VerifiedMovies

DOCUMENTO DI DESCRIZIONE AD ALTO LIVELLO DELLA DAPP REALIZZATA

Sommario

Introduzione

Il seguente documento fornisce una panoramica del progetto di stage e dello sviluppo di un'applicazione decentralizzata (dApp) chiamata VerifiedMovies. Lo stage mirava a esplorare e implementare le tecnologie blockchain, i protocolli di identità digitale e i concetti di identità auto-sovrana nel contesto di un'applicazione pratica.

Gli obiettivi principali dello stage erano molteplici:

- Acquisire esperienza pratica nello sviluppo di applicazioni decentralizzate utilizzando la tecnologia blockchain.
- Esplorare il potenziale dei protocolli di identità digitale e dell'identità sovrana per migliorare la privacy e la sicurezza dei dati.
- Implementare una dApp che sfrutti i concetti di blockchain e identità decentralizzata per un caso d'uso specifico.

Durante lo stage, l'attenzione si è concentrata sullo sviluppo di VerifiedMovies, una dApp progettata per fornire una piattaforma sicura e trasparente per la verifica e l'autenticazione dei film. Questo documento fornirà una descrizione dettagliata della dApp, delle tecnologie utilizzate, delle sfide incontrate e delle soluzioni implementate.

Il progetto, infatti, descrive l'implementazione di una piattaforma di un cinema basata sugli standard W3C Decentralized Identifier (DID) e Verifiable Credentials (VC), usando Zero Knowledge Proof (ZKP) per la verifica sicura delle informazioni trasmesse senza divulgare alcuna informazione sullo stesso.

Lavorando su VerifiedMovies, lo stage ha avuto l'obiettivo di promuovere una comprensione più approfondita delle tecnologie blockchain, dei protocolli di identità digitale e delle potenziali applicazioni dei sistemi decentralizzati. Il progetto ha offerto l'opportunità di esplorare l'intersezione tra blockchain, identità digitale e industria dell'intrattenimento, con l'obiettivo di contribuire al progresso di piattaforme sicure e affidabili.

Le sezioni successive approfondiranno i dettagli della dApp, le tecnologie utilizzate, i problemi affrontati durante lo sviluppo, le sfide incontrate e i potenziali scenari futuri per le implementazioni blockchain e le applicazioni decentralizzate. Il documento si concluderà con una sintesi delle attività svolte durante lo stage e delle competenze acquisite durante il processo.

Basato sullo studio autonomo delle tecnologie blockchain e della loro base, la piattaforma offre un insieme semplificato di funzioni connesse al sito di un cinema, comprendenti una registrazione, un accesso e una gestione del proprio profilo, permettendo l'accesso sicuro ai contenuti qualora univocamente dimostrato. Il contesto della blockchain serve a garantire la sicurezza e l'univocità delle informazioni trasmesse.

Descrizione

VerifiedMovies è un'applicazione decentralizzata (dApp) sviluppata durante lo stage, con l'obiettivo di fornire una piattaforma sicura e trasparente per la verifica e l'autenticazione dei film. La dApp sfrutta la tecnologia blockchain, i protocolli di identità digitale e i concetti di identità auto-sovrana per garantire l'integrità e l'autenticità dei dati dei film.

Le funzionalità implementate in VerifiedMovies includono:

- Registrazione e Accesso: VerifiedMovies offre agli utenti la possibilità di registrarsi e accedere alla piattaforma in modo sicuro e affidabile. Durante il processo di registrazione, agli utenti viene richiesto di fornire alcuni dati personali, come il nome utente, l'e-mail, la password e la data di nascita. Questi dati vengono criptati e associati in modo univoco a un Decentralized Identifier (DID) all'interno della blockchain. Durante il processo di accesso, VerifiedMovies genera un numero casuale unico e lo presenta all'utente, che deve firmare digitalmente questo numero utilizzando la propria chiave privata associata al DID. La firma viene verificata sulla blockchain e se corrisponde, all'utente viene concesso l'accesso alla piattaforma.

