HIGH LEVEL SYSTEM DESIGN

3 part process for decentralized identity management:

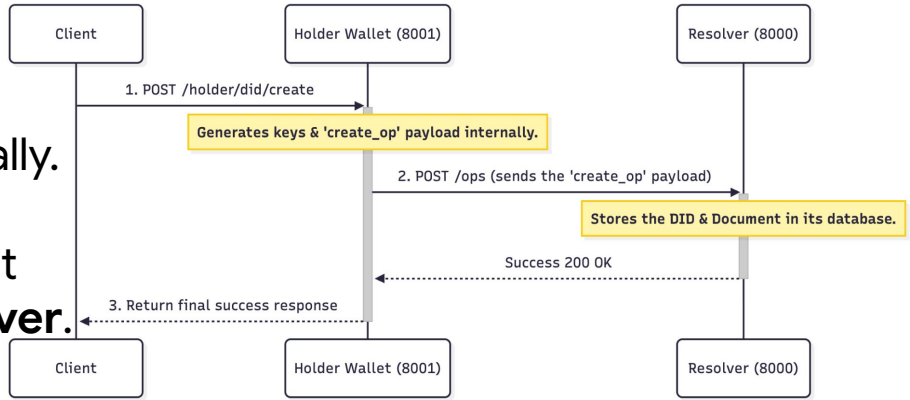
- DID creation
- VC issuance
- VC verification.

The key components involved in these flows are:

- **Client (for Holder)**
- **Holder Wallet (8001)**
- **Resolver (8000)**
- **Issuer API (8003)**
- **Issuer Wallet (8002)**
- **Verifier (8004)**

DID Creation Flow Overview

- Client sends `POST /holder/did/create` request to **Holder Wallet**.
- Holder Wallet **generates key pairs** and creates the **`create_op` payload** internally.
- Holder Wallet sends `POST /ops` request (with the `create_op` payload) to **Resolver**.



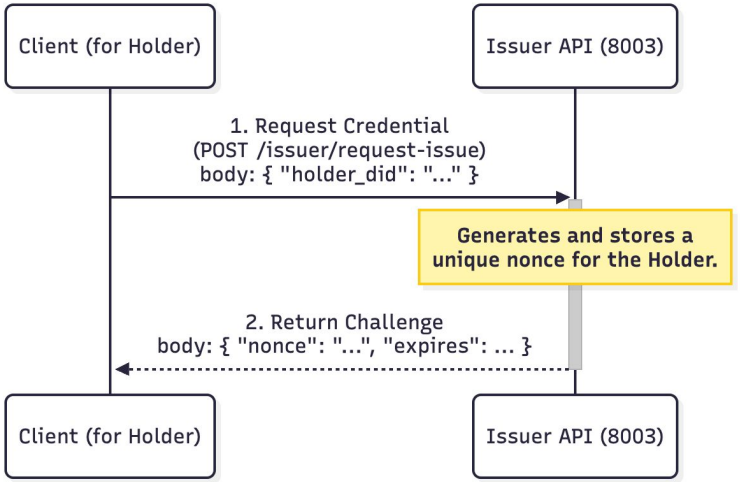
- Resolver **stores the DID and DID Document** in its internal database.
- Resolver responds with **200 OK (Success)** to Holder Wallet.
- Holder Wallet returns the **final success response** to the Client.

Credential Request Flow (Holder → Issuer)

- Client (on behalf of Holder) sends
  `POST /issuer/request-issue`
  (body includes the **Holder's DID**)

