Workgroup: Network Working Group
Internet-Draft: draft-halen-fed-tls-auth-11
Published: 3 April 2024
Intended Status: Informational
Expires: 5 October 2024
Authors: J. Schlyter   S. Halén
Kirei AB   The Swedish Internet Foundation

Federated TLS Authentication

Abstract

This document describes the Federated TLS Authentication (FedTLS) protocol, enabling secure end-to-end communication within a federated environment. Both clients and servers perform mutual TLS authentication, establishing trust based on a centrally managed trust anchor published by the federation. Additionally, FedTLS ensures unambiguous identification of entities, as only authorized members within the federation can publish metadata, further mitigating risks associated with unauthorized entities impersonating legitimate participants. This framework promotes seamless and secure interoperability across different trust domains adhering to common policies and standards within the federation.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction
  1.1. Reserved Words
  1.2. Terminology
2. Federation Chain of Trust
3. Metadata Repository
4. Metadata Submission
5. Maintaining Up-to-Date Metadata
6. Authentication
  6.1. Public Key Pinning
  6.2. Pin Discovery and Preloading
  6.3. Verification of Received Certificates
  6.4. Failure to Validate
7. Federation Metadata
  7.1. Federation Metadata claims
    7.1.1. Entities
  7.2. Metadata Schema
  7.3. Example Metadata
  7.4. Metadata Signing
  7.5. Example Signature Protected Header
8. Example Usage Scenarios
  8.1. Client
  8.2. Server
  8.3. SPKI Generation
  8.4. Curl and Public Key Pinning
9. Security Considerations
  9.1. TLS
  9.2. Federation Metadata Updates
  9.3. Verifying the Federation Metadata Signature
10. Acknowledgements
11. IANA Considerations
12. Normative References
13. Informative References
Appendix A. JSON Schema for FedTLS Metadata
Authors' Addresses

1. Introduction

This document outlines the Federated TLS Authentication (FedTLS) framework, which facilitates secure end-to-end communication between two parties within a federation. Both the client and server undergo mutual TLS authentication (as defined in [RFC8446]), establishing a robust foundation of trust. This trust relies on a central trust anchor held and published by the federation, acting as a trusted third party connecting distinct trust domains under a common set of policies and standards.

The FedTLS framework leverages a centralized repository of federation metadata to ensure secure communication between servers and clients within the federation. This repository includes information about public keys, certificate issuers, and additional entity details, such as organizational information and service descriptions. This centralized approach simplifies certificate management, promotes interoperability, and establishes trust within the federation. By directly accessing the federation metadata, efficient connections are established, eliminating manual configuration even for new interactions.

Without a FedTLS federation, implementing mutual TLS authentication often requires organizations to establish their own PKI infrastructure (or rely on third-party CAs) to issue and validate client certificates, leading to complexity and administrative burden. FedTLS allows the use of self-signed certificates, potentially reducing costs and administrative overhead. While self-signed certificates inherently lack the trust level of certificates issued by trusted CAs, the strong trust within the FedTLS framework is established through several mechanisms, including public key pinning [RFC7469] and member vetting procedures. This ensures the validity and authenticity of self-signed certificates within the federation, fostering secure communication without compromising trust.

The Swedish education sector illustrates the value of FedTLS by securing user lifecycle management endpoints through this framework. This successful collaboration between school authorities and service providers highlights FedTLS's ability to enable trust, simplify operations, and strengthen security within federated environments.

1.1. Reserved Words

This document is an Informational RFC, which means it offers information and guidance but does not specify mandatory standards. Therefore, the keywords used throughout this document are for informational purposes only and do not imply any specific requirements.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.2. Terminology

* **Federation**: A trusted network of entities that adhere to common security policies and standards,using FedTLS for secure communication.

* **Federation Metadata**: A cryptographically signed document containing critical information about all entities within the federation.

* **Metadata Repository**: A centralized repository storing information about all entities within the federation.

* **Member Metadata**: Information about entities associated with a specific member within the federation.

