How to Build Interoperable Decentralized Identity Systems with OpenID for Verifiable Credentials

Kristina Yasuda, Microsoft

Dr. Torsten Lodderstedt, yes

Kristina Yasuda

Identity Standards Architect

Microsoft

Dr. Torsten Lodderstedt

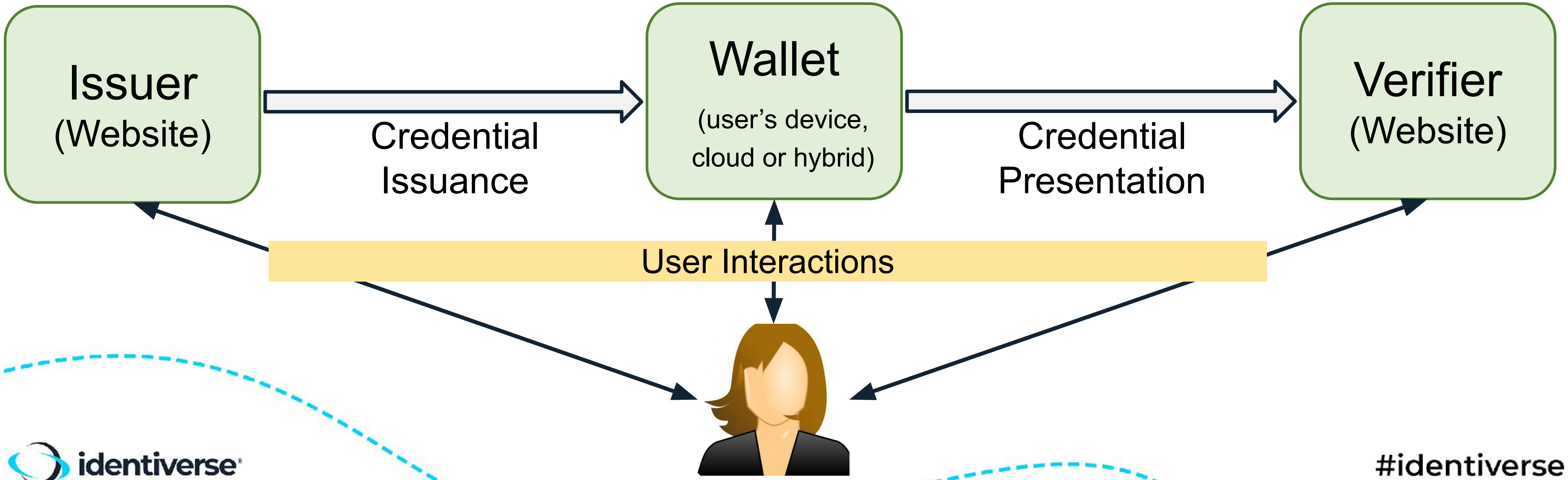
Managing Director

yes IDP GmbH

What is Decentralized Identity?

- The User presenting the Identity data directly to the Verifier from the Wallet
  - <> In the federated model where Identity data is sent directly from the IdP to the Verifier

- Usually expressed with the flow below:

