Network Working Group
Internet-Draft
Intended status: Informational
Expires: 28 February 2026

S. Halen
The Swedish Internet Foundation
J. Schlyter
Kirei AB

27 August 2025

Mutually Authenticating TLS in the context of Federations
draft-halen-fedae-03

Abstract

This informational independent submission to the RFC series describes a means to use TLS 1.3 to perform machine-to-machine mutual authentication within federations. This memo is not a standard. It does not modify the TLS protocol in any way, nor does it require changes to common TLS libraries. TLS is specified and standardized by the IETF's TLS working group.

The framework enables interoperable trust management for federated machine-to-machine communication. It introduces a centrally managed trust anchor and a controlled metadata publication process, ensuring that only authorized members are identifiable within the federation. These mechanisms support unambiguous entity identification and reduce the risk of impersonation, promoting secure and policy-aligned interaction across organizational boundaries.

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 28 February 2026.

Table of Contents

1. Introduction 3
1.1. Reserved Words 3
1.2. Terminology 4
2. Diverse Design Patterns 4
3. Trust Model 5
3.1. Role of the Federation Operator 5
3.2. Federation Members' Responsibilities 6
3.3. Chain of Trust 6
3.4. Member Vetting 7
3.5. Metadata Authenticity 8
4. Metadata Repository 8
4.1. Metadata Submission 9
4.2. Maintaining Up-to-Date Metadata 9
5. Authentication 10
5.1. Public Key Pinning 11
5.1.1. Benefits of Public Key Pinning 11
5.2. Pin Discovery and Preloading 12
5.3. Verification of Received Certificates 12
5.4. Failure to Validate 13
5.5. Certificate Rotation: 13
5.6. Implementation Guidelines 13
6. Federation Metadata 14
6.1. Federation Metadata Claims 14
6.1.1. Entities 16
6.2. Metadata Schema 20
6.3. Example Metadata 20
6.4. Metadata Signing 22
6.5. Example Signature Protected Header 22
7. Example Usage Scenarios 22
7.1. Client 23
7.2. Server 24
7.3. SPKI Generation 24
7.4. Curl and Public Key Pinning 25
8. Deployments of the MATF Framework 25
8.1. Skolfederation Moa 25
8.2. Swedish National Agency for Education 26
8.3. Sambruk's EGIL 26
9. Security Considerations 26
9.1. Security Risks and Trust Management 26
9.2. TLS 26

9.3. Federation Metadata Updates 27
9.4. Verifying the Federation Metadata Signature 27
9.5. Time Synchronization 27
10. Acknowledgements 27
11. IANA Considerations 28
12. Normative References 28
13. Informative References 28
Appendix A. JSON Schema for MATF Metadata 29
Authors' Addresses 34

1. Introduction

This document describes the Mutually Authenticating TLS in the context of Federations (MATF) framework, developed to complement multilateral SAML federations within the education sector. These federations often rely on just-in-time provisioning, where user accounts are created at first login based on information from the SAML assertion. However, educators need to be able to manage resources and classes before students access the service. MATF bridges this gap by using secure machine-to-machine communication, enabling pre-provisioning of user information using a trust model and metadata structure inspired by SAML federations.

MATF is designed specifically for secure authentication in machine-to-machine contexts, such as RESTful APIs and service-to-service interactions, and is not intended for browser-based authentication. Because its applicability in a browser environment has not been studied, using MATF within browsers is not recommended. Doing so may introduce risks that differ from those typically addressed by standard browser security models.

This work is not a product of the IETF, does not represent a standard, and has not achieved community consensus. It aims to address specific federation challenges and provide a framework for secure communication.

TLS is specified by the IETF TLS Working Group. TLS 1.3 is defined in [RFC8446]. Additional information about the TLS Working Group is available at https://datatracker.ietf.org/wg/tls/about/ (https://datatracker.ietf.org/wg/tls/about/).

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 MATF for secure communication.

