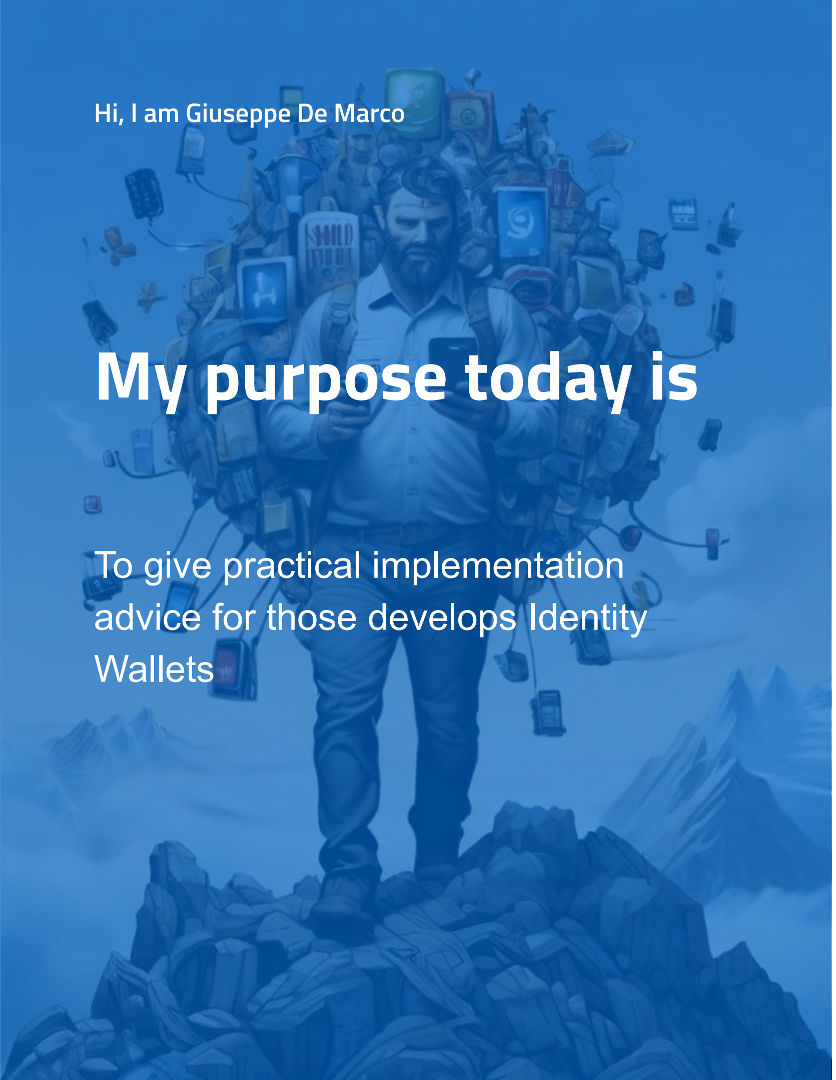
# Identity Wallet for Implementers With Deadlines

General software architecture and survival tips for forward-thinking developers

**TIIME – Trust and Internet Identity Meeting Europe**
Day 4, 1 February 2024 - Copenhagen

**Giuseppe De Marco**
Open Source Project Leader, Digital Identities



Hi, I am Giuseppe De Marco

# My purpose today is

To give practical implementation advice for those develops Identity Wallets

## TODAY WE TALK ABOUT

- → **Terms,** let's get aligned!
- → Components and roles of the **ecosystem**
- → **Trust Model** and Infrastructure of Trust
- → **Credential Data model and format**
- → Credential **Issuance**
- → Credential **Presentation**
- → Credential **Revocation**
- → **Open points and risks**

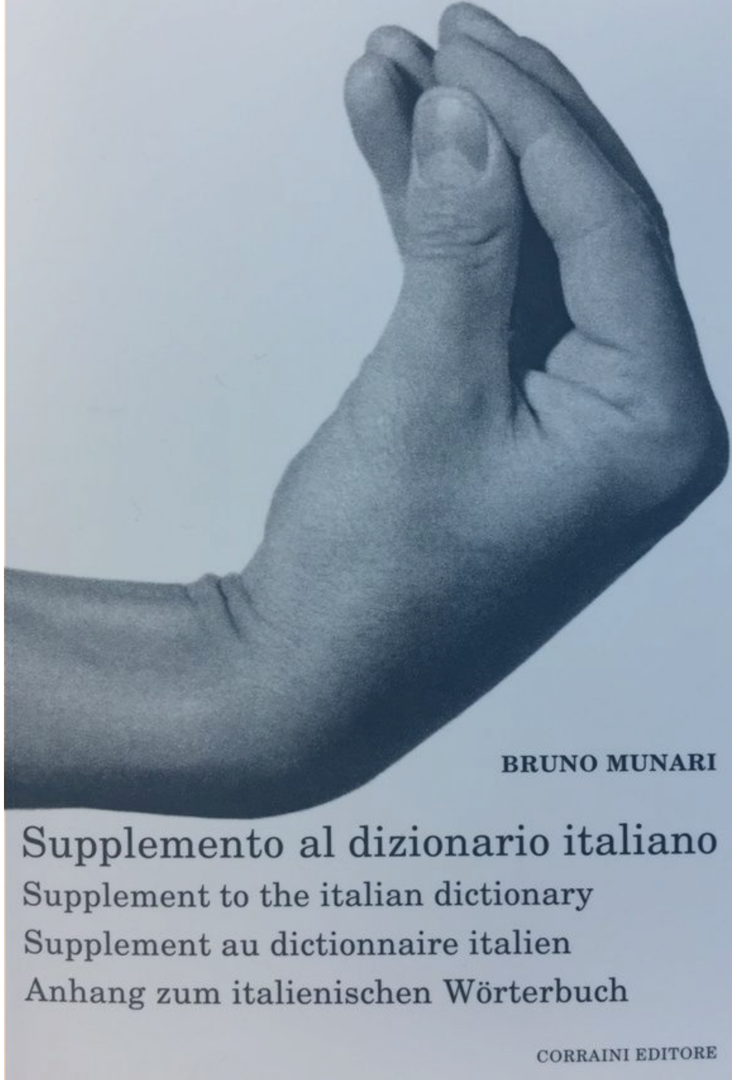
LET’S TRY TO GET ALIGNED

# Facing a Terminological Babylon

World Wide Web Consortium (**W3C**)
Internet Engineering Task Force (**IETF**)
International Organization for Standardization (**ISO**)
**OpenID** Foundation
Decentralized Identity Foundation (**DIF**)
eIDAS 2.0 (European Commission)

have produced overlapping technical specifications.

