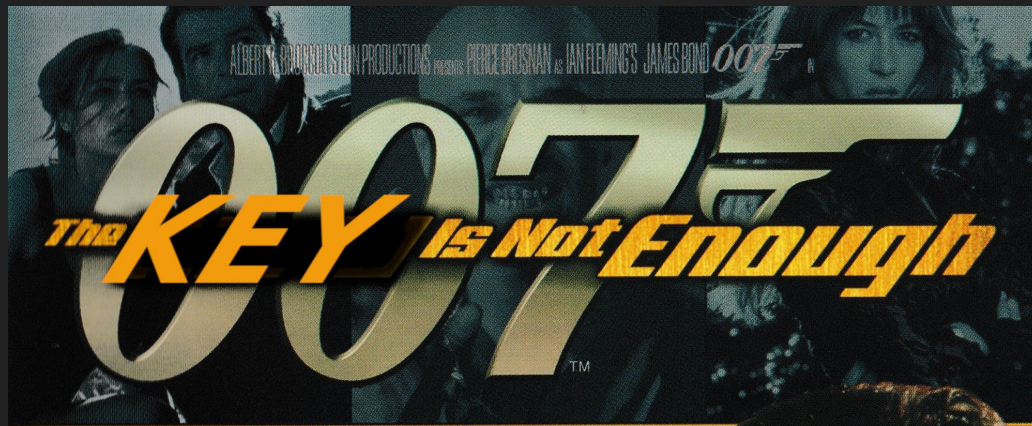




OSW 2023
🇬🇧 London

The Key Is Not Enough

OpenID Connect Federation 1.0

OAuth Security Workshop 2023 London

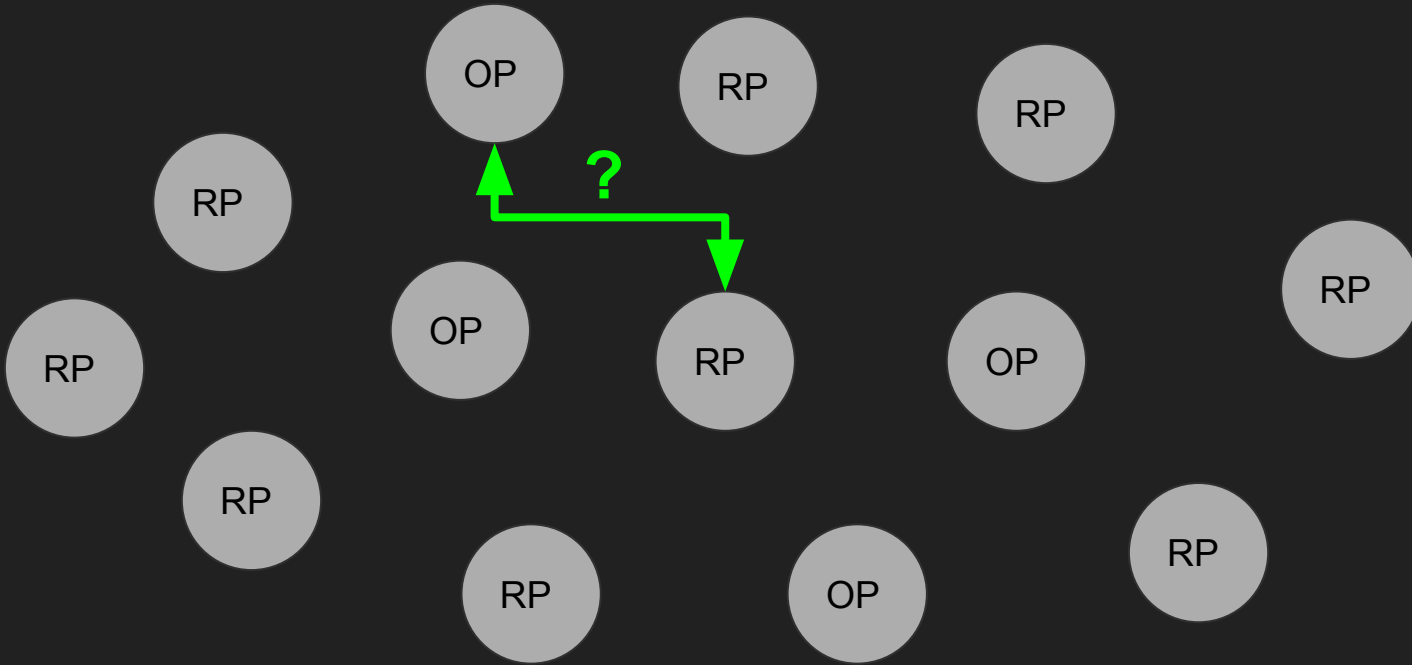
Roland Hedberg , Michael B. Jones , Giuseppe De Marco , John Bradley , Vladimir Dzhuvinov 