* Federation Member: An entity that has been approved to join the federation and can leverage MATF 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.

* Federation Metadata: A cryptographically signed document containing 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.

* 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. Diverse Design Patterns

MATF is designed to be flexible and adaptable to the varying needs of different federations. Federations can differ significantly in terms of size, scope, and security requirements, which makes it challenging to prescribe a one-size-fits-all trust framework and security measures.

For instance, in the European Union, the eIDAS (electronic Identification, Authentication, and trust Services) regulation establishes a framework for electronic identification and trust

services for electronic transactions within the EU. This regulation provides a comprehensive set of standards for secure electronic interactions across member states. National federations within EU member states adhere to these standards, ensuring interoperability and mutual recognition of electronic IDs across different countries.

Similarly, national federations, such as those found in education or healthcare sectors, often have their own specific trust frameworks and security measures tailored to their unique needs. These federations may leverage existing national identification systems or other trusted credentials to establish member identities and ensure secure interactions.

Organizations may also set up their own federations, tailored to the specific security requirements and trust models relevant to their context. For example, a private business federation might establish its own vetting processes and trust framework based on the nature of its business and the sensitivity of the data being exchanged.

By allowing federations the flexibility to tailor their trust frameworks and security measures, MATF can support a wide range of use cases. This flexibility is crucial for accommodating the diverse requirements and challenges faced by different federations, ensuring a secure and adaptable system for establishing trust and facilitating secure communication.

3. Trust Model

The MATF framework operates on a trust model that is central to its design and functionality. This section outlines the key components of this trust model and its implications for federation members and the federation operator.

3.1. Role of the Federation Operator

The federation operator plays a critical role in the MATF framework. This entity is responsible for:

* Managing the central trust anchor, which is used to establish trust across different domains within the federation.

* Vetting federation members to ensure they meet the required standards and policies.

* Maintaining and securing the federation metadata, which includes public key pins [RFC7469], issuer certificates, and other essential information.

Additionally, the federation operator SHOULD develop their own threat models to proactively identify potential risks and threats. This process involves examining the operating environment, evaluating both internal and external threats, and understanding how vulnerabilities can be exploited. The goal of the threat model is to enable the federation operator to establish mitigation strategies that address the identified risks.

The security and stability of the federation rely on the integrity and competence of the federation operator. Members must be able to fully trust this central authority, as its role is essential to maintaining the federation's reliability and security.

3.2. Federation Members' Responsibilities

Federation members share the responsibility of maintaining trust and security within the federation. Their responsibilities include:

* Adhering to the federation's security policies and procedures.

* Ensuring the accuracy and timeliness of their metadata submissions.

* Cooperating with the federation operator's vetting and security measures.

By fulfilling these responsibilities, federation members help sustain the overall trust framework that enables secure and reliable communication within the federation. 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.

3.3. Chain of Trust

Each federation operates within a trust framework that encompasses its own security policies and procedures. This framework is designed to ensure the integrity, authenticity, and confidentiality of communications within the federation. Key components of this framework include:

* Public key pinning [RFC7469] and preloading to thwart man-in-the-middle attacks by ensuring validated certificates.

* Regular updates and verification of federation metadata to prevent the use of outdated or compromised information.

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.

This trust framework is essential for enabling seamless and secure interoperability across different trust domains within the federation.

3.4. Member Vetting

To ensure the security and integrity of the MATF framework, a member vetting process is essential. Detailed vetting processes are beyond the scope of this document but can be guided by established frameworks such as eIDAS and eduGAIN.

The following are non-normative references to established frameworks:

* eIDAS: The eIDAS regulation establishes a framework for electronic identification and trust services within the European Union. It ensures secure and standardized electronic interactions across member states, facilitating mutual recognition of electronic IDs. Operators can refer to the eIDAS framework for guidance on robust authentication and identity verification processes [eIDAS].

