Science and Technology Facilities Council

Scientific Computing

Identity, Authentication, Authorization

Thematic CSC – Security

Tom Dack – STFC UKRI

CERN School of Computing

Who am I?

I currently wear multiple Authentication and Authorization hats…

- Service Manager for the Identity and Access Management service for the U.K. IRIS Collaboration
- Current Authorization Working Group Chair for the WLCG
- Leading Identity Management Community of Practice within the Square Kilometre Array Observatory's Science Regional Centre Network project

Working to ensure that all these communities (and others!) can interoperate

Questions? Feel free to contact at:
thomas.dack@stfc.ac.uk

UKRI Science and Technology Facilities Council
Scientific Computing

WLCG
Worldwide LHC Computing Grid

SKAO

Agenda

CERN School of Computing

Take Aways

- What is the difference between Authentication and Authorization
- How the global research community is connected through shared use of digital identities and understanding of why this is important for research
- An understanding of some of the different tools and methods involved

If you are developing a service that needs authentication or authorization, come back and look at this!

Introduction to Identity, Authentication, and Authorization,

…and why does it matter anyway

thematic CSC
CERN
School of Computing

What's an Identity?

Identity is…

- Concept of you as an individual – a collection of information that describes you.
- Clear in "real life" but may be fuzzy online.
- At any time, your identity may be unified or may be split into personas
  - e.g. work and home email accounts, or two different site passes for different organisations

Online Identity - Then

Traditionally a user would need to create a new ”identity” to use for every new service or account they used online

- Lots of accounts, scattered through every site a user has ever signed up for
- This results in *many* username and password pairs, which in turn leads to bad security practices
  - Credentials are reused for “simplicity”, or simply forgotten and lost

Online Identity - Now

In more modern applications, the process is evolving

- Increased importance of account credibility
  - Processes and systems in place to show that an associated identity is verified
- Increased adoption of single sign-on mechanisms, using a unified identity
  - Log-in with Social IDs, such as Facebook, Google, ORCID
- Can grant authorised access based on an identity’s attributes
  - Home institute, email address, etc

Authentication & Authorization
Or, Not letting everyone in

- Letting everyone access everything is often *a bad idea*

- Though this is not always true – the level of access control required involves considering the risks

- In the contest of research, we probably don't want anyone and everyone online being able to get in…

thematic CSC
CERN School of Computing

Authentication & Authorization: Controlling Access

Controlling access depends on:

- Verifying a user’s access to an account or identity – **authentication (AuthN)**
- Knowing that a user is allowed to do what they want to do – **authorization (AuthZ)**

These processes are usually combined, as key processes in a community’s **Authentication and Authorization Infrastructure - AAI**

SIDE NOTE:

- **AuthoriZation** - 🇺🇸 & 🇨🇦
- **AuthoriSation** - 🇬🇧 & the rest of the English-speaking world

I will have typed **AuthoriZe**, for consistency (and less Microsoft Rage) except in occurrences where it is a **named term** or **attribute**

… also I use the Americanised shorthand, **AuthZ**

thematic CSC
CERN
School of Computing

What makes AAI important for research?

A couple of key concepts underpin the use of identity in research:

- Confidentiality
- Traceability
- Suspension
- Attribution

What makes AAI important for research?

A couple of key concepts underpin this:

- Confidentiality
- Traceability
- Suspension
- Attribution

Confidentiality

- Whilst final research outputs are public, maintaining confidentiality before this point is important
- Particularly important for fields handling personal and medical data

What makes AAI important for research?

A couple of key concepts underpin this:

- Confidentiality
- Traceability
- Suspension
- Attribution

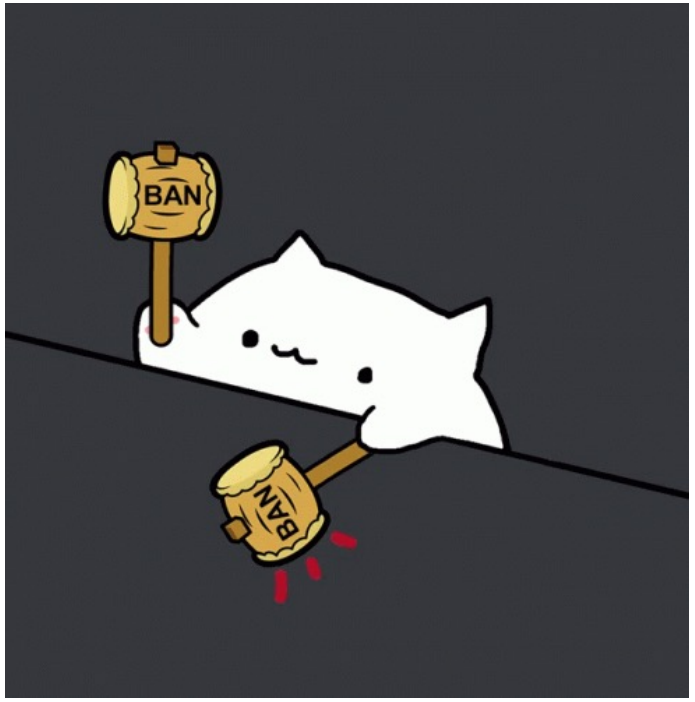
Traceability

- If something goes wrong – accidentally or maliciously – service owners need to be able to trace where this happened
- Knowing which account caused an issues is important for both support or suspension

What makes AAI important for research?

A couple of key concepts underpin this:

- Confidentiality
- Traceability
- Suspension
- Attribution



Suspension

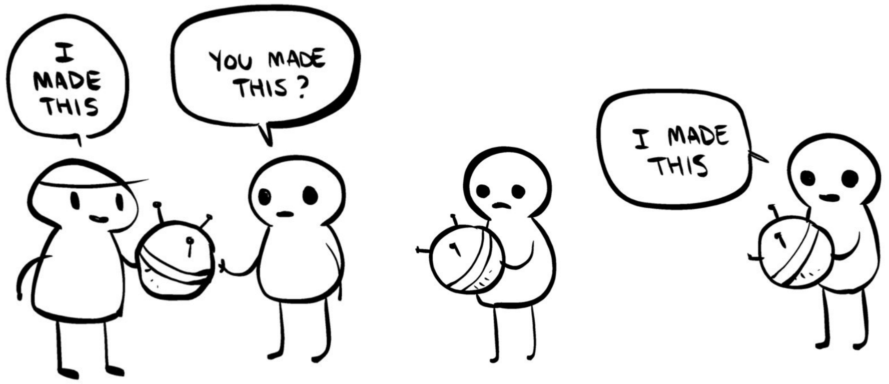
- Processes to suspend an individual user account in case of compromise or malicious activity
- Avoids downtime caused by stopping an entire service or resource, by isolating the problematic identity

What makes AAI important for research?

A couple of key concepts underpin this:

- Confidentiality
- Traceability
- Suspension
- Attribution →

- A single central identity provides a mechanism through which research can be attributed
- This helps to avoid problems arising from Identity changes – name changes from marriage, gender transition, etc
- ORCID is an example of this – life-long identifiers for researchers, which can be attached to publications, grant requests, etc



Tumblr – The Internet by Nedroid / January 30th, 2013

Not come across ORCID?
Visit *https://blog.inspirehep.net/2015/04/what-is-orcid-and-how-can-it-help-you/*
and https://home.cern/cern-people/updates/2018/01/get-yourself-orcid

AuthN & AuthZ For Distributed Research

thematic CERN School of Computing

AuthN & AuthZ for Distributed Research

Providing global access to computing resources – not easy!

- Global user community, with many members
- Distributed single infrastructure
- Not guaranteed that users know each other
- Not guaranteed that users will ever meet

Need a system for provisioning access, to be trusted across the grid

• *Not guaranteed that users know each other*

Who can you trust to know the users, in order to **authenticate** them?

Many options, including:
- The Infrastructure
- The Experimental Group or Research Community
- The Home Organisations
- A trusted Third Party

In most cases, a user's **Home Organisation** may have the most current information – especially if their access is a function of their affiliation.

Something you trust to know users is an Identity Provider (IdP)

thematic CSC CERN School of Computing

• ***Not guaranteed that users will ever meet***

But! A user's **Experimental Group or Research Community** may be better placed to tell you…

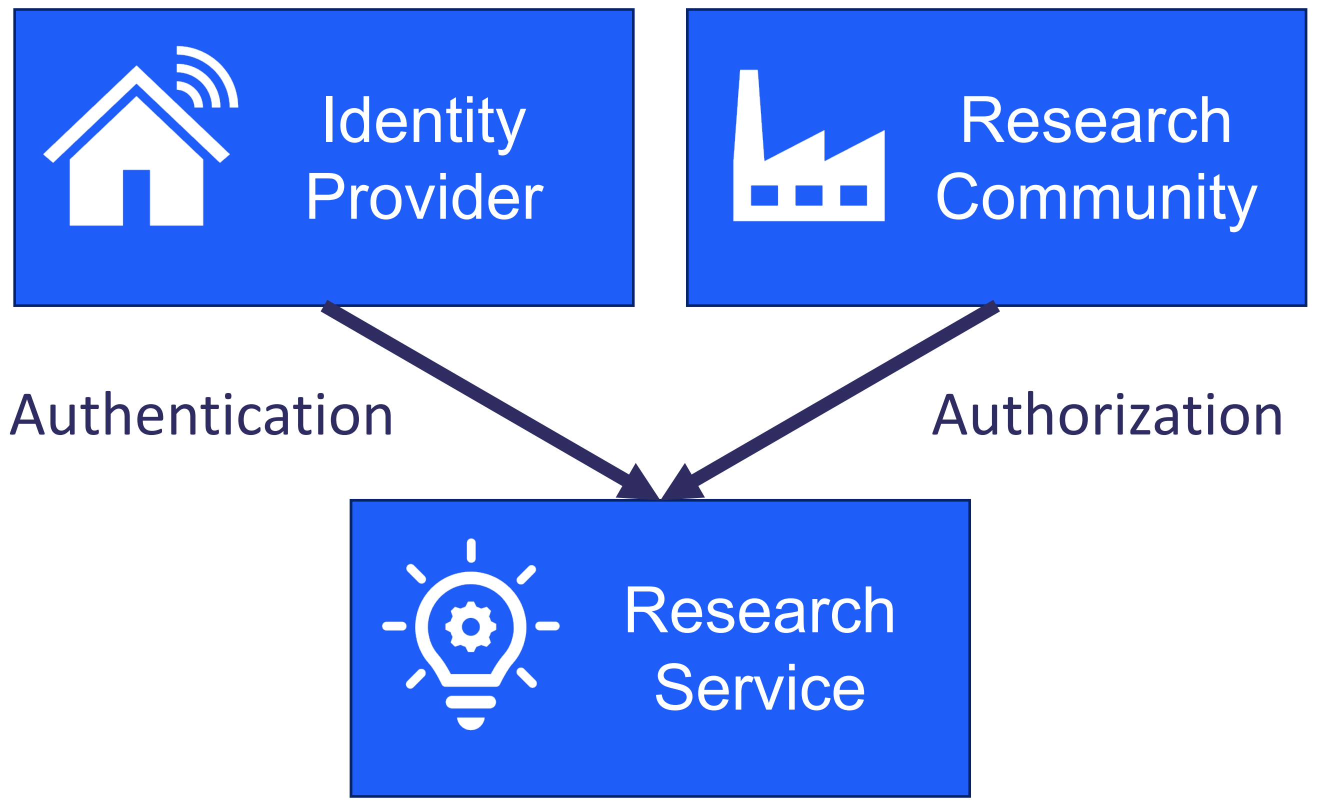
- Which group the user belongs to
- What roles (permissions) they should have
  - *what are they **authorised** to do*
  - *Are they a user, an admin, a super-user, etc?*
- The status of a user's policy acceptance – e.g. whether they have accepted the latest acceptable usage policy

thematic CSC
CERN
School of Computing

Putting the Pieces Together…

In distributed computing, we need to work with both Identity Providers and Research Communities to obtain the information needed to make access control decisions.

But to do that, we need methods to communicate this info…



CERN School of Computing

How does it work?

There are multiple possibilities, we’ll focus on the three most widely used methods for distributed authentication

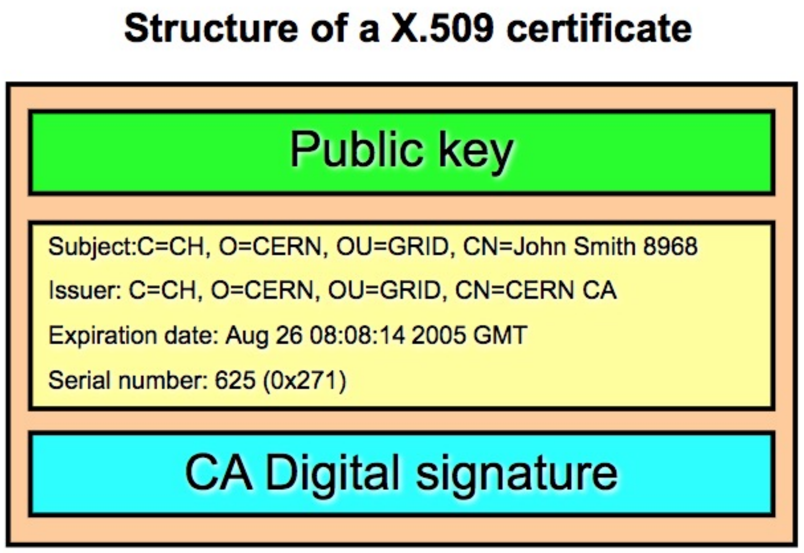
- Certificates
- SAML
    - XML bundles
- OAuth2
    - Tokens

AuthN & AuthZ Technologies: Certificates

What is a Certificate…

A Certificate is…

- A digital identity, representing an entity
  - could be a service/website, a machine, or a human individual
- Signed by a Certificate Authority (CA)
  - Self-signed certificates do exist but are not useful for situations requiring assured authentication.
  - Signed by taking a hash of the certificate, which is then encrypted with the CA's private key
- Long lived – typically a grid certificate will last a year
- User keeps an accompanying private key and password



Structure of a X.509 certificate
Public key
Subject:C=CH, O=CERN, OU=GRID, CN=John Smith 8968
Issuer: C=CH, O=CERN, OU=GRID, CN=CERN CA
Expiration date: Aug 26 08:08:14 2005 GMT
Serial number: 625 (0x271)
CA Digital signature