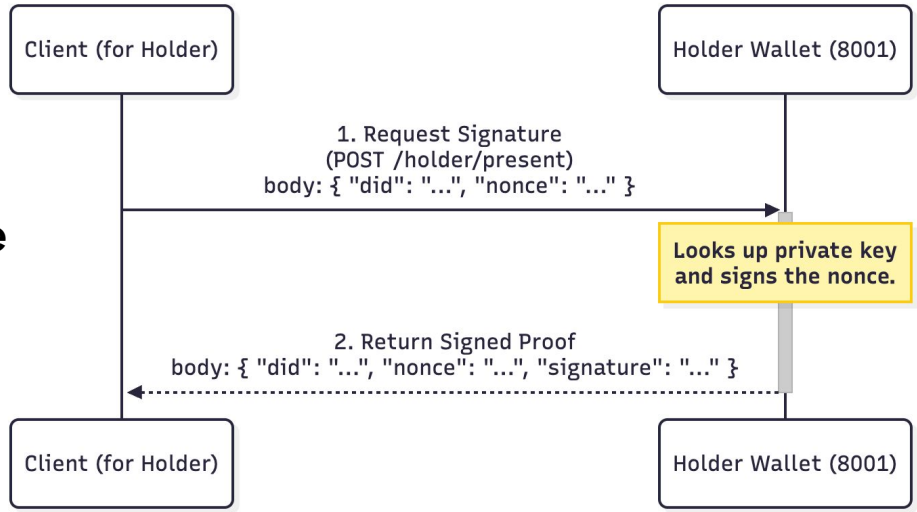
- **Issuer API (port 8003)**
  → **Generates a unique nonce** for the Holder.
  → Stores it internally with an expiry timestamp.

- Issuer returns the **challenge response** to the client:
  `{ "nonce": "...", "expires": "..." }`



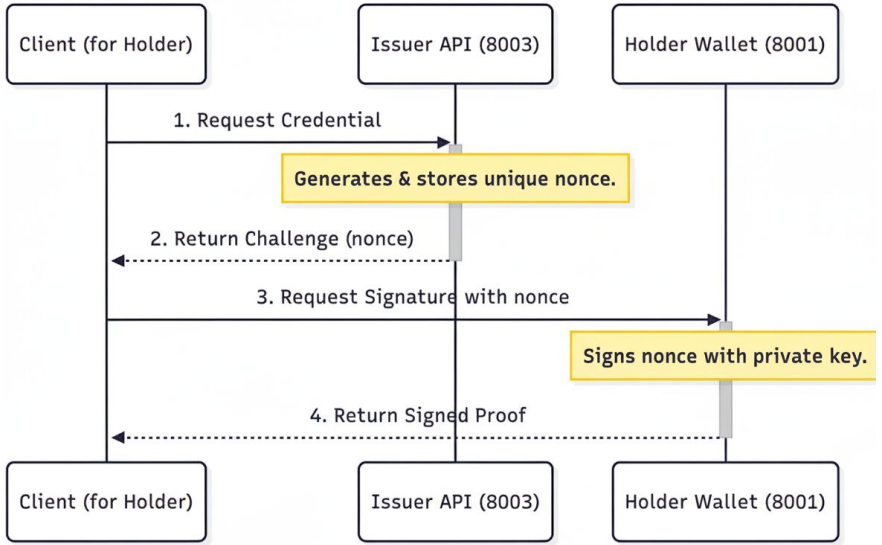
Signature (Proof of Possession) Flow

- Client sends `POST /holder/present` (body includes the **DID** and **nonce** received from Issuer.
- **Holder Wallet retrieves the private key** associated with the DID & **signs the nonce** to prove key ownership.
- Holder Wallet returns the **signed proof** `{ "did": "...", "nonce": "...", "signature": "..." }`



Credential Request + Proof Flow

- Client sends **credential request** → `POST/issuer/request-issue`.
- **Issuer API generates & stores a unique nonce** for Holder & returns **nonce** to Client.
- Client forwards the **nonce** to **Holder Wallet** with a **signature request**.
- **Holder Wallet**
  → **Looks up Holder's private key**.
  → **Signs the nonce** to prove control of DID.
  → Returns the **signed proof**.
- Client sends the signed proof back to **Issuer API** for verification.



