Analisi del Problema e Fattibilità

Matteo Casonato
Matteo Midena

14 luglio 2022

Indice

1 Introduzione

Di seguito riportiamo lo scopo del prodotto e del documento.

1.1 Scopo del documento

Lo scopo del documento è quello di riportare le tecnologie utili e legate al prodotto da realizzare, uno studio dei casi d'uso attualmente pensati e proposti nell'ambito e uno studio di una possibile soluzione ai problemi analizzati.

1.2 Scopo del prodotto

L'obiettivo richiesto è la realizzazione di un sistema di agent che permetta di svolgere le principali funzioni legate al rilascio, verifica, revoca di credenziali verificabili in ambito decentralizzato.
Riportiamo un esempio di un caso d'uso reale in cui uno studente richiede il proprio badge all'università per poter accedere poi a tutti i relativi servizi e agevolazioni, in questo caso per poter ottenere uno sconto in mensa.

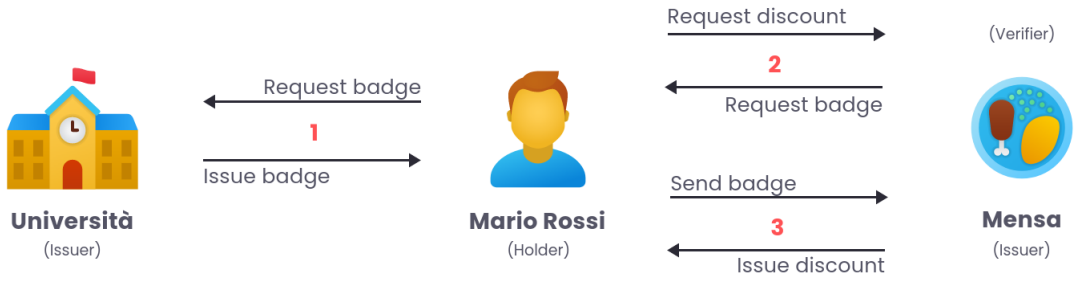


Figura 1: Caso d'uso in cui lo studente richiede prima il badge universitario, e poi lo sconto in mensa

2 Tecnologie Coinvolte

In questa sezione andiamo ad analizzare tutte le possibili tecnologie che verranno coinvolte durante lo sviluppo del progetto.

2.1 Cos'è una blockchain

La blockchain, che letteralmente significa "catena di blocchi", è una struttura dati condivisa e immutabile. Viene definita come un registro digitale le cui voci sono raggruppate a blocchi che vengono concatenati in ordine cronologico e la cui integrità è garantita dall'uso della crittografia. Si basa sul concetto in cui quando il contenuto viene scritto (tramite un processo normato), questo non è più modificabile né eliminabile, pena l'invalidazione dell'intero processo.
Le caratteristiche di base che caratterizzano questa tecnologia sono: digitalizzazione dei dati, decentralizzazione, disintermediazione (non è richiesto nessun intermediario), tracciabilità dei trasferimenti, trasparenza/verificabilità, immutabilità e programmabilità dei trasferimenti. Grazie a tali proprietà, la blockchain si pone come una valida alternativa in termini di sicurezza, affidabilità, trasparenza e costi alle banche dati e ai documenti gestiti in maniera centralizzata da autorità riconosciute e regolamentate (come banche, pubbliche amministrazioni, ecc...).

2.2 Differenze tra blockchain di layer 0, 1, 2

Le blockchain possono essere catalogate in base al tipo di architettura su cui si basano, in particolare possono basarsi su un layer 0, 1 o 2. In base a questo tipo di differenziazione offrono un diverso grado di astrazione e funzionalità. Come funziona per un linguaggio di programmazione più si sale di livello, più ci si allontana dallo "strato fisico".

2.2.1 Layer 0

È il livello più basso, quindi permette di agire direttamente sull'architettura blockchain senza dover mettere mano a layer e protocolli sottostanti. Qui inoltre è possibile creare dApps (decentralized applications), validare schemi di dati, "coniare" criptovalute e altro ancora. Tutto l'hardware, server, nodi e qualsiasi device collegato ai nodi fa parte di questo livello. A questo livello agiscono direttamente gli algoritmi di consenso, come proof-of-work, proof-of-stake, proof-of-activity.

2.2.2 Layer 1

Si occupano principalmente di rendere il protocollo base più scalabile. Si basa su due principali soluzioni:

- **Consensus protocol changes**: primo grande ed evidente cambiamento c'è stato con Ethereum, in cui i progetti hanno iniziato a spostarsi dal protocollo di consenso Proof-of-Work (PoW) a un più veloce e leggero Proof-of-Stake (PoS). Questo è avvenuto su Ethereum con il protocollo Casper.
- **Sharding**: si basa sul semplice concetto di dividere le transazioni in "frammenti" più piccoli (traduzione di "shards") che possono essere processati parallelamente dalla rete.

Pro

- Aggiunge semplicemente un livello superiore all'architettura, senza apportare grandi modifiche;
- Permette di eseguire un maggior numero di operazioni a parità di tempo;

Contro

- Inefficient Consensus Protocol: alcuni protocolli di consenso non sono adatti a gestire grandi traffici di transazioni e conseguentemente queste strutture non sono pronte per un'adozione di massa;
- Excessive Workload: il carico di lavoro su questo livello può diventare eccessivo e provocare grandi rallentamenti (è il caso di Ethereum e le sue conseguenti elevate gas fee).

2.2.3 Layer 2

Sono lo strato più alto di astrazione, e attualmente si pongono l'obiettivo di risolvere gran parte dei problemi presenti nel layer 1. Le soluzioni si dividono in due possibilità:

- **State Channels**: ha come componente principale un canale full-duplex tra due partecipanti che permette di svolgere compiti che tipicamente sono on-blockchain, off-blockchain. Questo permette di diminuire di molto i tempi di attesa visto che viene a mancare la componente terza, come ad esempio le azioni dei miner. Come funziona:
  - Una porzione di blockchain viene ritagliata (viene usata l'espressione "sealed off", ovvero sigillata) tramite multi-firma o sotto smart-contract, previo accordo tra le due parti coinvolte;
  - Quest'ultime possono interagire tra loro senza preoccuparsi di firmare le transazioni eseguite
  - Quando la sequenza di scambi termina, solo lo stato finale viene registrato sulla blockchain.

  La metafora più facile è quella del conto, è come se una persona aprisse un conto con un'altra e poi lo saldasse "a rate". Invece di registrare ogni singola rata sulla rete, viene registrato solo il saldo finale.
  Le soluzioni di questo tipo più comuni sono: la "Bitcoin's Lightning Network" (permette di eseguire tante microtransazioni) e la "Ethereum's Raiden Network" (permette anche di eseguire smart contracts sui canali). Entrambe utilizzano Hashed Timelock Contracts (HTLCs) per creare state channels.