* **Federation Member**: An entity that has been approved to join the federation and can leverage FedTLS for secure communication with other members.

* **Federation Operator**: The entity responsible for the overall operation and management of the federation, including managing the federation metadata, enforcing security policies, and onboarding new members.

* **Member Vetting**: The process of verifying and approving applicants to join the federation, ensuring they meet security and trustworthiness requirements.

* **Trust Anchor**: The federation's root of trust is established by the federation metadata signing key, which verifies the federation metadata and allows participants to confidently rely on the information it contains.

2. Federation Chain of Trust

Federation members submit member metadata to the federation. Both the authenticity of the submitted member metadata and the submitting member need to be ensured by the federation.

The federation operator aggregates, signs, and publishes the federation metadata, which combines all members' member metadata along with additional federation-specific information. By placing trust in the federation and its associated signing key, federation members trust the information contained within the federation metadata.

The trust anchor for the federation is established through the federation's signing key, a critical component requiring secure distribution and verification. To achieve this, the federation's signing key is distributed using a JSON Web Key Set (JWKS) [RFC7517], providing a flexible framework for exposing multiple keys including the signing key and keys for rollover. This structured approach ensures members can readily access the necessary keys for verification purposes.

An additional layer of security is introduced through thumbprint verification [RFC7638], where federation members can independently verify the key's authenticity. This involves comparing the calculated cryptographic thumbprint of the key with a trusted value, ensuring its integrity. Importantly, this verification process can be conducted through channels separate from the JWKS itself, enhancing security by eliminating reliance on a single distribution mechanism.

3. Metadata Repository

The FedTLS metadata repository serves as the cornerstone of trust within a federation. It acts as a central vault, securely storing all information about all participating federation members and their respective entities. This information, known as federation metadata, is presented as a JWS [RFC7515]to ensure its authenticity and integrity.

The metadata repository is subject to stringent security measures to safeguard the integrity and confidentiality of the stored information. This MAY involve:

* Member Management: The federation operator can centrally enforce security policies and vet new members before they are added to the repository.

* Access Controls: Only authorized members within the federation should have access to the repository.

* Regular Backups: Robust backup procedures ensure data recovery in case of unforeseen circumstances.

Before member metadata is added to the federation's repository, it is recommended that the submitted metadata undergo a validation process. This process aims to verify the accuracy, completeness, and validity of the information provided by a member. The validation process MAY include the following steps:

* Format Validation: The system checks if the submitted metadata adheres to the defined schema and format specifications.

* Unique Entity ID: Checks are performed to ensure that the entity_id in the submitted metadata is not already registered by

another member. Each entity within the federation must have a unique identifier.

* Unique Public Key Pins: Public key pins are utilized to locate the corresponding entity within the metadata upon establishing a connection. Through the validation process, these pins are ensured to be unique within the repository. This prevents ambiguity during connection establishment.

* Certificate Verification: The issuer certificates listed in the metadata are validated to ensure that the algorithms used in the certificates are well-known and secure, and that the certificates are currently valid and have not expired

* Organization: Verification is conducted to ensure the correctness of the organization name in the submitted metadata. Additionally, any other provided organizational information is verified to adhere to the federation policy.

* Tag Validation: Ensures that tags in the metadata adhere to the defined tag structure, verifying both mandatory and optional tags. This process is crucial for maintaining consistency and preventing unauthorized tags within a federation.

The FedTLS metadata repository serves as the vital foundation for establishing trust and enabling secure communication within a FedTLS environment. By providing a central, secure, and controlled repository for critical information, the metadata repository empowers members to confidently discover other trusted entities, and establish secure connections for seamless interaction.

4. Metadata Submission

It is up to the federation to determine which channels should be provided to members for submitting their metadata to the metadata repository. Members typically have the option to either upload the metadata directly to the repository, provided such functionality exists, or to send it to the federation operator through a designated secure channel. If an insecure channel is used, additional measures MUST be taken to verify the authenticity and integrity of the metadata. Such measures may include verifying the checksum of the metadata through another channel. The choice of submission channel may depend on factors such as the federation's guidelines and the preferences of the member.

