Workgroup: Interpeer Project
Published: 5 December 2022
Authors: J. Finkhaeuser, *Interpeer*; S. D. Penna, *ISEP*

Capabilities for Distributed Authorization

Abstract

Authorization is often the last remaining centralized function in a distributed system. Advances in compute capabilities of miniaturized CPUs make alternative cryptographic approaches feasible that did not find such use when first envisioned. This document describes the elements of such cryptographically backed distributed authorization schemes as a reference for implementations.

The RFC Editor will remove this note

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://specs.interpeer.org/draft-jfinkhaeuser-caps-for-distributed-auth/.

Discussion of this document takes place on the Interpeer mailing list interpeer@lists.interpeer.io, which is archived at https://lists.interpeer.io/pipermail/interpeer/. Subscribe at https://lists.interpeer.io/mailman/listinfo/interpeer.

Source for this draft and an issue tracker can be found at https://codeberg.org/interpeer/specs.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

In 1964, Paul Baran at the RAND Corporation described centralized, decentralized and distributed communications networks and their properties [RM3420]. Baran's argument was that because in distributed systems, each node can reach many other nodes, failure of a single node need not impact the ability of other nodes to communicate.

This resilience is desirable in distributed systems today. Therefore it seems an oversight that authentication and authorization in modern system is often a centralized function.

This document explores previous attempts at distributed authorization schemes, and outlines common elements of such solutions in order to provide a reference for future work.

1.1. Conventions and Definitions

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

In order to respect inclusive language guidelines from [NIST.IR.8366] and [I-D.draft-knodel-terminology-10], this document uses plural pronouns.

1.2. Problem Space

Distributed authorization is not a goal in itself, but may be desirable in distributed systems.

It's also worth exploring how the distribution of authorization functions related to authentication. In many systems, these are intrinsically linked. Logging in with a user name and password is one such example. Providing the correct password proves that the person at the keyboard is authorized to access a resource. But at the same time, providing the correct password in combination with a user name authenticates this user. Furthermore, any permissions granted to the user are typically linked to the user name, as that remains stable throughout password changes.

1.2.1. Authentication

Password-based authentication mechanisms require that the tuple of user name and password (or password hash) are sent to some central repository where records of such tuples are kept; if the tuple is found, the user name is authenticated.

This common scheme mixes different aspects to authentication, however, which are worth disambiguating.

Endowment: The act of logging in establishes an association between a user name and the person interacting with the device. More broadly speaking, (parts of) a three-way endowment are performed: an *identifier* is endowed with *attributes*, which describe a *person* in sufficient detail to identify them.

Secret Proving: Logging in also proves that the person interacting with the device is in possession of some secret, which should only be known to the person which merits having the user name linked to the secret associated with them.

The distinction becomes somewhat more relevant when we move towards distributed authentication schemes, which rely on public key cryptography.

1.2.1.1. Web of Trust

In Web of Trust based systems, starting with Philip R. Zimmermann's Pretty Good Privacy (PGP), public keys are exchanged with some metadata attached. This performs some part of endowment in that it provides the link between a public key and a user identifier (see [RFC4880], Section 11.1).

Other parts of endowment are not specified. These often consist of manual checks that the user identifier belongs to some person holding the corresponding private key, and may involve verifying of government issued identification documents. Once such a check is passed, the verifier issues a digital signature over the tuple of user identifier and public key to provide some proof that the verification has occurred.

Endowment in Web of Trust occurs when a sufficient number of sufficiently trustworthy signatures have been reached. Which number of trust level is deemed sufficient is in the control of the recipient of a transferable public key packets, however.

1.2.1.2. TLS Certificates

A similar concept is applied in TLS [RFC8446], where [X.509] certificates are used for endowment.

The major difference to Web of Trust based systems is how trust is established. Instead of relying on a recipient defined method of determining trust, certificates are issued by one of a set of well-known trust sources. Information on these is stored in root certificates, which are distributed to the machines participating in the system.

While there are globally issued root certificates for entities that perform endowment professionally, it is always possible for a system designer to include other root certificates.

1.2.1.3. Secret Proving

Neither [X.509] certificates nor the transferable public key packets in [RFC4880] provide any means for secret proving. This is left to other parts of TLS or PGP.

In TLS, the handshake during connection establishment is used to send challenges that only machines with the correct private key can respond to. PGP, which aims to provide privacy at rest, simply encrypts content with a secret key which is then encrypted with the recipient's public key. Other parties cannot decrypt this, which keeps content safe.