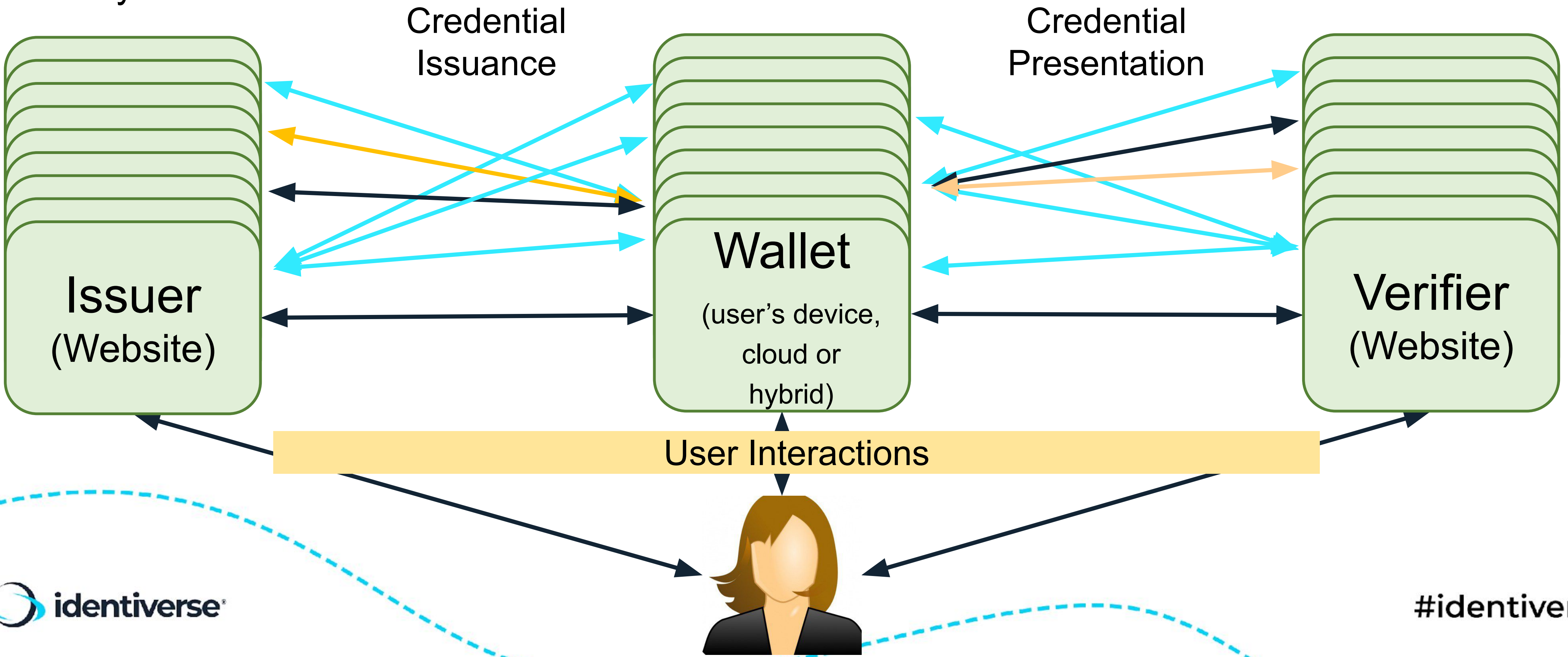


Verifiable Credentials: Benefits

- End-Users gain more privacy, and portability over their identity information.
- Cheaper, faster, and more secure identity verification, when transforming physical credentials into digital ones.
- Universal approach to handle identification, authentication, and authorization in digital and physical space.

Why Protocol Layer Interoperability is Crucial.

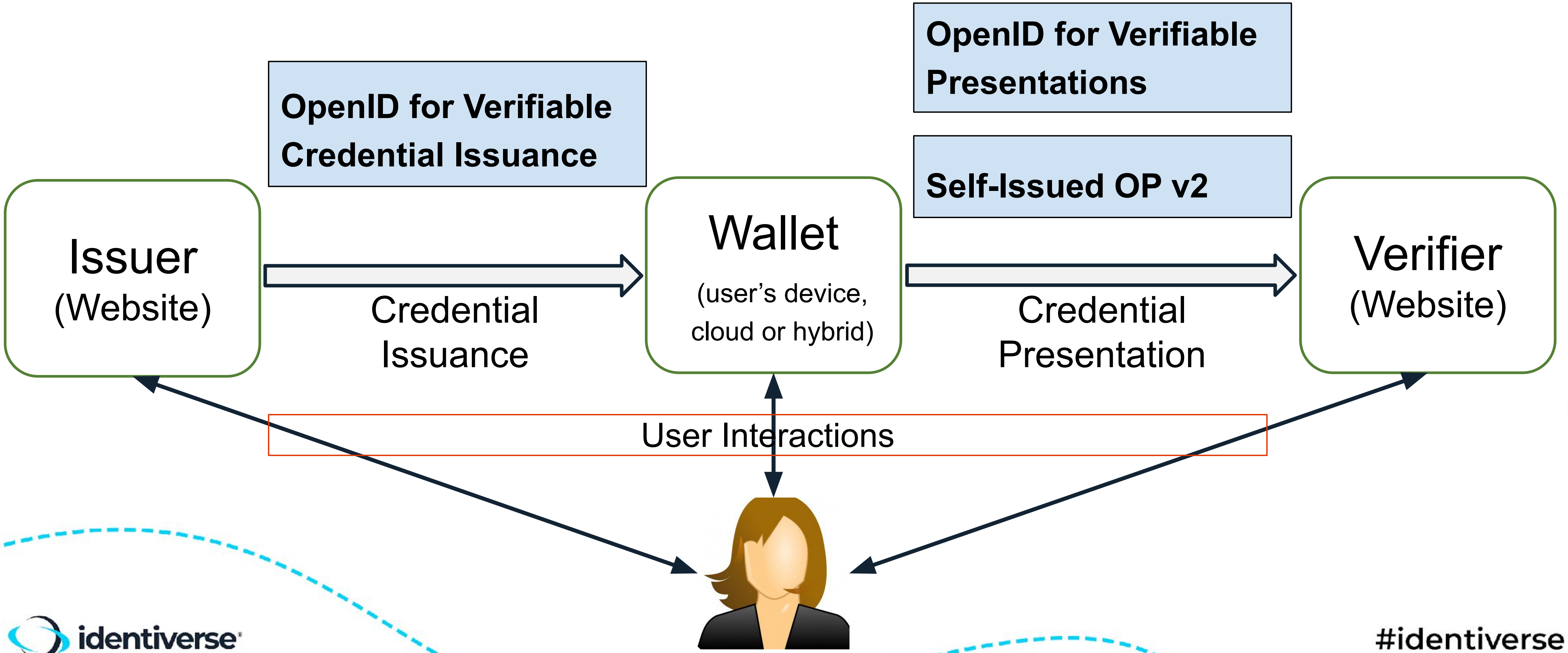
One entity needs to talk to the large the number of entities, to increase the value of “Decentralized Identity”.