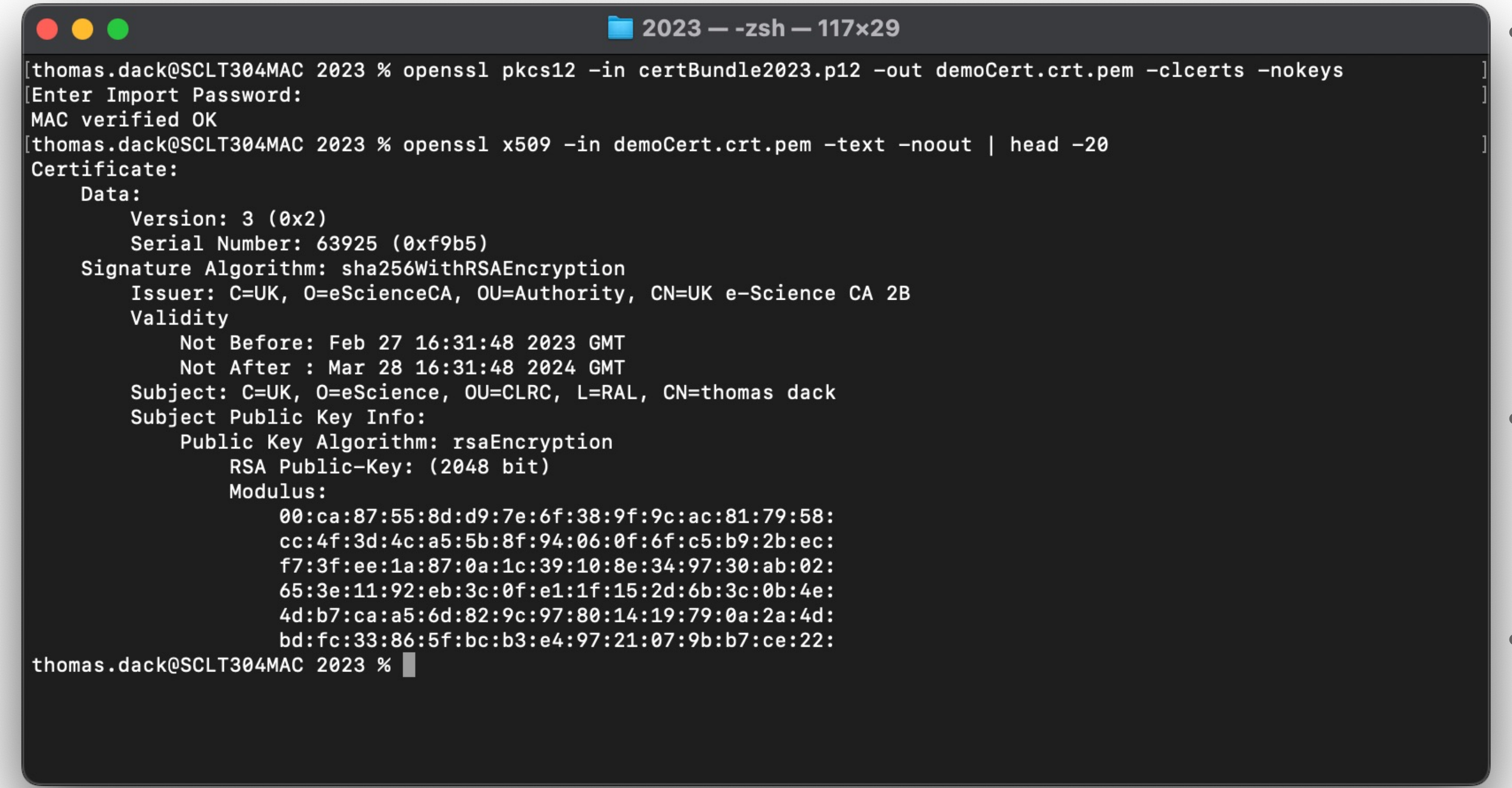
http://slideplayer.com/slide/10176602/

thematic CSC CERN School of Computing

Certificates for Research

- Certificates have been used for research since the early 2000s
- Certificate Authorities (CAs) regulated by the Interoperable Global Trust Federation (IGTF)
  - Signed by CA **IF** they can validate the identity
  - X509 is the form of certificate used in the Grid
- Authentication = Certificates
- Authorization = Certificate Extensions

Taking a peek inside…



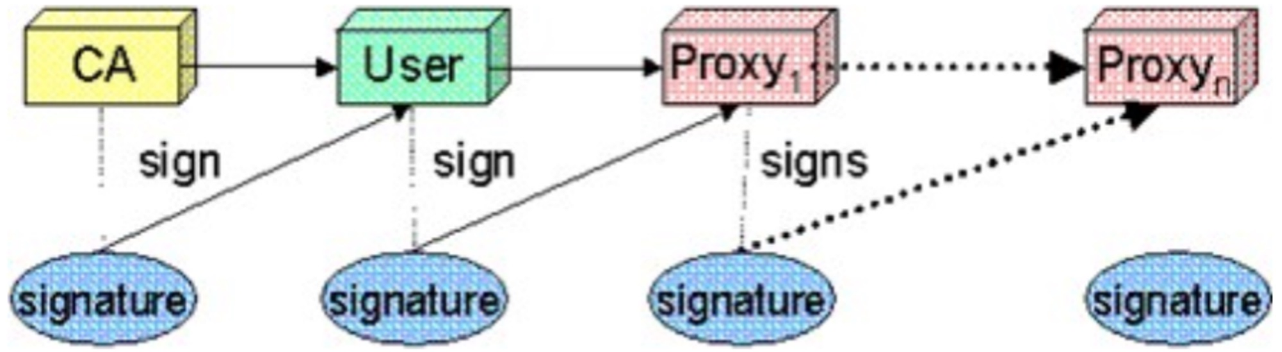
```
thomas.dack@SCLT304MAC 2023 % openssl pkcs12 -in certBundle2023.p12 -out demoCert.crt.pem -clcerts -nokeys
Enter Import Password:
MAC verified OK
thomas.dack@SCLT304MAC 2023 % openssl x509 -in demoCert.crt.pem -text -noout | head -20
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 63925 (0xf9b5)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=UK, O=eScienceCA, OU=Authority, CN=UK e-Science CA 2B
        Validity
            Not Before: Feb 27 16:31:48 2023 GMT
            Not After : Mar 28 16:31:48 2024 GMT
        Subject: C=UK, O=eScience, OU=CLRC, L=RAL, CN=thomas dack
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:ca:87:55:8d:d9:7e:6f:38:9f:9c:ac:81:79:58:
                    cc:4f:3d:4c:a5:5b:8f:94:06:0f:6f:c5:b9:2b:ec:
                    f7:3f:ee:1a:87:0a:1c:39:10:8e:34:97:30:ab:02:
                    65:3e:11:92:eb:3c:0f:e1:1f:15:2d:6b:3c:0b:4e:
                    4d:b7:ca:a5:6d:82:9c:97:80:14:19:79:0a:2a:4d:
                    bd:fc:33:86:5f:bc:b3:e4:97:21:07:9b:b7:ce:22:
thomas.dack@SCLT304MAC 2023 %
```

- Here, I have downloaded my "certificate bundle" from the UK Certificate Authority – this contains the certificate and a private key, protected by a password
- The first command extracts the certificate from the bundle
- The second uses the OpenSSL tool to look inside – you can see information about the issuer, the UK CA, and the subject, me

thematic CERN School of Computing

Proxy Certificates

- The WLCG does not use the original user certificate for job submission – instead, a proxy is generated.
- The user certificate is used to generate and sign a
  - Proxy Certificate
    - Identity of the user
    - Short lived
    - Expiration time
  - Private key
    - No password
    - Readable only by the user
- Proxy and its private key are sent off together and can generate new proxies



http://slideplayer.com/slide/10176602/

Why Generate a Proxy Certificate and not send the original?

We use proxies because:

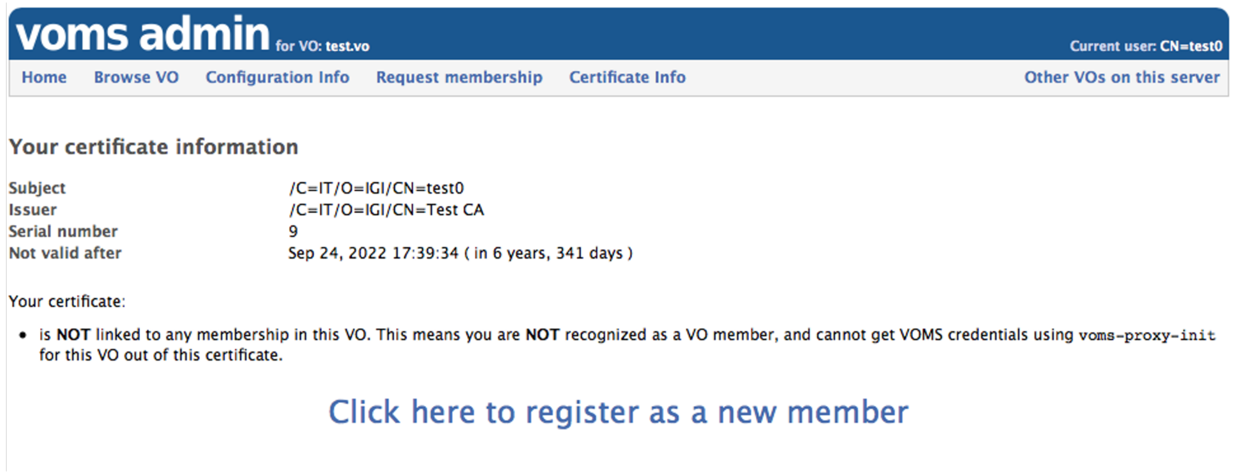
1. they have a shorter life time. If they are compromised the damage is limited. In addition, the private key of a certificate is protected by a password known only to the user - it's critical that that information remain private and so the user private key can't be used for signing by a system.
2. By giving a proxy to a service, that service can keep regenerating the proxy and acting like you. You don't want to give them all of your rights, just the one relevant for that particular service

thematic CSC
CERN School of Computing

The WLCG Example: VOMS Proxy Certificates

Already complex, and that was just for Authentication - we now need a way to add authorization

Taking WLCG as an example:
- Authorization rules are stored in the Virtual Organisation Management System (VOMS)
- A Virtual Organisation is an experiment or research community in our context.
- Who has registered with, and is known to, VOMS?



Your certificate information

Subject	/C=IT/O=IGI/CN=test0
Issuer	/C=IT/O=IGI/CN=Test CA
Serial number	9
Not valid after	Sep 24, 2022 17:39:34 (in 6 years, 341 days)

Your certificate:

- is **NOT** linked to any membership in this VO. This means you are **NOT** recognized as a VO member, and cannot get VOMS credentials using `voms-proxy-init` for this VO out of this certificate.

Click here to register as a new member

- **Grid Proxy** = Short lived certificate to be used for authentication to grid services
- **VOMS Extension** = Virtual Organisation specific information, e.g. role and capability
- **VOMS Proxy = Grid Proxy + VOMS Extension**

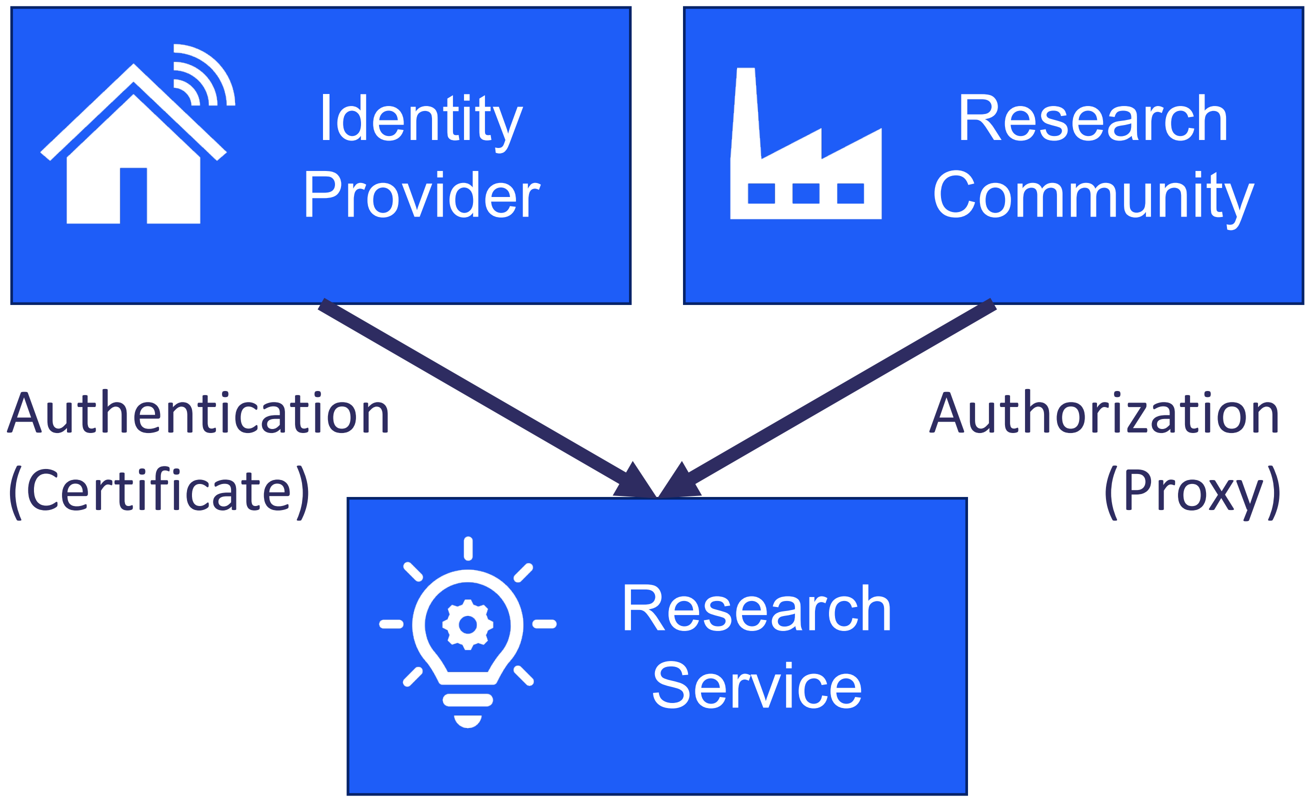
Putting the pieces together… Certificates

Good Bits

- Well established technology, services are set up to accept certificates
- Same credential valid for web and non-web

Bad Bits

- Security impact if compromised (and frequently compromised)
- Not user friendly
- Mobility issues



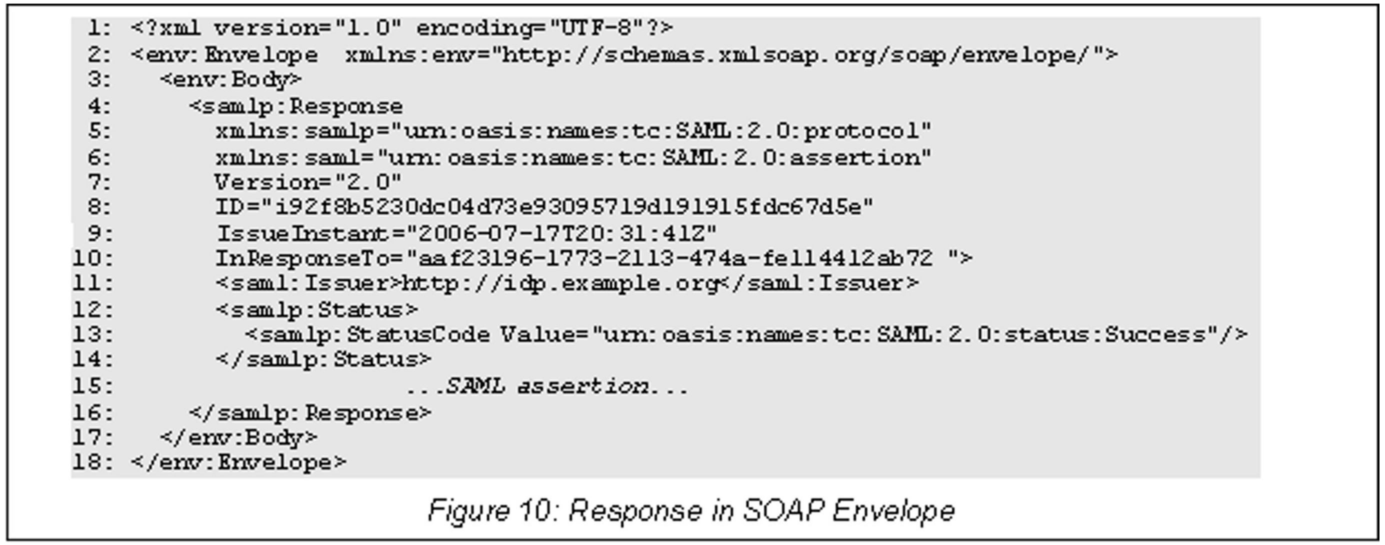
CERN School of Computing

AuthN & AuthZ Technologies: SAML

thematic CSC
CERN School of Computing

Security Assertion Markup Language

- Often used for Single-Sign-On implementations
- Historically used by the Research and Education sector
- Limited to web services
- Authentication assertions sent as XML packets
  - Can be encrypted or not
  - Contain user attributes, can contain authorization information



```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope  xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Body>
    <samlp:Response
      xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
      xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
      Version="2.0"
      ID="i92f8b5230dc04d73e93095719d191915fdc67d5e"
      IssueInstant="2006-07-17T20:31:41Z"
      InResponseTo="aaf23196-1773-2113-474a-fe114412ab72 ">
      <saml:Issuer>http://idp.example.org</saml:Issuer>
      <samlp:Status>
        <samlp:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
      </samlp:Status>
              ...SAML assertion...
    </samlp:Response>
  </env:Body>
</env:Envelope>
```