TLS and PGP are not the only public key cryptography based authentication systems, but they can stand in for the two most common classes of such systems: one aims to establish trust from authoritative sources. The other aims to establish trust based on the trust requirements of the recipient.

Both systems also strictly speaking separate endowment from secret proving. While in TLS the certificates are transmitted as part of the overall handshake, creating certificates nevertheless occurs beforehand. This temporal decoupling is a key property that may also be applied to authorization.

1.2.2. Authorization

Authorization occurs only after secret proving. Once an identity has been established, it is then mapped to privileges associated with said entity, which determine which object(s) it has access to.

There exist a plethora of methods to establish this mapping. Access-control lists (ACL) simply provide tuples of identities, privileges and associated objects. Role-based access control (RBAC) is effectively identical, if the identities specified are not those of individuals, but of groups (as a group member, an individual inhabits the associated role). A comparable approach is Organization-based access control (OrBAC), which not only abstracts the identity to that of a role, but performs a similar abstraction on the object and privilege.

More complex systems such as context- or lattice-based access control (CBAC and LBAC respectively) derive a mapping from properties of or labels attached to the individuals and objects. Finally, graph-based access control (GBAC) starts with a graph of an organization, and derives privileges from the properties inherited by being part of a larger organizational structure.

What these systems address is the problem of *managing* the mapping of an identity to access privileges for objects, where each system has advantages and disadvantages for various use cases.

Subject: The subject is the identity (individual or group/role) that intends to perform an action.

Action: The action the subject intends to perform may be as simple as reading or writing a resource, but can be more complex.

Object: Actions are performed on objects , such as a file or network resource.

Request Tuple: A request tuple consists of the subject, action and (optional) object.

Privilege: A privilege encodes whether or not an action is permitted.

Authorization Tuple: An authorization tuple encodes system state, and is thus a tuple of a subject, a privilege and an (optional) object.

The act of authorization translates from a request tuple to a Boolean response determining whether a request is permitted. A centralized authorization function provides this answer in real-time. Distributed authorization instead deals in authorization tuples.

It may be of interest that and authorization tuple is semantically equivalent to an RDF triple ([RDF]), in that it encodes a specific relationship between a subject and an object. Authorization tuples that consists solely of IRIs [RFC3987] is also syntactically an RDF triple. This implies that authorization tuples can encode arbitrarily complex authorization information by building the knowledge graph resulting from resolving such an RDF triple.

1.2.2.1. Single Point of Failure

A centralized function is very useful for managing authorization. The previous section on different access control methods should illustrate sufficiently that authorization management is a complex problem; complex enough for multiple competing management methods to emerge.

Faced with such a complex problem, it is no surprise that solutions tend to bring this function to a centralized location. Managing this complexity in one place is of course simpler than managing it across multiple locations.

The downside to this is that failure of this single location may mean failure of the system as a whole. Particularly vulnerable to this single point of failure are systems in which all access is controlled by specific privileges. Systems with publicly available parts may still provide those functions that do not rely on any privileges.

1.2.2.2. Temporal Coupling

The other class of problems with centralized authorization relate to the temporal coupling of granting access and resolving authorization queries . The abstract request introduced above of resolving an request tuple to a Boolean response tightly couples both steps.

This requires disambiguating between participants in such a system somewhat. From the perspective of the person operating the access control management system, granting access occurs whenever they make an entry into the database. The machine permitting an authorized user to perform an action, however, grants or denies access in the moment the action is requested. If this second form of access granting is based on a real-time authorization query, it couples granting access to such a query in time.

The key insight into distributing authorization effectively is that it has little to do with managing access control databases. Rather, it is related to temporally decoupling the authorization query from granting access.

1.3. Previous Work

Dividing the authentication problem into endowment and secret proving helps illustrate how web of trust systems introduce temporal decoupling between these functions, in a way that e.g. TLS does not.

In much the same way, dividing the authorization problem into querying an authorization database and granting access to an object suggests that authorization, too, can be temporally decoupled.

This section lists prior work where some temporal decoupling of this kind has been performed in the past.

1.3.1. Object-Capabilities (OCAP)

Dennis and Van Horn described an approach for securing computations in "multiprogrammed" systems in 1965/66 ([OCAP]). The context in which they operated had little to do with modern distributed systems.

However, they recognized the trend of running computing systems complex enough that multiple programmers would contribute to its overall function. This raised a desire for access control to individual sub-functions, which a security kernel within the operating system was to provide.

