

Blockchain for Identity Management



Classical Authentication



::: Introduction

Autenticazione: Implementazione di un accesso sicuro alle risorse attraverso la verifica dell'identità degli utenti. Essa deve essere validata e verificata.

Autorizzazione: Il diritto di accedere ad alcuni servizi o funzionalità è garantito ad un utente autorizzato sulla base di diversi fattori, come il ruolo all'interno di un'organizzazione o alcune policies più specifiche.

Accountability: Ogni attività svolta dall'utente deve essere salvata salvata in modo persistente e tracciata in modo da permettere una futura analisi forense per riscontrare possibili autori di attacchi informatici.



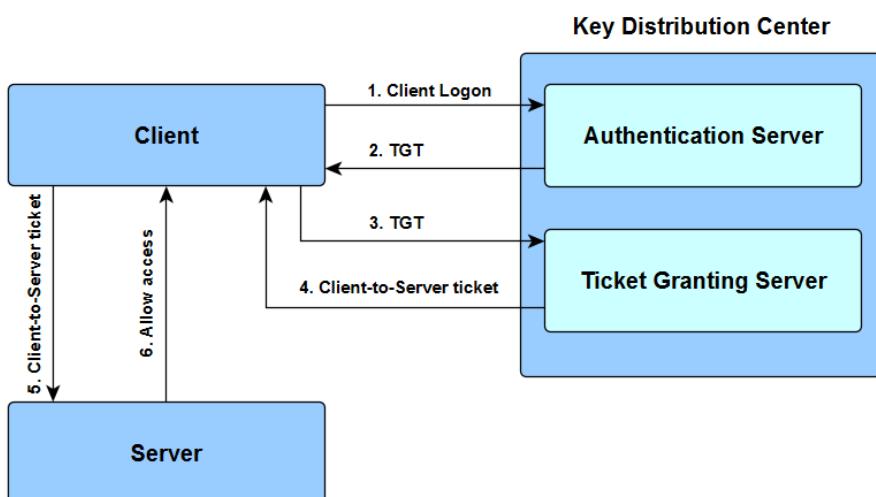
::: IdM – 1/7

Un'identità è la rappresentazione di un entità in un contesto particolare, composta da un identificatore unico e un insieme di attributi legati all'utente, e necessita di sistemi adeguati noti come Identity Manager (IdM), responsabili del controllo del ciclo di vita delle identità, della verifica della veridicità delle identità dichiarate e dello scambio di attributi di identità con sistemi paritetici. Sono stati proposti diversi modelli e architetture per la rappresentazione delle identità e la strutturazione dell'IdM: I modelli di identità basati su password sono i più semplici e comunemente utilizzati: ogni utente ha un identificatore unico nel sistema e una password data, da utilizzare per effettuare la propria autenticazione.



::: IdM – 2/7

1. Gli utenti forniscono in modo sicuro identificatori e password a IdM.
2. Se IdM trova una corrispondenza, l'autenticazione ha successo e ogni utente riceve in cambio un token di accesso.
3. Un tale token è una stringa opaca che identifica temporaneamente un utente e può essere utilizzata per accedere ai servizi che si fidano dell'IdM che lo rilascia.

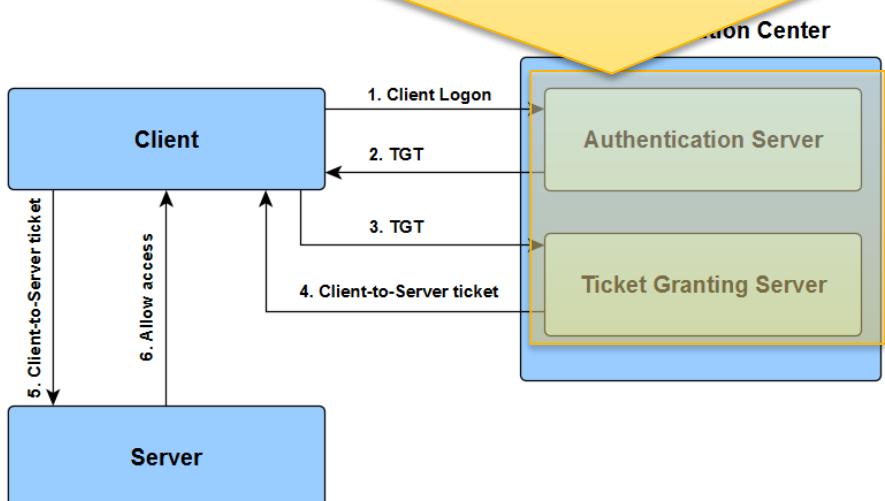


Un esempio è Kerberos, un protocollo di autenticazione di rete. Attualmente è la tecnologia di autenticazione predefinita utilizzata da Microsoft per autenticare gli utenti per i servizi all'interno di una rete locale.

::: IdM – 2/7

1 Gli utenti forniscono in modo sicuro identificatori e password a IdM

- Authentication Server (AS) per l'autenticazione iniziale e ne rilascia i ticket di concessione (TGT) per gli utenti.
- Server di Concessione dei Ticket (TGS) per rilasciare i ticket di servizio basati sui ticket di concessione iniziali (TGT).

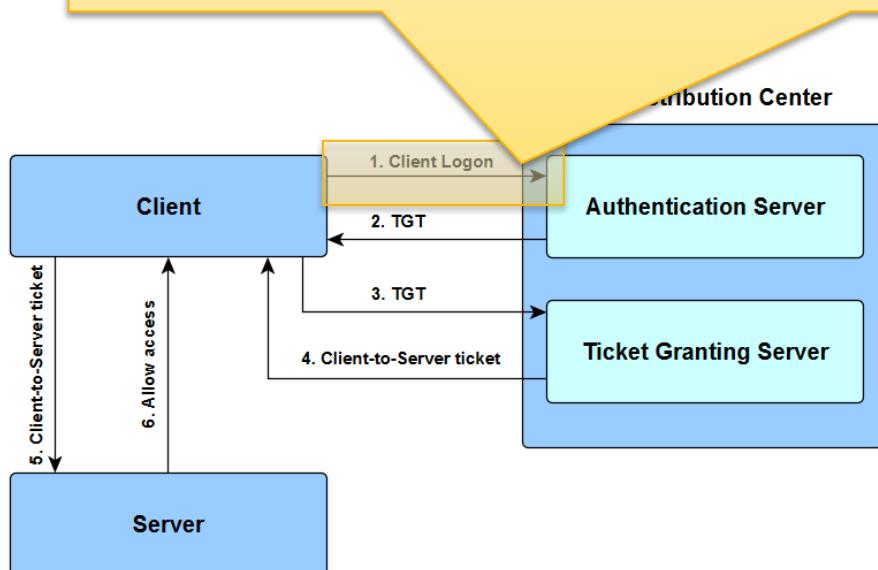


Un esempio è Kerberos, un protocollo di autenticazione di rete. Attualmente è la tecnologia di autenticazione predefinita utilizzata da Microsoft per autenticare gli utenti per i servizi all'interno di una rete locale.



::: IdM – 2/7

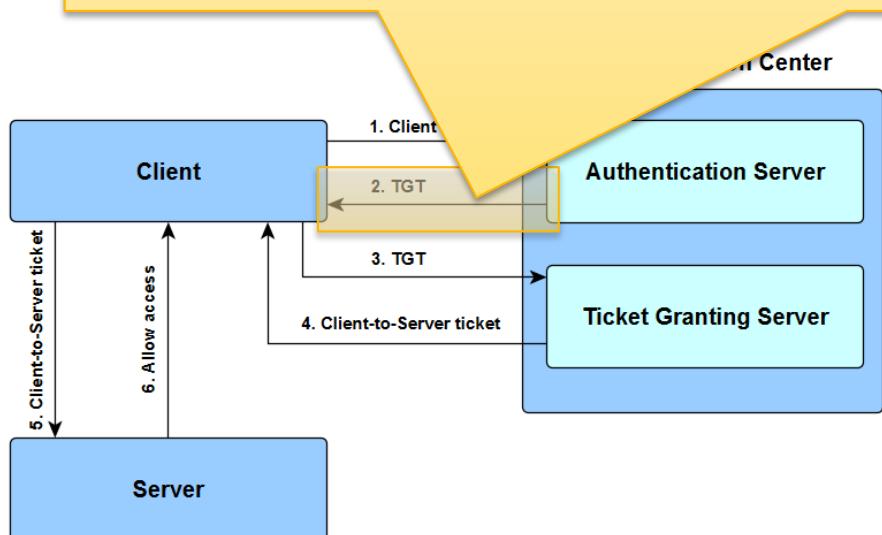
Il client invia una richiesta al Server di Autenticazione (AS) con l'ID utente in chiaro e chiede l'accesso a un server a nome dell'utente. Questa richiesta è parzialmente criptata con una chiave segreta che è la password dell'utente cliente.



Un esempio è Kerberos, un protocollo di autenticazione di rete. Attualmente è la tecnologia di autenticazione predefinita utilizzata da Microsoft per autenticare gli utenti per i servizi all'interno di una rete locale.

.. IdM ... 2/7

L'AS recupera la chiave segreta (la password dell'utente) dal database dell'utente in base all'ID ricevuto e utilizza la sua password come chiave per decifrare la richiesta. In questo modo, l'utente viene verificato. Successivamente, l'AS invia un token criptato con un'altra chiave segreta condivisa tra l'AS e il TGS

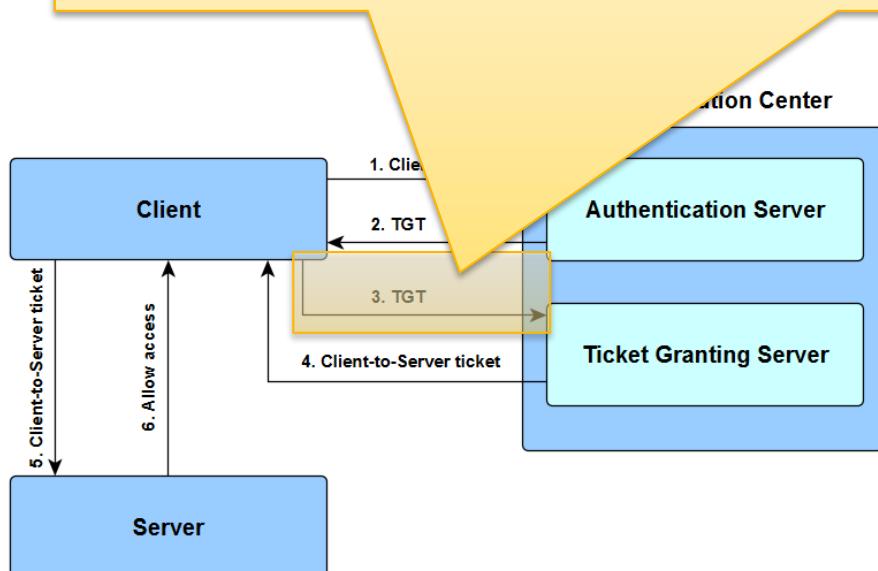


Un esempio è Kerberos, un protocollo di autenticazione di rete. Attualmente è la tecnologia di autenticazione predefinita utilizzata da Microsoft per autenticare gli utenti per i servizi all'interno di una rete locale.

::: IdM – 2/7

1. Gli utenti forniscono in modo sicuro identificatori e password a IdM.
2. Se IdM trova una corrispondenza, l'autenticazione ha successo e ogni utente riceve in cambio un token di accesso.

I client inviano il TGT cifrato al TGS richiedendo l'accesso al server.



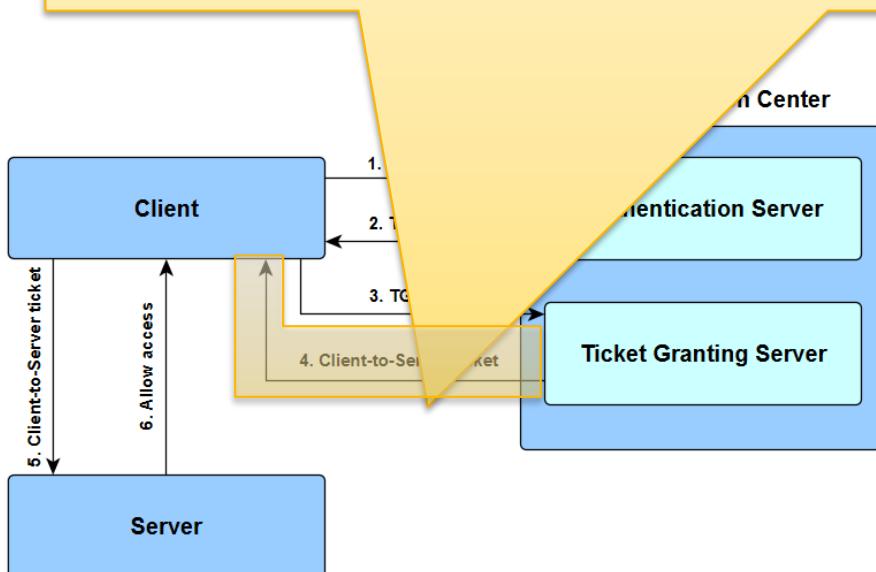
Un esempio è Kerberos, un protocollo di autenticazione di rete. Attualmente è la tecnologia di autenticazione predefinita utilizzata da Microsoft per autenticare gli utenti per i servizi all'interno di una rete locale.



::: IdM – 2/7

1. Gli utenti forniscono in modo sicuro identificatori e password a IdM.

Il TGS decifra il token con una chiave segreta condivisa con il Server di Autenticazione (AS) e invia un token di accesso al cliente, il quale è cifrato con un'altra chiave segreta condivisa con il server.



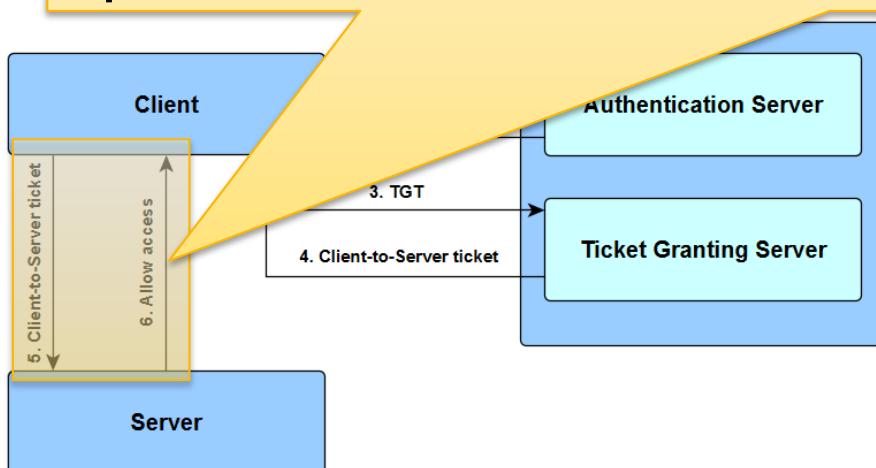
Un esempio è Kerberos, un protocollo di autenticazione di rete. Attualmente è la tecnologia di autenticazione predefinita utilizzata da Microsoft per autenticare gli utenti per i servizi all'interno di una rete locale.



::: IdM – 2/7

1. Gli utenti forniscono in modo sicuro identificatori e password a IdM.
2. Se IdM trova una corrispondenza, l'autenticazione ha successo e ogni utente riceve in cambio un token di accesso

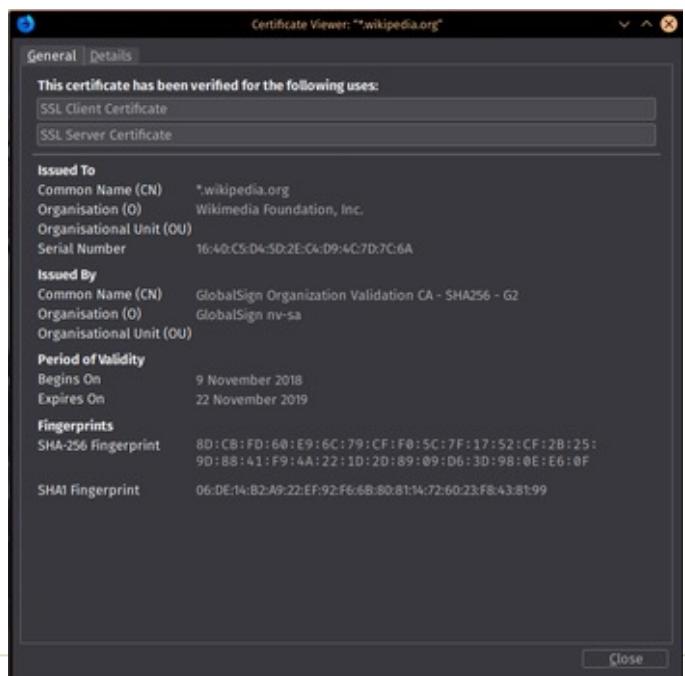
I client inviano una richiesta al server con il token Kerberos cifrato. Successivamente, il server consente l'accesso alle risorse richieste al cliente per un certo periodo di tempo specificato nel token.



autenticazione di rete. Attualmente è la tecnologia di autenticazione predefinita utilizzata da Microsoft per autenticare gli utenti per i servizi all'interno di una rete locale.

::: IdM – 3/7

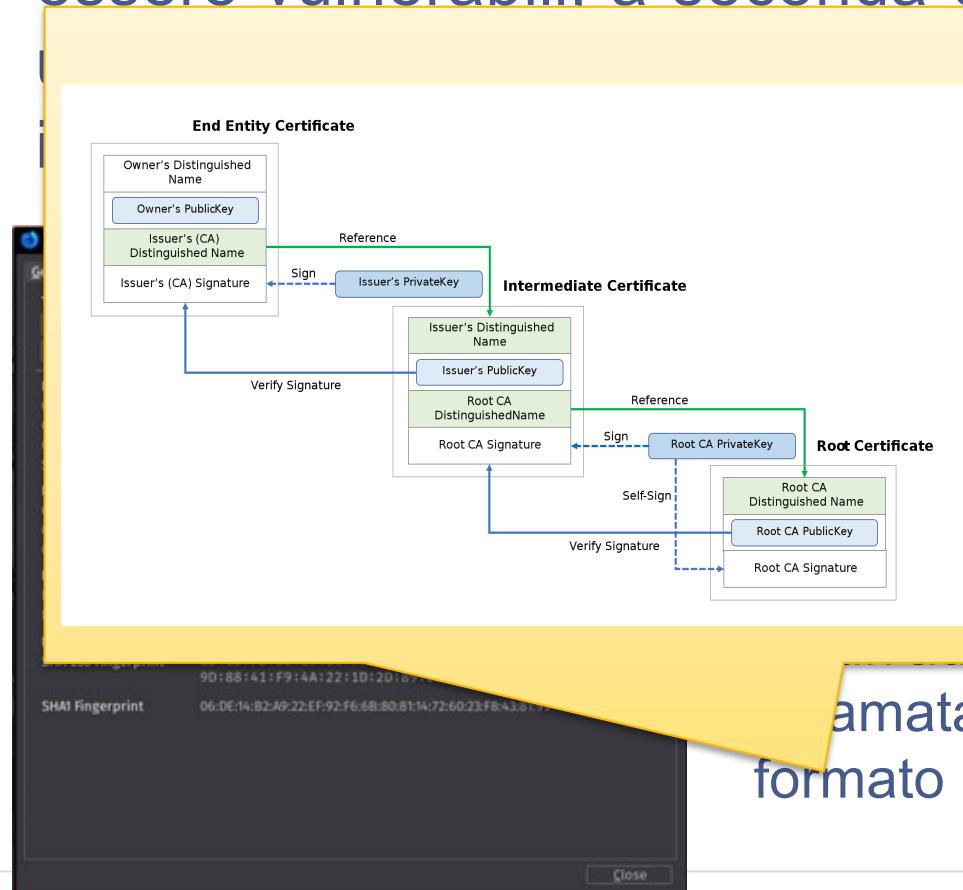
I metodi di autenticazione basati su password sono noti per essere vulnerabili, a seconda di quanto "forte" sia la password utilizzata, ad esempio, quanto facilmente può essere indovinata.



Un sistema di gestione delle identità basato su certificati fa uso di certificati digitali e crittografia a chiave pubblica. Un certificato digitale è una struttura dati che contiene l'identificatore dell'utente e la sua chiave pubblica e attributi di identità, verificati e firmati da un'autorità di certificazione di fiducia, chiamata Certification Authority (CA), in un formato inalterabile.

... IdM – 3/7

I metodi di autenticazione basati su password sono noti per essere vulnerabili. a seconda di quanto "forte" sia la password



Le CA sono organizzate in una gerarchia per gestire un grande numero di utenti. Ogni CA ha il proprio certificato firmato dalla CA di livello superiore. Solo la Root CA ha un certificato auto-firmato.

...amata Certification Authority (CA), in un formato inalterabile.

::: IdM – 4/7

Dopo che un utente ha ottenuto un certificato, può fornirlo a un fornitore di servizi, il quale può verificare la sua autenticità interagendo con l'autorità che lo ha emesso. Se il certificato viene verificato come autentico, l'utente viene autenticato.

X.509 è uno standard che definisce il formato dei certificati a chiave pubblica e stabilisce:

- Liste di revoca dei certificati, per distribuire dettagli su certificati considerati non validi da un'autorità di firma,
- Algoritmo di validazione del percorso di certificazione, che consente ai certificati di essere firmati da certificati di CA intermedi, i quali a loro volta sono firmati da altri certificati, raggiungendo infine un trust anchor



::: IdM – 4/7

Dopo che un utente ha ottenuto un certificato, può consegnarlo a un fornitore di servizi, che può verificare la sua autenticità interagendo con l'autorità che lo ha emesso. Se il certificato viene verificato come autentico, l'utente viene autenticato.

Version
Serial Number
Signature Algorithm Identifier
Issuer Name
Validity Period
Subject Name
Public Key Information
Issuer Unique ID
Subject Unique ID
Extensions

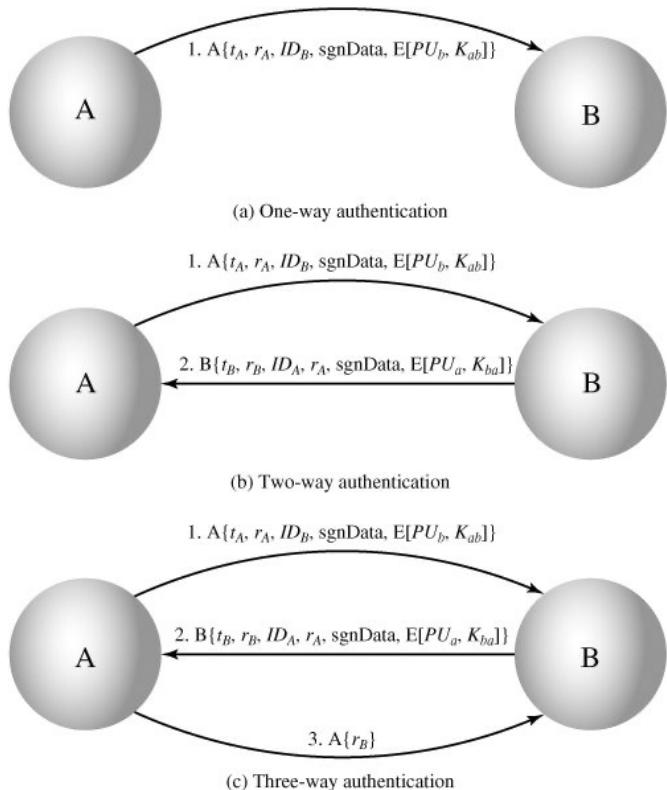


Certificate:
Data:
Version: 1 (0x0)
Serial Number: 7829 (0x1e95)
Signature Algorithm: md5WithRSAEncryption
Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc, OU=Certification Services Division, CN=Thawte Server CA/emailAddress=server-certs@thawte.com
Validity
Not Before: Jul 9 16:04:02 1998 GMT
Not After : Jul 9 16:04:02 1999 GMT
Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala, OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)
Modulus (1024 bit):
00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb: 33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
66:36:d0:8e:56:12:44:ba:75:eb:c8:1c:9c:5b:66: 70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
16:94:6:ee:f4:d5:6:f:d5:ca:b3:47:5:e:1b:0c:7b: c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
8:f:a0:21:c7:4c:d0:16:65:00:c1:f:d7:b8:80:e3: d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
e8:35:1c:9e:27:52:7e:41:f
Exponent: 65537 (0x10001)
Signature Algorithm: md5WithRSAEncryption
93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67: d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1: 5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
8:f0:ef:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:68:9f



::: IdM – 5/7

X.509 include anche tre procedure alternative di autenticazione destinate all'uso in una varietà di applicazioni.