- Integrazione dell'identità digitale: il progetto implementa i protocolli di identità digitale, come Self-Sovereign Identity (SSI), per consentire agli utenti di creare e gestire le proprie identità in modo sicuro e rispettoso della privacy. Ogni utente ha il controllo completo sui propri dati personali e può divulgare selettivamente le informazioni quando necessario. Utilizzando i Decentralized Identifier (DID), VerifiedMovies associa un identificatore univoco a ciascun utente, consentendo loro di presentare e verificare la propria identità attraverso l'uso di Verifiable Credential e Verifiable Presentation. Questo meccanismo garantisce l'autenticità e l'integrità delle informazioni senza dover divulgare dati personali sensibili.

- Interazione con l'utente: La dApp fornisce un'interfaccia utente intuitiva che permette agli utenti di sfogliare e cercare film, visualizzarne i dettagli e interagire con la piattaforma. Gli utenti possono lasciare recensioni e valutazioni per i film, migliorando l'engagement della comunità e fornendo informazioni utili agli altri utenti nella scelta dei film da guardare. Questa interazione facilita una comunità dinamica e partecipativa all'interno della piattaforma.

Le tecnologie utilizzate per sviluppare VerifiedMovies includono piattaforme blockchain, come Ethereum, che forniscono un registro decentralizzato e immutabile per la verifica dei film. Inoltre, VerifiedMovies sfrutta i protocolli di identità digitale basati su Self-Sovereign Identity (SSI) per consentire una gestione sicura e rispettosa della privacy dell'identità degli utenti. Inoltre, VerifiedMovies fa uso di tecniche di Zero Knowledge Proof (ZKP) per garantire la privacy dei dati, consentendo agli utenti di dimostrare determinate affermazioni senza rivelare informazioni sensibili.

Nel complesso, la piattaforma rappresenta una soluzione innovativa che sfrutta le tecnologie blockchain basate su un caso d'uso comune e reale; gli approcci di identità digitale auto-sovrana garantiscono la piena sicurezza delle informazioni trasmesse, costruendo una piattaforma sicura e trasparente per la verifica e l'autenticazione dei film, accedendo ai contenuti solo previa verifica specifica. La decentralizzazione, l'integrità dei dati e la privacy dell'identità degli utenti sono al centro del progetto, offrendo per quanto possibile un'esperienza semplice e sicura.

Analisi delle tecnologie

Nel capitolo dell'Analisi delle tecnologie, esploreremo in dettaglio le tecnologie utilizzate per implementare VerifiedMovies, concentrandoci sui protocolli di identità digitale, le Zero Knowledge Proof (ZKP) e l'identità auto-sovrana. Tutto il prodotto rende disponibile un'implementazione personalizzata di queste tecnologie.

Globalmente il prodotto è composto da:

- Una homepage
- Una pagina di registrazione
- Una pagina di login
- Una pagina di visualizzazione dei film con annesso meccanismo di verifica dell'età
    - I film possono essere recensiti oppure condivisi e anche ricercati per titolo nella stessa pagina
- Una pagina di creazione della prenotazione del titolo dopo aver passato la verifica dell'età per il titolo con la sua valutazione
- Una pagina di gestione del proprio profilo
- Una pagina di visualizzazione della lista di prenotazioni compiute dall'utente

Protocolli di identità digitale: Decentralized Identifiers (DID) e Verifiable Credentials (VC)

I protocolli di identità digitale svolgono un ruolo fondamentale in VerifiedMovies per garantire una gestione sicura e rispettosa della privacy dell'identità degli utenti. Uno dei protocolli chiave utilizzati è il concetto di Self-Sovereign Identity (SSI), che permette agli utenti di avere il controllo completo dei propri dati personali e di autenticare la propria identità senza dipendere da terze parti centralizzate.

Nel contesto di VerifiedMovies, i protocolli di identità digitale vengono implementati utilizzando i Decentralized Identifier (DID), che forniscono identificatori univoci a ciascun utente. Questi identificatori vengono utilizzati per associare e verificare le identità degli utenti all'interno della blockchain. Inoltre, VerifiedMovies utilizza i concetti di Verifiable Credential e Verifiable Presentation per consentire agli utenti di presentare e convalidare le proprie informazioni di identità in modo sicuro e criptograficamente firmato.

