Workgroup:	Interpeer Project
Internet-Draft:	draft-jfinkhaeuser-caps-for-distributed-auth-02
Published:	1 November 2023
Intended Status:	Informational
Expires:	4 May 2024

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.

This document is a working copy of [I-D.draft-jfinkhaeuser-caps-for-distributed-auth] and may be somewhat newer than the IETF version. Please consider the IETF submitted version authoritative at any time.

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.io/. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-jfinkhaeuser-caps-for-distributed-auth/.

Discussion of this document takes place on the interpeer Working Group mailing list (mailto:interpeer@lists.interpeer.io), which is archived at https://lists.interpeer.io/pipermail/interpeer/. Subscribe at https://lists.interpeer.io/mailman/listinfo/interpeer. Working Group information can be found at https://interpeer.io/.

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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.

2. 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-13], this document uses plural pronouns.

3. Use Cases

If resilience is a desirable feature in distributed systems as Baran suggested, and a system is the sum of its functions, it follows that all of a system's functions must be resilient to failure individually to achieve resilience in the overall system.

The following explores use cases in which reliance on a centralized authorization negatively impacts system resilience, even when the remaining system is sufficiently resilient in principle.

3.1. IoT On-boarding

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

In such scenarios, new devices rarely have connectivity before completing the on-boarding process. 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.

This specific problem is also addressed in [POA-IOT] and related work.

3.2. 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 because connectivity is guaranteed. There are, however, classes of scenarios where this cannot be expected.

3.2.1. Remote Location

In order to guarantee BVLOS operations in very remote locations, research projects such as [ADACORSA] assume use cases in which ground control stations hand over control of a UAV to each other. In remote locations, two ground control stations between which handover occurs may not have direct connectivity to each other, nor indirect connectivity through the internet.

In such cases, it is necessary for the UAV to act as trustworthy transmission channel, by storing and forwarding authorization information.

3.2.2. 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 forcibly 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 must include prerequisites for dealing with absent connectivity. Such an emergency control handover must be possible even when connectivity is absent.

3.2.3. 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.

3.3. Zero Round-Trip-Time (0-RTT)

If fast authorization is a goal, reducing the number of roundtrips to establish a privilege follows. Reducing this to 0-RTT implies being able to store and send verifiable authorization data, rather than engaging in a more complex handshake.

Of course, authorization can only follow when authentication already occurred. Authentication in a 0-RTT protocol is predicated on prior key exchange and verification.

Both [WIREGUARD] and DTLS 1.3 [RFC9147] offer 0-RTT handshakes. In the former, keys are pre-shared out of band, because WireGuard is used to establish static VPN tunnels. Because mutual authentication is assumed to be part of this process, authenticated encryption is sufficient to ensure that the keys are safely associated with network addresses in a 0-RTT roundtrip.

By contrast, DTLS simply offers different kinds of handshakes. 0-RTT can only be used for reconnection when a previous full handshake has provided sufficient authentication.

In either case, when 0-RTT authentication is offered, 0-RTT authorization should also be offered in order not to introduce more latency at this latter stage.

3.4. Bike Rental

Standing in for similar schemes, urban bike rental is a candidate for distributed authorization.

As a user, you typically use a mobile application to purchase some usage time via an internet enabled service. You then use the application to open a lock on a candidate bike, use it, and lock it again on return.

But the authorization to use bikes from the service does not expire with the return of a bike. It is a feasible and supported use case to lock one bike e.g. to go into a shop, then find a new bike to return the shopping home, all within the same booked time slot.

If centralized authorization systems are used, each of these unlocking operations will require internet connectivity. If connectivity is not given, access to the service will be denied, even though contractually speaking, it must be given.

3.5. Principle of Least Authority (POLA)

The Principle of Least Authority refers to a design method for authorization systems in which a process is given only the authority required to perform its purpose, and no more.

This is in contrast to identity-based authorization systems, in which processes can perform any activity that an identified user is permitted to perform. More advanced systems such as Role-Based Access Control (RBAC) function similar to identity-based authorization, except that each identified user can hold multiple roles, and roles are typically more strictly limited.