How it all began

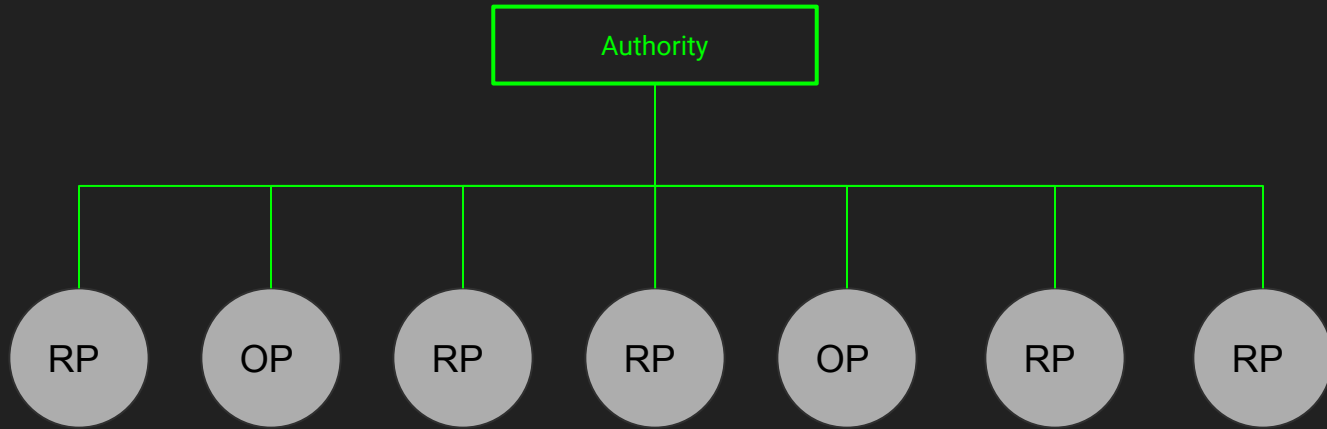
- 2011 - The 💡 idea is born in Roland Hedberg's mind when working on OIDC
- 2016 - First draft 📄 of OpenID Connect Federation 1.0
- 2020 - First interop 🔌 between implementations
- 2021 - The valley of desperation: 😞 "Why is nobody interested?"
- 2022 - Wow, Italy 🇮🇹 adopts OIDC Federation for their national eID 🎉 !!!



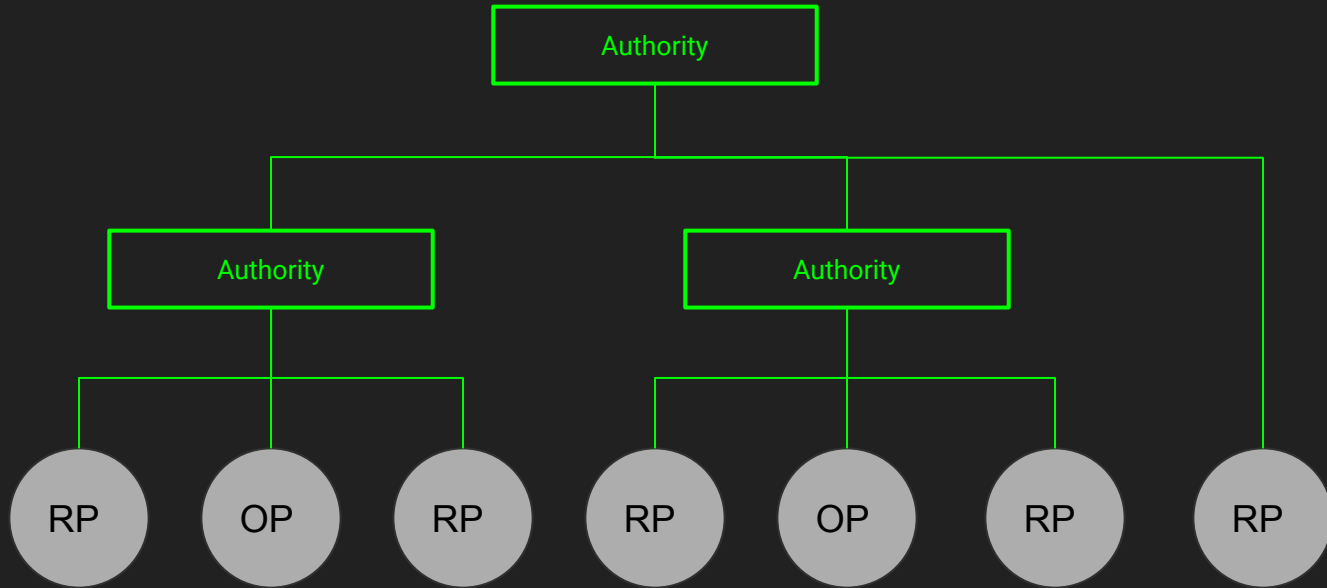
How was OIDC federation invented?