The key differentiator to other systems was that in OCAP, a calling process was to present a "capability", a serialized token to the process being invoked. This capability was intended to encode all relevant information the called process would need to determine whether the caller was permitted to perform such an action.

These properties of being serializable and containing all relevant authorization information imply that, conceptually, capabilities are cached results of an authorization query . The called process can then perform access granting without issuing such a query itself, thereby temporally decoupling the two functions.

1.3.2. Identity-Capabilities (ICAP)

The OCAP system proved to have a particular weakness, namely that "the right to exercise access carries with it the right to grant access". This is the result the information encoded in an OCAP capability: it contains a reference to the object and action to perform, but does not tie this to any identity.

In 1988, Li Gong sought to address this with an Identity-Capability model ([ICAP]). Including an identity in the capability token arrives at the authorization tuple in Section 1.2.2. Furthermore, ICAP introduces the notion of capability use in networked systems. ICAP does this by temporally decoupling the authorization query from access granting.

The main criticism levelled against the paper and capability-based approaches in general in the following years was that some functions were missing, such as a check for revocations. Proposals to address this often added centralized functions again, which led to criticism of the distributed approach in general.

1.3.3. Pretty Good Privacy

While we previously discussed PGP in terms of authentication in Section 1.2.1.1, a key property of PGP is the introduction of trust signatures ([RFC4880], Section 5.2.3.13).

Trust signatures do not merely authenticate a user, they introduce a kind of authorization as well, as they carry specific notions for what the provided public key may be trusted for. The trust signature thus encodes specific kinds of privileges of an authorization tuple , while the public key encodes a subject . The only component missing in the tuple is the object .

While the authorization tuple in PGP is incomplete, the system is based on public key cryptography, and can thus be used to securely verify a binding between the tuple elements.

1.3.4. JSON Web Tokens (JWT)

JSON Web Tokens ([RFC7519]) provide a standardized way for serializing access tokens. Current uses are in systems with centralized authorization functions such as OAuth ([RFC6749]).

However, the fundamental notion of capabilities, that a serializable token carries authorization information, is provided also here. Furthermore, JWT combines this with cryptographic signatures, providing for - in theory - temporal decoupling as previously discussed.

It's well worth pointing out that JWT is suitable as a portable, modern capability format - all it requires is to encode all necessary information within its fields.

1.3.5. Power of Attorney

The oldest kind of prior work in this field is the concept of Power of Attorney, as exercised throughout much of human history.

In a Power of Attorney system, an authority (a king, etc.) grants a token (an official seal, ...) to a subordinate which makes this subordinate recognizable as carrying some of the king's powers and privileges.

Modern day Power of Attorney systems abound, and may be formalized as notarized letters granting such and such rights to other people.

Capability-based authorization schemes are no different to this kind of system in principle. In both kinds of systems, the token itself encodes the privileges of the bearer.

1.4. Use Cases

Use cases relate to one or more of the issues explored in the problem space.

1.4.1. IoT On-boarding

On-boarding IoT devices into an overall system requires authentication and authorization; this may be mutual.

In such scenarios, new devices rarely have connectivity before completing on-boarding. It follows that authentication and authorization must work in a fully offline fashion, which in turn requires that authorization tokens provided to the device contain all information required for the authorization step. As described in Section 1.3.1, this translates to a requirement of temporally decoupling access granting from an authorization query.

1.4.2. UAV Control Handover

A similar argument applies to control handover of unmanned aerial vehicles (UAV). The concept of Beyond Visual Line of Sight (BVLOS) missions is to send drones into places that are harder or more costly to reach for humans.

Control handover refers to transferring operational control for a drone from one ground control station to (GCS) another. Control handover bears similarities to IoT on-boarding in that the drone is on-boarded to a new control system (and the previous system relinquishes control).

In general, aviation authorities such as FAA, EASA, etc. expect control handover to occur under ideal circumstances, in which centralized authorization schemes suffice. There is, however, a class of scenarios where connectivity to a central service cannot be guaranteed.

1.4.2.1. Remote Location

In order to guarantee BVLOS operations in very remote locations, research projects such as [ADACORSA] assume use cases in which two ground control stations between which handover occurs to not have connectivity to each other.

In such cases, it is necessary for the UAV to act as a time-delayed transmission channel for authorization information between the GCSes.

1.4.2.2. Emergency Response

Emergency response teams may require UAVs in the vicinity to immediately clear the airspace and go to ground. This effectively translates to the emergency response team operating a ground control station that takes over control and issues a single command.

