OpenID Federations Spec Overview

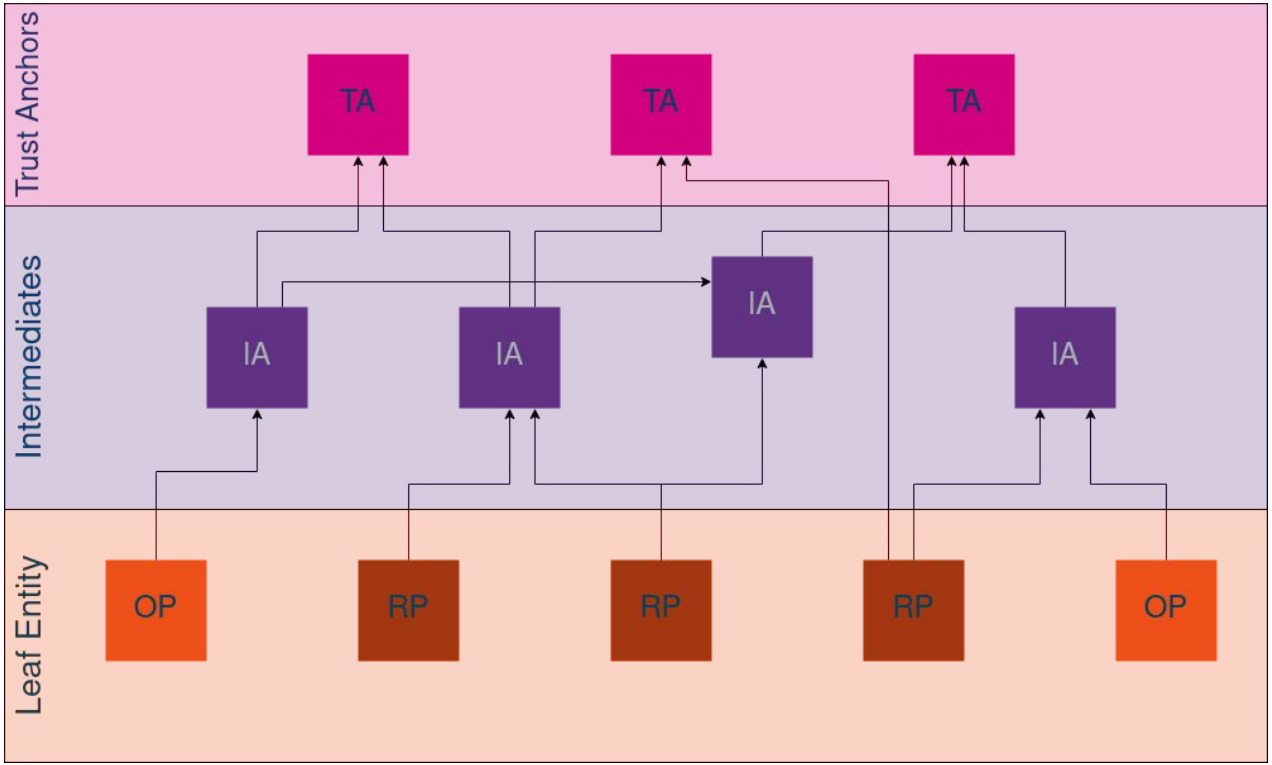
Gabriel Zachmann, KIT
Diana Gudu, KIT

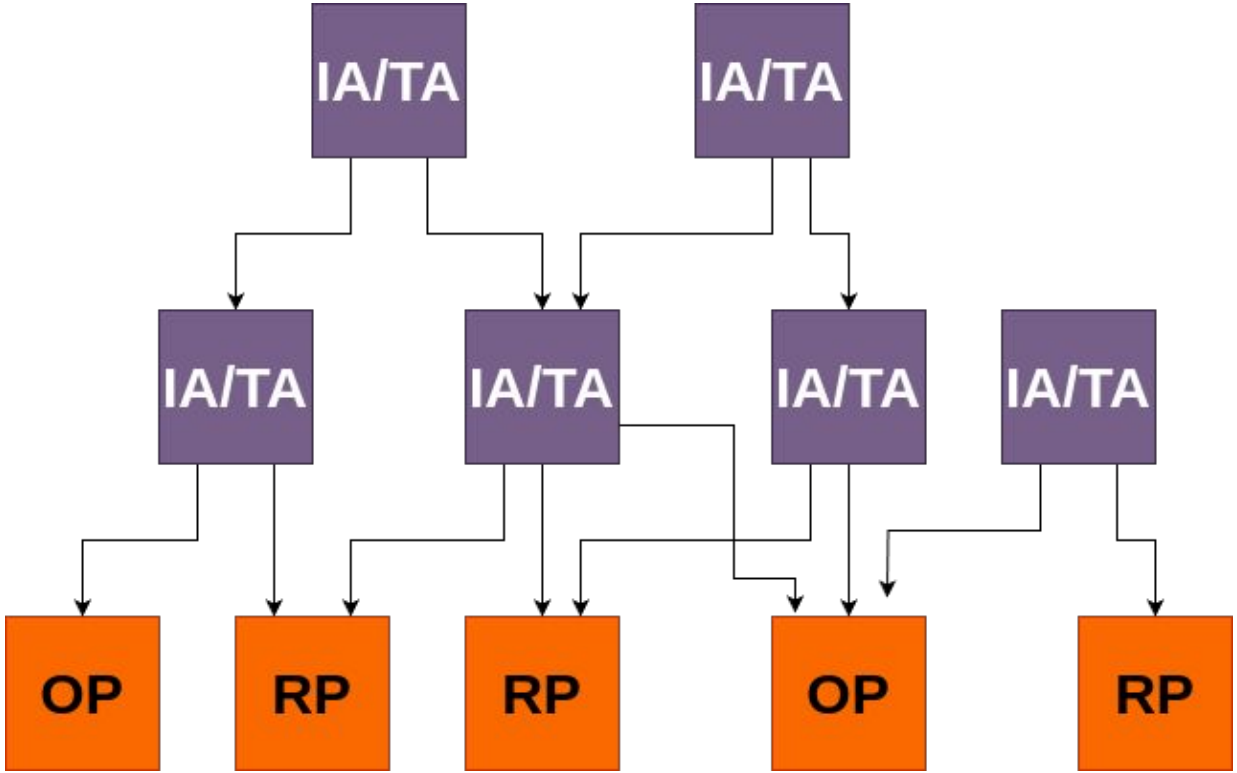
2025-03-12

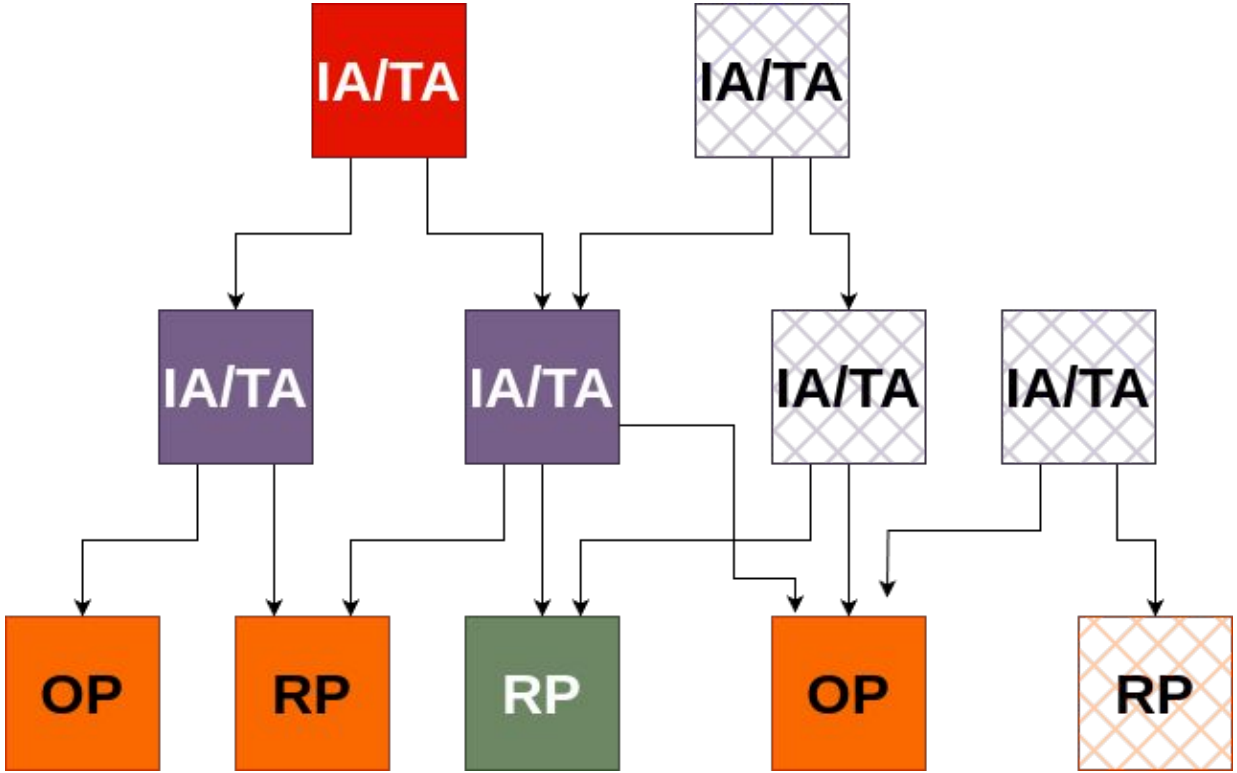
- OpenID Federation Specification
- Authlete (Comm Java Impl) Docu with graphs
  - (some nice graphs, but might not be completely up2date)