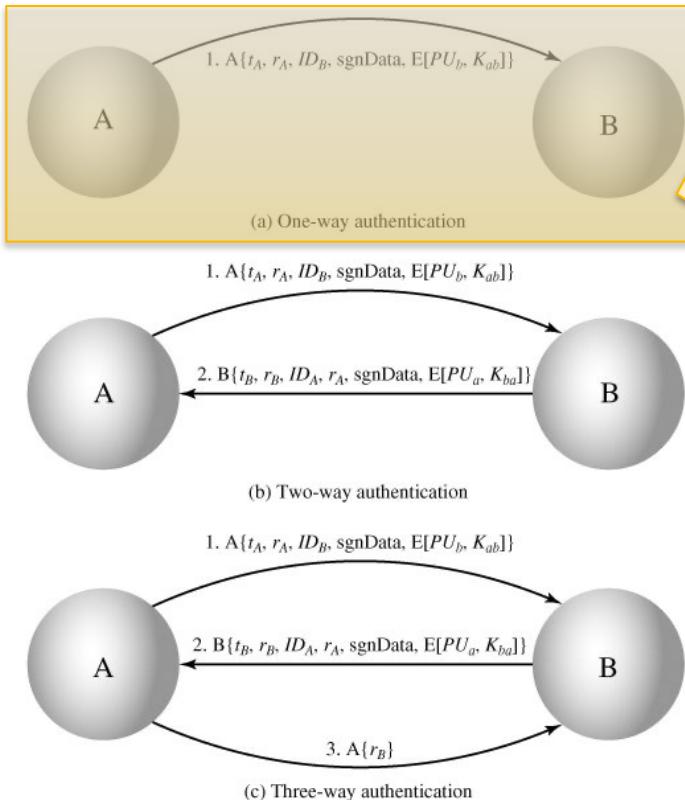


Le due parti devono conoscere la chiave pubblica dell'altra, sia ottenendone i certificati dalla directory che perché il certificato è incluso nel messaggio iniziale inviato da ciascuna parte.



::: IdM – 5/7

X.509 include anche tre procedure alternative di autenticazione destinate all'uso in una rete.

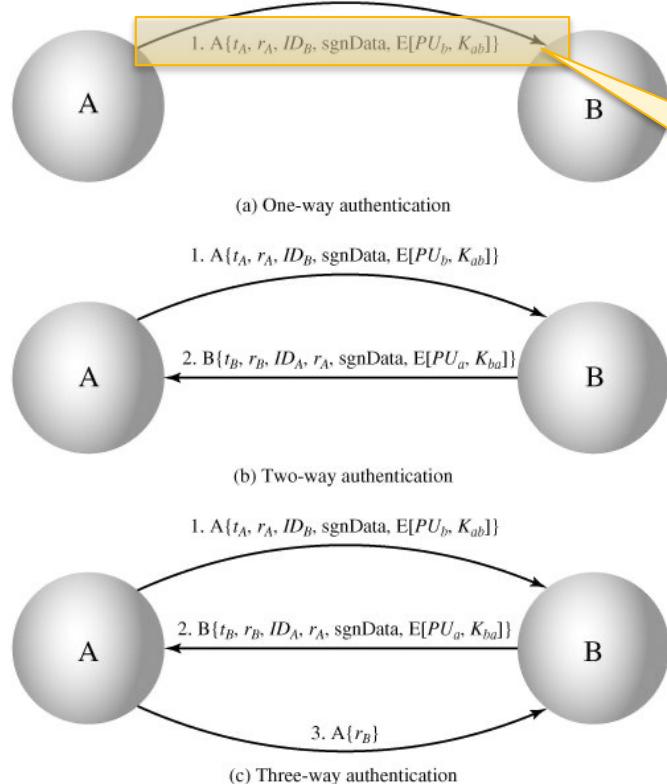


L'autenticazione unidirezionale coinvolge un singolo trasferimento di informazioni da un utente (A) a un altro (B) e stabilisce quanto segue:

1. L'identità di A e che il messaggio è stato generato da A;
2. Che il messaggio era destinato a B;
3. L'integrità e l'originalità del messaggio (non è stato inviato più volte).

::: IdM – 5/7

X.509 include anche tre procedure alternative di autenticazione destinate all'uso in una varietà di applicazioni.



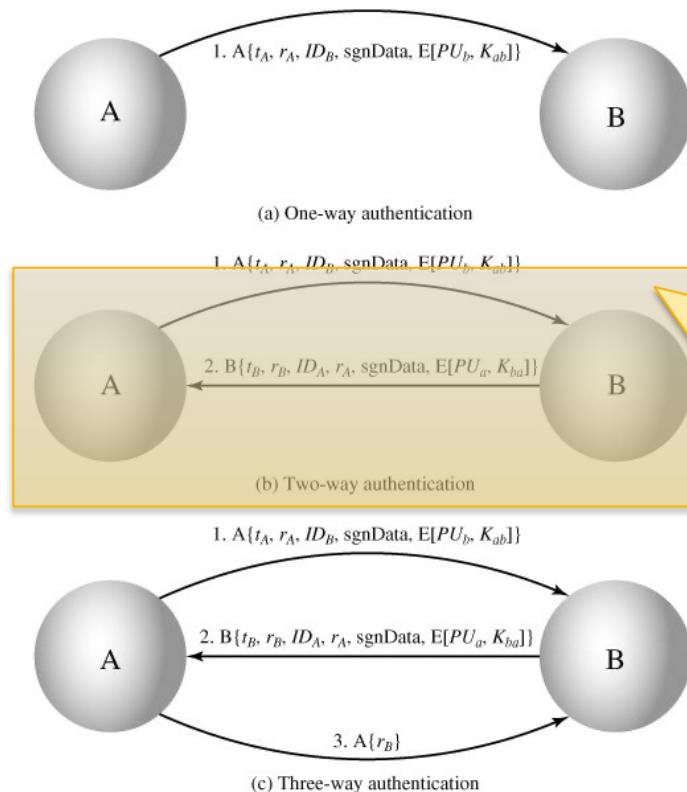
Le due parti devono conoscere la chiave pubblica dell'altra, sia
che prendono i certificati dalla
direttamente perché il certificato

Il messaggio include un
il timestamp t_A , un nonce r_A e
l'identità di B, ed è firmato con
la chiave privata di A.



::: IdM – 5/7

X.509 include anche tre procedure destinate all'uso in una val-

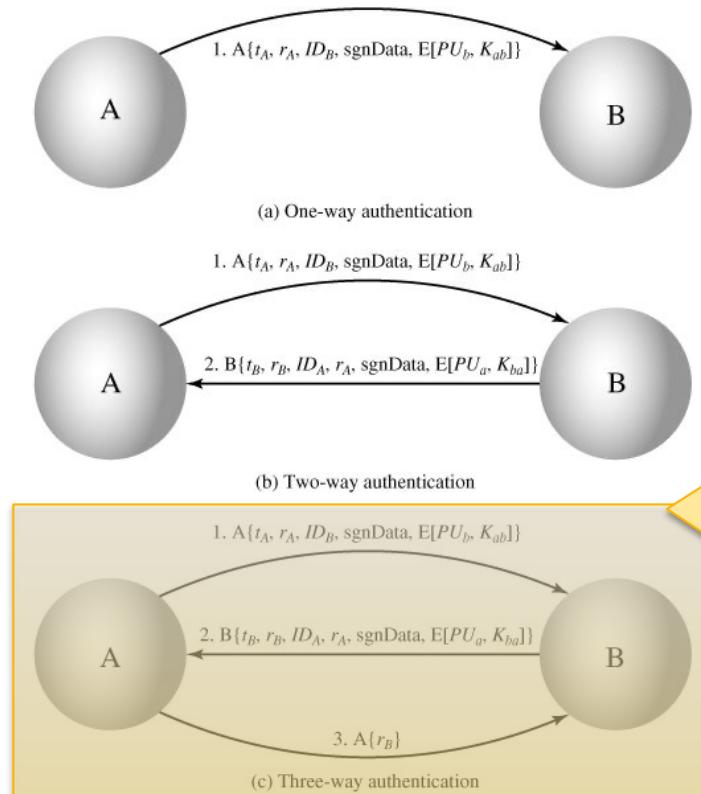


1. L'identità di B e che il messaggio di risposta è stato generato da B;
2. Che il messaggio era destinato ad A;
3. L'integrità e l'originalità della risposta.

L'autenticazione bidirezionale consente quindi ad entrambe le parti in comunicazione di verificare l'identità dell'altra.

::: IdM – 5/7

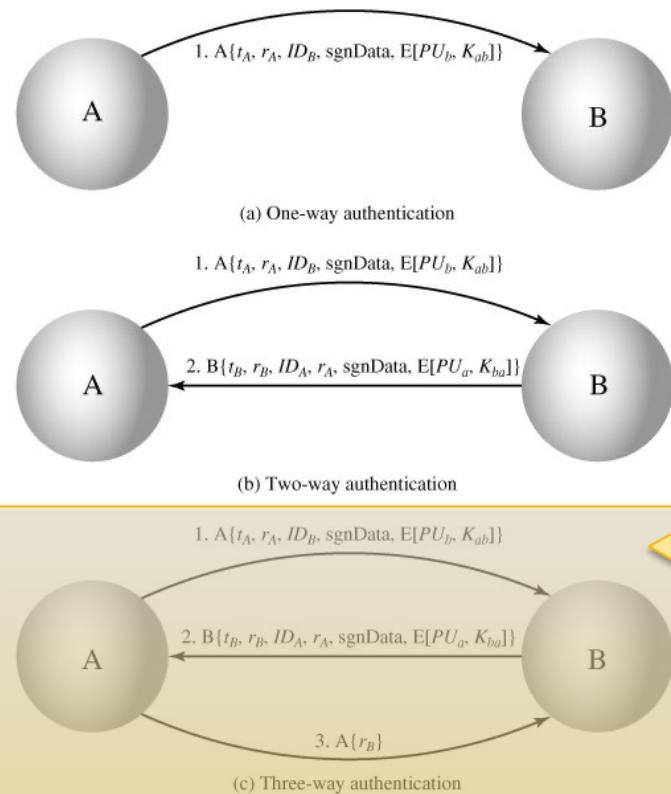
X.509 include anche tre procedure di autenticazione destinate all'uso in una valutazione.



Nell'autenticazione a tre vie, viene inclusa una messaggio finale da A a B, che contiene una copia firmata del nonce r_B . Poiché entrambi i nonces vengono rispiegati dalla controparte, ciascuna parte può verificare il nonce restituito per rilevare gli attacchi di ripetizione. Questo approccio è necessario quando gli orologi sincronizzati non sono disponibili.

::: IdM – 5/7

X.509 also includes three alternative authentication procedures that are intended for use across a variety of applications.



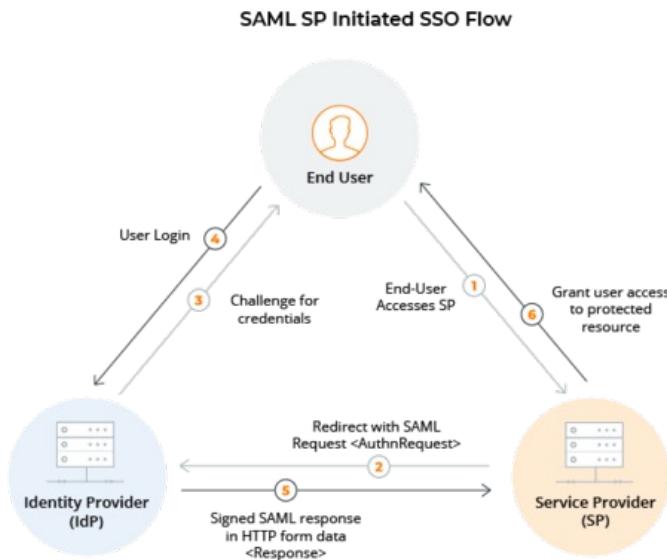
The public key used by one side is published by the other side, because it is initialized by the other side.

In three-way authentication, a final message from A to B is included, which contains a signed copy of the other side's nonce r_B . Because both nonces are echoed back by the other side, each side can check the returned nonce to detect replay attacks. This approach is needed when synchronized clocks are not available.



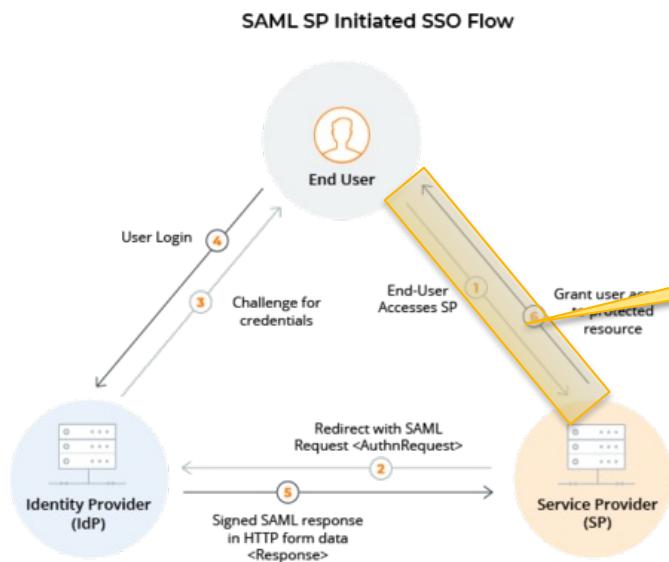
::: IdM – 6/7

Security Assertion Markup Language (SAML) è uno standard aperto per lo scambio di dati di autenticazione e autorizzazione tra le parti, in particolare tra un fornitore di identità e un fornitore di servizi, ed è un linguaggio di markup basato su XML per le asserzioni di sicurezza.



::: IdM – 6/7

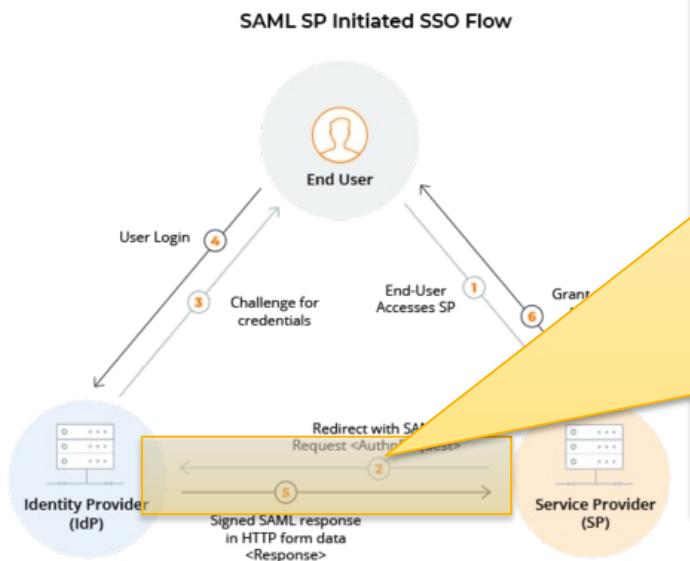
Security Assertion Markup Language (SAML) è uno standard aperto per lo scambio di dati di autenticazione e autorizzazione tra le parti, in particolare tra un fornitore di identità e un fornitore di servizi, ed è un linguaggio di markup basato su XML per le asserzioni di sicurezza.



L'utente finale avvia il processo di accesso presso i Service Provider (SP).

::: IdM – 6/7

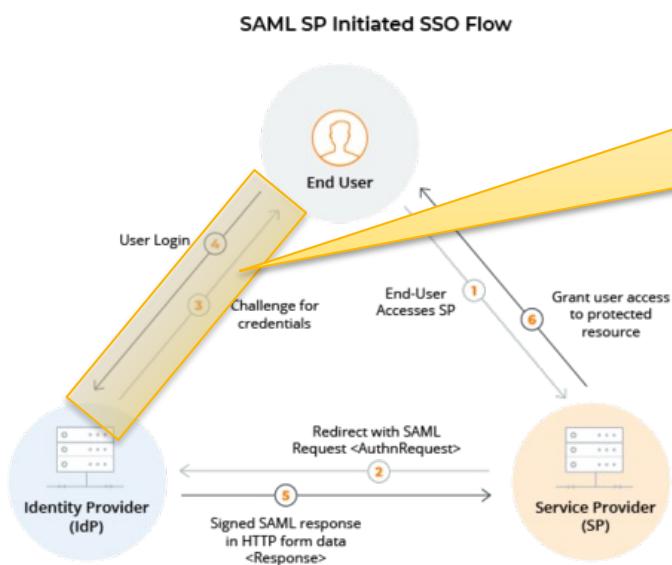
Security Assertion Markup Language (SAML) è uno standard aperto per lo scambio di dati di autenticazione e autorizzazione tra le parti, in particolare ~~tra un fornitore di identità e un fornitore di servizi, ed XML per le asserzioni di~~



Il SP reindirizzerà l'utente all'Identity Provider (IdP) con una richiesta SAML (AuthnRequest). La richiesta SAML conterrà le informazioni necessarie affinché l'IdP autentichi l'utente finale e risponda allo SP con la corretta asserzione SAML (SAMLResponse).

::: IdM – 6/7

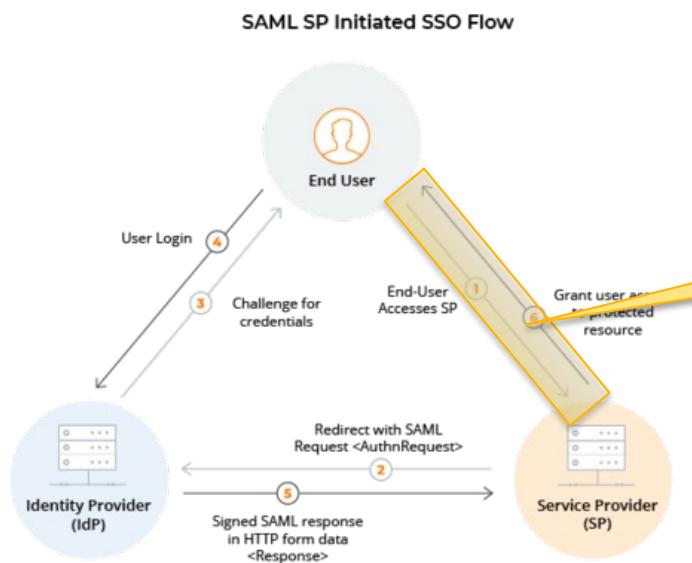
Security Assertion Markup Language (SAML) è uno standard aperto per lo scambio di dati di autenticazione e autorizzazione tra le parti, in particolare tra un fornitore di identità e un fornitore di servizi, ed è un linguaggio di markup basato su XML per le asserzioni di sicurezza.



L'Identity Manager (IdM) può sfidare l'utente finale per completare la fase di accesso.

::: IdM – 6/7

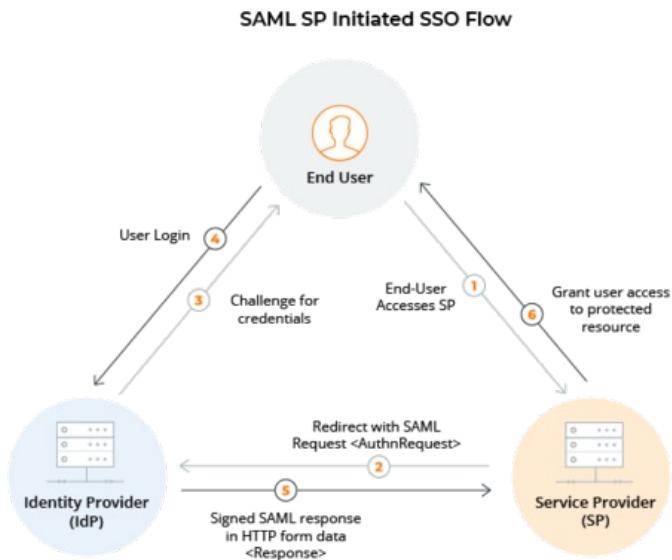
Security Assertion Markup Language (SAML) è uno standard aperto per lo scambio di dati di autenticazione e autorizzazione tra le parti, in particolare tra un fornitore di identità e un fornitore di servizi, ed è un linguaggio di markup basato su XML per le asserzioni di sicurezza.



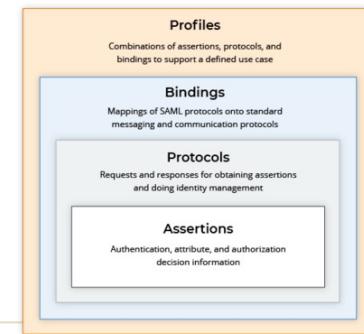
Alla fine, il SP può concedere o negare l'accesso all'utente finale.

::: IdM – 6/7

Security Assertion Markup Language (SAML) è uno standard aperto per lo scambio di dati di autenticazione e autorizzazione tra le parti, in particolare tra un fornitore di identità e un fornitore di servizi, ed è un linguaggio di markup basato su XML per le asserzioni di sicurezza.



Gli standard di SAML forniscono una richiesta/risposta per lo scambio di messaggi XML tra questi ruoli.



::: IdM – 7/7

Nonostante offrano buoni livelli di sicurezza, le soluzioni basate su certificati possono rivelarsi molto costose e ingombranti.

La gestione delle identità (IdM) può avere tre possibili architetture:

1. Isolata: il fornitore di servizi agisce anche come IdM;
2. Centralizzata: è presente un singolo IdM per gestire le identità di un insieme di fornitori di servizi all'interno di un determinato dominio;
3. Federata: diversi IdM vengono utilizzati all'interno di un dominio di trust federato in modo che le identità provenienti da diversi domini (vale a dire, emesse da diversi IdM) siano riconosciute su tutti i domini per realizzare il Single-Sign-On (SSO).

L'ultima soluzione è più scalabile ed efficiente, ma più complessa da realizzare. In questo contesto, entrano in gioco le blockchain.

Blockchain Implementation



::: IdM – Blockchain Implementation

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/access/Ownable.sol";

contract IdentityManagement is ERC721, Ownable {
    struct Identity {
        string name;
        uint256 age;
        string email;
        string phoneNumber;
    }
    mapping(uint256 => Identity) private identities;
    uint256 private tokenIdCounter;
```



:... IdM – Blockchain Implementation

```
constructor() ERC721("IdentityManagement", "IDM") {}
```

```
function createIdentity(
    string memory _name,
    uint256 _age,
    string memory _email,
    string memory _phoneNumber
) public onlyOwner {
    uint256 tokenId = tokenIdCounter;
    tokenIdCounter++;
    Identity storage newIdentity = identities[tokenId];
    newIdentity.name = _name;
    newIdentity.age = _age;
    newIdentity.email = _email;
    newIdentity.phoneNumber = _phoneNumber;
    _safeMint(msg.sender, tokenId);
}
```

::: IdM – Blockchain Implementation

```
function getIdentity(uint256 _tokenId) public view
    returns (
        string memory,
        uint256,
        string memory,
        string memory
    )
{
    require(_exists(_tokenId), "Identity does not exist");
    Identity storage identity = identities[_tokenId];
    return (
        identity.name,
        identity.age,
        identity.email,
        identity.phoneNumber
    );
}
```

::: IdM – Blockchain Implementation

```
function updateIdentity(  
    uint256 _tokenId,  
    string memory _name,  
    uint256 _age,  
    string memory _email,  
    string memory _phoneNumber  
) public onlyOwner {  
    require(_exists(_tokenId), "Identity does not exist");  
  
    Identity storage identity = identities[_tokenId];  
    identity.name = _name;  
    identity.age = _age;  
    identity.email = _email;  
    identity.phoneNumber = _phoneNumber;  
}  
}
```

Self-Sovereign Identity (SSI)



::: Decentralized Identity

Centralized Identity

Register once, trusted by one

Service provider establishes and maintains users' identity and related data in its system



Federated Identity

Register once, trusted by many

Service provider trusts identities established and maintained by other identity providers



Decentralized Identity

Register once, trusted globally

Users have a self-managed digital identity independent of individual service providers



La gestione dell'identità è evoluta da un modello centralizzato, ad un modello federato (SAML, OIDC), fino a giungere ad un modello decentralizzato.

::: Decentralized Identity

Centralized Identity

Register once, trusted by one

Service provider establishes and maintains users' identity and related data in its system



Federated Identity

Register once, trusted by many

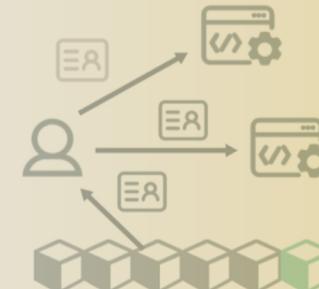
Service provider trusts identities established and maintained by other identity providers



Decentralized Identity

Register once, trusted globally

Users have a self-managed digital identity independent of individual service providers



Nei modelli decentralizzati non è più presente un'autorità per la verifica delle credenziali, ma piuttosto sono presenti dei registri pubblici condivisi dove vengono memorizzate informazioni pubbliche relative all'identità.

... FI vs. SSI



::: Verifiable Credentials (VCs)

Il modello decentralizzato si basa sul concetto di Verifiable Credential (VC), che è utilizzato per verificare l'identità degli utenti.

Nel mondo fisico, una credenziale potrebbe consistere in: informazioni relative all'identificazione del soggetto della credenziale; informazioni relative all'autorità che rilascia la credenziale; informazioni relative ai vincoli sulla credenziale; ecc.

Una credenziale verificabile può rappresentare tutte le stesse informazioni rappresentate da una credenziale fisica. L'aggiunta di tecnologie, come le firme digitali, rende le credenziali verificabili più evidenti di manipolazioni e più affidabili rispetto ai loro equivalenti fisici.



::: Verifiable Credentials (VCs)

Il modello decentralizzato si basa sul concetto di Verifiable Credential (VC), che è utilizzato per verificare l'identità degli utenti.

La VC mette l'utente al centro del sistema, creando così la user-centric authentication.