L'utilizzo dell'intera applicazione prescinde dall'utilizzo dello smart contract SelfSovereignIdentity di Alessio De Biasi.

Esso offre funzionalità avanzate che consentono la creazione e la gestione di documenti di identità unici per ciascun utente, associando a ciascuno un \glsfirstoccur{\gls{didg}} considerato univoco.

Globalmente, il contratto definisce le seguenti strutture dati:

Parent, che rappresenta il collegamento gerarchico tra i documenti di identità rilasciato, associando un identificativo al genitore, un campo di conferma validità ed una firma;

- *VerificationMethod*, che rappresenta il metodo di verifica associato al documento di identità, comprensivo di un indice, un identificativo, un metodo di verifica di firma digitale, un controllore, un identificativo della blockchain associata e un indirizzo dell'account associato a blockchain da verificare;
- *Service*, che rappresenta il servizio associato al documento di identità, comprensivo di un identificativo, un tipo di servizio e il suo indirizzo \textit{URL};
- *DidDocumentData*, che rappresenta i metadati associati ad uno specifico documento di identità, con un suo identificativo, la sua prova di validità, tre mappe per i tipi di autenticazione, deleghe di capacità e servizi dei documenti ad essi associati e un e un \textit{Parent}e. Ciò risulta utile nella verifica della catena di fiducia implementata successivamente;

- DidDocument, che rappresenta il documento di identità associato all'utente e comprende un identificativo, un metodo di verifica, una delegazione di capacità, un servizio e un Parent;
- ResolutionResult, che rappresenta il risultato della risoluzione del documento di identità, comprendente un identificativo il documento di tipo DidDocument associato;
- ChainResolutionResult, un vettore di stringhe che racchiude i risultati della risoluzione dei vari documenti di identità associati.

Il contratto prevede una serie di metodi, ciascuno con un suo scopo, da me successivamente utilizzati nella parte front-end dell'applicazione.

Ogni metodo descritto è stato utilizzato all'interno di una pagina o componente a seconda dello scopo voluto; globalmente, possiamo specificare:

- createDid, che permette la creazione di un documento di identità, associando un \textit{DID} univoco all'utente che ha effettuato la transazione e ritorna il \textit{DID} del nuovo utente aggiunto;
- *createChildTrustedDid*, in grado di creare un nuovo \textit{DID Document} che afferma la delegazione di fiducia dall'utente certificatore (cosiddetto \textit{certification autority}) all'utente con il \textit{DID} specificato. Questo permette di creare la catena degli \textit{issuer} fidati, partendo dalla detta certification autority e arrivando fino all'utente che ha effettuato la transazione, verificando i suoi dati in modo sicuro. Esso prende come parametri l'indirizzo dell'utente a cui delegare la fiducia e la firma della certification autority;
- *initializeDidDocument*, richiamato dal metodo createDid, che inizializza il documento di identità associato all'utente, prendendo come parametri il DID dell'utente e il suo indirizzo;
- *addCapabilityDelegation*, che permette di aggiungere una delega di capacità al documento di identità, prendendo come parametro l'indirizzo dell'utente che autenticherà la transazione;
- *addService*, che permette di aggiungere un servizio al documento di identità, prendendo come parametri, l'identificativo, il tipo di servizio e il suo indirizzo \textit{URL};
- *deactivate*, il quale disattiva il DID Document associato all'utente che ha effettuato la transazione;
- *resolve*, che ritorna il DID Document associato all'utente secondo un determinato DID passato come parametro;
- *resolveChain*, che ritorna la catena di fiducia percorsa avendo come ultimo nodo l'utente con il DID} specificato e ritorna la lista di utenti certificatori a cui si è passati per arrivare all'utente finale;
- *getAuthentication*, per ottenere il metodo di autenticazione associato al documento di identità e provare per certo l'utente abbia acceduto secondo il metodo corretto.