How can an RP and OP establish mutual trust?

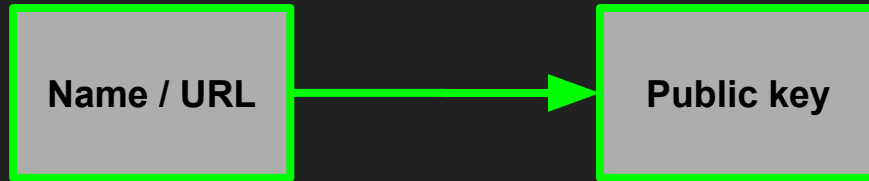


Let's introduce a trusted authority



Better still, allow for authority delegation

Let's introduce public key attestation to enable
OPs and RPs to authenticate one another



An authority will attest the public keys of trusted OPs and RPs to enable
them to mutually authenticate

X.509 certs, what else?



Bootstrapping OIDC: Trusting a URL is not enough

```
{  
  "application_type": "web",  
  "grant_types": ["authorization_code"],  
  "response_types": ["code"],  
  "redirect_uris": ["https://example.com/cb"],  
  "token_endpoint_auth_method":  
    "private_key_jwt",  
  "client_name": "My new app",  
  ...  
}
```

- OpenID Connect 1.0 relies on the concept of **RP and OP metadata**
- The metadata includes **critical parameters**, such as redirect URIs and public JWKs
- The metadata must be **mutually trusted** before the RP can register with the OP and send the end-user for authentication
- The authority in the federation must be able to **attest entity metadata**

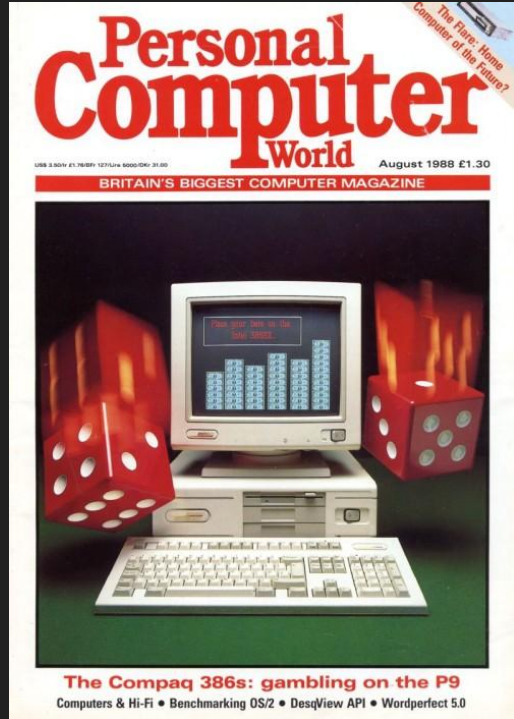
X.509 certs, what else?