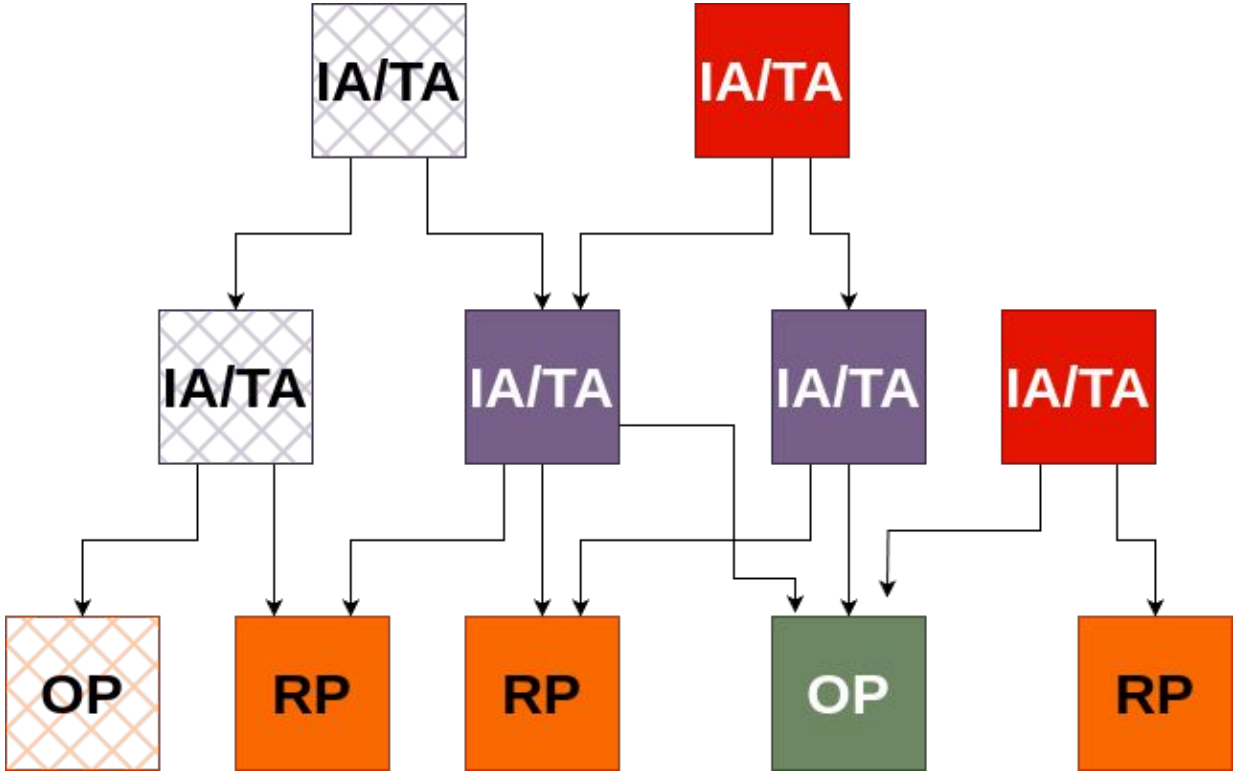
What Is OpenID Federations (oidfed)?

- Trust Framework
  - To build the federation
- In principle protocol agnostic
- Designed with OIDC in mind