On a spectrum where identity-based authorization exists on one end, and RBAC occupies some middle ground, POLA exists on the opposite end of identity-based authorization.

Capabilities are eminently suitable for use with POLA, because they can encode arbitrarily fine-grained access control data.

In principle, POLA is very similar to how modern mobile operating systems encode permissions: each app must explicitly request e.g. access to the camera, internet, file system, etc. However, where such permissions are applied only locally, capabilities can be transmitted to be applied on remote nodes. In effect, a local application can be granted access to a networked, rather than merely a local camera.

3.6. Human Rights Considerations

[RFC8280], updated by [I-D.draft-irtf-hrpc-guidelines-20], lists a number of distinct objectives that help support human rights in protocol design. The above distributed authorization scheme addresses a number of them, such as Connectivity, Reliability, Content agnosticism, Integrity, Authenticity, Pseudonymity, Censorship Resistance, Outcome Transparency, Adaptability, Decentralization and Security, and by way of producing this document, Open Standards.

Rather than address each in detail, suffice to say that the use of pseudonymous public keys, and proofs based on cryptographic signatures, the majority of these objectives are reached.

It remains to highlight that the scheme outlined in this document observes the end-to-end principle, precisely by temporally decoupling different concerns. This permits for almost arbitrarily disrupted connectivity, and thus also censorship resistance. As capabilities can travel entirely out-of-band to any resource data, e.g. by sneakernet or similar means, they can be a building block of protocols that provide better human rights protections than systems that rely on temporal coupling of authorization concerns.

4. Problem Space

As outlined in the above use-cases, distributed authorization is not a goal in itself, but may be desirable in a distributed system.

In many systems, authentication and authorization 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.

Since authorization is closely related to authentication, it is worth exploring briefly how authentication has dealt with distributing its operations.

4.1. 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.

Identification: The act of establishing the *attributes* that describe a *person* in sufficient detail to identify them.

Endowment: The act of associating attributes derived during the *identification* phase with an *identifier* such as may be used in *secret proving*.

Secret Proving: Logging in to a system requires proving that one is in possession of some *secret*. It usually also requires providing some *identifier* that the secret is associated with -- this permits tying secret proving back to *identification* and *endowment* steps.

This distinction becomes somewhat more relevant when we move towards distributed authentication schemes, which rely on public key cryptography. It's worth noting that it is possible to construct systems that rely solely on secret proving, even without identifiers of any sort.

What is typically understood as "authentication", however, is a combination of secret proving and endowment. Specifically, once the secret is proven, a user database is consulted to endow the identifier with associated attributes (i.e. a user profile).

In particular, identification occurs in a preparation phase, according to the system's policies -- sometimes prior to when an identifier is first associated with a secret, sometimes after that event, and often before access to the system's functions is granted.

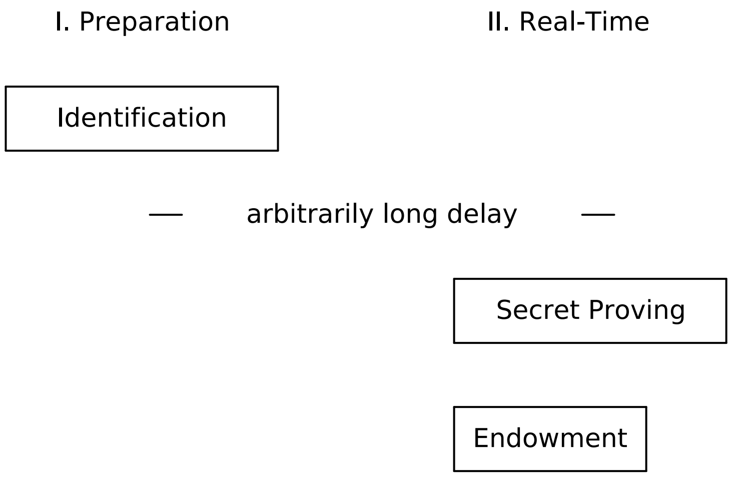

Figure 1: Phases in Password based Authentication Schemes

Note that a finer grained model is possible in which we distinguish between when identifiers are endowed with attributes *in principle* vs. *for use of the system*, but that is unlikely to clarify this document. We prefer a simpler model here.