- **Nested blockchains**: si basa sulla soluzione chiamata Plasma, caratterizzata come segue:
  - La rete principale (la main chain) stabilisce le regole di base dell'intero sistema e non prende parte alle operazioni svolte a meno che non debba risolvere dispute;

- Si basa su un sistema in cui la main chain si appoggia a multipli livelli di blockchain. Questi livelli sono connessi tra loro in modo da formare una struttura del tipo padre-figlio. La chain padre delega il lavoro alla chain figlia in grado di eseguire quella particolare istruzione e che ritornerà poi il risultato indietro al padre.
- Questo sistema permette di ridurre il carico della root chain e se costruita e usata correttamente è in grado di aumentare la scalabilità della rete esponenzialmente

Pro

- Evita problemi di disordine per quanto riguarda i compiti di una blockchain;
- Evita di pagare inutili fees su microoperazioni e microtransazioni;

Contro

- Minore controllo sui dati a basso livello;

2.3 Confronto blockchain permissionless e permissioned

Analizziamo in dettaglio le differenze tra le due tipologie di blockchain.

2.3.1 Permissioned

Sono blockchain chiuse o che hanno un layer che si occupa di gestire gli accessi. Questo stesso layer si occupa anche di gestire le azioni che i partecipanti della rete sono autorizzati a svolgere.
Il concetto chiave è che quindi un utente può accedere alla rete, leggere e scrivere informazioni in blokchain solo se gli viene dato l'accesso. Questo tipo di rete quindi supporta anche un certo grado di personalizzazione, ad esempio la verifica dell'identità di nuovi utenti può essere fatta dalle persone presenti in rete, invece che esclusivamente dal proprietario della rete. Ed è possibile ovviamente personalizzare le attività possibili in rete.
Proprio per questo motivo le permissioned blockchain possono essere centralizzate o parzialmente decentralizzate.
Un esempio di permissioned blockchain è Ripple.

Consensus model L'algoritmo di consenso usato in queste reti solitamente è differente da quello utilizzato per la controparte permissionless, i principali sono:

- **Practical Byzantine Fault Tolerance (PBFT) consensus:** è un algoritmo voting-based. In questo modello, la sicurezza della rete è garantita finchè la percentuale minima richiesta di nodi si comporta in modo onesto e funziona correttamente.
- **Federated consensus:** si basa su un numero limitato di firmatari fidati per ogni nodo della blockchain. Utilizzano un unico generatore di blocchi che riceve le transazioni e le filtra di conseguenza.
- **Round-robin consensus:** in questo modello i nodi sono selezionati in modo pseudo-randomico per creare blocchi, ogni nodo quindi deve aspettare un certo numero di cicli prima di essere scelto nuovamente per aggiungere un nuovo blocco.

Pro

- alto livello di privacy e sicurezza;
- flessibile: può essere parzialmente decentralizzata o completamente centralizzata in base ai casi d'uso;
- altamente personalizzabile: come descritto precedentemente, può soddisfare configurazioni e integrazioni basate sui bisogni dell'organizzazione;
- la sua grandezza limitata rende queste reti scalabili e performanti

Contro

- grande mancanza di trasparenza, questo si traduce in possibili rischi di corruzione (il controllo è nelle mani di un gruppo privato);
- la sicurezza della rete dipende direttamente dell'integrità dei suoi membri, un'alterazione dei dati per un loro tornaconto è sempre possibile;
- sono regolamentate e censurate, non c'è una libertà completa.

2.3.2 Permissionless

È il modello più diffuso tra le principali criptovalute (come Bitcoin e Ethereum). Come si evince dal suo nome, queste reti permettono l'accesso a chiunque, sono quindi decentralizzate e pubbliche. La parola "permissionless" implica la mancanza di vincoli o censura sulle possibili azioni.
Tecnicamente, come da protocollo, ognuno può usare la rete e farci quello che vuole. Le permissionless blockchain si avvicinano al concetto originale di blockchain sviluppato da Satoshi Nakamoto.
Questa accessibilità al pubblico si paga in termini di trade-off sulla velocità, spesso queste reti infatti risultano più lente della controparte permissioned, che ha molti meno membri.
Le transazioni, che contengono le informazioni salvate in blockchain, sono validate direttamente dal pubblico. Il principale meccanismo di consenso uso è il proof-of-work (POW) e proof-of-stake (POS).

Pro

- alta trasparenza;
- decentralizzata, le informazioni non sono in repository centralizzate e questo rende i dati sicuri, accessibili e affidabili per tutti (sono per questo considerati praticamente non hackerabili);
- sicurezza, un attaccante dovrebbe attaccare con successo almeno il 51

Contro

- grande richiesta di energia e potenza di calcolo per raggiungere in consenso;
- per sua natura le informazioni sono pubbliche, quindi presenta problemi di privacy;
- l'anonimicità della rete non offre possibilità di negare l'accesso a utenti malevoli o malintenzionati.

2.4 Ethereum

Ethereum è una blockchain **permissionless**, ossia aperta a chiunque voglia interagire con essa: il compromesso è l'introduzione delle **fee**, ossia una "tassa" da pagare ogni volta che si vuole modificare lo stato dell'**Ethereum Virtual Machine** (EVM), per mitigare i problemi che introduce il fattore permissionless (ad esempio lo spam di transazioni). In caso di sola lettura non occorre pagare alcuna fee.
Chiunque quindi può visionare cosa accade nella blockchain, e soprattutto può utilizzarla. Per farlo, basterà generare un **wallet** (ossia un indirizzo nella blockchain), trasferirci alcuni fondi dall'esterno per poter pagare le fee, e iniziare ad interagire con le applicazioni decentralizzate che sono già sviluppate, oppure semplicemente trasferire fondi ad altri wallet.
Per poter effettivamente utilizzare delle applicazioni su Ethereum vengono utilizzati gli **Smart Contracts**.

2.5 Hyperledger

Hyperledger Foundation è un'organizzazione **senza scopo di lucro** che riunisce tutte le risorse e le infrastrutture necessarie per garantire ecosistemi fiorenti e stabili attorno a progetti blockchain di software open source.
Il personale della Hyperledger Foundation fa parte di un team più ampio della **Linux Foundation** che vanta anni di esperienza nella fornitura di servizi di gestione di programmi per progetti open source.

2.6 Hyperledger Besu

Hyperledger Besu è un client Ethereum progettato per essere adatto alle imprese per i casi di utilizzo di **reti pubbliche e private permissioned**, che richiedono un'elaborazione delle transazioni sicura e ad alte prestazioni.
La blockchain Besu quindi è **EVM compatibile**: dal punto di vista degli sviluppatori e degli utenti, l'interazione con essa sarà molto simile all'interazione con Ethereum. Pone però particolare attenzione sulle funzionalità di **privacy e permissioning**, infatti solamente chi è autorizzato (ossia chi possiede un nodo connesso) può interagire con il sistema, e questa è la differenza sostanziale con Ethereum.
La privacy viene considerata sia a livello esterno alla rete, sia a livello interno: tramite le **transazioni private** non tutti i nodi possono accedere a particolari informazioni, e i nodi che vogliono usufruire delle transazioni private devono avere un nodo **Tessera** associato, che si occuperà della parte crittografica.

