t2trg
Internet-Draft
Intended status: Experimental
Expires: 9 January 2026

C. Amsüss
M. Tiloca
R. Höglund
RISE AB

8 July 2025

Using onion routing with CoAP
draft-amsuess-t2trg-onion-coap-04

Abstract

The CoAP protocol was designed with direct connections and proxies in mind. This document defines mechanisms by which chains of proxies can be set up. In combination, they enable the operation of hidden services and of clients similar to how Tor (onion routing) enables it for TCP-based protocols.

Discussion Venues

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

Discussion of this document takes place on the Thing-to-Thing Research Group mailing list (t2trg@irtf.org), which is archived at https://mailarchive.ietf.org/arch/search/?email_list=t2trg.

Source for this draft and an issue tracker can be found at https://gitlab.com/chrysn/onion-coap.

Status of This Memo

Table of Contents

1. Introduction

The Constrained Application Protocol (CoAP) protocol [RFC7252] was designed with direct connections and proxies in mind. This document defines mechanisms by which chains of proxies can be set up. In combination, they enable the operation of hidden services and of clients similar to how Tor (onion routing) enables it for TCP-based protocols.

Onion CoAP is a framework for building a multi-party anonymization network with the goal of enhancing endpoints' privacy through anonymous communication, while providing protection against traffic analysis. The key goal of its design is to provide confidential and anonymous communication.

Achieving this goal critically depends on multiple non-colluding parties providing particular network services. Specifically, Onion CoAP takes advantage of layered encryption of messages and relaying of traffic through a chain of proxies. Each layer of encryption is removed at successive proxies in the chain, ensuring that no single proxy or network monitor can determine both the source and destination of a given communication flow.

Onion CoAP uses Object Security for Constrained RESTful Environments (OSCORE) [RFC8613] to protect communications, and Ephemeral Diffie-Hellman Over COSE (EDHOC) [RFC9528] for establishing the required OSCORE associations.

1.1. Terminology

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.

2. Uses cases

Using Onion CoAP allows for minimizing information disclosure, by hiding the network location of the communicating endpoints as well as their traffic pattern. This also prevents linking the physical position of individuals that use communication endpoints to their organizational or personal affiliations, or inferring their activities. For example, Onion CoAP is relevant in the following use cases.

2.1. Transport Applications

Some industrial applications can have particularly high privacy and confidentiality requirements, by deeming physical positions and communication patterns of the different parties to be sensitive information and a critical asset to protect.

For instance, this can be the case for businesses operating in logistics and for their customers, as interested in a highly discrete tracking of transported goods that exposes as little information as possible.

The use of Onion CoAP ensures that the physical location of goods remains undisclosed to unintended and unauthorized parties, thus reducing risks such as tampering, theft, or industrial espionage, e.g., by competitors or saboteurs. Furthermore, anonymizing the communication endpoints and their traffic patterns protects cargo details and logistics chains from being linked to specific operators, routes, or customers.

A concrete example is a transport container that includes tracking sensors belonging to a manufacturer company. The container contains products made by such manufacturer and is physically anonymized (i.e., the container does not provide any visual hint pointing to the manufacturer).

As the container is moved from one harbor to another, the sensor therein is intended to send a status update to the manufacturer server. When this happens, the manufacturer does not want to reveal itself and its position as something understandable by observing the communication started by the sensor.

For instance, the manufacturer wants to keep the details of its logistics strategy confidential, including the routes and schedule of its delivery and distribution process. The reliability and commercial success of the manufacturer may well rely on such an optimized combination or routes and schedule, which is a key asset to not expose to competitors.

2.2. Hidden Deployments

In some scenarios, not only is the physical location of communicating endpoints a critical asset to protect, but even their deployment and topology should be hidden and impractical to infer. That is, the fact that communication devices are deployed and the details of their deployment should themselves be hidden.

For example, one may want to build a network of (mobile) sensors and actuators that are confidentially and secretly deployed in locations which must not be revealed, and from where they act as clients, servers, or both.

Concrete use cases can include the supporting of military actions such as intelligence collection and assistance for in-the-field operations, or circumventing repressive actions such as censorship and (pervasive) traffic monitoring.

3. High-level Overview

3.1. Design considerations

The network described in this document is designed to allow participation of Class 1 devices as defined in [RFC7228] as servers and clients. It should reuse building blocks these devices will already implement if they use EDHOC [RFC9528] for authenticated key establishment and OSCORE [RFC8613] for encryption. Operations that are costly for constrained devices, such as creating and verifying signatures, should not be part of regular operation.