* eduGAIN: eduGAIN is an interfederation service connecting identity federations worldwide, primarily within the research and education sectors. It ensures high standards of security and

interoperability, allowing institutions to collaborate seamlessly. eduGAIN's processes for vetting, as described in [eduGAIN], can serve as a useful reference.

3.5. Metadata Authenticity

Ensuring the authenticity of metadata is crucial for maintaining the security and trustworthiness of the MATF framework. The specific mechanisms for ensuring metadata authenticity are beyond the scope of this document and must be defined by the federation or regulatory bodies.

4. Metadata Repository

The MATF metadata repository 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 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, the submitted metadata MUST undergo a validation process. This process aims to verify the accuracy, completeness, and validity of the information provided by a member. The validation process MUST include, at a minimum but not limited to, the following checks:

* 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 [RFC7469] are used to identify client entities within the federation metadata during the connection validation process. When a server validates a client's TLS connection, it extracts the pin from the client's TLS certificate and matches it against entries in the federation metadata. The requirements for pin uniqueness and usage are detailed in Section 6.1.1.1.

* 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

* Tag Validation: Ensures that tags, as defined in Section 6.1.1.1 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 MATF metadata repository serves as the vital foundation for establishing trust and enabling secure communication within a MATF 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.

4.1. 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.

4.2. Maintaining Up-to-Date Metadata

In a MATF 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.

The following outlines the procedures for keeping metadata up-to-date:

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

  - Defining regulations for metadata management that MUST include, at a minimum but not limited to, expiration and cache time management.

  - Implementing mechanisms to update the published federation metadata, ensuring it adheres to the expiration time (exp as defined in Section 6.4) and cache TTL (cache_ttl as defined in Section 6.1) specifications.

* Member Responsibility: Members must follow the federation's metadata management regulations and refresh their local metadata store according to the defined expiration and cache regulations.

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.

5. 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.

5.1. Public Key Pinning

MATF implements public key pinning as specified in [RFC7469]. Public key pinning associates one or more unique public keys with each endpoint within the federation, stored in the federation metadata. During a connection, clients and servers extract the public key from the received certificate and validate it against the pre-configured public key pins retrieved from the federation metadata.

5.1.1. Benefits of Public Key Pinning

The decision to utilize public key pinning in the MATF framework was driven by several critical factors aimed at enhancing security and ensuring trust:

5.1.1.1. Interfederation Trust

In interfederation environments, where multiple federations need to trust each other, public key pinning remains effective. Each federation can pin the public keys of entities in other federations, ensuring trust across boundaries. Unlike private certificate chains, which can become complex and difficult to manage across multiple federations, public key pinning provides a straightforward mechanism for establishing trust. MATF interfederation addresses this challenge by aggregating metadata from all participating federations into a unified metadata repository. This shared metadata enables secure communication between entities in different federations, ensuring consistent key validation and robust cross-federation trust and security.

5.1.1.2. Fortifying Security Against Threats

Public key pinning provides a robust defense mechanism by directly binding a peer to a specific public key. This ensures that only the designated key is trusted, preventing attackers from exploiting fraudulent certificates. By eliminating reliance on external trust intermediaries, this approach significantly enhances resilience against potential threats.

5.1.1.3. Use of Self-Signed Certificates

The use of self-signed certificates within the federation leverages public key pinning to establish trust. By bypassing external CAs, servers and clients rely on the federation’s mechanisms to validate trust. Public key pinning ensures that only the specific self-signed public keys, identified by key pins in the metadata, are trusted.

5.1.1.4. Revocation

If any certificate in a certificate chain is compromised, the revocation process can be complex and slow. This complexity arises because not only the compromised certificate but potentially multiple certificates within the chain might need to be revoked and reissued. Public key pinning mitigates this complexity by allowing clients to explicitly trust a specific public key, thereby reducing dependency on the entire certificate chain's integrity.

If a leaf certificate is compromised within a MATF federation, the revocation process involves removing the pin associated with the compromised certificate and publishing updated metadata that includes a new pin corresponding to the replacement certificate. This approach eliminates reliance on traditional certificate revocation mechanisms and shifts the trust relationship to the specific, updated public key identified by its pin.