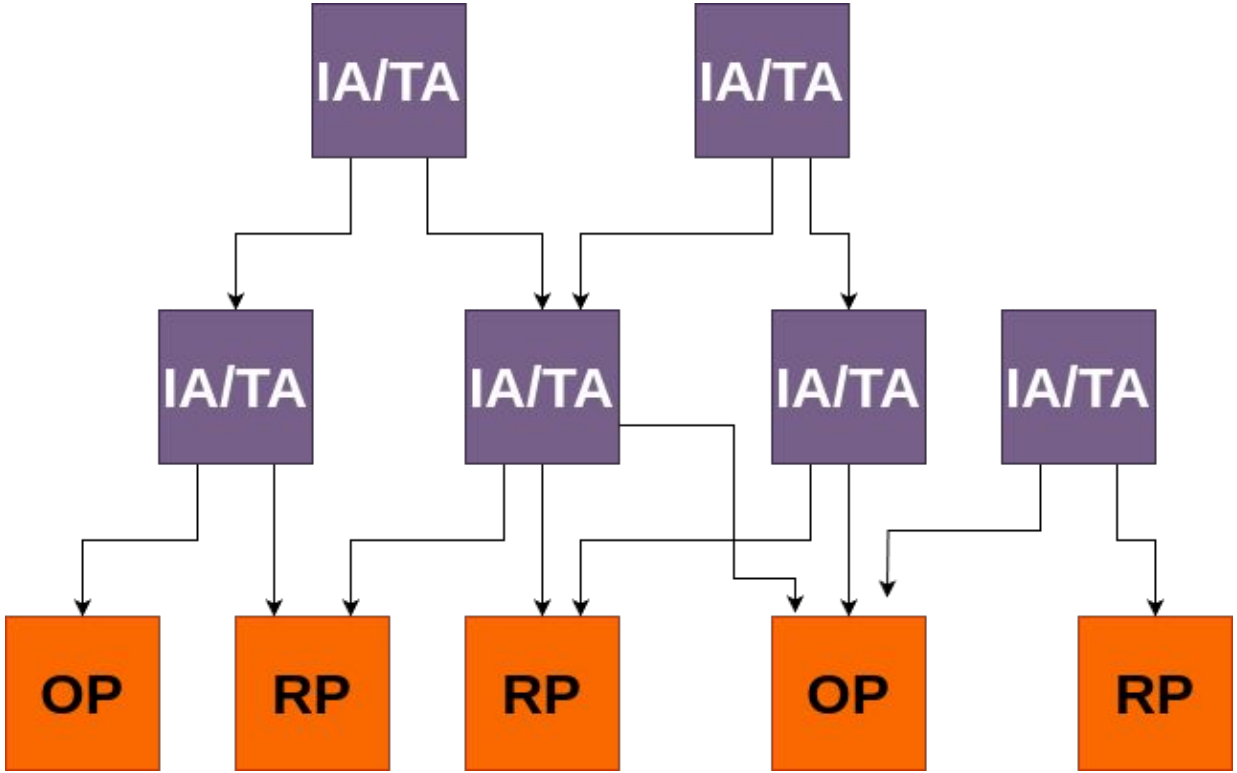

Trust Anchors
TA
TA
TA
Intermediates
IA
IA
IA
IA
Leaf Entity
OP
RP
RP
RP
OP

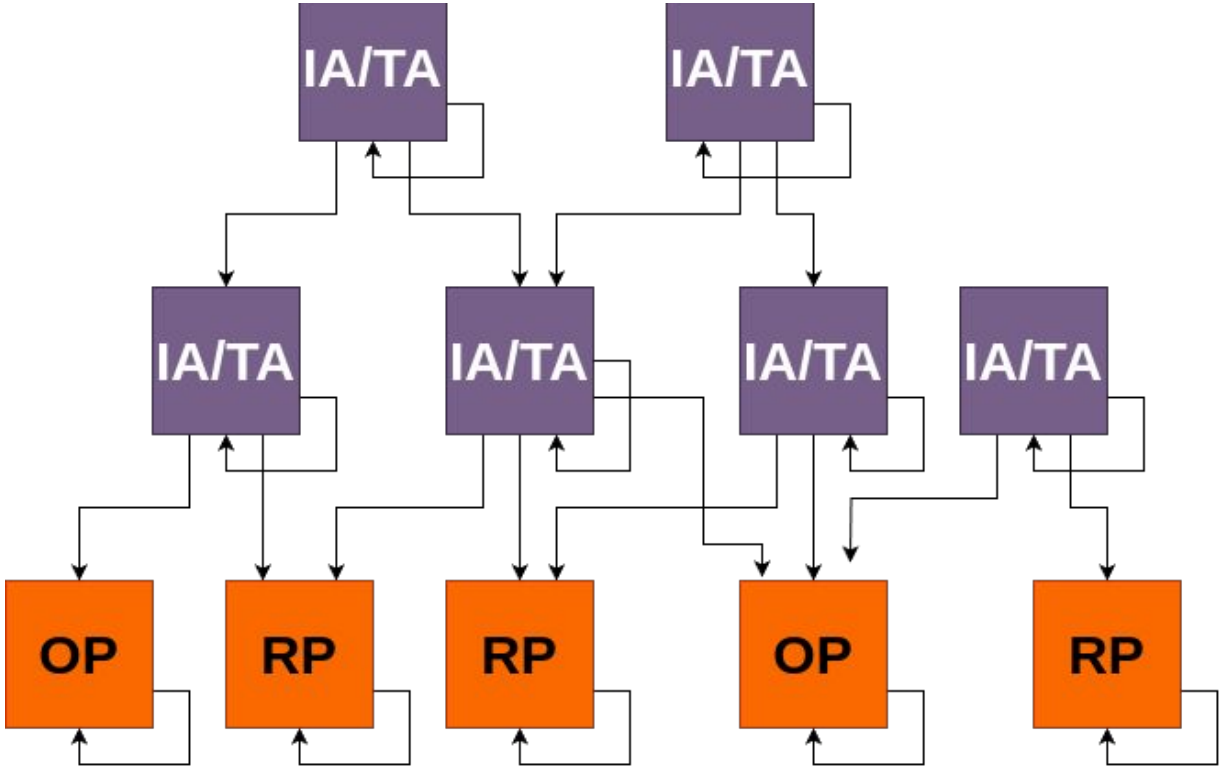

IA/TA
IA/TA
IA/TA
IA/TA
IA/TA
IA/TA
OP
RP
RP
OP
RP


IA/TA
IA/TA
IA/TA
IA/TA
IA/TA
IA/TA
OP
RP
RP
OP
RP


IA/TA
IA/TA
IA/TA
IA/TA
IA/TA
IA/TA
OP
RP
RP
OP
RP

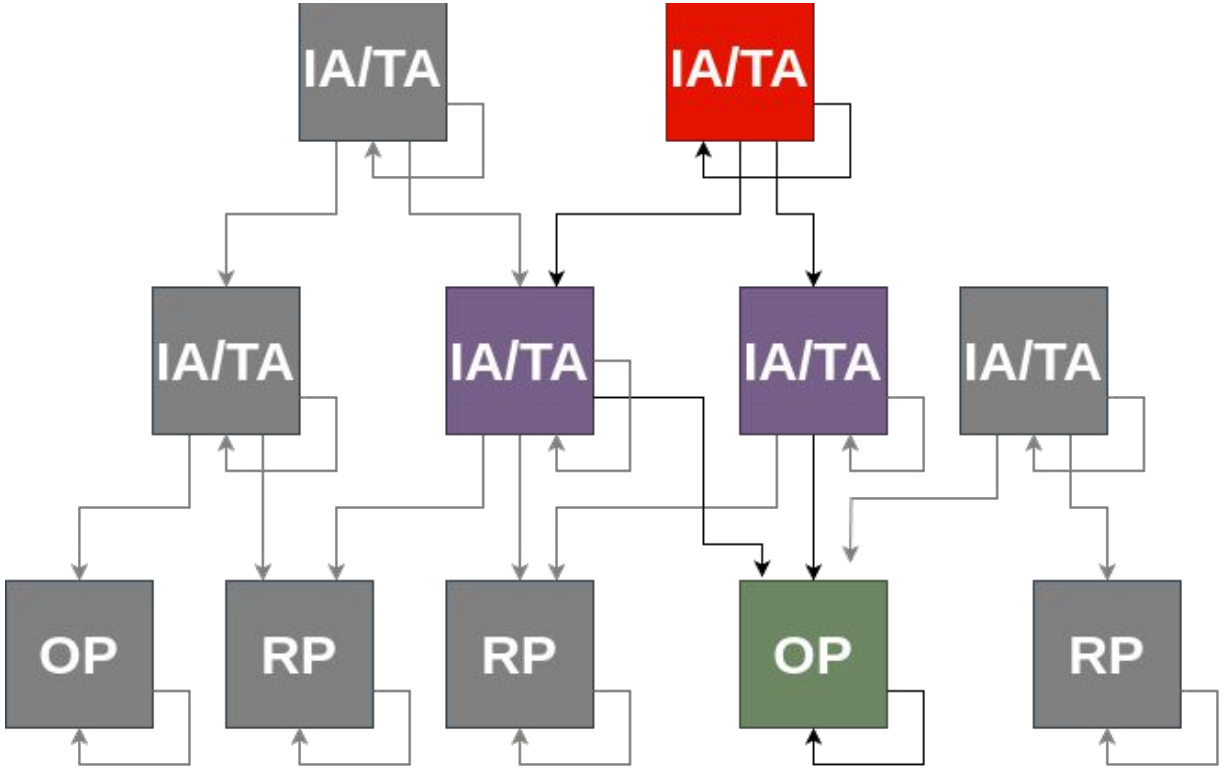
- Entity Identifier
  - URI -> OP: issuer url
- Entity Statement
  - Signed JWT
  - Contains information needed for the subject's entity to participate in federation
- Entity Configuration:
  - Self-signed Entity Statement
  - Endpoint: /.well-known/openid-federation


IA/TA
IA/TA
IA/TA
IA/TA
IA/TA
IA/TA
OP
RP
RP
OP
RP


IA/TA
IA/TA
IA/TA
IA/TA
IA/TA
IA/TA
OP
RP
RP
OP
RP

Upfront Basics

- Entity Identifier
  - URI -> OP: issuer url
- Entity Statement
  - Signed JWT
  - Contains information needed for the subject's entity to participate in federation
- Entity Configuration:
  - Self-signed Entity Statement
  - Endpoint: /.well-known/openid-federation
- Trust Chain
  - Chain of Entity Statements from a Leaf Entity [via Intermediates] to a Trust Anchor

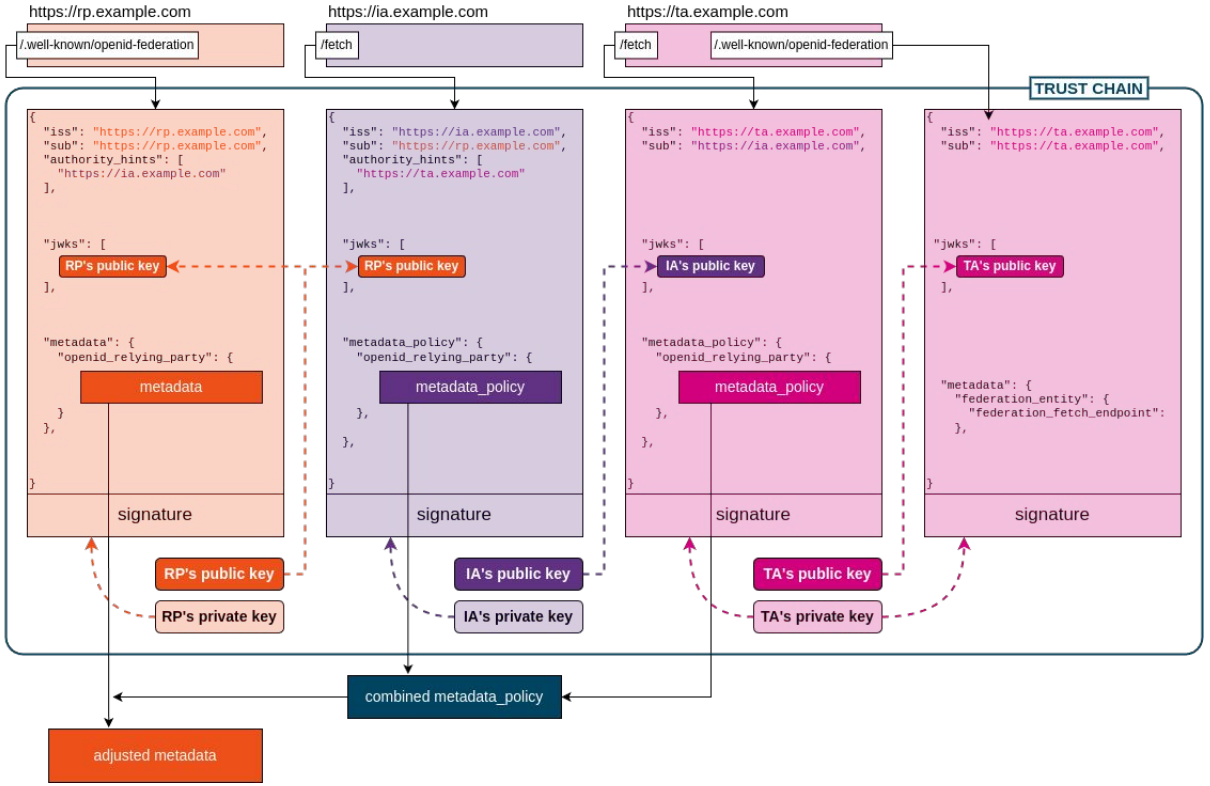

IA/TA
IA/TA
IA/TA
IA/TA
IA/TA
IA/TA
OP
RP
RP
OP
RP

Upfront Basics

- Entity Identifier
  - URI -> OP: issuer url
- Entity Statement
  - Signed JWT
  - Contains information needed for the subject's entity to participate in federation
- Entity Configuration:
  - Self-signed Entity Statement
  - Endpoint: /.well-known/openid-federation
- Trust Chain
  - Chain of Entity Statements from a Leaf Entity [via Intermediates] to a Trust Anchor
- OIDC Metadata
  - Combined from a Trust Chain
- Federation Endpoints