4.1.1. 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 such as identification 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 identification has occurred.

Endowment in Web of Trust occurs when a sufficient number of sufficiently trustworthy signatures have been reached. The precise number of signatures and trust levels to be deemed sufficient is in the control of the recipient of transferable public key packets, however.

In either case, the major distinction compared to passwowrd based authentication schemes is that endowment is shifted from the real-time to the preparation phase.

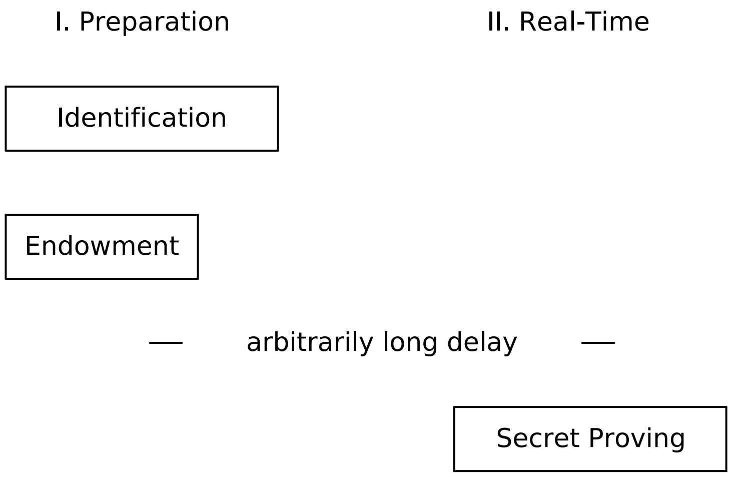

Figure 2: Phases in Public Key based Authentication Schemes

This shift alone is successful in removing the reliance on some user database in the real-time phase.

4.1.2. 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.

Other than this, [X.509] certificates behave like the web of trust in that endowment is shifted to the preparation phase.

4.1.3. 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.

4.2. Authorization

Authorization occurs only after secret proving. Once an identity has been established, it is then mapped to associated privileges, 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.

In the abstract, however, they each operate on the following pieces of information:

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. Similar to authentication above (Section 4.1), we can subdivide this into distinct steps that occur in phases.

Authorization Management: The first phase is authorization management , in which an *identifier* is associated with *authorization tuples* according to the management scheme in use.

Authorization Query: In the authorization query phase, a *request tuple* is used to look up in some store whether the request yields a positive or negative response. Conceptually, this resolves a *request tuple* into an *authorization tuple* -- or lack thereof. This effectively yields a boolean response to the request.

Access Granting: The final stage is to grant access based on whether the *authorization query* step succeeded or failed.

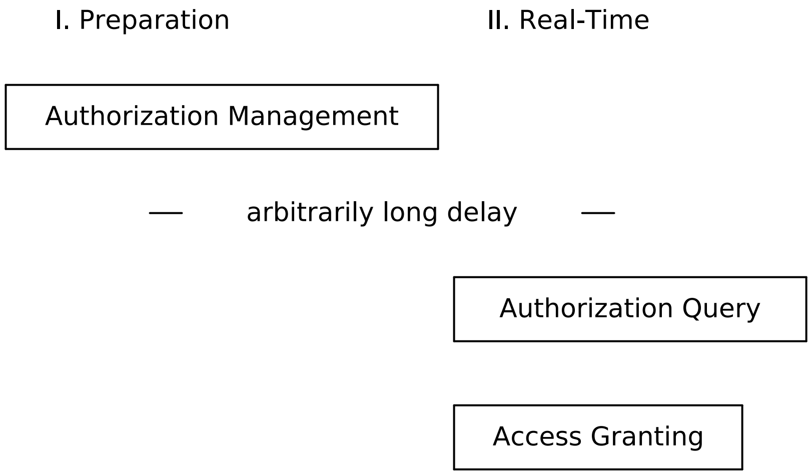

Figure 3: Phases in traditional Authorization

Management of authorization always occurs in a preparation phase. Upon request for access to a resource, systems perform authorization query, immediately followed by access granting. The conceptual translation from request tuple to authorization tuple is an implementation detail, and typically remains conceptual at best.