```
{
  // set the context, which establishes the special terms we will be using
  // such as 'issuer' and 'alumniOf'.
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://www.w3.org/ns/credentials/examples/v2"
  ],
  // specify the identifier for the credential
  "id": "http://university.example/credentials/1872",
  // the credential types, which declare what data to expect in the credential
  "type": ["VerifiableCredential", "ExampleAlumniCredential"],
  // the entity that issued the credential
  "issuer": "https://university.example/issuers/565049",
  // when the credential was issued
  "validFrom": "2010-01-01T19:23:24Z",
  // claims about the subjects of the credential
  "credentialSubject": {
    // identifier for the only subject of the credential
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    // assertion about the only subject of the credential
    "alumniOf": {
      // identifier for the university
      "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
      // name of the university
      "name": "Example University"
    }
  }
}
```



::: Verifiable Credentials (VCs)

Il modello decentralizzato si basa sul concetto di Verifiable Credential (VC), che è utilizzato per verificare l'identità degli utenti.

La VC mette l'utente al centro del sistema, creando così la

Una VC può essere costituita da uno o più attributi che vanno a costituire le "claims", ovvero delle relazioni subject-property-value

```
{
  // set the context, which establishes the special terms we will be using
  // such as 'issuer' and 'alumniOf'.
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://www.w3.org/ns/credentials/examples/v2"
  ],
  // specify the identifier for the credential
  "id": "http://university.example/credentials/1872",
  // the credential types, which declare what data to expect in the credential
  "type": ["VerifiableCredential", "ExampleAlumniCredential"],
  // the entity that issued the credential
  "issuer": "https://university.example/issuers/565049",
  // when the credential was issued
  "validFrom": "2010-01-01T19:23:24Z",
  // claims about the subjects of the credential
  "credentialSubject": {
    // identifier for the only subject of the credential
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    // assertion about the only subject of the credential
    "alumniOf": {
      // identifier for the university
      "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
      // name of the university
      "name": "Example University"
    }
  }
}
```



::: Verifiable Credentials (VCs)

Il modello decentralizzato si basa sul concetto di Verifiable Credential (VC), che è utilizzato per verificare l'identità degli utenti.

La VC mette l'utente al centro

Vengono inoltre definiti attributi relativi a chi ha rilasciato le credenziali, sul tipo di credenziali e sulla validità temporale.

```
{
  // set the context, which establishes the special terms we will be using
  // such as 'issuer' and 'alumniOf'.
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://www.w3.org/ns/credentials/examples/v2"
  ],
  // specify the identifier for the credential
  "id": "http://university.example/credentials/1872",
  // the credential types, which declare what data to expect in the credential
  "type": ["VerifiableCredential", "ExampleAlumniCredential"],
  // the entity that issued the credential
  "issuer": "https://university.example/issuers/565049",
  // when the credential was issued
  "validFrom": "2010-01-01T19:23:24Z",
  // claims about the subjects of the credential
  "credentialSubject": {
    // identifier for the only subject of the credential
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    // assertion about the only subject of the credential
    "alumniOf": {
      // identifier for the university
      "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
      // name of the university
      "name": "Example University"
    }
  }
}
```



::: Verifiable Credentials (VCs)

Il modello decentralizzato si basa sul concetto di Verifiable Credential (VC), che è utilizzato per verificare l'identità degli utenti.

La VC mette l'utente al centro del sistema, creando così la user-centric authentication.

L'identità del proprietario delle credenziali viene definita attraverso l'uso di Decentralized Identifiers (DIDs)

```
{
  // set the context, which establishes the special terms we will be using
  // such as 'issuer' and 'alumniOf'.
  "@context": [
    "https://www.w3.org/ns/credentials/v2",
    "https://www.w3.org/ns/credentials/examples/v2"
  ],
  // specify the identifier for the credential
  "id": "http://university.example/credentials/1872",
  // the credential types, which declare what data to expect in the credential
  "type": ["VerifiableCredential", "ExampleAlumniCredential"],
  // the entity that issued the credential
  "issuer": "https://university.example/issuers/565049",
  // when the credential was issued
  "validFrom": "2010-01-01T19:23:24Z",
  // claims about the subjects of the credential
  "credentialSubject": {
    // identifier for the only subject of the credential
    "id": "did:example:ebfeb1f712ebc6f1c276e12ec21",
    // assertion about the only subject of the credential
    "alumniOf": {
      // identifier for the university
      "id": "did:example:c276e12ec21ebfeb1f712ebc6f1",
      // name of the university
      "name": "Example University"
    }
  }
}
```



::: Decentralized Identifiers (DIDs)

DID (Decentralized Identifier): un DID è un identificatore composto da caratteri alfanumerici, è unico e attraverso la risoluzione è possibile consultare il DIDDocument.



::: Decentralized Identifiers (DIDs)

DID (Decentralized Identifier): un DID è un identificatore composto da caratteri alfanumerici, è unico e attraverso la risoluzione è possibile consultare il DIDDocument.



La sintassi formale di un DID prevede che la stringa inizi con il prefisso DID, esso identifica lo schema.



::: Decentralized Identifiers (DIDs)

DID (Decentralized Identifier): un DID è un identificatore composto da caratteri alfanumerici, è unico e attraverso la risoluzione è possibile consultare il DID Document.



Il DID Method definisce come deve essere effettuata la risoluzione del DID. Ogni tipo di DID indica le relative regole di creazione/passaggio del documento DID.



::: Decentralized Identifiers (DIDs)

DID (Decentralized Identifier): un DID è un identificatore composto da caratteri alfanumerici, è unico e attraverso la risoluzione è possibile consultare il DID Document.



Il DID Method Specific è l'estensione che caratterizza il DID per la specifica risorsa.



::: DID Document

Ogni indirizzo DID si risolve in DID Document che è un documento JSON contenente varie informazioni per interagire con il proprietario del DID.

Un DID Document include chiavi pubbliche crittografiche, service endpoint, parametri di autenticazione, timestamp e altri metadati