Trust Chain

Trust Chain


https://rp.example.com
/.well-known/openid-federation
https://ia.example.com
/fetch
https://ta.example.com
/fetch
/.well-known/openid-federation
TRUST CHAIN
"iss": "https://rp.example.com",
"sub": "https://rp.example.com",
"authority_hints": [
"https://ia.example.com"
],
"jwks": [
RP's public key
],
"metadata": {
"openid_relying_party": {
metadata
}
},
signature
"iss": "https://ia.example.com",
"sub": "https://rp.example.com",
"authority_hints": [
"https://ta.example.com"
],
"jwks": [
RP's public key
],
"metadata_policy": {
"openid_relying_party": {
metadata_policy
},
},
signature
"iss": "https://ta.example.com",
"sub": "https://ia.example.com",
"jwks": [
IA's public key
],
"metadata_policy": {
"openid_relying_party": {
metadata_policy
},
},
signature
"iss": "https://ta.example.com",
"sub": "https://ta.example.com",
"jwks": [
TA's public key
],
"metadata": {
"federation_entity": {
"federation_fetch_endpoint":
},
signature
RP's public key
RP's private key
IA's public key
IA's private key
TA's public key
TA's private key
combined metadata_policy
adjusted metadata

Trust Chain Resolution

Prerequisites:

- Leaf’s Entity ID
- Trust Anchors

Trust Chain Resolution - Flow

Trust Chain Resolution

Trust Chain Resolution - Flow

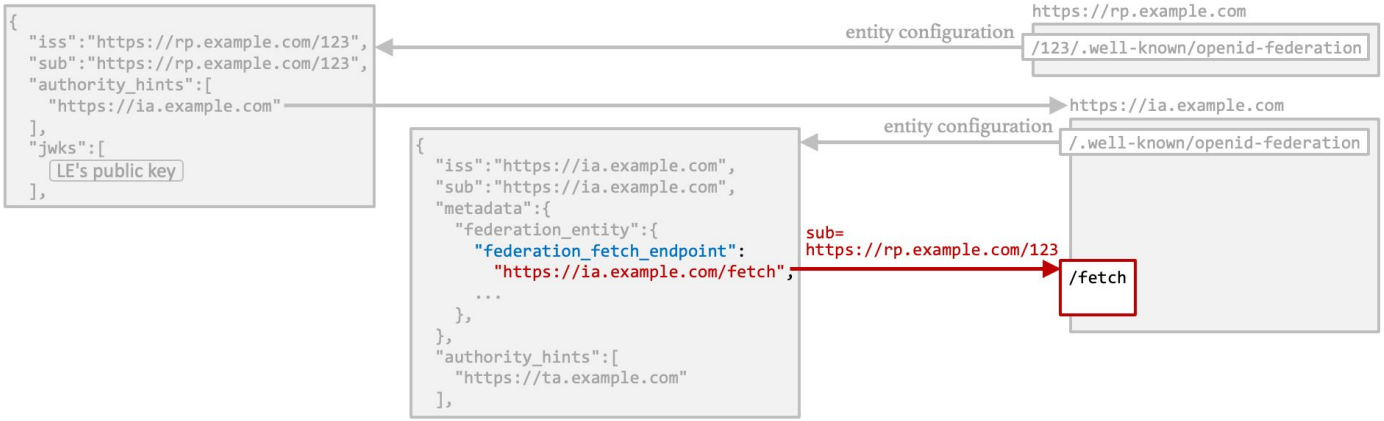
Trust Chain Resolution

Trust Chain Resolution - Flow

Trust Chain Resolution

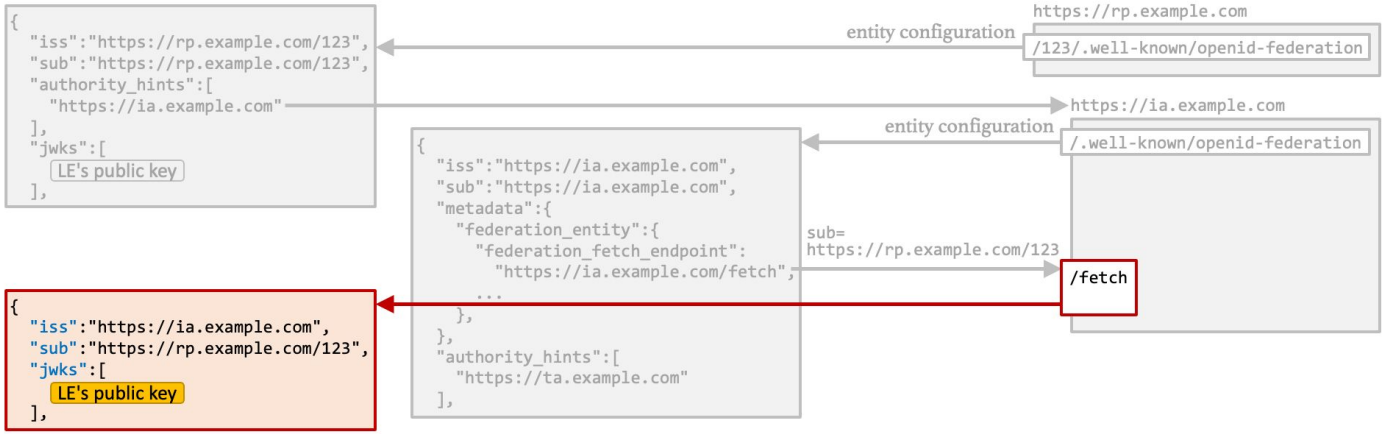
Trust Chain Resolution - Flow

Trust Chain Resolution



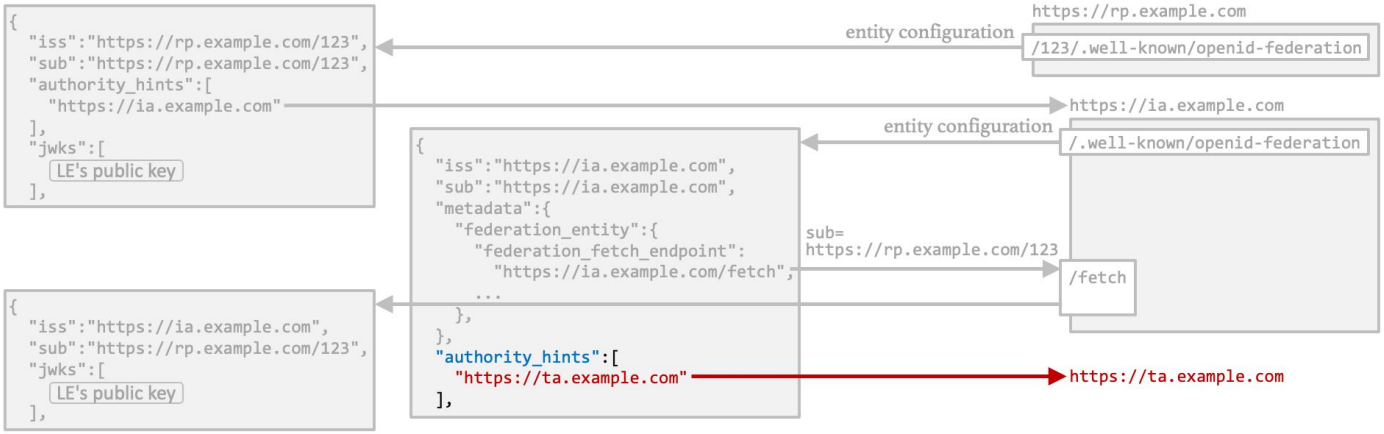
Trust Chain Resolution - Flow

Trust Chain Resolution



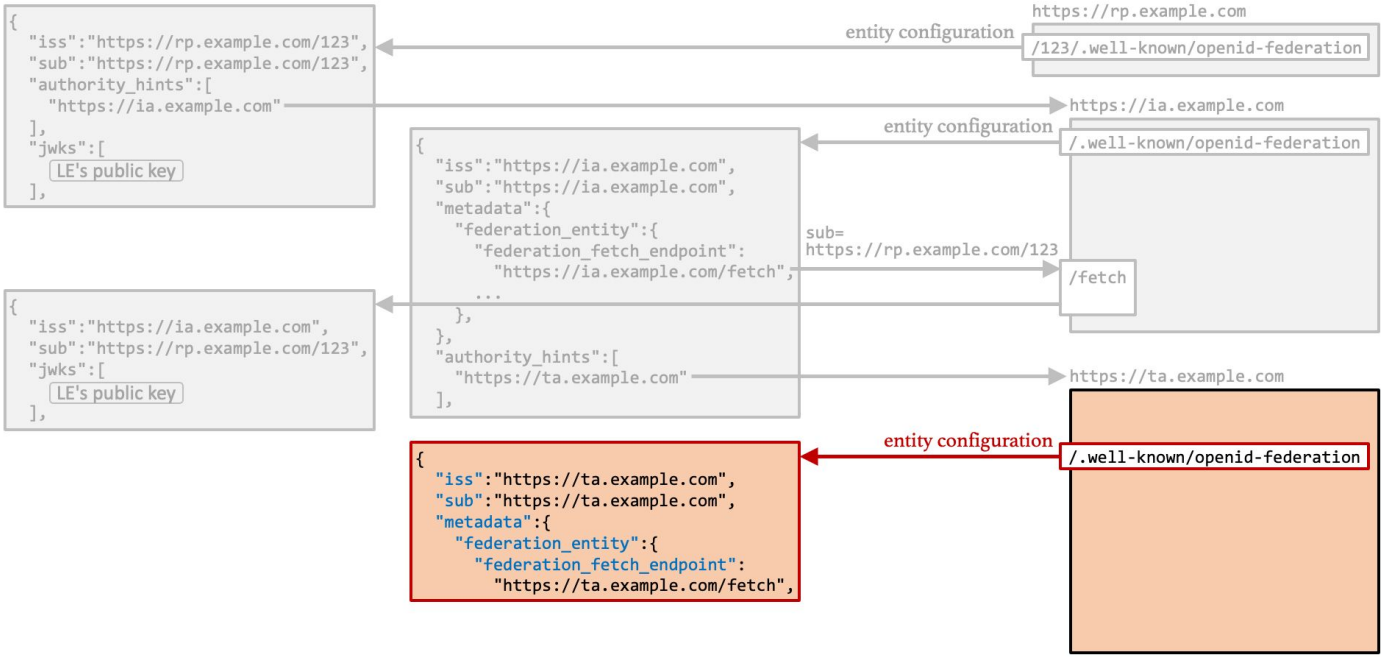
Trust Chain Resolution - Flow

Trust Chain Resolution



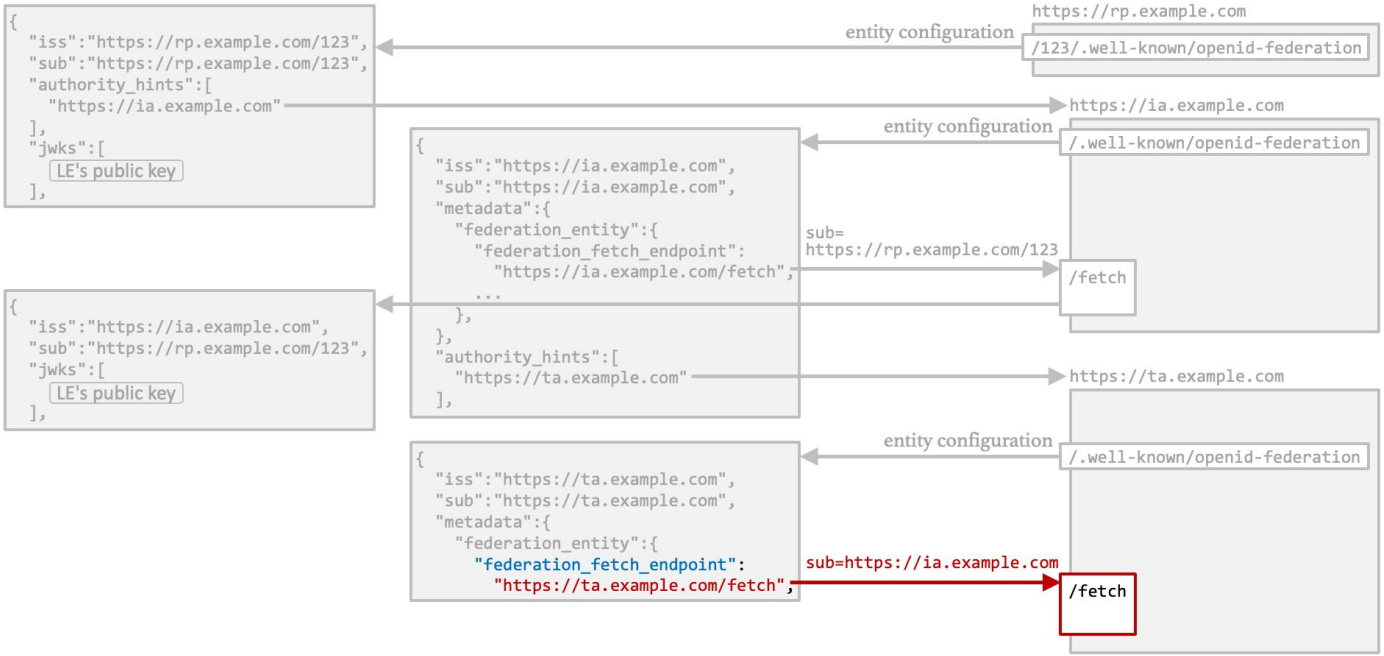
Trust Chain Resolution - Flow

Trust Chain Resolution



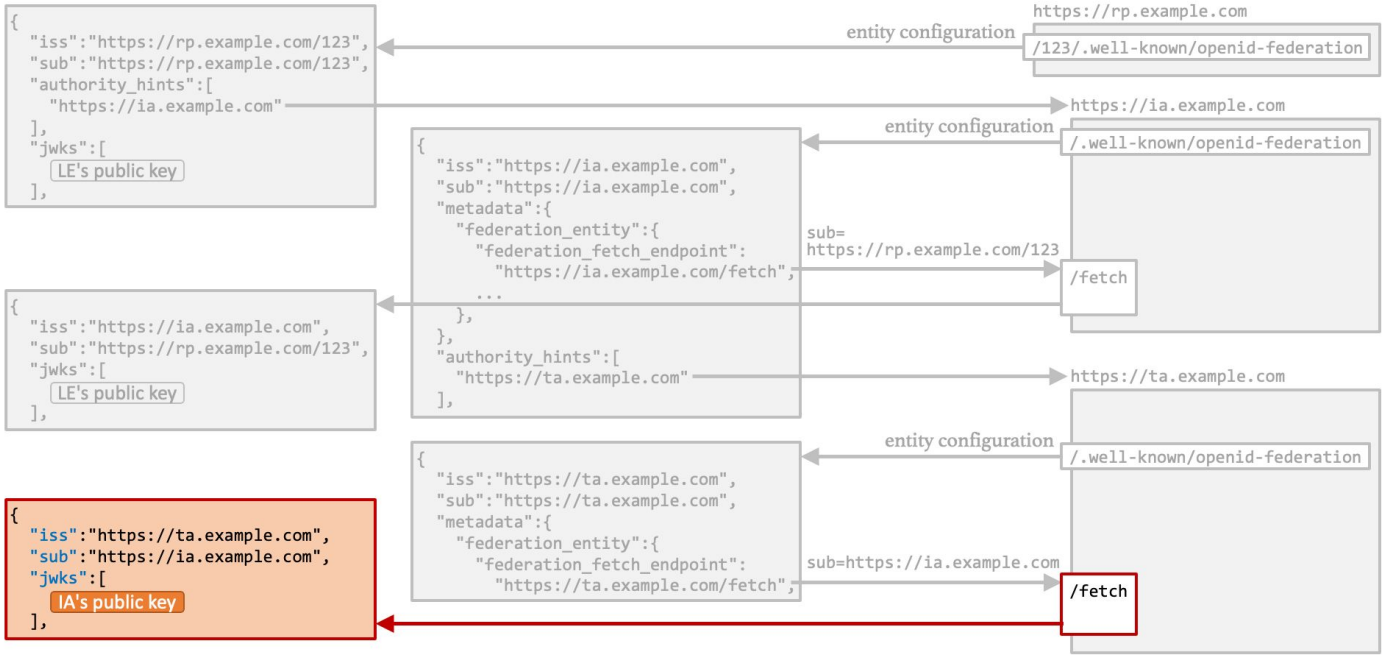
Trust Chain Resolution - Flow

Trust Chain Resolution



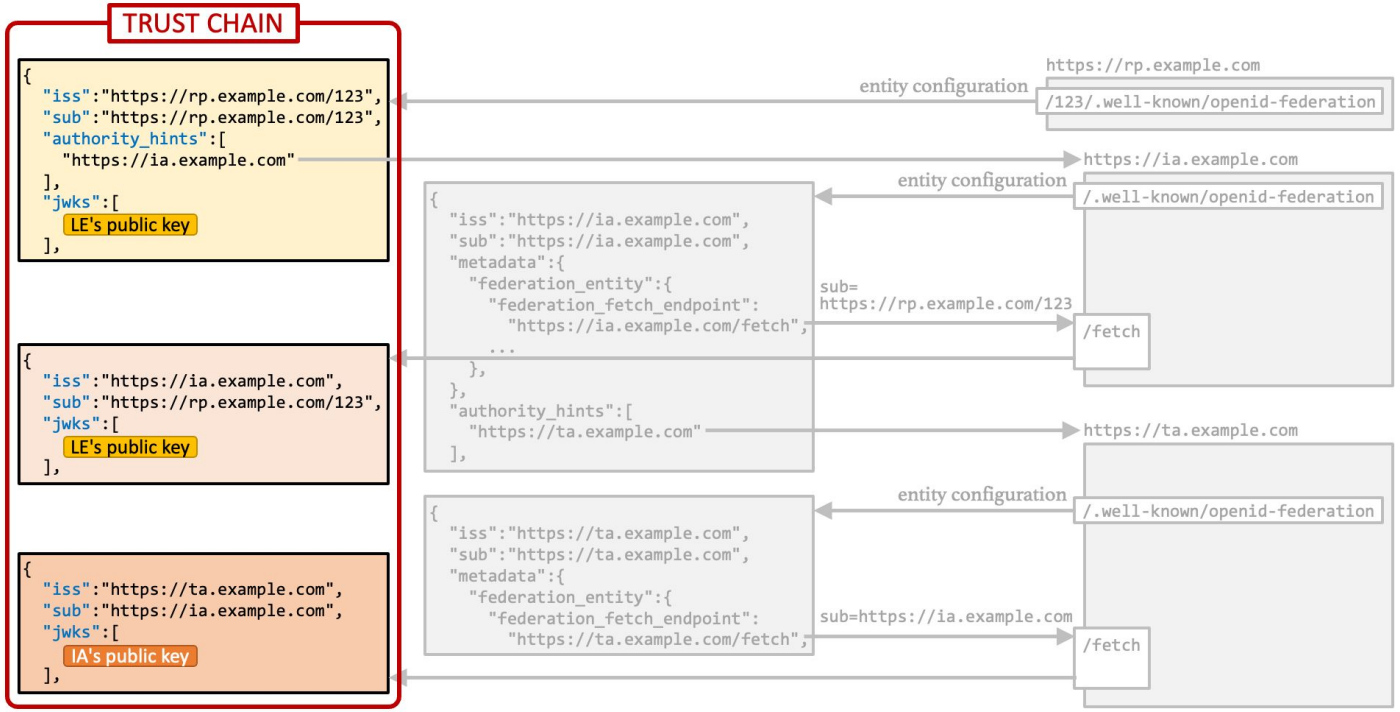
Trust Chain Resolution - Flow

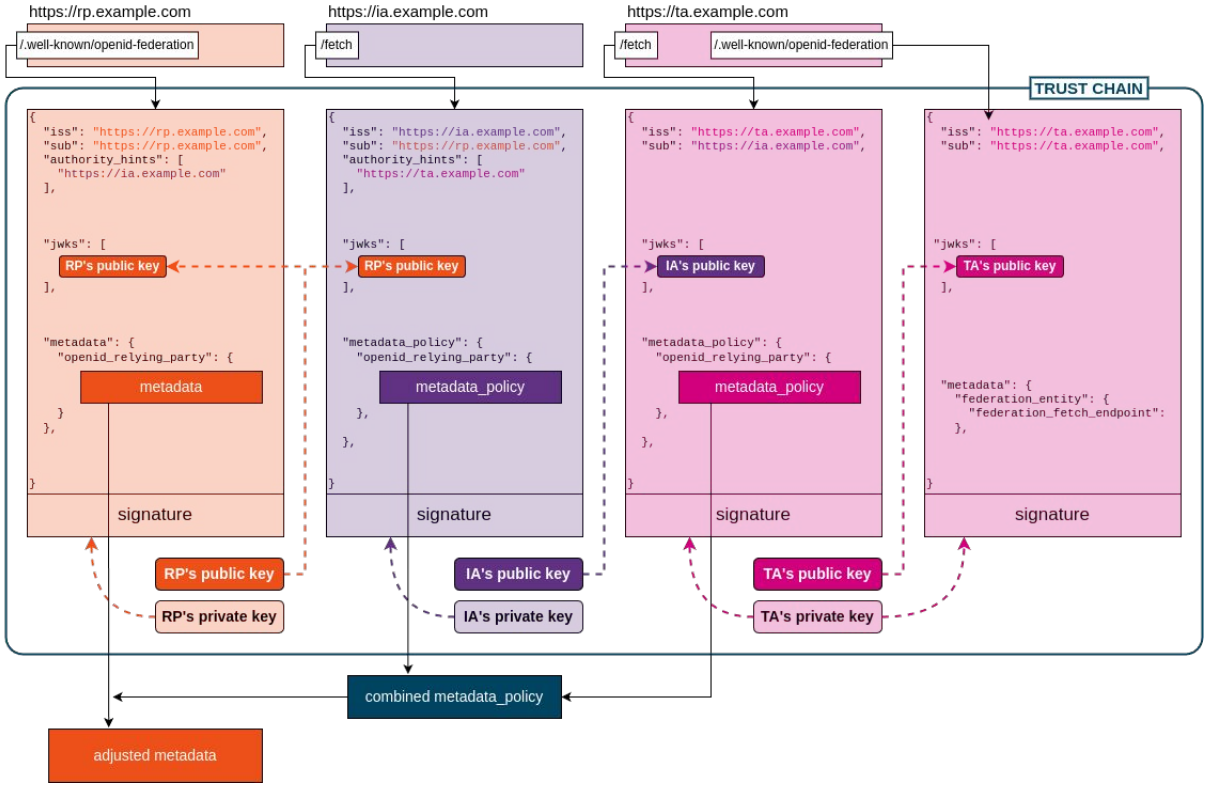
Trust Chain Resolution



Trust Chain Resolution - Flow

Trust Chain Resolution




https://rp.example.com
/.well-known/openid-federation
https://ia.example.com
/fetch
https://ta.example.com
/fetch
/.well-known/openid-federation
TRUST CHAIN
"iss": "https://rp.example.com",
"sub": "https://rp.example.com",
"authority_hints": [
"https://ia.example.com"
],
"jwks": [
RP's public key
],
"metadata": {
"openid_relying_party": {
metadata
}
},
signature
"iss": "https://ia.example.com",
"sub": "https://rp.example.com",
"authority_hints": [
"https://ta.example.com"
],
"jwks": [
RP's public key
],
"metadata_policy": {
"openid_relying_party": {
metadata_policy
},
},
signature
"iss": "https://ta.example.com",
"sub": "https://ia.example.com",
"jwks": [
IA's public key
],
"metadata_policy": {