In particolare, il processo viene così descritto:

- All'avvio dell'applicazione, viene creata una catena di issuer secondo i metodi appositi nello smart contract di Alessio De Biasi e un DID univoco associato all'utente. Come richiesto da Alessio, tale DID deve essere già in possesso all'utente. Per non dipendere da meccanismi di autenticazione esterna, l'applicazione utilizza un meccanismo di challenge-response *self-contained*, cioè:

    - Viene creato un oggetto Web3 che si connette a un nodo locale Ethereum attraverso l'URL "http://localhost:8545".
    - Viene specificato l'indirizzo del contratto Ethereum con cui interagire, identificato dalla variabile "contractAddress".
    - Viene istanziato un oggetto "contract" utilizzando la libreria Web3 e l'ABI (Application Binary Interface) del contratto specificato.
    - Viene ottenuto l'elenco degli account Ethereum disponibili nell'ambiente locale tramite la funzione "getAccounts()" di Web3.

    - Viene generata una firma digitale per un numero casuale (randomNumber) utilizzando la chiave privata dell'account Ethereum corrente (accounts[0]). La firma viene ottenuta tramite la funzione "sign()" di Web3.
    - Viene chiamato il metodo "createDid()" del contratto per creare un identificatore decentralizzato (DID) per l'utente corrente. La chiamata viene effettuata utilizzando l'account Ethereum corrente come mittente.
    - Viene concatenato il DID dell'utente con una stringa "#key-1" per ottenere il valore didverifiable per poter effettuare correttamente la verifica.
    - Viene creato un oggetto "proof" che rappresenta la prova da verificare. L'oggetto contiene informazioni come il contesto, il tipo, il metodo di crittografia, la data di creazione, lo scopo della prova, il metodo di verifica, il valore della prova (randomNumber) e il valore della firma (signature).
    - Viene chiamato il metodo "getAuthentication()" del contratto passando il valore didverifiable come parametro per verificare la prova. La chiamata restituisce un risultato che viene assegnato alla variabile "verification".
    - Viene utilizzata la funzione "recover()" di Web3 per estrarre l'indirizzo Ethereum che ha firmato la prova. Viene utilizzato il valore randomNumber come messaggio e proof.signatureValue come firma.
    - Viene confrontato l'indirizzo Ethereum estratto (recovered) con l'indirizzo restituito dalla chiamata a "getAuthentication()" (verification[5] in quanto è il quinto oggetto ritornato dall'array). Se i due indirizzi corrispondono, la prova viene considerata verificata.
    - Se la prova viene verificata con successo, viene aggiornato lo stato dell'utente con il DID ottenuto e viene chiusa una finestra modale di verifica. In seguito, l'utente viene reindirizzato alla pagina "/movies".
    - Se la prova non viene verificata, viene mostrato un messaggio di avviso indicando che la verifica non è andata a buon fine e si richiede all'utente di riprovare.

- In quanto richiesto esplicitamente da parte di Alessio, tale meccanismo è stato implementato in questo modo per non dipendere dalla piattaforma MetaMask, nota per permettere l'accesso e firmare le transazioni in modo semplice e veloce sulla piattaforma Ethereum.

Successivamente, l'utente è portato alla pagina di visualizzazione di tutti i film; quando clicca su un film, prima di poter effettuare la prenotazione, l'utente è portato a verificare la propria età, secondo i passaggi di seguito descritti in dettaglio:

- Viene creata una Verifiable Credential (VC) che specifica il DID dell'utente e la sua appartenenza ad uno schema comune di issuers creati all'avvio dell'applicazione. Questo permette di creare una catena di fiducia, che verrà poi testata con la chiamata dello smart contract di Alessio De Biasi, su cui questa implementazione ha dovuto basarsi. Ciascun Decentralized Identifier è univoco per utente e nel sito viene utilizzato per il meccanismo di registrazione ed autenticazione.
    - La firma digitale utilizzata nella creazione della VC è la CL Signature 2019 che usa questi passi
        - Generare una chiave privata per l'emittente, utilizzata per firmare la credenziale.
        - Generare una chiave pubblica dalla chiave privata, utilizzata per verificare la firma.
        - Generare un valore nonce, utilizzato per creare la prova di correttezza della firma.
        - Calcolare il valore di impegno, creato moltiplicando il nonce per la chiave pubblica, che viene utilizzato per creare la prova di correttezza della firma.
        - Calcolo del valore di risposta, creato moltiplicando la chiave privata per il valore di impegno, utilizzato per creare la prova di correttezza della firma.