But how to include metadata in the attestation?

```
{  
  "application_type": "web",  
  "grant_types": ["authorization_code"],  
  "response_types": ["code"],  
  "redirect_uris": ["https://example.com/cb"],  
  "token_endpoint_auth_method": "private_key_jwt",  
  "client_name": "My new app",  
  ...  
}
```

1988 - The year the X.509 certificate is invented



- The 386 PC and the high-density 1.44 MB floppy disk become a hot thing
- ASN.1/DER is a hot thing
- The WWW is not a thing yet
- HTTP is not a thing yet
- Web APIs are not a thing yet
- JSON is not a thing yet
- JWT is not a thing yet

Devise an X.509 cert ASN.1 extension???



JWT comes to rescue



The Entity Statement (ES)

- Signed **JWT**
- Attests the entity's **public JWKs**
- Enables an entity to publish its **metadata**
- Supports **multi-typed entities**
- Enables an authority for its subordinates:
 - to **attest metadata parameters**
 - to **define metadata parameter policies**
- May contain **trust marks**
- Has an **expiration time**
- Is **extensible** by protocols and applications

```
{  
  "iss": "https://trust\_anchor.example.com",  
  "sub": "https://op.example.com",  
  "iat": 1516239022,  
  "exp": 1516298022,  
  "jwks": { ... },  
  "metadata": {  
    "openid_provider": { ... }  
  },  
  "metadata_policy": {  
    "openid_provider": { ... }  
  },  
  "trust_marks": [ ... ]  
}
```

Example Trust Anchor metadata policy for OPs and RPs

```
"metadata_policy" :  
{  
  "openid_provider":  
  {  
    "id_token_signing_alg_values_supported":  
    {  
      "subset_of": [ "ES256", "ES384", "ES512" ],  
      "superset_of": [ "ES256" ]  
    }  
  }  
}
```

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

Enforces security and interoper **profiles**.

Subordinate OPs and RPs must **comply** else their Trust Chains become invalid!

Simple Trust Chain

Entity Statement

iss: https://trust_anchor.com
sub: <https://rp.example.com>
jwks: {...}
metadata_policy: {...}

Issued and signed by
the Trust Anchor

Sets the metadata
policy for subordinates



Entity Configuration

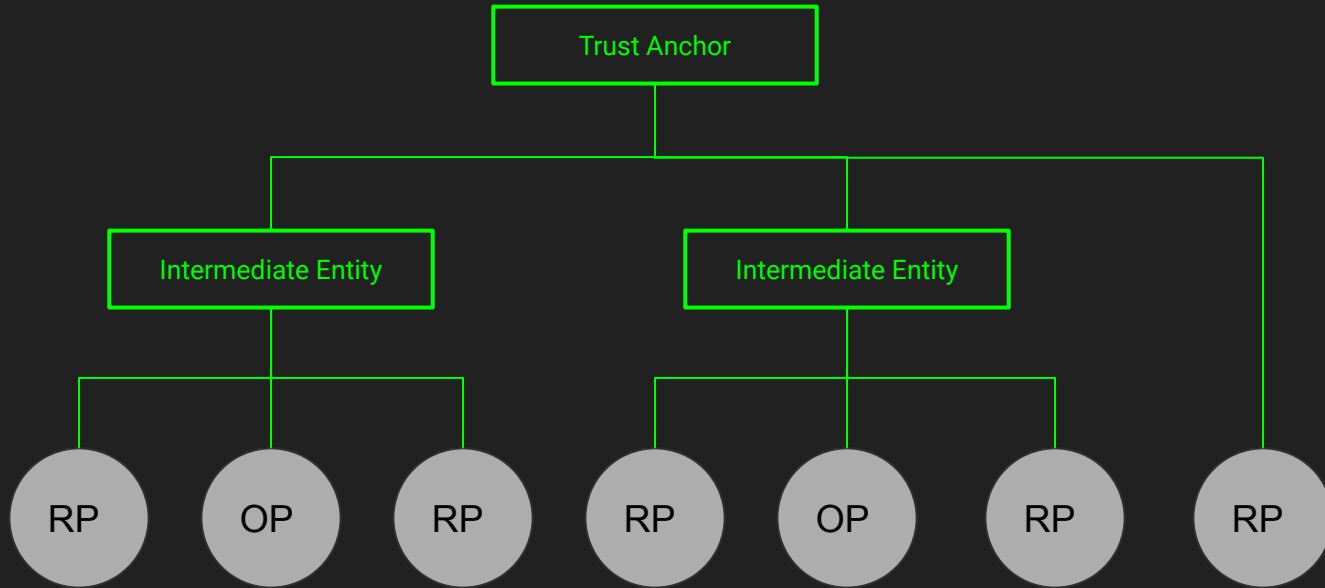
iss: <https://rp.example.com>
sub: <https://rp.example.com>
jwks: {...}
metadata: {...}

