Workgroup: Network Working Group
Internet-Draft: draft-halen-fed-tls-auth-00
Published: 12 October 2020
Intended Status: Informational
Expires: 15 April 2021
Authors: J. Schlyter  S. Halén
Kirei AB  The Swedish Internet Foundation

# Federated TLS Authentication

## Abstract

This document describes how to establish a secure end-to-end channel between two parties within a federation, where both client and server are mutually authenticated. The trust relationship is based upon a trust anchor held and published by the federation. A federation is a trusted third party that inter-connect different trust domains with a common set of policies and standards.

## Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 15 April 2021.

**Table of Contents**


1. Introduction
   1.1. Reserved Words
2. Federation Chain of Trust
3. Authentication
4. Federation Metadata
   4.1. Federation Metadata claims
      4.1.1. Entities
   4.2. Metadata Schema
   4.3. Metadata Signing
   4.4. Metadata Example
5. Usage Examples
   5.1. Client
   5.2. Server
   5.3. SPKI Generation
   5.4. Curl and Public Key Pinning
6. Security Considerations
   6.1. TLS
   6.2. Federation Metadata Updates
   6.3. Federation Metadata Signing Key
7. IANA Considerations
8. Acknowledgements
9. Normative References
10. Informative References

Authors' Addresses


## 1. Introduction

This document describes how to establish a secure end-to-end channel between two parties within a federation, where both client and server are mutually authenticated (TLS [RFC8446]). The trust relationship is based upon a trust anchor held and published by the federation. A federation is a trusted third party that inter-connect different trust domains with a common set of policies and standards. The federation aggregates and publishes information ("federation metadata") about all the federated entities including certificate issuers and public key information. When the term "federation metadata" is used in this document, it always refer to the aggregated information published by a federation in the sense of this document.

The federation provides a common framework for providing endpoint information. When two parties establish a connection, the information of the other endpoint is retrieved from the metadata. The parties leverage this every time they commence a transaction with a new entity in the federation.

Mutual TLS authentication involves provisioning of key material. This key exchange is performed through the publication of the federation metadata by the federation and the use of that federation metadata by its members. Without a federation, the server side is often required to create a public key infrastructure (PKI) in order to distribute client certificates.The Swedish education sector uses federated TLS authentication to secure endpoints used for user lifecycle management . That Federation is a collaboration between school authorities and service providers. If the certificate distribution would be handled without a federated framework, it would mean a higher administrative burden and a higher risk of compromised security.

### 1.1. Reserved Words

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].

## 2. Federation Chain of Trust

The members of a federation submit their issuer certificates and other requested data (in this document called "member metadata") to the federation. Both the authenticity of the submitted member metadata and the submitting member MUST be assured by the federation. How this is achieved is out of scope for this document. The federation operator aggregates, signs and publishes the federation metadata, i.e., an aggregation of all members' member metadata and some additional information added by the federation. By trusting the federation and its certificate, federation members trust the federation metadata content.

The root of the chain of trust is the federation metadata signature and the trust anchor is the federation's signing certificate. That certificate needs to be securely distributed, there MUST be an out-of-band function to verify the certificate. The distribution of the federation's certificate is out-of-scope of this document.

## 3. Authentication

All TLS sessions between clients and servers are authenticated via mutual TLS authentication. Trust is limited to the set of public key pins published for each endpoint in the federation metadata. Public key pinning associates a public key with an endpoint to reduce the risk of attacks with rogue certificates.

Upon connection, the endpoints (client and server) MUST validate the other endpoint's certificate against the published matching public key pin. Issuers in metadata are only used to help validate the server and client certificate. It is up to each implementation to

decide if these are needed. Failure to validate triggers termination of the connection.

If a TLS session is terminated separately from the application (e.g., when using a reverse proxy), the TLS session termination point can validate the certificate issuer and defer public key pin matching to the application given that the peer certificate is transferred to the application (e.g., via a HTTP header).

## 4. Federation Metadata

Federation metadata is published as an JWS [RFC7519]. Entities have an organization claim that is used for identification. Server endpoints include a base URI to connect to the endpoint. Servers and clients also have a list of public key pins used to limit valid endpoint certificates.

Public key pinning is defined in [RFC7469]. Clients and servers preloads pins as defined in [RFC7469], section 2.7

### 4.1. 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)

How long (in seconds) to cache metadata. Effective maximum TTL is the minimum of HTTP Expire and TTL

*Entities (REQUIRED)

List of entities (see Section 4.1.1)

#### 4.1.1. 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)

URI that identifies the entity. It MUST be globally unique.

Example: "https://example.com"

*organization (OPTIONAL)

A name identifying the organization that the entity's metadata represents.

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 4.1.1.1).

*clients (OPTIONAL)

List of the entity's clients (see Section 4.1.1.1).

#### 4.1.1.1. Servers / Clients

A list of the entity's servers and clients.