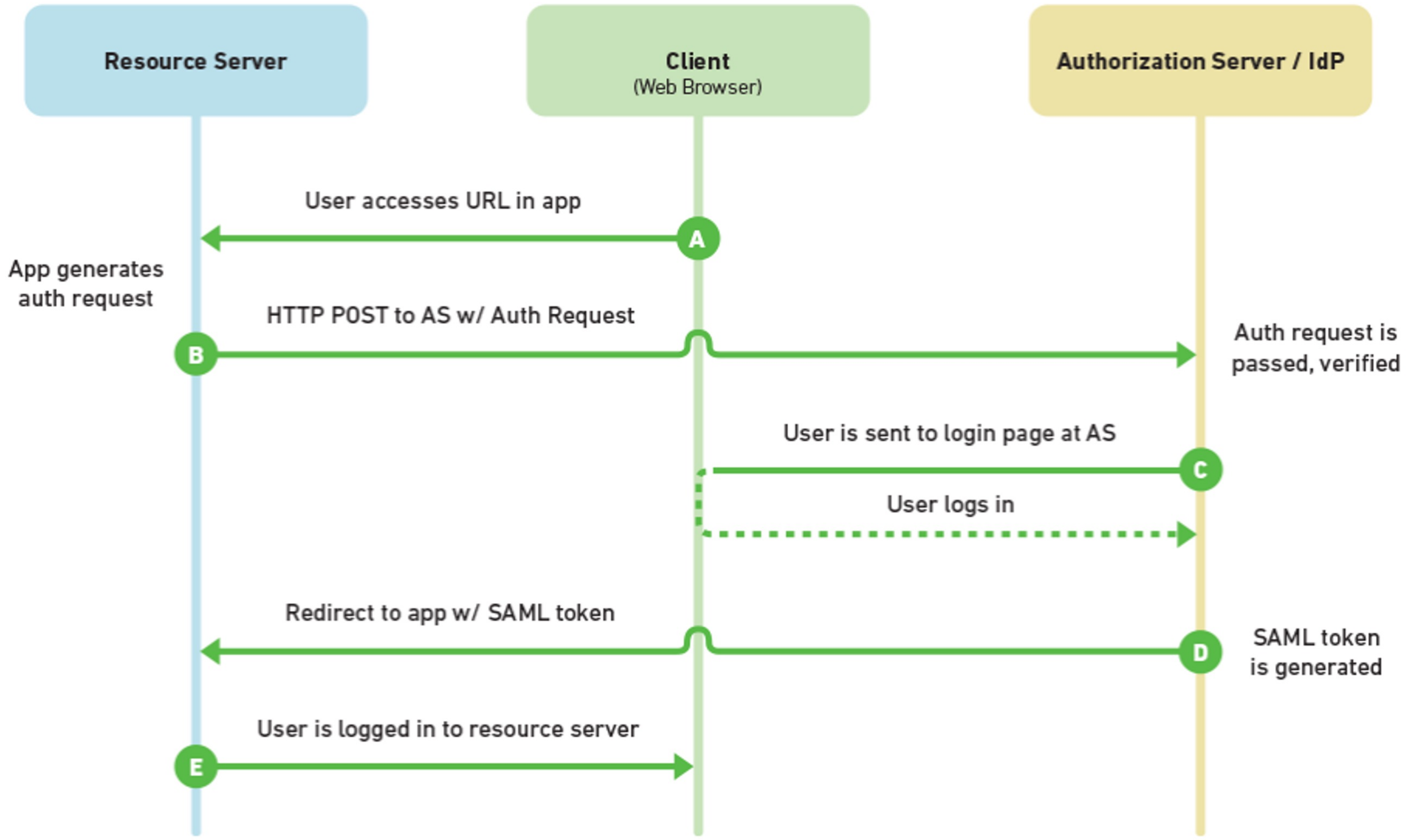
Figure 10: Response in SOAP Envelope

http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html

thematic CERN School of Computing

SAML Protocol

- Client
  - User on their browser
- Resource Server
  - A website requiring authentication
- Authorization server/IdP
  - Home Organisation


Resource Server
Client
(Web Browser)
Authorization Server / IdP
User accesses URL in app
A
App generates auth request
HTTP POST to AS w/ Auth Request
B
Auth request is passed, verified
User is sent to login page at AS
C
User logs in
Redirect to app w/ SAML token
D
SAML token is generated
User is logged in to resource server
E

thematic CERN School of Computing

https://www.mutuallyhuman.com/blog/2013/05/09/choosing-an-sso-strategy-saml-vs-oauth2/

SAML Trust Federations

A group of Service Providers and Identity Providers that have agreed to work together.

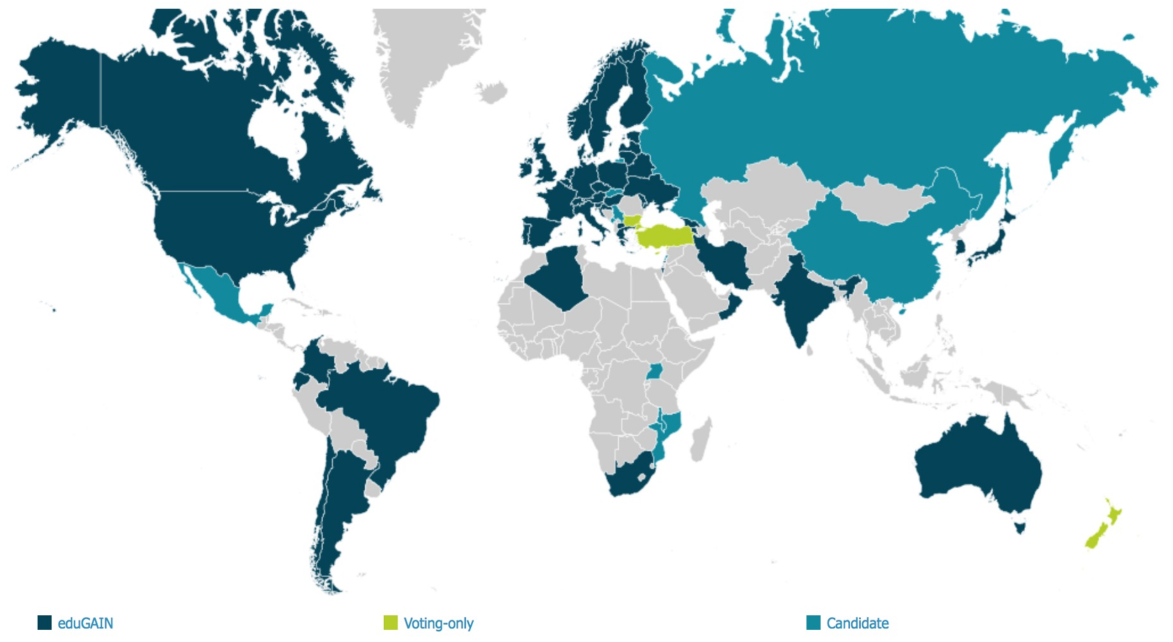
- Federation metadata collects XML descriptions of each organisation, along with their certificate
- Federation metadata is signed by the Federation and distributed to all members
- Everyone has access to everyone’s certificates, issued by a trusted source

CERN School of Computing

Federation of Federations: Interfederation

The big example: eduGAIN

“The eduGAIN service interconnects identity federations around the world, simplifying access to content, services and resources for the global research and education community. eduGAIN enables the trustworthy exchange of information related to identity, authentication and authorization (AAI). ”



https://technical.edugain.org/

thematic CSC
CERN School of Computing

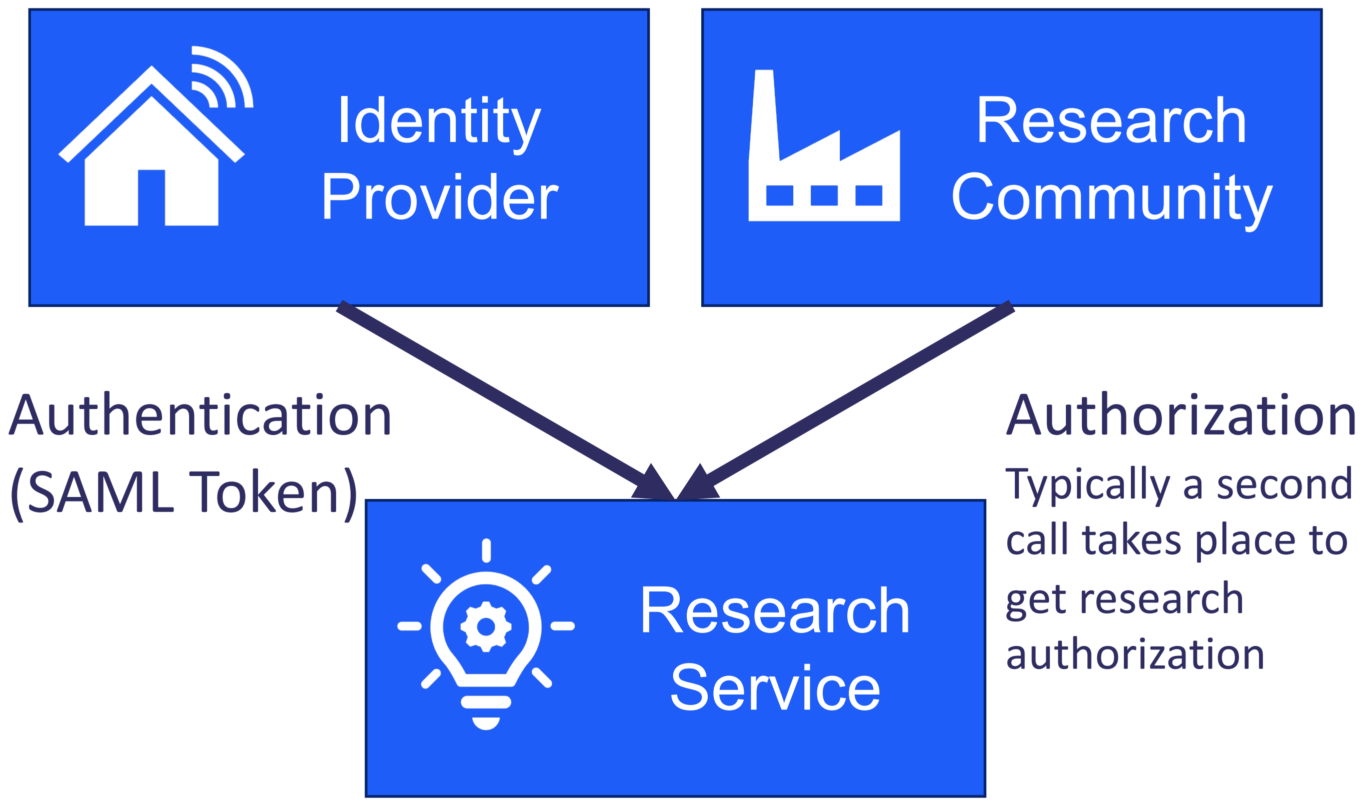
Putting the pieces together… SAML

Good Bits

- Mature, scalable federations
- Secure protocol

Bad Bits

- Only works for web services
- Significant implementation effort


Identity Provider
Research Community
Authentication (SAML Token)
Authorization
Typically a second call takes place to get research authorization
Research Service

AuthN & AuthZ Technologies: OAuth & OIDC Tokens

OAuth and OIDC Tokens

There are a few components in a "token" flow. Key to this, a token is…

- a JSON Web Token
  - "JWTs are an open, industry standard RFC 7519 method for representing claims securely between two parties." https://jwt.io/introduction/
  - Tokens are encoded strings of data, issued by an issuer with which the client (receiver) has a trusted relationship.
- In this context, JWTs are used to communicate authentication and authorization information using the **OAuth 2.0** and **OIDC** protocols

OAuth 2.0	OIDC
Open **Auth**orization	Open **ID** **C**onnect

Oauth 2.0 – the Access Token

- An open standard for access delegation, and most often used for allowing users to grant a website access to their information on another website, without giving them their password
    - Examples include signing into a third-party website using your Google, GitHub, or Orcid account
- An access token is provided to the third party, from whichever service is acting as the Authorization Server, which it can then use to retrieve a protected resource

Want to know more about OAuth after the lecture?

https://oauth.net/

Oauth 2.0 – Terminology

- **Protected Resource**
  - The identity or data which is to be shared
  - *eg: your email address, or the ability to post to your Twitter*
- **Resource Owner**
  - The user who owns the **Protected Resource**
  - *eg: you!*
- **Client**
  - The application that wants access on behalf of the **Resource Owner**
  - *eg: the website you want to register with using your Google identity*
- **Authorization Server**
  - The application which knows the **Resource Owner**, and where the **Resource Owner** already has an account
  - *eg: The Google, Twitters, Orcids of the world*
- **Resource Server**
  - Where **the Protected Resource** lives, and what the **Client** wants to use
  - *eg: the API from which the **Client** can access the **Protected Resource***

Owner

Client

AuthZ

Resc

thematic CSC
CERN
School of Computing

Oauth 2.0 – Terminology

- **Protected Resource**
  - The identity or data which is to be shared
  - *eg: your email add*
- **Resource Owner**
  - The user who own
  - *eg: you!*
- **Client**
  - The application tha
  - *eg: the website yo*
- **Authorization Serve**
  - The application wh has an account
  - *eg: The Google, Twit*
- **Resource Server**
  - Where **the Protec**
  - *eg: the API from w*

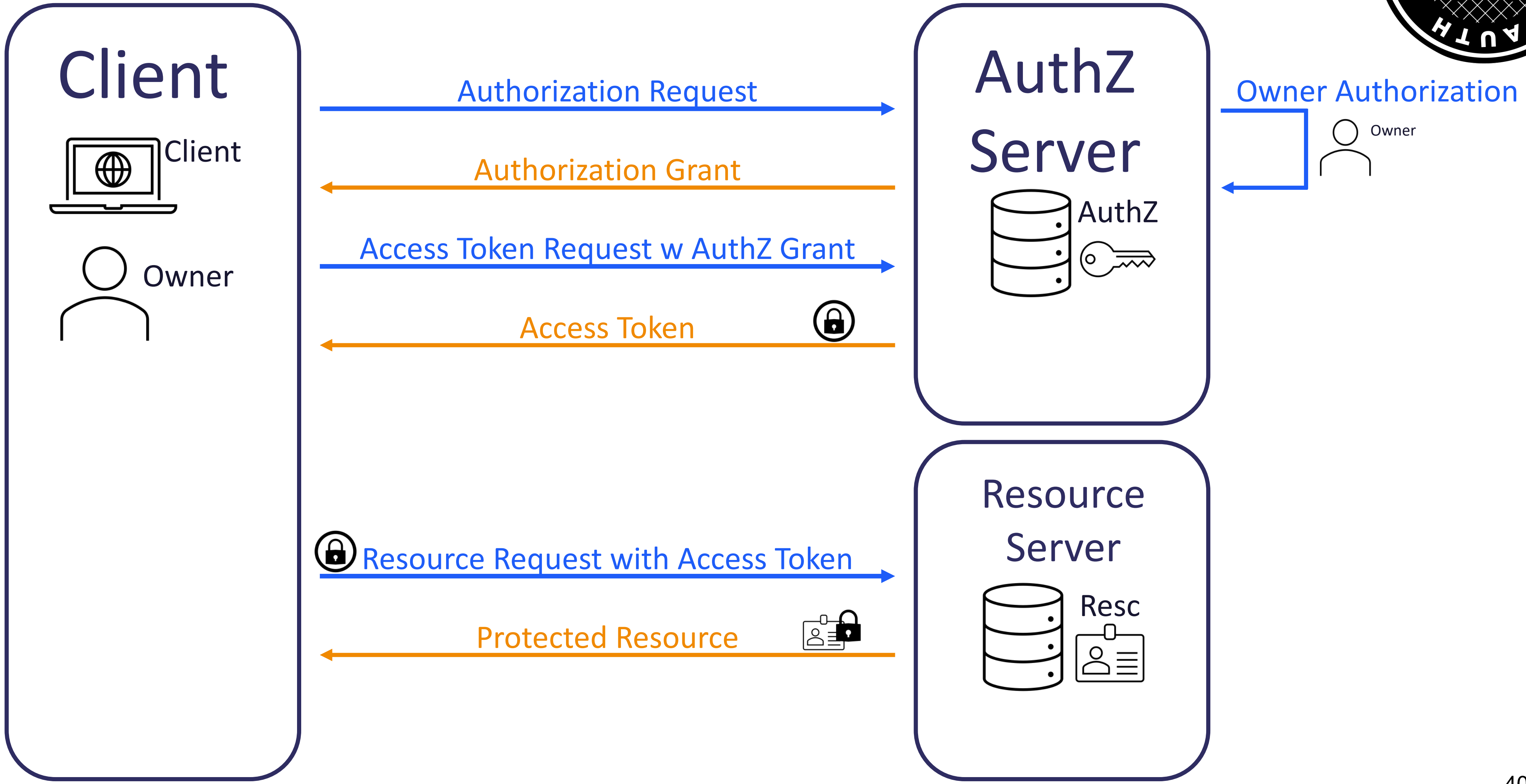
A quick aside…
OAuth 2.0 defines two types of clients:
Confidential and **Public Clients**