Different specifications define similar concepts and terminologies, but often use different names for similar meaning.

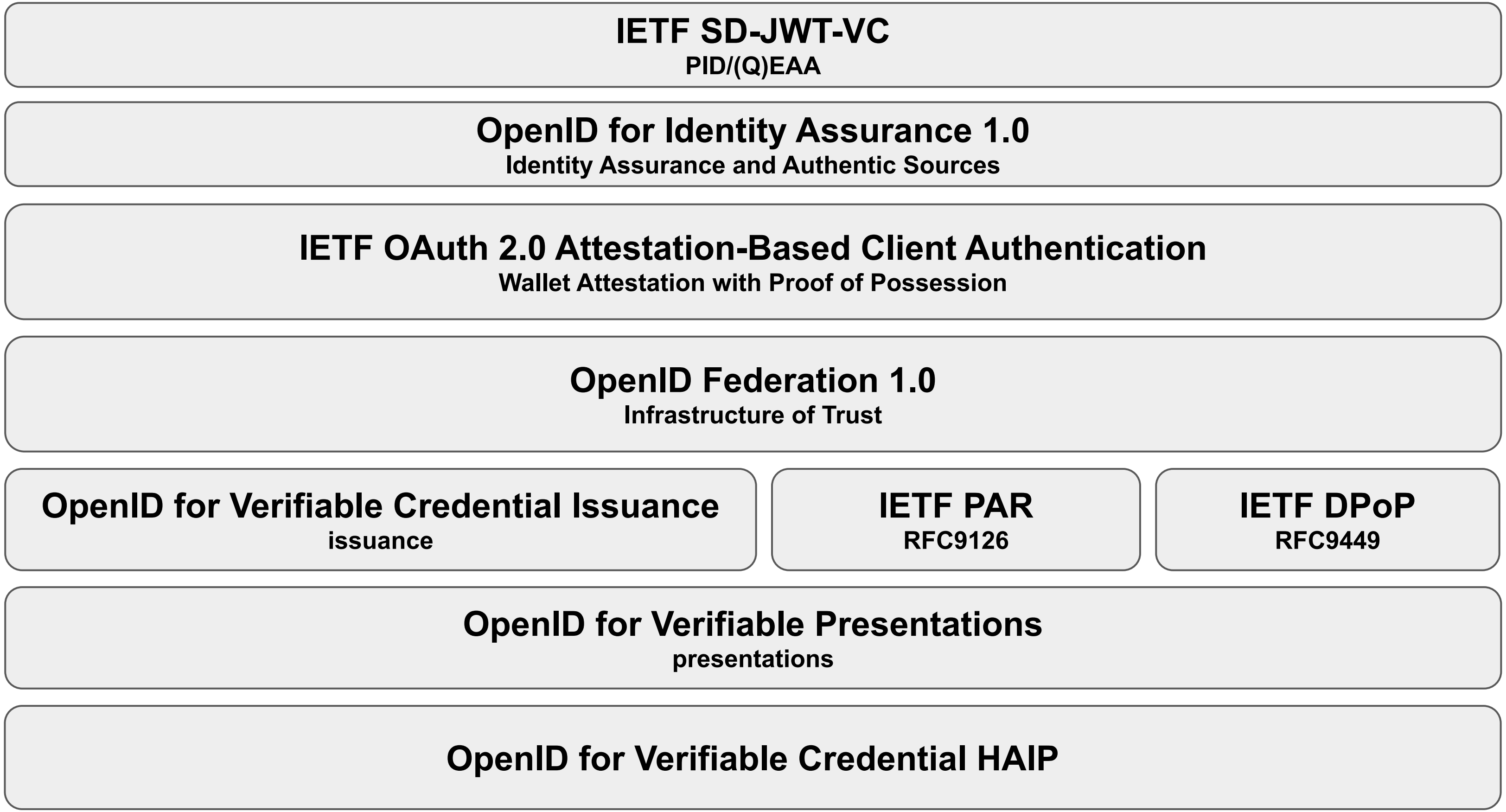


BRUNO MUNARI

Supplemento al dizionario italiano
Supplement to the italian dictionary
Supplement au dictionnaire italien
Anhang zum italienischen Wörterbuch