By contrast, distributed authorization instead deals in authorization tuples, which can be stored and distributed out-of-band. Because of this, they can encode the least authority (see POLA, Section 3.5), independent of which mapping system was chosen to manage authorization policies.

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] are also syntactically RDF triples. This implies that authorization tuples can encode arbitrarily complex authorization information by building the knowledge graph resulting from resolving such an RDF triple.

4.2.1. Single Point of Failure

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. Permitting an authority to manage this complexity in a single place is of course simpler than managing it across multiple locations.

It is tempting to then couple this authorization management function with the access granting function. The downside of this is that it also centralizes the access granting function, creating a single point of failure.

Particularly vulnerable to this single point of failure are private 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.

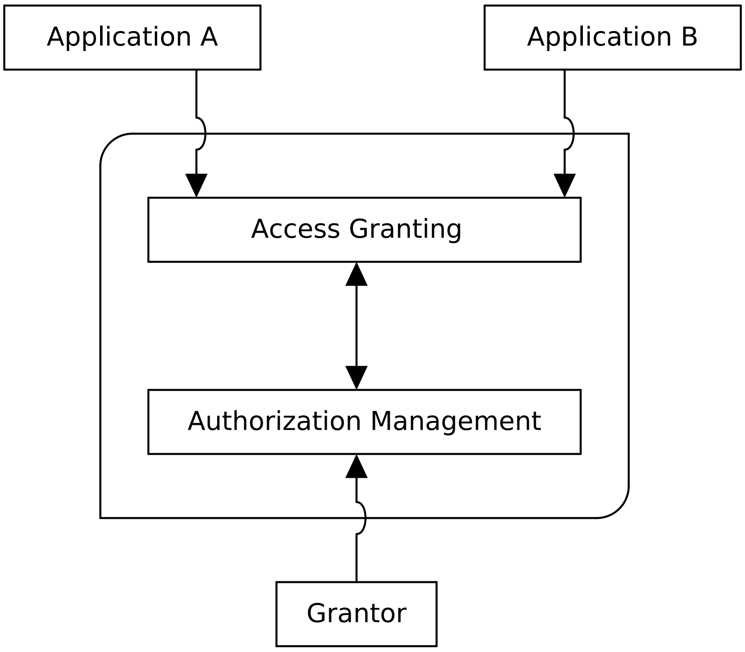

Figure 4: Single Point of Failure due to the complexity of Authorization Management

4.2.2. 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 a request tuple to a Boolean response tightly couples both steps.

It may be beneficial to disambiguate between participants in such a system.

From the perspective of the person operating the access control management system, granting access occurs whenever they make an entry into an authorization 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 the temporal dimension.