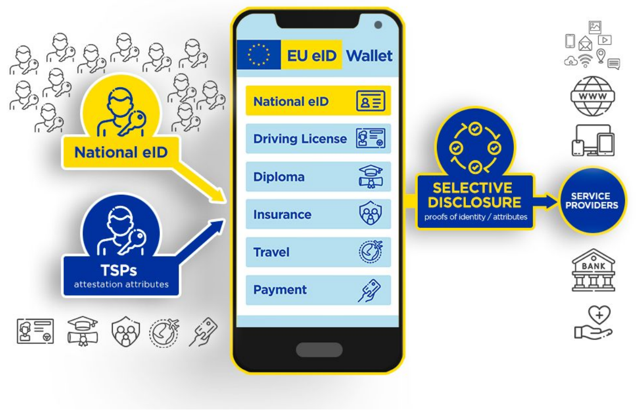
Problems we identified & how we solved them

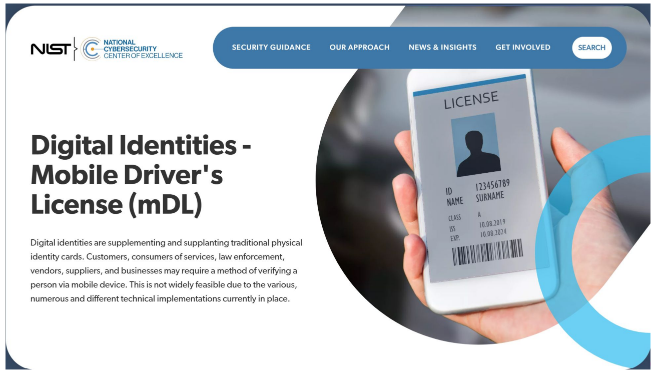
Problem		Solution
A lot of entirely new Protocols. (Hard to get security right, steep learning curve)	⇒	Building upon currently widely used protocols: OAuth 2.0 and OpenID Connect. (Secure, already understood)
No clear winner among Credential Formats	⇒	Designing a Credential Format agnostic protocol
Reluctance to use only DIDs. No clear winner among DID methods	⇒	Designing a protocol agnostic to the Key Resolution mechanism. (No need to use DIDs)
Participating entities cannot typically establish trust upfront, using traditional mechanisms.	⇒	Flexibility in Trust Management. Third Party Trust.

...so here comes OpenID for Verifiable Credentials!



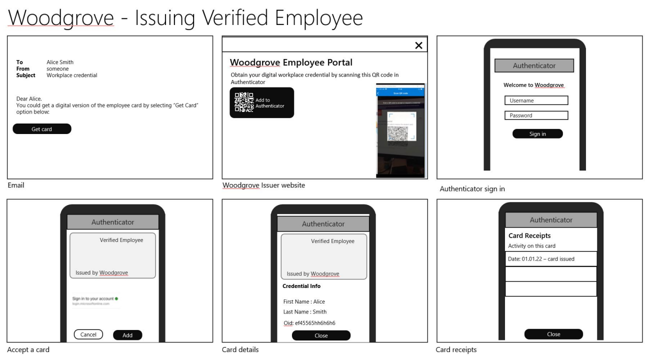
Adoption (selected use-cases)





Digital Identities - Mobile Driver's License (mDL)

Digital identities are supplementing and supplanting traditional physical identity cards. Customers, consumers of services, law enforcement, vendors, suppliers, and businesses may require a method of verifying a person via mobile device. This is not widely feasible due to the various, numerous and different technical implementations currently in place.



[The European Digital Identity Wallet Architecture and Reference Framework[1]](https://cloudsignatureconsortium.org/new-eu-eidas-regulation-a-quantum-leap-for-electronic-identity/) **(eIDAS ARF/EUDIW)** requires OID4VCI, OID4VP and SIOPv2 for online use-cases

[NIST National Cybersecurity Center of Excellence[2]](https://www.nccoe.nist.gov/projects/digital-identities-mdl) plans to implement reference implementation for OID4VP to present mdocs/mDL

[DIF JWT VC Presentation Profile[3]](https://identity.foundation/jwt-vc-presentation-profile/#workplace-credential) uses OID4VP for request and presentation of W3C JWT VCs and SIOPv2 for user authentication. Implementers: Ping Identity, Microsoft, IBM, Spruce, Auth0, Gen Digital

identiverse

[1] https://cloudsignatureconsortium.org/new-eu-eidas-regulation-a-quantum-leap-for-electronic-identity/
[2] https://www.nccoe.nist.gov/projects/digital-identities-mdl
[3] https://identity.foundation/jwt-vc-presentation-profile/#workplace-credential