Issued and self-signed
by the RP

Sets the metadata
policy for subordinates

Mapping the terms

OIDC Federation 1.0	X.509
Entity Statement (ES)	Public key certificate
Trust chain	Certificate chain
Trust Anchor (TA)	Root Certificate Authority (CA)
Intermediate Entity	Intermediate CA
Trust Mark	-



Example OpenID Connect 1.0 Federation with a single Trust Anchor

Web APIs to navigate federations and aid trust decisions

	OIDC Federation 1.0	X.509
Well-known URL	✓	✗
Fetch statement about subject from authority	✓	✗
List subjects of authority	✓	✗
Resolution helpers	✓	✗
Trust mark status	✓	✗
Query an authority for expired and revoked subject keys	✓	✗

OIDC Federation 1.0 vs X.509

	OIDC Federation 1.0	X.509
Attest public keys	✓	✓
Attest and police entity metadata	✓	✗
Support complex trust topologies incl. trust marks	✓	✗
Web APIs	✓	✗

The Magic of Automatic Registration

`https://server.example.com/authorize?`

`redirect_uri=https%3A%2F%2Frp.example.com%2Fauthz_cb`

`&scope=openid+email`

`&response_type=code`

`&client_id=https%3A%2F%2Frp.example.com`

`&request=eyJ0cnVzdF9jaGFpbil6WyJleUpoYkdj...`

The **Trust Chain** is fetched

The Magic of Automatic Registration

`https://server.example.com/authorize?`

`redirect_uri=https%3A%2F%2Frp.example.com%2Fauthz_cb`

`&scope=openid+email`

`&response_type=code`

`&client_id=https%3A%2F%2Frp.example.com`

`&request=eyJ0cnVzdF9jaGFpbil6WyJleUpoYkdj...`

`&trust_chain=eyJhbGciOiJSUzI1NiIsImtpZCI6I...`

The **Trust Chain** is inlined

The Magic of Automatic Registration

`https://server.example.com/authorize?`

`redirect_uri=https%3A%2F%2Frp.example.com%2Fauthz_cb`

`&scope=openid+email`

`&response_type=code`

`&client_id=https%3A%2F%2Frp.example.com`

`&request=eyJ0cnVzdF9jaGFpbil6WyJleUpoYkdj...`

1988

**Public Key
Infrastructure**

X.509



2023

**Trust
Infrastructure**

OIDC Federation 1.0

2023 - How it's going

- Infiltrating and subverting OIDC4VC ☐
- Infiltrating and subverting eIDAS 2.0 & the EUDI  wallet
- Italy  adopts the Trust Chain for their wallet implementation 🎉
- The second valley of even grander desperation:

“When are we going to finish this spec?” 😞 😞 😞

The future?

- 2024 - At long last 😊 the spec is final and we celebrate the standard 🎉 !!!
- 2030 - The world 🌍 has deserted X.509 for the Trust Chain 🚗 🏃 🏃 🌈
- 2035 - The OSW goes 🚀 to Mars, Elon Musk 👽 is our invited speaker



MTI

With **automatic registration**:

- OP and RP must support **private_key_jwt** or **mTLS** client authentication
- OP must publish its Entity Statement at **/.well-known/openid-federation**
- RP must publish its Entity Statement at **/.well-known/openid-federation** or include a **trust_chain** in the OpenID authentication request

With **explicit registration**:

- OP must publish its Entity Statement at **/.well-known/openid-federation**
- The RP registers with its Entity Statement at the OP **federation_registration_endpoint**
- The RP must **renew** its client registration to continue using the OP

Remaining work

1. External metadata from well-known endpoints (PR #589)?
2. New endpoint / web API to aid RPs using explicit client registration verify the registered metadata returned by the OP for being compliant with the trust chain policies
3. Formal analysis?
4. Anything else?