Verifiable Credential Issuance Flow

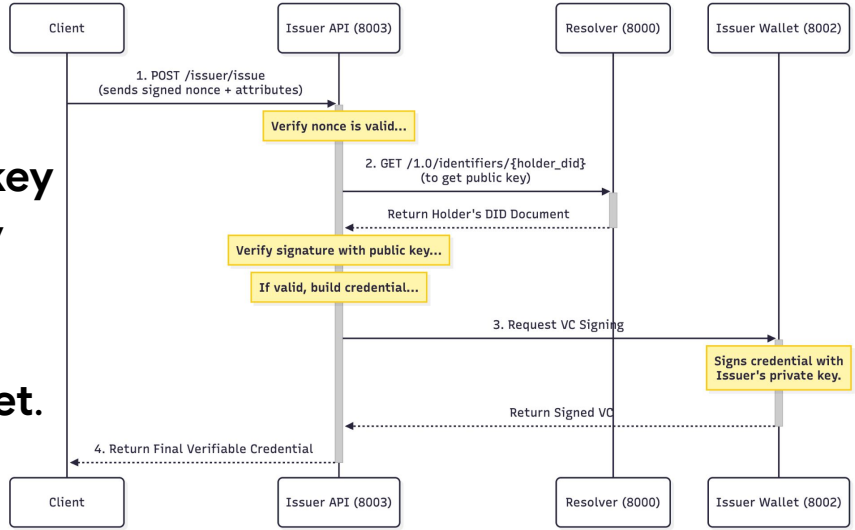
- Client sends `POST /issuer/issue`
  (includes **signed nonce + credential attributes**)

- **Issuer API → Verifies nonce validity**

  → **Resolving Holder's DID Doc** to get **pub key**
  → **Verifies Holder's sign** using retrieved key
  → If valid, **builds VC**

- Issuer requests **VC signing** from **Issuer Wallet**.
  → Wallet **signs cred** using **Issuer's pvt key**.

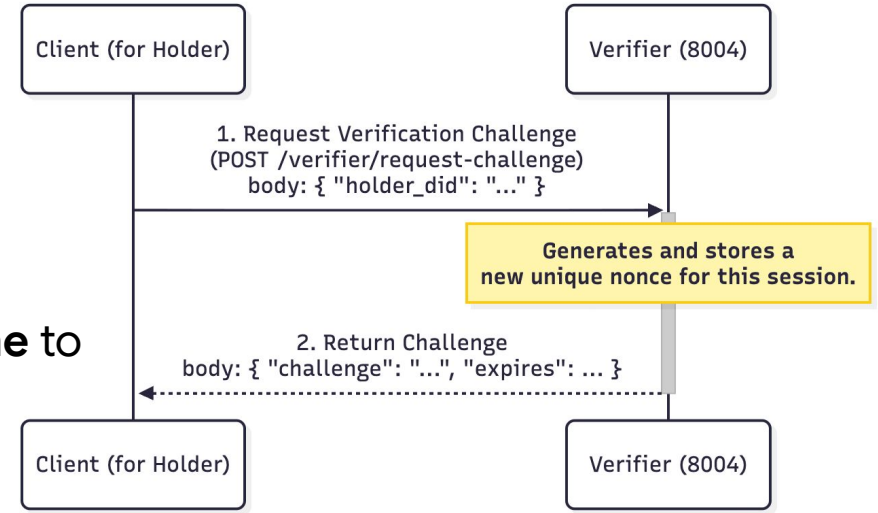
- Issuer API returns the **final signed VC** to the Client.



Verification Challenge Flow

- Client sends
  `POST /verifier/request-challenge`
  (body includes the **Holder's DID**)

- **Verifier API**
  → **Generates and stores a new unique nonce** for the verification session.
  → Returns the **challenge** and **expiry time** to the Client.



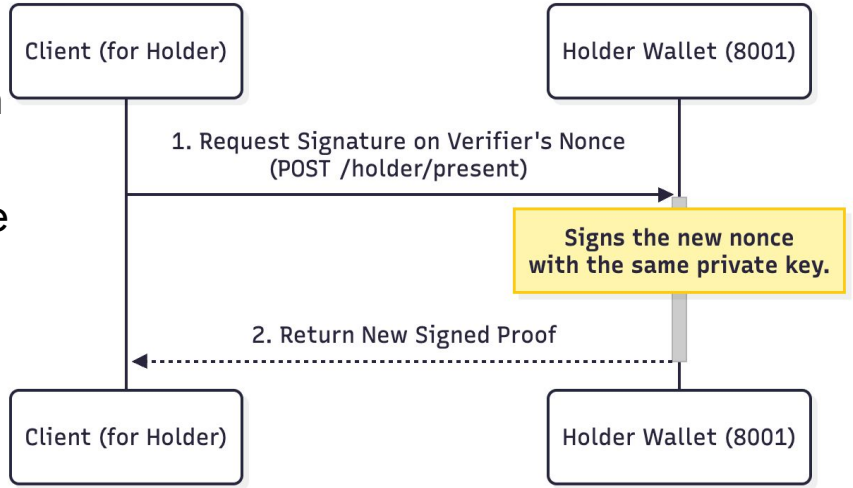
.