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

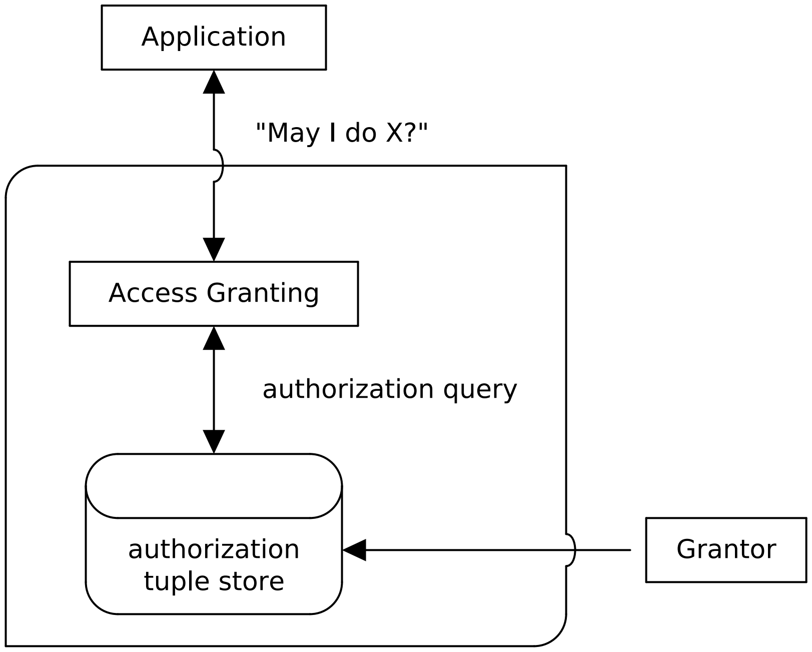

Figure 5: Temporal Coupling due to bundling Access Granting with Authorization Queries

5. 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.

5.1. 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.

5.2. 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 4.2.

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.

5.3. Pretty Good Privacy

While we previously discussed PGP in terms of authentication in Section 4.1.1, 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.

5.4. 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. One serialization format in JWT for this is [UCAN].

5.5. ZCAP-LD

Aimed at the linked data space, [ZCAP-LD] is an expression of cryptographic capabilities for authorization that relies heavily on linked data. While conceptually, the specification shares many similarities with the capability concept in this document, the use of linked data can lead to systems that do not provide for temporal decoupling .

Linked data has the downside here that data relationships may need to be resolved at the time of access granting , thus effectively re-introducing parts of an authorization query again at this point.

The concept of distributed systems underlying linked data thus differs fundamentally from the one in [RM3420]. Where the former treats distribution as distribution of concerns across different services providing parts of the linked data set, the latter is more concerned with resilience, specifically how to continue operating in the (temporary) absence of such services.

5.6. SPKI Certificate Theory

Cryptographic capabilities are not a new concept. [RFC2693] provides a thorough analysis of how they may be modelled in [X.509] certificates, and includes a discussion on various methods for modelling authorization that also draws comparison to PGP above.

The two major drawbacks of this document are:

1. Being a thorough theoretical discussion, it is dense. It does not provide an easy path for implementors to design capability based authorization systems; as a reference, however, it is indispensable.
2. With its focus on X.509 certificates as the encoding and transport mechanism for authorization material, it does not lend itself well to a number of use cases; this may be why overlapping proposals exist.

   Recall how DTLS [RFC9147] contains provisions for handling large certificates, and that JWT and ZCAP-LD provide serializations more suitable to web technology. Arguably, a discussion of the principles of authorization removed from the specifics of X.509 are required.

In the light of that RFC, this document focuses on relative simplicity and freedom from specific serialization formats for capabilities.

5.7. Macaroons

Capabilities bear resemblance to Macaroons, presented in 2014 [MACAROONS]. The Macaroons paper outlines a system of *caveats* under which access may be granted to the bearer of a macaroon token. Macaroons are secured using an HMAC. The paper also lays out the potential use of public-key based security for Macaroons.

Macaroons may be viewed as a specific implementation of the concept of capabilities with the focus of providing authorization in cloud environments. The paper positions the solution primarily as adding finer granularity to bearer tokens. In doing so it lays out a more complex caveat scheme than this document, which excludes the specifics of the privileges that may be issued.

5.8. CapBAC

The CapBAC system [CAPBAC] is focused on IoT applications, and in particular attempts to solve the complexity problems in this application domain. The paper is predominantly concerned with mapping abstract authorization concepts into specific privileges encoded in a capability. It is thus an excellent companion and motivation for this document.

The focus of this document, by contrast, is on mapping out the minimal necessary components of making capabilities workable.

5.9. 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. [POA-IOT] describes such a system for the Internet-of-Things.

6. 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.

More specifically, it moves the authorization query step from the real-time phase to the preparation phase, much as public key based authentication does.