#identiverse

Open Source libraries

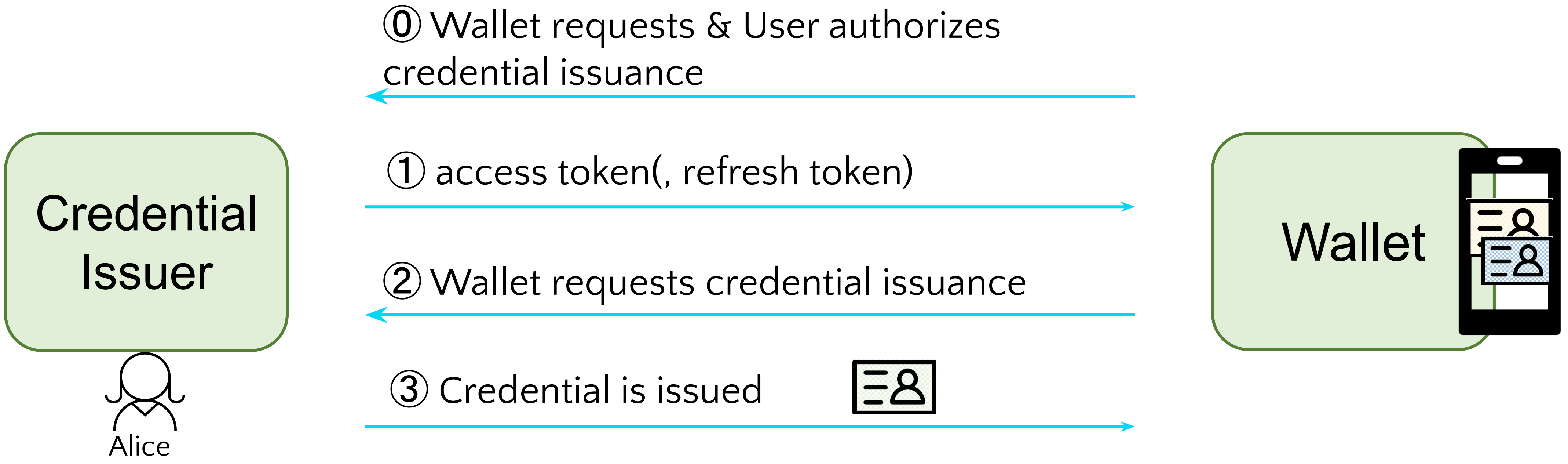
- Walt.id
    - https://github.com/walt-id/waltid-ssikit (Kotlin)
- Sphereon
    - https://github.com/Sphereon-Opensource/SIOP-OpenID4VP (Typescript)
    - https://github.com/Sphereon-Opensource/OpenID4VCI-client (Typescript)
    - https://github.com/Sphereon-Opensource/ssi-sdk (Typescript)
- Microsoft
    - https://github.com/microsoft/VerifiableCredential-SDK-Android (Kotlin)
    - https://github.com/microsoft/VerifiableCredential-SDK-iOS (Swift)
- Spruce
    - https://github.com/spruceid/oidc4vci-rs (Rust)
    - https://github.com/spruceid/oidc4vci-issuer (Rust)
- EBSI
    - https://api-pilot.ebsi.eu/docs/libraries (Javascript)
- Impierce Technologies
    - https://github.com/impierce/openid4vc (Rust)

Let us tell you more about the protocol

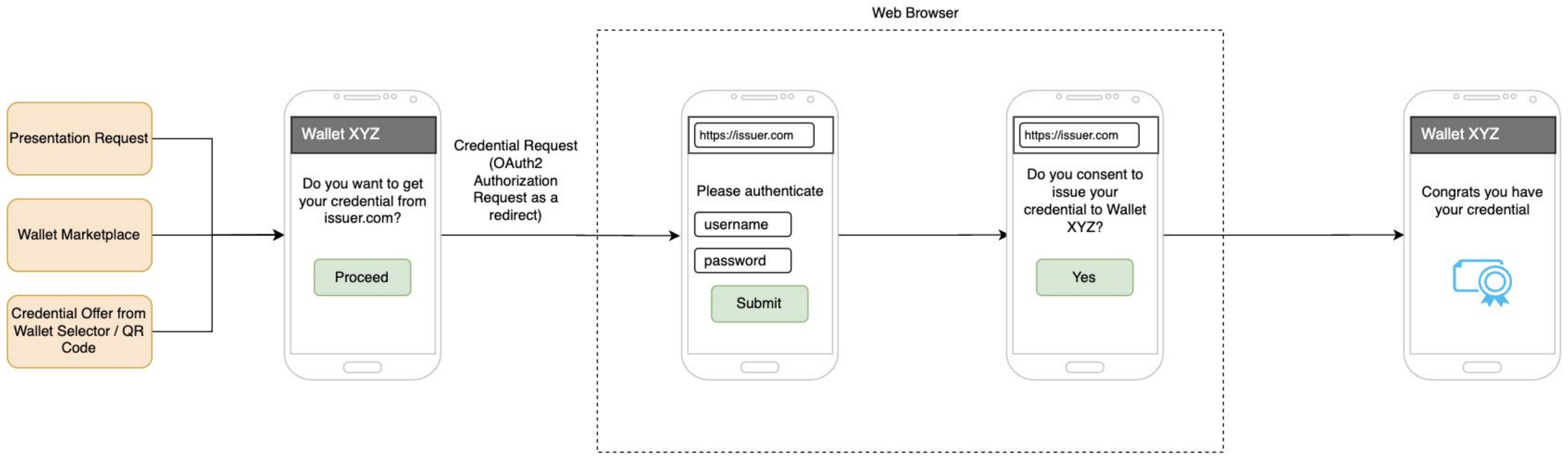
OpenID for Verifiable Credential Issuance (Highlights)

- It's an OAuth-protected API (Credential Endpoint at the Resource Server)
- Supports various Security levels (including high security with hardware bound keys)
- Various business requirements supported (ex. remote and in-person provisioning)
- Different user-experiences can be achieved (multiple ways to initiate the flow)
- Issuer can check Wallet's capabilities & Wallet can discover Issuer metadata