3.2. Components overview

This document introduces separate mechanisms that in combination enable setups similar to how Tor is used for anonymous web access and anonymous hosting of web sites. Some of the mechanisms need no new protocol components, but merely describe which existing steps are used to obtain the desired results.

* A client can use EDHOC to establish a unilaterally authenticated OSCORE context with proxies (see Section 4.1).

* A server can use EDHOC to establish a unilaterally authenticated OSCORE context to establish a reverse proxy address (see Section 4.2).

* A discovery mechanism for proxies (see Section 4.4).

* A naming and discovery mechanism for hidden services (see Section 4.3).

Note that these mechanisms should be largely independent: A server that does not intend to hide its position can still advertise a cryptographic name at its real network coordinates, and thus be available both to clients that do hide their location (even if their proxies do not work as “exit nodes” in Tor terminology) and to clients on a local network.

Figure 1 illustrates an example topology, and Figure 2 illustrates a cross-section of the OSCORE layers along one path.

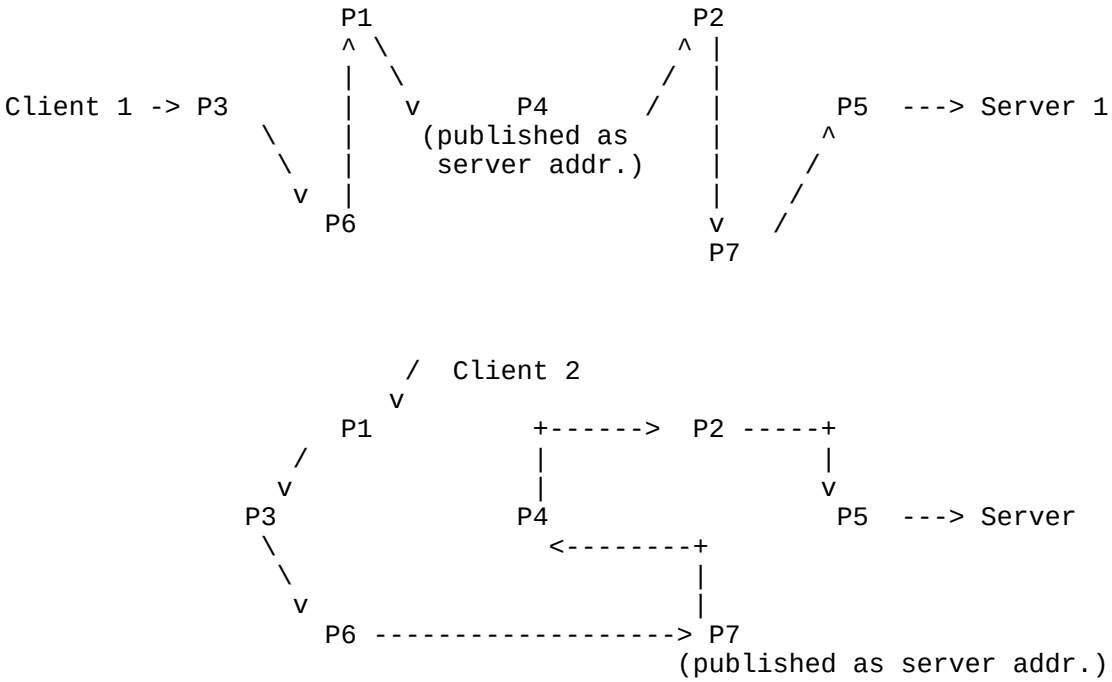

Figure 1: Example topology of an onion style CoAP network showing two routes in separate graphs. Note that the hop P4→P2 is present in both chains, and can be pooled into one TLS connection