5. Maintaining Up-to-Date Metadata

In a FedTLS federation, accurate and current metadata is essential for ensuring secure and reliable communication between members. This necessitates maintaining up-to-date metadata accessible by all members.

* Federation Metadata: The federation operator publishes a JWS containing an aggregate of all entity metadata. This JWS serves as the source of truth for information about all members within the federation. Outdated information in the JWS can lead to issues like failed connections, discovery challenges, and potential security risks.

* Local Metadata: Each member maintains a local metadata store containing information about other members within the federation. This information is retrieved from the federation's publicly accessible JWS. Outdated data in the local store can hinder a member's ability to discover and connect with other relevant entities.

Here's how metadata is kept up-to-date:

* Member Responsibility: The primary responsibility for maintaining accurate metadata lies with each member. Members are obligated to:

  - Promptly update their member metadata whenever any relevant information changes and submit it to the metadata repository.

  - Periodically refresh their local metadata store, regardless of whether a caching mechanism is used. This ensures they retrieve the latest information from the federation's JWS, even if they have cached data.

* Federation Operator Role: The Federation Operator plays a crucial role in maintaining data integrity within the federation. Their responsibilities include:

  - Defining clear guidelines for metadata updates, member responsibilities, and expiration time management.

  - Implementing automated mechanisms to update the published JWS containing the aggregate member metadata, ensuring it adheres to the expiration time (exp, see Section 7.4) and cache TTL (cache_ttl, see Section 7.1) specifications.

By adhering to these responsibilities, the Federation ensures that information remains valid for the defined timeframe and that caching mechanisms utilize up-to-date data effectively.

6. Authentication

All communication established within the federation leverages mutual TLS authentication, as defined in [RFC8446]. This mechanism ensures the authenticity of both communicating parties, establishing a robust foundation for secure data exchange.

6.1. Public Key Pinning

To further fortify this trust and mitigate risks associated with fraudulent certificates issued by unauthorized entities, the federation implements public key pinning as specified in [RFC7469]. Public key pinning associates a unique public key with each endpoint within the federation, stored in the federation metadata. During connection establishment, clients and servers validate the received certificate against the pre-configured public key pins retrieved from the federation metadata. This effectively thwarts attempts to utilize fraudulent certificates impersonating legitimate endpoints.

6.2. Pin Discovery and Preloading

Peers in the federation retrieve these unique public key pins, serving as pre-configured trust parameters, from the federation metadata. The federation MUST facilitate the discovery process, enabling peers to identify the relevant pins for each endpoint. Information such as organization, tags, and descriptions within the federation metadata aids in this discovery.

Before initiating any connection, both clients and servers preload the chosen pins in strict adherence to the guidelines outlined in section 2.7 of [RFC7469]. This preloading ensures connections only occur with endpoints possessing matching public keys, effectively blocking attempts to use fraudulent certificates.

6.3. Verification of Received Certificates

Upon connection establishment, both endpoints (client and server) must either leverage public key pinning or validate the received certificate against the published pins. Additionally, the federation metadata contains issuer information, which implementations MAY optionally use to verify certificate issuers. This step remains at the discretion of each individual implementation.

In scenarios where a TLS session terminates independent of the application (e.g., via a reverse proxy), the termination point can utilize optional untrusted TLS client certificate authentication or validate the certificate issuer itself. Depending on the specific implementation, pin validation can then be deferred to the application itself, assuming the peer certificate is appropriately transferred (e.g., via an HTTP header).

6.4. Failure to Validate

It is crucial to note that failure to validate a received certificate against the established parameters, whether through pinning or issuer verification, results in immediate termination of the connection. This strict approach ensures only authorized and secure communication channels are established within the federation.

7. Federation Metadata

Federation metadata is published as a JWS [RFC7515]. The payload contains statements about federation members entities.