2.6.1 Permissioning

Una permissioned network è una rete che gestisce i permessi su un insieme di nodi e account, permettendo a solo quelli autorizzati di accedere alla rete. In particolare quello che si può fare relativamente ai permessi sugli account è:

- richiedere dettagli sull'identità
- sospendere gli account
- escludere contratti invalidi in una denylist
- limitare le azioni che un account può compiere

La specifica dei permessi può essere specificata in due modi:

Localmente: si lavora a livello del nodo, ogni nodo della rete ha un file di configurazione dei permessi. Questi tipi di permessi hanno effetto sul nodo in questione ma non sul resto della rete. Questo significa che non è necessario un coordinamento con il resto della rete e si può agire direttamente per la protezione del nodo.

Onchain: si attua attraverso uno smart contract sulla rete. Specificare i permessi onchain implica che tutti i nodi possono leggere e aggiornare la configurazione dei permessi. Questo tipo di gestione richiede coordinazione per aggiornare le regole e la rete potrebbe non attuarle immediatamente in quanto, ad esempio, lo smart contract potrebbe forzare un minimo numero di voti prima di attuare la modifica.

2.6.2 Privacy

Si intende l'abilità di mantenere le transazioni private tra i partecipanti coinvolti, gli altri partecipanti non devono poter accedere al contenuto di quest'ultime o alla lista dei partecipanti.

2.6.3 Tessera

Besu usa un private transaction manager, Tessera, un progetto open-source sotto Apache 2.0 scritto in Java, che ha il compito principale di gestire la privacy sui client Ethereum che hanno privacy-enabled (come Hyperledger Besu appunto). Le principali funzioni di Tessera sono:

- generare e mantenere le coppie chiavi private/pubbliche
- gestire e scoprire tutti i nodi nella rete
- fornisce un'interfaccia API per la comunicazione tra i nodi Tessera e un API per la comunicazione con i client Ethereum.

Ogni nodo Besu che riceve e invia transazioni private necessita un associato nodo Tessera. Quindi le transazioni passano dal nodo Besu all'associato nodo Tessera. Quest'ultimo cripta la transazione privata e la distribuisce point-to-point ai nodi Tessera partecipanti alla transazione.
Di default, ogni partecipante a una private network utilizza il suo nodo Besu e Tessera, e l'uso di un sistema multi-tenancy permette a più di un partecipante di usare lo stesso nodo Besu e Tessera.

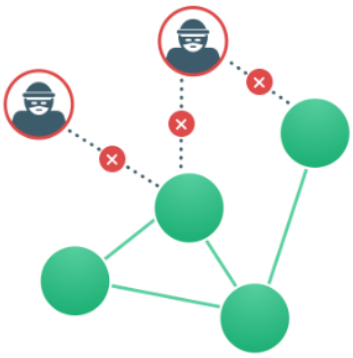


Figura 2: Chi non ha accesso alla rete non può interagirci in nessun modo

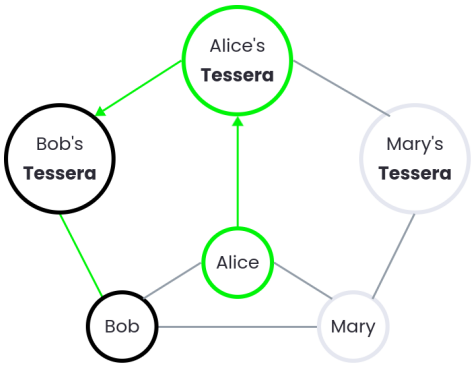


Figura 3: Alice manda una transazione privata a Bob, e per Mary non sarà visibile

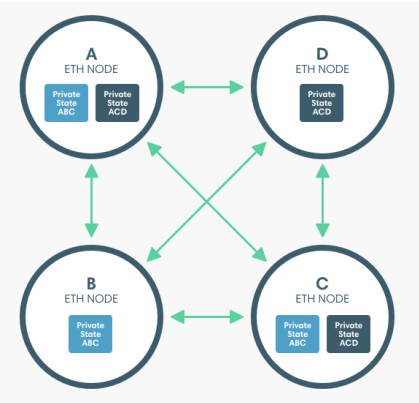


Figura 4: Visibilità ristretta di due Privacy Groups (azzurro e blu)

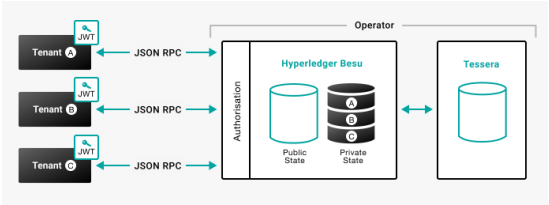


Figura 5: L'amministratore della coppia di nodi Besu e Tessera può dare l'accesso ad altri Tenant, ossia utenti.

2.7 Hyperledger Fabric

Hyperledger Fabric è una **piattaforma ledger distribuita** (DLT), di livello aziendale, sperimentata e aperta. Prevede controlli avanzati sulla privacy così che solo i dati che si desidera condividere vengano condivisi **tra i partecipanti** alla rete noti come "autorizzati".
Offre un'architettura **modulare** che rende disponibili componenti (meccanismo di consenso, servizi per l'adesione e la gestione dei membri della blockchain ecc.) che, con la logica del plug and play, possono essere attivati nell'ambito di una blockchain. Si può affermare che è molto **simile a Besu** (anch'essa rete permissioned e privacy-oriented), con la grossa differenza che **non è EVM** compatibile, quindi gli smart contract verranno scritti in linguaggi come Java e Go, invece di Solidity.

2.8 EBSI

EBSI (European Blockchain Services Infrastructure) è un'infrastruttura di servizi che permette di gestire le **credenziali** di identità e di istruzione dei cittadini europei. Questi casi d'uso mirano a facilitare la mobilità di studenti, giovani professionisti e imprenditori, nonché a **garantire e verificare l'autenticità** delle informazioni digitali in diversi settori.
La sua struttura è piuttosto complessa, composta da due layer fondamentali: il

layer inferiore è cositutito dalla **blockchain** di EBSI, che custodisce tutte le credenziali e gli smart contract con i quali si può interagire con esse, mentre il **layer superiore** implementa un'architettura a **microservizi** ed **API**, che si interfacciano con la blockchain.
In pratica, per dialogare con la blockchain di EBSI (se non si dispone di un nodo) occorre interfacciarsi tramite l'architettura a microservizi e API messe a disposizione. Le blockchain utilizzate da EBSI sono Hyperledger Fabric e Hyperledger Besu.