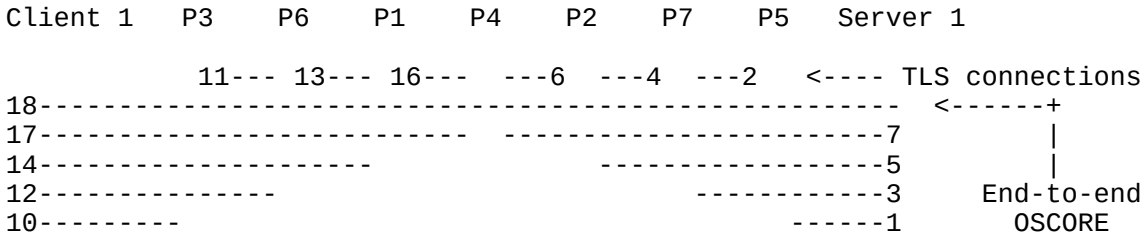

Figure 2: Cross section of the Client 1 → Server 1 connection of [TBD reference from label]. Numbers indicate the sequence in which EDHOC is performed (precisely: message 1 is transmitted over the wire through whichever encapsulation), and are placed on the side of the initiator. Not depicted at step 8: Server 1 and/or P4 (TBD) publishes P4 as the public address of the hidden service; 9: client obtains the list of available proxies. 15: Client 1 looks up the introduction point of Server 1 through the proxy chain up to P1 to discover P4 (may also be implicit).

4. Mechanisms

4.1. Client proxy chain

A client can pick one or more proxies to hide its position in the network.

Without OSCORE proxies, only one proxy hop can be chosen, because the CoAP requests contains at most two addresses: The address in the IP header, and the address in the Uri-Host option. With the mechanisms introduced in [I-D.ietf-core-oscore-capable-proxies], CoAP request can contain a Uri-Host option in each layer of OSCORE, effectively building a source routing chain.

To build the chain, the client first chooses its first proxy hop, and runs EDHOC to establish an OSCORE context. In this process, the proxy authenticates with its long-term credentials, whereas the client uses an ephemeral key (a plain CWT Cliams Set, [RFC8392]). The process can take as little as one round-trip per proxy; when message 3 of EDHOC is sent along with the OSCORE message (see [I-D.ietf-core-oscore-edhoc]) that contains the next hop' s message 1,

Once one proxy context is established, EDHOC can be run through that proxy with the next proxy, until a chain of sufficient length has been established. Care has to be taken to never use one of the later proxies with any chain other than the chain through which the connection was established, for otherwise the client can be deanonymized mor easily.

When forwarding messages, every forward proxy strips off a layer of OSCORE from the request, and adds one to the response.

Possible optimizations:

* Can EDHOC be run without transmitting two public keys (G_X and G_I) for the client? (I.e., Can G_X be re-used as G_I without harm to EDHOC (likely not), and how would that be communicated?)

* For hops that are only ever used with a single next-hop, as is typical with all but the first proxy (see guidance below): Can default values for Proxy-Scheme and Uri-Host be communicated during EDHOC, values that would later be elided? Otherwise, every request would contain explicit addresses of the full chain. If taken to the extreme, this might be setting up a SCHC context that also compresses parts of the OSCORE option, where the client tells each proxy what the KID used with the next proxy is, and uses the same sender sequence number for the hops. (This has own security considerations; might be necessary to apply offsets, at which point it gets overly complex).

  Effectively, setting a default value for Proxy-Scheme and Uri-Host makes that (originally forward) proxy a reverse proxy.

4.1.1. Guidance for setting up proxy chains

TBD: This section should contain guidance distilled from Tor operations. In particular, it might recommend that a client pick one proxy hop as a long-term first hop, while building the remaining chain individually for each new origin server.

Following common tor practice, it is expected that typical chain lengths are around 3 hops. Note that the amount of processing on the peer side is independent of the length of the chain chosen by a host. If a client deems a one-hop setup sufficient and only has resources for maintaining one extra OSCORE context, it can still use a server that is hidden behind a 3 long proxy chain.

4.2. Server proxy chain

A server can pick one or more proxies to hide its position in the network.

Unlike forward proxies, which are configured per request, this requires a dedicated mechanism.

TBD: This document does not yet specify such a mechanism, but may draw upon the reverse proxy request of Section 2 of [I-D.amsuess-core-resource-directory-extensions].

When forwarding messages, every reverse proxy adds a layer of OSCORE to the request, and removes one from the response.

Possible optimizations:

* Rather than explicitly advertise the name for which the proxies should be set up, the advertised name could be derived from the CRED_x used in EDHOC.

4.3. Naming and name resolution

The mechanisms discussed in [I-D.amsuess-t2trg-rdlink] can be used by hidden services to come up with names for their services. (That document will need to be updated to use mechanisms from Appendix F of [I-D.ietf-core-transport-indication]).

