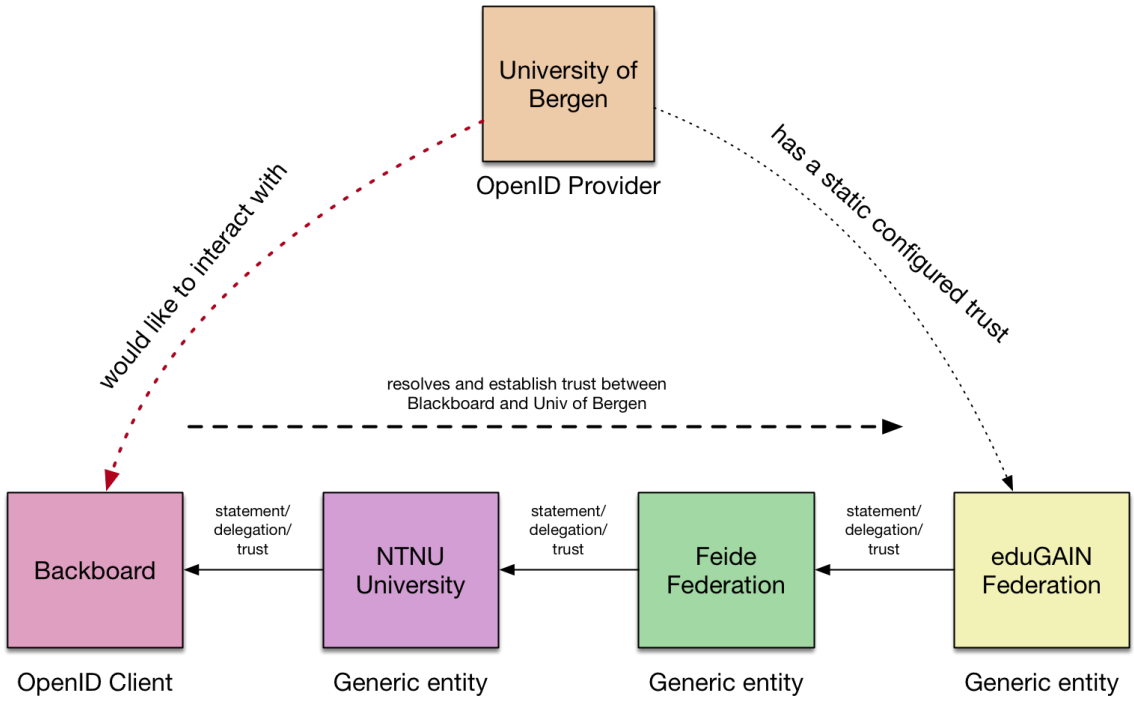
Federations - trust between entities

Some reflections how to establish trust between entities using current standard building blocks, preparing a trust layer to support both OAuth and OpenIDConnect Federations.

The basic trust pattern

The need for two entities to interact and establish trust even if they have never beforce exchanged messages is very common. It is used when your browser fetches a web page from a web server and when a SAML service providers ask a student to login. The same basic pattern will also apply for OpenID Federations, but the trust fabric should not be limited to OpenID.

To approach a generic solution we look at all involved parties as generic entities, with the ability to state something about another entity.



In the figure above the University of Bergen Identity Provider would like to resolve trust with an OpenID Client. The seeking entity has configured a static trust anchor delegating trust to the eduGAIN entity. eduGAIN states that Feide can be trusted, Feide states that NTNU can be trusted, and NTNU states that the Blackboard service provider can be trusted. When the provider resolve the chain of trust between the target entity and the locally configured trust root, it can derive trust of and start to interact with the target entity.

Entity statements

Let's refer to the statement that an entity issues about another entity as *Entity Statements*.

To technically establish and validate trust statements we use cryptography, entity statements contains a digital signature using asymmetric keys. All entities holds a self generated key pair.

In this context it is rather obvious to make use of JWTs for entity statements. It one of the building blocks that OpenID relies on.