As emergency responses are, by definition, typically required in situations where normal operations cannot be assumed, this includes the assumption that connectivity cannot be assumed. Nevertheless, such an emergency control handover must be possible.

1.4.2.3. Mobile Ground Control Stations

A comparable scenario to the above describes situations in which UAV attach to a mobile ground control station. Specific scenarios may range from cave exploration to investigating burning buildings.

The commonality here is that the UAV cannot establish connectivity to a wider system, but can connect to the mobile GCS. This in turn may act as a communications relay to the outside world, but may be too limited in capacity to permit online, centralized authorization.

2. Elements of a Distributed Authorization Scheme

As explored in the previous sections, the most fundamental aspect of a distributed authorization scheme is that it decouples access granting from authorization queries by serializing the results in such a way that they can be transmitted and evaluated at a later date. This effectively shifts the focus of distributed authorization systems away from request tuples towards authorization tuples.

This implies certain things about the contents of a capability token, but it also introduces other elements and roles into the overall scheme.

2.1. Grantor

A grantor, sometimes called principal, has authority over an object, and generates authorization tuples for use in the overall system.

As we describe cryptographic systems, a grantor is represented by an asymmetric key pair. Endowment for a grantor is out of scope of this document; for the purposes of distributed authorization, the grantor key pair *is* the grantor.

2.1.1. Grantor Identifier

A grantor identifier uniquely identifiers the public key of the key pair; this may be identical to a serialized form of the public key itself, or a cryptographic hash over it (fingerprint), or some alternative scheme.

What is sufficient is that there **MUST** exit a mechanism for uniquely mapping the grantor public key to the grantor identifier and vice versa. This mapping permits verification.

2.1.2. Grantor Signature

The grantor undersigns a capability by adding a cryptographic signature to it.

2.2. Agent

The agent is the element in a distributed system that executes a requested action after verifying a capability. It typically manages objects itself, or provides access to them.

2.3. Verifier

The verifier is a role in the system that verifies a capability. While verifiers can exist in a variety of system nodes, it's a mandatory part of the agent role.

Outside of the agent, verifiers may exist in intermediary nodes that mediate access to agents. An example here might be an authorization proxy that sits between the public internet and a closed system. While it may not be an agent in and of itself, it can still decide to reject invalid requests, and only forward those to agents that pass verification and its own forwarding rules.

2.4. Time-Delayed Transmission Channel

We introduce the concept of a time-delayed transmission channel to illustrate that communications between grantor and verifier is not possible in real-time.

In practice, of course the communications channel does not have to be time- delayed. But treating it as such implies that granting access must be temporally decoupled from the authorization query.

2.5. Grantee

The grantee is the entity to which a privilege is granted.

A grantee **SHOULD** also be represented by an asymmetric key pair in order to perform distributed authentication.

2.5.1. Grantee Identifier

A grantee identifier is the identifier used as the subject in an authorization tuple.

If the grantee is equivalent to an asymmetric key pair, it **MUST** also be possible to map the grantee identifier to the grantee public key and vice versa. Such a mapping **SHOULD** be feasible to perform without connectivity in order to maintain the distributed authentication mechanisms achieved through the use of asymmetric cryptography.

2.6. Object

An object is a resource the grantee wishes to access. This can be a file, or a networked service, etc.

2.6.1. Object Identifier

The object identifier uniquely identifiers an object. This document places no syntactic restrictions upon the object identifier, other than that there exists a canonical encoding for it. For the purposes of cryptographic signing and verification, the object identifier **MUST** be treated as equivalent to its canonical encoding.

2.7. Privilege

A privilege encodes whether an action (on an object) is permitted (for a subject); see {#sec:authorization} for an explanation.

For the purposes of creating capabilities, a privilege must have a canonical encoding. The semantics of each privilege are out of scope of this document, and to be defined by the systems using distributed authorization.

That being said, a typical set of privileges might include read and write privileges for file-like resources.

2.8. Validity Metadata

In practical applications of distributed authorization scheme, validity of a capability may be further scoped. We already discussed the need to scope it to an authorization tuple, but further restrictions are likely desirable.

For example, a set of `not-before` and `not-after` timestamps exist in e.g. [X.509] certificates; similar temporal validity restrictions are likely required in practical systems.

However necessary they may be in practice, however, such additional validity metadata has no bearing on the fundamental concepts outlined in this document, and is therefore considered out of scope here.