Metadata is used for authentication and service discovery. A client select a server based on metadata claims (e.g., organization, tags). The client then use the selected server claims base_uri, pins and if needed issuers to establish a connection.

Upon receiving a connection, a server validates the received client certificate using the client's published pins. Server MAY also check other claims such as organization and tags to determine if the connections is accepted or terminated.

7.1. Federation Metadata claims

This section defines the set of claims that can be included in metadata.

*version (REQUIRED)

Schema version follows semantic versioning (https://semver.org)

*cache_ttl (OPTIONAL)

Specifies the duration (in seconds) for caching the downloaded federation metadata. This enables caching independent of specific HTTP implementations or configurations, beneficial for scenarios where the underlying communication mechanism is not solely HTTP-based.

*Entities (REQUIRED)

List of entities (see Section 7.1.1)

7.1.1. Entities

Metadata contains a list of entities that may be used for communication within the federation. Each entity describes one or more endpoints owned by a member. An entity has the following properties:

* entity_id (REQUIRED)

  A URI that uniquely identifies the entity. This identifier MUST NOT collide with any other entity_id within the federation or with any other federation that the entity interacts with.

  Example: "https://example.com"

* organization (OPTIONAL)

  A name identifying the organization that the entity's metadata represents. The federation operator MUST ensure a mechanism is in place to verify that the organization claim corresponds to the rightful owner of the information exchanged between nodes. This is crucial for the trust model, ensuring certainty about the identities of the involved parties. The federation operator SHOULD choose an approach that best suits the specific needs and trust model of the federation.

  Example: "Example Org".

* issuers (REQUIRED)

  A list of certificate issuers that are allowed to issue certificates for the entity's endpoints. For each issuer, the issuer's root CA certificate is included in the x509certificate property (PEM-encoded).

* servers (OPTIONAL)

  List of the entity's servers (see Section 7.1.1.1).

* clients (OPTIONAL)

  List of the entity's clients (see Section 7.1.1.1).

7.1.1.1. Servers / Clients

A list of the entity's servers and clients.

* description (OPTIONAL)

  A human readable text describing the server or client.

  Example: "SCIM Server 1"

* base_uri (OPTIONAL)

The base URL of the server (hence required for endpoints describing servers).

Example: "https://scim.example.com/"

*pins (REQUIRED)

A list of Public Key Pins [RFC7469]. Each pin has the following properties:

-alg (REQUIRED)

The name of the cryptographic hash algorithm. The only allowed value is "sha256".

Example: "sha256"

-digest (REQUIRED)

The public key of the end-entity certificate converted to a Subject Public Key Information (SPKI) fingerprint, as specified in section 2.4 of [RFC7469]. For clients, the digest MUST be globally unique for unambiguous identification. However, within the same entity_id object, the same digest MAY be assigned to multiple clients.

Example: "+hcmCjJEtLq4BRPhrILyhgn98Lhy6DaWdpmsBAgOLCQ="

*tags (OPTIONAL)

A list of strings that describe the endpoint's capabilities.

Tags are fundamental for discovery within a federation, aiding both servers and clients in identifying appropriate connections.

-Servers: Tags enable servers to identify clients with specific characteristics or capabilities. For instance, a server might want to serve only clients with particular security clearances or those supporting specific protocol versions. By filtering incoming requests based on relevant tags, servers can efficiently identify suitable clients for serving.

-Clients: Tags also assist clients in discovering servers offering the services they require. Clients can search for servers based on tags indicating supported protocols or the type of data they handle. This enables clients to efficiently locate servers meeting their specific needs.

Federation-Specific Considerations

While tags are tied to individual federations and serve distinct purposes within each, several key considerations are crucial to ensure clarity and promote consistent tag usage:

- Well-Defined Scope: Each federation MUST establish a clear scope for its tags, detailing their intended use, allowed tag values, associated meanings, and any relevant restrictions. Maintaining a well-defined and readily accessible registry of approved tags is essential for the federation.