```
did:example:123456789abcdefghi
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:example:123456789abcdefghi",
  "publicKey": [
    {
      "id": "did:example:123456789abcdefghi#keys-1",
      "type": "RsaVerificationKey2018",
      "owner": "did:example:123456789abcdefghi",
      "publicKeyPem": "Public key goes here.."
    },
    {
      "authentication": [
        // this key can be used to authenticate as DID ...9938
        "type": "RsaSignatureAuthentication2018",
        "publicKey": "did:example:123456789abcdefghi#keys-1"
      ],
      "service": [
        {
          "type": "ExampleService",
          "serviceEndpoint": "https://example.com/endpoint/8377464"
        }
      ]
    }
}
```

The standard elements of a DID doc

1. **DID** (for self-description)
2. **Set of public keys** (for verification)
3. **Set of auth methods** (for authentication)
4. **Set of service endpoints** (for interaction)
5. **Timestamp** (for audit history)
6. **Signature** (for integrity)

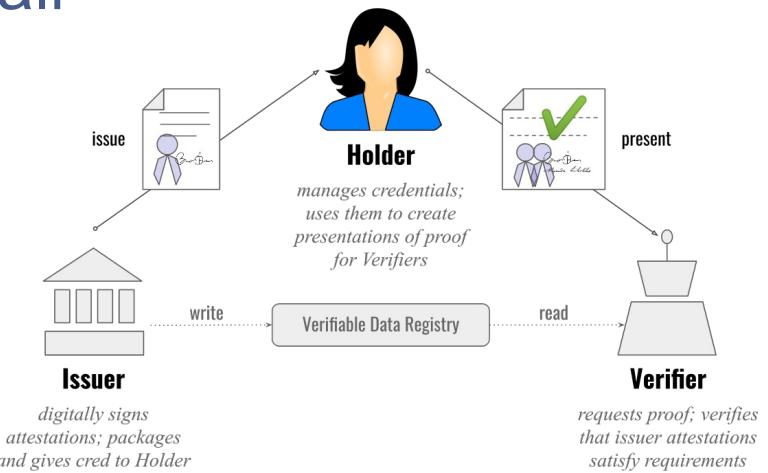


::: VCs Data Model

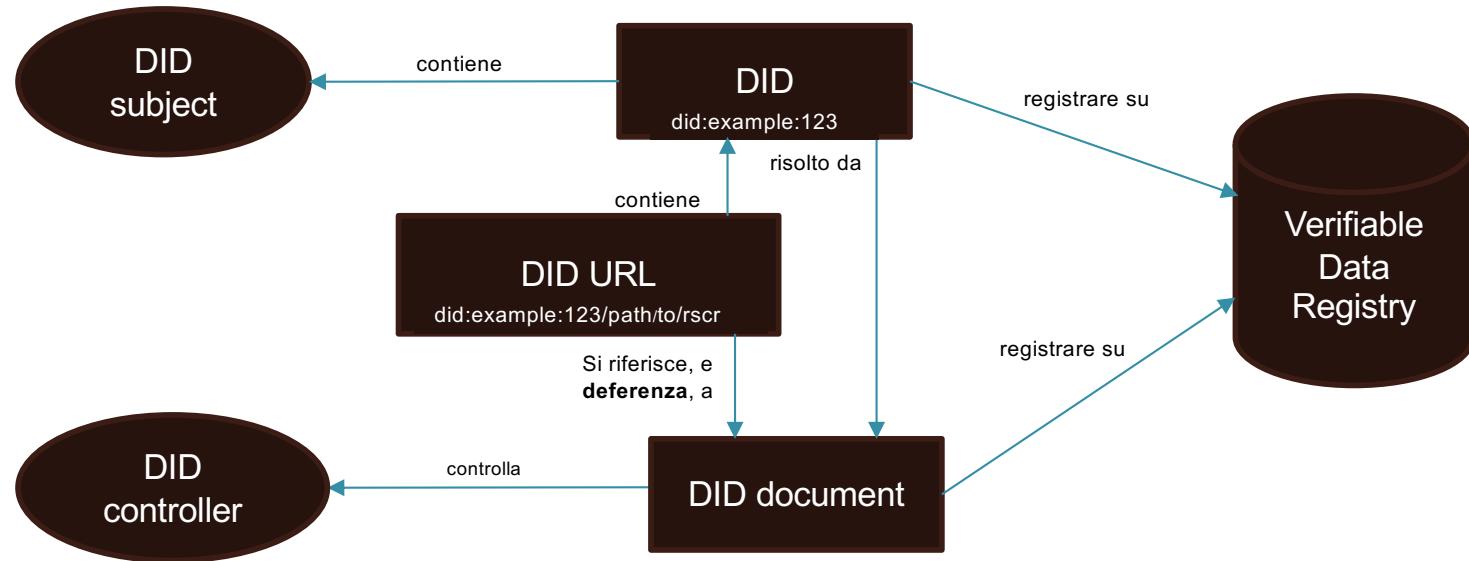
Per far funzionare tutto l'ecosistema delle VCs è necessario chiarire tre ruoli fondamentali:

- **Issuer**: chi rilascia le credenziali
- **Holder**: chi possiede le credenziali
- **Verifier**: chi verifica le credenziali

Attraverso l'uso di un Verifiable Data Registry è possibile effettuare operazioni di controllo e gestione delle chiavi.



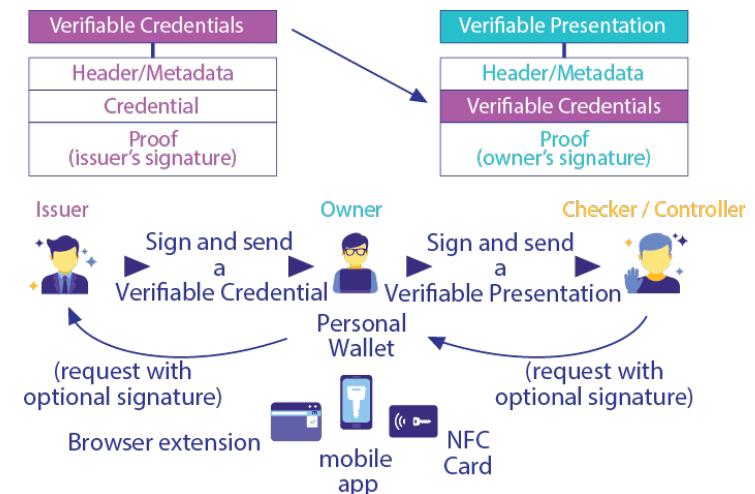
::: VCs Data Model



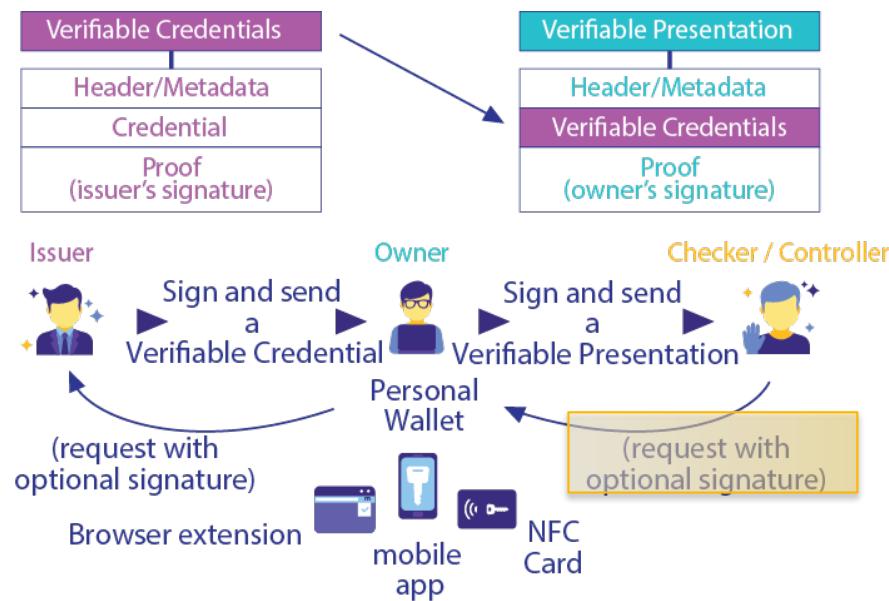
::: Verifiable Presentation (VP)

Dopo il rilascio delle credenziali, l'utente è in grado di utilizzare tali credenziali per autenticarsi sui servizi che ne permettono l'utilizzo.

In questo caso, le credenziali saranno inviate attraverso una Verifiable Presentation, che è necessaria per verificare la proprietà sulla credenziale.

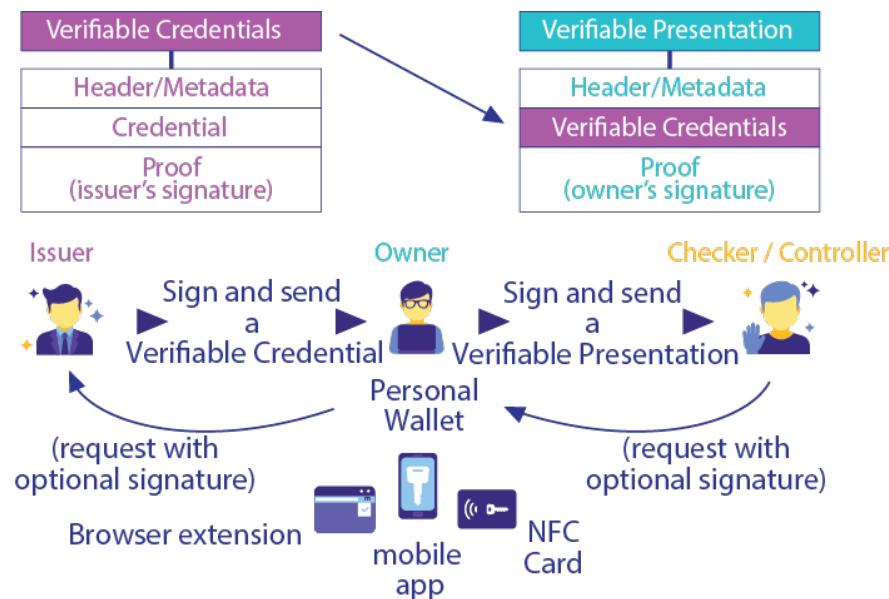


::: Verifiable Presentation (VP)



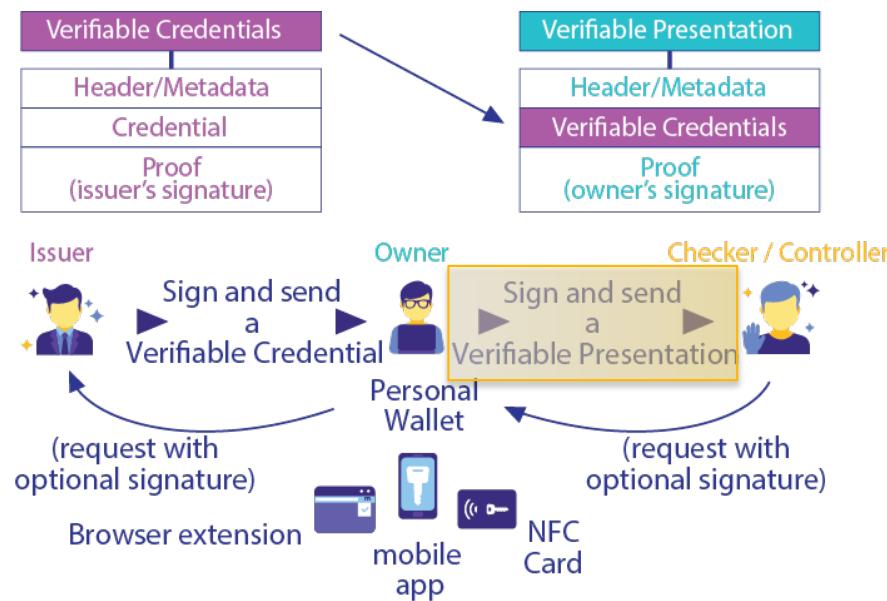
1. Il Controller (Verifier) invia una richiesta di verifiable presentation, che contiene gli attributi che devono essere rilasciati dall'Owner (Holder) per la verifica dell'identità

::: Verifiable Presentation (VP)



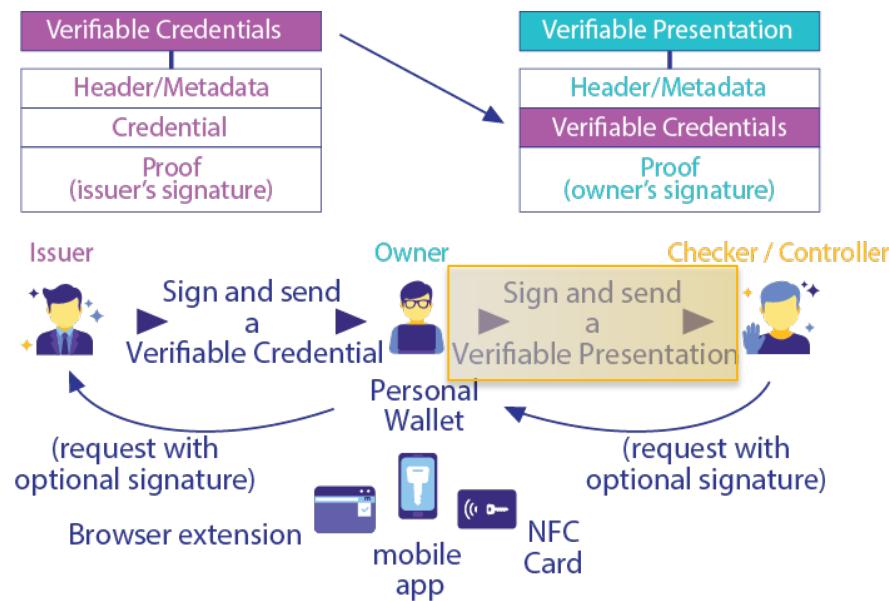
2. L'Holder controlla la richiesta, andando a vedere se gli attributi richiesti sono in linea con le sue aspettative, e un framework cercherà all'interno del wallet l'attributo richiesto.

::: Verifiable Presentation (VP)



3. Se l'Holder possiede l'attributo e vuole condividerlo produce una Verifiable Presentation contenente la VC in questione e la sua firma in modo che il Controller ne possa verificare la proprietà.

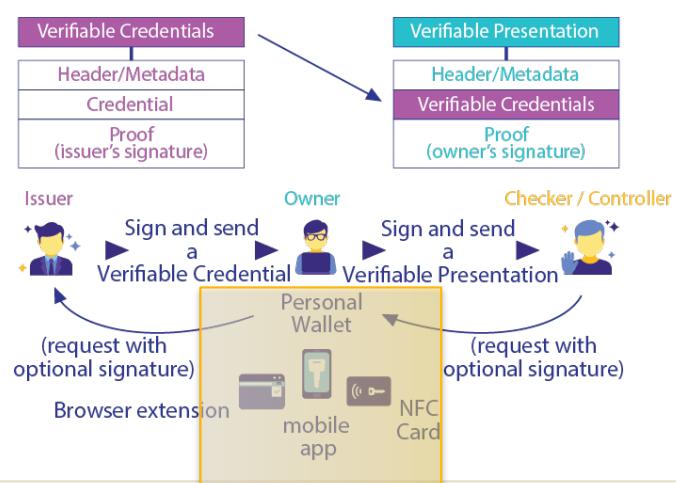
::: Verifiable Presentation (VP)



4. Il Verifier consulta il verify registry per assicurarsi che le firme sono vere, e che le credenziali non siano state revocate.

::: Digital Wallet

Tipicamente le credenziali rilasciate vengono conservate in wallet hardware che permettono di avere un elevato livello di sicurezza e di resistere ai possibili attacchi (mobile app, NFC Card). Attualmente, diverse soluzioni commerciali sono già esistenti, e quasi tutte si basano sull'utilizzo di una blockchain.



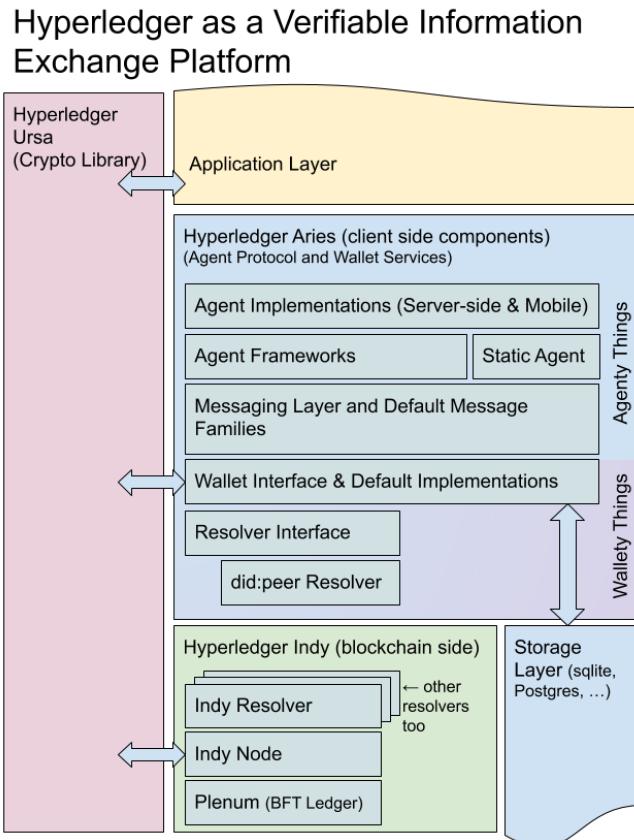
::: Hyperledger Aries

Hyperledger Aries fornisce una serie di strumenti modulare per la creazione di infrastrutture di identità decentralizzate (DID) e sistemi di attestazione. L'obiettivo principale di Hyperledger Aries è quello di facilitare lo scambio sicuro di informazioni di identità tra parti diverse senza la necessità di un'autorità centralizzata.

Questo HL offre tutte le componenti necessarie al funzionamento di SSI tra cui Wallet, protocollo di scambio di credenziali, gestione del DID, gestione delle chiavi.



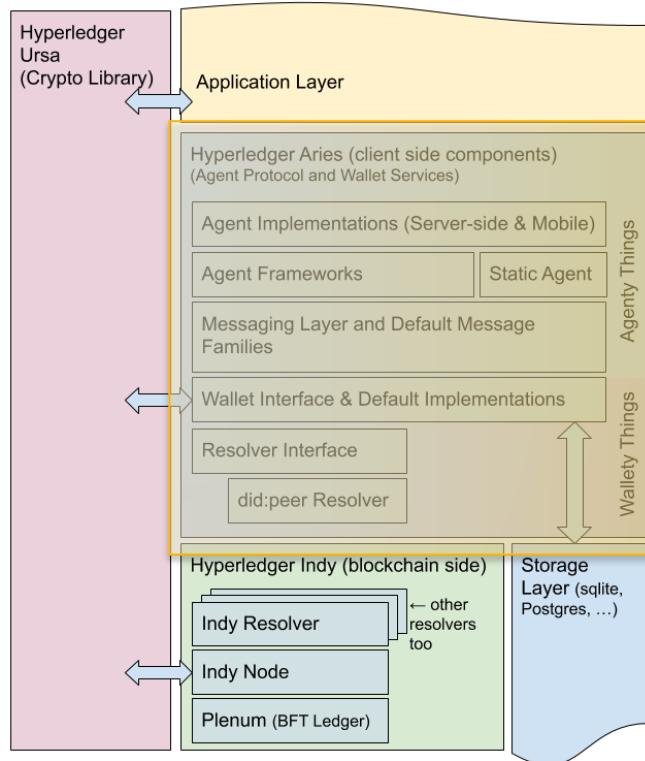
::: Hyperledger Aries



Hyperledger Aries si basa su Hyperledger Indy e Hyperledger Ursa per garantire una infrastruttura completa per la gestione di tutto il processo di rilascio e verifica delle credenziali.

::: Hyperledger Aries

Hyperledger as a Verifiable Information Exchange Platform

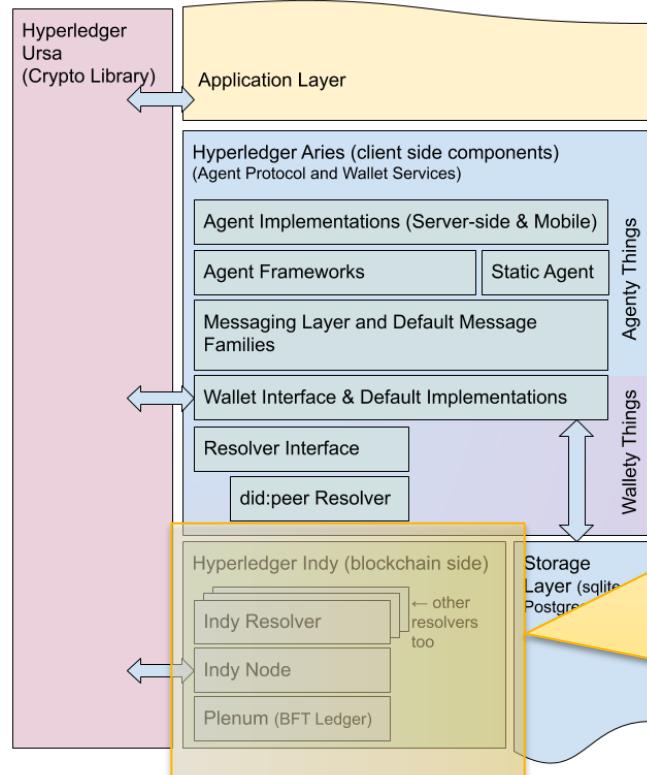


Aries si occupa principalmente della comunicazione con gli altri due HL attraverso interfacce web, agenti e risoluzione di DID



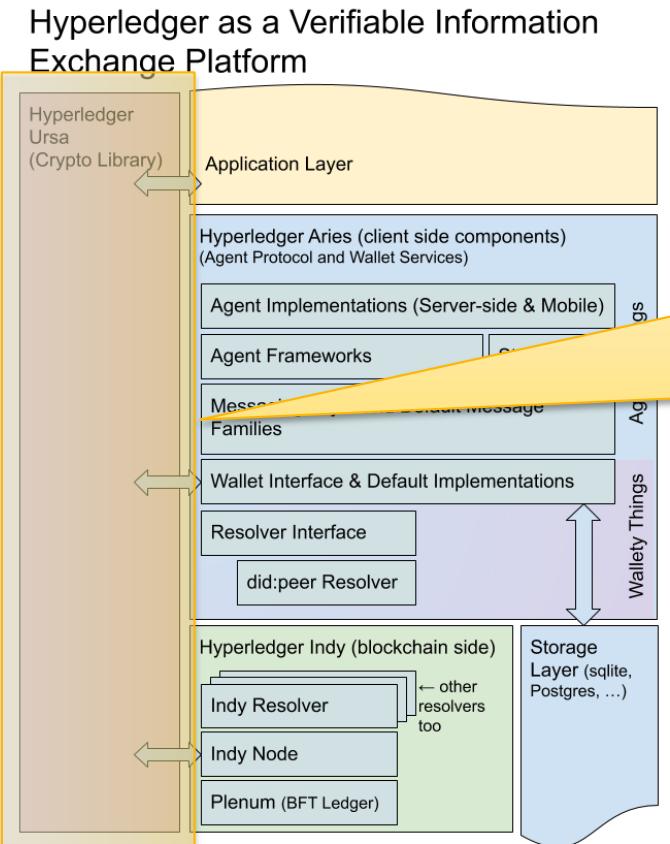
::: Hyperledger Aries

Hyperledger as a Verifiable Information Exchange Platform



Indy si occupa della gestione dei Wallet e del consenso nella rete. Esso offre anche una gestione del distributed ledger su cui vengono registrate le operazioni

::: Hyperledger Aries



Ursa invece si occupa della crittografia alla base della SSI. Offre librerie crittografiche per implementare in modo efficiente il protocollo.

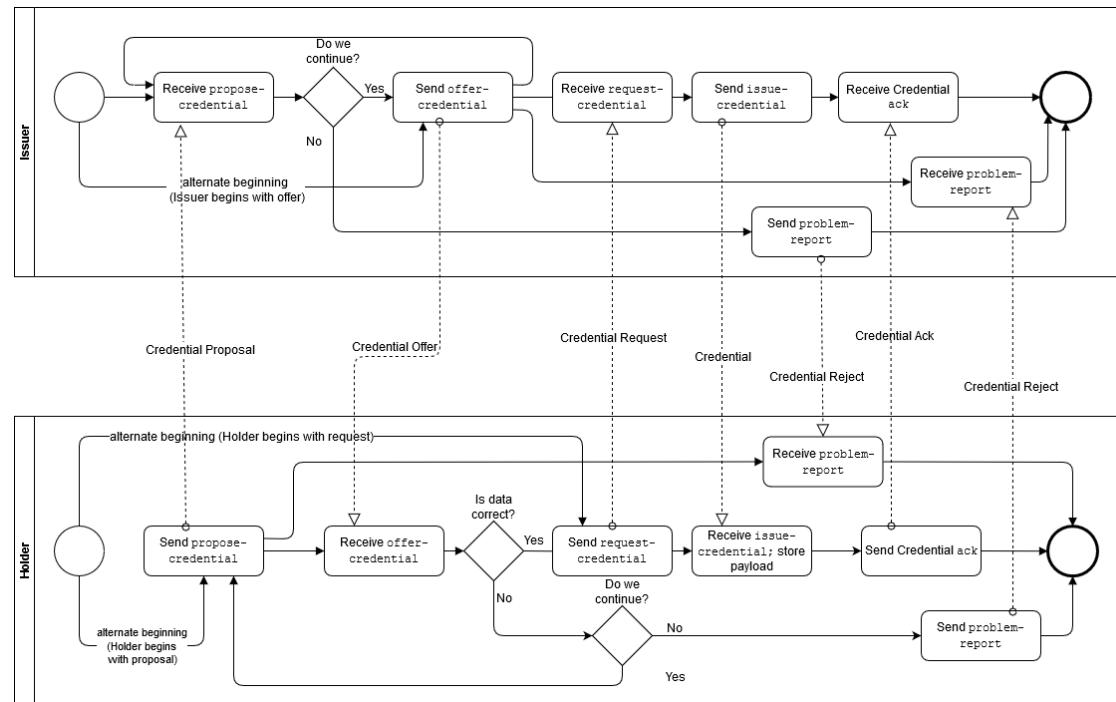
Autenticazione Aries RFC 0023: DID Exchange Protocol 2.0

Requester State Machine							
↓ States	Events →	receive explicit or find implicit invitation	send request	receive response	send complete	receive problem-report	send problem-report
start		transition to "invitation-received"	impossible	impossible	impossible	impossible	impossible
invitation-received		impossible	transition to "request-sent"	impossible	impossible	impossible	transition to "start" or "abandoned"
request-sent		impossible	impossible	transition to "response-received"	impossible	transition to "invitation-received" or "abandoned"	impossible
response-received		impossible	impossible	impossible	transition to "completed"	impossible	transition to "request-sent" or "abandoned"
abandoned		impossible	impossible	impossible	impossible	impossible	impossible
completed							

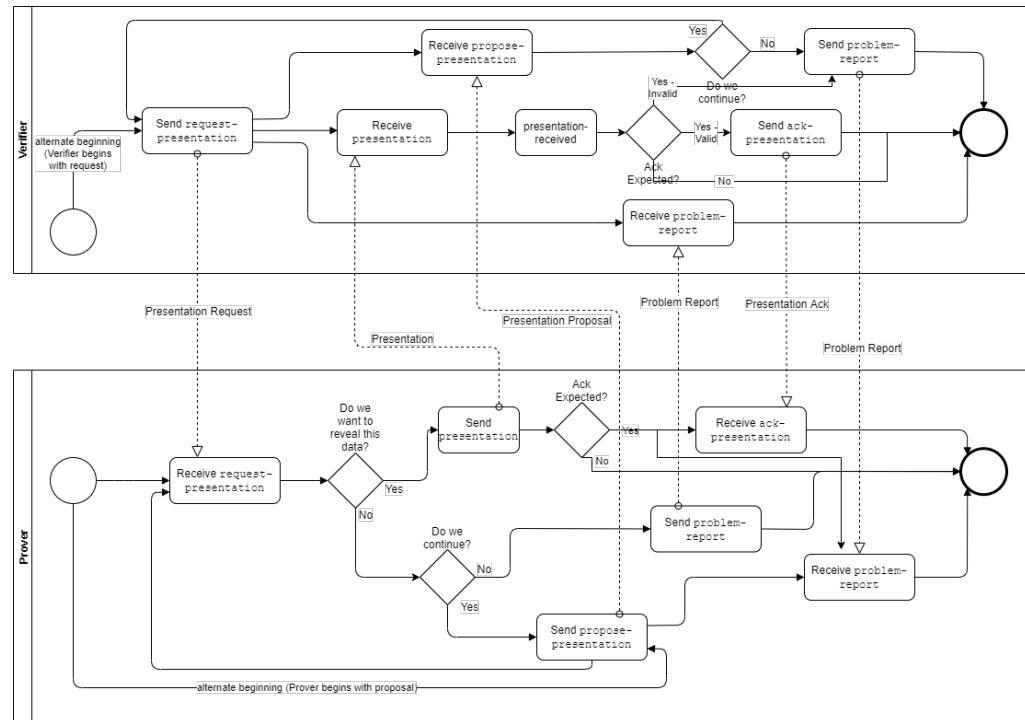
Responder State Machine							
↓ States	Events →	send explicit or publish implicit invitation	receive request	send response	receive complete	receive problem-report	send problem-report
start		transition to "invitation-sent"	impossible	impossible	impossible	impossible	impossible
invitation-sent		impossible	transition to "request-received"	impossible	impossible	transition to "start" or "abandoned"	impossible
request-received		impossible	impossible	transition to "response-sent"	impossible	impossible	transition to "invitation-sent" or "abandoned"
response-sent		impossible	impossible	impossible	transition to "completed"	transition to "request-received" or "abandoned"	impossible
abandoned		impossible	impossible	impossible	impossible	impossible	impossible
completed							



Autenticazione Aries RFC 0023: DID Exchange Protocol 2.0



... Aries RFC 0454: Present Proof Protocol 2.0



::: Zero-Knowledge Proof (ZKP)

HL Aries offre la possibilità di implementare SSI, ma possiamo fare di meglio.

Ogni volta che il server chiede informazioni sull'identità viene generata una VP che viene condivisa dall'Holder. Questa VP espone gli attributi dell'utente. Attraverso ZKP è possibile anonimizzare gli attributi andando a rispondere a specifiche «domande».

Aries supporta ZKP attraverso un nuovo Hyperledger.

Self-Sovereign Identity (SSI) Attribute-Based Web Authentication

Biagio Boi^a, Marco De Santis^b and Christian Esposito^{b,c}
University of Salerno, Fisciano, Italy

Keywords: Self-Sovereign Identity, Hyperledger Aries, Attribute-Based Authentication.

Abstract: Web authentication is primarily based on password usage, representing the weakest link in the entire security chain. The number of services offered over the web is continuously increasing, and with them also the number of required passwords that users need to create and securely store. Despite various standards for password-less or multi-factor authentication, another issue is that most web authentication means use an identity provider (or a federation of providers) advocated to create, manage and check digital identity claims; able to profile user habits related to web navigation and violate rights in terms of privacy. Recently, we are witnessing a radical change of perspective, where identity checks and enforcement are moved away from the providers and more focused on users. Within such user-centric approaches, Self-Sovereign Identity (SSI) has faced progressive popularity, and some authentication mechanisms based on SSI have been proposed. This paper aims to describe a solution based on Hyperledger Aries which is capable to achieve zero-knowledge proof to make an attribute-based authentication and authorization for the web able to cope with the recent legal obligations in terms of privacy.



::: Hyperledger AnonCreds

Sono un formato di credenziali verificabili (VC) che aggiungono importanti capacità di protezione della privacy basate su zero-knowledge proof (ZKP) alle garanzie di base dei VC.

Hl AnonCreds è il formato di VC più comunemente utilizzato al mondo e ha una specifica aperta formale. È agnostico rispetto al registro e può essere utilizzato in qualsiasi registro decentralizzato.

::: AnonCreds

Le credenziali verificabili ZKP di AnonCreds offrono funzionalità che molti considerano importanti per i casi d'uso dell'identità digitale in particolare e per i dati verificabili in generale. Queste caratteristiche includono:

- Un'implementazione completa del “Trust Triangle” delle credenziali verificabili di livello 3 del modello Trust over IP.
- Flussi completi per l'emissione di credenziali verificabili (dall'emittente al titolare) e per la richiesta, la generazione e la verifica di presentazioni di affermazioni verificabili (dal titolare al verificatore).
- Modelli di dati completamente definiti per tutti gli oggetti dei flussi, comprese le credenziali verificabili, le richieste di presentazione e le presentazioni provenienti da credenziali multiple.
- Applicazioni completamente definite di primitive crittografiche.



::: AnonCreds

Le credenziali verificabili ZKP di AnonCreds offrono funzionalità che molti considerano importanti per i casi d'uso dell'identità digitale in particolare e per i dati verificabili in generale. Queste caratteristiche includono. L'uso di uno schema di firma cieca nel processo di emissione per migliorare le protezioni della privacy disponibili per il titolare che riceve la credenziale, tra cui:

- Solo il titolare conosce la firma finale della credenziale, mentre l'emittente conosce solo una versione cieca che solo il titolare può sbloccare.
- L'emittente può firmare i messaggi alla cieca, non venendo mai a conoscenza del loro valore, pur essendo in grado di garantire che il titolare conosca il valore del messaggio firmato. Questo viene utilizzato per il segreto di collegamento, che il titolare si impegna a rispettare durante l'emissione; l'emittente firma alla cieca questo messaggio insieme al resto degli attributi della credenziale e poi il titolare rimuove l'oscuramento per ottenere una firma sugli attributi della credenziale, compreso il valore del segreto di collegamento non oscurato.



::: AnonCreds

Le credenziali verificabili ZKP di AnonCreds offrono funzionalità che molti considerano importanti per i casi d'uso dell'identità digitale in particolare e per i dati verificabili in generale. Queste caratteristiche includono:

- L'uso di Zero Knowledge Proofs (ZKP) nel processo di presentazione verificabile per migliorare le protezioni della privacy a disposizione del titolare nella presentazione dei dati ai verificatori, tra cui:
 - Dimostrare che alcuni valori fanno parte di una credenziale firmata senza rivelarli, tra cui: la firma della credenziale, il segreto del collegamento e qualsiasi attributo che il titolare non desidera rivelare. Questo aiuta a prevenire la correlazione basata su questi valori.
 - L'uso di identificatori non rivelati per il binding del titolare per impedire la correlazione basata su tali identificatori.
 - L'uso di prove di predicato per ridurre la condivisione di PII e di dati potenzialmente correlati, in particolare le date (nascita, emissione/scadenza della credenziale, ecc.).
 - Uno schema di revoca che dimostri che una presentazione è basata su credenziali che non sono state revocate dagli emittenti senza rivelare identificatori di revoca correlabili.



::: Vantaggi delle Anoncreds

Protezione della Privacy:

- Le Anoncreds consentono di dimostrare la validità di una dichiarazione senza dover rivelare più informazioni di quelle strettamente necessarie.
- Grazie alle zero-knowledge proofs, l'utente può autenticarsi senza esporsi a rischi di divulgazione non autorizzata di dati personali.

Differenza Rispetto alle Credenziali Convenzionali:

- A differenza delle credenziali tradizionali, che richiedono spesso la divulgazione di molte informazioni personali, le Anoncreds offrono un livello di riservatezza superiore.
- L'utente mantiene il controllo su quali dettagli vengano condivisi, proteggendo la propria identità.



::: Vantaggi delle Anoncreds

Agnostiche Rispetto al Registro:

- La flessibilità delle Anoncreds le rende adatte a essere utilizzate in diversi contesti e registri decentralizzati.
- Possono essere integrate senza problemi in sistemi esistenti, offrendo una soluzione versatile per la gestione delle identità.

Sicurezza Avanzata:

- L'utilizzo delle zero-knowledge proofs garantisce un elevato livello di sicurezza, prevenendo potenziali minacce alla sicurezza delle informazioni durante l'autenticazione.



::: Funzionamento delle Anoncreds

Generazione e Verifica delle Anoncreds:

- Esploriamo il processo di generazione delle Anoncreds, dove vengono create in modo sicuro e privato.