- Confidential applications can hold credentials with which they use to authenticate themselves to the AuthZ Server in a secure way. They require a trusted backend server to store the secret(s).
- *eg – a web application with a secure backend*

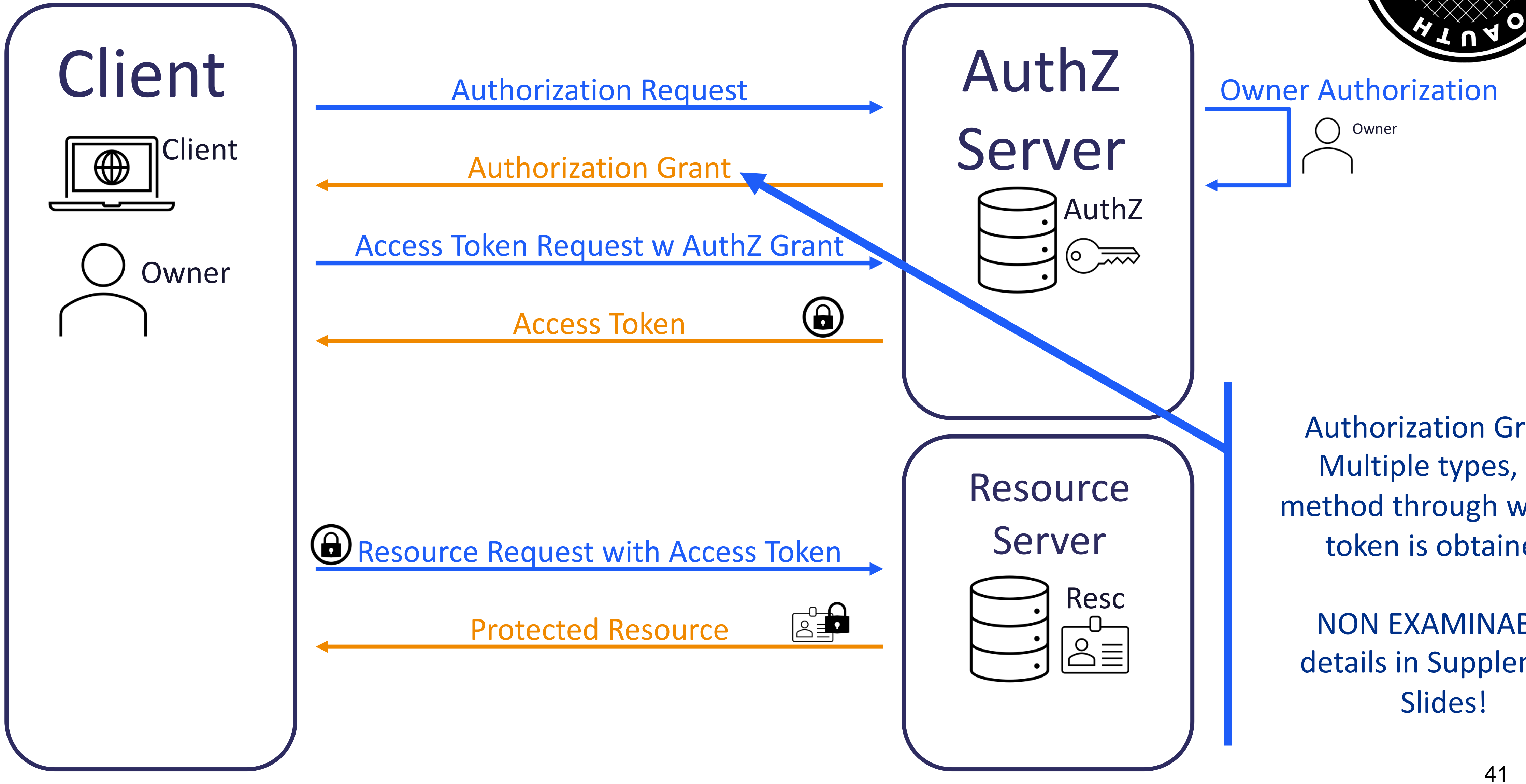
- Public clients **cannot** hold credentials securely.
- *eg – a native desktop or mobile application, or a JavaScript-based client-side web application (single-page app)*

thematic CSC
CERN School of Computing

OAuth 2.0 – How things work



OAuth 2.0 – How things work



Authorization Grant? Multiple types, the method through which a token is obtained.

NON EXAMINABLE: details in Supplement Slides!

Oauth 2.0 – the Access Token

- OAuth is an **Authorization** flow – does not have a method for user **Authentication**

- Enter…

OpenID Connect: OIDC

OIDC: The ID Token

- **OpenID Connect (OIDC)** is an identity layer on top of the base OAuth 2.0 protocol
- OIDC provides the ability for app and web developers to authenticate users without the need to directly store credential sets
- OIDC enables the Authorization Server to act as an **Identity Provider (IdP)**. An OAuth 2.0 Authorization Server implementing OIDC is also referred to as an **OpenID Provider (OPs).**
- A client which uses an OP for authentication is a **Relying Party (RPs)**.
- OIDC identifies a set of personal attributes that can be exchanged between Identity Providers and the apps that use them, and includes an approval step so that users can consent (or deny) the sharing of this information

Want to know more about OIDC after the lecture

https://openid.net/connect/

OIDC: The ID Token

- In OAuth flows involving a user, the user will authenticate with the Authorization Server before providing consent for the server to release information to the client. OpenID Connect utilises this process to authenticate to the client
- The Token Endpoint will provide two separate tokens: a standard OAuth Access Token, and the OIDC ID Token.
- The ID Token will contain information about the user pertaining to what the client has requested
- When the client receives the ID Token, it may then read off requested claims

OIDC: Requesting Info with Scopes

- When the client makes its token request, it must use **Scopes** to specify which privileges are being requested in the token
- In OAuth 2.0, Scopes correspond to what resources are available when a protected resource is accessed
- In OIDC, Scopes correspond to the specific sets of information to be made available as Claim Values
- OIDC defines a set of standard Scopes, but does allow additional Scope values to be defined and used – see the supplement slides for examples!

An Example Token Request

`https://example.com/oauth/auth?`	➡ The Token Endpoint Address
`response_type=code`	➡ Authorization Grant requested
`&scope=openid%20profile%20email`	➡ The Scopes Requested
`&client_id=EXAMPLE_CLIENTID`	➡ Unique ID of the Client
`&redirect_uri=EXAMPLE_REDIRECT_URI`	➡ The URL at the client where the token should be returned

OIDC: Default Token Claims

as defined by OIDC Core

```
{ "iss":
 "https://server.example.com",
 "sub": ″24400320",
 "aud": "s6BhdRkqt3″,
 "exp": 1311281970,
 "iat": 1311280970,
 "auth_time": 1311280969″
}
```

Claim	Summary
`iss`	REQUIRED: Issuer identifier for who issued the response. This is a case sensitive URL
`sub`	REQUIRED: Subject identifier. A locally unique identifier which must never be reassigned, this identifies the user within the issuer. A case sensitive string ≤ 255 ASCII characters in length
`aud`	REQUIRED: Audiences that this token is intended for, using the Client ID as its identifying value.
`exp`	REQUIRED: Expiration time for the token, after which it MUST NOT be used for processing.
`iat`	REQUIRED: Time at which the JWT was issued
`auth_time`	Time when end-user authentication occurred. If a `max_age` request is made or `auth_time` is requested as an essential claim, this is REQUIRED – otherwise it is OPTIONAL

CERN School of Computing

OIDC: Standard Identity Claims

as defined by OIDC Core

Claim	Summary
`name`	The end-user's full name, including all name parts. `given_name`, `family_name`, `middle_name` and `nickname` may also be used.
`preferred_username`	A shorthand name by which the user wishes to be referred to by at the RP. This MAY be any valid JSON string – including special characters such as @, /, or whitespace. The RP MUST NOT rely on this value being unique.
`email`	End-User's preferred e-mail address. Its value MUST conform to the RFC 5322 addr-spec syntax. The RP MUST NOT rely upon this value being unique.
`phone_number`	End-user's preferred telephone number.
`address`	End-User's preferred postal address.
`profile`	URL of the End-User's profile page. The contents of this Web page SHOULD be about the End-User.
… And others	`picture, website, email_verified, gender, birthdate, zoneinfo, locale, phone_number_verified, updated_at`

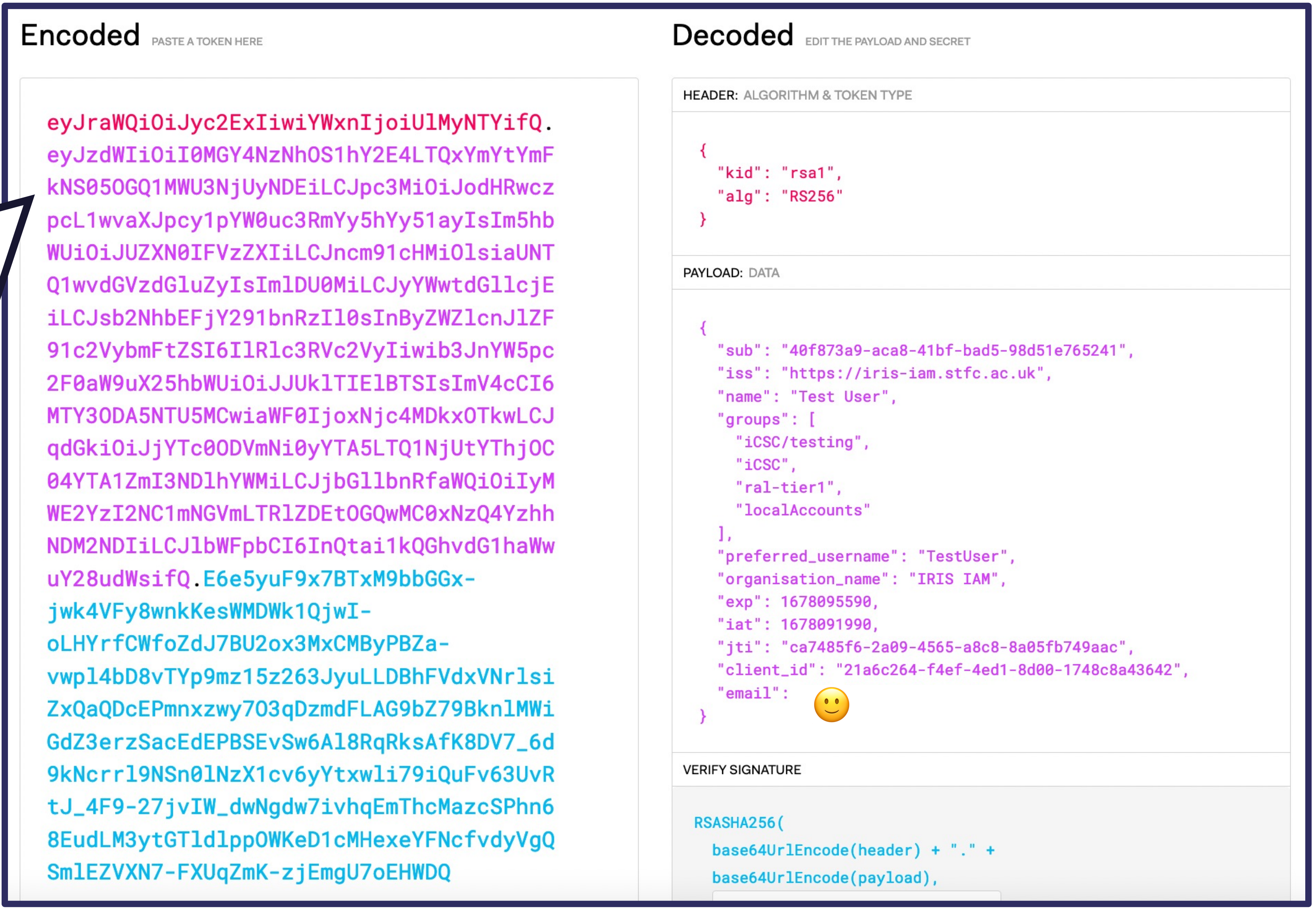
OIDC: Additional Claims

- The OpenID Connect Core specification only defines the previous small set of claims as standard
- However, OP's MAY provide additional claims about the End-User
- A typical example includes `groups`, a list of groups the user is registered with at the OP
- Any additional Claim will need to have a corresponding Scope, to allow it to be requested
  - Alternatively, extra claims could be included within the `profile` scope (not claim!) – this is how the OP to be used within the exercises provides its `groups` claim

Scope	Summary
`profile`	OPTIONAL. This scope value requests access to the End-User's default profile Claims, which are: `name`, `family_name`, `given_name`, `middle_name`, `nickname`, `preferred_username`, `profile`, `picture`, `website`, `gender`, `birthdate`, `zoneinfo`, `locale`, and `updated_at`. See supplement slides for other scopes!