"openid_relying_party": {
metadata_policy
},
},
signature
"iss": "https://ta.example.com",
"sub": "https://ta.example.com",
"jwks": [
TA's public key
],
"metadata": {
"federation_entity": {
"federation_fetch_endpoint":
},
signature
RP's public key
RP's private key
IA's public key
IA's private key
TA's public key
TA's private key
combined metadata_policy
adjusted metadata

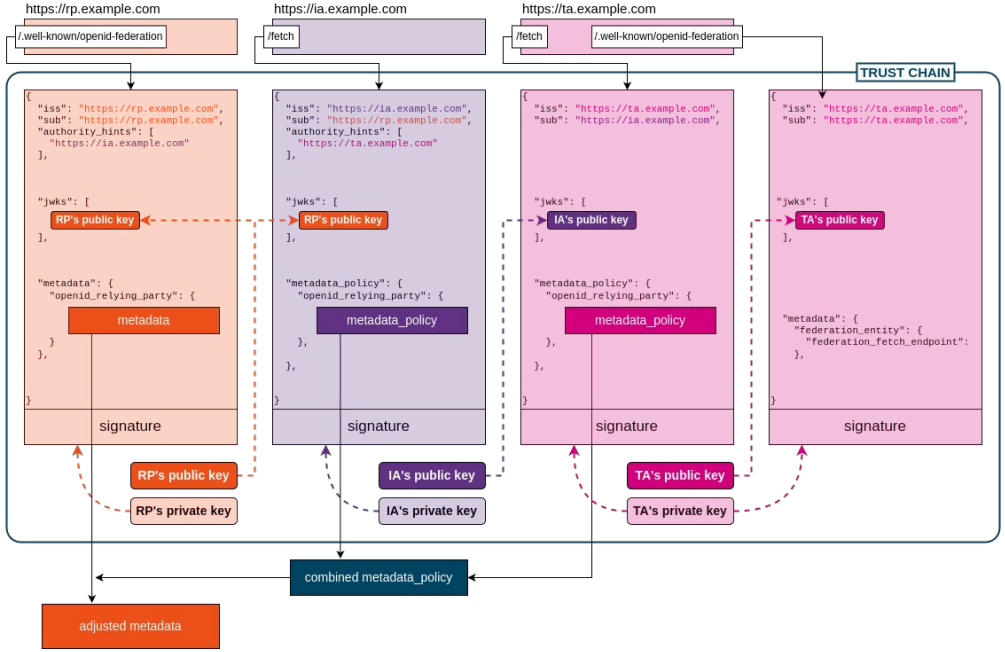
Trust Chain Verification

- Chain Integrity
- Signatures
- Expirations -> minimum expiration is chain expiration
- Valid Metadata
- Choose one chain, if multiple options

Metadata

- A Leaf Entity's metadata is obtained by:
  - Combining metadata_policy in the chain
  - Applying the (combined) policy to the metadata in its Entity Configuration

  - (Superior can also set metadata for its direct subordinate, through metadata)



- Different metadata types:
  - openid_relying_party oauth_client oauth_resource
    - client metadata + client_registration_types
  - openid_provider oauth_authorization_server
    - OP metadata + metadata about registration + auth
  - federation_entity
    - endpoints, fed metadata
- General claims for all types:
  - organization_name, contacts, logo_uri, policy_uri, homepage_uri, signed_jwks_uri, jwks_uri, jwks

Metadata Policy

- metadata_policy has **policy entries**:
  - Metadata parameter, e.g. id_token_signed_response_alg
  - One or more **operators** with their operator value(s)
- Example:

```
"metadata_policy" : {
  "openid_relying_party": {
    "id_token_signed_response_alg": {
      "default": "ES256",
      "one_of" : ["ES256", "ES384", "ES512"]
    }
  }
}
```