5.2. 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, allowing peers to identify the relevant pins for each endpoint. Information such as organization, tags, and descriptions within the federation metadata supports this discovery.

Before initiating any connection, clients and servers MUST preload the designated pins from the federation metadata. This aligns with the principle described in Section 2.7 of [RFC7469], which introduces optional sources for pinning information, with the federation metadata serving as one such source. Preloading pins restricts connections to endpoints with matching public keys, mitigating the risks posed by fraudulent certificates.

5.3. 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).

5.4. Failure to Validate

A received certificate that fails validation MUST result in the immediate termination of the connection. This strict enforcement ensures that only authorized and secure communication channels are established within the federation.

5.5. Certificate Rotation:

To replace a certificate, whether due to expiration or other reasons, the following procedure must be followed:

1. Publishing New Metadata: When a certificate needs to be changed, federation members publish new metadata containing the pin (SHA256 thumbprint) of the new public key. This ensures that the new pin is available to all federation members.

2. Propagation Period: Allow time for the updated metadata to propagate throughout the federation before switching to the new certificate. This overlap period ensures that all nodes recognize the new pin and avoid connection issues.

3. Switching to the New Certificate: After ensuring the new metadata has propagated, members switch to the new certificate in their TLS stack.

4. Removing Old Pin: After successfully switching to the new certificate, members must publish updated metadata that excludes the old pin. This final step ensures that only the current public keys are trusted.

5.6. Implementation Guidelines

Public key validation MUST always be enforced, either through direct pinning or by deferring validation to the application.

For clients, public key validation typically occurs within the application handling the TLS session, either by enforcing direct pinning or by extracting and validating the public key against the published pins.

For servers, validation depends on deployment. If the application terminates the TLS session, it performs direct pinning or extracts and validates the public key. If a reverse proxy terminates the TLS session, it can enforce direct pinning or forward the certificate to the application (e.g., via an HTTP header) for validation.

Implementations SHOULD, when possible, rely on libraries with native support for pinning. Libcurl, for example, supports pinning via the PINNEDPUBLICKEY option. In Python, the cryptography library can extract public keys, while the requests package together with urllib3 can intercept certificates. Go provides crypto/tls and crypto/x509 for certificate inspection and public key extraction. In Java, java.security.cert.X509Certificate enables public key extraction, while java.net.http.HttpClient allows pinning enforcement using a custom SSLContext and TrustManager. The choice of library is left to the discretion of each implementation.

If bypassing standard CA validation is possible, it SHOULD be done. If not, the issuers listed in the federation metadata MUST be used as the trust store to validate certificate issuers while still enforcing key pinning. Without issuer validation against issuers in metadata, self-signed certificates would not be accepted. These mechanisms ensure compatibility with existing TLS infrastructure while maintaining strict security guarantees.

6. 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 selects a server based on metadata claims (e.g., organization, tags). The client then uses 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 connection is accepted or terminated.

6.1. Federation Metadata Claims

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

* iat (REQUIRED)

  Identifies the time at which the federation metadata was issued.

- Data Type: Integer

- Syntax: NumericDate as defined in [RFC7519], Section 4.1.6

- Example: 1755514949

* exp (REQUIRED)

  Identifies the expiration time on or after which the federation metadata is no longer valid. Once the exp time has passed, the metadata MUST be rejected regardless of cache state.

  - Data Type: Integer

  - Syntax: NumericDate as defined in [RFC7519], Section 4.1.4

  - Example: 1756119888

* iss (REQUIRED)

  A URI uniquely identifying the issuing federation. This value differentiates federations, prevents ambiguity, and ensures that entities are recognized within their intended context. Verification of the iss claim enables recipients to determine the origin of the information and to establish trust with entities within the identified federation.