Along with the service's public key (that is announced as part of the name), the published record may also include the public key of the introduction point, as that will allow the client to establish an extra layer with the introduction point. As the published record is not trusted, the client can use the EAD option described in Appendix D of [I-D.ietf-core-transport-indication] to verify the proxy's public key as part of the end-to-end session. If client and server support this, they can rule out that an attacker might advertise itself as the introduction address and could thus monitor large portions of the traffic toward a hidden service (even though that attacker would still not learn the location of the server, the location of hidden clients, or the content of the communication). As an alternative (TBD: when would which be chosen), the client's last chosen proxy, when seeing the cryptographic address of the hidden service, may not just establish an EDHOC session with the introduction proxy, but also with the hidden service, therein performing the same verification. The server should therefore allow for at least one level of nesting within incoming EDHOC sessions.

4.4. Proxy discovery

A mechanism for discovering forward proxies is already described in [I-D.ietf-core-transport-indication]; discovery of reverse proxies suitable for servers will depend on the precise mechanism used.

4.4.1. Discovering the introduction proxy for services

Services with cryptographic identifiers outlined in {#naming} can register these names in a distributed Resource Directory following the same [I-D.amsuess-t2trg-rdlink] style setup. Unlike described there, they would not enter their network address into the distributed directory, but the address of their most remote reverse proxy (the introduction point).

This directory propagates changes relatively fast, limited by the performance of the underlying Distributed Hash Table (DHT).

Clients looking for services may not need to use the discovery service directly: Instead, they can send requests to a proxy of their chosing, and rely on the proxy to utilize the directory to look up a next hop. (They do need to perform discovery of the introductory node if they want to hide the ciphertext of their conversation from their last proxy and establish a secure connection to the introduction proxy chosen by the server, verifying it using the EAD option described in Appendix D of [I-D.ietf-core-transport-indication] instead of relying on their own last proxy).

4.4.2. Discovery for eligible forward and reverse proxies

In order to hide their location, clients as well as servers need to discovery lists of eligible proxies, along with metadata that indicates whether the proxy is willing to proxy to arbitrary locations on the Internet, or merely to hidden peers.

That distinction in forwrad proxies would be similar to how Tor distinguishes relay and exit nodes. In reverse proxies, there is an analogous distinction that is not so much based on policy but rather on the structure of the authority component used by that reverse proxy: If the proxy can offer names that are resolvable on regular CoAP stacks (i.e., DNS can resolve it to a global IP address), then regular CoAP clients can use the introduction address as an entry point. The hidden service trusts the user to establish an end-to-end connection: If the client is unauthenticated (i.e., using a plain CCS as its credential), the hidden server can not tell whether the incoming EDHOC session is end-to-end or merely set up by a proxy, let alone whether the client is using a chain of proxies or not. Many proxies may not offer such names, and services may not want to rely on such names anyway -- in that case, clients are required to use (most probably by proxy) the DHT in which services are announced.

Maintenance of this list is out of scope of this document, but the produced list will have some properties required for the constrained devices: * For each proxy that is available to form a hiding circuit, the list includes: * the proxy’ s cryptographic identity (eg. in a CCS): to authenticate the proxy, * affiliation information (operator and location): this enables hiding nodes to find paths of probably non-colluding proxies * optionally a public IP address: this enables nodes to use the proxy as a first hop * The list is updated regularly, with an update rate measured in hours or a few days. * The list needs to be signed by independent entities. (This is the only place in the whole setup where signatures are required: it appears unrealistic that the maintainers of the network will be online to perform non-signing challenges for the document all the time. Devices that can not even perform that verification might have a

trusted source, possibly their firmware update source, that performs the verification for them). * The list's size will excede the memory capacity of individual devices, so it needs to be split up, possibly in a way similar to a Merkle tree. (At a bare minimum, a Tor sized network of 10k nodes with 32 bytes of key material for each node would already exceed the RAM available to Class 2 devices [RFC7228]). It may be beneficial for long-term stability if the list is structured such that there is always a fragment with long-term stable addresses that nodes can store.

TBD: Describe operations of this service in a separate document.

The three tasks of proxying, participation in the distributed Resource Directory and participation in the dissemination of the proxy list are conceptually separate. None the less, it is expected that proxies eligible for the list will perform all those roles.