- Calcolo del valore della firma, utilizzato per creare la prova di correttezza della firma.
- Calcolo della prova di correttezza della firma, utilizzata per dimostrare che la firma è valida.

```
const vc: VCDIVerifiableCredential = {
        '@context': ['https://www.w3.org/2018/credentials/v1'],
        id: 'http://localhost:3000/ageCredentialSchema',
        type: ['VerifiableCredential'],

        // According to the only source found on the subject:
https://blog.goodaudience.com/cl-signatures-for-anonymous-credentials-93980f720d99
        // this implements a base for ZKP for anonymous credentials (CL signature)
for second layer solutions;
        // according to https://arxiv.org/pdf/2208.04692.pdf this signature type is
standardized (page 23/30, the table)
        // and classified for JSON CL Signatures, used for example in Sovrin or
Hyperledger Indy.
        // This allows credential binding with persistent identifier, and that's
exactly what it's used here.
        credentialSchema: {
            id: userDid ? userDid : "",
            type: "VerifiableCredential"
        },
        issuer: {
          id: issuerDid,
          publicKey: publicKeyHex //the issuer's public key
        } as IssuerObject,
        issuanceDate: new Date().toISOString(),
        credentialSubject: {
          id: userDid,
          age: 25,
          type: 'VerifiableCredential',
        } as CredentialSubject,
        proof: {
          type: "CLSignature2019", // used for anonymous credentials and ZKP for
second layer solutions
          issuerData: issuerDid, // the issuer's DID
          attributes: masterSecret, // ths field is made of attributes not defined
inside the schema - i.e. the master secret attribute
          signature: signature, // the signature is generated with the user's
private key
          signatureCorrectnessProof: proof.signatureCorrectnessProof, //generate
before a proof of correctness then sign it with the issuer's private key (inside
AnonCred example this is the primaryProof)
        },
      };
      return vc;
```

```
  }
```

Fonte di questa implementazione: https://www.w3.org/TR/vc-data-model/#example-a-verifiable-credential-that-supports-cl-signatures

- Il flusso di verifica completo seguito è adatto alle soluzioni blockchain di secondo livello, come specificato da https://hyperledger.github.io/anoncreds-spec/#anoncreds-setup-data-flow e adattato al tipo di firma digitale CL Signature 2019 secondo lo schema specificato dalla stessa pagina. Tale firma viene utilizzata e verificata per autenticare le transazioni compiute.
    - 1. L'emittente crea uno schema di credenziali
    - 2. L'emittente crea una definizione di credenziale
    - 3. L'emittente crea un'offerta di credenziali
    - 4. L'emittente crea una richiesta di credenziale
    - 5. L'emittente rilascia una credenziale
    - 6. Il titolare memorizza una credenziale
    - 7. Il titolare crea una richiesta di prova
    - 8. Il titolare crea una prova
    - 9. Il verificatore verifica una prova
- Il tutto viene poi salvato in un Verifiable Data Registry (che in questo caso è una blockchain).
- L'holder è l'utente, l'issuer è viene generato all'inizio del sito, il verificatore è lo stesso sito del cinema con il suo meccanismo interno

- Viene creata una Verifiable Presentation (VP) secondo la seguente codifica, adattata al meccanismo di firma digitale precedentemente descritto, che permette di implementare il meccanismo Zero Knowledge Proof, specificando l'appartenenza comune ad uno schema senza rivelare altre informazioni personali:

```
-         const vp: VerifiablePresentation = {
-           '@context': [
-             "https://www.w3.org/2018/credentials/v1",
-             "https://www.w3.org/2018/credentials/examples/v1"
-           ],
-           type: "VerifiablePresentation",
-           verifiableCredential: [vc],
-           proof: {
-             type: "CLSignature2019",
-             proofValue: {
-               signatureValue: {
-                 r: signature.r.toString(16),
-                 s: signature.s.toString(16),
-               },
-               signatureCorrectnessProof: signatureCorrectnessProof,
-             },
-           }
-         };
-         return vp;
-       }
```