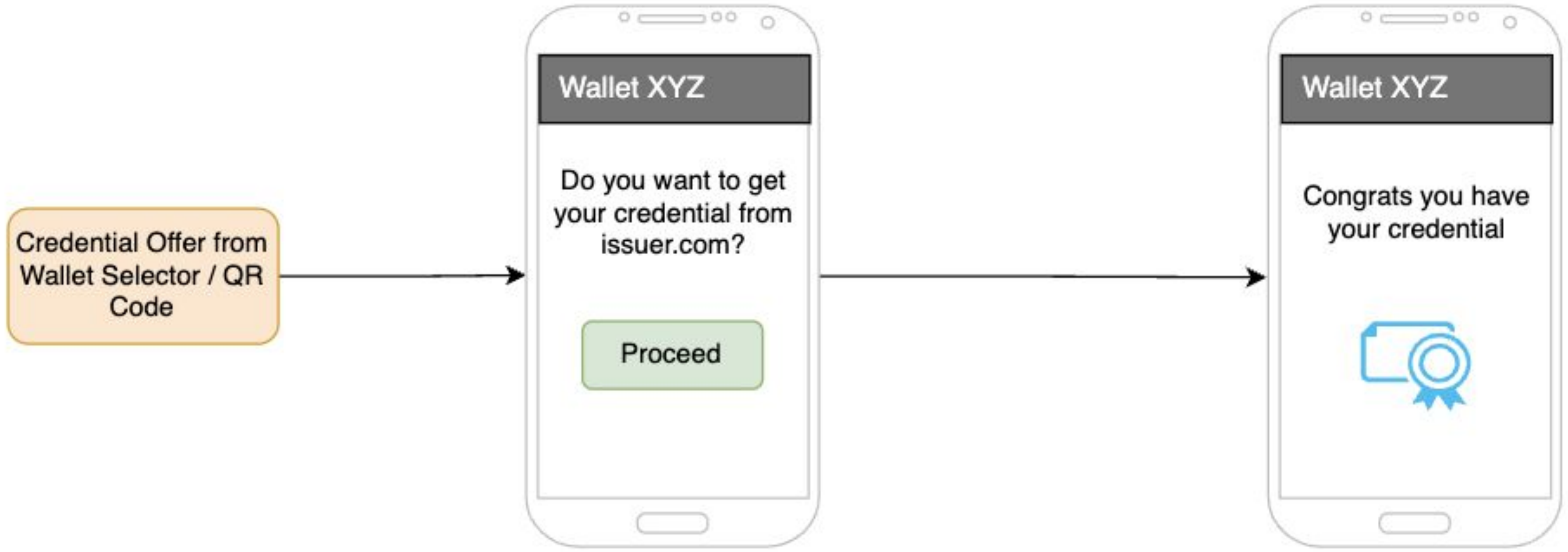
Protocol Flow


⓪ Wallet requests & User authorizes credential issuance
Credential Issuer
① access token(, refresh token)
Wallet
② Wallet requests credential issuance
③ Credential is issued
Alice

Authorization Code Flow


Web Browser
Presentation Request
Wallet Marketplace
Credential Offer from Wallet Selector / QR Code
Wallet XYZ
Do you want to get your credential from issuer.com?
Proceed
Credential Request (OAuth2 Authorization Request as a redirect)
https://issuer.com
Please authenticate
username
password
Submit
https://issuer.com
Do you consent to issue your credential to Wallet XYZ?
Yes
Wallet XYZ
Congrats you have your credential