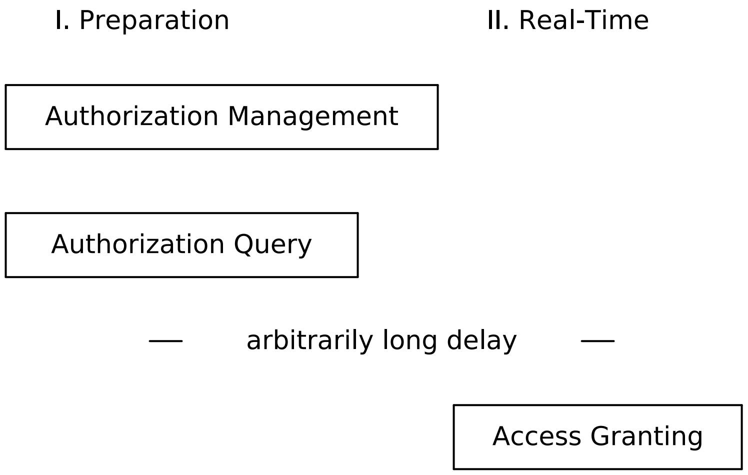

Figure 6: Phases in Capability-Based Authorization

It similarly employs public key cryptography to achieve this. Once the authorization query has resulted in the presence or absence of an *authorization tuple*, that fact is recorded in some serialized and signed data, the *capability* token itself. This can then be verified later for access granting.

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

6.1. 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.

6.1.1. 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** exist a mechanism for uniquely mapping the grantor public key to the grantor identifier and vice versa. This mapping permits verification.

6.1.2. Grantor Signature

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

6.1.3. Grantor vs. Issuer

It's worth noting that this makes the grantor identical to the more commonly used term of an "issuer". That latter term is useful when discussing cryptographic actions, i.e. the issuance of a signature and by extension the signed data.

However, beyond cryptography, the term has little semantic meaning. It does not describe the meaning behind issuing a signature. The grantor term by contrast encodes that in issuing a cryptographic signature, a privilege is granted. It has meaning for the domain of authorization.

6.2. 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.

6.3. 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.

6.4. 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.

This is not fundamentally different to the delay between the preparation and real-time phase employed so far in this document, except to add the requirement that the capability is transmitted *at some point* during this delay.

In practice, this could be at the beginning of the delay, or when the access request is made. They key point, however, is that this transmission of the capability encompasses *all* the communication that occurs between the functions in the real-time phase, and those in the preparation phase, and thus decouples these functions in time.

6.5. 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.

6.5.1. 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.

6.6. Object

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

6.6.1. 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.

6.7. 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.

6.8. 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.

6.9. 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.

6.9.1. 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.