CORRAINI EDITORE

# Using IETF and OpenID



**IETF SD-JWT-VC**
PID/(Q)EAA

**OpenID for Identity Assurance 1.0**
Identity Assurance and Authentic Sources

**IETF OAuth 2.0 Attestation-Based Client Authentication**
Wallet Attestation with Proof of Possession

**OpenID Federation 1.0**
Infrastructure of Trust

**OpenID for Verifiable Credential Issuance**
issuance

**IETF PAR**
RFC9126

**IETF DPoP**
RFC9449

**OpenID for Verifiable Presentations**
presentations

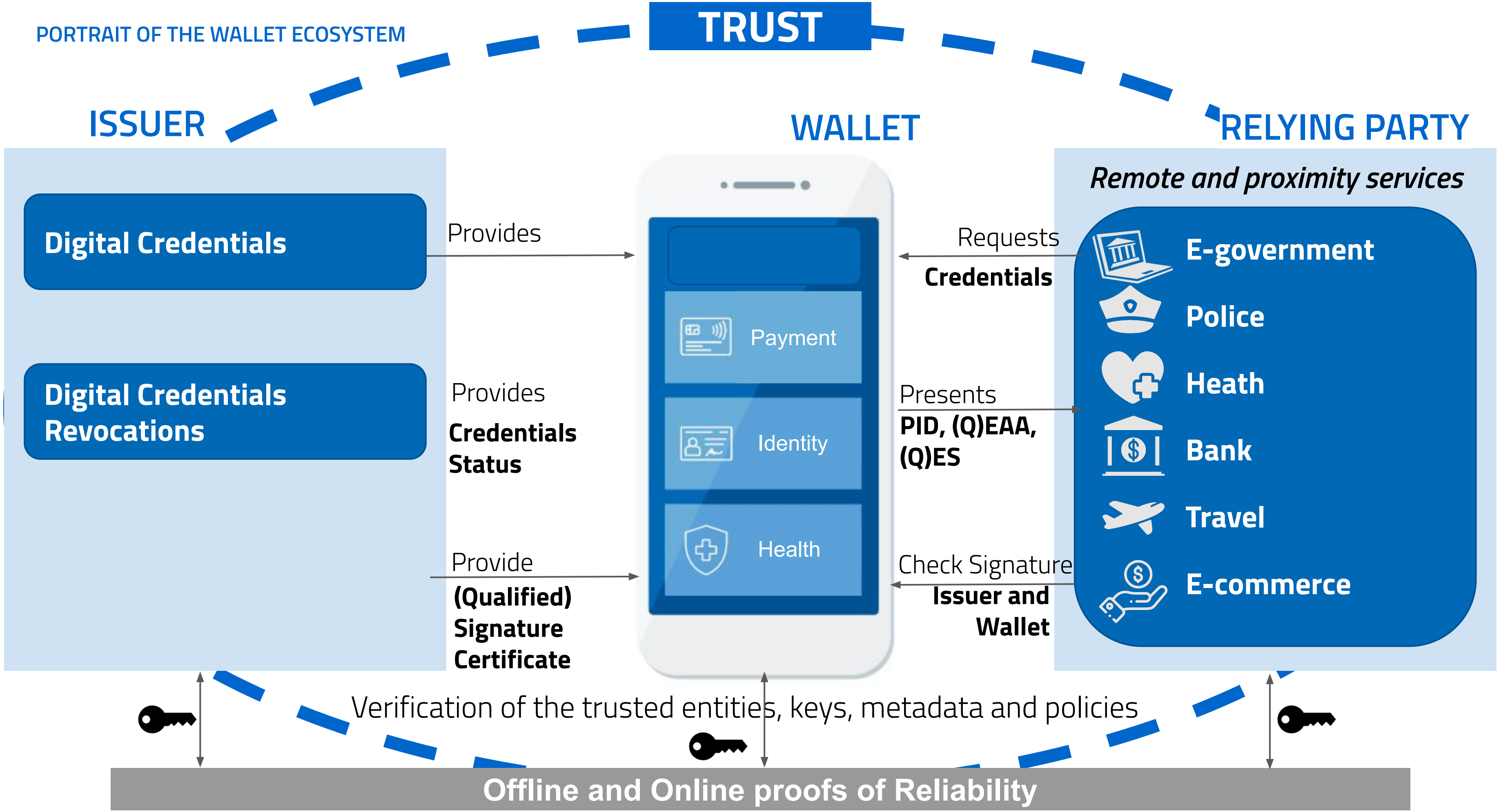
**OpenID for Verifiable Credential HAIP**

SPEAKING WELL MEANS THINKING WELL

# Words are Important

- **Verifiable Credential** *AND/OR* **Digital Credential** AND/OR eIDAS **PID/(Q)EAA**
- Credential Issuer: actually it is an OAuth 2.0 RS.
- **Relying Party** AND/OR **Verifier** (OpenID vs. ISO)
- Trusted Third Party above all (Intermediates included)
- **Wallet Attestation** AND/OR **Wallet Trust Evidence**
- Wallet Solution (Wallet Provider, Wallet Instance, Wallet Secure Cryptographic Device)

PORTRAIT OF THE WALLET ECOSYSTEM



TRUST

ISSUER

WALLET

RELYING PARTY

Digital Credentials

Provides

Digital Credentials Revocations

Provides **Credentials Status**

Provide **(Qualified) Signature Certificate**

Payment

Identity

Health

Requests **Credentials**

Presents **PID, (Q)EAA, (Q)ES**

Check Signature **Issuer and Wallet**

*Remote and proximity services*

E-government

Police

Heath

Bank

Travel

E-commerce

Verification of the trusted entities, keys, metadata and policies

Offline and Online proofs of Reliability

WALLET ECOSYSTEM IS NOT A PIECE OF CAKE, IT IS AN ENTIRE STACKED CAKE!

# How To Approach The … Cake.

- Divide the components by specific contexts, **assign the components to experts in specific domains**.
- eIDAS LoA High is not high enough; the qualified electronic signature kit can be shared among different people, and smartphone hardware is not certifiable. If we want to start immediately, we must **use a Level Of Assurance Substantial** across the entire stack.
- External Hardware Tokens and Smartcards are UX nightmares, remote HSM is something to be explored (also with the proximity/offline flows in mind).