2.9. Capability

A capability provides a serialized encoding of previously listed elements:

1. Fundamentally, a capability **MUST** encode an authorization tuple, consisting of:
    1. A subject identifier.
    2. A privilege.
    3. An object identifier.
2. A grantor identifier **MAY** be required in order to identify the grantor key pair used in signing and verification.
3. Validity Metadata **SHOULD** be included in practical systems.
4. In order for a verifier to ensure the validity of a capability, it **MUST** finally contain a grantor signature over all preceding fields.

The authorization tuple permits an agent to determine what kind of access to grant or deny. The grantor identifier provides information to the verifier about key pairs used in the authorization. While the signature proves to the verifier that the grantor did indeed authorize access, the validity metadata limits access to whichever additional scope the grantor decided upon.

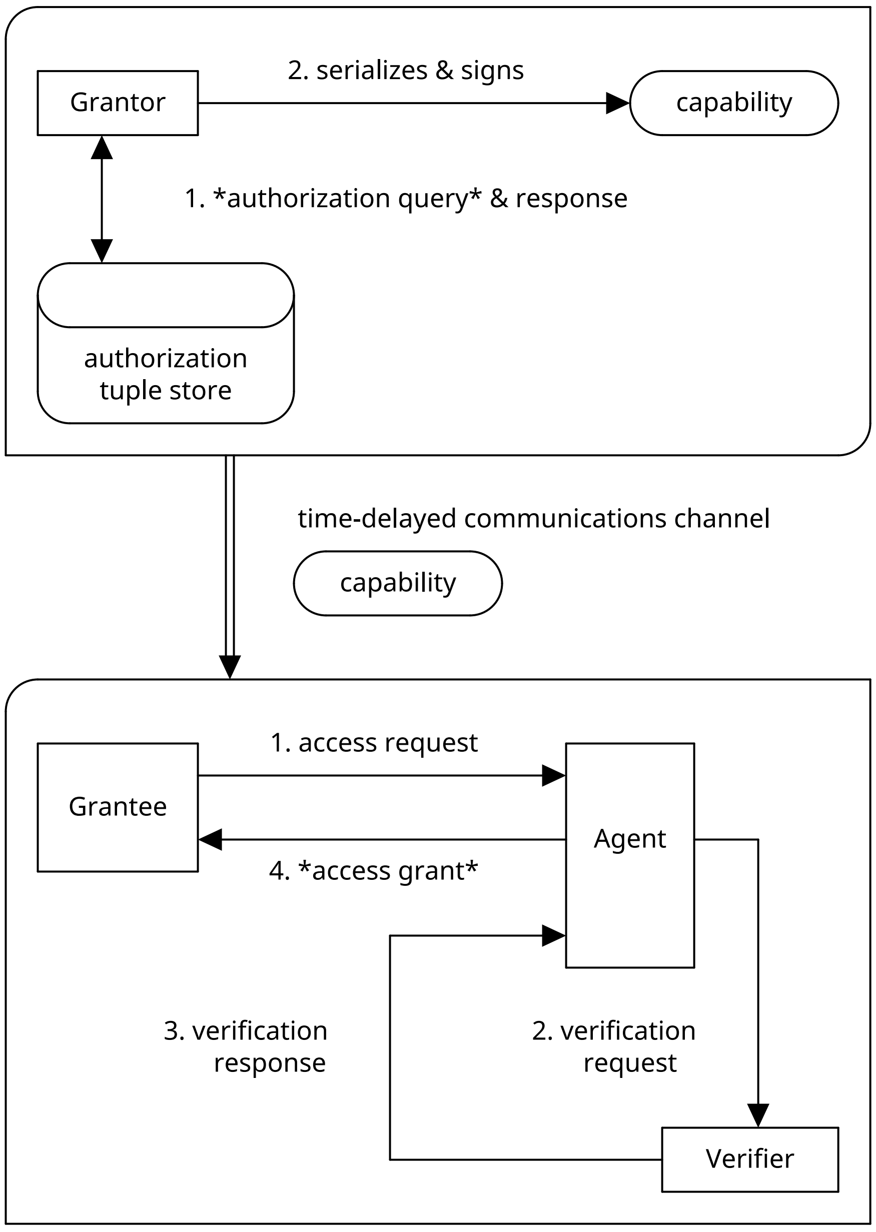
2.9.1. Extensions

Note that each of the fields in an authorization tuple may be treated as a list of zero or more such elements. While a longer discussion of this is out of scope for this document, two notes should be made:

1. Implementations must provide clarity what it means to provide a list. Does the capability apply to each element in the list individually, or to some combination? This is highly specific to the semantics of each capability, so cannot be covered here.
2. A tuple consisting of a subject and privilege only (zero objects) effectively turns into a statement about the subject, and no longer relates to authorization concerns. However, other aspects of a distributed trust system still apply. This is the approach taken by Pretty Good Privacy.

