in first phase.

At use (second phase), an ephemeral identity is constructed based on whether these attributes match the requester: conceptually, the same tuple results.

→ Draft needs to distinguish this and allow for these differences.

Process

1) Some authorization occurs in advance, generates capability.
2) Capability can be stored and transmitted freely and with arbitrary delay.
3) When access is requested, the capability can be consulted to resolve whether access should be granted.



Feedback so far

“Reinvents the wheel” compared to RFC2693: yes, but X.509 not required. Consider the complexity added to DTLS just for transmitting large X.509 certs.

Grantor and issuer, grantee and subject are the same thing: yes and no. Issuer describes a role related to cryptographic operations, grantor has authorization semantics.

What about post-quantum security?: probably means 0-RTT is not so easy, but that’s less of a problem for the abstract scheme.

Derived and Future Work

We have a more specific scheme as well as a compact encoding (<500 Bytes).

- Other draft(s) on specific constraints, encoding, etc.

Future (?):

- JWT encoding
- alignment with RFC2693
- Expression in CoAP
- What about GNAP, SPICE, etc?

Interpeer Project

- Web: https://interpeer.io/
- Code: https://codeberg.org/interpeer/
- Mailing: https://lists.interpeer.io/
- We’re a non-profit: https://interpeer.io/donations/

THANK YOU

INTERPEER