6.10. Authorization Process

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

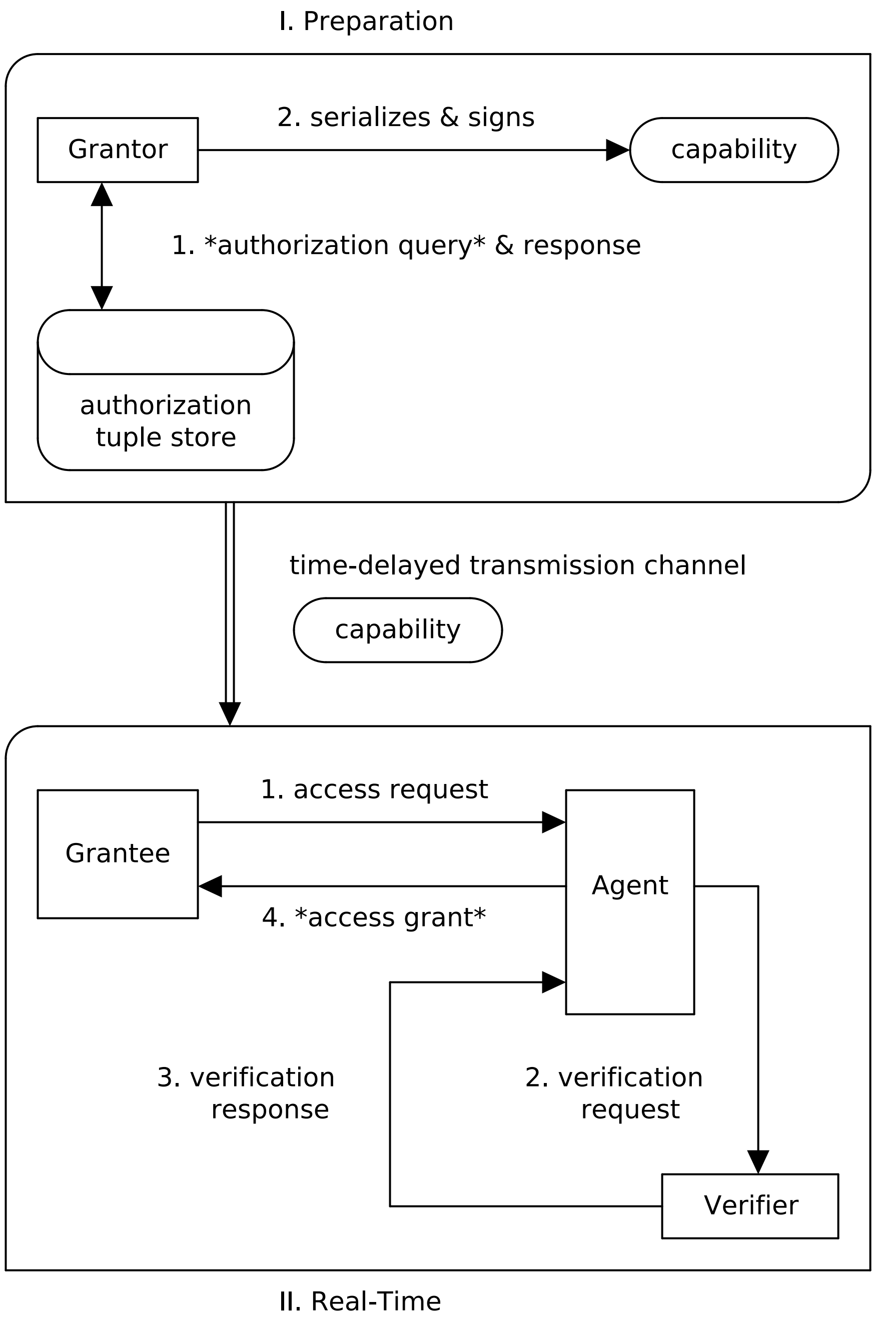

Figure 7: Process for Capability-based Distributed Authorization

The process is split into the two preparation and real-time 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 transmission 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.

7. 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.

8. Related Considerations

8.1. Human Rights Considerations

This document lists human rights considerations as a use case, see Section 3.6.

8.2. Protocol Considerations

There are no specific protocol considerations for this document.

However, protocols transmitting capabilities **MAY** provide some relief to human rights concerns Section 3.6, e.g. by providing confidentiality via encrypted transmission.

8.3. 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.

As such, some security considerations are supported by the use of capabilities for distributed authorization, such as preventing unauthorized usage and inappropriate use.

Some notes on specific considerations follow.

8.3.1. 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 4.2.1) 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.

8.3.2. Revocation

As ICAP was criticized for introducing a centralized solution for revocatins, (see Section 5.2), 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 transmission 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.

8.3.3. Replay Attacks

As presented, capabilities can be arbitrarily replayed, which suggests that replay attacks are possible.

This is only somewhat correct. It is correct that an attacker *who has successfully authenticated with the key specified in the capability* can successfully replay capabilities at will.

The difference to traditionally authorizing systems is that changes in the authorization tuple store do not reach the second phase until revocation tokens have been issued and distributed. The risk here is exactly the same as in e.g. TLS based systems without real-time certificate revocation lists [RFC5280].

In principle, nothing prevents a system employing capabilities to add similar real-time revocation lists. It must be noted that such a system will then re-introduce a potential single point of failure.

Depending on the security guarantees the system intends to provide, a better approach, and generally the approach envisioned here, may be to scope the validity of capabilities to a suffiicently short time frame that such replays are unlikely to cause significant harm.

In deciding how to guarantee appropriate levels of security, it should also be considered that it is in the nature of fully distributed systems that information may always be inconsistently distributed -- therein lies their greatest risk. At the same time, this risk of inconsistency is also what produces their greatest appeal, namely resilience in the face of partial communication or system failures.