- Viene verificata la correttezza di questa firma digitale adottando l'algoritmo *secp256k1* di Bitcoin, al fine di verificare la prova di correttezza tramite il calcolo della curva ellittica associata alla transazione
- Viene chiamato lo smart contract di Alessio De Biasi per verificare la catena di fiducia degli issuer iniziali; sostanzialmente, viene creata una catena fittizia, con una *certification authority* che firma a cascata la creazione di DID figli per una serie di 3 issuer; l'utente si fida dell'ultimo nodo della catena e, a cascata, questa viene percorsa e viene ritrovata l'appartenenza al primo nodo della catena, dimostrando che il meccanismo implementato funziona correttamente.

Ogni meccanismo viene descritto più in dettaglio all'interno di:

https://github.com/gabrielrovesti/Stage-e-tesi-UniPD/blob/main/Tesi/tesi.pdf

Qui viene dato uno sguardo più nel dettaglio delle azioni principali compiute dal sistema.

Problematiche e soluzioni

Lo sviluppo della dApp è stata di non poche problematiche, in quanto un'implementazione di questa complessità per una singola persona triennale e senza supporto esterno, se non i pochi chiarimenti dati da Alessio De Biasi non sono state positive. Tutto sommato, capendo a fondo le singole tecnologie di firme digitali esplorate per conto mio e le conoscenza maturate durante lo sviluppo mi hanno permesso ugualmente di completare pienamente il prodotto.

La problematica principale è stata data dalla complessità del meccanismo di firma digitale e dal dover comprendere passo per passo 2 standard interi (DID e VC/VP) e la Zero Knowledge Proof, territorio estremamente teorico e molto fumoso. Leggendo i vari papers in materia, sono riuscito a trovare un'implementazione di firma digitale soddisfacente gli standard W3C presenti. Infatti, ho dovuto realizzare un meccanismo personalizzato di verifica analizzando le poche implementazioni di blockchain di secondo livello interamente da solo, capendo cosa sarebbe potuto essere fattibile a livello di codifica e di meccanismo logico.

La soluzione implementata, per quanto semplificata per ovvi motivi di implementazione temporale, risulta essere pienamente funzionante e frutto di un'analisi approfondita del prodotto e dei suoi standard. La parte di Zero Knowledge Proof e verifica della firma digitale presente, in particolare, è stata una parte che ho curato completamente da solo, dato che anche il citato Alessio De Biasi non ha volutamente potuto né voluto fornire alcun supporto concreto.

Tramite lo studio dei riferimenti prima citati e di una serie di ulteriori qui listati, mi è stato possibile realizzare il tutto concretamente:

- https://blog.goodaudience.com/cl-signatures-for-anonymous-credentials-93980f720d99
- https://w3c-ccg.github.io/ld-cryptosuite-registry/
- https://medium.com/finema/anonymous-credential-part-2-selective-disclosure-and-cl-signature-b904a93a1565
- https://hyperledger.github.io/anoncreds-spec

La stessa interazione con la blockchain e la relativa configurazione tra frontend e smart contract è stato un altro passo cruciale e che ha richiesto non poco tempo, essendo un ulteriore passo compiuto in completa solitudine. Grazie all'uso di un template iniziale trovato su GitHub mi è stato possibile realizzare concretamente il prodotto.

Nonostante tutto, la soluzione trovata, dati i riferimenti, il prodotto stesso e la tesi, è pienamente attinente agli standard e anzi esplora dei territori fin qui non sviluppati e sono fiero della mia implementazione.

Sfide, limiti e scenari futuri

Nel contesto dell'applicazione che ho sviluppato, è fondamentale esaminare attentamente i problemi e le sfide legate all'implementazione della SSI su blockchain Ethereum. Uno dei principali problemi è la gestione delle chiavi private e pubbliche degli utenti, garantendo al contempo un'esperienza utente sicura e intuitiva. Inoltre, la scalabilità della rete rappresenta una sfida significativa quando si tratta di consentire un numero sempre crescente di utenti a interagire con l'applicazione senza compromettere le prestazioni. Affrontare queste problematiche richiede un'analisi approfondita e l'implementazione di soluzioni innovative.