- Successivamente, analizziamo il lato della verifica, dove il destinatario può convalidare la validità della credenziale senza dover conoscere i dettagli sottostanti.

Ruolo delle Zero-Knowledge Proofs:

- Approfondiamo il concetto fondamentale delle zero-knowledge proofs nel contesto delle Anoncreds.
- Vedremo come queste prove a conoscenza zero permettano a chi presenta la credenziale di dimostrare possesso di informazioni senza rivelarle.

Scenari di Utilizzo Pratico:

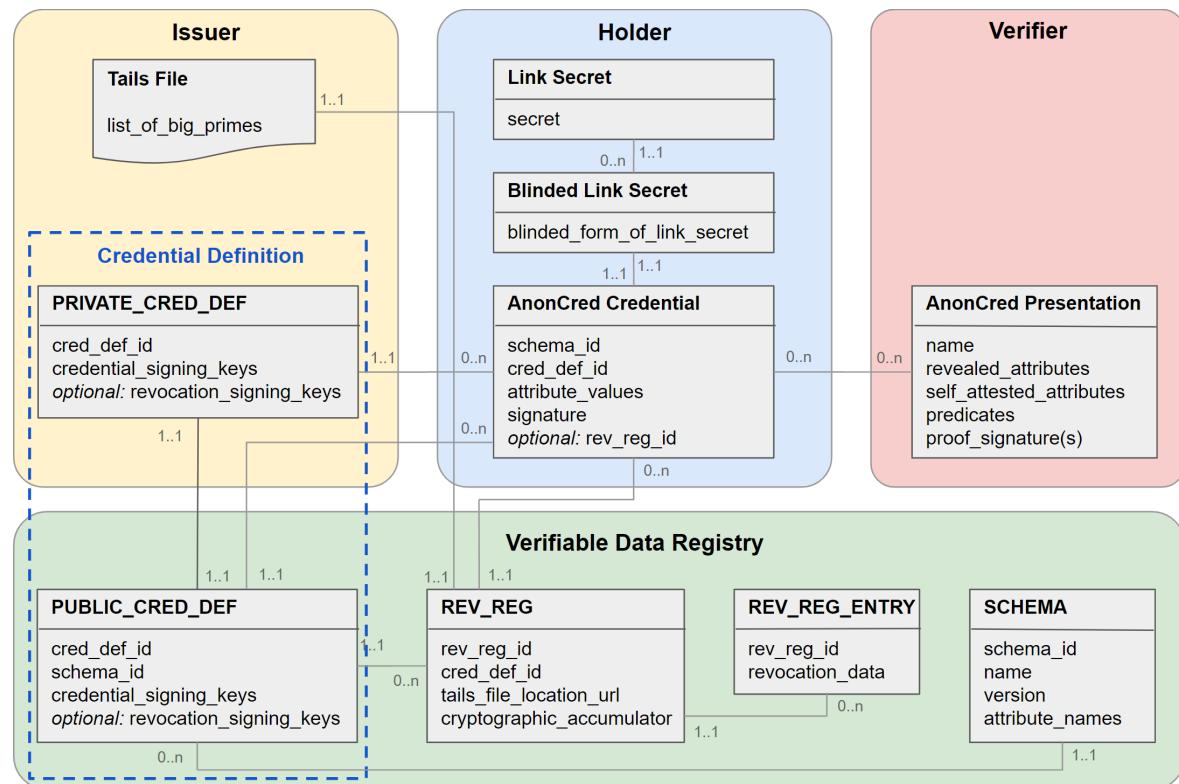
- Esploriamo esempi concreti di come le Anoncreds possono essere utilizzate in scenari del mondo reale.
- Da transazioni finanziarie a autenticazioni online, vedremo come questa tecnologia può migliorare la sicurezza e la privacy.

... AnonCreds

La Specifica Anoncreds:

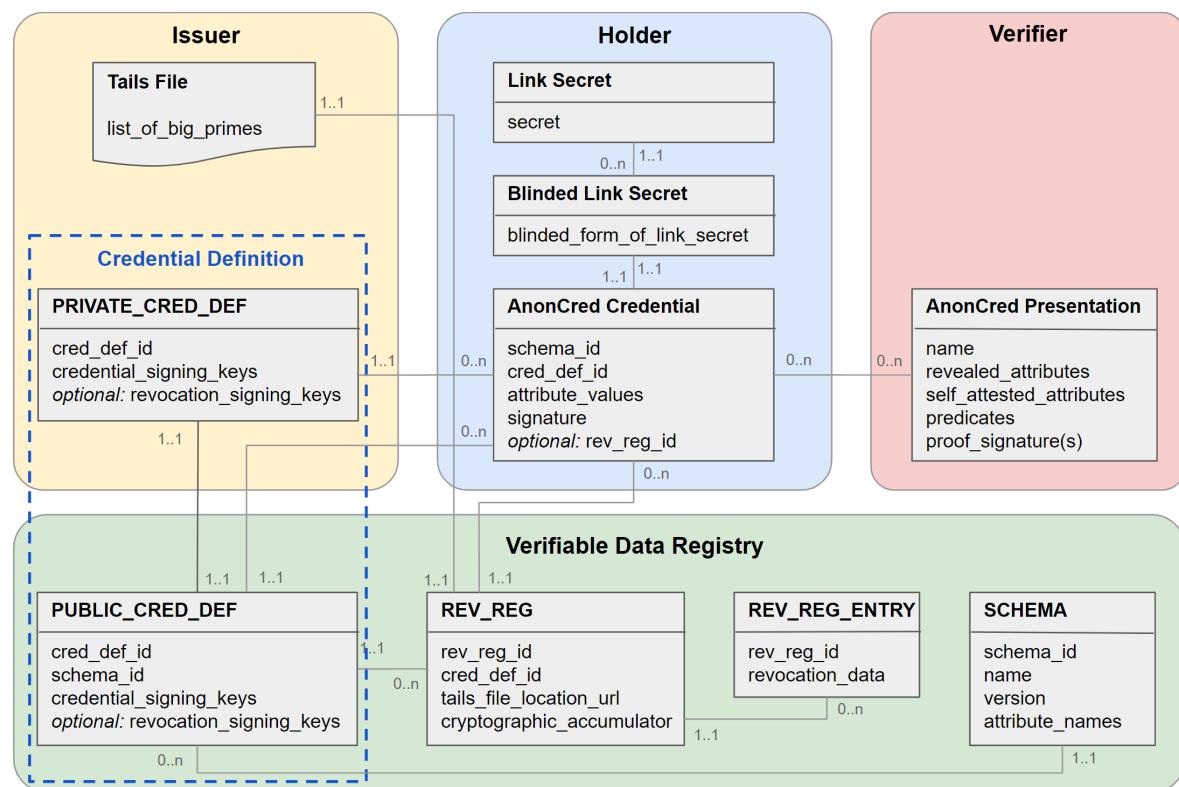
-Esaminiamo in dettaglio la specifica Anoncreds per capire i protocolli e gli algoritmi che ne regolano la creazione e la verifica.

-Questa specifica fornisce le linee guida tecniche necessarie per implementare correttamente le Anoncreds in vari contesti.



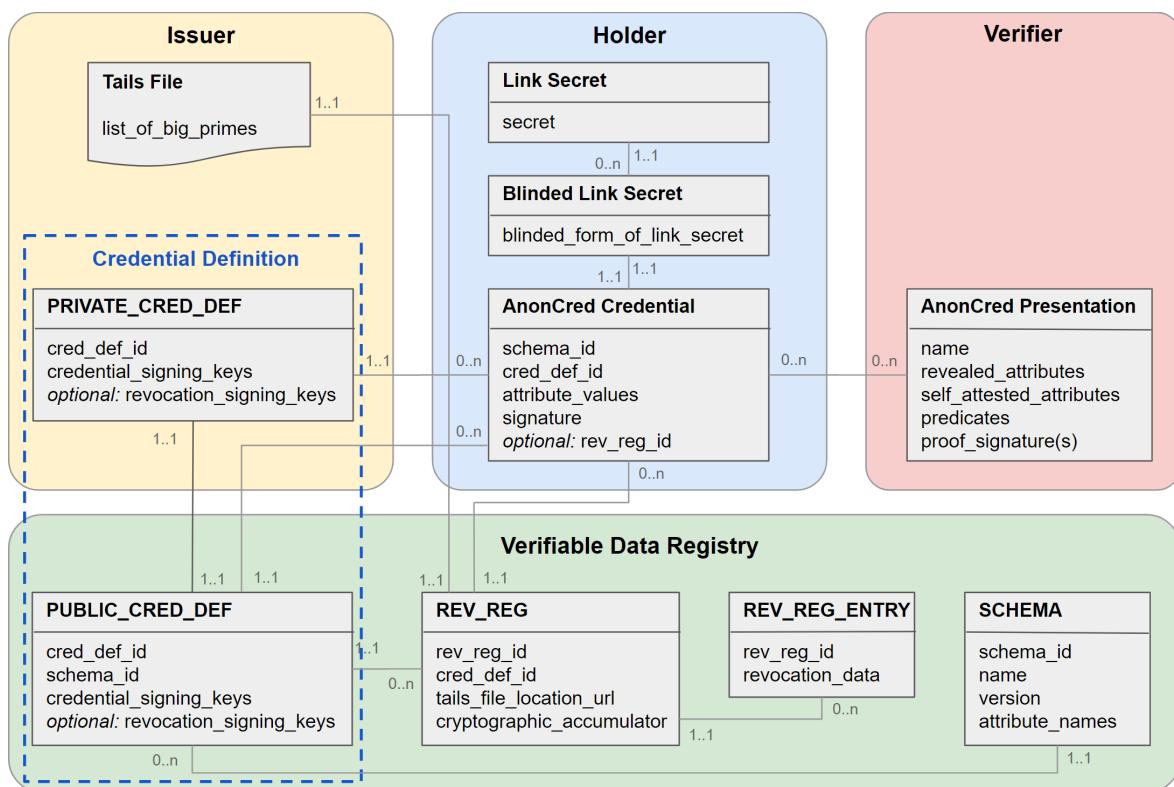
... AnonCreds

Gli **Issuer** possono creare una "Credential Definition" basata su uno Schema, consentendo loro di emettere credenziali AnonCreds ai Holders. Una Credential Definition comprende informazioni private (chiavi di firma dell'Issuer) e pubbliche (chiavi pubbliche dell'Issuer), quest'ultima memorizzata su un VDR per la verifica.



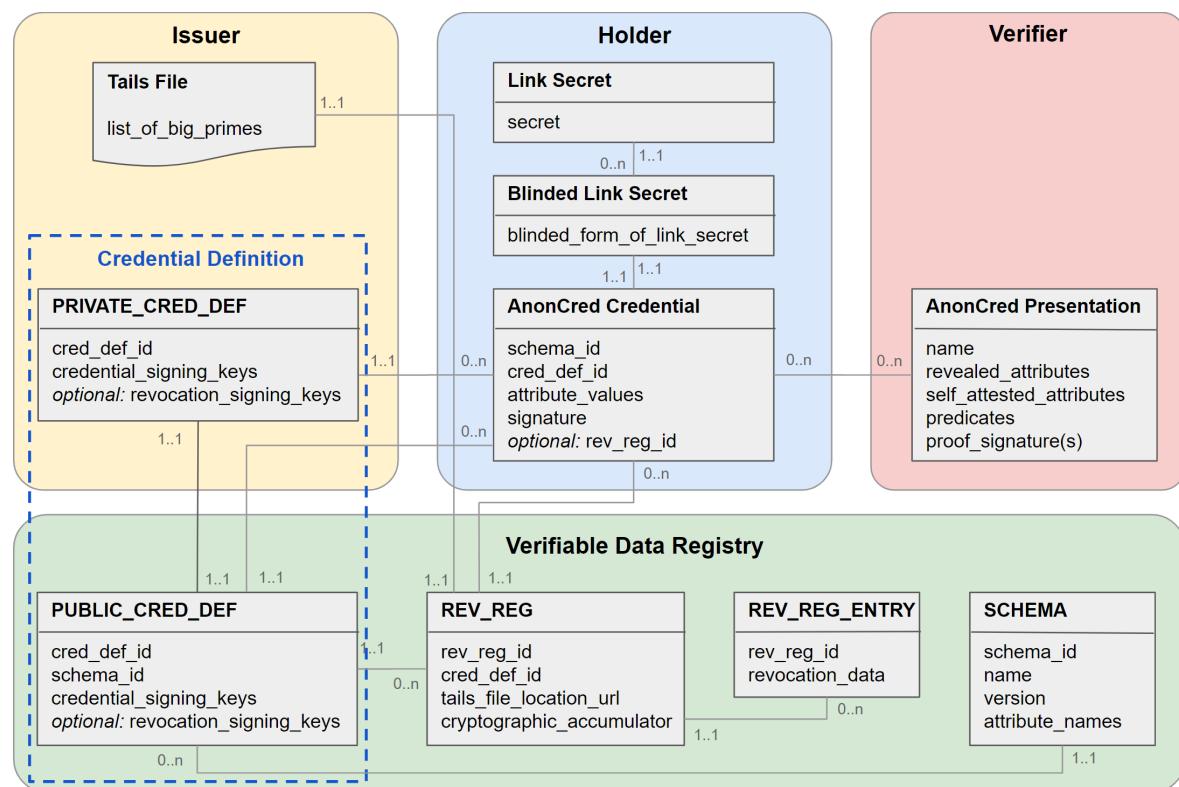
::: AnonCreds

Holder ha un "link secret" che lega la credenziale al Holder. Quando una credenziale viene emessa, il **Holder** genera un "blinding factor" e lo invia all'**Issuer**. Quest'ultimo verifica il commitment e produce una firma cieca sulla credenziale. L'**Holder** rimuove il blinding factor per ottenere una firma sulla credenziale, utilizzando lo stesso link secret per dimostrare l'affiliazione di varie credenziali durante la presentazione.



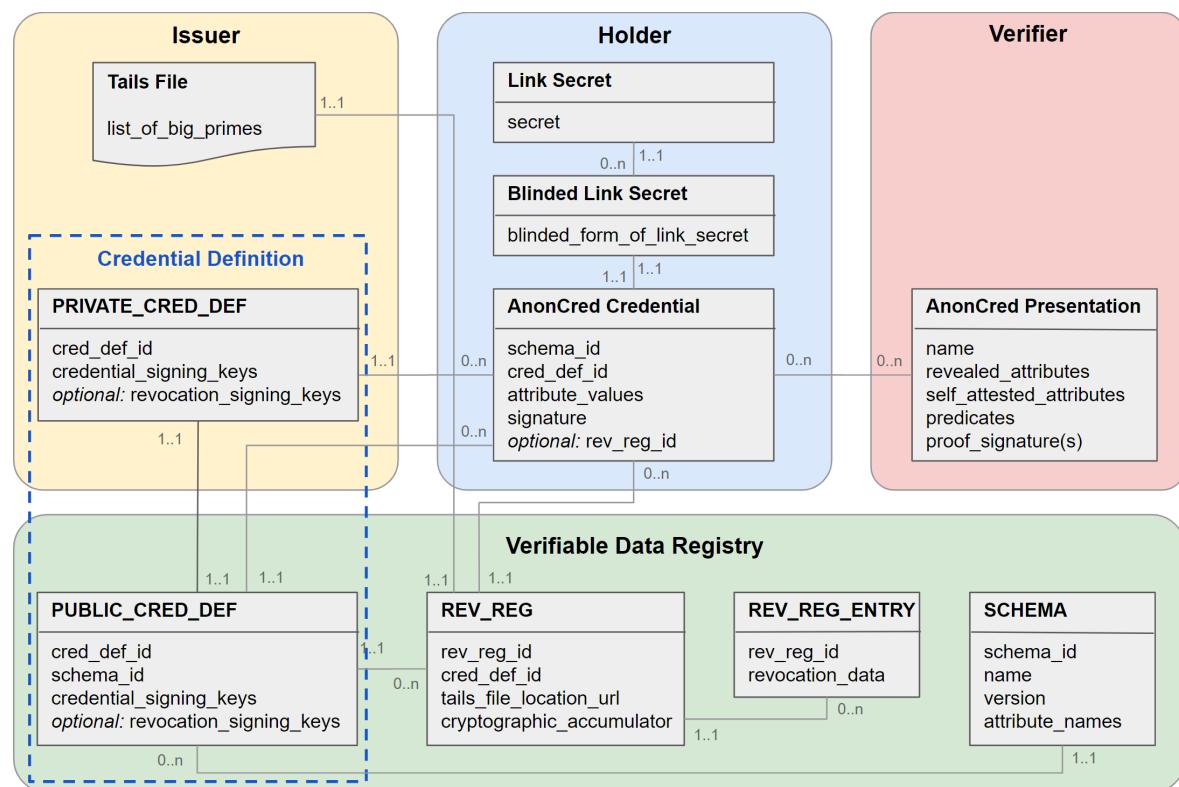
::: AnonCreds

Holder non presentano mai i dati grezzi delle credenziali ai **Verifiers**. Invece, inviano una "Verifiable Presentation," derivata dalla credenziale AnonCreds, che dimostra la correttezza dei dati senza rivelare le firme originali. Inoltre, dimostrano la conoscenza del link secret senza rivelarlo ai **Verifiers**



... AnonCreds

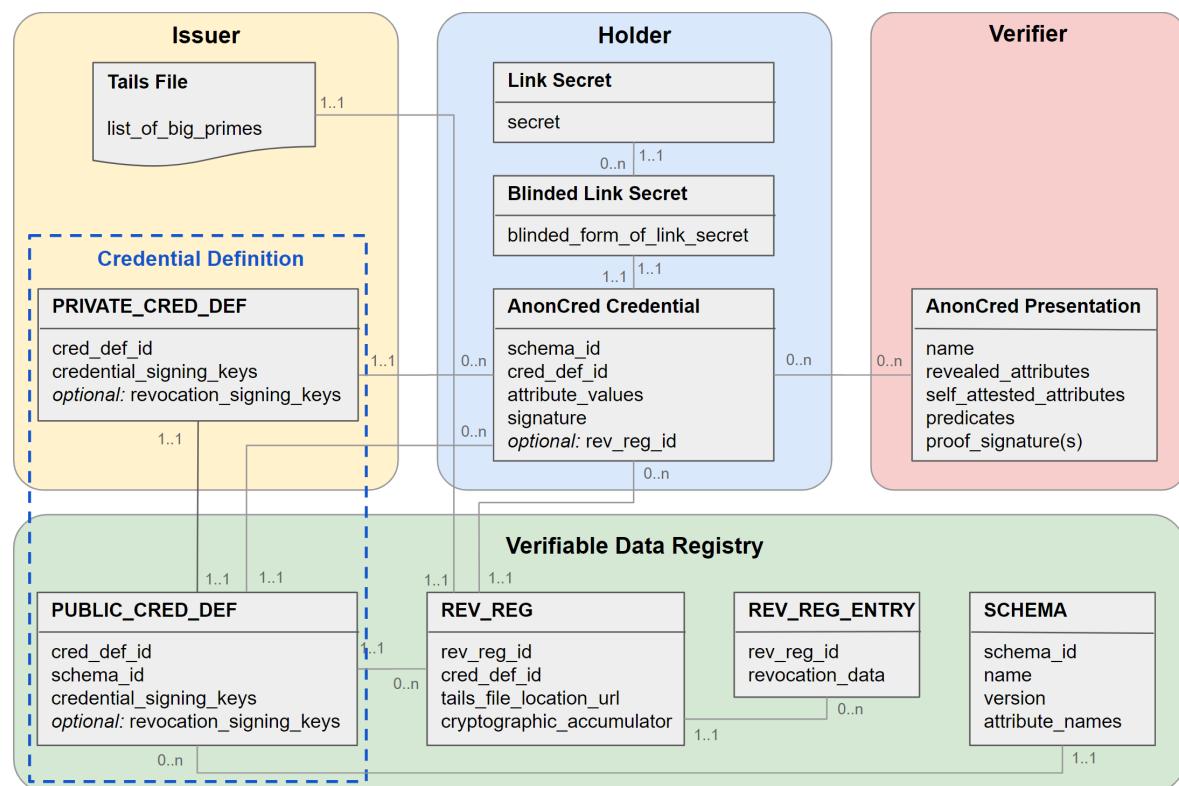
AnonCreds consente la revoca delle credenziali emesse.
Gli **Issuer** devono memorizzare una "Revocation Registry Definition" associata alla "Public Credential Definition" su un VDR. Una "Revocation Registry Definition" può avere "Revocation Status Lists."



::: AnonCreds

Holder utilizzano queste informazioni per generare una "Non-Revocation Proof," dimostrando che la credenziale non è stata revocata.

I **Verifiers** usano queste informazioni per verificare la "Non-Revocation Proof" del Holder. Un "Tails File" supporta il meccanismo di revoca, con ciascuna "Revocation Registry Definition" che richiede esattamente un "Tails File."



... Emissione AnonCreds

Preparazione della "Credential Offer"

- L'Issuer crea una "Credential Offer" contenente un impegno sulla credenziale da emettere.

Invio della "Credential Offer" al Holder

- L'Issuer invia la "Credential Offer" al Holder per valutazione.

Valutazione e Recupero dei Dati da Verifiable Data Registry

- Il Holder valuta l'offerta e recupera i dati dalla Verifiable Data Registry.

Preparazione della "Credential Request"

- Utilizzando i dati della "Credential Offer" e della "Public Credential Definition," il Holder prepara una "Credential Request."

Invio della "Credential Request" all'Issuer

- Il Holder invia la "Credential Request" all'Issuer, contenente un impegno crittografico al suo link secret.

Verifica e Preparazione della Credenziale da Parte dell'Issuer

- L'Issuer verifica la "Credential Request" e, se accettata, prepara la credenziale.

Invio della Credenziale al Holder

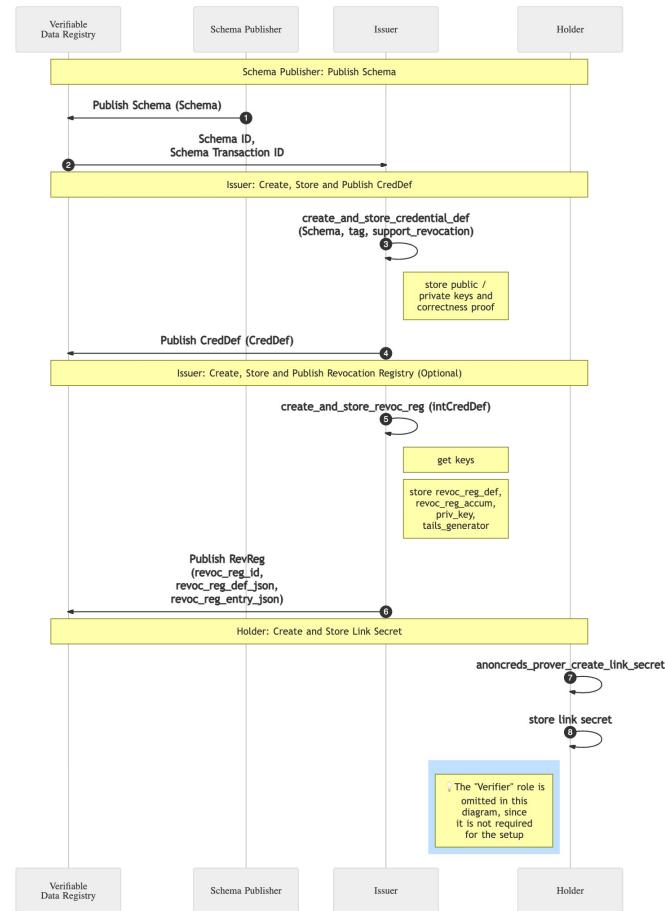
- L'Issuer invia la credenziale al Holder.

Rimozione del "Blinding Factor" e Verifica da Parte del Holder

- Il Holder rimuove il "blinding factor" dalla credenziale e la verifica.

Archiviazione Sicura della Credenziale da Parte del Holder

- Il Holder conserva in modo sicuro la credenziale.



... Verifica AnonCreds

Il flusso delle operazioni per richiedere, creare e verificare una presentazione verificabile è illustrato nel diagramma di sequenza del flusso di dati di presentazione AnonCreds.

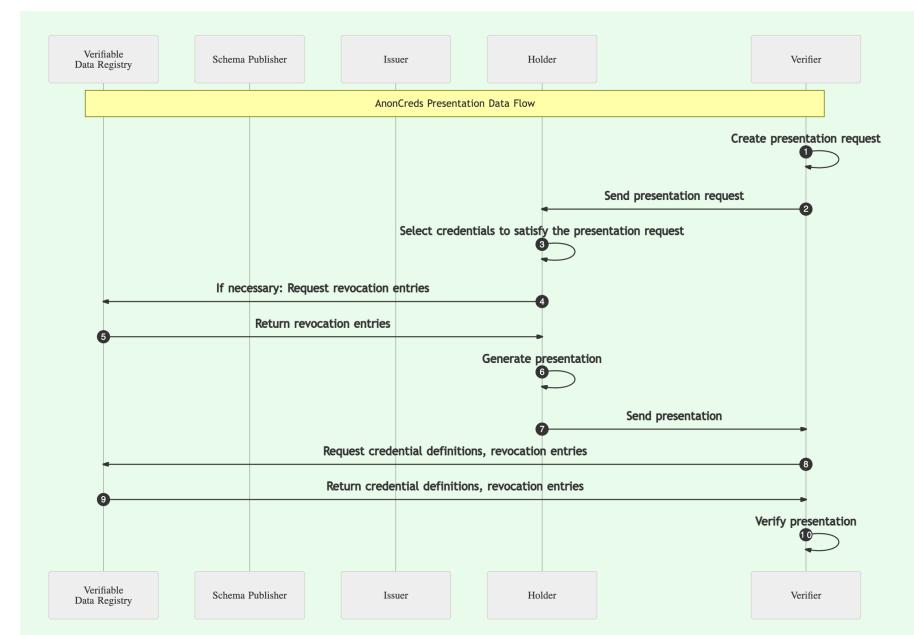
Il **Verifier** avvia il processo nel passo 1 creando e inviando una richiesta di presentazione al **Holder**.

Nel passo 2, il **Verifier** invia la richiesta di presentazione al **Holder**.

Nei passaggi 3-6, l' **Holder** raccoglie le informazioni necessarie e crea la presentazione verificabile per soddisfare la richiesta di presentazione ricevuta dal **Verifier**. Se l'**Holder** non dispone delle credenziali necessarie per soddisfare la richiesta, può ignorare la presentazione.

Nel passo 7, il **Holder** invia la presentazione verificabile secondo la richiesta di presentazione al **Verifier**.

Nei passi 8-10, il **Verifier** raccoglie le informazioni necessarie e verifica la presentazione verificabile, accettandola se la firma è valida, altrimenti rifiuta la presentazione verificabile.



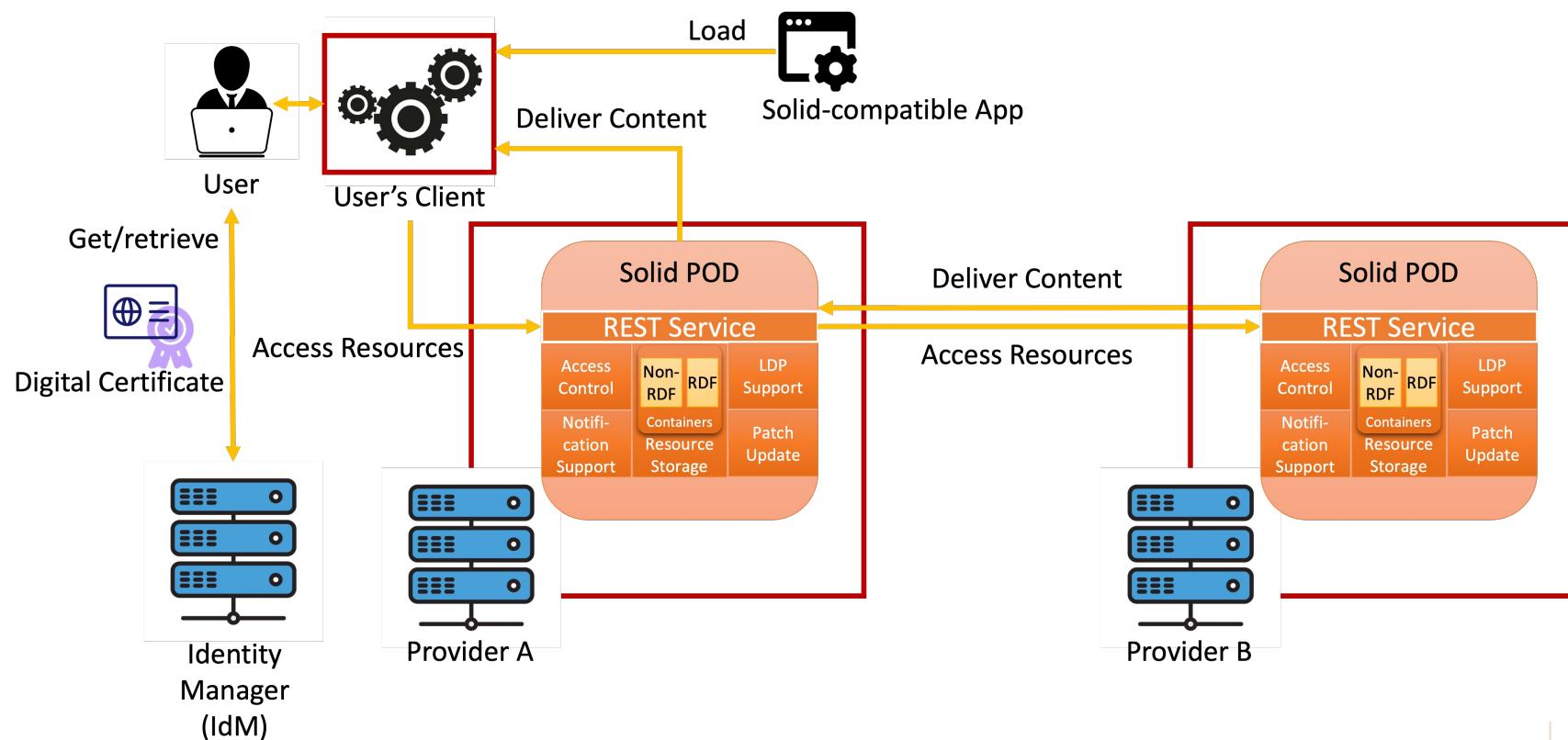


Social Linked Data (SOLID)



... Solid

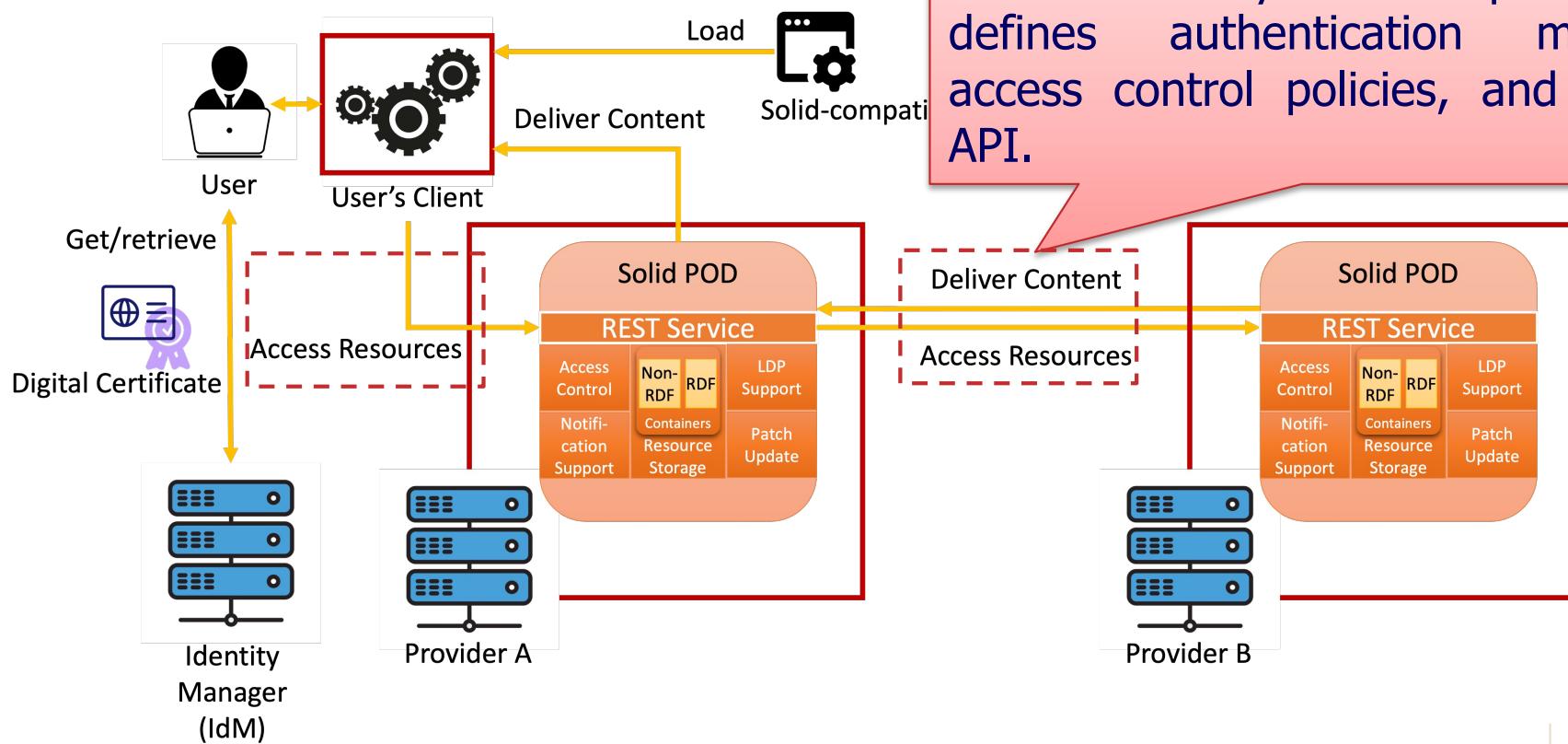
Solid aims at realizing decentralised data management, where users have full control of their data within the pods.



... Solid

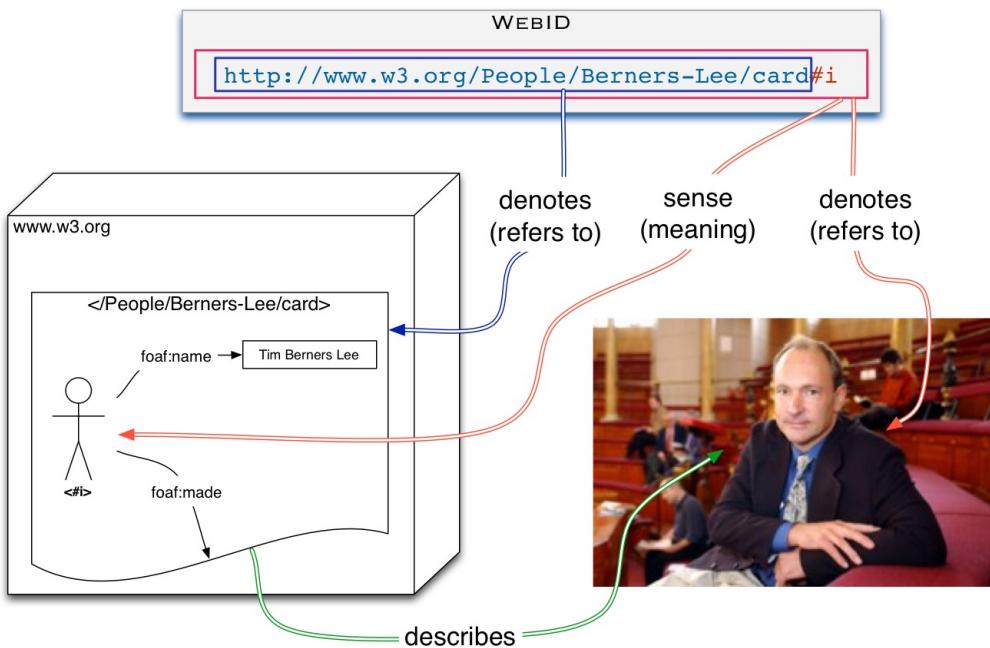
Solid aims at realizing decentralised data management, where users have full control of their data.

Access to data by Solid apps is standardised by the Solid protocol, which defines authentication mechanisms, access control policies, and a RESTful API.



::: WebID

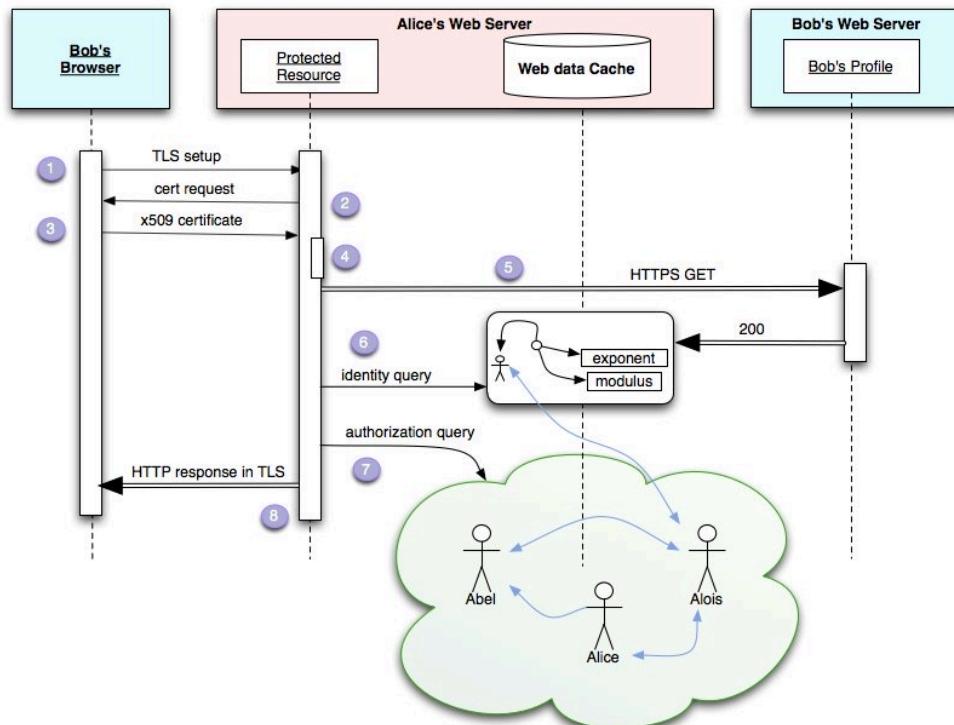
Identities in Solid are expressed using WebID, which consists in a URI naming an agent on the Web. If a WebID is looked up, a profile document or WebID-Profile is obtained where its referent is described.



WebIDs can be associated to software agents or users to represent their identities and implement authentication and authorization schemes.

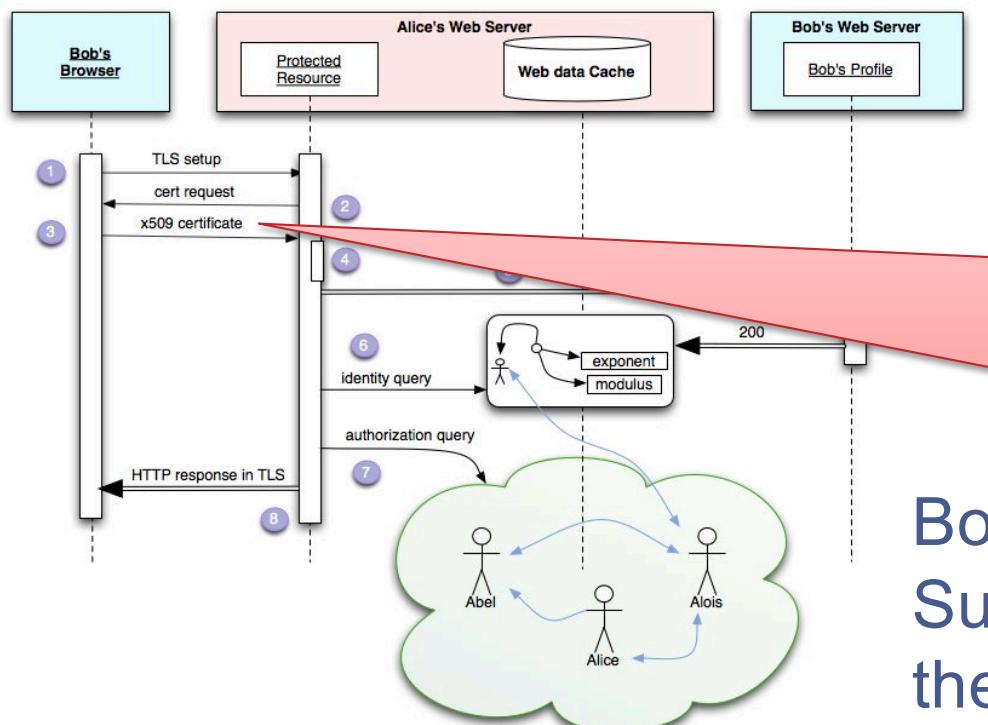
::: WebID-TLS

Bob connects to a protected resource on Alice web server by using HTTPS. After establishing a connection, the digital certificate of Bob is requested.



... WebID-TLS

Bob connects to a protected resource on Alice web server by using HTTPS. After establishing a connection, the digital certificate of Bob is requested.



Public Key Info:

Public Key Algorithm: rsaEncryption

Public-Key: (1024 bit)

Modulus:

00:e8:f9: [snip] :c6:af:2e

Exponent: 65537 (0x10001)

X509v3 extensions:

X509v3 Subject Alternative Name:

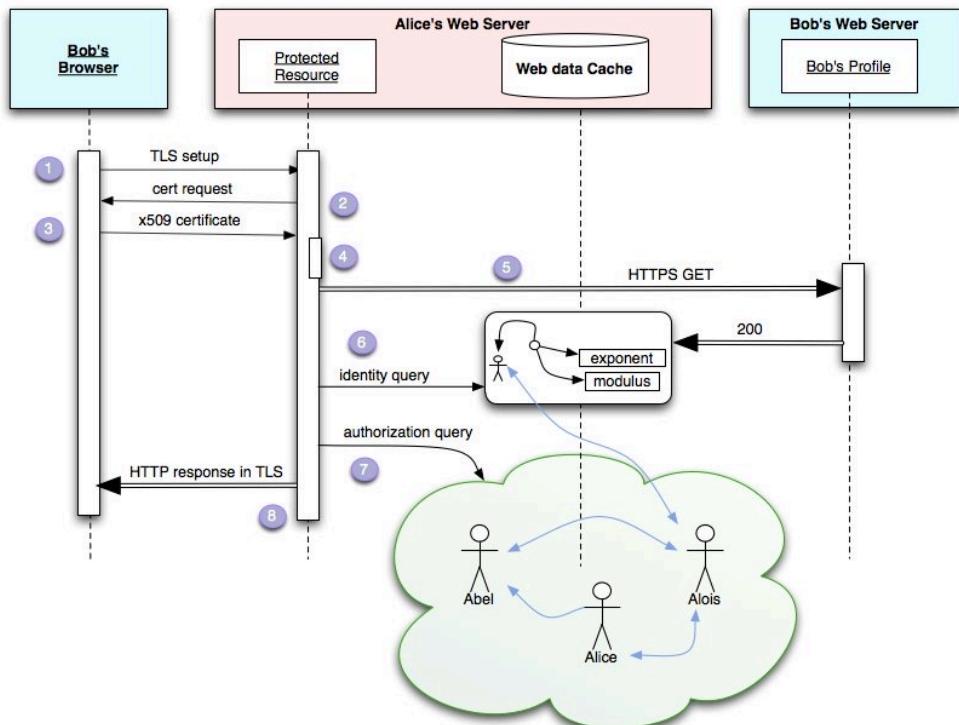
URI: <https://bob.net/id/bob#me>

Bob's WebID is obtained from the Subject Alternative Name field of the certificate.



::: WebID-TLS

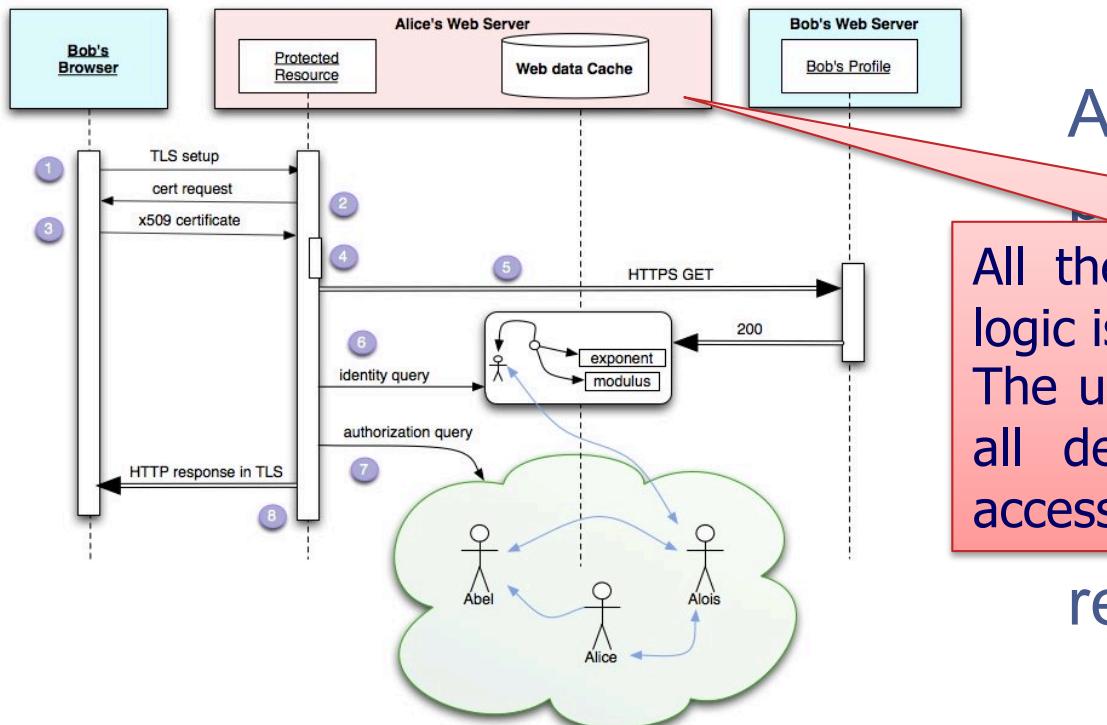
Alice's server fetches Bob's WebID Profile, a document in a machine readable way using W3C standards.



Alice's server checks that the profile relates Bob's WebID to the public key found in the certificate. A match proves she is in communication with the agent referred to by the received WebID.

::: WebID-TLS

Alice's server fetches Bob's WebID Profile, a document in a machine readable way using W3C standards.



Alice's server checks that the file relates Bob's WebID to

All the identification and authentication logic is within the Alice's resource. The user Bob reveal his/her identity and all details to the resource he/she is accessing

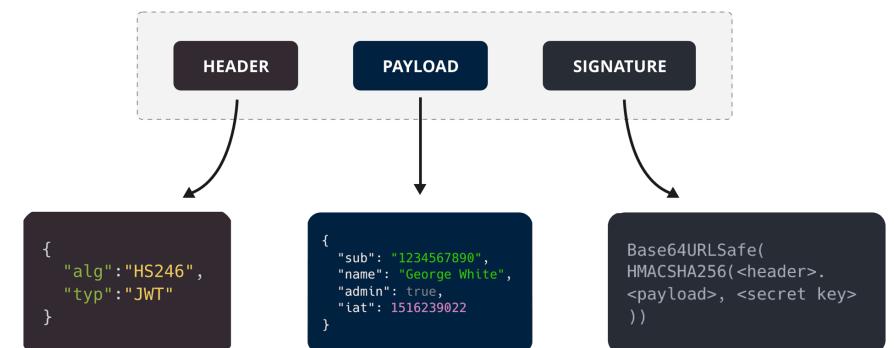
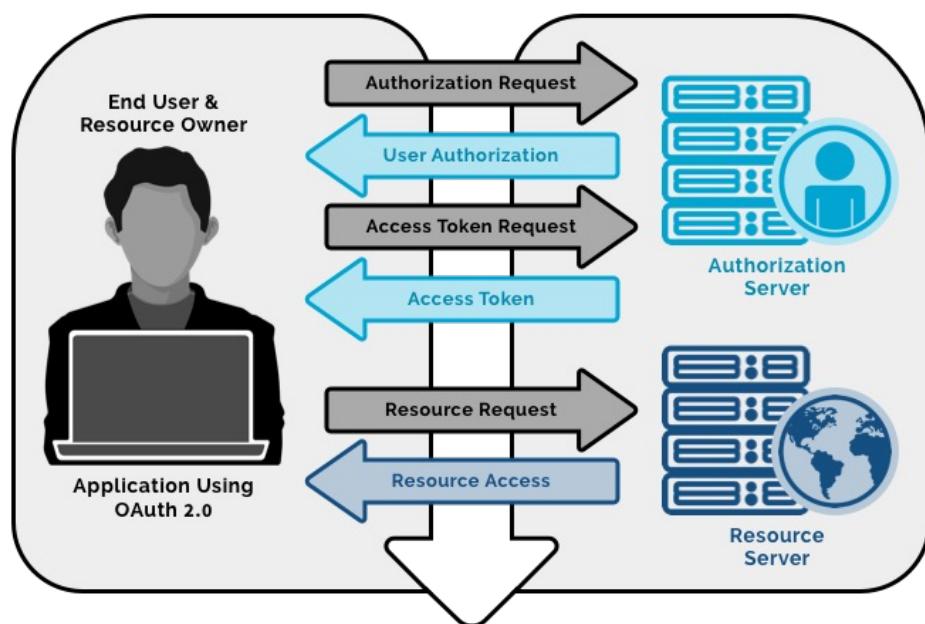
received WebID.



... OAuth 2.0 Framework

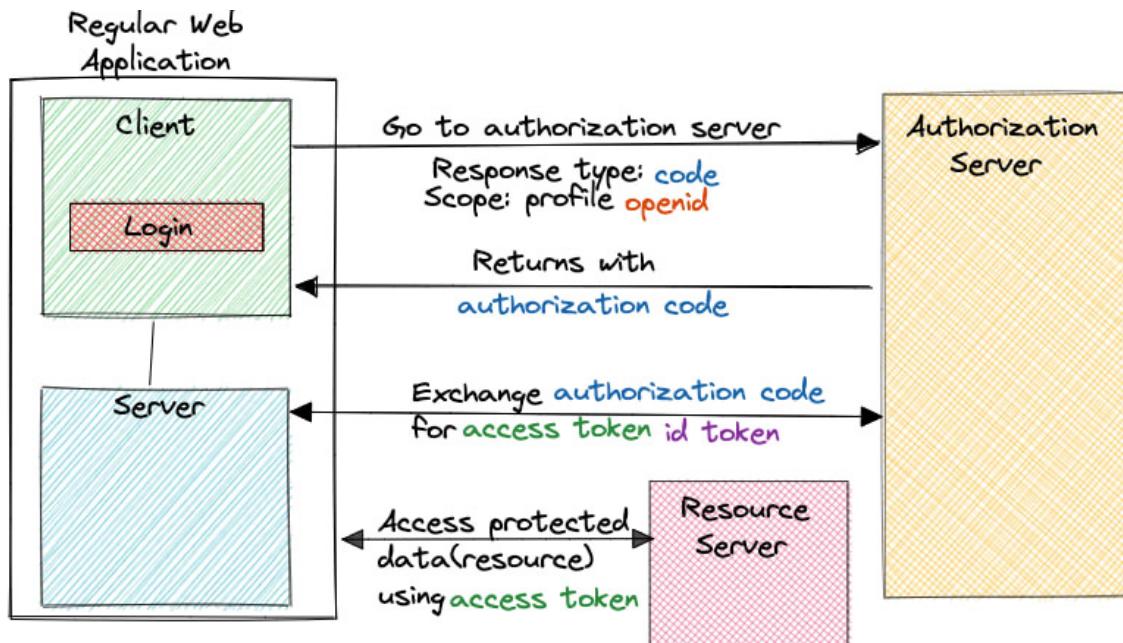
The OAuth 2.0 authorization framework allows a user to grant an access to the user's protected resources without necessarily revealing their long-term credentials or even their identity.

The user Bob interacts with an authorization server, which releases to valid users an Access in JSON web token (JWT) format.



... OpenID Connect (OIDC) 2.0

OIDC is an identity layer built on top of the OAuth 2.0 framework to verify the identity of the end-user and to obtain basic user profile information using JSON web tokens (JWTs).



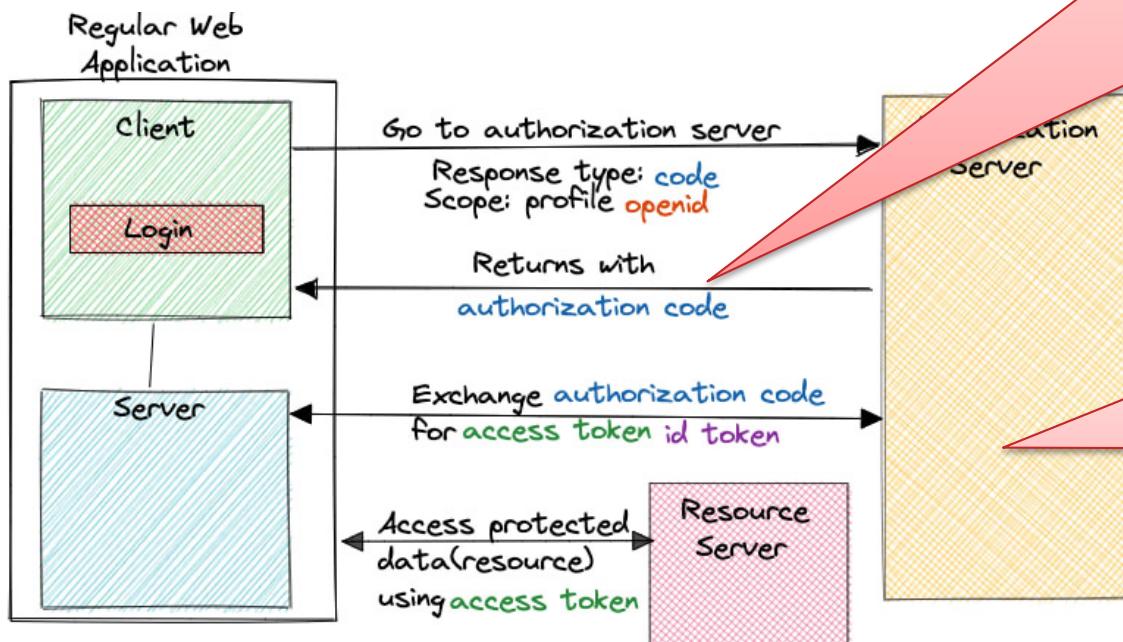
OIDC gives users one login for multiple sites: when logging in to a website, a user is redirected to a preferred OpenID site where he/she logs in, and then taken back to the website.



... OpenID Connect (OIDC) 2.0

OIDC is an identity layer built on top of the OAuth 2.0 framework to verify the identity of users based on basic user profile information.

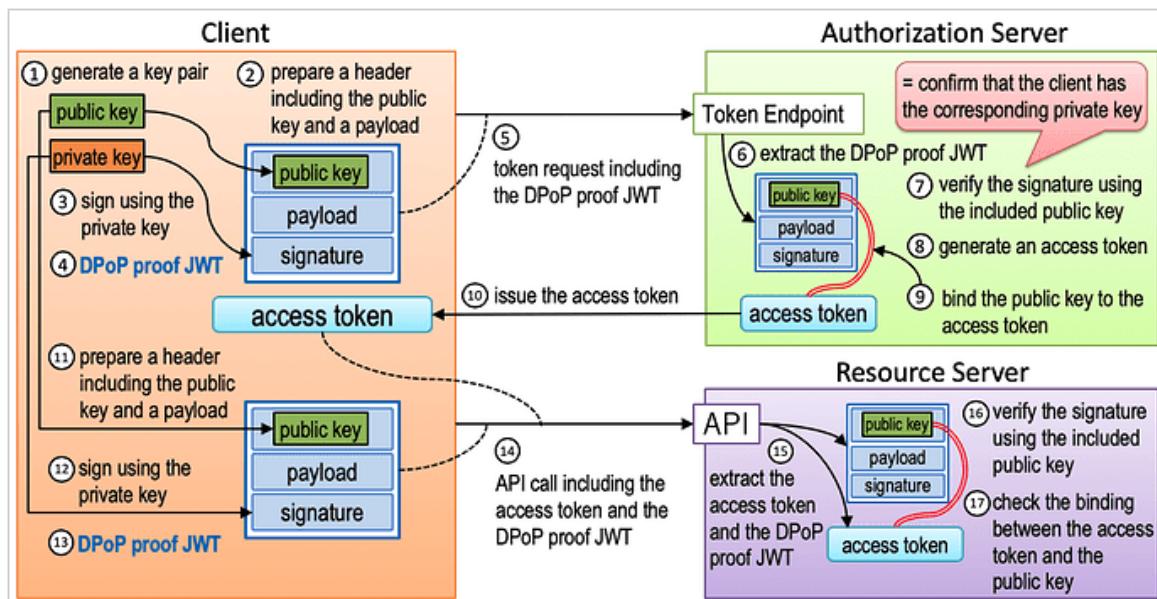
ID Token contains a set of personal attributes about a user and is usually expressed in JWTs.



OIDC login allows users one identity across multiple sites: it links the user to a single IdM which can learn behaviours of the Data Subject.

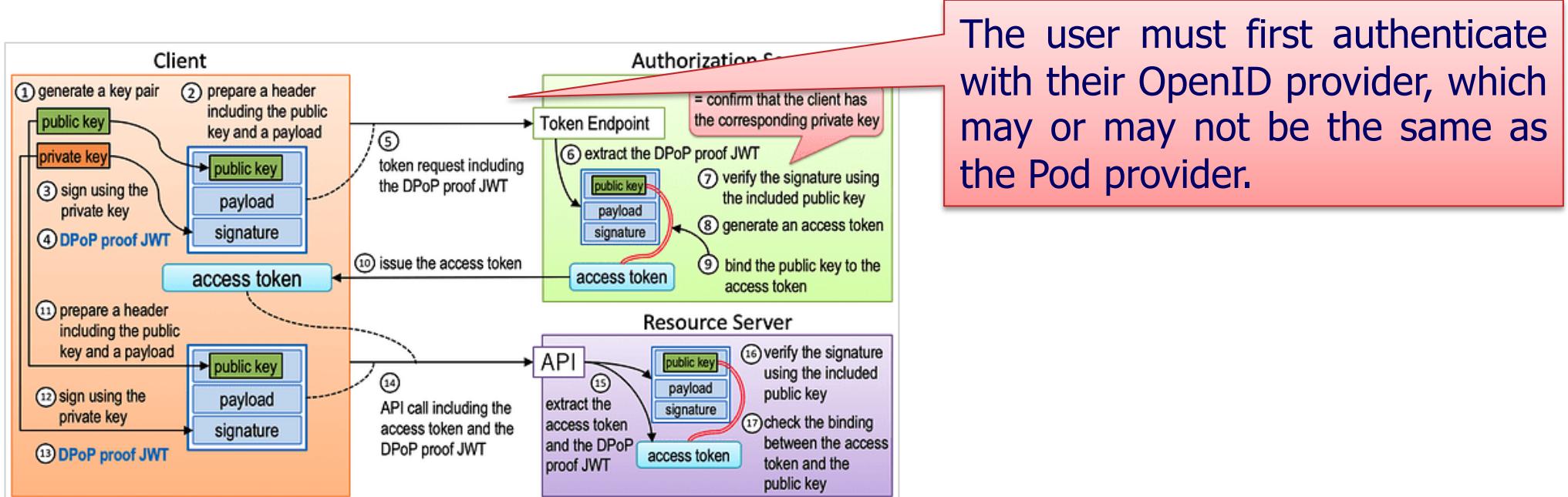
::: Solid-OIDC

The Solid OpenID Connect (Solid OIDC) specification defines how resource servers verify the identity of relying parties and end users based on the authentication of an OpenID provider.



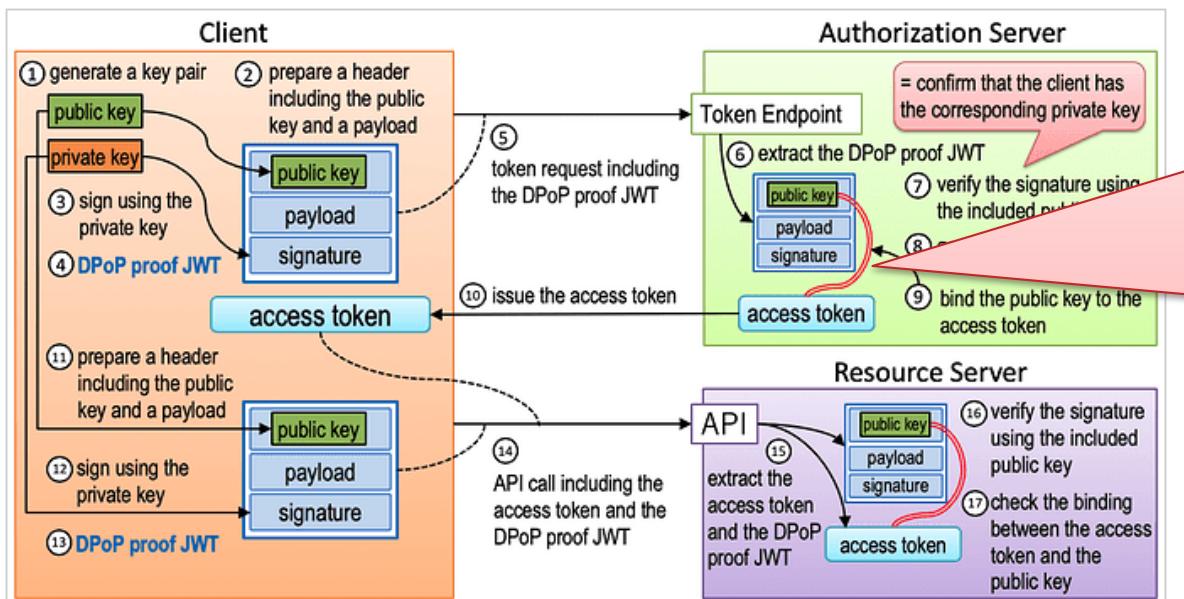
... Solid-OIDC

The Solid OpenID Connect (Solid OIDC) specification defines how resource servers verify the identity of relying parties and end users based on the authentication of an OpenID provider.



... Solid-OIDC

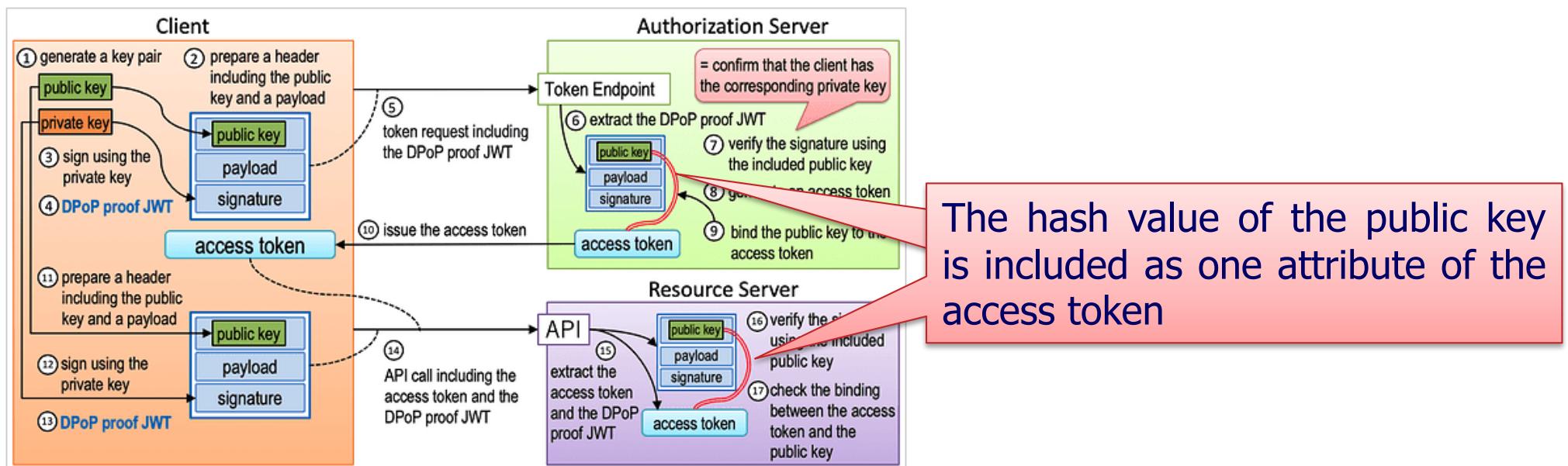
The Solid OpenID Connect (Solid OIDC) specification defines how resource servers verify the identity of relying parties and end users based on the authentication of an OpenID provider.



A Demonstration of Proof of Possession (DPoP) token is used to whether the client application presenting the access token is the valid owner of the access token, whether the client application is the same one that the access token has been issued to.

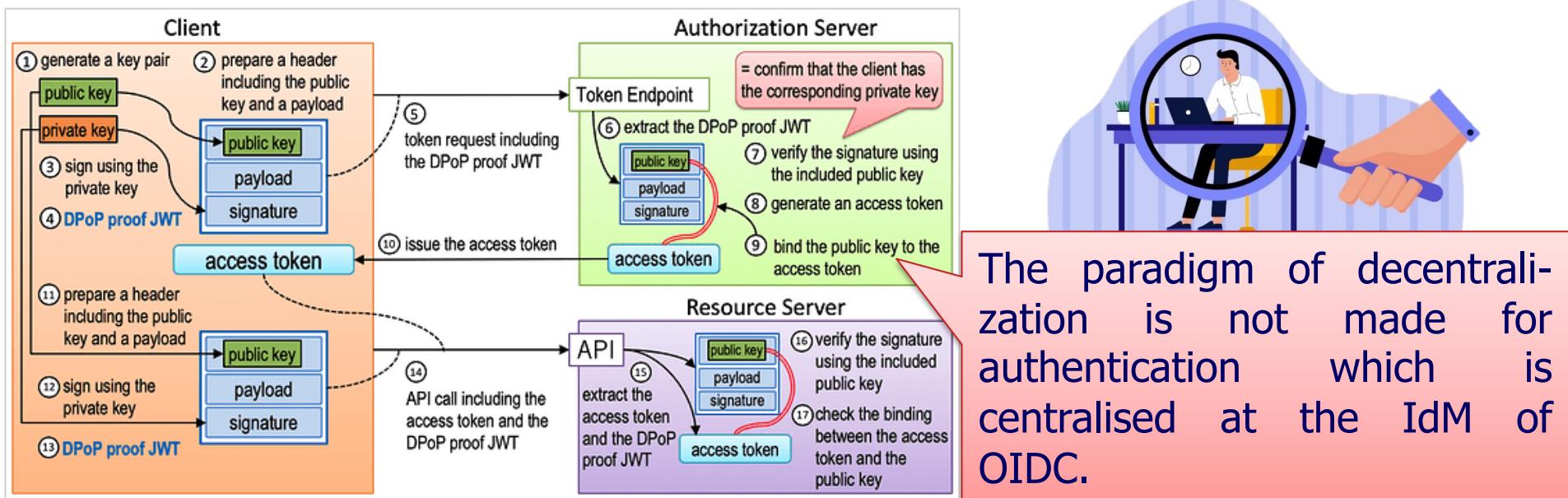
::: Solid-OIDC

The Solid OpenID Connect (Solid OIDC) specification defines how resource servers verify the identity of relying parties and end users based on the authentication of an OpenID provider.



... Solid-OIDC

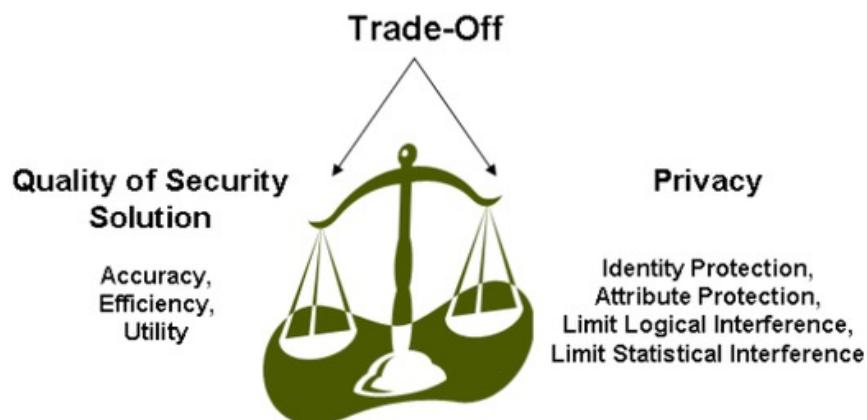
The Solid OpenID Connect (Solid OIDC) specification defines how resource servers verify the identity of relying parties and end users based on the authentication of an OpenID provider.



::: Solid-OIDC

Solid-OIDC falls short of GDPR requirements: IdM can learn over time which relying on party the user logs in to.

Privacy may be preserved by using anonymous (i.e., unauthenticated) access to resources.



However, this is in contract to the accountability principle of GDPR and the security concerns of restricting sensitive resources to known and authorised users.

::: Decentralised IdM

Centralized Identity

Register once, trusted by one

Service provider establishes and maintains users' identity and related data in its system



Federated Identity

Register once, trusted by many

Service provider trusts identities established and maintained by other identity providers



Decentralized Identity

Register once, trusted globally

Users have a self-managed digital identity independent of individual service providers

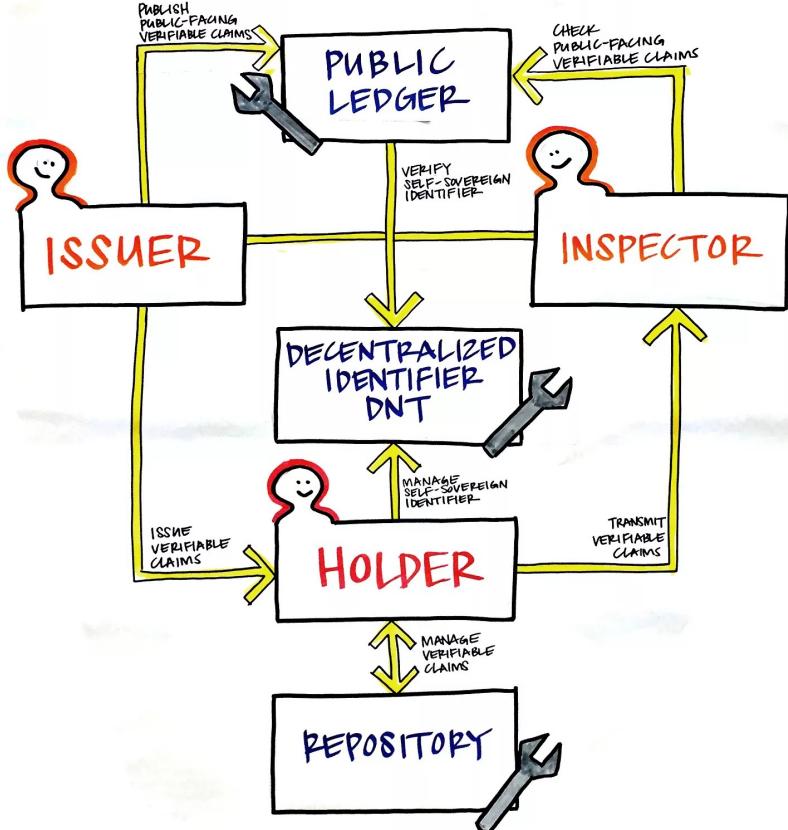


Decentralized identity gives users control over their identifying information by collecting and verifying information with a wallet.



... Self-Sovereign Identity (SSI)

SELF-SOVEREIGN IDENTITY ARCHITECTURE 1.0

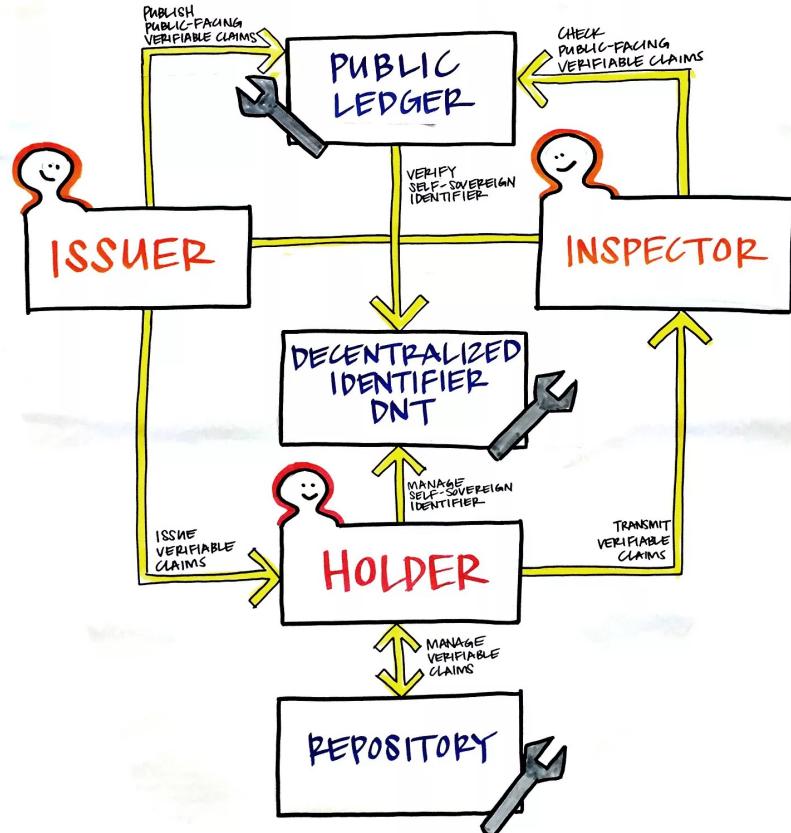


The User creates a pair of public and private keys in a wallet, where the public key is hashed and stored on a blockchain.

The issuer creates the user's Verifiable Credentials, certifies it by signing it with its own private key, and publishes it as Verifiable Claims on the.

... Self-Sovereign Identity (SSI)

SELF-SOVEREIGN IDENTITY ARCHITECTURE 1.0



Users can access third-party services as Inspector using a token that's easily verified by comparing the hash values of the identification record with the hash values of the certification record with claims from the blockchain.

::: Hyperledger Indy+Aries

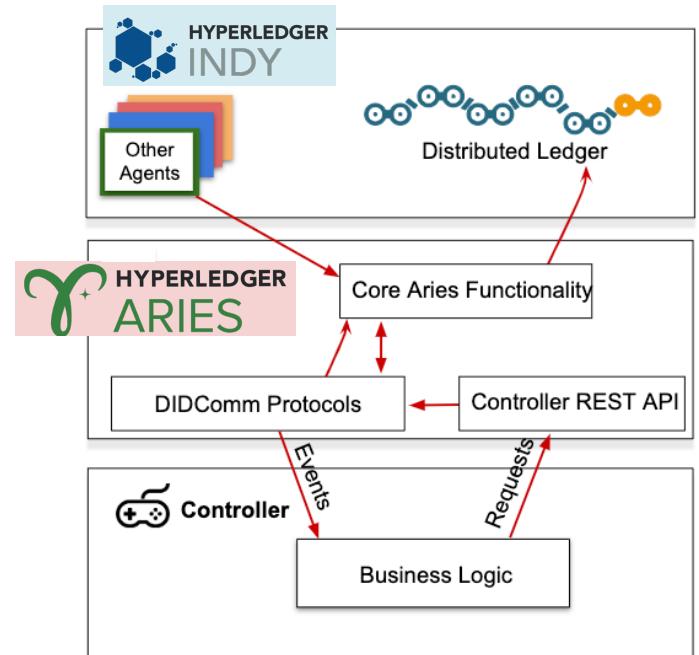
Research Questions



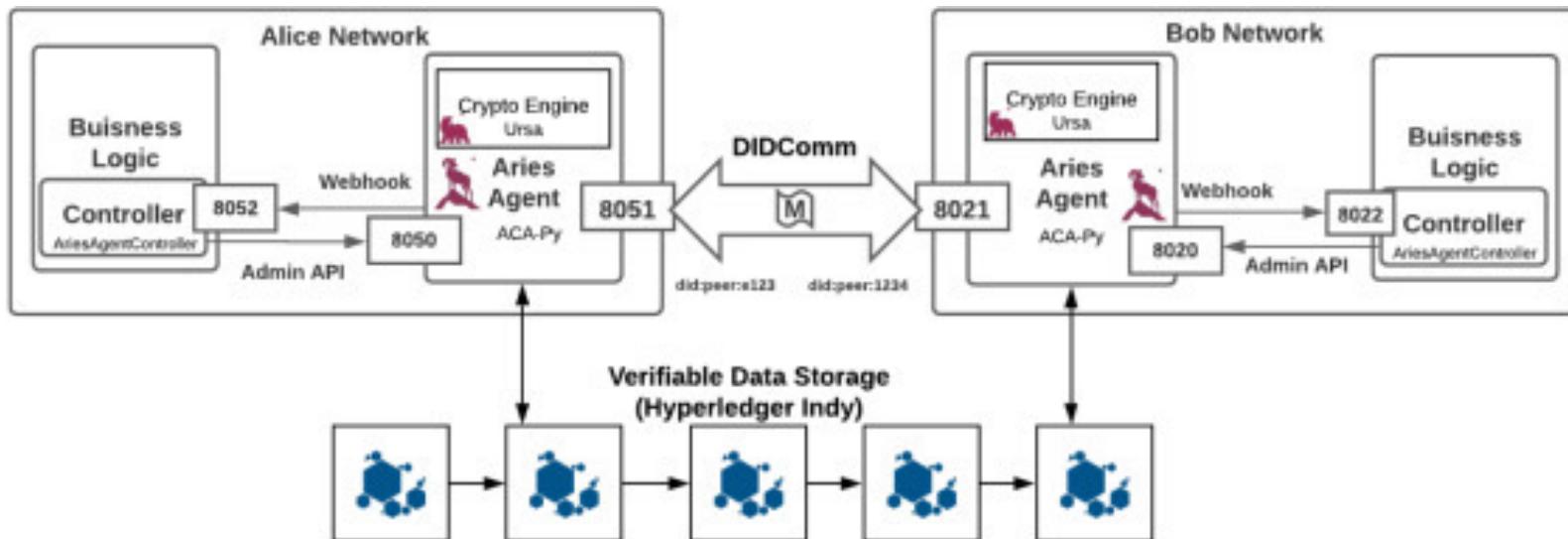
Is there a mature implementation of the SSI scheme to be usable within Solid?

Hyperledger Indy provides tools, libraries, and reusable components for providing digital identities rooted on blockchains.

Hyperledger Aries provides a shared, reusable, interoperable tool kit to create, transmit and store verifiable digital credentials.



::: Hyperledger Indy+Aries



Aries provides an agent to generate VC cryptographically protected by the engine provided by Ursa and to be stored and retrieved from the blockchain of Indy.

An authentication solution can be implemented by properly interacting with the ACAPy agent of Aries.

::: SOLID-SSI

Research Questions



How can we integrate an SSI-based identity management and authentication within Solid?

::: SOLID-SSI

Research Questions



How can we integrate an SSI-based identity management and authentication within Solid?

In order to integrate Hyperledger Aries in Solid, it is necessary to define which role the server will play. It can be both Issuer and Verifier, able to create and test credentials.