8.4. Privacy Considerations

As part of supporting human rights considerations as a first class use case, exploring privacy considerations as covered by [RFC6973] is worthwhile.

In particular, distributed authorization schemes address the concerns of: intrusion and misattribution (as related to pseudonyms only).

It's also worth highlighting that surveillance and stored data concerns, as well as disclosure, are *not* addressed. In order for distributed capabilities to work, *any* likely recipient needs to be able to decode them.

The threat model then assumes that all capability data is accessible to anyone, which is why the use of pseudonymous public-key based identifiers is suggested. Sufficient care must be taken in key rotation, etc. in order to provide additional protections.

Note that despite this, nothing prevents a system from encrypting capabilities for use only by a single authorized party, which means that these last concerns can be addressed in the surrounding system.

8.5. IANA Considerations

This document has no IANA actions.

9. References

9.1. 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>.

9.2. Informative References

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

[CAPBAC] Gusmeroli, S., Piccione, S., and D. Rotondi, "A capability-based security approach to manage access control in the Internet of Things", Elsevier BV, Mathematical and Computer Modelling vol. 58, no. 5-6, pp. 1189-1205, DOI 10.1016/j.mcm.2013.02.006, September 2013, <https://doi.org/10.1016/j.mcm.2013.02.006>.

[I-D.draft-irtf-hrpc-guidelines-20] Grover, G. and N. ten Oever, "Guidelines for Human Rights Protocol and Architecture Considerations", Work in Progress, Internet-Draft, draft-irtf-hrpc-guidelines-20, 4 October 2023, <https://datatracker.ietf.org/doc/html/draft-irtf-hrpc-guidelines-20>.

[I-D.draft-jfinkhaeuser-caps-for-distributed-auth] Finkhäuser, J. and S. P. ISEP, "Capabilities for Distributed Authorization", Work in Progress, Internet-Draft, draft-jfinkhaeuser-caps-for-distributed-auth-01, 1 June 2023, <https://datatracker.ietf.org/doc/html/draft-jfinkhaeuser-caps-for-distributed-auth-01>.

[I-D.draft-knodel-terminology-13] Knodel, M. and N. ten Oever, "Terminology, Power, and Inclusive Language in Internet-Drafts and RFCs", Work in Progress, Internet-Draft, draft-knodel-terminology-13, 18 January 2023, <https://datatracker.ietf.org/doc/html/draft-knodel-terminology-13>.

[ICAP] Gong, L., "A secure identity-based capability system", IEEE Comput. Soc. Press, Proceedings. 1989 IEEE Symposium on Security and Privacy, 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/>.

[MACAROONS] Birgisson, A., Politz, J. G., Erlingsson, Ú., Taly, A., Vrable, M., and M. Lentczner, "Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud", 2014, <https://research.google/pubs/pub41892/>.

[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>.

[POA-IOT] Vattaparambil Sudarsan, S., Schelén, O., and U. Bodin, "Multilevel Subgranting by Power of Attorney and OAuth Authorization Server in Cyber–Physical Systems", Institute of Electrical and Electronics Engineers (IEEE), IEEE Internet of Things Journal vol. 10, no. 17, pp. 15266-15282, DOI 10.1109/jiot.2023.3265407, September 2023, <https://doi.org/10.1109/jiot.2023.3265407>.

[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/>.

[RFC2693] Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., and T. Ylonen, "SPKI Certificate Theory", RFC 2693, DOI 10.17487/RFC2693, September 1999, <https://www.rfc-editor.org/rfc/rfc2693>.

[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>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[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>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/rfc/rfc9147>.

[UCAN] UCAN Working Group, "User Controlled Authorization Network", n.d., <https://ucan.xyz/>.

[WIREGUARD] Donenfeld, J. A., "WireGuard: Next Generation Kernel Network Tunnel", 1 June 2020, <https://www.wireguard.com/papers/wireguard.pdf>.

[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.

[ZCAP-LD] Lemmer-Webber, C., Sporny, M., and M. S. Miller, "Authorization Capabilities for Linked Data v0.3", 22 January 2023, <https://w3c-ccg.github.io/zcap-spec/>.

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/