Policy Operators

Policy Operator Definition

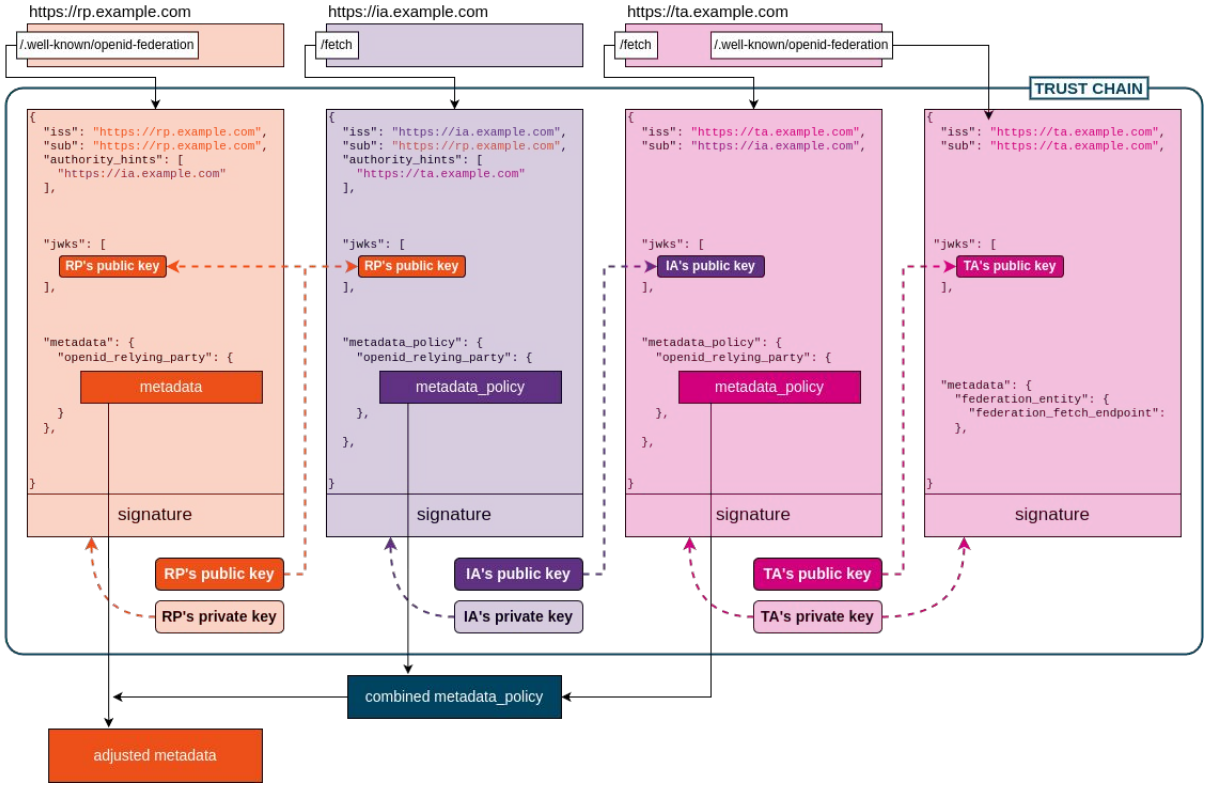
- Name
- Metadata parameter JSON value types
  - Mandatory
  - Optional
- Action
  - Value check
  - Value modification
  - Both
- Operator JSON value types
  - Mandatory
  - Optional
- Allowed Operator Combinations
- Order
- Merging

Metadata Policy - Operators

- Value modifiers:
  - value — Set this value
  - add — Add this value (if not present)
  - default — Set this value if none set
- Value checks:
  - one_of — Value must be one of the listed
  - subset_of — Intersection (potential modifier)
  - superset_of — Defines values must be included
  - essential — Indicates if a value is required

- Additional Operators may be used

OpenID Federations in One Figure



Federation Endpoints

Federation Endpoint

- **Fetch Endpoint**
- **Subordinate Listing Endpoint**
- **Resolve Endpoint**
- **Trust Mark Status Endpoint**
- **Trust Marked Entities Listing Endpoint**
- **Trust Mark Endpoint**
- Historical Keys Endpoint
  - /.well-known/openid-federation-historical-jwks

Federation Fetch Endpoint

- Used to collect entity statements when building the trust chain.

- Request: GET
  - Parameters: sub REQUIRED
  - Example:
    - GET https://ia.example.com/fetch?sub=https://rp.example.com/123
- Response: Entity Statement about the sub

Federation Subordinate Listing

- Query list of all Entities immediately subordinate
- Can be filtered by entity_type, trust_marked, trust_mark_id, intermediate
- Response: JSON Array of entity ids

Example response:

```
200 OK
Content-Type: application/json

[
  "https://ntnu.andreas.labs.uninett.no/",
  "https://blackboard.ntnu.no/openid/callback",
  "https://serviceprovider.andreas.labs.uninett.no
     /application17"
]
```

Federation Resolve Endpoint

- Fetch resolved metadata as trusted by the resolver
- GET Request: sub, trust_anchor, [entity_type]
- Response (signed JWT): metadata, trust_chain, [trust_marks]

Trust Mark Status Endpoint

- Used to check whether a Trust Mark has been issued to an entity and is still active or not.
- Query is sent to the trust mark issuer.
- GET Request: sub, trust_mark_id, [iat]
- Response: active

Trust Marked Entities Endpoint

- The Trust Marked Entities listing endpoint is exposed by Trust Mark Issuers and lists all the Entities for which Trust Marks have been issued and are still valid.
- GET Request: trust_mark_id, [sub]
- Response: JSON Array of entity ids

Trust Mark Endpoint

- The Trust Mark endpoint is exposed by a Trust Mark Issuer to provide Trust Marks to subjects.
- GET Request: trust_mark_id, sub
- Response: Trust Mark JWT

Trust Marks?

- Signed JWT -> Signed by a federation-accredited authority
- Content: iss, sub, trust_mark_id, iat, [logo_uri], [exp], [ref], [delegation]

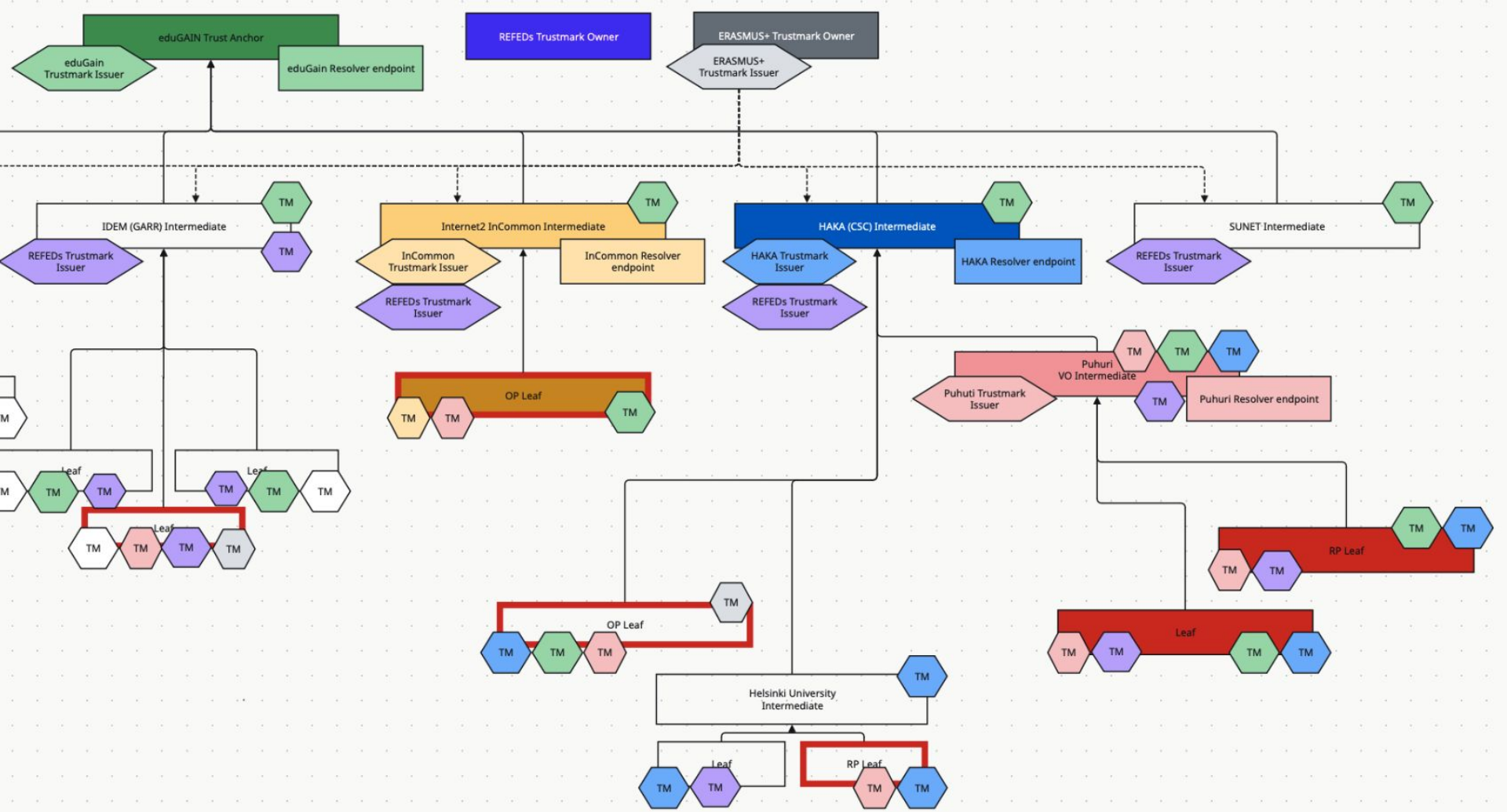
```
{
  "iss": "https://secusign.org",
  "sub": "https://example.com/op",
  "iat": 1579621160,
  "id": "https://secusign.org/level/A",
  "logo_uri": "https://secusign.org/static/levels/
    certification-level-A-150dpi-90mm.svg",
  "ref": "https://secusign.org/conformances/"
}
```