An entity statement will include:

The identifier of the issuer (`iss`)

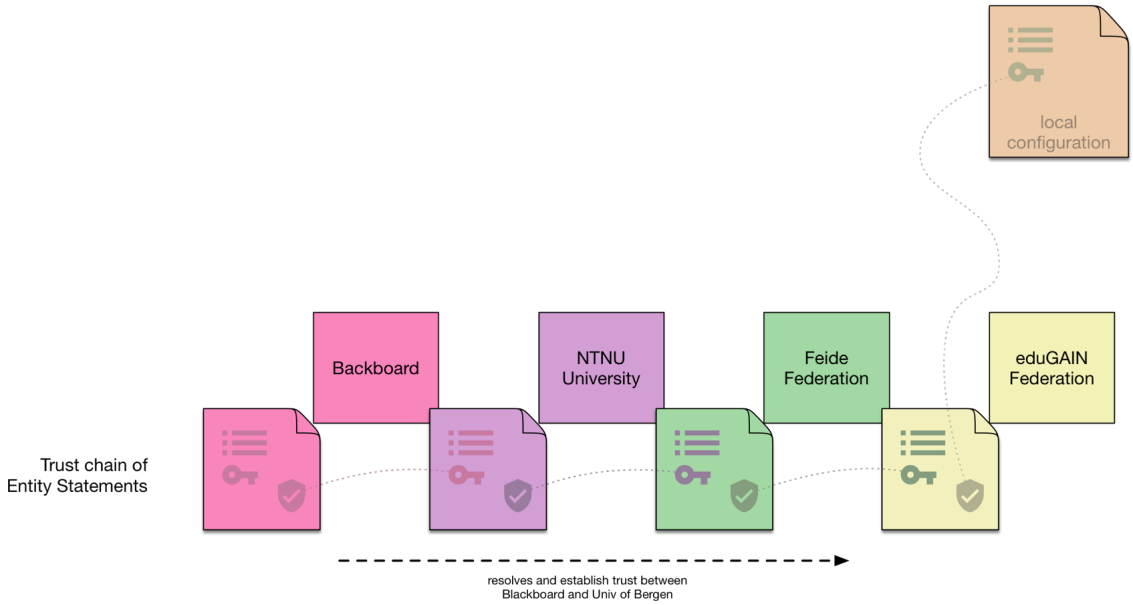
The identifier of the subject (`sub`)

A set of one or more protocol specific metadata describing the entity. In example an entity can be both a SAML Service Provider and an OpenID Connect client – it can have two set of metadata.

An expression of limited trust. It is very likely that the subject should be trusted less than the issuer. This information can often be baked into the protocol specific metadata. In example an OpenID Client metadata object can contain a claim about allowed scopes. An provider can be limited to a set of top level domains for the realm of the user IDs of its users.

The public key of the subject (`jwks`). This way we can trust that a statement origins from the same target as the entity statment is delegating trust to.

A cryptographic signature using the private key of the issuer.



Validation of trust

The seeking entity would need to validate all the signature, and link the end of the chain to local configuration and the target entity respectively.

To make use of the metadata from the chain in any protocols, such as OpenID Connect, the metadata needs to be flattened. The closer to the trust root, the higher priority when it comes to conflicting claims. The result of this process should be a metadata object that is compatible to metadata when no federations is involved and metadata is statically configured.