OIDC: Putting this together…

The token is three Base64-URL strings separated by dots that can be easily passed in HTML and HTTP environments



Encoded PASTE A TOKEN HERE

```
eyJraWQiOiJyc2ExIiwiYWxnIjoiUlMyNTYifQ.
eyJzdWIiOiI0MGY4NzNhOS1hY2E4LTQxYmYtYmF
kNS05OGQ1MWU3NjUyNDEiLCJpc3MiOiJodHRwcz
pcL1wvaXJpcy1pYW0uc3RmYy5hYy51ayIsIm5hb
WUiOiJUZXN0IFVzZXIiLCJncm91cHMiOlsiaUNT
Q1wvdGVzdGluZyIsImlDU0MiLCJyYWwtdGllcjE
iLCJsb2NhbEFjY291bnRzIl0sInByZWZlcnJlZF
91c2VybmFtZSI6IlRlc3RVc2VyIiwib3JnYW5pc
2F0aW9uX25hbWUiOiJJUklTIElBTSIsImV4cCI6
MTY3ODA5NTU5MCwiaWF0IjoxNjc4MDkxOTkwLCJ
qdGkiOiJjYTc0ODVmNi0yYTA5LTQ1NjUtYThjOC
04YTA1ZmI3NDlhYWMiLCJjbGllbnRfaWQiOiIyM
WE2YzI2NC1mNGVmLTRlZDEtOGQwMC0xNzQ4Yzhh
NDM2NDIiLCJlbWFpbCI6InQtai1kQGhvdG1haWw
uY28udWsifQ.E6e5yuF9x7BTxM9bbGGx-
jwk4VFy8wnkKesWMDWk1QjwI-
oLHYrfCWfoZdJ7BU2ox3MxCMByPBZa-
vwpl4bD8vTYp9mz15z263JyuLLDBhFVdxVNrlsi
ZxQaQDcEPmnxzwy7O3qDzmdFLAG9bZ79BknlMWi
GdZ3erzSacEdEPBSEvSw6Al8RqRksAfK8DV7_6d
9kNcrrl9NSn0lNzX1cv6yYtxwli79iQuFv63UvR
tJ_4F9-27jvIW_dwNgdw7ivhqEmThcMazcSPhn6
8EudLM3ytGTldlppOWKeD1cMHexeYFNcfvdyVgQ
SmlEZVXN7-FXUqZmK-zjEmgU7oEHWDQ
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "kid": "rsa1",
  "alg": "RS256"
}
```

PAYLOAD: DATA

```
{
  "sub": "40f873a9-aca8-41bf-bad5-98d51e765241",
  "iss": "https://iris-iam.stfc.ac.uk",
  "name": "Test User",
  "groups": [
    "iCSC/testing",
    "iCSC",
    "ral-tier1",
    "localAccounts"
  ],
  "preferred_username": "TestUser",
  "organisation_name": "IRIS IAM",
  "exp": 1678095590,
  "iat": 1678091990,
  "jti": "ca7485f6-2a09-4565-a8c8-8a05fb749aac",
  "client_id": "21a6c264-f4ef-4ed1-8d00-1748c8a43642",
  "email":      🙂
}
```

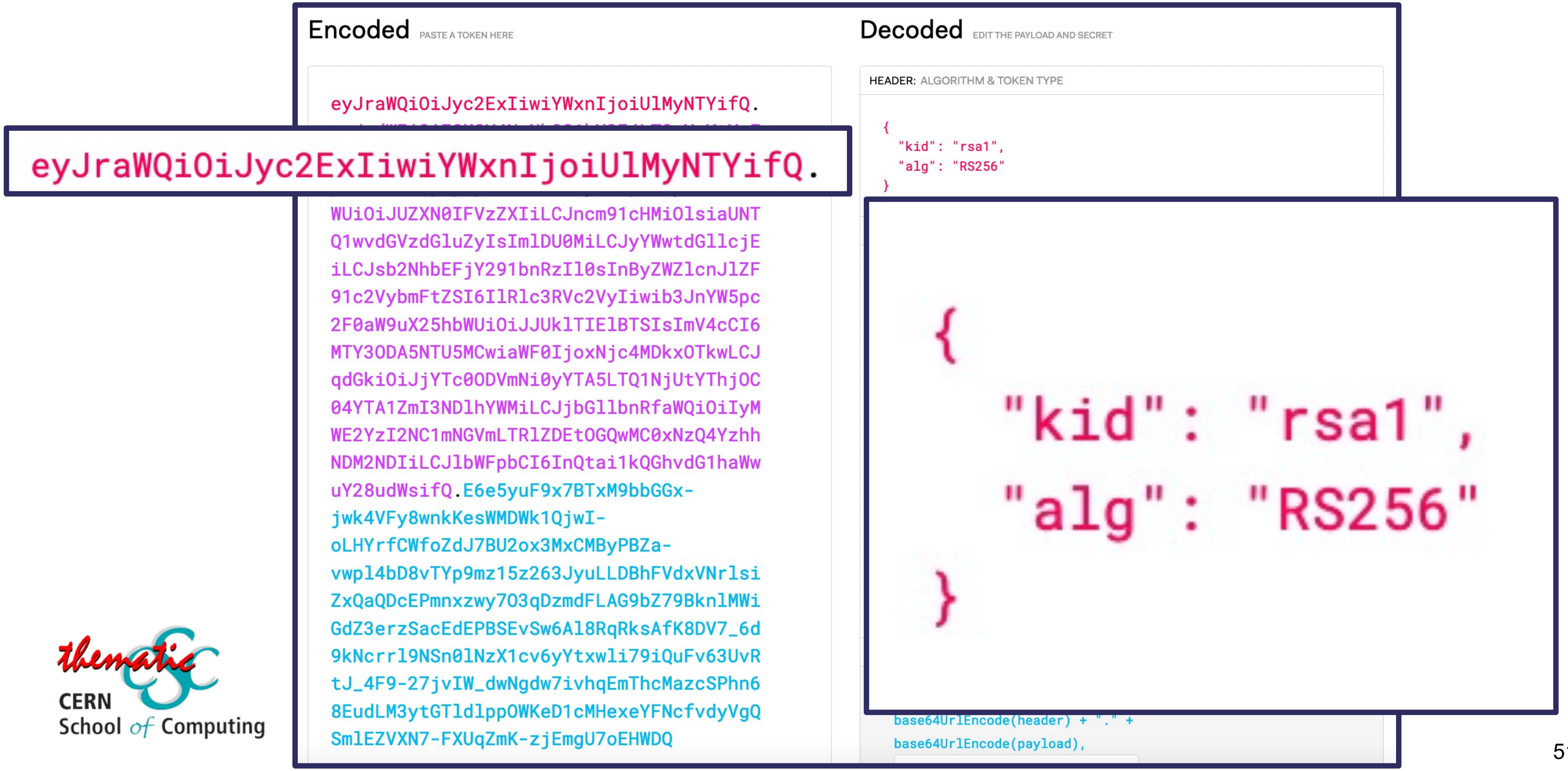
VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
```

Handy token decoding taken from: https://jwt.io/

Check it out for more info on JWTs or when using them yourself!

OIDC: Putting this together…



Encoded PASTE A TOKEN HERE

eyJraWQiOiJyc2ExIiwiYWxnIjoiUlMyNTYifQ.
WUiOiJUZXN0IFVzZXIiLCJncm91cHMiOlsiaUNT
Q1wvdGVzdGluZyIsImlDU0MiLCJyYWwtdGllcjE
iLCJsb2NhbEFjY291bnRzIl0sInByZWZlcnJlZF
91c2VybmFtZSI6IlRlc3RVc2VyIiwib3JnYW5pc
2F0aW9uX25hbWUiOiJJUklTIElBTSIsImV4cCI6
MTY3ODA5NTU5MCwiaWF0IjoxNjc4MDkxOTkwLCJ
qdGkiOiJjYTc0ODVmNi0yYTA5LTQ1NjUtYThjOC
04YTA1ZmI3NDlhYWMiLCJjbGllbnRfaWQiOiIyM
WE2YzI2NC1mNGVmLTRlZDEtOGQwMC0xNzQ4Yzhh
NDM2NDIiLCJlbWFpbCI6InQtai1kQGhvdG1haWw
uY28udWsifQ.E6e5yuF9x7BTxM9bbGGx-
jwk4VFy8wnkKesWMDWk1QjwI-
oLHYrfCWfoZdJ7BU2ox3MxCMByPBZa-
vwpl4bD8vTYp9mz15z263JyuLLDBhFVdxVNrlsi
ZxQaQDcEPmnxzwy7O3qDzmdFLAG9bZ79BknlMWi
GdZ3erzSacEdEPBSEvSw6Al8RqRksAfK8DV7_6d
9kNcrrl9NSn0lNzX1cv6yYtxwli79iQuFv63UvR
tJ_4F9-27jvIW_dwNgdw7ivhqEmThcMazcSPhn6
8EudLM3ytGTldlppOWKeD1cMHexeYFNcfvdyVgQ
SmlEZVXN7-FXUqZmK-zjEmgU7oEHWDQ

eyJraWQiOiJyc2ExIiwiYWxnIjoiUlMyNTYifQ.

Decoded EDIT THE PAYLOAD AND SECRET

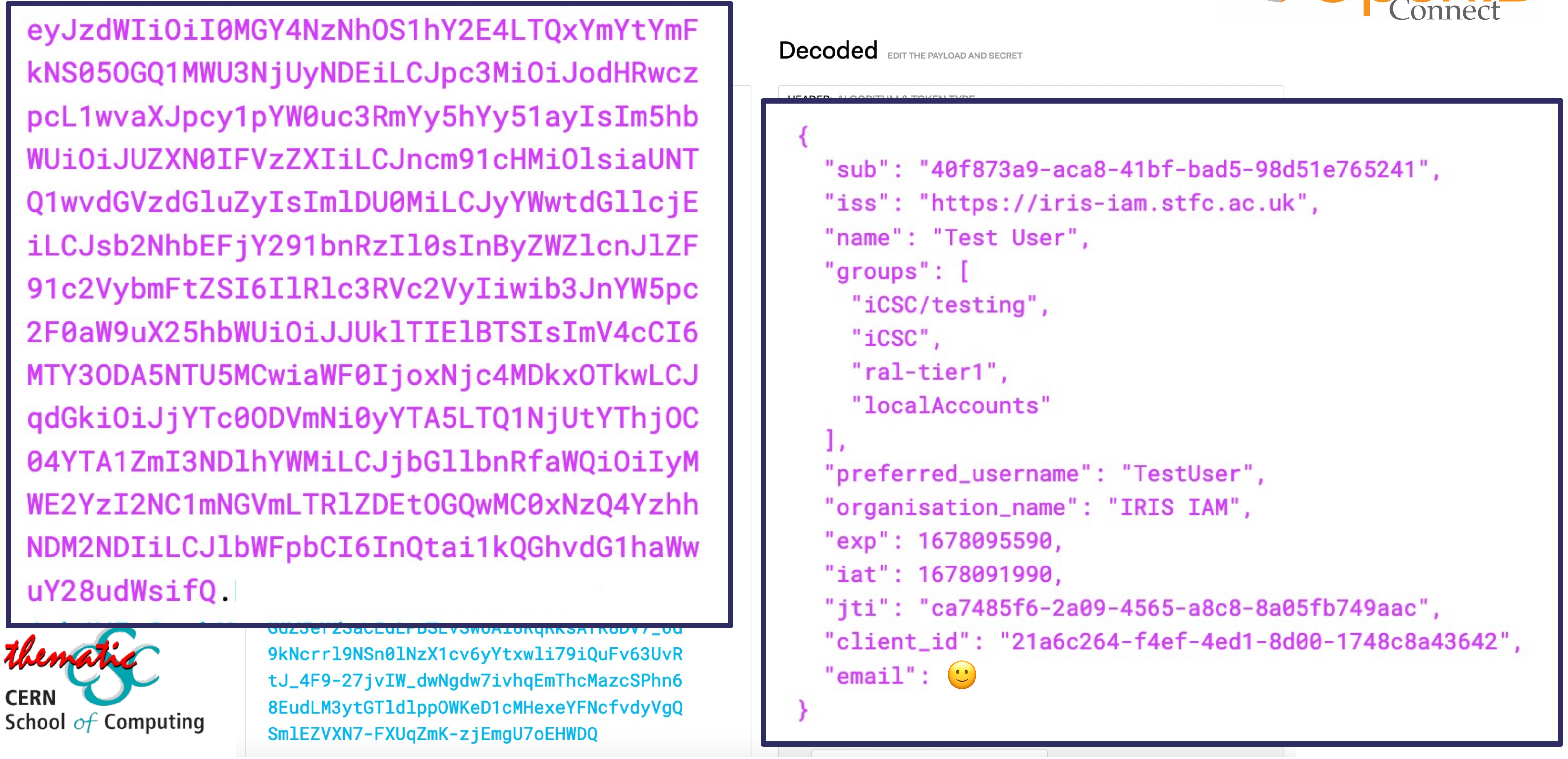
HEADER: ALGORITHM & TOKEN TYPE

```
{
  "kid": "rsa1",
  "alg": "RS256"
}
```

```
{
  "kid": "rsa1",
  "alg": "RS256"
}
```

```
base64UrlEncode(header) + "." +
base64UrlEncode(payload),
```

OIDC: Putting this together…



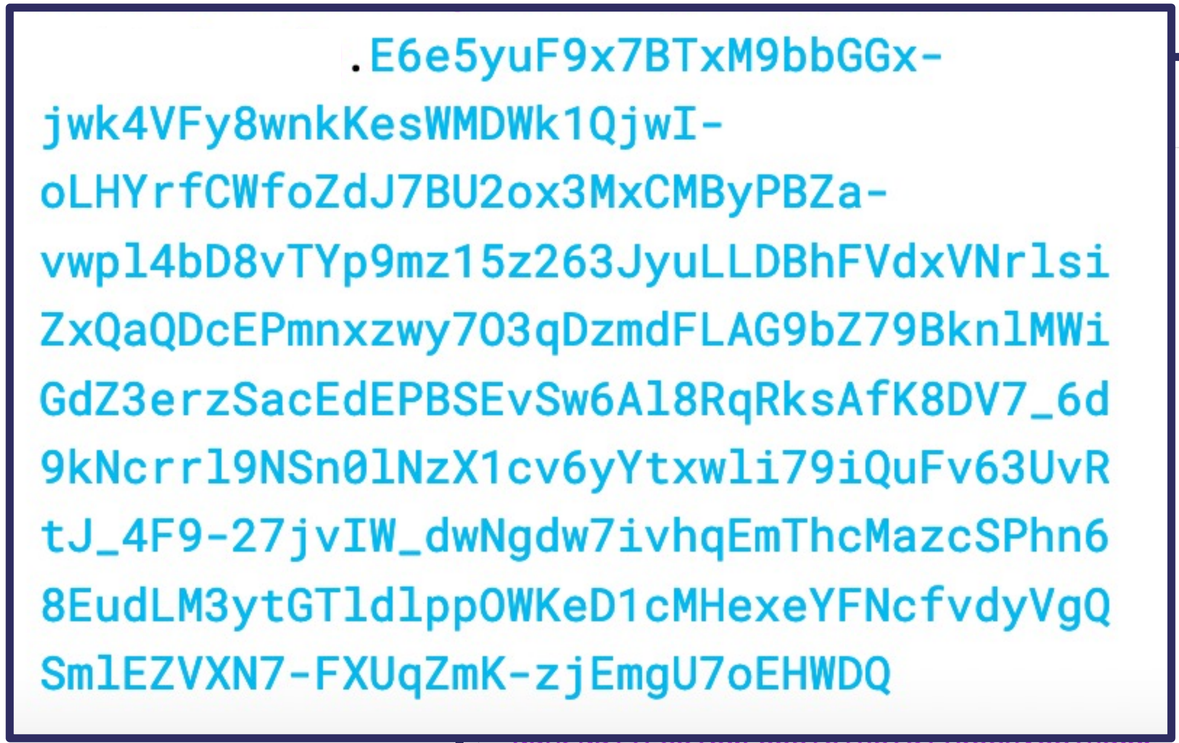
Decoded EDIT THE PAYLOAD AND SECRET

```
eyJzdWIiOiI0MGY4NzNhOS1hY2E4LTQxYmYtYmF
kNS05OGQ1MWU3NjUyNDEiLCJpc3MiOiJodHRwcz
pcL1wvaXJpcy1pYW0uc3RmYy5hYy51ayIsIm5hb
WUiOiJUZXN0IFVzZXIiLCJncm91cHMiOlsiaUNT
Q1wvdGVzdGluZyIsImlDU0MiLCJyYWwtdGllcjE
iLCJsb2NhbEFjY291bnRzIl0sInByZWZlcnJlZF
91c2VybmFtZSI6IlRlc3RVc2VyIiwib3JnYW5pc
2F0aW9uX25hbWUiOiJJUklTIElBTSIsImV4cCI6
MTY3ODA5NTU5MCwiaWF0IjoxNjc4MDkxOTkwLCJ
qdGkiOiJjYTc0ODVmNi0yYTA5LTQ1NjUtYThjOC
04YTA1ZmI3NDlhYWMiLCJjbGllbnRfaWQiOiIyM
WE2YzI2NC1mNGVmLTRlZDEtOGQwMC0xNzQ4Yzhh
NDM2NDIiLCJlbWFpbCI6InQtai1kQGhvdG1haWw
uY28udWsifQ.
```

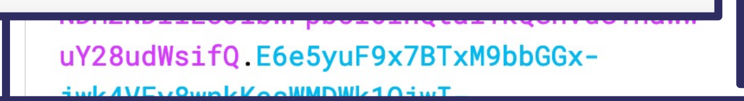
```
9kNcrrl9NSn0lNzX1cv6yYtxwli79iQuFv63UvR
tJ_4F9-27jvIW_dwNgdw7ivhqEmThcMazcSPhn6
8EudLM3ytGTldlppOWKeD1cMHexeYFNcfvdyVgQ
SmlEZVXN7-FXUqZmK-zjEmgU7oEHWDQ
```