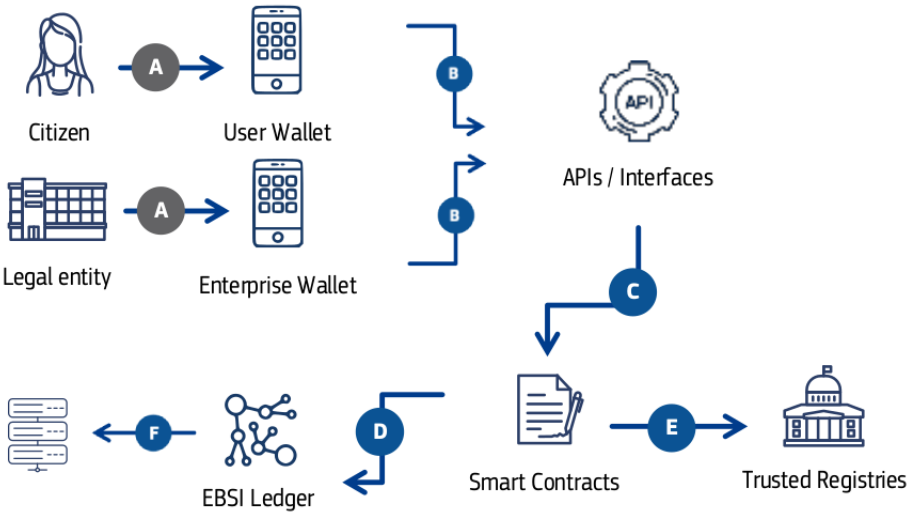


Figura 6: Il flow dell'interazione con EBSI

Sopra è stato riportato un esempio relativo al flow dell'iterazione di una normale operazione con EBSI. Come si può vedere sia gli utenti che le organizzazioni interfacciano, utilizzando un wallet, con le API, questo significa che nessuno ha davvero idea di cosa facciano effettivamente gli **smart contract** di EBSI poichè **non sono pubblici** e l'unica cosa su cui si può fare riferimento è la relativa documentazione fornita. Questo è un esempio di quello che significa utilizzare una blockchain permissioned, il **controllo** è completamente **detenuto da EBSI** stessa.
Saranno le API a questo punto che interagiranno direttamente con gli smart contrat, in cui sono presenti tutte le informazioni relative ai trusted registries, DID e altro.

2.8.1 Architettura

Andando più nel dettaglio sull'architettura di EBSI, questa è divisa su multipli layer, ed è costruita come un'infrastruttura a micro-servizi. I livelli sono progrettati in modo da soddisfare nel modo più efficiente possibile anche i requisiti attualmente sconosciuti.

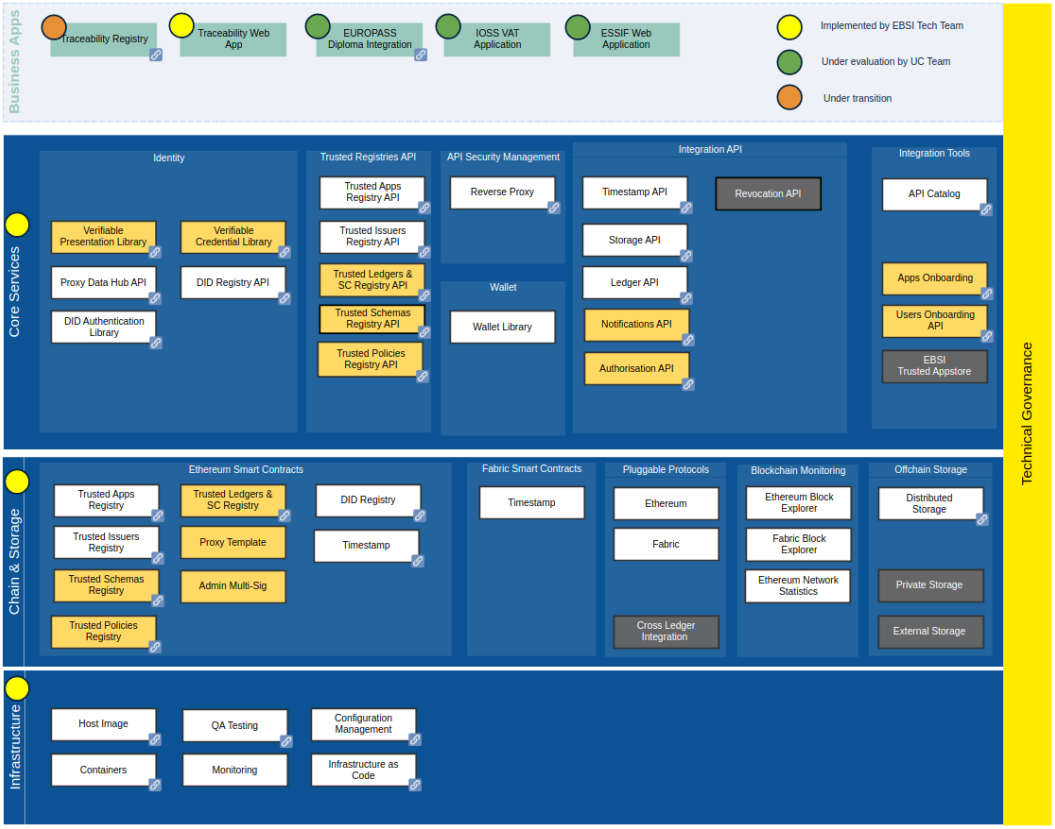
Core Services rappresenta l'interfaccia che la piattaforma EBSI espone a livello di use case. Quest'ultimo fornisce due grandi benefici:

- migliorare la velocità di sviluppo dei casi d'uso fornendo un'interfaccia consistente e granulare;
- maggior sicurezza della rete creando una separazione tra le applicazioni esterne e le infrastrutture dei sistemi.

Chain e Storage Layer comprende i protocolli blockchain correntemente supportati da EBSI e aggiunge i dati salvati off-chain. I suoi principali sottolayer sono appunto off-chain storage e smart contract.

Infrastructure Layer rappresenta tutti gli elementi infrastrutturali richiesti per costruire un nodo EBSI. Raggruppa quindi tutti gli strumenti, metodi e servizi necessari per sviluppare, costruire, testare, caricare relativi al servizi dell'EBSI node image.

Figura 7: Diagramma dell'architettura di EBSI V2



2.8.2 Smart Contract

In questa sezione sono riportate le informazioni ricavate dagli smart contract caricati e con i quali è possibile interfacciarsi su EBSI.
Tutti i contratti seguono lo standard EBSI, ovvero presentano le proprietà own-able, upgrade-able, implementazione diamond standard e proxy.

DID Registry Smart Contract DID registry è un servizio principale della suite EBSI per salvare DID e DID Document e permette le seguenti interazioni:

1. inserire un DID/DID Document
2. aggiornare un DID/DID Document
3. revocare un DID/DID Controlling key
4. risolvere un DID (e ottenere un DID Document)
5. risolvere una specifica versione di un DID Document

Il registro è implementato come uno smart contract Ethereum ed è deployato su EBSI Besu Ledger.
È pensato per supportare il salvataggio di DID Document e link a questi. Dalla documentazione ufficiale di EBSI si ricava inoltre che per le limitazioni degli smart contract solo specifici tipi di firme possono essere validate, in EBSI V2.0 la verifica dei DID Document è fatta **solo** a livello API, questo significa che la computazione avviene off-chain. Inoltre i DID method che si affidano a specifici protocolli che non sono supportati da EBSI non sono supportati.
Ci sono diversi approcci all'ancoraggio di DID e DID Document a seconda del caso d'uso:

- DID Document salvato nello smart contract (parzialmente o totalmente)
- DID Document salvato off-chain (parzialmente o totalmente), EBSI fornisce un Cassandra-DB decentralizzato. Si va a salvare solo un puntatore a dove è salvato il DID document off-chain.
- DID Document può essere salvato localmente. L'ultimo hash DID viene ancorato al ledger (non è specificato chiaramente come avvenga la risoluzione del DID).