- Validation Mechanisms: Implementing validation mechanisms for tags is highly recommended. This may involve a dedicated operation or service verifying tag validity and compliance with the federation's regulations. Such validation ensures consistency within the federation by preventing the use of unauthorized or irrelevant tags.

Pattern: ^[a-z0-9]{1,64}$

Example: ["scim", "xyzzy"]

7.2. Metadata Schema

The FedTLS metadata schema is defined in Appendix A. This schema specifies the format for describing entities involved in FedTLS and their associated information.

Note: The schema in Appendix A is folded due to line length limitations as specified in [RFC8792].

7.3. Example Metadata

The following is a non-normative example of a metadata statement. Line breaks within the issuers' claim is for readability only.

```
{
  "version": "1.0.0",
  "cache_ttl": 3600,
  "entities": [{
    "entity_id": "https://example.com",
    "organization": "Example Org",
    "issuers": [{
      "x509certificate": "-----BEGIN CERTIFICATE-----\nMIIDDDCCAf
      SgAwIBAgIJAIOsfJBStJQhMA0GCSqGSIb3DQEBCwUAMBsxGTAXBgNV\nBAM
      MEHNjaW0uZXhhbXBsZS5jb20wHhcNMTcwNDA2MDc1MzE3WhcNMTcwNTA2MD
      c1\nMzE3WjAbMRkwFwYDVQQDDBBzY2ltLmV4YW1wbGUuY29tMIIBIjANBgk
      qhkiG9w0B\nAQEFAAOCAQ8AMIIBCgKCAQEAyr+3dXTC8YXoi0LDJTH0lTfv
      8omQivWFOr3+/PBE\n6hmpLSNXK/EZJBD6ZT4Q+tY8dPhyhzT5RFZCVlrDs
      e/kY00F4yoflKiqx9WSuCrq\nZFr1AUtIfGR/LvRUvDFtuHo1MzFttiK8Wr
      wskMYZrw1zLHTIVwBkfMw1qr2XzxFK\njt0CcDmFxNdY5Q8kuBojH9+xt5s
      ZbrJ9AVH/OI8JamSqDjk9ODyGg+GrEZFClP/B\nxa4Fsl04En/9GfaJnCU1
      NpU0cqvWbVUlLOy8DaQMN14HIdkTdmegEsg2LR/XrJkt\nho16diAXrgS25
      3xbkdD3T5d6lHiZCL6UxkBh4ZHRcoftSwIDAQABo1MwUTAdBgNV\nHQ4EFg
      QUs1dXuhGhGc2UNb7ikn3t6cBuU34wHwYDVR0jBBgwFoAUs1dXuhGhGc2U\
      nNb7ikn3t6cBuU34wDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAA
      OCAQEA\nrR9wxPhUa2XfQ0agAC0oC8TFf8wbTYb0ElP5Ej834xMMW/wWTSA
      N8/3WqOWNQJ23\nf0vEeYQwfvbD2fjLvYTyM2tSPOWrtQpKuvulIrxV7Zz8
      A61NIjblE3rfea1eC8my\nTkDOlMKV+wlXXgUxirride+6ubOWRGf92fgze
      DGJWkmm/a9tj0L/3e0xIXeujxC7\nMIt3p99teHjvnZQ7FiIBlvGc1o8FD1
      FKmFYd74s7RxrAusBEAAmBo3xyB89cFU0d\nKB2fkH2lkqiqkyOtjrlHPoy
      6ws6g1S6U/Jx9n0NEeEqCfzXnh9jEpxisSO+fBZER\npCwj2LMNPQxZBqBF
      oxbFPw==\n-----END CERTIFICATE-----"
    }],
    "servers": [{
      "description": "SCIM Server 1",
      "base_uri": "https://scim.example.com/",
      "pins": [{
        "alg": "sha256",
        "digest": "+hcmCjJEtLq4BRPhrILyhgn98Lhy6DaWdpmsBAgOLCQ="
      }],
      "tags": [
        "scim"
      ]
    }],
    "clients": [{
      "description": "SCIM Client 1",
      "pins": [{
        "alg": "sha256",
        "digest": "+hcmCjJEtLq4BRPhrILyhgn98Lhy6DaWdpmsBAgOLCQ="
      }]
    }]
  }]
}
```