```
{
  "sub": "40f873a9-aca8-41bf-bad5-98d51e765241",
  "iss": "https://iris-iam.stfc.ac.uk",
  "name": "Test User",
  "groups": [
    "iCSC/testing",
    "iCSC",
    "ral-tier1",
    "localAccounts"
  ],
  "preferred_username": "TestUser",
  "organisation_name": "IRIS IAM",
  "exp": 1678095590,
  "iat": 1678091990,
  "jti": "ca7485f6-2a09-4565-a8c8-8a05fb749aac",
  "client_id": "21a6c264-f4ef-4ed1-8d00-1748c8a43642",
  "email": 🙂
}
```

OIDC: Putting this together…

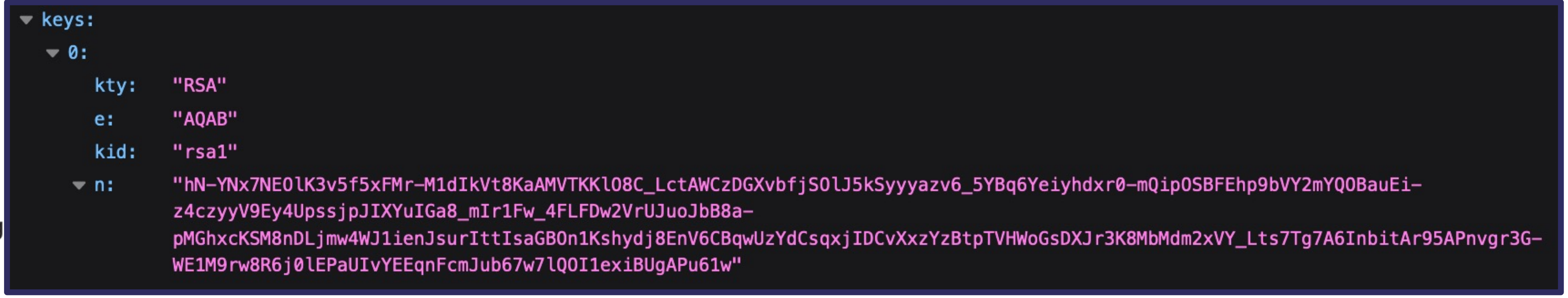


```
.E6e5yuF9x7BTxM9bbGGx-
jwk4VFy8wnkKesWMDWk1QjwI-
oLHYrfCWfoZdJ7BU2ox3MxCMByPBZa-
vwpl4bD8vTYp9mz15z263JyuLLDBhFVdxVNrlsi
ZxQaQDcEPmnxzwy7O3qDzmdFLAG9bZ79BknlMWi
GdZ3erzSacEdEPBSEvSw6Al8RqRksAfK8DV7_6d
9kNcrrl9NSn0lNzX1cv6yYtxwli79iQuFv63UvR
tJ_4F9-27jvIW_dwNgdw7ivhqEmThcMazcSPhn6
8EudLM3ytGTldlppOWKeD1cMHexeYFNcfvdyVgQ
SmlEZVXN7-FXUqZmK-zjEmgU7oEHWDQ
```



```
uY28udWsifQ.E6e5yuF9x7BTxM9bbGGx-
```

JWK signing key published at a URL at the OP, such as:
https://wlcg.cloud.cnaf.infn.it/jwk
The specific URL is detailed in the OP metadata, found at:
`.well-known/openid-configuration`
For Example:
https://wlcg.cloud.cnaf.infn.it/.well-known/openid-configuration



```
▾ keys:
  ▾ 0:
      kty:   "RSA"
      e:     "AQAB"
      kid:   "rsa1"
    ▾ n:     "hN-YNx7NEOlK3v5f5xFMr-M1dIkVt8KaAMVTKKlO8C_LctAWCzDGXvbfjSOlJ5kSyyyazv6_5YBq6Yeiyhdxr0-mQipOSBFEhp9bVY2mYQOBauEi-
             z4czyyV9Ey4UpssjpJIXYuIGa8_mIr1Fw_4FLFDw2VrUJuoJbB8a-
             pMGhxcKSM8nDLjmw4WJ1ienJsurIttIsaGBOn1Kshydj8EnV6CBqwUzYdCsqxjIDCvXxzYzBtpTVHWoGsDXJr3K8MbMdm2xVY_Lts7Tg7A6InbitAr95APnvgr3G-
             WE1M9rw8R6j0lEPaUIvYEEqnFcmJub67w7lQOI1exiBUgAPu61w"
```

CERN School of Computing

OIDC: UserInfo Endpoint

- The ID token is not the only way for a RP to get information about a user – it can also make a request to the Ops `UserInfo` endpoint
- A UserInfo Request is made by the client using either a HTTP `GET` or HTTP `POST`. The Access Token previously obtained from an OIDC Authentication request MUST be included as a Bearer Token
- The OIDC Core recommendation is that the request uses the HTTP `GET` method, with the Access Token sent using the `Authorization` header field

A non-normative example of a UserInfo Request:

```
GET /userinfo HTTP/1.1
Host: server.example.com
Authorization: Bearer SlAV32hkKG
```

NOT EXAMINABLE

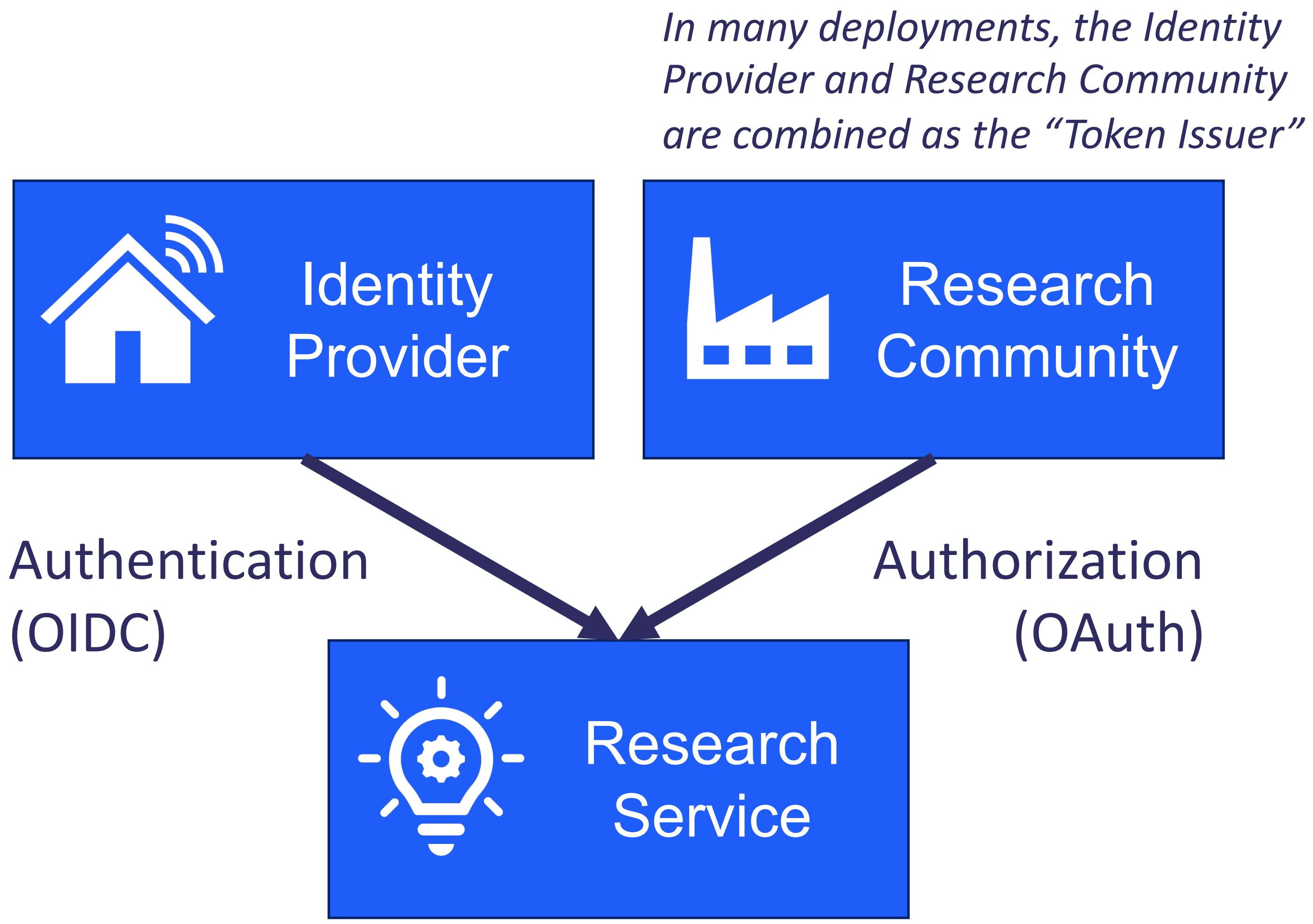
Putting the pieces together… OAuth & OIDC

Good Bits

- Tokens widely accepted
- "Easy" to implement
- Works for non-web

Bad Bits

- Current identity federation status immature


In many deployments, the Identity Provider and Research Community are combined as the "Token Issuer"
Identity Provider
Research Community
Authentication (OIDC)
Authorization (OAuth)
Research Service

CERN School of Computing

… but which is “best”?

thematic CSC
CERN
School of Computing

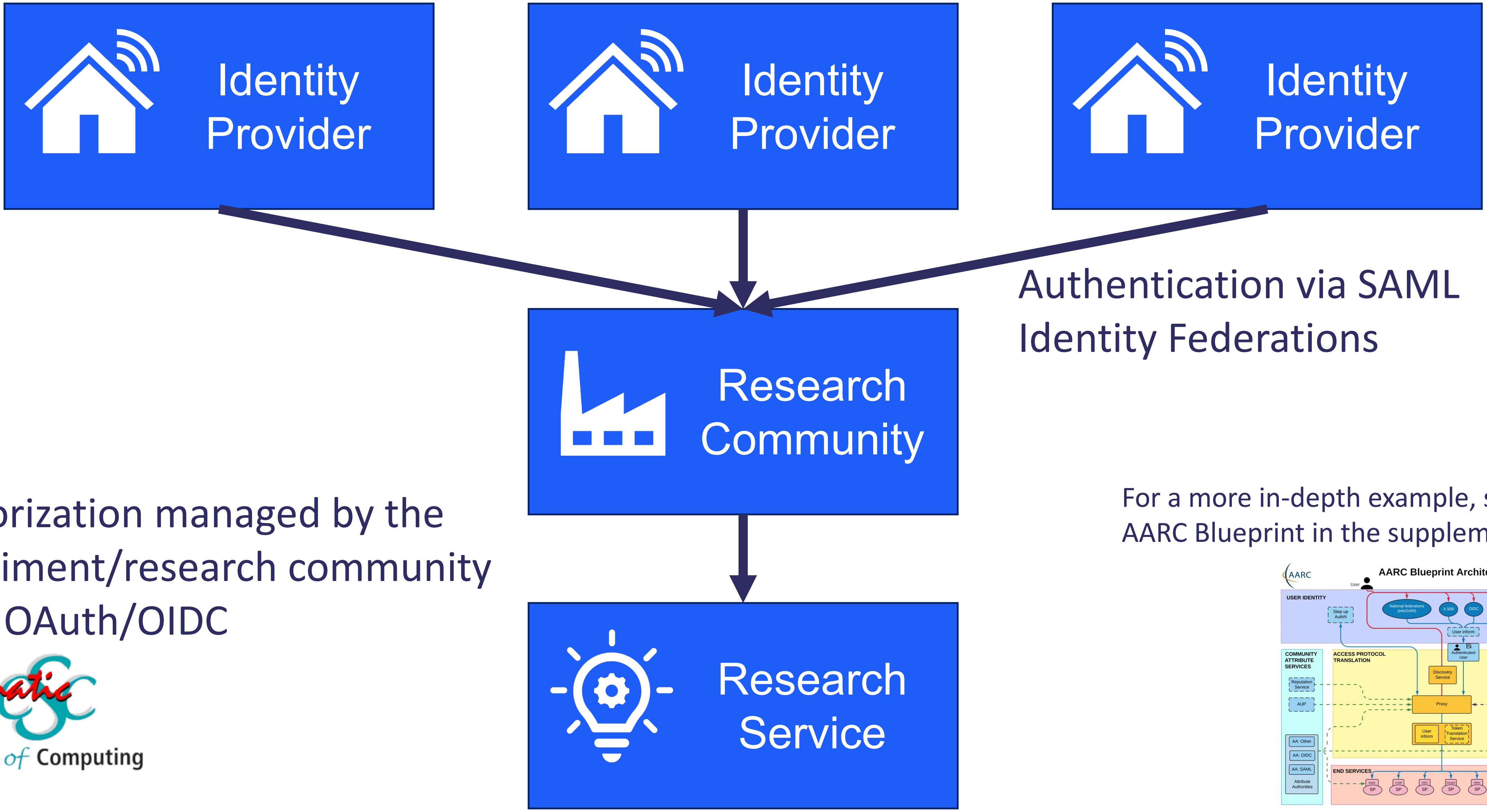
A recap

We have looked at 3 different technologies for distributed AAI

	Certificates	SAML	OAuth2
Web?	Yes	Yes	Yes
Command Line?	Yes	No	Yes
Advantage?	Established	Scalable	Widely accepted
Disadvantage?	Security & Usability	Usability, Non-web	Scalability
Example	Grid Certificates	Your Home Organisation	ORCID, Github

CERN School of Computing

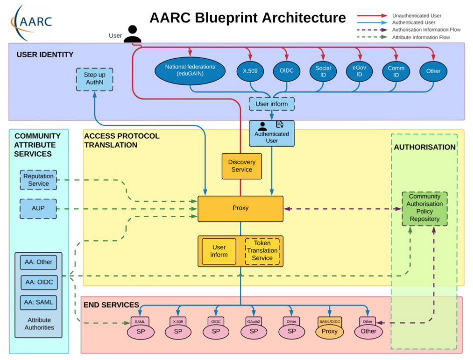
Perhaps a combination of multiple…



Authentication via SAML Identity Federations

Authorization managed by the experiment/research community using OAuth/OIDC

For a more in-depth example, see the AARC Blueprint in the supplement slides


AARC
AARC Blueprint Architecture
USER IDENTITY
COMMUNITY ATTRIBUTE SERVICES
ACCESS PROTOCOL TRANSLATION
AUTHORISATION
END SERVICES
Proxy

thematic CSC
CERN School of Computing

A note on incident response…

Identity management, authentication, and authorization all form key tools within Incident Response – see Barbara's lecture for more!
Taking some examples on how different implementations interact…

WLCG Certificate Federation

- Common security policies
- Central suspension mechanism (Argus)
- Infrastructure CSIRT (Computer Security Incident Response Team)

Very mature setup with international participation in trust initiatives (IGTF)

SAML Federations

- Established Security Framework, Sirtfi
- No central suspension mechanism
- No central operational security or incident response capability

Still a long way to go before Research Communities trust them to the same extent

WLCG OAuth2 Token Issuers

- Suspension possible experiment wide

Procedures a work in progress as the infrastructure develops – see the supplement slides for more info!

NOT EXAMINABLE

Take Aways

- What is the difference between Authentication and Authorization
- How the global research community is connected through shared use of digital identities and understanding of why this is important for research
- An understanding of some of the different tools and methods involved

Science and Technology Facilities Council

Questions?

Science and Technology Facilities Council

Thank you

Science and Technology Facilities Council @STFC_matters Science and Technology Facilities Council

Supplementary Slides
All Non-Examinable Content

1 OAuth Authorization Grant Types

2 OIDC Scope Examples

3 Towards Tokens for the WLCG

thematic CERN School of Computing

OAuth Authorization Grant Types

NOT EXAMINABLE

thematic CSC
CERN School of Computing

OAuth 2.0 – Authorization Grants

- The OAuth 2.0 framework specifies several different methods through which a **Client** can verify itself to the **Authorization Server** - these are known as Authorization Grants
  - Authorization Code
  - PKCE
  - Client Credentials
  - Device Code
  - Refresh Token

Key Authorization Grant Concepts

- Redirect URL – a URL at the client which the AuthZ server will deliver an issued token to
- ClientID – the "username" of the client
- ClientSecret – the "password" of the client

CERN School of Computing

OAuth 2.0 – Authorization Grants

- The OAuth 2.0 framework specifies several different methods through which a **Client** can verify itself to the **Authorization Server** - these are known as Authorization Grants
  - Authorization Code
  - PKCE
  - Client Credentials
  - Device Code
  - Refresh Token

Authorization Code

- Used by both confidential and public clients
- An authorization code is exchanged for an access token
- When the user returns to the client via the Redirect URL, the Authorization Code is extracted from the URL and used to obtain the Access Token

OAuth 2.0 – Authorization Grants

- The OAuth 2.0 framework specifies several different methods through which a **Client** can verify itself to the **Authorization Server** - these are known as Au
  - Authoriza
  - PKCE
  - Client Cr
  - Device C
  - Refresh Token