Proof of Ownership (Holder Signing Phase)

- Client sends
  `POST /holder/present`
  (body includes **Verifier's nonce)**

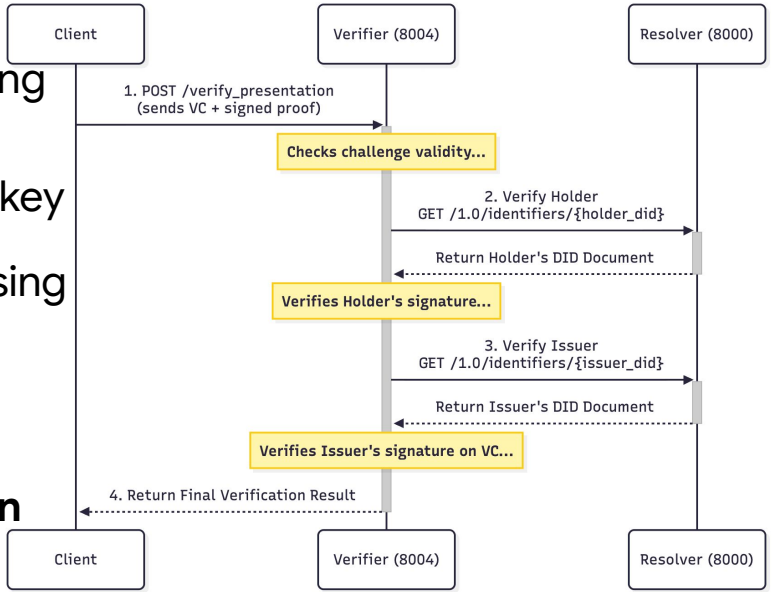
- **Holder Wallet**
  → **Looks up the private key** associated with the Holder's DID.
  → **Signs the verifier's nonce** using the same private key used during credential issuance.