OpenID Connect Federation

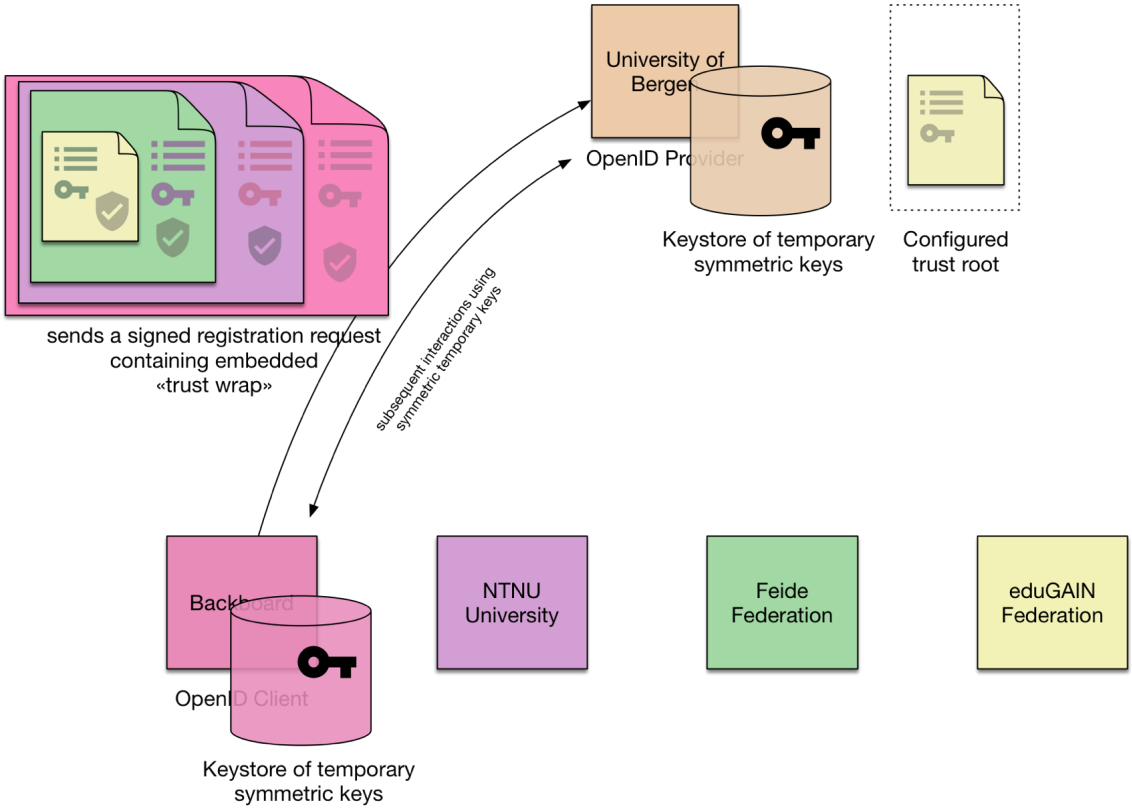
This section describes some aspects of the current OpenID Connect Federation specification (http://openid.net/specs/openid-connect-federation-1_0.html).

The clients uses dynamic client registration in order to establish trust with the provider. The client will prepare a "trust wrap". The outer entity statements does embed parent statements and signing them. This means the provider should get all needed trust data presented, and will decide whether it accepts the trust data.

Thereafter the provider will create a temporary symmetric key for the client. This key needs to be stored in a key store both at the provider and at the client. These keys will have a limited lifetime, and when they expired the client will need to perform a new registration request.

There is no persistent unique identifeir of the client. The client will get the client_id generated by the provider at the registration time, and the client may get different client_ids from the same provider over the application life time.

This framework is specific for OpenID Connect, and the trust establishment is different between the client and the provider. In contract to the provider, the client will establish trust to the provider by fetching metadata using WebFinger.