7.4. Metadata Signing

The federation metadata is signed with JWS and published using JWS JSON Serialization according to the General JWS JSON Serialization Syntax defined in [RFC7515]. It is RECOMMENDED that federation metadata signatures are created with algorithm *ECDSA using P-256 and SHA-256* ("ES256") as defined in [RFC7518].

The following federation metadata signature protected headers are REQUIRED:

*alg (Algorithm)

Identifies the algorithm used to generate the JWS signature [RFC7515], section 4.1.1.

*iat (Issued At)

Identifies the time on which the signature was issued. Its value MUST be a number containing a NumericDate value [RFC7519], section 4.1.6.

*exp (Expiration Time)

Identifies the expiration time on and after which the signature and federation metadata are no longer valid. The expiration time of the federation metadata MUST be set to the value of exp. Its value MUST be a number containing a NumericDate value [RFC7519], section 4.1.4.

*iss (Issuer)

A URI uniquely identifying the issuing federation, playing a critical role in establishing trust and securing interactions within the FedTLS framework. The iss claim differentiates federations, preventing ambiguity and ensuring entities are recognized within their intended context. Verification of the iss claim, along with the corresponding issuer's certificate, enables relying parties to confidently determine information origin and establish trust with entities within the identified federation. This ensures secure communication and mitigates potential security risks [RFC7519], section 4.1.1.

*kid (Key Identifier)

The key ID is used to identify the signing key in the key set used to sign the JWS [RFC7515], section 4.1.4.

7.5. Example Signature Protected Header

The following is a non-normative example of a signature protected header.

```
{
    "alg": "ES256",
    "exp": 1707739718,
    "iat": 1706875718,
    "iss": "https://fedtls.example.com",
    "kid": "c2fb760e-f4b6-4f7e-b17a-7115d2826d51"
}
```

8. Example Usage Scenarios

The examples in this section are non-normative.

The following example describes a scenario within the federation "Skolfederation" where FedTLS is already established. Both clients and servers are registered members of the federation. In this scenario, clients aim to manage cross-domain user accounts within the service. The standard used for account management is SS 12000:2018 (i.e., a SCIM extension).

```
+----------------------------------------------+
|                                              |
|             Federation Metadata              |
|                                              |
+---+--------------------------+---------------+
    |                          |
   (A)                        (A)
    |                          |
    v                          v
+---+-----+       +------------+--------------+
|Local MD|        |         Local MD          |
+---+-----+       +----+------------- ---+----+
    |                  |                 |
   (B)                (C)               (F)
    |                  |                 |
    v                  v                 v
+---+-----+       +----+---+        +----+---+
|         |       |        |        |        |
| Client  |       | Reverse|        |  App   |
|         +--(D)-->+ Proxy +--(E)-->+        |
|         |       |        |        |        |
|         |       |        |        |        |
+---------+       +--------+        +--------+
```

A. Entities collect member metadata from the federation metadata.

B. The client pins the server's public key pins.

C. The reverse proxy trust anchor is setup with the clients' certificate issuers.

D. The client establishes a connection with the server using the base_uri from the federation metadata.

E. The reverse proxy forwards the client certificate to the application.

F. The application converts the certificate to a public key pin and checks the federation metadata for a matching pin. The entity's entity_id should be used as an identifier.

8.1. Client

A certificate is issued for the client and the issuer is published in the federation metadata together with the client's certificate public key pins

When the client wants to connect to a remote server (identified by an entity identifier) the following steps need to be taken:

1. Find possible server candidates by filtering the remote entity's list of servers based on tags.

2. Connect to the server URI. Include the entity's list of certificate issuers in the TLS clients list of trusted CAs, or trust the listed pins explicitly.