  - Data Type: String

  - Syntax: URI, as defined in [RFC7519], Section 4.1.1

  - Example: "https://federation.example.org"

* version (REQUIRED)

  Indicates the schema version of the federation metadata. This ensures compatibility between members of the federation by defining a clear versioning mechanism for interpreting metadata.

  - Data Type: String

  - Syntax: Must adhere to Semantic Versioning (https://semver.org (https://semver.org)).

  - Example: "1.0.0"

* cache_ttl (OPTIONAL)

Specifies the duration in seconds for caching downloaded federation metadata, allowing for independent caching outside of specific HTTP configurations, particularly useful when the communication mechanism isn't HTTP-based. In the event of a metadata publication outage, members can rely on cached metadata until it expires, as indicated by the exp claim in the JWS payload, defined in Section 6.4. Once expired, metadata MUST no longer be trusted. If omitted, a mechanism to refresh metadata MUST still exist to ensure the metadata remains valid.

- Data Type: Integer
- Syntax: Integer representing the duration in seconds.
- Example: 3600

* entities (REQUIRED)

  Contains the list of entities within the federation.

  - Data Type: Array of Objects
  - Syntax: Each object MUST conform to the entity definition, as specified in Section 6.1.1.

6.1.1. 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 within any other federation that the entity interacts with.

  - Data Type: URI
  - Syntax: A valid URI.
  - 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.

- Data Type: String

- Syntax: A name identifying the organization represented by the entity.

- Example: "Example Org"

* issuers (REQUIRED)

A list of certificate issuers allowed to issue certificates for the entity's endpoints. For each issuer, the issuer's root CA certificate MUST be included in the x509certificate property, PEM-encoded. Certificate verification relies on public key pinning, with the list of allowed issuers used only when a certificate chain validation mechanism is unavoidable. For self-signed certificates, the certificate itself acts as its own issuer and MUST be listed as such in the metadata.

- Data Type: List of Objects

- Syntax: Each object contains a issuer certificate, PEM-encoded.

- Example: Issuer truncated for readability.

```
"issuers": [{
  "x509certificate": "-----BEGIN CERTIFICATE-----\nMIIDDD"
}]
```

* servers (OPTIONAL)

Contains the list of servers within the entity.

- Data Type: Array of Objects

- Syntax: Each object MUST conform to the server definition, as specified in Section 6.1.1.1.

* clients (OPTIONAL)

Contains the list of clients within the entity.

- Data Type: Array of Objects

- Syntax: Each object MUST conform to the client definition, as specified in Section 6.1.1.1.

6.1.1.1. Servers / Clients

A list of the entity’s servers and clients.

* description (OPTIONAL)

  A human readable text describing the server or client.

  - Data Type: String

  - Syntax: Free-form text describing the server or client.

  - Example: "SCIM Server 1"

* base_uri (OPTIONAL)

  The base URL of the server, which is required for endpoints that describe server.

  - Data Type: URI

  - Syntax: A valid URL.

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

* pins (REQUIRED)

  A list of objects representing Public Key Pins [RFC7469].

  - Data Type: Array of Objects

  - Syntax: A list of objects, where each object represents a single public key pin with the following properties:

    o alg (REQUIRED)

      The name of the cryptographic hash algorithm. Currently, the RECOMMENDED value is ’sha256’. As more secure algorithms are developed over time, federations should be ready to adopt these newer options for enhanced security.

+ Data Type: String

+ Syntax: The name of the algorithm.

+ Example: "sha256"

o 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.

+ Data Type: String

+ Syntax: SPKI fingerprint.

+ Example: "+hcmCjJEtLq4BRPhrILyhgn98Lhy6DaWdpmsBAgOLCQ="

- Example:

```
"pins": [{
  "alg": "sha256",
  "digest": "+hcmCjJEtLq4BRPhrILyhgn98Lhy6DaWdpmsBAgOLCQ="
}]
```

* tags (OPTIONAL)

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

- Data Type: Array of Strings

- Syntax: Strings describing endpoint capabilities.