2.10. Authorization Process

Having identified the elements, we can now describe an abstract process in a distributed authorization system.



The process is split into two phases.

In the first phase, the grantor issues an authorization query (((authorization query))) to an authorization tuple store, which stands in here for the specific process by which authorization is managed, and produces tuples. Based on the response, it serializes a capability and adds its signature over it.

The capability then gets transmitted via the time-delayed communications channel to the second phase, providing temporal decoupling between the phases.

In the second phase, the grantee requests access to some object from the agent. The agent must send a verification request to the verifier (which may be a subroutine of the agent; no network transmission is implied here). The verifier responds by either permitting access or not. If access is permitted, the agent grants access to the grantee. Because the capability encodes all required information for the verifier to perform this step, it does not need access to the authorization tuple store itself.

Note that the capability can be transmitted to any entity in the second phase; all that is relevant is that it ends up at the verifier. If it is transmitted to the grantee, it has to pass it on to the agent as part of the access request. If the agent receives it, it has to pass it on to the verifier as part of the verification request.

3. Delegation of Authority

One of the more powerful applications of the power of attorney system is that it is possible to further delegate authority. The constraint is that no entity can provide more authority in a sub-grant than it possessed in the first place.

The ability to generate sub-grants is easily provided in a specialized privilege. Such a privilege must encode the specific other privileges a grantee may in turn grant to other parties.

As this may include the ability to grant further sub-grants, implementations **MUST** take care here. They **MAY** wish to include a limit on the depth to which sub-grants may be further delegated.

4. Related Considerations

4.1. Human Rights Considerations

What follows is a list of objectives derived from [RFC8280], each with a brief statement how this document addresses each concern, or why it does not.

4.1.1. In Scope

Connectivity: Distributed authorization observes end-to-end principle, by temporally decoupling distinct functions in the authorization process, so that connectivity can be almost arbitrarily unreliable.

Reliability: Capabilities provide reliability by removing the need for real-time reliability.

Content agnosticism: The elements of authorization tuples are only defined in the abstract; as noted in {#sec:authorization}, they can encode arbitrarily complex relationships. Of course, the semantics **SHOULD** be somewhat focused on authorization.

Integrity: The use of cryptographic signatures in capabilities also provides integrity checking, but no explicit mechanism for distinguishing between integrity failures and authenticity is provided.

Authenticity: Proving authenticity is a core concept of cryptographic capabilities.

Pseudonymity: This document deliberately distinguishes between endowment and secret proving in {#sec:authentication}, precisely because authorization does not need to make use of such information.

Censorship resistance: Temporally decoupling authorization queries from access granting provides one element in a censorship resistant system, as it permits for capabilities to travel out-of-band by means that circumvent potential censorship, such as e.g. via a sneakernet, etc.

Outcome Transparency: This document's main focus is on illustrating the outcomes of distributed authorization schemes.

Adaptability: As this document provides a generalized view on distributed authorization schemes and capabilities, it is highly adaptable in specific implementations.

Decentralization: Decentralization - rather, distribution - is the goal of this document.

Open Standards: Care has been taken to define this specification in reference to other open standards (see the references section).

Security: As this document describes an authorization scheme, security is of great concern. However, guidelines in [BCP72] do not fully apply to an abstract description. See {#sec:security-considerations} for more detail.

4.1.2. Out of Scope

Confidentiality: Confidentiality is out of scope of the general concept of capabilities; capability content is almost by definition public data. However, nothing prevents an implementation from providing confidentiality for the capability elements outside of the signature.

Heterogeneity Support: There is no specific heterogeneity support in distributed authorization schemes.

Remedy: Remedy (new in [I-D.draft-irtf-hrpc-guidelines-16] often stands in stark contrast to anonymity and pseudonymity, both of which are out of scope. Implementations must take care to balance these concerns.

Internationalization: This document focuses on abstract concepts; internationalization is not in scope.

Localization See internationalization; this is not in scope.

Privacy: For privacy protections, implementations must themselves ensure that any information encoded in a capability cannot be abused to resolve otherwise private information.

Anonymity: Anonymity is out of scope for this document. While pseudonymous use is feasible if sufficient care is taken, authorization always ties to an identity, however transient.

Accessibility: Accessibility concerns are out of scope.

4.2. Protocol Considerations

There are no specific protocol considerations for this document.