THE PATH TO THE TRUST

# Trust Resolution and Trust Chain building

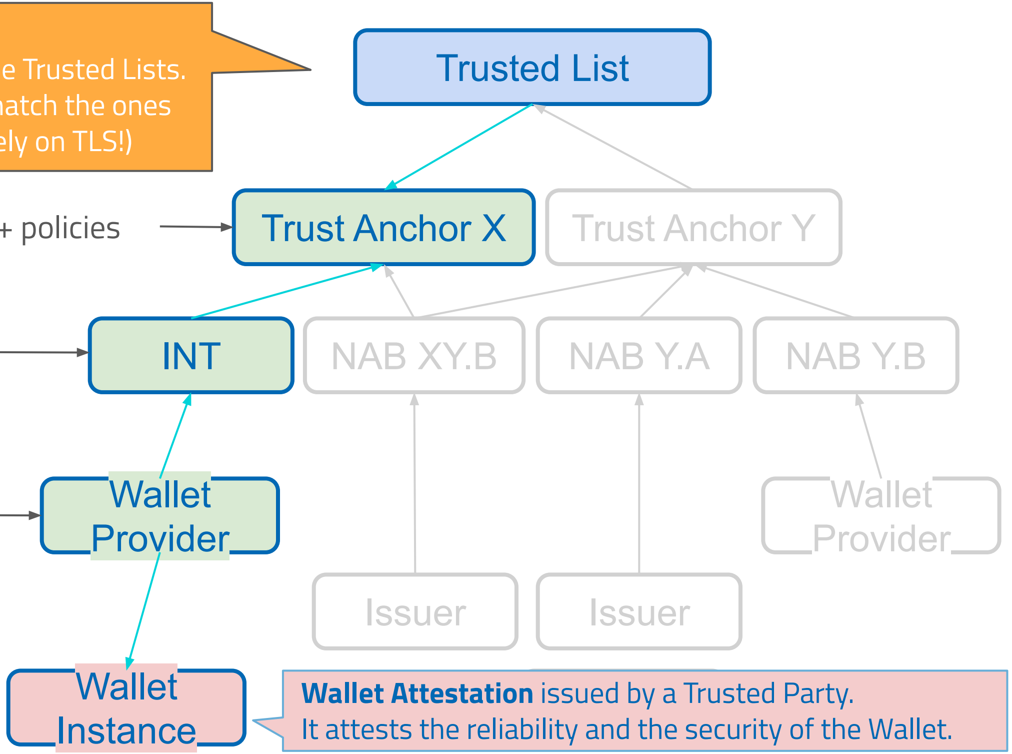
**REQUIREMENTS**
1. All the Trust Anchors URLs/public keys must be  taken from the Trusted Lists.
2. The Trust Anchors public keys published on their own MUST match the ones obtained from the Trusted Lists -> double check (don't rely entirely on TLS!)

3. Get information/keys about the Intermediate from the TA + policies

2. Get information from the Leaf's Superior (Intermediate)
   What the Intermediate says about the Leaf
   The Leaf's key to verify the Leaf's information + policies
   What the Intermediate says about itself.

1. Get information from the Leaf
   What the Leaf says about itself.

**OUTPUT**
1. Trust Chain
2. **Final metadata** according to the processed policies
3. Verified **Trust Marks**



**Wallet Attestation** issued by a Trusted Party.
It attests the reliability and the security of the Wallet.

# Digital Credential Data Model and Format

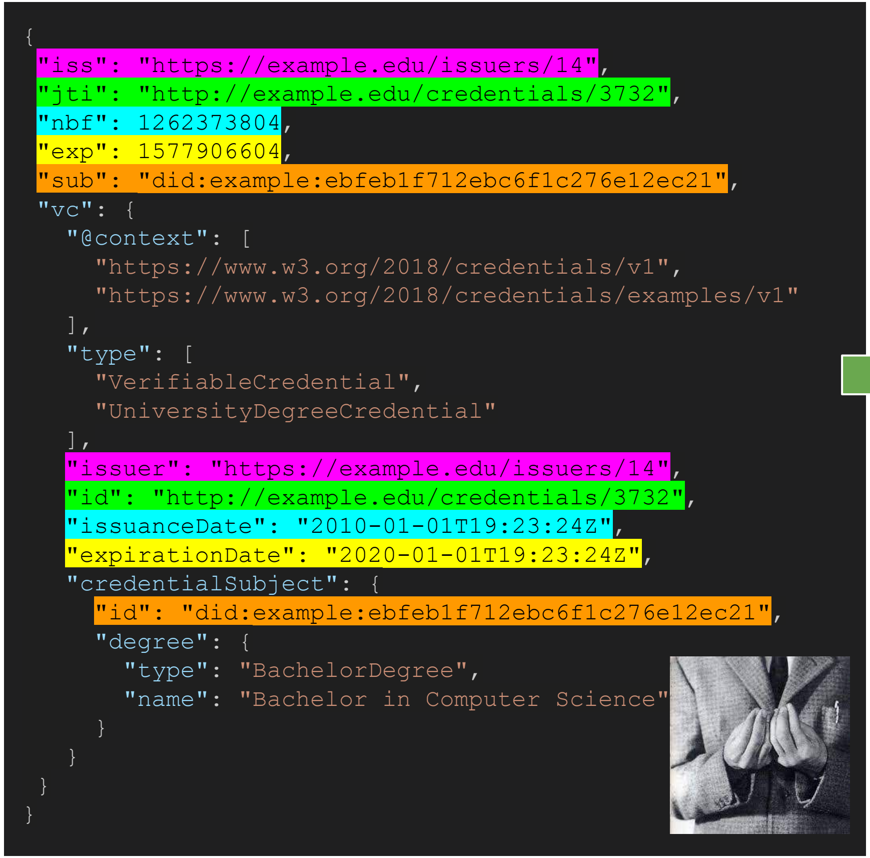
Two simple rules:

1. Start with a JSON including all the valuable R&S attributes, Using well-established user claims in OpenID, eduPerson and SHAC schemas.
2. Use SD-JWT to make them selectively disclosable and signed within a JWT

For today:

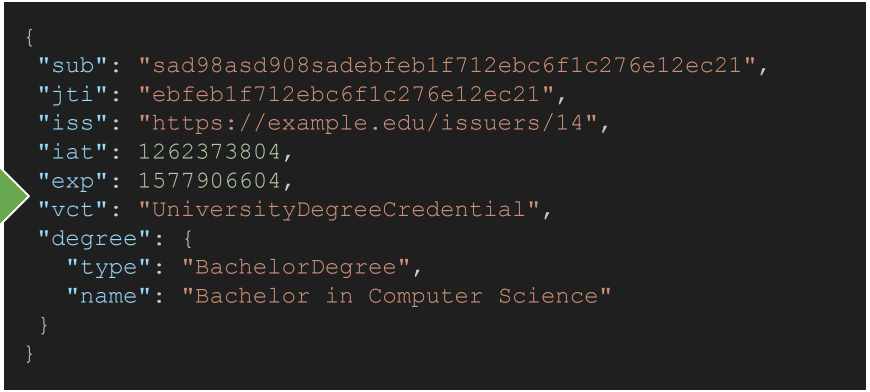
- I don’t use advanced cryptography using AnonCreds and / or BLS Signature, since they are still not standardized
- I have implemented mdoc cbor with Python and published it under Identity Python. SD-JWT and mdoc cbor are equivalent but SD-JWT is simpler. Do we need ISO mdoc cbor for R&S?
- I don’t use W3C VC Data model for the following reasons …

# No duplicate information in SD-JWT VC



```
{
  "iss": "https://example.edu/issuers/14",
  "jti": "http://example.edu/credentials/3732",
  "nbf": 1262373804,
  "exp": 1577906604,
  "sub": "did:example:ebfeb1f712ebc6f1c276e12ec21",
  "vc": {
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
      "https://www.w3.org/2018/credentials/examples/v1"
    ],
    "type": [
      "VerifiableCredential",
      "UniversityDegreeCredential"
    ],
    "issuer": "https://example.edu/issuers/14",
    "id": "http://example.edu/credentials/3732",
    "issuanceDate": "2010-01-01T19:23:24Z",
    "expirationDate": "2020-01-01T19:23:24Z",
    "credentialSubject": {
      "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
      "degree": {
        "type": "BachelorDegree",
        "name": "Bachelor in Computer Science"
      }
    }
  }
}
```

JWT-VC payload with duplicate information highlighted
(VCDM 1.1 but similar in VCDM 2.0)



```
{
  "sub": "sad98asd908sadebfeb1f712ebc6f1c276e12ec21",
  "jti": "ebfeb1f712ebc6f1c276e12ec21",
  "iss": "https://example.edu/issuers/14",
  "iat": 1262373804,
  "exp": 1577906604,
  "vct": "UniversityDegreeCredential",
  "degree": {
    "type": "BachelorDegree",
    "name": "Bachelor in Computer Science"
  }
}
```