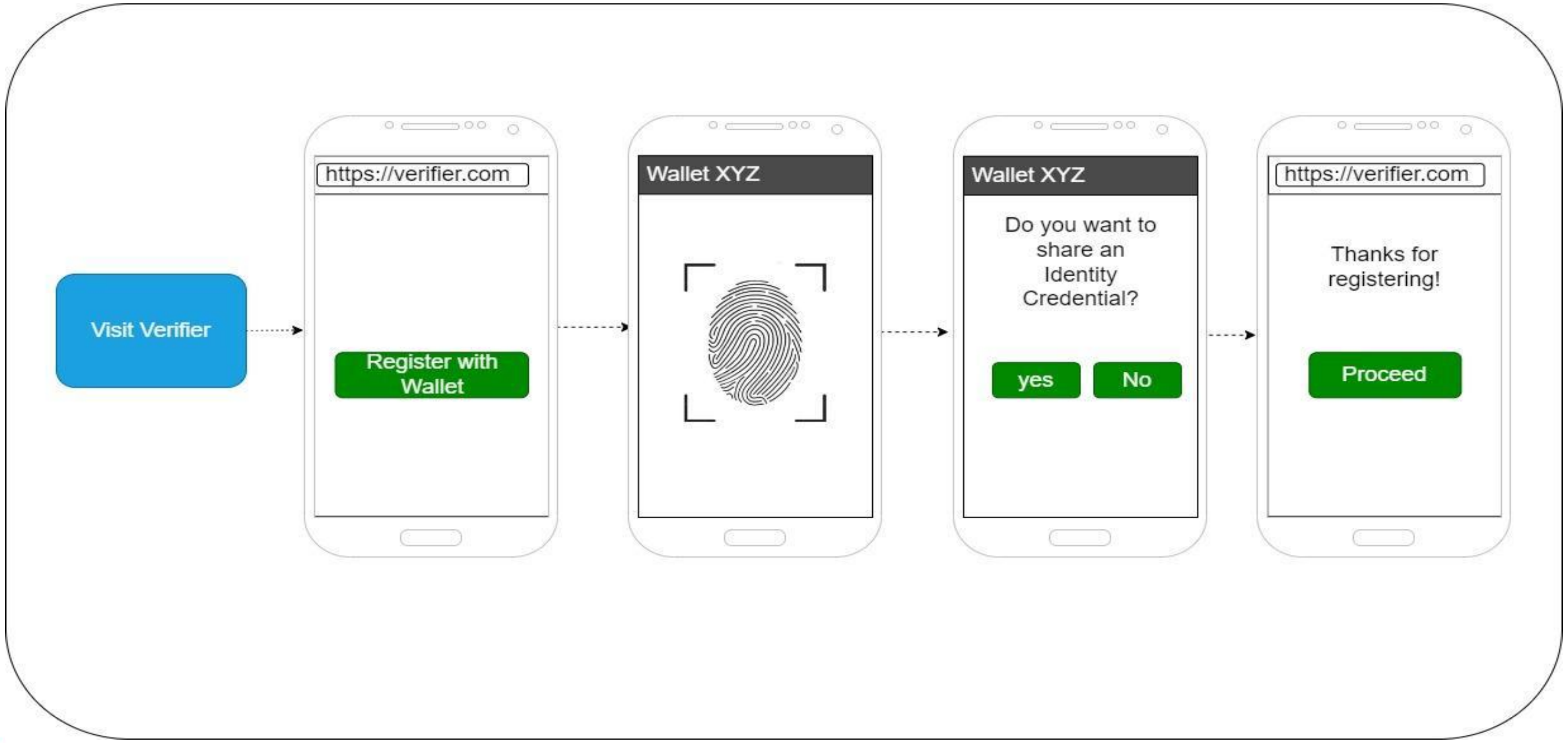
Pre-Authorized Code Flow


Credential Offer from Wallet Selector / QR Code
Wallet XYZ
Do you want to get your credential from issuer.com?
Proceed
Wallet XYZ
Congrats you have your credential

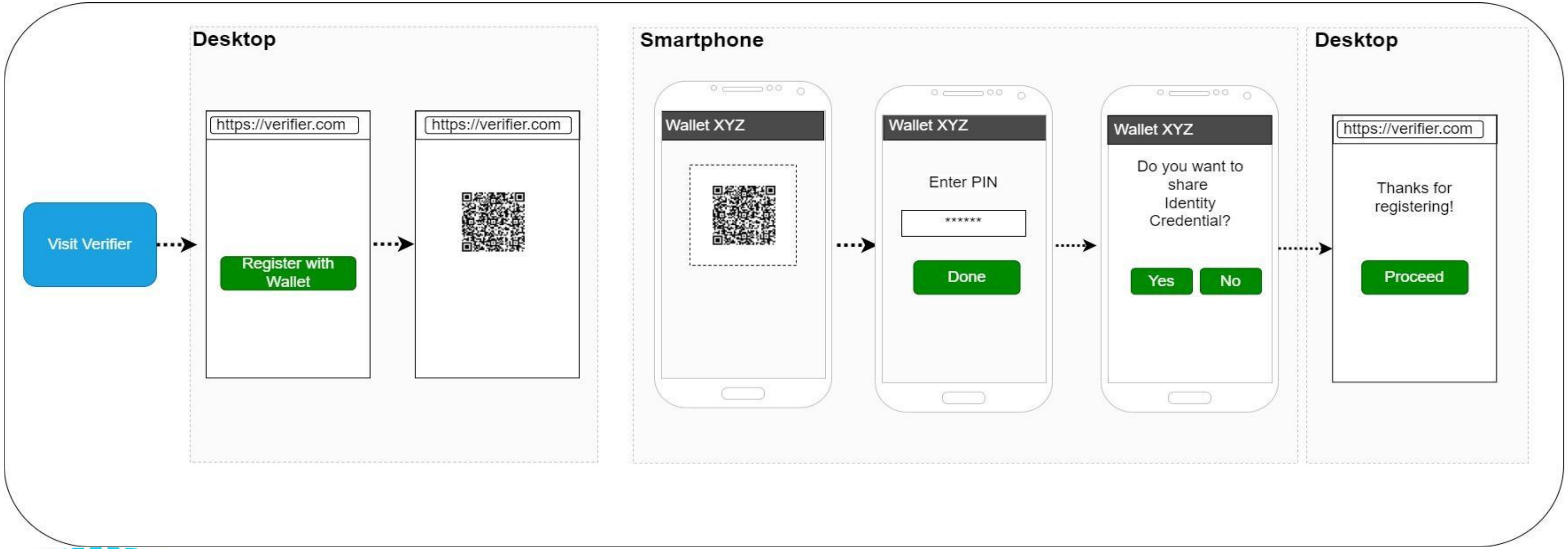
OpenID for Verifiable Presentations (Highlights)

- Designed for high degree of privacy
- Supports various Security levels (e.g. mutual authentication among the parties)
- Different user-experience can be achieved (same-device and cross-device)
- Presentation of multiple Credentials supported
- Various Wallet deployment models supported
    - All local to a native app
    - Cloud Wallet with a backend
    - Browser wallet

Same Device Presentation



Cross Device Presentation



Features of OpenID for Verifiable Credentials

1) It is NOT only about W3C Verifiable Credentials.