Nodes partipating in this network will always keep at least some verified fragments of the list across restarts, and should be provisioned with a current state of the list at setup time. As the proxies also provide the list, devices can obtain the latest version through the first EDHOC connection they establish with a proxy they know from the most recent version the have. For the unlikely event that all stored proxies have become unavailable, nodes may accept recent signed versions of the list through other means.

4.5. Establishing TLS connections between proxies

Proxy-to-proxy requests, which are the majority of transmitted request, are transmitted between unconstrained devices across the Internet. As such, protecting those connections with an extra layer of TLS (as specified in [RFC8323]) is desirable, because

* it profits from TCP's flow control,

* it hides more request metadata (even though none of the metadata visible at this point should be linkable to the server or the client, with the exception of message length), and

* it allows requests to be mixed across MTU and thus to hide their length and number (provided there is sufficient traffic on the link to ensure that multiple messages are processed within one Nagle interval).

[TBD: Explore whether coercing traffic through specific pairs of nodes instead of random node pairings would make sense. If it is dangerous, maybe servers might pair up on their own to ensure that it is hard to monitor their ingress and egress traffic for correlation.]

A challenge in establishing TLS connections on that link is that proxies are advertised with EDHOC credentials in the network's discovery area. The tools of [I-D.tschofenig-tls-cwt] may help bridging that gap. If that work does not progress, proxies might establish an EDHOC session inside an intially unauthenticated / self-signed TLS session, tying the sessions together by the use of a data item exported from the TLS key material exporter.

4.6. Other tricks

TBD. Current ideas:

* For increased privacy, it may make sense to spool requests and responses in proxies for "as long as practical". Those setting up the proxy may indicate that in the security context. While it increases the proxy's memory requirements a lot to keep several seconds of traffic around, it is not expected that these proxies will be operating at network capacity limits.

* Add chatter between proxies. With the stark contrast between constrained device bandwidths and Internet bandwidths, this can be tolerable.

* Access point assistance: While all of this is aimed at constrained devices as defined in [RFC7228], devices of Class 1 may not be capable of using the full proxy discovery service or setting up security contexts with all the hops in the chain. Instead, they may only provide end-to-end encryption, and use a service provided by a local node (e.g. the border router in a 6LoWPAN network [RFC8138]) to set up the hops. Such a setup can also be used to defer policy decisions to the network, which may decide to advertise its own address as an introduction point, or use a suitable length of reverse proxies.

* The introduction point would be the only suitable location to place a caching proxy when [I-D.amsuess-core-cachable-oscore] is used. As the server is be aware that cacheable OSCORE is used, it can select a reverse proxy that was advertised with caching capabilities (with that metadatum still TBD).

4.7. Overhead and optimizations

TBD. Main points:

* Establishing a default Uri-Host likely gives most savings.

* For intermediate hops, using a shorter AEAD tag might be an option.

5. Security Considerations

* When using proxy chains, only contact a proxy through the one chain it is set up with, and only accept messages into a context if they were transported in the hop they are expected to be received from.

  It is of utmost importance to not have observably different behavior between messages with an unknown context and messages whose context is known but not expected at this point. For example, if an attacker controls a server's introduction point and intends to deanonymize clients, it may attempt to send responses directly to the suspected address of the client.

  In implementations, this can be mitigated by first looking up the list of contexts depending on the outer layer, and then looking up inside that list whether the security context is known and the message expected.

* What are the effects of sequence numbers on correlation? Is it good or bad to use the same sequence number for all hops in a chain?

6. IANA Considerations

TBD.

7. References

7.1. Normative References

[I-D.ietf-core-oscore-capable-proxies]
Tiloca, M. and R. Hglund, "OSCORE-capable Proxies", Work in Progress, Internet-Draft, draft-ietf-core-oscore-capable-proxies-04, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-capable-proxies-04>.

[I-D.ietf-core-oscore-edhoc]
Palombini, F., Tiloca, M., Hglund, R., Hristozov, S., and G. Selander, "Using Ephemeral Diffie-Hellman Over COSE (EDHOC) with the Constrained Application Protocol (CoAP) and Object Security for Constrained RESTful Environments (OSCORE)", Work in Progress, Internet-Draft, draft-ietf-core-oscore-edhoc-11, 9 April 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-core-oscore-edhoc-11>.

[I-D.ietf-core-transport-indication]
Amsss, C. and M. S. Lenders, "CoAP Transport Indication", Work in Progress, Internet-Draft, draft-ietf-core-transport-indication-09, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-core-transport-indication-09>.