SD-JWT VC payload

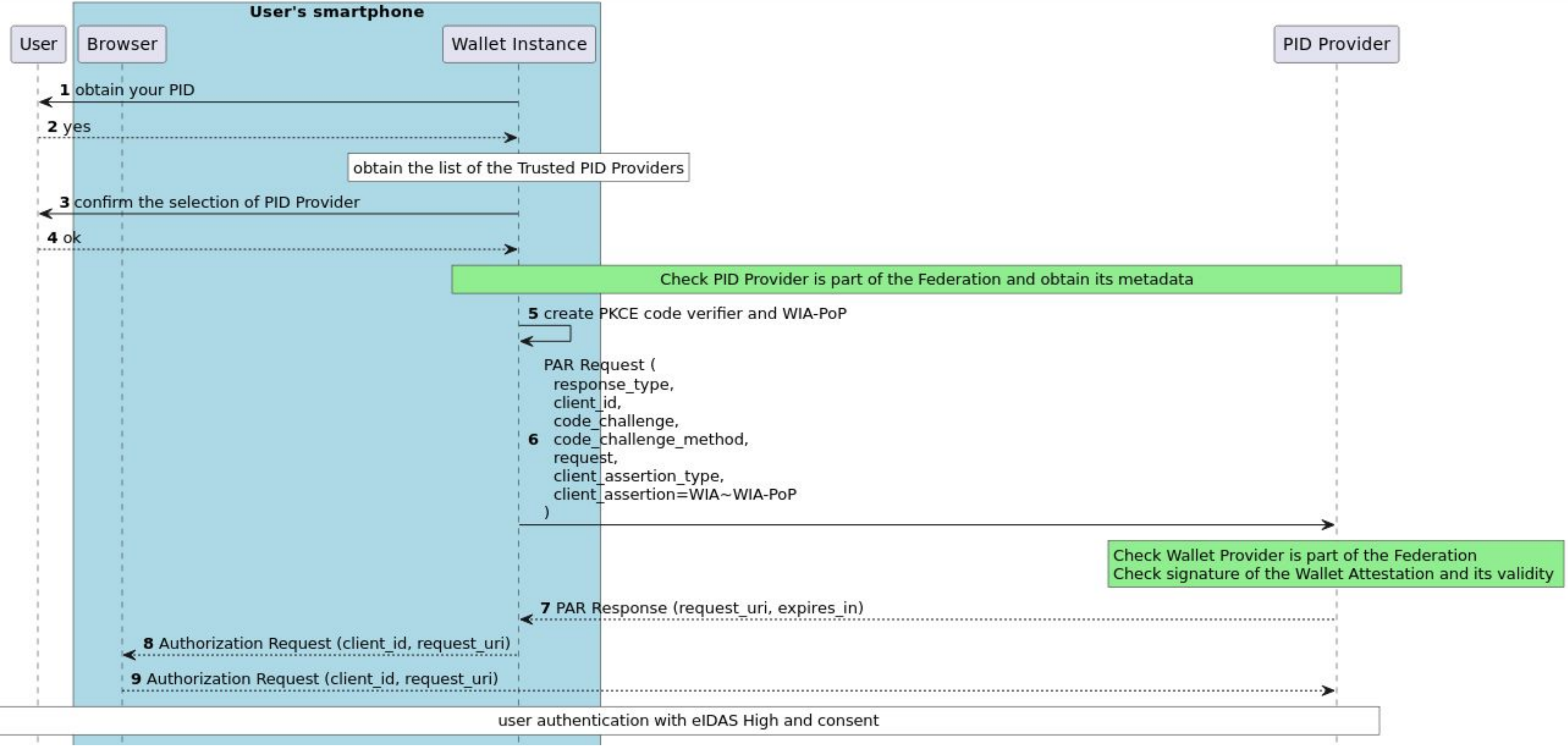
credits to Daniel Fett

## WALLET ATTESTATION

- Issued by the Wallet Provider
- It attests Wallet Security
  - It doesn’t disclose **key_type** and **user_authentication**
  - it doesn’t contain any personal data
  - its subject is the wallet instance NOT the User
  - It uses [draft-oid4vc-haip-sd-jwt-vc] and **[NIST] AAL**
    - AAL means Wallet Authentication Assurance level
    - Definition of the **AAL** levels **in a common platform** is required
      - to not disclose any hardware specific component and user preferences [[HAIP conflicts]]
      - Many hardware features and peculiarities must be grouped within the AAL levels, defined within a mobile security framework, in order to ensure trust levels without disclosing the hardware or the preferences of its user.

```
{
  "alg": "ES256",
  "kid": "5t5YYpBhN-EgIEEI5iUzr6r0MR02LnVQ0OmekmNKcjY",
  "trust_chain": [  "eyJhbGciOiJFUz...6S0A", ... ],
  "typ": "wallet-attestation+jwt",
}
.
{
  "iss": "https://wallet-provider.example.org",
  "sub": "vbeXJksM45xphtANnCiG6mCyuU4jfGNzopGuKvogg9c",
  "iat": 1687281195,
  "exp": 1687288395,
  "aal": "https://wallet-provider.example.org/LoA/substantial",
  "cnf":  {"jwk": { ... }}
}
```

## CREDENTIAL ISSUANCE



User's smartphone

User

Browser

Wallet Instance

PID Provider

**1** obtain your PID

**2** yes

obtain the list of the Trusted PID Providers

**3** confirm the selection of PID Provider

**4** ok

Check PID Provider is part of the Federation and obtain its metadata

**5** create PKCE code verifier and WIA-PoP

**6** PAR Request (
response_type,
client_id,
code_challenge,
code_challenge_method,
request,
client_assertion_type,
client_assertion=WIA~WIA-PoP
)

Check Wallet Provider is part of the Federation
Check signature of the Wallet Attestation and its validity

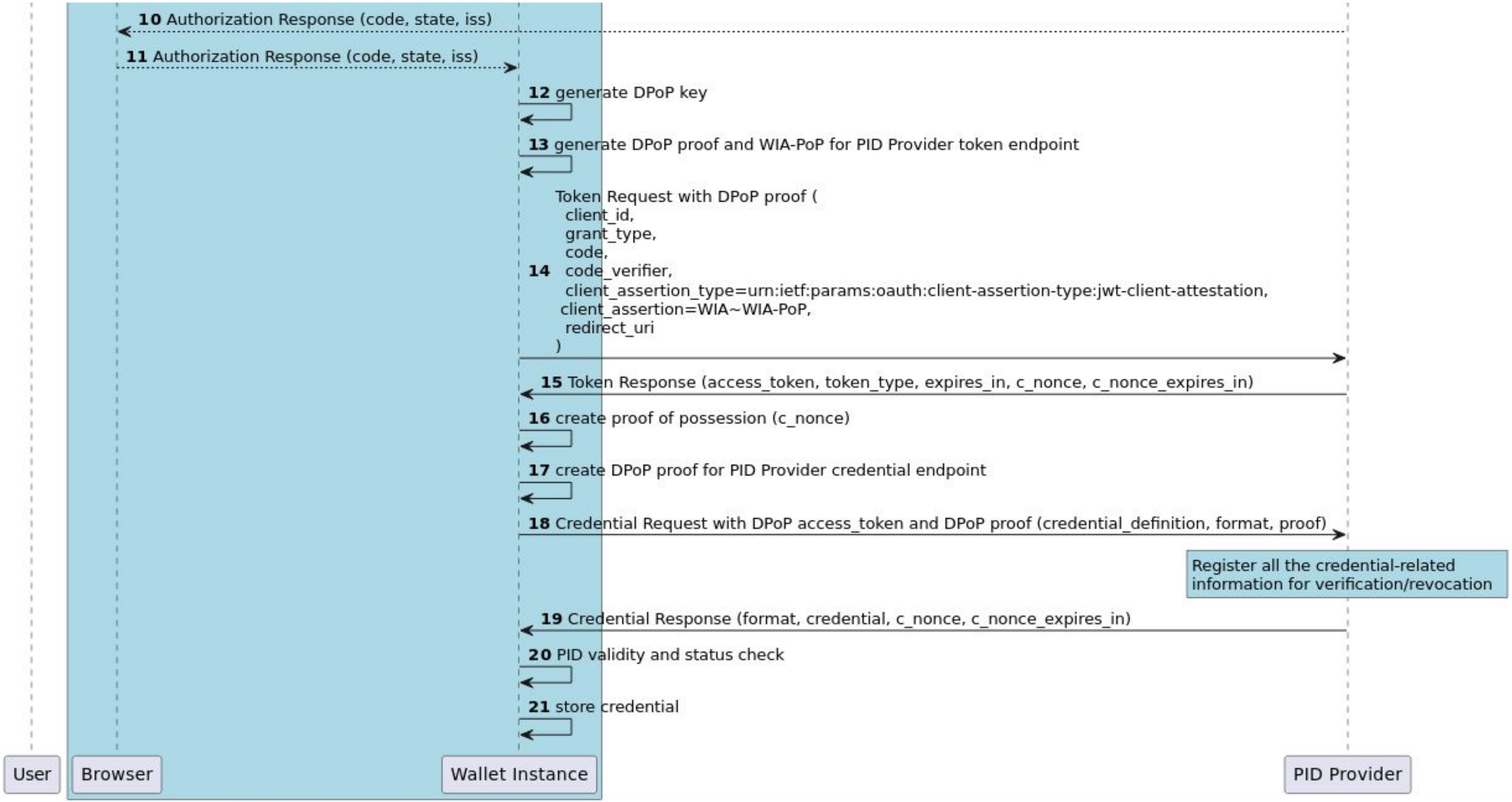
**7** PAR Response (request_uri, expires_in)