Per lo Use Case di ESSIF (European Self Sovereing Identity Framework) hanno scelto la prima opzione e salvare DID, DID Document e Public Controlling Key direttamente sullo smart contract presente sul ledger.

Timestamp Smart Contract Sostanzialmente registra in blockchain quello che succede salvando un timestamp che verrà poi usato per eseguire delle operazioni di hashing.

Trusted Apps Registry Smart Contract (TAR) Offre un servizio di arcoraggio fidato per le API dei principali servizi di EBSI e applicazioni fidate esterne (presenti all'interno di una lista). Questo contratto va a salvare nome, DID, accessi, policies e altre claims. Viene utilizzato da tutti i servizi EBSI per la gestione degli accessi e per la sua natura immutabile permette di capire se un'applicazione è fidata e autorizzata per eseguire date operazioni.

Trusted Issuers Registry Smart Contract (TIR) Offre un servizio di ancoraggio fidato per Issuer sicuri non collegati a nessun particolare dominio business. Contiene quindi una lista di issuer e per ognuno di questi una lista di oggetti generici (eID, VC, Verifiable Attestation).
Usando questo servizio, è possibile verificare se il DID dell'issuer è presente all'interno della Trusted Issuers Smart Contract list e identificare il relativo ambito di emissione (ovvero quali object).

Trusted Ledgers and Smart Contracts Registry (TLSCR) Offre un servizio di ancoraggio fidato per:

- trusted smart contract deployati sui trusted ledger(s) di EBSI
- riferimento ai futuri nodi che verranno aggiunti in futuro per estendere la piattaforma EBSI.

Gli smart contract possono essere indirizzati anche per il relativo service name, quindi anche in caso di un aggiornamento, con relativa modifica dell'indirizzo dello smart contract, il riferimento rimarrà invariato.

Trusted Policies Registry Smart Contract Questo contratto detiene un set di variabili che contengono le informazioni relativamente a regole di accessibilità. Le variabili sono principalmente di due tipi, si riferiscono a regole o contengono dati relativi all'utente (si è deciso di non scendere ulteriormente nei dettagli di quest'ultime).
Il principale scopo di questo contratto è di avere un single point of truth sia per le API che per gli smart contract, dove gli utenti (in base agli attributi) hanno un accesso definito alle risorse di EBSI.

2.9 walt.id

La suite di software di walt.id, progetto open source sviluppato da un team situato in Europa, consiste in un insieme di kit che ci permette di aggiungere funzionalità di SSI al nostro prodotto. Nello specifico, sfruttando l'**SSI Kit**, potremo generare **chiavi** crittografate, registrare e risolvere **DIDs**, creare, emettere presentare e verificare **VCs** e molto altro.
Questo kit può essere utilizzato tramite CLI (command line interface) o via REST API. Al momento il kit supporta i **metodi** `did:key`, `did:web` e `did:ebsi` (quest'ultimo ci permette di dialogare con la blockchain di EBSI).
Dato che di default dialoga con altre API (es: API di EBSI), se si volesse registrare ad esempio un DID in una mainnet non ancora registrata, come Sovrin, occorrerebbe **aggiungere il metodo** `did:sov` alla lista dei metodi supportati, implementando anche il dialogo con le API di Sovrin. Per fare ciò occorrerebbe sviluppare un **nuovo modulo** ed integrarlo all'SSI Kit, sviluppando qualcosa di simile a quello che è stato fatto per EBSI, anche se non si sa quale sia la complessità.
Il metodo `did:ebsi` è ridirezionabile alla propria blockchain privata locale, ma per funzionare occorrerebbe caricare gli **smart contract di EBSI** nella blockchain, e quindi il kit non comunicherebbe con i nostri (nuovi) smart contract, perchè lo farebbe solamente con quelli **originali** di EBSI.
La soluzione, come descritto meglio nella sezione "Discussione possibili soluzioni", sarebbe quella di effettuare testing direttamente nella mainnet di EBSI, e di effettuare il deploy dei nostri smart contracts su reti alternative.

3 Casi d'uso concreti

Lo scopo di questa sezione è la descrizione e l'approfondimento di tutti i casi d'uso individuati in riferimento agli obiettivi da raggiungere.

3.1 Poste italiane Loyalty Programs

Di seguito descriviamo il caso d'uso Loyalty Programs portato da Poste Italiane.

3.1.1 Come funziona

L'idea è quella di creare un sistema di ricompensa il cui centro è il cliente. La piattaforma di Poste Italiane è basata su blockchain ed è completamente decentralizzata e ogni partner utilizza il proprio nodo (in caso fornito da Poste Italiane), in modo da creare un ecosistema equo con responsabilità e interessi distribuiti.
Si vuole sfruttare l'immediatezza della blockchain in modo che il rilascio dei punti (certificati) sia praticamente istantaneo e real-time.
L'utente accederà ai propri dati da un'applicazione mobile di Poste Italiane. Quest'ultima permetterà di generare un nuovo wallet. Ogni volta che vuole utilizzare uno dei suoi premi, l'utente può creare un voucher che combina i punti ottenuti dai vari programmi e verrà riscattato dal sistema. I punti sono trasferiti sul wallet dell'utente tramite l'uso di un QR code generato dall'applicazione (rappresenta l'indirizzo del wallet), dove i partner e i commercianti possono trasferire i punti.
Questo sistema fa uso di 3 componenti:

- Wallet digitali: unico punto di accesso a tutti i programmi presenti sulla rete
- Tokens: rappresentazione digitale dei punti o voucher (rappresentazione di un valore programmabile)
- Smart Contracts: permettono di stabilire delle regole per gestire i token e il funzionamento dell'intera rete

I vantaggi offerti da questo tipo di soluzione sono principalmente quelli garantiti dall'uso della tecnologia blockchain, ovvero:

- decentralizzazione
- tracciabilitià
- disintermediazione
- trasparenza
- immutabilità

3.1.2 Uso di Hyperledger Besu

L'utilizzo di Hyperledger Besu come blockchain, permette ai partner, di mantenere l'indipendenza nel gestire i propri programmi riducendo la complessità di integrazione, mentre per i clienti, grazie al possesso di un wallet mobile, migliora l'esperienza di utilizzo poichè possiedono un singolo punto di accesso a tutti i programmi forniti.

Loyalty Point e Vouchers Ogni loyalty point è sostanzialmente un ERC20 token. La piattaforma prevede inoltre la possibilità di convertire una certa quantità di questi punti in un ERC721 token, che rappresenta un voucher. Ogni partener può definire il proprio tasso e regole di conversione e scriverlo all'interno della loro istanza del token ERC20.
Gli smart contract LOYALTY coordinano la creazione dei token, il riscatto e la liquidazione dei voucher. La creazione di un voucher, dopo l'autorizzazione dello smart contract LOYALTY, è gestita da uno smart contract ERC721.

L'uso di Hyperledger Besu offre i benifici di connettere Hyperledger e Ethereum.

3.2 Casi d'uso di EBSI

Riportiamo i casi d'uso elencati nel sito ufficiale di EBSI.