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

- Example: ["scim", "xyzzy"]

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

- Server Tags: Tags associated with servers are used by clients to discover servers offering the services they require. Clients can search for servers based on tags that indicate supported protocols or the type of data they handle, enabling discovery of compatible servers.

- Client Tags: Tags associated with clients are used by servers to identify clients with specific characteristics or capabilities. For instance, a server might only accept connections from clients that support particular protocols. By filtering incoming requests based on these tags, servers can identify suitable clients.

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.

6.2. Metadata Schema

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

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

6.3. Example Metadata

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

```
{
  "exp": 1755514949,
  "iat": 1756119888,
  "iss": "https://federation.example.org",
  "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="
        }]
      }]
    }]
  }
```

6.4. Metadata Signing

Federation metadata is signed using JWS and published using JWS JSON Serialization according to the General JWS JSON Serialization Syntax defined in [RFC7515]. Federation metadata signatures are RECOMMENDED to be created using the algorithm _ECDSA using P-256 and SHA-256_ ("ES256") as defined in [RFC7518]. However, to accommodate evolving cryptographic standards, alternative algorithms MAY be used, provided they meet the security requirements of the federation.

The following protected JWS header parameters are REQUIRED:

* alg (Algorithm)

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

* kid (Key Identifier)

  Identifies the signing key in the key set used to sign the JWS [RFC7515], Section 4.1.4.

6.5. Example Signature Protected Header

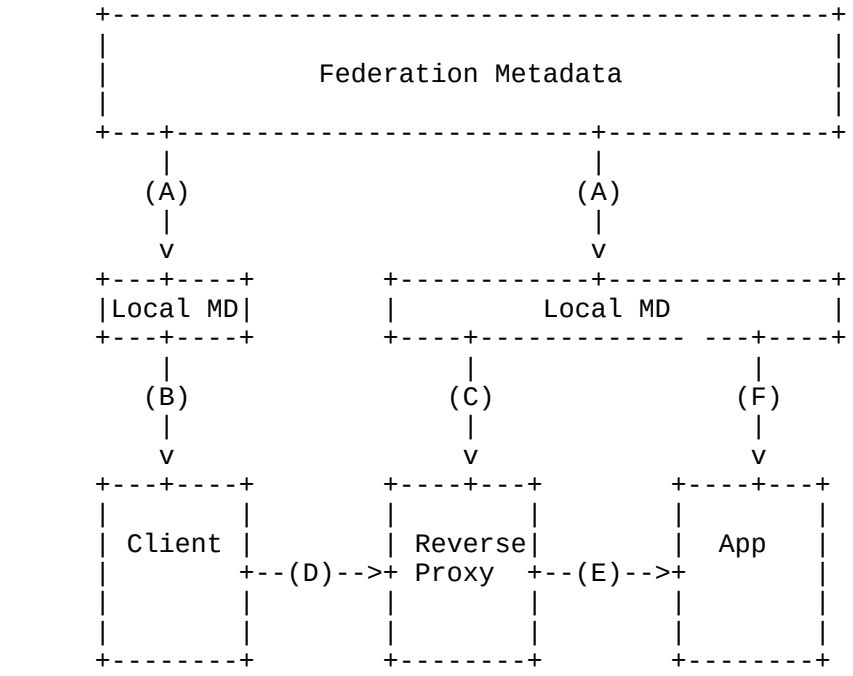
The following is a non-normative example of a signature protected header.

```
{
    "alg": "ES256",
    "kid": "c2fb760e-f4b6-4f7e-b17a-7115d2826d51"
}
```

7. Example Usage Scenarios

The examples in this section are non-normative.

The following example describes a scenario within the federation "Skolfederation" where MATF 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).



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.

7.1. 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 is not used during the TLS handshake, the client MUST perform a post-connection validation against the entity’s published pins.

4. Commence transactions.

7.2. 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.