**8** Authorization Request (client_id, request_uri)

**9** Authorization Request (client_id, request_uri)

user authentication with eIDAS High and consent

## CREDENTIAL ISSUANCE



**10** Authorization Response (code, state, iss)

**11** Authorization Response (code, state, iss)

**12** generate DPoP key

**13** generate DPoP proof and WIA-PoP for PID Provider token endpoint

**14** Token Request with DPoP proof (
client_id,
grant_type,
code,
code_verifier,
client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-client-attestation,
client_assertion=WIA~WIA-PoP,
redirect_uri
)

**15** Token Response (access_token, token_type, expires_in, c_nonce, c_nonce_expires_in)

**16** create proof of possession (c_nonce)

**17** create DPoP proof for PID Provider credential endpoint

**18** Credential Request with DPoP access_token and DPoP proof (credential_definition, format, proof)

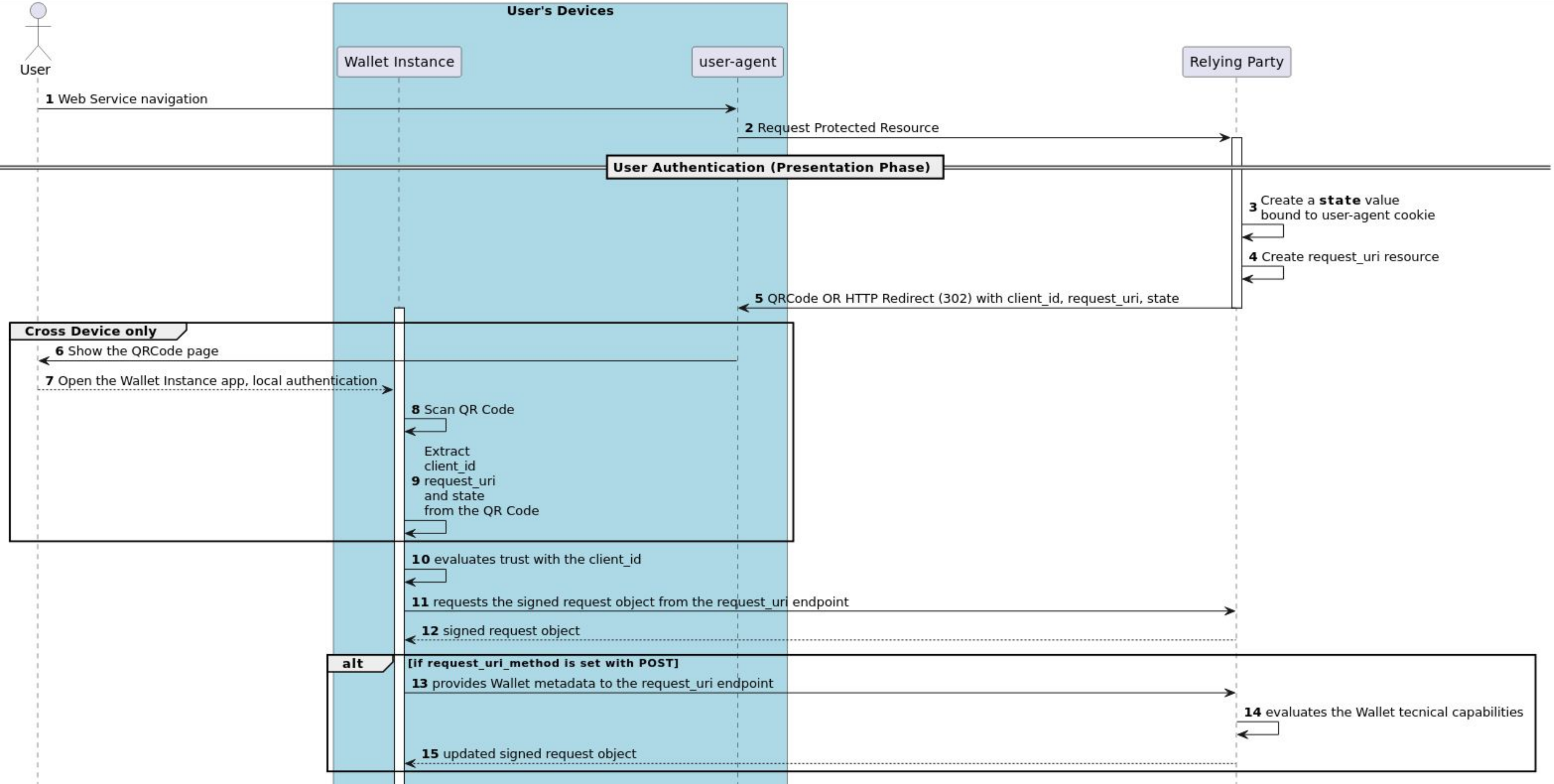
Register all the credential-related information for verification/revocation