2) Does not require the usage of DLT (or Blockchain).

3) We are an open standardization community. Implementer's feedback is incorporated in an agile and transparent manner.

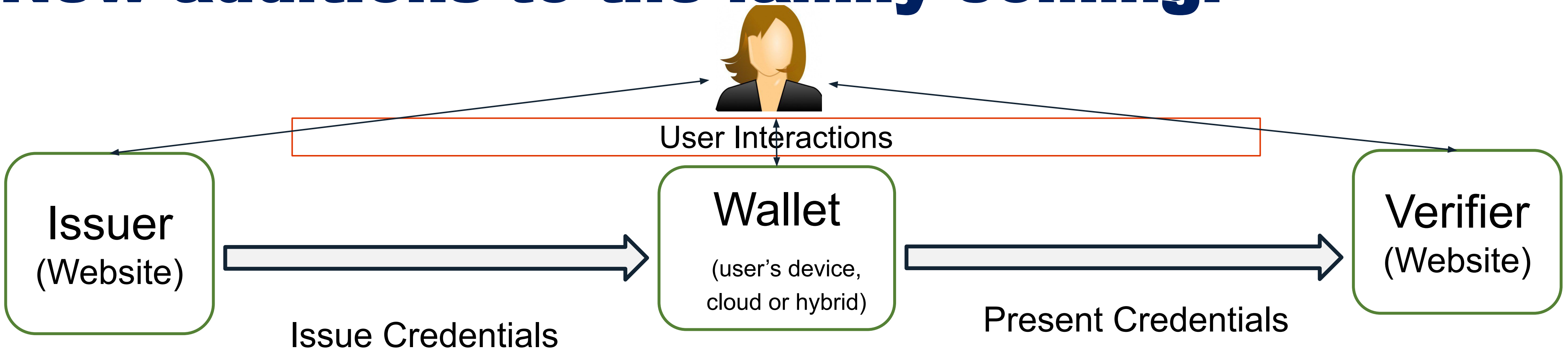
4) It is **modular and flexible** to cater for the needs of different legislations and use-cases.

5) Complemented by active work on profiles to help the developers interoperate.

🟦 Core specs

🟪 additional specs

New additions to the family coming!