*description (OPTIONAL)

A human readable text describing the server.

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)

End-entity certificate base64 encoded Subject Public Key Information (SPKI) fingerprint [RFC7469], for client the digest MUST be globally unique. MAY, locally in the same entity_id object, be assigned to multiple clients.

Example: "+hcmCjJEtLq4BRPhrILyhgn98Lhy6DaWdpmsBAgOLCQ="

*tags (OPTIONAL)

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

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

Example: ["scim", "xyzzy"]

### 4.2. Metadata Schema

The federation metadata JSON schema (in YAML format) can be found at https://github.com/dotse/tls-fed-auth.

### 4.3. Metadata Signing

The federation metadata is signed with JWS and published using JWS JSON Serialization. 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.

*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.

*iss (Issuer)

URI that identifies the publisher of federation metadata. The issuer claim MUST be used to prevent conflicts of entities of the same name from different federations.

*kid (Key Identifier)

The key ID is used to identify the signing key in the key set used to sign the JWS.

### 4.4. Metadata Example

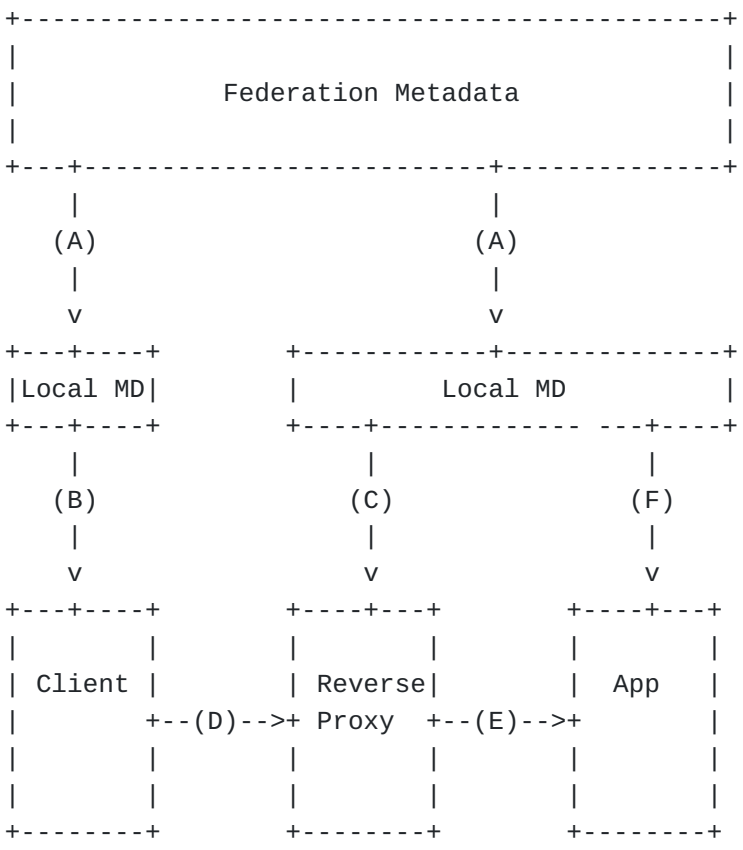
The following is a non-normative example of a federation metadata statement. Line breaks within the issuers' claim is for readability only.

```
{
  "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="
      }]
    }]
  }]
}
```

## 5. Usage Examples

The examples in this section are non-normative.

The example below is from the federation called "Skolfederation" where federated TLS authentication is already in use. Clients and servers are registered in the federation. The clients intend to manage cross-domain user accounts within the service. The standard used for account management is SS 12000:2018 (i.e., a SCIM extension).




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.

### 5.1. 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

### 5.2. 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.

### 5.3. 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
```

### 5.4. Curl and Public Key Pinning

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

```
curl --cert client.pem --client.key --pinnedpubkey 'sha256//0Ok2aNf
crCNDMhC2uXIdxBFOvMfEVtzlNVUT5pur0Dk=' https://host.example.com
```

## 6. Security Considerations

### 6.1. TLS

Security considerations for TLS 1.3 [RFC8446] are described in Section 10 and Appendices C, D and E of RFC 8446.

### 6.2. Federation Metadata Updates

Frequent check of the federation metadata aggregate to use the most recent version is required. It is required to check the federation metadata periodically to find out if entities, pins and issuers are still active.

### 6.3. Federation Metadata Signing Key

To ensure the validity of the federation metadata the refresh process must verify the signature on each and every federation metadata fetch. The federation's public key authenticity must be assured and verified in a secure way.

## 7. IANA Considerations

This document has no IANA actions.

## 8. Acknowledgements

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.

## 9. 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>.

## 10. Informative References

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

## Authors' Addresses

Jakob Schlyter

Kirei AB

Email: jakob@kirei.se

Stefan Halén
The Swedish Internet Foundation

Email: stefan.halen@internetstiftelsen.se