7.3. 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
```

7.4. 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
```

8. Deployments of the MATF Framework

The MATF framework has proven its practical value and robustness through successful deployments in several environments.

8.1. Skolfederation Moa

Skolfederation Moa [Moa], is a federation designed to secure communication between digital educational resources and schools. MATF is developed to meet Moa's needs and enables secure data exchange for schools, municipalities, educational platforms, and services across Sweden.

The community plays a crucial role in this type of federation. Members are active participants, and the FO ensures the federation runs smoothly and serves their needs. Moa's success highlights the importance of collaboration, with members and the FO working together to maintain trust, security, and interoperability in the education sector.

The deployment of MATF in the Swedish education sector has provided several key insights. Maintaining an accurate registry of metadata ownership with reliable contact information is essential for troubleshooting and ensuring accountability. The deployment also demonstrated the importance of setting reasonable expiration times for metadata. Too short an expiration can hinder the ability to implement contingency plans for publishing new metadata during outages.

Metadata validation is necessary to maintain a stable federation. While manual validation may be sufficient in the early stages of a federation, it becomes unmanageable as the federation scales. Without an automated validation process, incorrect metadata uploaded by members is likely to go undetected, leading to publication of incorrect metadata.

The signing key is needed to sign metadata. Under fallback scenarios, even if metadata can be retrieved from elsewhere, without access to the signing key, it is impossible to publish metadata. Therefore, secure and redundant management of the signing key is

crucial to enable fallback mechanisms and ensure reliable signing and distribution of metadata. If metadata is retrieved from a location other than the official repository, it is mandatory to validate its signature to maintain trust and ensure the authenticity of the metadata.

8.2. Swedish National Agency for Education

The Swedish National Agency for Education [SkolverketMATF] leverages MATF within its digital national test platform to establish a robust authentication mechanism. The platform utilizes an API for client verification prior to secure data transfer to the agency's test service, ensuring the integrity and confidentiality of educational data.

8.3. Sambruk's EGIL

Sambruk's EGIL [EGIL], a platform providing digital services to municipalities, has successfully integrated the MATF framework. This deployment demonstrates the framework's adaptability to support a wide range of digital service infrastructures.

These deployments highlight the effectiveness of the MATF framework in enhancing security and interoperability within the educational sector.

9. Security Considerations

9.1. Security Risks and Trust Management

The security risks associated with the MATF framework are confined to each individual federation. Both the federation operator and federation members share the responsibility of maintaining trust and security within the federation. Proper handling and management of metadata, as well as thorough vetting of federation members, are crucial to sustaining this trust and security. Each federation operates within a trust framework, which includes its own security policies and procedures to ensure the integrity and reliability of the federation.

9.2. TLS

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

9.3. 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 [RFC7469], and valid issuer certificates. The use of outdated metadata may expose systems to security risks, such as interaction with revoked entities or acceptance of manipulated data.

9.4. Verifying the Federation Metadata Signature

Ensuring data integrity and security within the MATF framework relies on verifying the signature of downloaded federation metadata. This verification process confirms the data's origin, ensuring it comes from the intended source and has not been altered by unauthorized parties. By establishing the authenticity of the metadata, trust is maintained in the information it contains, including valid member public key pins and issuer certificates. To achieve a robust implementation, it is crucial to consider the security aspects outlined in [RFC7515], which describes security considerations related to algorithm selection, key compromise, and signature integrity.

9.5. Time Synchronization

Maintaining synchronized clocks across all federation members is critical for the security of the MATF framework. Inaccurate timestamps can compromise the validity of digital signatures and certificates, hinder reliable log analysis, and potentially expose the system to time-based attacks. Therefore, all federation members MUST employ methods to ensure their system clocks are synchronized with a reliable time source.

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 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>.

[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>.

[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>.

[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>.

[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

[EGIL] Sambruk, "EGIL - manage your school’s digital user accounts efficiently", 2022, <https://sambruk.se/egil-dnp/>.