OpenID for Verifiable Credential Issuance

OpenID for Verifiable Presentations

Self-Issued OP v2

OpenID for Verifiable Presentations over BLE

Security and Trust in OpenID for Verifiable Credentials

Certification Suite

High-Assurance Profile

High Assurance Profile of OpenID4VC with SD-JWT-VC

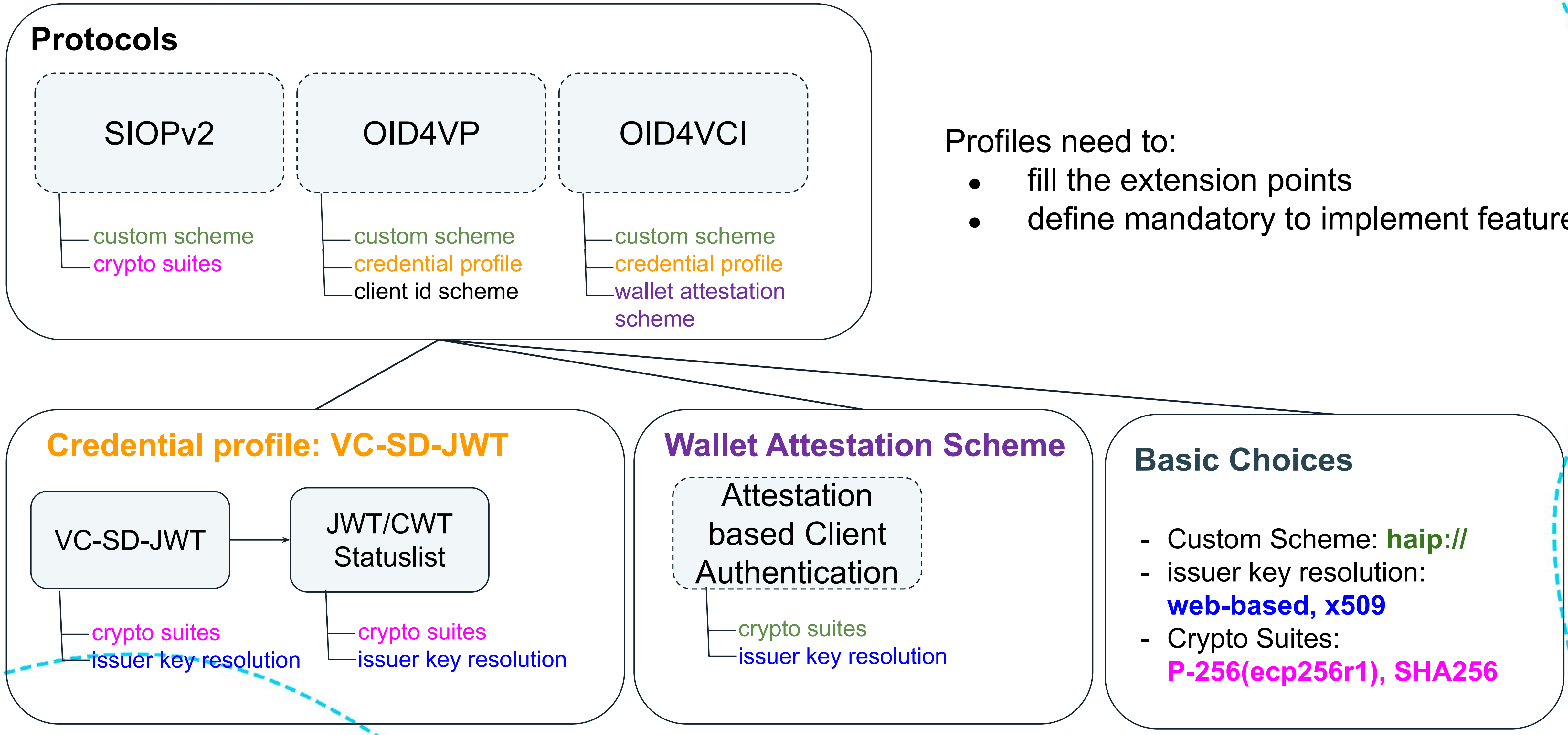
Profiling OpenID4VC

- OpenID4VC is a framework
- Interoperability requires “profiling”
- Profile defines:
    - mandatory to implement elements of the protocols, (e.g., grant types, etc.)
    - wallet invocation mechanism (i.e., custom URL scheme)
    - authentication requirements for Verifiers and Wallets
    - Credential Format(s) with
        - issuer identification and key resolution
        - holder key binding
    - Crypto algorithms

High Assurance Profile of OpenID4VC with SD-JWT-VC

- Interoperability across parties while being **privacy preserving** and **able to fulfill security and regulatory requirements**
- Intended audience
  - eIDAS ARF (through OIDF/EC liaison)
  - CA DMV wallet
  - Basis for OWF project(s)
  - Basis for Userinfo Interoperability profile
  - IDunion Tech Stack
  - GAIN PoC
  - Japanese government (Trusted Web project)
  - other jurisdictions
  - private companies / infrastructure companies

OID4VC High Assurance Interoperability Profile with SD-JWT VC



- Custom Scheme: **haip://**
- issuer key resolution: **web-based, x509**
- Crypto Suites: **P-256(ecp256r1), SHA256**

Profiles need to:

- fill the extension points
- define mandatory to implement features

OpenID for Verifiable Credential Issuance

- Pre-authorization code flow and authorization code flow are both required.
- Sender-constrained Tokens using DPoP required
- Credential Offer
  - for both pre-authorization code flow and authorization code
  - custom scheme “haip://” for wallet invocation
- Authorization at Issuer with Pushed Authorization Requests (PAR)
- Wallet Authentication with sender-constrained JWTs
- "scope” parameter to requesting authorization for credential issuance
- Only required endpoint is Credential Endpoint
  - Batch Credential Endpoint is required for dual issuance of SD-JWT-VC and mdocs

OpenID for Verifiable Presentations

- custom scheme “haip://” for wallet invocation.
- Response type: “vp_token”.
- Response mode: “direct_post” with redirect_uri.
- Using “request_uri” to send Authorization Request is required.
- Presentation Definition is sent using “presentation_definition” parameter
- Subset of the Presentation Exchange Syntax in order to simplify implementation and prevent security issues
- Verifier Authentication with
  - x.509 Certificates or
  - Sender-constrained JWTs

SIOP v2

- custom URL scheme “haip://” for wallet invocation
- subject_syntax_types_supported value MUST be urn:ietf:params:oauth:jwk-thumbprint
- Verifier Authentication with
  - x.509 Certificates or
  - Sender-constrained JWTs

Credential Format

- SD-JWT VC with JSON payload (“typ”: “vc+sd-jwt”)
  - both compact and JSON serialization
- Definition of mapping to VCDM base media type
- Issuer identification and key resolution
  1. **Web PKI based**: iss=issuer URL used to obtain jwks_uri + key id in the `kid` JWS header
  2. **x.509:** iss=SAN in x.509 cert + x.509 cert chain in the `x5c` JWS header
- Holder binding:
  - `cnf` JWT claim with jwk
- Credential Revocation: Bitmap type style Status list using JWTs

SD-JWT VC
with web PKI based Issuer key resolution

```
{
 "alg": "ES256",
 "typ": "vc+sd-jwt",
 "kid":"4"
}
```

```
{
 "iss": "https://credential-issuer.example.com",
 "iat": 1516239022,
 "exp": 1516247022,
 "type": "Identity",
 "_sd": [
   "UiuRGkTW7e_5UQauGeQRQdF8u3WYevS4Fs0IuB_DgYM",
   "tmPlXq0MID-oRXbUNHyoVZrc9Qkm8cwJTohVyOVlUgQ",
   "vTz0JI103v4k4pKIloT83Yzi33L1SdZlWBPmsfJBefk"
 ],
 "_sd_alg": "sha-256",
 "cnf": {
   "jwk": {
     "kty": "EC",
     "crv": "P-256",
     "x":
"TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
     "y":
"ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
   }
 }
}
```

Crypto

- For signing and signature validation:
  - ES256 algorithm and ECDSA keys using the P-256 (secp256k1)
- As hash algorithm to generate and validate the digests in the SD-JWT VC:
  - SHA256

Call to Action: Implement, Implement, Implement

The information can be found at https://openid.net/openid4vc/

THANK YOU!