Inoltre, l'ambito della Self Sovereign Identity risulta certamente un ambito interessante e completamente sicuro da esplorare per lo sviluppo di soluzioni di identità sicure; uno standard e un meccanismo certamente interessante da vedere, comprendere e immaginare. Rimane un meccanismo complicato da realizzare, per quanto portato a molti e interessanti sviluppi futuri.

Nell'ambito della nostra app, ho considerato attentamente le implicazioni e le limitazioni dell'uso delle soluzioni di second layer all'interno della blockchain Ethereum. Sebbene queste soluzioni offrano un modo promettente per affrontare le limitazioni di scalabilità della blockchain, è importante considerare l'interoperabilità con il layer principale e garantire la sicurezza delle transazioni attraverso i canali di second layer. L'utilizzo della firma digitale da me realizzata e utilizzata implementa pienamente a livello logico la descrizione fisica e logica di questi standard.

La sicurezza è stato un aspetto fondamentale di tutta l'applicazione, ma anche la semplicità. Per quanto complesso sia stato, ho mantenuto tutto lo sviluppo in locale e sono comunque riuscito a sviluppare correttamente ogni cosa, dall'inizio alla fine. I dati dell'utente vengono criptati in AES in localStorage, l'utilizzo di blockchain fa in modo che le transazioni pubbliche vengano realizzate in modo immutabile, allo stesso tempo incapsulando i dati dell'utente in modo sicuro e privato.

La mia soluzione rappresenta certamente un punto di partenza per realizzare degli standard complessi e che richiedono uno studio approfondito ed attento, possibilmente supportato da più persone e non da un singolo.

Conclusioni

Ritengo il tirocinio un’esperienza positiva in linea di massima, nonostante la complessità dell’implementazione, dato il mio interesse e curiosità verso tematiche per molti versi inesplorate e che meritano di essere conosciute, principalmente dagli addetti del settore, data la totale non intuitività del tema. Rimane innegabile che, da un punto di vista di sicurezza, la blockchain sia un territorio che spero verrà esplorato ed ampliato molto nei prossimi anni.

L'ambito della SSI e della ZKP si è rivelato affascinante e promettente, rappresentando un meccanismo di identità sicura che merita di essere esplorato e compreso. L'implementazione della firma digitale e l'utilizzo della blockchain hanno contribuito a creare un ambiente immutabile per le transazioni pubbliche, mantenendo al contempo i dati degli utenti sicuri e privati. Questo rappresenta solo l'inizio di un percorso che richiede uno studio approfondito e il coinvolgimento di più persone per sviluppare standard complessi e avanzati.

Nel contesto dell'applicazione sviluppata, ho considerato attentamente le implicazioni e le limitazioni delle soluzioni di second layer all'interno della blockchain Ethereum. Sebbene queste soluzioni offrano un modo promettente per affrontare le limitazioni di scalabilità, è fondamentale garantire l'interoperabilità con il layer principale e la sicurezza delle transazioni attraverso i canali di second layer. La mia implementazione ha cercato di bilanciare in modo efficace sicurezza e semplicità, mantenendo i dati degli utenti criptati in modo sicuro e utilizzando la blockchain per transazioni immutabili.

Guardando al futuro, è evidente che la blockchain rappresenta un territorio di sicurezza e innovazione in continua espansione. È importante continuare a seguire da vicino gli sviluppi del settore, collaborando con esperti e partecipando attivamente alla comunità per rimanere all'avanguardia nelle soluzioni e negli standard emergenti.

In conclusione, il tirocinio è stata un'esperienza preziosa che mi ha permesso di comprendere e affrontare le sfide dell'implementazione della SSI su blockchain Ethereum. Ho iniziato a coltivare un forte interesse verso le tematiche coinvolte, nonché la sicurezza e l'innovazione, esplorando nuovi orizzonti nella blockchain e contribuendo al progresso della tecnologia in questi standard ancora inesplorati. Per un tirocinio triennale condotto in solitaria, quanto ottenuto è un piccolo ma grande passo per dimostrare la piena applicabilità di questi concetti.