[Moa] The Swedish Internet Foundation, "Machine and Organization Authentication", 2022, <https://wiki.federationer.internetstiftelsen.se/x/LYA5AQ>.

[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>.

[SkolverketMATF] Swedish National Agency for Education, "Authentication API for User Management", 2023, <https://github.com/skolverket/dnp-usermanagement/blob/main/authentication-api/README.md>.

[eIDAS] European Commission, "eIDAS: electronic Identification, Authentication and trust Services", 2014, <https://eidas.ec.europa.eu/>.

[eduGAIN] GEANT Association, "eduGAIN: Interfederation service connecting research and education identity federations worldwide", 2023, <https://edugain.org>.

Appendix A. JSON Schema for MATF Metadata

The following JSON Schema defines the structure of MATF metadata. It conforms to draft 2020-12 of the JSON Schema standard.

Version: 1.0.0

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

{
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "$id": "https://mtlsfed.se/schema/matf-metadata-schema.json",
    "title": "JSON Schema for Mutually Authenticating TLS in the con\
\text of Federations",
    "description": "Version: 1.0.0",
    "type": "object",
    "additionalProperties": true,
    "required": [
        "iat",
        "exp",
```

```
            "iss",
            "version",
            "entities"
        ],
        "properties": {
            "iat": {
                "title": "Issued at",
                "description": "Time at which the metadata was issued (U\
   \NIX timestamp)",
                "type": "integer",
                "minimum": 0,
                "examples": [
                    1755514949
                ]
            },
            "exp": {
                "title": "Expiration time",
                "description": "Time at which the metadata expires (UNIX\
   \ timestamp)",
                "type": "integer",
                "minimum": 0,
                "examples": [
                    1756119888
                ]
            },
            "iss": {
                "title": "The federation issuing the metadata",
                "description": "A URI that uniquely identifies the feder\
   \ation that issued the metadata",
                "type": "string",
                "format": "uri",
                "minLength": 1,
                "examples": [
                    "https://example.com/federation"
                ]
            },
            "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. T\
   \he effective maximum is bounded by the exp claim.",
            "type": "integer",
            "minimum": 0,
            "examples": [
                3600
            ]
        },
        "entities": {
            "type": "array",
            "minItems": 1,
            "items": {
                "$ref": "#/$defs/entity"
            }
        }
    },
    "$defs": {
        "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",
                        "minItems": 1,
                        "items": {
                            "$ref": "#/$defs/cert_issuers"
                        }
                    },
                    "servers": {
                        "type": "array",
                        "items": {
                            "$ref": "#/$defs/endpoint"
                        }
                    },
                    "clients": {
                        "type": "array",
                        "items": {
                            "$ref": "#/$defs/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",
                        "minItems": 1,
                        "items": {
                            "$ref": "#/$defs/pin_directive"
                        }
                    }
                }
            },
            "cert_issuers": {
                "title": "Certificate issuers",
                "type": "object",
                "additionalProperties": false,
                "required": [
                    "x509certificate"
                ],
                "properties": {
                    "x509certificate": {
                        "title": "X.509 Certificate (PEM)",
                        "type": "string",
                        "pattern": "^-----BEGIN CERTIFICATE-----(?:\\r?\\
   \\n)(?:[A-Za-z0-9+/=]{64}\\r?\\n)*(?:[A-Za-z0-9+/=]{1,64}\\r?\\n)---\
   \--END CERTIFICATE-----(?:\\r?\\n)?$"
                    }
                }
            },
            "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+/]{43}=$",
                        "examples": [
                            "HiMkrb4phPSP+OvGqmZd6sGvy7AUn4k3XEe8OMBrzt8\
   \="
                        ]
                    }
                }
            }
        }
   }
```

Authors' Addresses

Stefan Halen
The Swedish Internet Foundation
Email: stefan.halen@internetstiftelsen.se

Jakob Schlyter
Kirei AB
Email: jakob@kirei.se