3. If pinning was not used, validate the received server certificate using the entity's published pins.

4. Commence transactions.

8.2. Server

A certificate is issued for the server and the issuer is published in the federation metadata together with the server's name and certificate public key pin.

When the server receives a connection from a remote client, the following steps need to be taken:

1. Populate list of trusted CAs using all known entities' published issuers and required TLS client certificate authentication, or configure optional untrusted TLS client certificate authentication (e.g., optional_no_ca).

2. Once a connection has been accepted, validate the received client certificate using the client's published pins.

3. Commence transactions.

8.3. SPKI Generation

Example of how to use OpenSSL to generate a SPKI fingerprint from a PEM-encoded certificate.

```
openssl x509 -in <certificate.pem> -pubkey -noout | \
openssl pkey -pubin -outform der | \
openssl dgst -sha256 -binary | \
openssl enc -base64
```

8.4. Curl and Public Key Pinning

Example of public key pinning with curl. Line breaks are for readability only.

```
curl --cert client.pem --key client.key --pinnedpubkey 'sha256//0Ok
2aNfcrCNDMhC2uXIdxBFOvMfEVtzlNVUT5pur0Dk=' https://host.example.com
```

9. Security Considerations

9.1. TLS

The security considerations for TLS 1.3 [RFC8446] are detailed in Section 10, along with Appendices C, D, and E of RFC 8446.

9.2. Federation Metadata Updates

Regularly updating the local copy of federation metadata is essential for accessing the latest information about active entities, current public key pins, and valid certificates. The use of outdated metadata may expose systems to security risks, such as interaction with revoked entities or acceptance of manipulated data. If specified in the federation metadata, cache_ttl values SHOULD be respected.

9.3. Verifying the Federation Metadata Signature

Ensuring data integrity and security within the FedTLS framework relies on verifying the signature of downloaded federation metadata. This process confirms the data's origin, validating that it comes from the intended source and has not been altered by unauthorized parties. Through the process of verifying the metadata's authenticity, trust is established in the information it contains, including valid member certificates and public key pins.

10. Acknowledgements

This project was funded through the NGI0 PET Fund, a fund established by NLnet with financial support from the European Commission's Next Generation Internet programme, under the aegis of DG Communications Networks, Content and Technology under grant agreement No 825310.

The authors would like to thank the following people for the detailed review and suggestions:

*Rasmus Larsson

*Mats Dufberg

*Joe Siltberg

*Stefan Norberg

*Petter Blomberg

The authors would also like to thank participants in the EGIL working group for their comments on this specification.

11. IANA Considerations

This document has no IANA actions.

12. Normative References

[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/info/rfc2119>.

[RFC7469] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April 2015, <https://www.rfc-editor.org/info/rfc7469>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[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/info/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/info/rfc8446>.

13. Informative References

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/info/rfc7517>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

[RFC7638] Jones, M. and N. Sakimura, "JSON Web Key (JWK) Thumbprint", RFC 7638, DOI 10.17487/RFC7638, September 2015, <https://www.rfc-editor.org/info/rfc7638>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/info/rfc8792>.

Appendix A. JSON Schema for FedTLS Metadata

This JSON schema defines the format of FedTLS metadata.