```
{  
  "label": "solid-agent",  
  "walletConfig": {  
    "id": "solid-wallet",  
    "key": "testkey00000000000000000000000000000000"  
  },  
  "indyLedgers": [  
    {  
      "id": "bcovrin-test-net",  
      "isProduction": false,  
      "indyNamespace": "bcovrin:test",  
      "genesisTransactions": genesisTransactionsBCovrinTestNet  
    }  
  ],  
  "autoAcceptCredentials": AutoAcceptCredential.ContentApproved,  
  "autoAcceptConnections": true,  
  "endpoints": [  
    "http://172.19.148.62:3002"  
  ]  
}
```



... SOLID-SSI

VON Network is a portable development-level Indy Node network, it is not possible to issue credentials starting from non-registered DID. For this reason, it's necessary to register a new DID on the public ledger.

The screenshot shows the BCovrin Test dashboard interface. On the left, under 'Validator Node Status', there are four nodes listed: Node1, Node2, Node3, and Node4. Each node has a status card with its DID, Uptime, Txns, and Indy-node version. Below this is a link to 'Detailed Status'. On the right, there are two sections: 'Connect to the Network' (with tabs for 'Genesis Transaction' and 'JSON') and 'Authenticate a New DID' (with fields for 'Wallet seed', 'DID (optional)', 'Alias (optional)', 'Role' (set to 'Endorser'), and a 'Register DID' button).

Once having defined the agent run by the Solid server, it's necessary to release credentials to the user.

Aries allows issuing credentials in both Indy or JSON-LD format.

::: SOLID-SSI

In Indy Credentials, it is necessary to define a schema, which is publicly available on the public ledger and everyone can use it to release credentials.