However, protocols transmitting capabilities **MAY** provide some relief to human rights concerns {#sec:human-rights-considerations}, e.g. by providing confidentiality via encrypted transmission.

4.3. Security Considerations

This document does not specify a network protocol. In fact, it deliberately requires no specific protocol for transmitting capabilities. As such, much of [BCP72] does not apply.

However, distributed authorization does not require the invention of new cryptographic constructs; the document is deliberately phrased such that the choice of such constructs remains implementation defined.

4.3.1. Confidentiality

Confidentiality concerns are out of scope of this document.

4.3.2. Data Integrity

Data integrity is provided as a side effect of requiring a cryptographic signature over capability payloads.

4.3.3. Peer Entity Authentication

Peer entity authentication is not, strictly speaking, a concern of this document. However, it should be noted that an agent in the process **MUST** obviously authenticate a grantee before proceeding to verify capabilities.

As noted previously, distributed authorization does not require endowment in this process, however.

4.3.4. Non-Repudiation

Non-repudiation is strictly speaking out of scope of this document because key exchange is out of scope. However, key exchange is a necessary precondition for verifying capabilities. Non-repudiation is one of the guarantees that verification provides.

4.3.5. Unauthorized Usage

Preventing unauthorized usage is the core concern of this document.

4.3.6. Inappropriate Usage

Any scheme that prevents unauthorized use may also be extended to prevent inappropriate use. However, such additions are out of scope.

4.3.7. Denial of Service

Denial of service mitigation is out of scope, because this document does not describe a protocol.

However, as avoiding a single point of failure (Section 1.2.2.1) is one of the problems that distributed authorization schemes address, it can easily be argued that preventing denial of service is a major concern of this document, and consequently fully addressed here.

4.3.8. Replay Attacks

Capabilities can be "replayed" as much as an attacker wants. As they encode a state, such state does not change when it is received multiple times.

However, implementations **MUST** take care not to invite replay attacks when designing their specific validity metadata (Section 2.8). Furthermore, implementations **MUST** include such metadata in the signature.

4.3.9. Message Insertion

Message insertion is of no concern to this document.

4.3.10. Message Deletion

Message deletion is of little concern to this document, except insofar as if the transmission of capabilities is disrupted, access granting cannot proceed. Implementations **MUST** take care to provide safeguards against this as their threat model requires.

4.3.11. Message Modification

Message modification is prevented by the cryptographic signatures in capabilities.

4.3.12. Man-In-The-Middle

As this document does not provide a protocol specification, this consideration does not apply.

4.3.13. Key Usage

This document relies on secure key exchange.

4.3.14. Revocation

As ICAP was criticized for introducing a centralized solution for revocatins, (see Section 1.3.2), a modern distributed authorization system must adequately consider these.

Fortunately, anything that can encode the granting of a privilege can also encode the removal of said grant, by - essentially - encoding a negative privilege. Doing so provides distributed revocations by the same overall mechanism that distributed authorization is provided. A sequence of grants and revocations for a particular request tuple will map to a sequence of Boolean values, and can so be understood as a bit stream.

This introduces a new requirement, namely that verifiers can reconstruct the bit stream in order to understand the latest, most up-to-date state. Unfortunately, this can be hard due to the time-delayed nature of the communications channel.

Fortunately, research into conflict-free replicated data types has yielded several methods for ordering also partially received streams, which can be applied here by providing appropriate validity metadata. This yields eventually consistent states in a distributed authorization system, which in many cases may be sufficient.

It is not the purpose of this document to prescribe any particular method for ordering grants and revocations into a consistent stream, nor whether revocations are used at all. However, implemtations **MUST** take care to consider this aspect.

4.4. Privacy Considerations

This section lists privacy considerations as covered by [RFC6973] and distributed authorization's relationship to them.

4.4.1. Surveillance

The surveillance concerns outlined in [RFC6973] specifically relate to network protocols; this document does not describe such a protocol.

As noted previously, pseudonymous use is well supported by this kind of scheme, while fully anonymous use is not. If appropriate confidentiality is provided by implementations, distributed authorization schemes can be considered fairly resistant to surveillance.

That being said, systems **SHOULD** still use transport encryption in order to further mitigate against surveillance.

4.4.2. Stored Data Compromise

Stored data concerns are largely identical to surveillance concerns in the context of distributed authorization.

4.4.3. Intrusion

Authorization schemes in general aim to protect against intrusion.

4.4.4. Misattribution

Misattribution in [RFC6973] refers to misattribution to individuals, which relates to endowment. This document considers endowment out of scope.