Version: 1.0.0

```
=============== NOTE: '\\' line wrapping per RFC 8792 ================

{
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "$id": "https://www.fedtls.se/schema/fedtls-metadata-schema.json\
\",
    "title": "JSON Schema for Federated TLS Authentication",
    "description": "Version: 1.0.0",
    "type": "object",
    "additionalProperties": true,
    "required": [
        "version",
        "entities"
    ],
    "properties": {
        "version": {
            "title": "Metadata schema version",
            "description": "Schema version follows semantic versioni\
\ng (https://semver.org)",
            "type": "string",
            "pattern": "^\\d+\\.\\d+\\.\\d+$",
            "examples": [
                "1.0.0"
            ]
        },
        "cache_ttl": {
            "title": "Metadata cache TTL",
            "description": "How long (in seconds) to cache metadata.\
\ Effective maximum TTL is the minimum of HTTP Expire and TTL",
            "type": "integer",
            "minimum": 0,
            "examples": [
                3600
            ]
        },
        "entities": {
            "type": "array",
            "items": {
                "$ref": "#/components/entity"
            }
        }
    },
    "components": {
        "entity": {
            "type": "object",
            "additionalProperties": true,
            "required": [
                "entity_id",
                "issuers"
```

```
            ],
            "properties": {
                "entity_id": {
                    "title": "Entity identifier",
                    "description": "Globally unique identifier for t\
\he entity.",
                    "type": "string",
                    "format": "uri",
                    "examples": [
                        "https://example.com"
                    ]
                },
                "organization": {
                    "title": "Name of entity organization",
                    "description": "Name identifying the organizatio\
\n that the entity's metadata represents.",
                    "type": "string",
                    "examples": [
                        "Example Org"
                    ]
                },
                "issuers": {
                    "title": "Entity certificate issuers",
                    "description": "A list of certificate issuers th\
\at are allowed to issue certificates for the entity's endpoints. Fo\
\r each issuer, the issuer's root CA certificate is included in the \
\x509certificate property (PEM-encoded).",
                    "type": "array",
                    "items": {
                        "$ref": "#/components/cert_issuers"
                    }
                },
                "servers": {
                    "type": "array",
                    "items": {
                        "$ref": "#/components/endpoint"
                    }
                },
                "clients": {
                    "type": "array",
                    "items": {
                        "$ref": "#/components/endpoint"
                    }
                }
            }
        },
        "endpoint": {
            "type": "object",
            "additionalProperties": true,
```

```
            "required": [
                "pins"
            ],
            "properties": {
                "description": {
                    "title": "Endpoint description",
                    "type": "string",
                    "examples": [
                        "SCIM Server 1"
                    ]
                },
                "tags": {
                    "title": "Endpoint tags",
                    "description": "A list of strings that describe \
\the endpoint's capabilities.",
                    "type": "array",
                    "items": {
                        "type": "string",
                        "pattern": "^[a-z0-9]{1,64}$",
                        "examples": [
                            "xyzzy"
                        ]
                    }
                },
                "base_uri": {
                    "title": "Endpoint base URI",
                    "type": "string",
                    "format": "uri",
                    "examples": [
                        "https://scim.example.com"
                    ]
                },
                "pins": {
                    "title": "Certificate pin set",
                    "type": "array",
                    "items": {
                        "$ref": "#/components/pin_directive"
                    }
                }
            }
        },
        "cert_issuers": {
            "title": "Certificate issuers",
            "type": "object",
            "additionalProperties": false,
            "properties": {
                "x509certificate": {
                    "title": "X.509 Certificate (PEM)",
                    "type": "string"
```

```
                }
            }
        },
        "pin_directive": {
            "title": "RFC 7469 pin directive",
            "type": "object",
            "additionalProperties": false,
            "required": [
                "alg",
                "digest"
            ],
            "properties": {
                "alg": {
                    "title": "Directive name",
                    "type": "string",
                    "enum": [
                        "sha256"
                    ],
                    "examples": [
                        "sha256"
                    ]
                },
                "digest": {
                    "title": "Directive value (Base64)",
                    "type": "string",
                    "pattern": "^(?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+\
\/]{2}==|[A-Za-z0-9+/]{3}=)?$",
                    "examples": [
                        "HiMkrb4phPSP+OvGqmZd6sGvy7AUn4k3XEe8OMBrzt8\
\="
                    ]
                }
            }
        }
    }
}
```

Authors' Addresses

Jakob Schlyter
Kirei AB

Email: jakob@kirei.se

Stefan Halén
The Swedish Internet Foundation

Email: stefan.halen@internetstiftelsen.se