```
{  
  "attributes": [  
    "webid"  
  ],  
  "schema_name": "solid",  
  "schema_version": "1.0"  
}
```

```
{  
  "cred_def_id": "2sMU7txCQcde3fPoWwoL2W:3:CL:20:default",  
  "schema_id": "2sMU7txCQcde3fPoWwoL2W:2:solid:1.0"  
}
```



... SOLID-SSI

The schema can be used in the credential definition message, used to release credentials by Issuer.

```
"auto_issue": true,  
  "comment": "string",  
  "connection_id": "2c5c3f39-430d-46d7-98ef-9759ac8cb967",  
  "credential_preview": {  
    "@type": "issue-credential/1.0/credential-preview",  
    "attributes": [  
      {  
        "name": "webid",  
        "value": user.id  
      }  
    ]  
  },  
  "filter": {  
    "indy": {  
      "cred_def_id": "2sMU7txCQcde3fPoWwoL2W:3:CL:20:default",  
      "issuer_did": "2sMU7txCQcde3fPoWwoL2W",  
      "schema_id": "2sMU7txCQcde3fPoWwoL2W:2:solid:1.0",  
      "schema_issuer_did": "2sMU7txCQcde3fPoWwoL2W",  
      "schema_name": "solid",  
      "schema_version": "1.0"  
    }  
  }  
}
```

Aries Framework internally will do the checks and before sending the credentials to the endpoint associated with the connection_id field it will attach the proof. Such proof will be sent to the user jointly with the credential.



... SOLID-SSI

The schema can be used in the credential definition message, used to release credentials by Issuer.