3.2.1 Identity

È il principale caso d'uso, permette ad EBSI di implementare la verifica di credenziali identificative permettendo agli utenti di creare e controllare la propria identità digitale (quindi identificazione, autenticazione e altri informazioni legate).

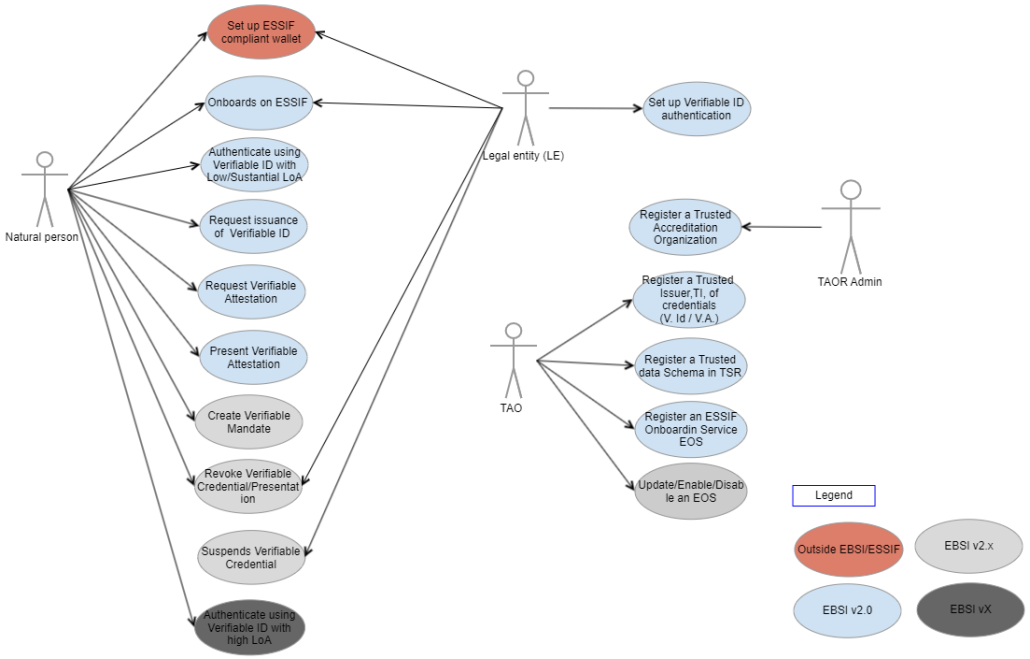


Figura 8: Caso d'uso riportato direttamente da EBSI

3.2.2 Diploma

Lo scopo è di consentire a EBSI di verificare le crendeziali scolastiche, quindi ad esempio il diploma rilasciato da uno stato membro A può essere verificato da un'università o terze parti, situati ad esempio in uno stato membro B.

3.2.3 Social security

Questo caso d'uso adetterà di EBSI di implementare la verifica cross-border della **copertura previdenziale dei lavoratori distaccati**, ossia la verifica del docu-

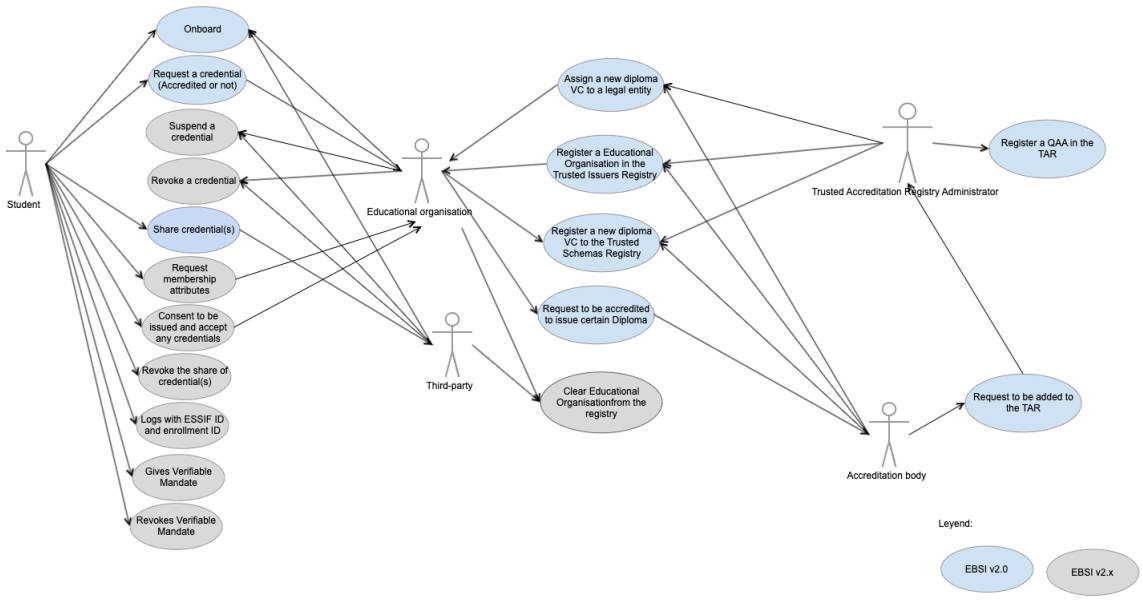


Figura 9: Caso d'uso riportato direttamente da EBSI

mento PDA-1. Ciò significa che un'**istituzione** competente per la sicurezza sociale in uno Stato membro **rilascia** il documento PDA-1 come attestazione verificabile e un **ispettore** in un altro Stato membro lo **verifica**.

3.2.4 Document Traceability

Questo caso d'uso mira a sfruttare la blockchain per creare **audit trail** digitali affidabili, automatizzare i controlli di conformità (ad esempio nei processi sensibili ai tempi) e dimostrare l'**integrità** dei dati. L'intento è anche quello di fornire funzionalità generiche di **registrazione** e **tracciabilità** per altri Use Case EBSI.

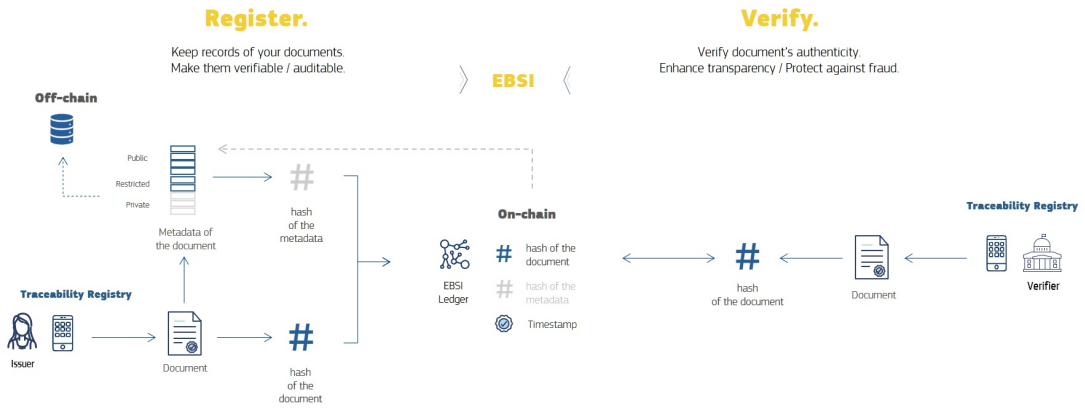


Figura 10: Concetti chiave del caso d'uso

3.3 Casi d'uso di un contesto universitario

Poiché si è discusso molto di alcuni esempi di adozione di quanto riportato nel documento in ambito universitario, citiamo qui quelli di maggior importanza e un'analisi più approfondita di due di questi.