```
https://example.com/oauth/auth?
    response_type=code
    &scope=EXAMPLE_REQUESTED_SCOPES
    &client_id=EXAMPLE_CLIENTID
    &redirect_uri=EXAMPLE_REDIRECT_URI
```

blic clients

- An authorization code is exchanged for an access token
- When the user returns to the client via the Redirect URL, the Authorization Code is extracted from the URL and used to obtain the Access Token

OAuth 2.0 – Authorization Grants

- The OAuth 2.0 framework specifies several different methods through which a **Client** can verify itself to the **Authorization Server** - these are known as Authorization Grants
  - Device Code
  - Refresh Token

```
curl --header "Authorization: Basic EXAMPLE_SECRET"
--data "grant_type=authorization_code&code=EXAMPLE_CODE"
--request POST https://example.com/oauth/token
```

- Used by both confidential and public clients
- An Authorization code is exchanged for an access token
- When the user returns to the client via the Redirect URL, the Authorization Code is extracted from the URL and used to obtain the Access Token

OAuth 2.0 – Authorization Grants

- The OAuth 2.0 framework specifies several different methods through which a **Client** can verify itself to the **Authorization Server** - these are known as Authorization Grants
  - Authorization Code
  - PKCE
  - Client Credentials
  - Device Code
  - Refresh Token

PKCE: ***Proof Key for Code Exchange***

- An extension to Authorization Code, which aims to prevent Cross Site Request Forgery (CSRF) and Authorization code injection attacks
- Not a replacement for a ClientSecret, and therefore does not enable treating a Public Client as Confidential

thematic CSC
CERN
School of Computing

OAuth 2.0 – Authorization Grants

- The OAuth 2.0 framework specifies several different methods through which a **Client** can verify itself to the **Authorization Server** - these are known as Authorization Grants
  - Authorization Code
  - PKCE
  - Client Credentials
  - Device Code
  - Refresh Token

Client Credentials

- Must ONLY be used by Confidential Clients
- The Client authenticates itself directly with the Authorization Server, for example using it's ClientID and ClientSecret pair or with an public/private key pair
- As Client Authentication is used as the Authorization grant, no further AuthZ is required
- Often used for a client to obtain information about itself

thematic CSC
CERN
School of Computing

OAuth 2.0 – Authorization Grants

- The OAuth 2.0 framework specifies several different methods through

```
curl --request POST \
 --url 'https://EXAMPLE/oauth/token’ \
 --header 'content-type: application/x-www-form-urlencoded’ \
 --data grant_type=client_credentials \
 --data client_id=EXAMPLE_CLIENT_ID \
 --data client_secret=EXAMPLE_CLIENT_SECRET \
 --data audience=EXAMPLE_AUDIENCE
```

ClientSecret pair or with an public/private key pair
- As Client Authentication is used as the Authorization grant, no further AuthZ is required
- Often used for a client to obtain information about itself

CERN School of Computing

OAuth 2.0 – Authorization Grants

- The OAuth 2.0 framework specifies several different methods through which a **Client** can verify itself to the **Authorization Server** - these are known as Authorization Grants
  - Authorization Code
  - PKCE
  - Client Credentials
  - Device Code
  - Refresh Token

Device Code

- This flow is intended for use by browserless or input constrained clients – such as command line
- The user is directed to visit a URL at the AuthZ Server in a separate browser, along with a user code to identify the device
- The original device continuously polls the AuthZ server with the generated device code until the user completes the interaction, the code expires, or another error occurs

CERN School of Computing

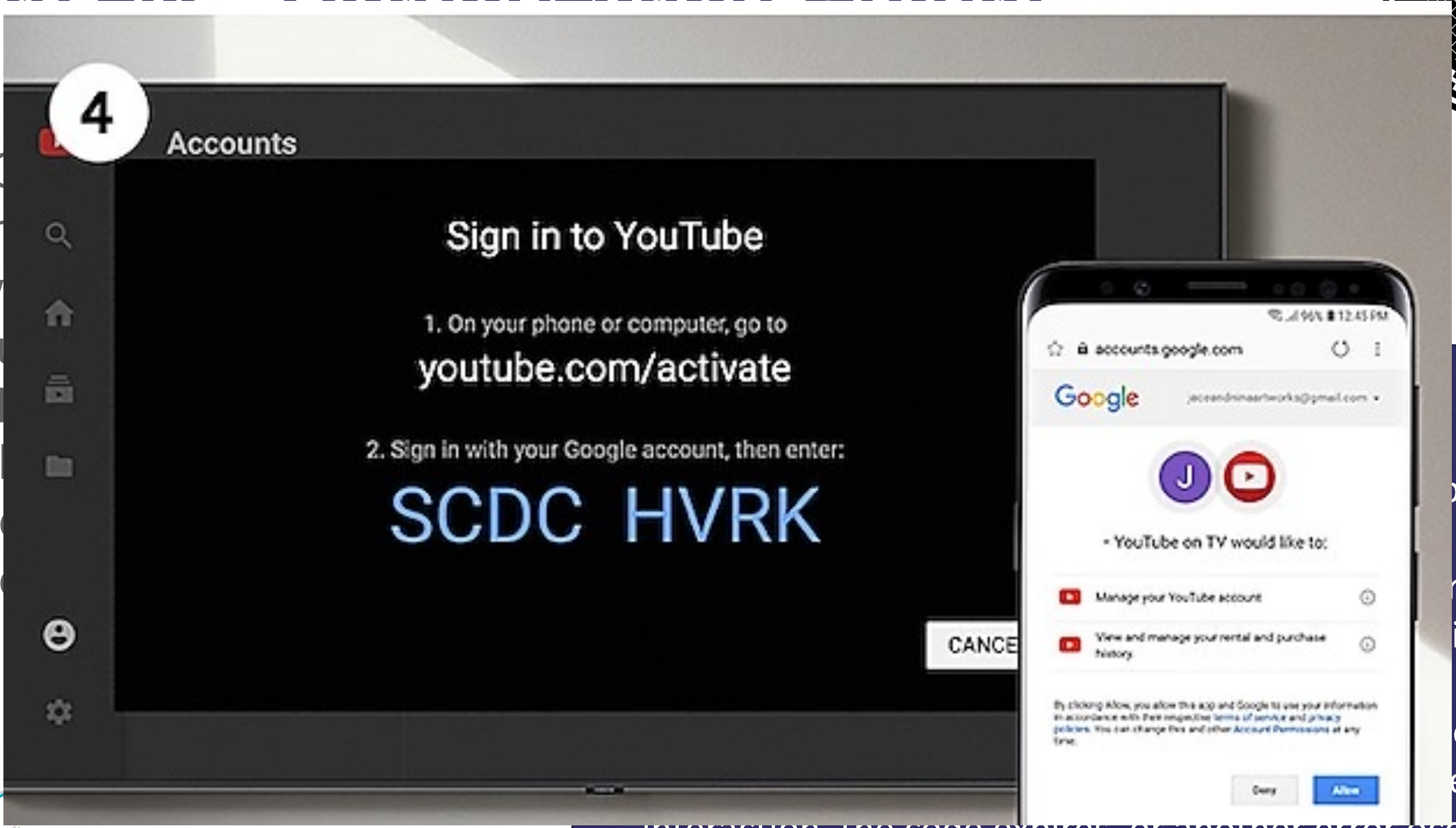
OAuth 2.0 – Authorization Grants

- The C
  which
  know
  - Au
  - Pl
  - Cl
  - De
  - Re

out

rver in a
ify the

erver with
es the
interaction, the code expires, or another error occurs



Sign in to YouTube

1. On your phone or computer, go to
youtube.com/activate

2. Sign in with your Google account, then enter:
SCDC HVRK

accounts.google.com

Google

YouTube on TV would like to:

Manage your YouTube account

View and manage your rental and purchase history

By clicking Allow, you allow this app and Google to use your information in accordance with their respective terms of service and privacy policies. You can change this and other Account Permissions at any time.

Deny Allow

OAuth 2.0 – Authorization Grants

- The OAuth 2.0 framework specifies several different methods through which a **Client** can verify itself to the **Authorization Server** - these are known as Authorization Grants
  - Authorization Code
  - PKCE
  - Client Credentials
  - Device Code
  - Refresh Token

Refresh Token

- This is a mechanism for allowing a client to get a new Access Token after an initial one has expired, without further user interaction
- Not a replacement for a ClientSecret, and therefore does not enable treating a Public Client as Confidential

OAuth 2.0 – Authorization Grants

```
To get a Refresh token, the client needs to request the scope:
     scope=offline_access&