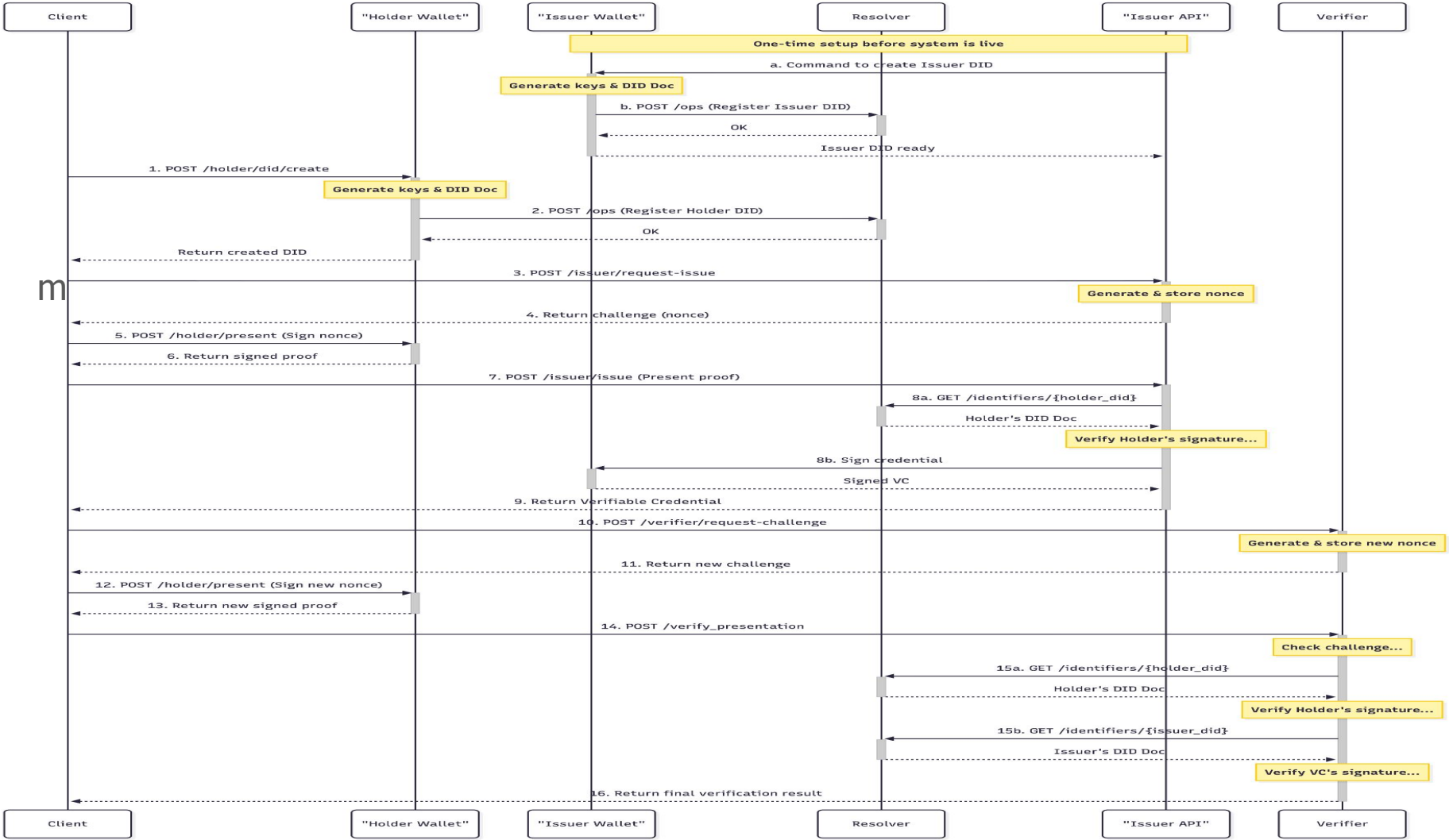
- Wallet returns the **new signed proof** back to the Client:
  `{ "did": "...", "nonce": "...", "signature": "..." }`



Verifiable Credential Validation Flow

- Client sends `POST /verify_presentation` (includes **VC + signed proof**.
- **Verifier**
  → **Checks challenge validity**
  → Requests **Holder's DID Doc** from **Resolver** using `GET /1.0/identifiers/{holder_did}`.
  → **Verifies Holder's sign** on challenge using pub key
- Verifier fetches **Issuer's DID Doc** from Resolver using `GET /1.0/identifiers/{issuer_did}`.
  → **Verifies Issuer's signature** on the VC.
- If all checks pass, Verifier returns **Final Verification Result (Success)**.




Client
"Holder Wallet"
"Issuer Wallet"
Resolver
"Issuer API"
Verifier
One-time setup before system is live
a. Command to create Issuer DID
Generate keys & DID Doc
b. POST /ops (Register Issuer DID)
OK
Issuer DID ready
1. POST /holder/did/create
Generate keys & DID Doc
2. POST /ops (Register Holder DID)
OK
Return created DID
3. POST /issuer/request-issue
Generate & store nonce
4. Return challenge (nonce)
5. POST /holder/present (Sign nonce)
6. Return signed proof
7. POST /issuer/issue (Present proof)
8a. GET /identifiers/{holder_did}
Holder's DID Doc
Verify Holder's signature...
8b. Sign credential
Signed VC
9. Return Verifiable Credential
10. POST /verifier/request-challenge
Generate & store new nonce
11. Return new challenge
12. POST /holder/present (Sign new nonce)
13. Return new signed proof
14. POST /verify_presentation
Check challenge...
15a. GET /identifiers/{holder_did}
Holder's DID Doc
Verify Holder's signature...
15b. GET /identifiers/{issuer_did}
Issuer's DID Doc
Verify VC's signature...
16. Return final verification result
Client
"Holder Wallet"
"Issuer Wallet"
Resolver
"Issuer API"
Verifier