3.3.1 UC1 - Emissione Esame

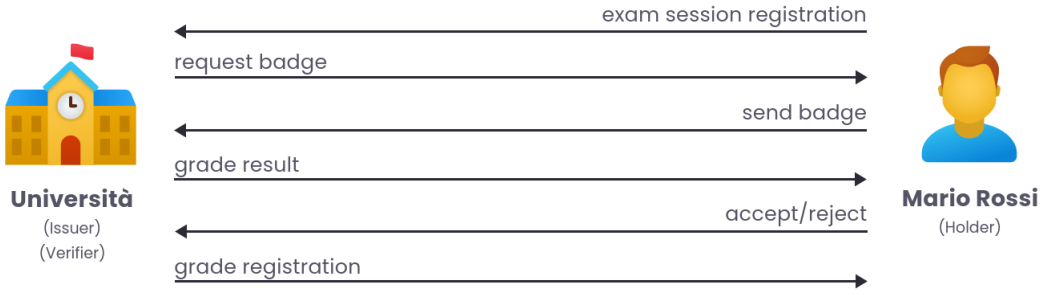


Figura 11: Flow relativo al caso d'uso

In questo caso d'uso mostriamo la sequenza di azioni che intercorrono dalla registrazione, da parte di uno studente, ad un appello d'esame fino alla registrazione del voto:

1. Lo studente si registra all'appello d'esame
2. l'università richiede all'utente il badge universitario, che attesta il fatto che lo studente sia effettivamente iscritto
3. lo studente invia il suo badge per essere verificato
4. lo studente si presenta all'appello e svolge l'esame
5. successivamente alla correzione, l'università invia il risultato allo studente
6. lo studente decide se accettare o rifiutare il voto
7. l'università, se lo studente accetta, rilascia ufficialmente il voto dell'esame

3.3.2 UC2 - Emissione Diploma

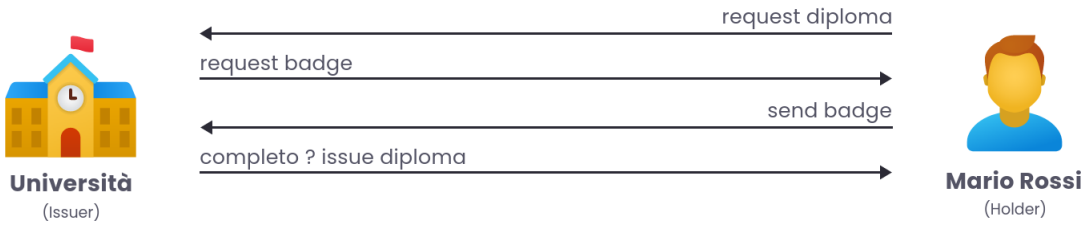


Figura 12: Flow relativo al caso d'uso

3.3.3 UC3 - Richiesta borsa di studio (analizzato)

La situazione raffigurata è quella di uno studente che effettua una richiesta per ottenere una borsa di studio. Lo scenario principale di questo caso d'uso è sviluppato sui seguenti punti:

1. lo studente effettua la richiesta per ricevere la borsa di studio

Figura 13: Flow relativo al caso d'uso

2. l'università richiede all'utente il badge universitario, che attesta il fatto che lo studente sia effettivamente iscritto
3. lo studente invia il suo badge per essere verificato
4. a questo punto l'università richiede all'utente di dichiarare se la sua media è superiore alla soglia richiesta
5. attraverso una ZKP l'utente risponde all'università
6. nel caso in cui sia tutto positivo l'università provvede a rilasciare la borsa di studio.

3.3.4 UC4 - Accesso risorse con sconto via VC

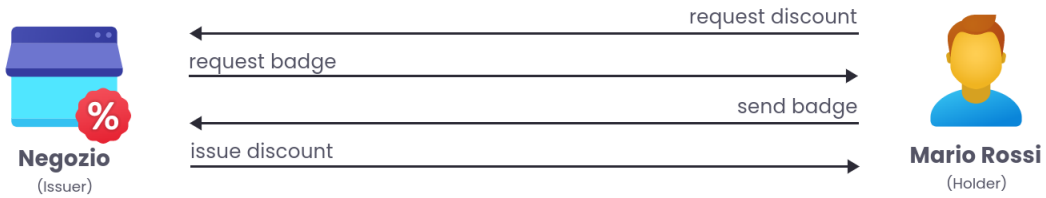


Figura 14: Flow relativo al caso d'uso

3.3.5 UC5 - Accesso risorse con sconto via ZKP

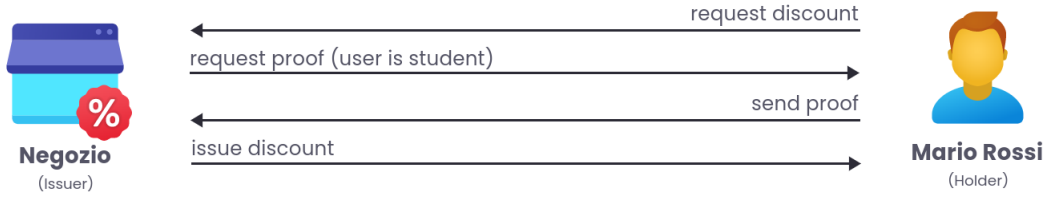


Figura 15: Flow relativo al caso d'uso

4 Discussione possibili soluzioni

Sulla base di quanto riportato, il prodotto dovrà gestire la parte degli Agent (Issuer, Verifier e Revoker), e quindi di interazioni con le varie credenziali. Questo può avvenire **off-chain**, e quindi in un layer superiore, oppure **on-chain**, ossia direttamente in blockchain.

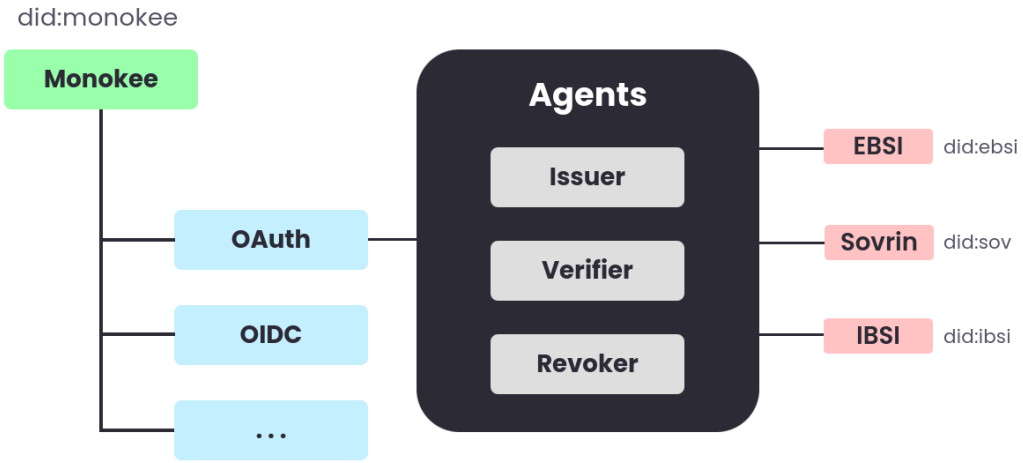


Figura 16: Gli agent si pongono tra i protocolli di rete e le blockchain che contengono i DID

4.1 Soluzione off-chain

Come spiegato precedentemente, tramite il kit SSI di walt.id è possibile generare **chiavi** crittografate, registrare e risolvere **DIDs**, creare, emettere presentare e verificare **VCs**. E' quindi un coltellino svizzero della SSI, e permette di sviluppare Agent off-chain, grazie ai protocolli di rete (es: OIDC) con i quali è già integrato.