[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>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/rfc/rfc7252>.

[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>.

[RFC8323] Bormann, C., Lemay, S., Tschofenig, H., Hartke, K., Silverajan, B., and B. Raymor, Ed., "CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets", RFC 8323, DOI 10.17487/RFC8323, February 2018, <https://www.rfc-editor.org/rfc/rfc8323>.

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/rfc/rfc8392>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/rfc/rfc8613>.

[RFC9528] Selander, G., Preu Mattsson, J., and F. Palombini, "Ephemeral Diffie-Hellman Over COSE (EDHOC)", RFC 9528, DOI 10.17487/RFC9528, March 2024, <https://www.rfc-editor.org/rfc/rfc9528>.

7.2. Informative References

[I-D.amsuess-core-cachable-oscore]
Amsss, C. and M. Tiloca, "Cacheable OSCORE", Work in Progress, Internet-Draft, draft-amsuess-core-cachable-oscore-11, 6 July 2025, <https://datatracker.ietf.org/doc/html/draft-amsuess-core-cachable-oscore-11>.

[I-D.amsuess-core-resource-directory-extensions]
Amsss, C., "CoRE Resource Directory Extensions", Work in Progress, Internet-Draft, draft-amsuess-core-resource-directory-extensions-11, 6 November 2024, <https://datatracker.ietf.org/doc/html/draft-amsuess-core-resource-directory-extensions-11>.

[I-D.amsuess-t2trg-rdlink]
Amsss, C., "rdlink: Robust distributed links to constrained devices", Work in Progress, Internet-Draft, draft-amsuess-t2trg-rdlink-01, 23 September 2019, <https://datatracker.ietf.org/doc/html/draft-amsuess-t2trg-rdlink-01>.

[I-D.tiloca-core-oscore-capable-proxies-06]
Tiloca, M. and R. Hglund, "OSCORE-capable Proxies", Work in Progress, Internet-Draft, draft-tiloca-core-oscore-capable-proxies-06, 5 April 2023, <https://datatracker.ietf.org/doc/html/draft-tiloca-core-oscore-capable-proxies-06>.

[I-D.tschofenig-tls-cwt]
Tschofenig, H. and M. Brossard, "Using CBOR Web Tokens (CWTs) in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", Work in Progress, Internet-Draft, draft-tschofenig-tls-cwt-02, 13 July 2020, <https://datatracker.ietf.org/doc/html/draft-tschofenig-tls-cwt-02>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <https://www.rfc-editor.org/rfc/rfc7228>.

[RFC8138] Thubert, P., Ed., Bormann, C., Toutain, L., and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header", RFC 8138, DOI 10.17487/RFC8138, April 2017, <https://www.rfc-editor.org/rfc/rfc8138>.

Appendix A. High-Level Comparison with Tor

TBD.

Appendix B. Examples

TBD.

Appendix C. Change log

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

Since -03:

* Refresh to keep document active.

* Minimal editorial fixes.

Since -02:

* Fixes in the markdown.

* Revised table of contents.

* Updated and fixed references.

* Editorial fixes.

* Acknowledgments.

* Include use cases.

* Add introduction.

Since -01:

* Elaborate on bootstrapping: Describe list of proxies.

* Add design considerations section.

* Suggest exported material from TLS to bind EDHOC sessions if TLS-CWT does not fly.

* Various clarifications.

Since -00:

* Shaped into separate sections on the bits and pieces involved.

* Added illustrations.

* Moved all points from the previous notes in with the new text.

Since [I-D.tiloca-core-oscore-capable-proxies-06]:

* The main body of the text was moved here and will be absent from the -07 version of that document.

* An abstract was added.

Acknowledgments

The authors sincerely thank Carsten Bormann and Simon Bouget for their comments and feedback.

The work on this document has been partly supported by the Sweden's Innovation Agency VINNOVA and the Celtic-Next project CYPRESS.

Authors' Addresses

Christian Amsss
Austria
Email: christian@amsuess.com

Marco Tiloca
RISE AB
Isafjordsgatan 22
SE-16440 Kista
Sweden
Email: marco.tiloca@ri.se

Rikard Hglund
RISE AB
Isafjordsgatan 22
SE-16440 Kista
Sweden
Email: rikard.hoglund@ri.se