**19** Credential Response (format, credential, c_nonce, c_nonce_expires_in)

**20** PID validity and status check

**21** store credential

# PRESENTATION



User's Devices

User

Wallet Instance

user-agent

Relying Party

**1** Web Service navigation

**2** Request Protected Resource

**User Authentication (Presentation Phase)**

**3** Create a **state** value bound to user-agent cookie

**4** Create request_uri resource

**5** QRCode OR HTTP Redirect (302) with client_id, request_uri, state

**Cross Device only**

**6** Show the QRCode page

**7** Open the Wallet Instance app, local authentication

**8** Scan QR Code

**9** Extract client_id request_uri and state from the QR Code

**10** evaluates trust with the client_id

**11** requests the signed request object from the request_uri endpoint

**12** signed request object

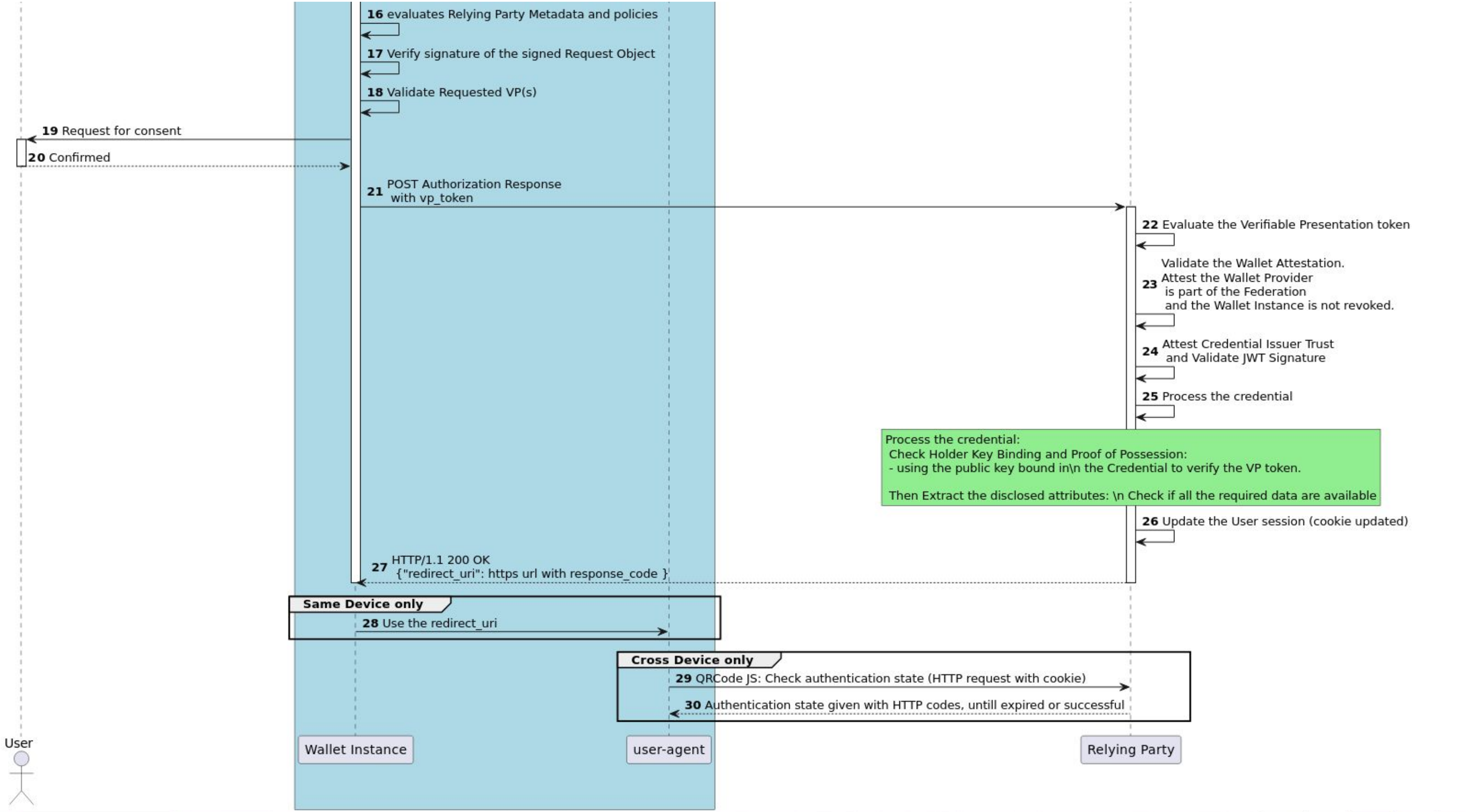
**alt** **[if request_uri_method is set with POST]**

**13** provides Wallet metadata to the request_uri endpoint

**14** evaluates the Wallet tecnical capabilities

**15** updated signed request object

# PRESENTATION



16 evaluates Relying Party Metadata and policies

17 Verify signature of the signed Request Object

18 Validate Requested VP(s)

19 Request for consent

20 Confirmed

21 POST Authorization Response
with vp_token

22 Evaluate the Verifiable Presentation token

23 Validate the Wallet Attestation.
Attest the Wallet Provider
is part of the Federation
and the Wallet Instance is not revoked.

24 Attest Credential Issuer Trust
and Validate JWT Signature

25 Process the credential

Process the credential:
Check Holder Key Binding and Proof of Possession:
- using the public key bound in\n the Credential to verify the VP token.

Then Extract the disclosed attributes: \n Check if all the required data are available

26 Update the User session (cookie updated)

27 HTTP/1.1 200 OK
{"redirect_uri": https url with response_code }

Same Device only

28 Use the redirect_uri

Cross Device only

29 QRCode JS: Check authentication state (HTTP request with cookie)

30 Authentication state given with HTTP codes, untill expired or successful

# REVOCATIONS

OAuth 2.0 Status Lists:
https://www.ietf.org/archive/id/draft-ietf-oauth-status-list-00.html

OAuth 2.0 Status Attestations:
https://datatracker.ietf.org/doc/draft-demarco-status-attestations/

# Thank You For Your Attention!

For further clarifications, ideas, proposals, or discussions, contact me at:

- demarcog83@gmail.com

If you have the desire and aptitude, contribute to the developments on the Wallet Interoperability Framework, a project born from R&S people for R&S people:

- **https://github.com/WalletInteroperabilityLab/eudi-wif/**