L'unico problema è il fatto che per ora gli unici ecosistemi supportati sono EBSI/ESSIF, Gaia-X e Velocity Network. Se si volessero considerare altre reti che fungano da Trusted Data Registry, occorrerebbe creare degli adattatori per conto della suite, oppure considerare altre librerie (ad esempio MATTR, che supporta la rete ION, una rete che si appoggia a Bitcoin).
Di seguito elenchiamo le librerie analizzate, tutte supportano i metodi *did:key* e *did:web*.

SSI Suite	Metodi supportati	Rete associata
walt.id	did:ebsi	EBSI
mattr.global	did:ion	ION
veramo.io	did:ethr	Ethereum

In supporto a queste librerie, si potrebbero integrare i software Universal Resolver e Universal Registrar, sviluppati dal DIF, che supportano diversi metodi e permettono di risolvere o registrare un DID.

4.2 Soluzione on-chain

Per poter spostare tutta la computazione on-chain è necessario affidarsi a degli **smart contract**. In particolare sarà necessario svilupparne uno per ogni tipologia di **Agent**, quindi almeno un issuer, verifier e revoker.
Il principale problema di utilizzare una blockchain per questo tipo di operazioni è legato alla **trasparenza** delle transazioni e dei dati salvati. In particolare la scelta di quest'ultima presenta una grande distinzione:

4.2.1 Blockchain Permissionless

Se scegliessimo di usare una blockchain permissionless (come ad esempio Ethereum), tutte le informazioni presenti in blockchain sarebbero accessibili **da qualsiasi utente**. Questo significa che tutti i dati scritti, ad esempio da un *issuer*, sono consultabili da chiunque, portando ovviamente a **problemi di privacy**.
Una possibile soluzione potrebbe essere quella di **criptare** le informazioni sfruttando le coppie di chiavi [*public key, private key*], in modo tale da garantire (in base al tipo di combinazione) autenticità e riservatezza. Qui sotto è riportato un possibile scenario: In questo caso come si può vedere è l'**utente** ad avere il controllo, la

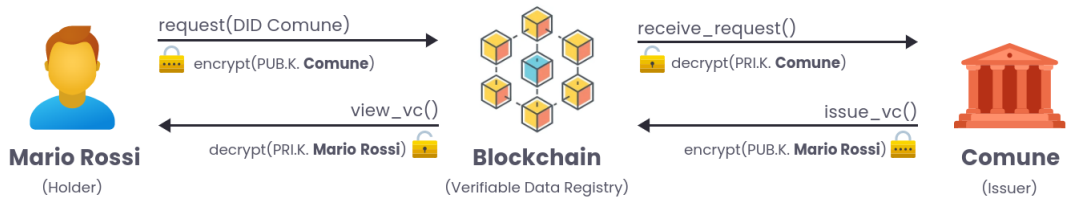


Figura 17: Scenario di richiesta e rilascio della carta d'identità on-chain

richiesta proviene direttamente da lui e l'*issuer* avrà solo il compito di rilasciare la credenziale verificabile.
Dallo schema si può notare che, in questo modo, il **comune** non ha più la possibilità di vedere le credenziali rilasciate, poiché solo gli utenti destinatari potranno decriptarle. Una possibile soluzione al problema potrebbe essere quella di creare una **ridondanza** dei dati, criptandoli solo con la chiave pubblica del comune (in modo che solo quest'ultimo ne abbia l'accesso).

Vantaggi:

- soluzione in cui è l'utente ad essere al centro;
- non è necessario creare una private network e gestirla;

Svantaggi:

- necessità di provvedere un meccanismo di criptazione dei dati;

4.2.2 Blockchain Permissioned

Usando invece una blockchain permissioned (come ad esempio Hyperledger Besu), le informazioni presenti in blockchain sono accessibili solo ai nodi e agli account autorizzati. Questo significa che l'utente non sarà più al centro, saranno gli agent a fare da intermediari.

Vantaggi:

- non è necessario provvedere un meccanismo di criptazione dei dati, solo chi vogliamo vi avrà accesso;

Svantaggi:

- non si avrà più la centralità dell'utente, questo comporta che sarà sempre l'agent a provvedere all'inizio dell'operazione;

4.3 Ambiente di sviluppo

In base alla soluzione scelta si avrà un diverso ambiente di testing, riportiamo di seguito le diverse possibilità.

4.3.1 Off-chain

Per testare la suite di walt.id (o altre suite di generazione/verifica/revoca DIDs) la strada più semplice ed efficace è quella di generare/verificare/revocare DID di test nelle reti a disposizione (ad esempio EBSI). L'alternativa sarebbe quella di utilizzare delle testnet pubbliche, ma per molte reti permissioned (come EBSI) non esistono, oppure di creare una private network per ogni blockchain con le quali si vuole interagire, ma sarebbe molto oneroso in termini di risorse e tempo.

4.3.2 On-chain su blockchain permissioneless

L'ambiente di testing è già pronto, sarà sufficiente scegliere una testnet di Ethereum ed eseguire i nostri test direttamente deployando lì i nostri contratti.

4.3.3 On-chain su blockchain permissioned

In questo caso non è disponibile una testnet su cui eseguire i nostri test, probabilmente proprio perchè si rischierebbe di avere una rete in cui i permessi dovrebbero essere concessi praticamente a chiunque, andando contro quella che è una rete che gestisce anche gli accessi. Il risultato ottenuto sarebbe quello di avere una normale testnet Ethereum pubblica.
La soluzione è quindi quella di andare a creare una rete privata ed effettuare direttamente là il testing.

5 Link utili

5.1 Ethereum

Intro to Ethereum: Link
EVM: Link
Smart Contracts: Link

5.2 Hyperledger Besu

Privacy: Link
Tessera: Link
Poste Case Study: Link

5.3 Ebsi

Home: Link
EBSI Verifiable Credentials Playbook: Link
API docs: Link
FAQs: Link
Smart Contract: Link

5.4 Verifiable Credential

What is a verifiable credential: Link

5.5 Decentralized Identifier

DID core (w3.org): Link

5.6 SSI Kits

walt.id docs Link
MATTR docs Link
Veramo docs Link