JWT Federation - a different approach

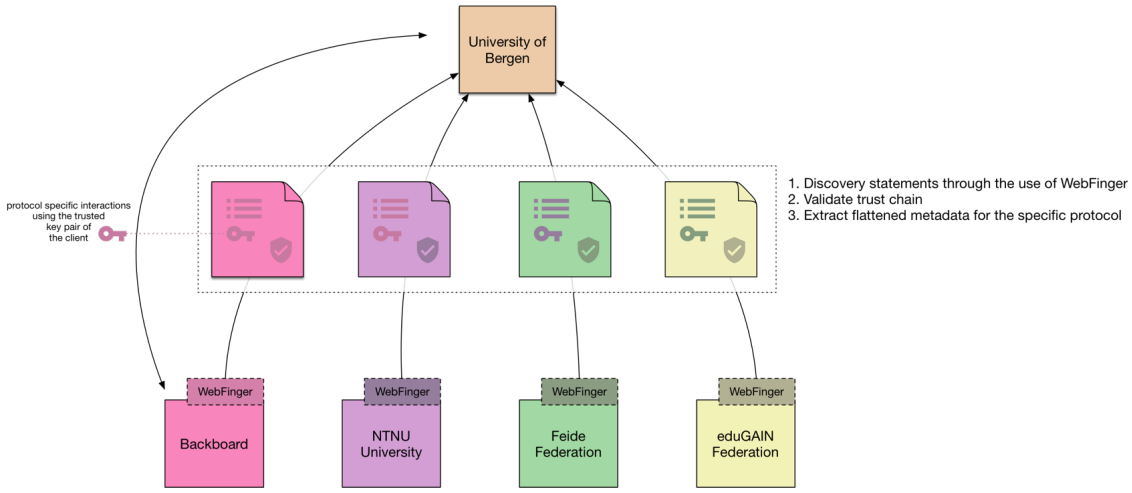
The entity statements are protocol independent, and can be extended to any number of parallell protocols, including OpenID Connect, OAuth and more.

The approach also identical for the client and the provider. The WebFinger Discovery process of OpenID Connect Providers in the core specification is extended to be used to discover any entity (including clients) not only providers.

By using the WebFinger protocol, all entities can answer these requests:

> Please state something about yourself?
>
> Which other entities do you know that can state something about you?
>
> What can you state about this entity?

When applied to OpenID Connect, there is use of client registration. When the provider receives an incomming authentication request, it will have the client_id of the client. This client_id is sufficient to perform discovery of metadata. And the provider will then subsequently fetch the whole chain of metadata, and then proceed and possibly accept the incomming authentication request.



For the protocol specific communication, the client will use the key pair reflected in the entity statement. There is no need to register a temporary symmetric key pair between the two entities.

Reasoning behind the JWT Federartions design

All entities can be discovered the same way, using WebFinger.

Trust framework is not specific for OpenID Connect – the same infrastructure can be used for adding other protocols to solve more complex use cases. In particular there would be a huge need for expressing trust between entities that would like to exchange data (OAuth, integrations and more).

The russian doll wrapping of entity statements add some complex, and no added security nor functionality.

In the OpenID Connect federations there is no built in support for an entity to fetch a statements from its trust parent, so this needs to be done manually or with another protocol. Since these statements will be short lived, this would need to automated, and there needs to be an additional layer of message exchange beyond the OIDC spec. JWT federations involves the distribution and exchange of updated trust statements, so there will be no need for yet another protocol.

The fact that all entities can be referred to with a globally unique identifier is very useful when deploying real life federations.

Because only the seeking entity knows its own trust root, it is better if the seeking entities attempts to discover the trust path, instead of the target entity presents what it thinks is a mathing trust chain. The future holds complex federation structures, and JWT Federations supports multiple simulatouly complex networks of trust that should be automatically discovered at runtime.

Managing state in both clients and providers for temporary symmetric keys adds cost and complexity. Caching keys that times out, and making sure that all involved entities has the same perception of the validity period is complex.

Reflecting updated configuration would need that the client performs a new client registration procedure against all its providers.

Promoting more use of asymmetric crypto in OpenID Connect is a good thing. It increases security, and makes it easier to deploy more secure tokens, such as sender constrained tokens and the use of signed requests as an alternative to bearer tokens.

Out of band validation of trust chain will simplify implementation and make troubleshooting possible. When entity statments are available indepedent of an actual user authentication, it would be possible to prefetch and validate trust chain and populate a software component with prevalidated metadata. Also it would be easier to make tools and utilities for troubleshooting, diagnosing and monitoring when these tools can run idenpendently from actual authenticaiton flows.

OAuth.no · UNINETT AS (http://www.uninett.no) · Andreas Åkre Solberg (https://no.linkedin.com/in/andreassolberg) · erlang (https://twitter.com/@erlang)