curl --request POST \
 --url 'https://{yourDomain}/oauth/token’ \
 --header 'authorization: Basic {yourApplicationCredentials}’ \
 --header 'content-type: application/x-www-form-urlencoded’ \
 --data grant_type=refresh_token \
 --data 'client_id={yourClientId}’ \
 --data 'refresh_token={yourRefreshToken}
```

therefore does not enable treating a Public Client as Confidential

thematic CERN School of Computing

OIDC Scope Examples

NOT EXAMINABLE

OIDC: Default Scopes

as defined by OIDC Core

Scope	Summary
openid	REQUIRED. Informs the Authorization Server that the Client is making an OpenID Connect request. If the openid scope value is not present, the behaviour is entirely unspecified.
profile	OPTIONAL. This scope value requests access to the End-User's default profile Claims, which are: name, family_name, given_name, middle_name, nickname, preferred_username, profile, picture, website, gender, birthdate, zoneinfo, locale, and updated_at.
email	OPTIONAL. This scope value requests access to the email and email_verified Claims.
address	OPTIONAL. This scope value requests access to the address Claim
phone	OPTIONAL. This scope value requests access to the phone_number and phone_number_verified Claims.
offline_access	OPTIONAL. This scope value requests that an OAuth 2.0 Refresh Token be issued that can be used to obtain an Access Token that grants access to the End-User's UserInfo Endpoint even when the End-User is not present (not logged in).

Towards Tokens For the WLCG

NOT EXAMINABLE

thematic CSC
CERN
School of Computing

Transition to Tokens - Motivations

- OAuth and OIDC protocols have been adopted by a wide range of software and systems, thanks to their prevalence in industry – in particular within the social identity space
  - This prevalence enables developers to utilise developed libraries, and facilitates integration and interoperability

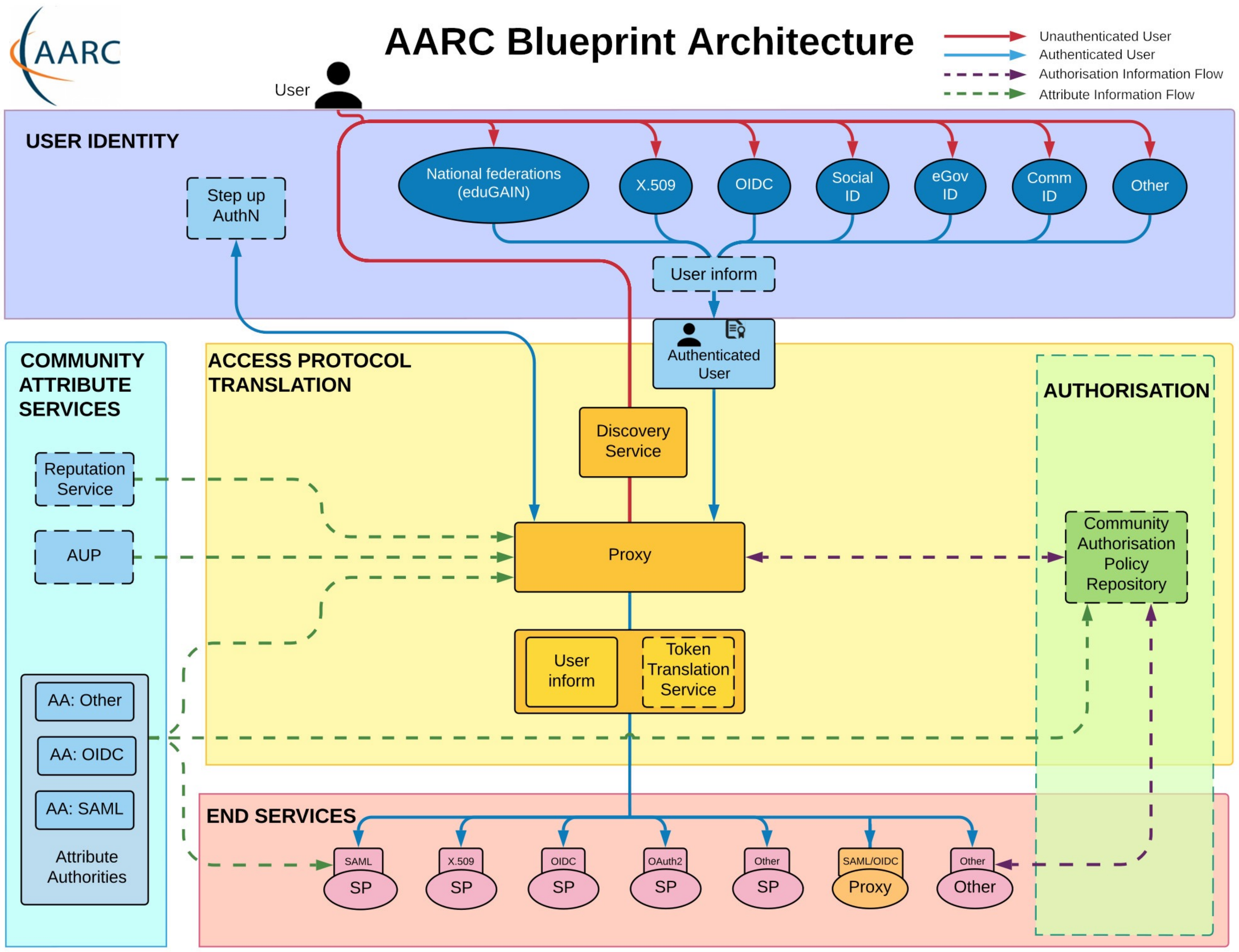
- Certificates, whilst established within WLCG, are not familiar outside this context - and are often viewed as unintuitive. Conversely, Token-based flows are becoming increasingly commonplace, and can lead to a better user experience as a result

thematic CERN School of Computing

Token Authentication and Authorization Infrastructure

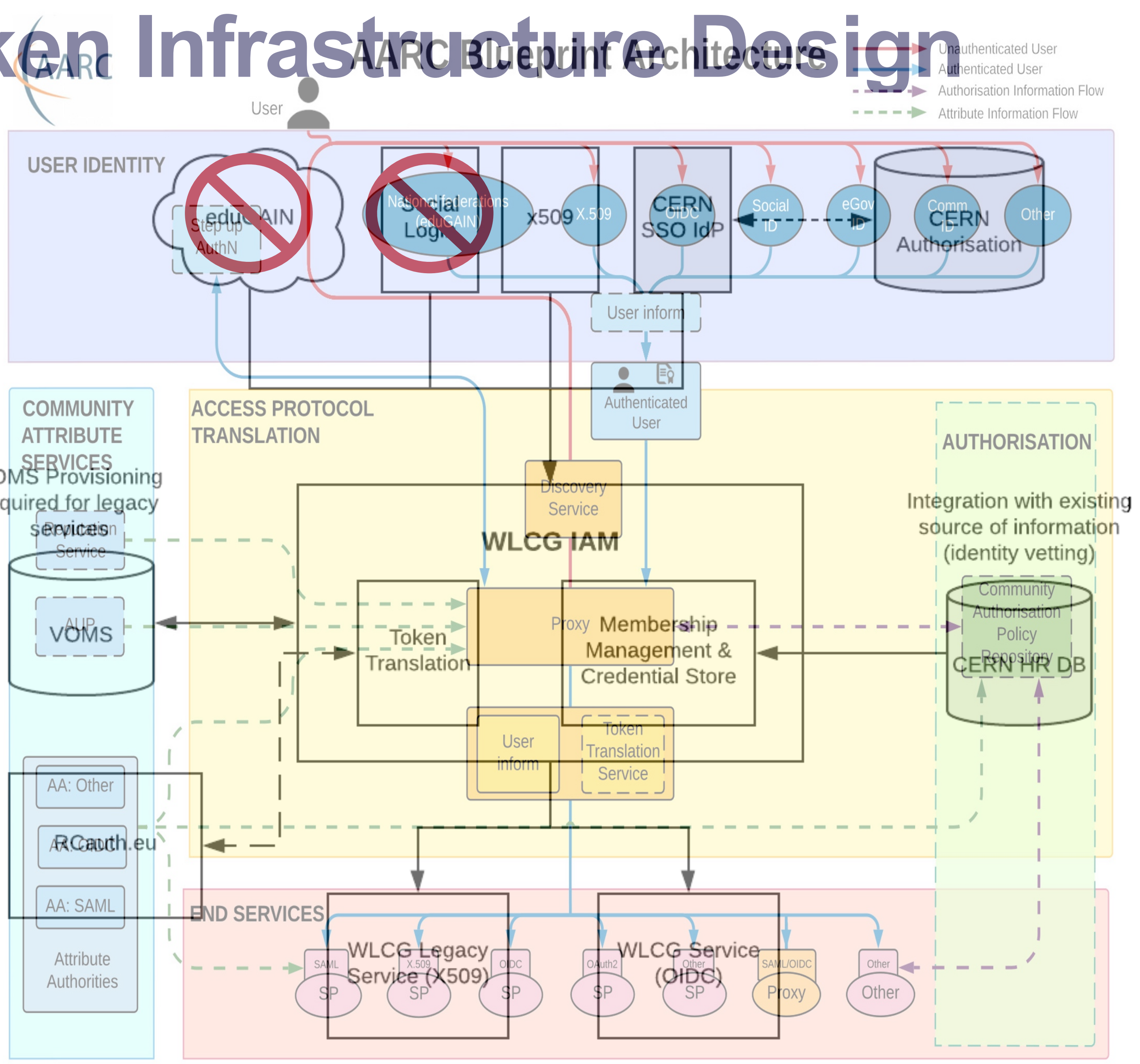
- In the planned WLCG infrastructure, there will be an OP per VO, which users may access using their CERN Account
- This model unifies the IdP and Research community, with both being represented by the Token Issuer
- The Infrastructure design has been informed by the **AARC Blueprint architecture** - a set of software building blocks that can be used to implement federated access management solutions for international research collaborations

The AARC Blueprint



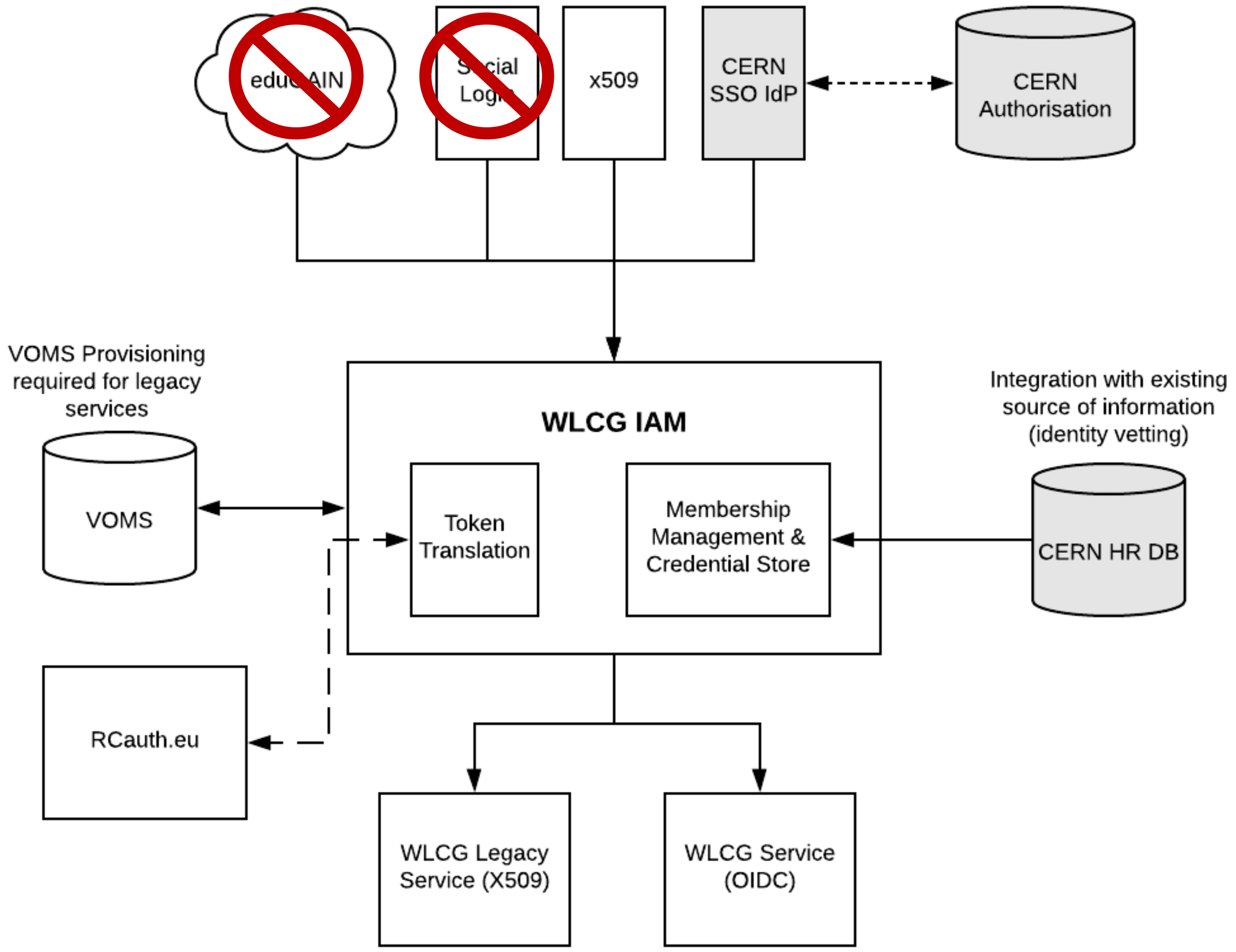
thematic CSC
CERN School of Computing

WLCG Token Infrastructure Design

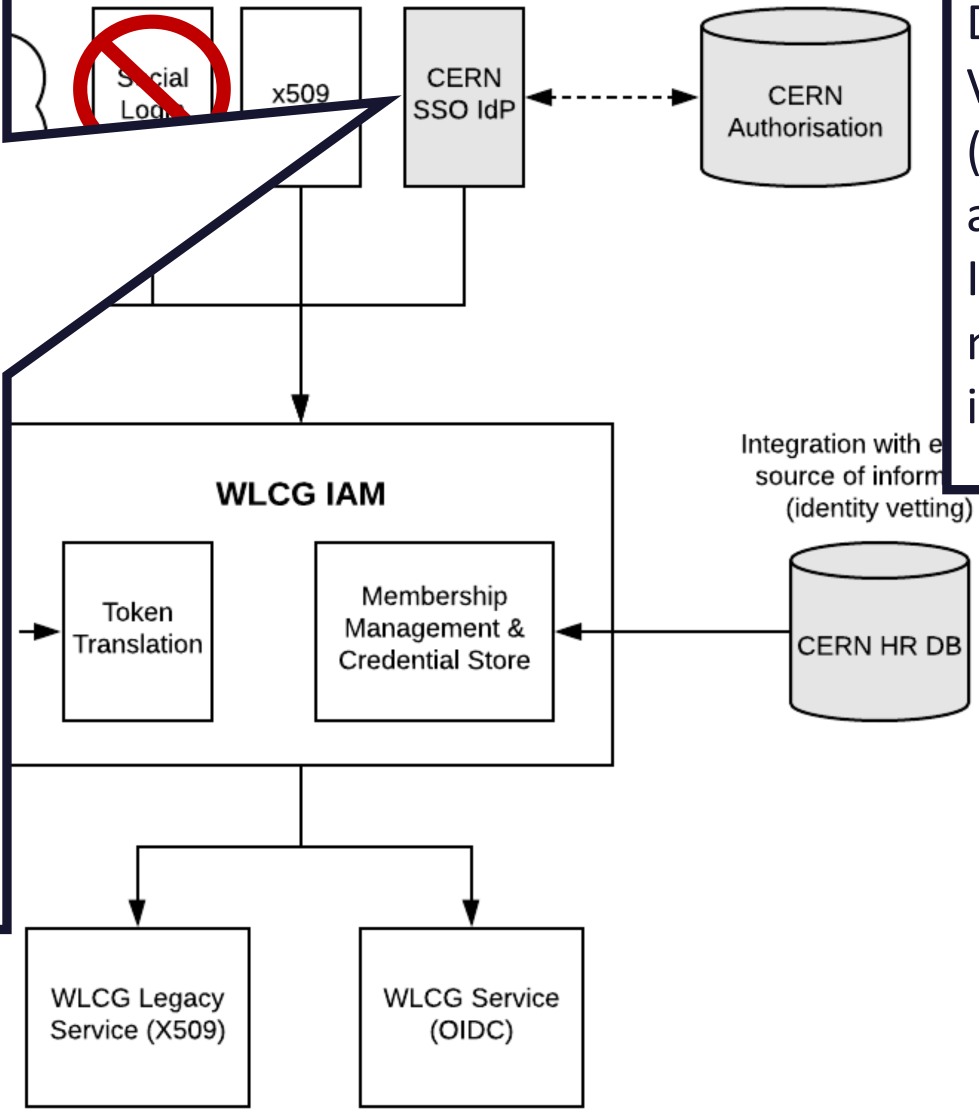


CERN School of Computing

WLCG Token Infrastructure Design


eduGAIN
Social Login
x509
CERN SSO IdP
CERN Authorisation
VOMS Provisioning required for legacy services
VOMS
WLCG IAM
Token Translation
Membership Management & Credential Store
Integration with existing source of information (identity vetting)
CERN HR DB
RCauth.eu
WLCG Legacy Service (X509)
WLCG Service (OIDC)

WLCG Token Infrastructure Design



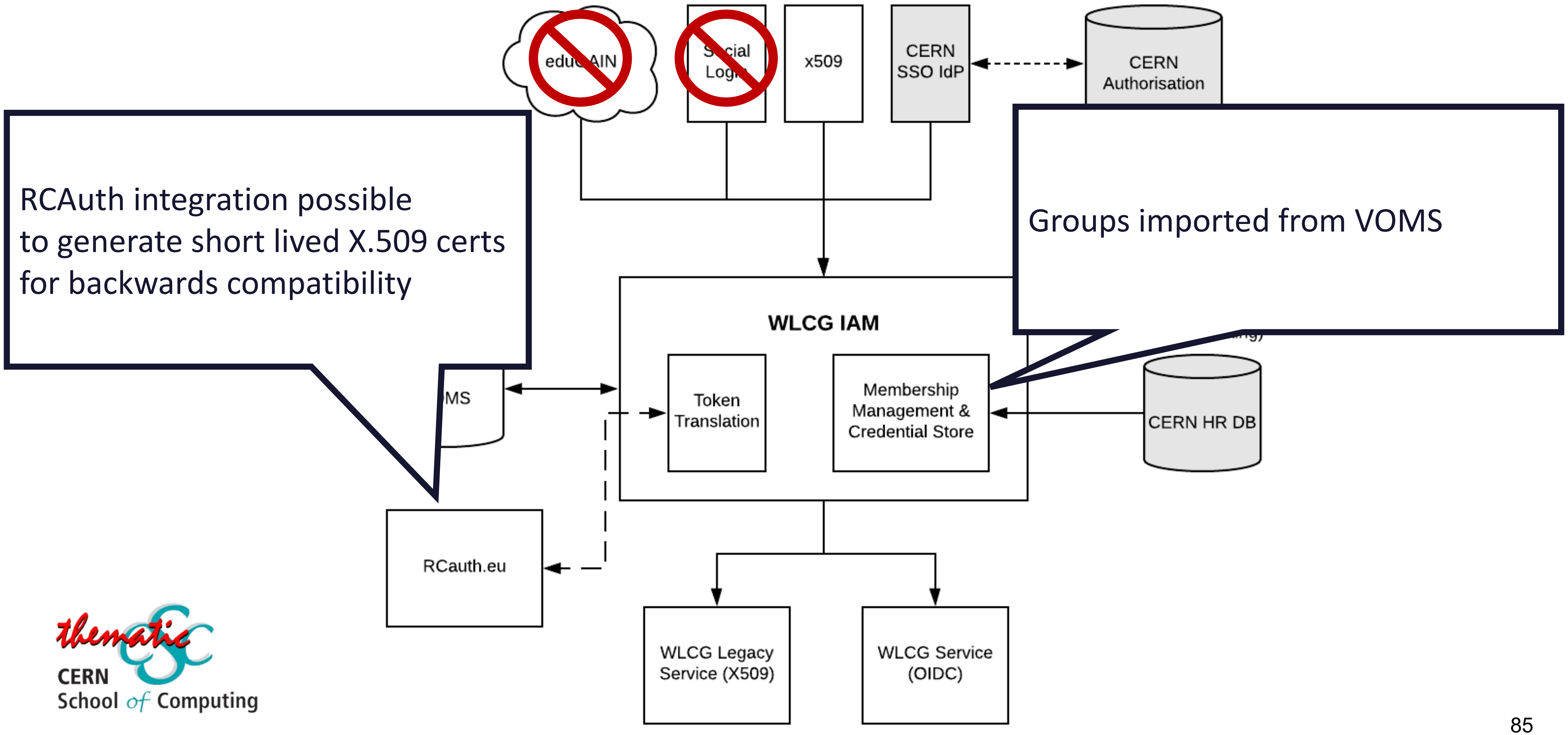
CERN SSO releases:

- Name,
- Email,
- CERN Person ID (indicates HR has performed ID check),
- CERN Kerberos Principal
- ...

Currently all researchers have CERN accounts but aim is to work towards removing this need in future

CERN Person ID is checked against CERN HR DB. Affiliation with Virtual Organisation (experiment) is verified, as well as end dates.
If the check is OK, the membership
is approved.

WLCG Token Infrastructure Design



RCAuth integration possible to generate short lived X.509 certs for backwards compatibility

Groups imported from VOMS

WLCG Token Schema

- In order to serve authorization information a VO, the WLCG token schema definies extra claims – `wlcg.groups` for Group-based authorization and `scopes` for Capability-based authorization
- `wlcg.groups` semantics are equivalent to existing VOMS groups, and will be initially imported directly from VOMS
  - Eg: /atlas/production
- `scopes` is used to provide capability to a specific token, rather than permanent authorization to a user
  - Format `$AUTHZ:$PATH` where `$PATH` is mandatory (may be '/' for *)
  - Eg: storage.read:/atlas
- For more details, you can see the published schema: https://zenodo.org/record/3460258#.Y-YqUxPMLVs

thematiC CERN School of Computing