4.4.5. Correlation

Distributed authorization tokens cannot protect against correlation, unless confidentiality concerns are addressed. This is, however, an implementation concern.

4.4.6. Identification

Similar arguments can be made for identification concerns. Distributed authorization is fundamentally concerned with identifying elements in a system, but if confidentiality is provided, identification is not possible.

4.4.7. Secondary Use

Secondary use concerns are not in scope of this document.

4.4.8. Disclosure

Disclosure concerns are effectively identical to confidentiality concerns in that capabilities may leak identifiers, which may be pseudonymous.

4.4.9. Exclusion

Exclusion is a network protocol consideration, and does not apply here.

4.5. IANA Considerations

This document has no IANA actions.

5. References

5.1. Normative References

[BCP72] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/rfc/rfc3552>.

[NIST.IR.8366]
Miller, K., Alderman, D., Carnahan, L., Chen, L., Foti, J., Goldstein, B., Hogan, M., Marshall, J., Reczek, K., Rioux, N., Theofanos, M., and D. Wollman, "Guidance for NIST staff on using inclusive language in documentary standards", National Institute of Standards and Technology (U.S.), DOI 10.6028/nist.ir.8366, April 2021, <https://doi.org/10.6028/nist.ir.8366>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/rfc/rfc6973>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8280] ten Oever, N. and C. Cath, "Research into Human Rights Protocol Considerations", RFC 8280, DOI 10.17487/RFC8280, October 2017, <https://www.rfc-editor.org/rfc/rfc8280>.

[RM3420] Baran, P., "On Distributed Communications: I. Introduction to Distributed Communications Networks", RAND Corporation, DOI 10.7249/rm3420, 1964, <https://doi.org/10.7249/rm3420>.

5.2. Informative References

[ADACORSA] "Airborne data collection on resilient system architectures", 1 May 2020, <https://www.kdt-ju.europa.eu/projects/adacorsa>.

[I-D.draft-irtf-hrpc-guidelines-16] Grover, G. and N. ten Oever, "Guidelines for Human Rights Protocol and Architecture Considerations", Work in Progress, Internet-Draft, draft-irtf-hrpc-guidelines-16, 10 November 2022, <https://datatracker.ietf.org/doc/html/draft-irtf-hrpc-guidelines-16>.

[I-D.draft-knodel-terminology-10] Knodel, M. and N. ten Oever, "Terminology, Power, and Inclusive Language in Internet-Drafts and RFCs", Work in Progress, Internet-Draft, draft-knodel-terminology-10, 11 July 2022, <https://datatracker.ietf.org/doc/html/draft-knodel-terminology-10>.

[ICAP] Gong, L., "A secure identity-based capability system", IEEE Comput. Soc. Press, Proceedings. 1989 IEEE Symposium on Security and Privacy pp. 56-63, DOI 10.1109/secpri.1989.36277, January 2003, <https://doi.org/10.1109/secpri.1989.36277>.

[ISOC-FOUNDATION] Internet Society Foundation, "Internet Society Foundation", n.d., <https://www.isocfoundation.org/>.

[OCAP] Dennis, J. and E. Van Horn, "Programming semantics for multiprogrammed computations", Association for Computing Machinery (ACM), Communications of the ACM vol. 9, no. 3, pp. 143-155, DOI 10.1145/365230.365252, March 1966, <https://doi.org/10.1145/365230.365252>.

[RDF] RDF Working Group of the World Wide Web Consortium (W3C), "RDF 1.1 Concepts and Abstract Syntax", 25 February 2014, <https://www.w3.org/TR/rdf11-concepts/>.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, DOI 10.17487/RFC3987, January 2005, <https://www.rfc-editor.org/rfc/rfc3987>.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <https://www.rfc-editor.org/rfc/rfc4880>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/rfc/rfc6749>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/rfc/rfc7519>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[X.509] International Telecommunications Union, "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks", ITU-T Recommendation X.509, ISO Standard 9594-8, March 2000.

Acknowledgments

Jens Finkhäuser's authorship of this document was performed as part of work undertaken under a grant agreement with the Internet Society Foundation [ISOC-FOUNDATION].

Index

Authors' Addresses

Jens Finkhäuser
Interpeer gUG (haftungsbeschraenkt)
Email: ietf@interpeer.io
URI: https://interpeer.io/

Sérgio Duarte Penna
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431
4249-015 Porto
Portugal
Email: sdp@isep.ipp.pt
URI: https://isep.ipp.pt/