```
"cred_offer": {  
    "indy": {  
        "schema_id": "2sMU7txCQcde3fPoWwoL2W:2:solid:1.0",  
        "cred_def_id": "2sMU7txCQcde3fPoWwoL2W:3:CL:24:default",  
        "key_correctness_proof": {  
            "c": "84503882629592362595757960889958677283224904746470301060102913710742792064",  
            "xz_cap": "916655472344431368709883289481087625639961221485339536622878687266613",  
            "xr_cap": [  
                [  
                    "webid",  
                    "16744010761128588131999398948839112067377006217322511538475378681412662  
                ],  
                [  
                    "master_secret",  
                    "13267392878833695231110139682710940926383435936452460838871515830375934  
                ]  
            ]  
        }  
    },  
    "nonce": "1207589291460843313260089"  
}
```

A nonce field is used in order to prevent reply attacks.



... SOLID-SSI

If Bob accept the credentials, send a message with the signature of the received credential request with his key.

```
"cred_request": {  
    "indy": {  
        "prover_did": "QSFeDAkGP5ECNBPUDYKh4z",  
        "cred_def_id": "2sMU7txCQcde3fPoWwoL2W:3:CL:24:default",  
        "blinded_ms": {  
            "u": "69083427311489001319246597741696310286498054375129546261402145456633052969",  
            "ur": null,  
            "hidden_attributes": [  
                "master_secret"  
            ],  
            "committed_attributes": {}  
        },  
        "blinded_ms_correctness_proof": {  
            "c": "10428822110205544660496120821829046253346928080356351928367537806681148943",  
            "v_dash_cap": "63669302525184132648016568091574145533117906531093891057145807758",  
            "m_caps": {  
                "master_secret": "1241958076078738692839908228408019263598650531271216873346",  
            },  
            "r_caps": {}  
        },  
        "nonce": "848658656471105539394506"  
    }  
}
```

::: SOLID-SSI

Finally, the user can check the credentials stored in its wallet.

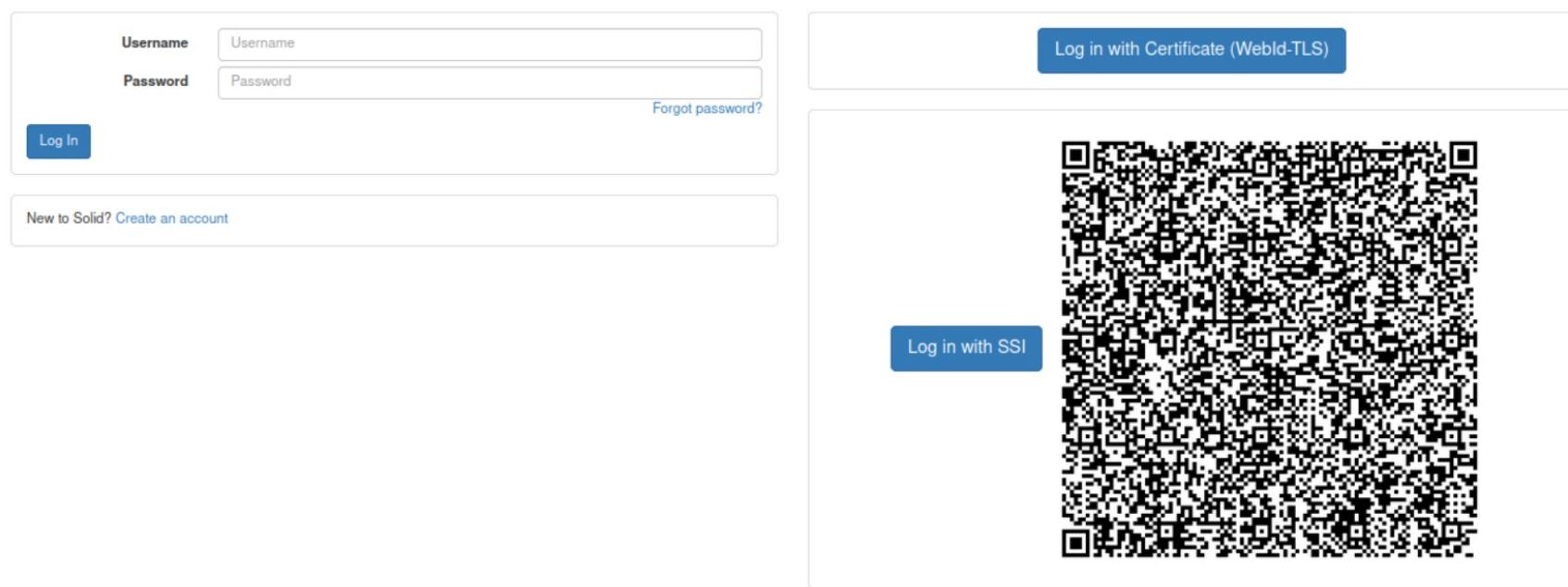
```
{  
    "results": [  
        {  
            "referent": "da60ad4c-0c46-423d-9cf4-75e969594c7b",  
            "attrs": {  
                "webid": "martini"  
            },  
            "schema_id": "2sMU7txCQcde3fPoWwoL2W:2:solid:1.0",  
            "cred_def_id": "2sMU7txCQcde3fPoWwoL2W:3:CL:24:default",  
            "rev_reg_id": null,  
            "cred_rev_id": null  
        }  
    ]  
}
```

Once the registration phase has ended with success, the user can use his Indy Credentials for the access.



::: SOLID-SSI

Login



A QR Code is available and can be scanned by the app associated with the wallet used for credentials definition and create a secure connection by exchanging the DIDs.



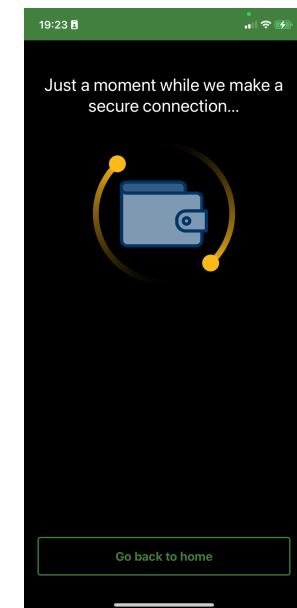
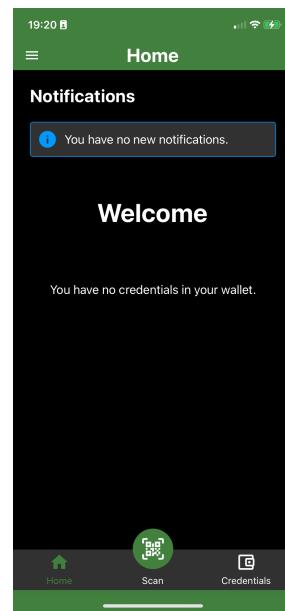
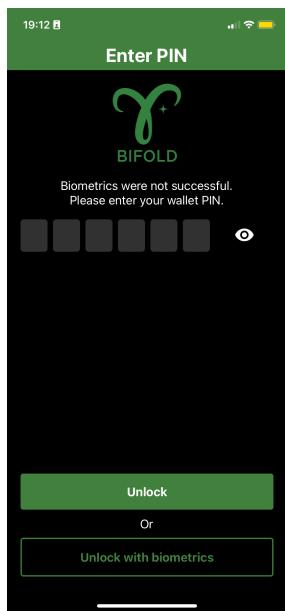
UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA



0111

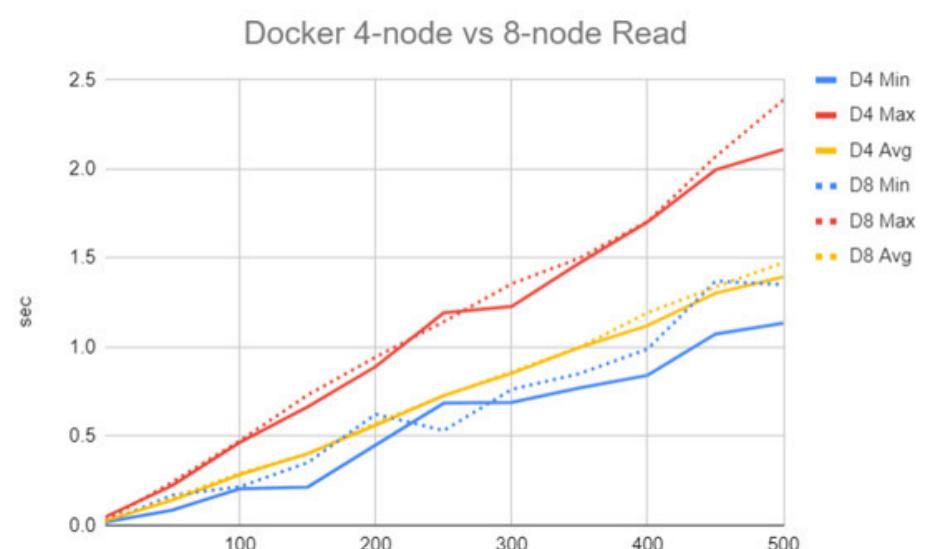
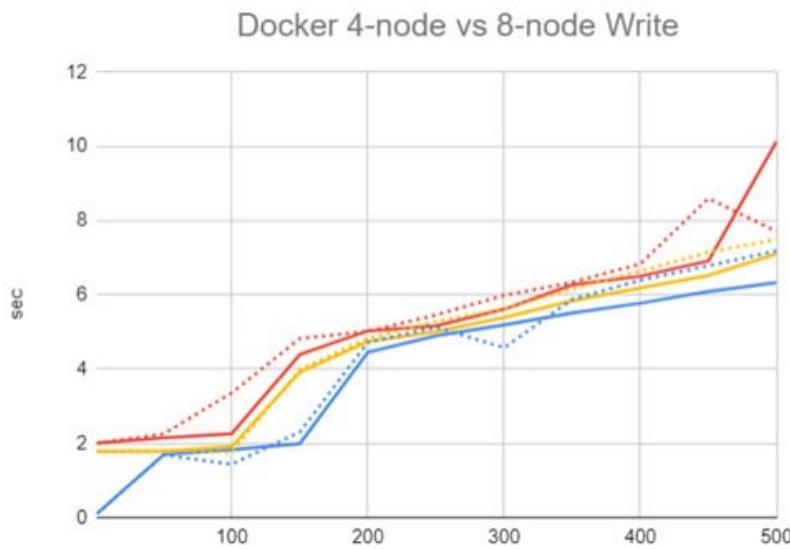
::: SOLID-SSI

The user needs an application (or an agent) to scan the QR code. Such application can be unlocked using biometric data or PIN and stores all credentials within an Indy wallet. If keys are compromised an error will break the execution.



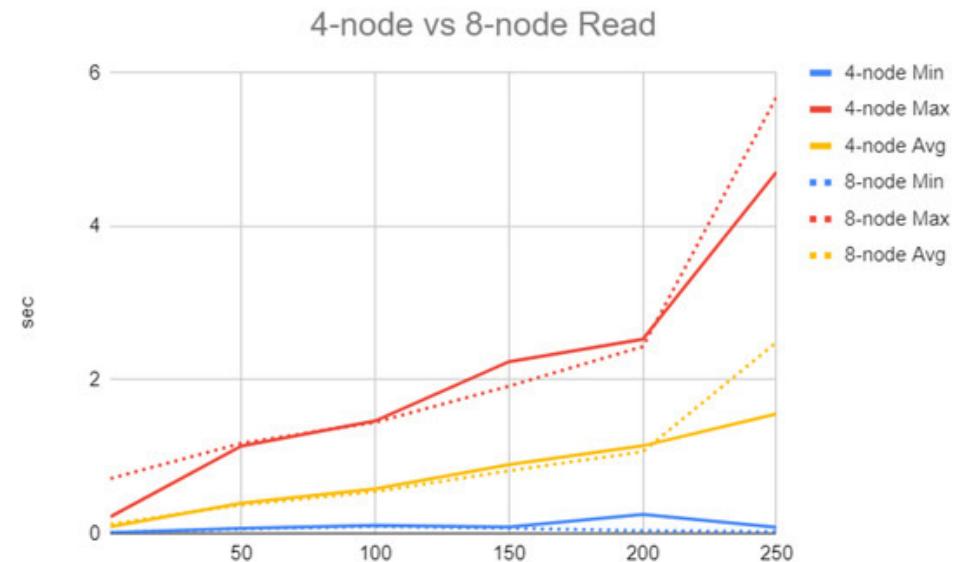
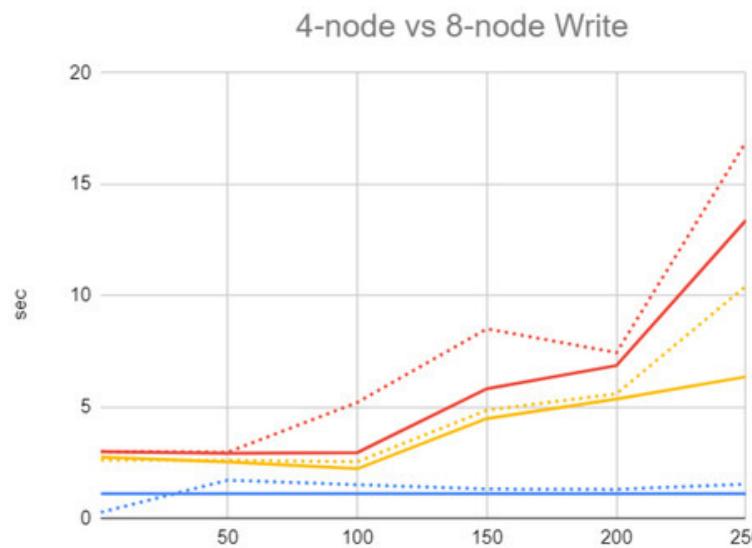
... Performance Analysis

We have measured the latency for writing and reading credentials within the SSI solution by leveraging on a local deployment of Aries/Indy.



... Performance Analysis

We have measured the latency for writing and reading credentials within the SSI solution by leveraging on a local deployment of Aries/Indy. A similar analysis has been done with a network of remote nodes with varying size.



::: Conclusions

Such approach aims at decentralize the way Solid authenticates users. We have implemented a proof-of-concept to test the validity of such an approach.

<https://github.com/biagioboi/ssi-solid-integration-doc>

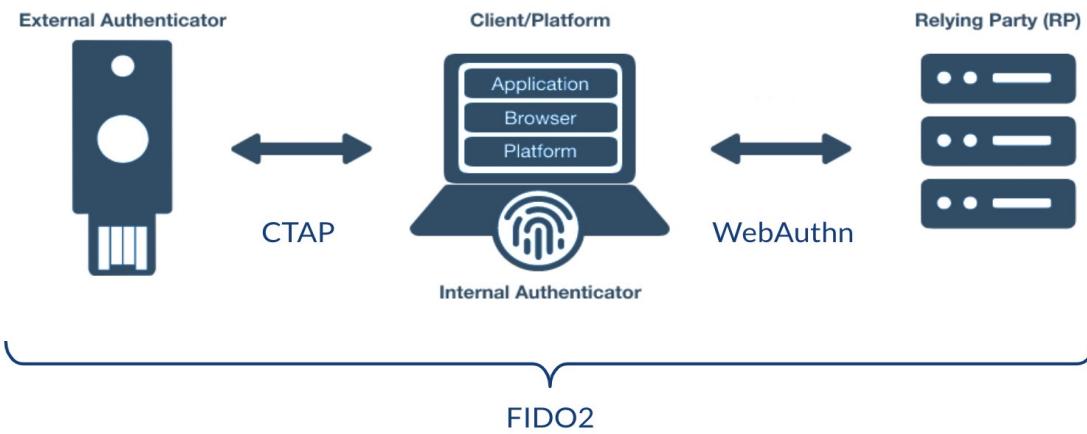
We plan as future work further investigation on the security and privacy degree offered by this solution, and its possible integration with the authorization solution provided by the Access Control Policies (ACP) within Solid.

We are also investigating other possible standards.



... FIDO 2

The FIDO2 project is a joint effort between the FIDO Alliance and the World Wide Web Consortium (W3C), whose goal is to create strong authentication for the web.

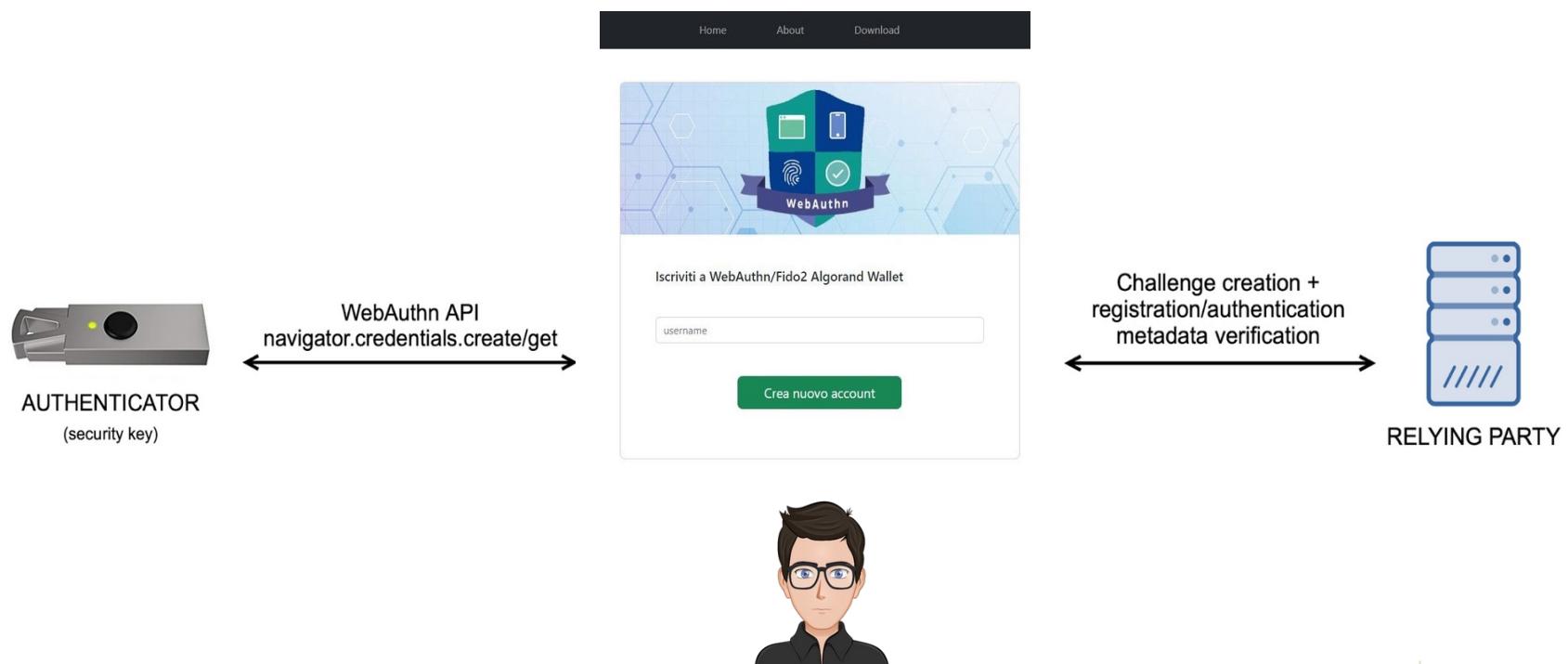


FIDO2 provides a passwordless way to authenticate users and addresses security, convenience, privacy, and scalability issues that passwords do not.

Based on cryptography, there may be an internal authenticator and an external one with a proper device, to support MFA.

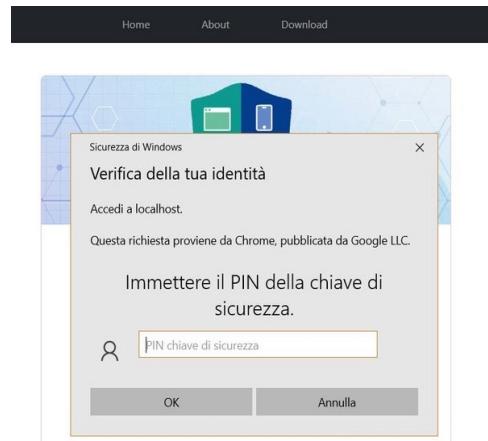
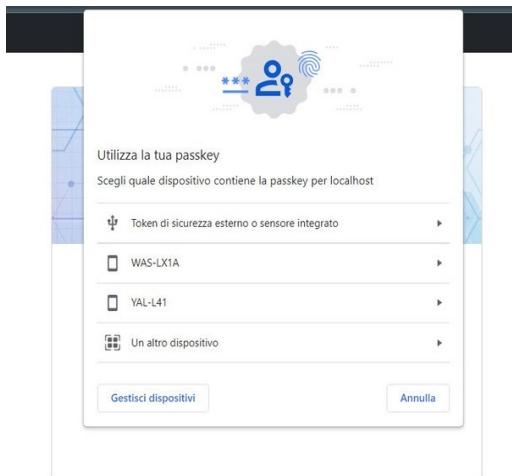
... FIDO 2

The user registration phase provides for the generation of a WebAuthn key pair and an Algorand key pair.



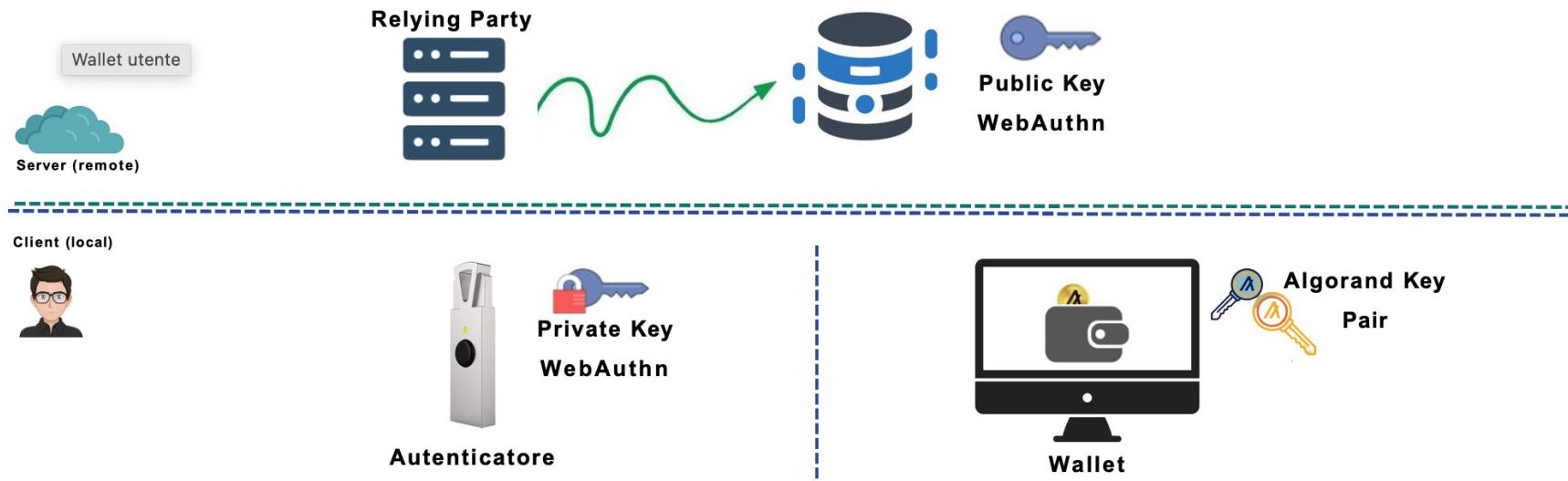
... FIDO 2

The user selects the authenticator to use and gives his authorization to generate credentials.



... FIDO 2

The user selects the authenticator to use and gives his authorization to generate credentials.

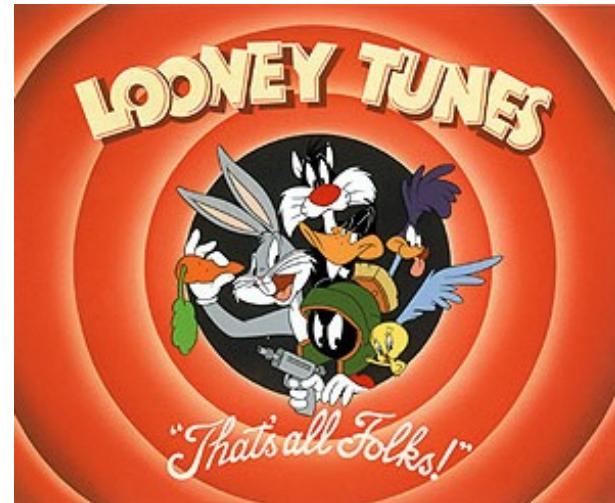


The access to a digital wallet of Algorand is managed by the FIDO2 authentication.

::: FIDO 2

We are working on integrating FIDO2 within the context of a Pod server in Solid so as to have a different way of authentication.

The advantage would be to narrow the attack vectors, minimize user error, and improve user experience. The drawback is related to the cost of purchasing an external authenticator and handling it securely.



**Any question?
Thanks for your attention...**