- Trust Anchors can restrict who can issue a certain trust mark

```
"trust_marks_issuers": {
  "https://openid.net/certification/op": ["*"],
  "https://refeds.org/wp-content/uploads/2016/01/Sirtfi-1.0.pdf":
    ["https://swamid.se"]
}
```

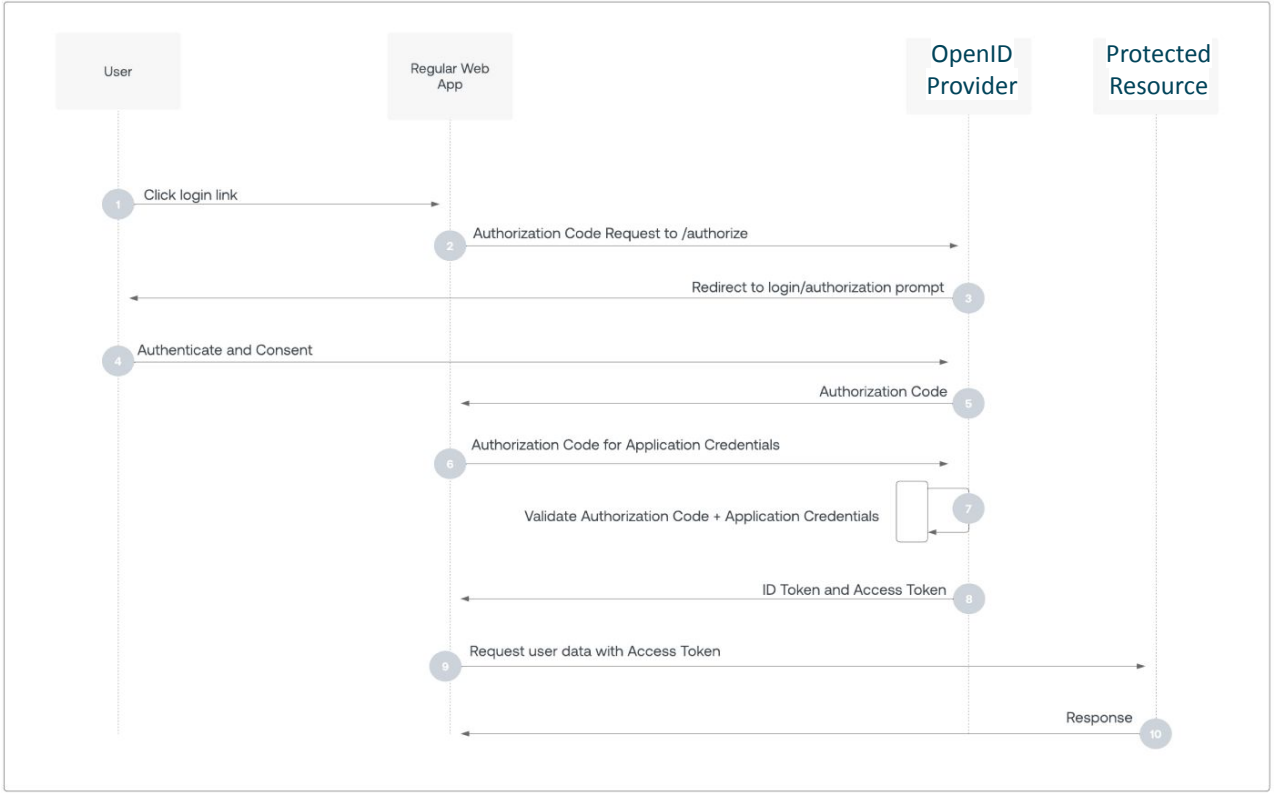
Trust Mark Delegation

- Trust Mark Issuance can be delegated by the Trust Mark Owner to Trust Mark Issuers
- The Trust Mark Issuer must be part of the federation, the Trust Mark Owner might not be
- Trust Mark Owner issues a Delegation JWT to the Trust Mark Issuer
  - This delegation JWT is included in the Trust Mark JWT
- Trust Anchor publishes information about Trust Mark Owners in its Entity Configuration
  - This includes the Trust Mark Owners jwks used for verifying the Delegation JWT


eduGAIN Trust Anchor
eduGain Trustmark Issuer
eduGain Resolver endpoint
REFEDs Trustmark Owner
ERASMUS+ Trustmark Owner
ERASMUS+ Trustmark Issuer
IDEM (GARR) Intermediate
REFEDs Trustmark Issuer
Internet2 InCommon Intermediate
InCommon Trustmark Issuer
InCommon Resolver endpoint
REFEDs Trustmark Issuer
HAKA (CSC) Intermediate
HAKA Trustmark Issuer
HAKA Resolver endpoint
REFEDs Trustmark Issuer
SUNET Intermediate
REFEDs Trustmark Issuer
OP Leaf
Puhuri VO Intermediate
Puhuti Trustmark Issuer
Puhuri Resolver endpoint
Leaf
Leaf
Leaf
OP Leaf
RP Leaf
Leaf
Helsinki University Intermediate
Leaf
RP Leaf
TM

OIDC Communication

Plain OIDC Authorization Code Flow



OIDC Communication

- No classical client registration
- How is trust & configuration established?

Two types:

- Automatic Registration
- Explicit Registration

Automatic Registration

- RP does no Registration!
- RP uses Entity Identifier as the client_id
- OP fetches and verifies trust chains. -> client metadata
  - Verify request authentication

OP can decide if a Auth request uses OID fed automatic registration:

- OP supports OID Fed and Automatic Registration
- Incoming client_id is a valid URL
- Client ID is not a registered client

Explicit Registration

- RP does explicit registration with OP prior to other requests
- Similar to Dynamic Client Registration, but with Trust Chains
  - RP resolves OPs metadata
  - RP sends signed Entity Statement (or Trust Chain) about itself as request
    - Adjusted to OP's metadata
  - OP verifies RP's metadata
  - OP registers client
  - OP returns signed Entity Statement (the registered metadata)
  - RP verifies it; can use it
- Federation Registration Endpoint

Explicit Registration - Expiration

- OID fed explicit client registration is not valid forever
  - Entity Statements all have expiration times
- RP must expect that the registration becomes invalidated at any time
  - Re-register

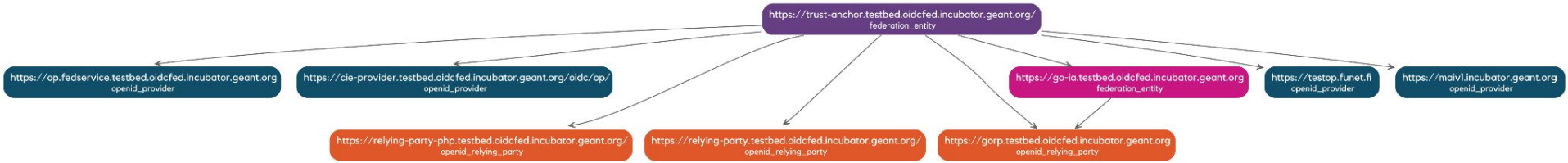
Tooling

- Implementations:
  - https://openid.net/developers/openid-federation-implementations/

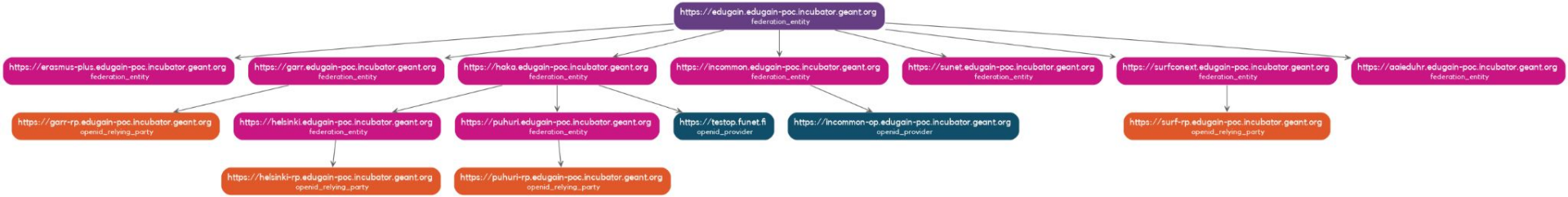
Implementations & Tools

Implementation	RP	OP	IA/TA	Resolver	Trust Mark Issuer	Visualization
Marko's SimpleSAMLphp https://github.com/simplesamlphp/openid		✓				
Henri's Shibboleth https://git.shibboleth.net/view/?p=java-idp-oidc.git;a=summary (dev/JOIDC-222 branch, still heavy in development!)		✓				
Roland's fedservice https://github.com/SUNET/fedservice	✓	✓	✓	✓	✓	
Gabriel's Go https://github.com/zachmann/go-oidfed	✓		✓	✓	✓	
Diana's ofcli https://github.com/dianagudu/ofcli						✓
Guiseppe's federation browser https://github.com/italia/openid-federation-browser						✓

https://trust-anchor.testbed.oidcfed.incubator.geant.org/federation_entity



https://edugain.edugain-poc.incubator.geant.org/



Thank you

Any questions?

The research leading to these results has received funding from
the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 856726 (GN4-3).