UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA


Corso di Laurea in Informatica

# Implementazione di un sistema di autenticazione mediante la tecnologia Blockchain Ethereum

Relatore:
**Christiancarmine ESPOSITO**

Candidato:
**Alberto MONTEFUSCO**
**Mat. 0512106272**

ANNO ACCADEMICO 2021/2022

*Stay hungry.*
*Stay foolish.*

# Indice

# Capitolo 0

# Premessa

Le Web Application odierne si basano sul principio di offrire servizi in cambio dei dati degli utenti. Questo comporta sia una centralizzazione che una perdita del controllo dei dati da parte degli utenti, generando problemi di sicurezza. La Blockchain, attraverso un ecosistema decentralizzato, punta a ridare tale controllo agli utenti, concentrandosi in modo particolare su temi riguardanti la privacy.

Per garantire un'autenticazione sicura, nell'ambito della tesi, si introduce la tecnologia Blockchain basata su Ethereum, affiancata dallo sviluppo di un robusto Smart Contract: un accordo stipulato tra varie parti proprio come un qualsiasi tradizionale contratto scritto tranne per il fatto che questi contratti intelligenti sono realizzati in codice informatico e incorporati in una Blockchain. Di conseguenza, è stata sviluppata un'applicazione web decentralizzata (Web DApp), la quale, a differenza delle app tradizionali, non ha bisogno di alcun mediatore per funzionare.

Il flusso di esecuzione della Web DApp realizzata è il seguente: l'utente si registrerà inserendo all'interno di un form i suoi dati, quali username o e-mail, password, un secondo campo di inserimento password (per controllare che l'utente non abbia commesso errori nella digitazione) e due pulsanti per scegliere il ruolo che vorrà avere l'utente all'interno del sistema. La registrazione sarà confermata soltanto dopo aver inserito la private key associata al seguente account: la private key è una chiave personale e segreta conservata in maniera sicura su MetaMask e dovrà essere inserita solo una volta, quando l'utente effettua la prima transazione e di conseguenza carica il contratto sulla Blockchain.Una volta effettuata la registrazione, l'utente potrà eseguire l'operazione di Login soltanto dopo l'approvazione dell'amministratore.
Gli amministratori possono attivare, cioè attribuire il ruolo di "Admin" o di "User", a qualsiasi account (ancora non attivo) di qualsiasi utente registrato al sistema. Inoltre, per gli account registrati e già attivi con il ruolo di "User", qualsiasi amministratore può promuoverli al ruolo di "Admin", mentre, solo l'amministratore proprietario, cioè colui

che ha effettuato il login al sito in quel momento, può rinunciare al suo diritto di essere "Admin" e quindi diventare "User", oppure potrà disattivare il suo account e di conseguenza perdere qualsiasi ruolo.

Nella fase iniziale del progetto è stato scritto uno smart contract in Solidity reso sicuro e robusto tramite la libreria OpenZeppelin per la definizione dei ruoli assunta dagli utenti: "Admin" o "User".
Le operazioni di assegnazione o revoca dei ruoli sono effettuate solo dall'account che possiede come ruolo "`DEFAULT_ADMIN_ROLE`" all'interno dello smart contract. Di conseguenza, tutti gli altri utenti sono impossibilitati nel modificare o nell'accedere a quei servizi che potranno essere eseguiti dall'amministratore. Gli account, invece, che hanno come ruolo "`USER_ROLE`" potranno effettuare solamente il login al sito oltre che a navigare tra le varie pagine, ma non potranno mai accedere alla sezione di gestione account riservata agli amministratori.
La registrazione di un nuovo utente avviene memorizzando, all'interno della struct "User" dello smart Contract "`Authentication.sol`", le seguenti informazioni: l'account esadecimale che vuole effettuare la transazione, l'indirizzo e-mail o username e la password che sarà cifrata. La cifratura avviene tramite la funzione *keccak256* ed è previsto un controllo per verificare se l'attuale utente è già registrato all'interno della Web DApp KryptoAuth.
La funzione di login verifica se l'username e la password inviati coincidono con quelli memorizzati nella Blockchain restituendo un booleano.

Successivamente, è stato effettuato il deploy dello Smart Contract eseguendo, prima, l'applicazione Ganache e dopodiché, all'interno del terminale, (nel package "smart contract" del progetto KryptoAuth) è stato eseguito:

- `truffle compile`, per verificare la presenza di errori sintattici all'interno dello Smart Contract;
- `truffle migrate`, per deployare lo Smart Contract sulla Blockchain Ganache (per effettuare un reset delle connessioni è possibile eseguire `truffle migrate --reset`).

Arrivati a questo punto è stata creata la Web DApp utilizzando il framework Spring Boot come backend e HTML, CSS, JS per il frontend. In particolare, in questa fase è stato tradotto lo Smart Contract `Authentication.sol` (scritto in Solidity) in una classe Java, tramite:

- il compilatore *solcjs*, per la generazione dei file `Authentication.abi` e `Authentication.bin`;

- la libreria *Web3j*, per la creazione della classe `Authentication.java`.

Infine, è stata creata e configurata una nuova rete per il crypto wallet MetaMask in modo da poter effettuare l'interfacciamento alla Blockchain di test Ganache. I parametri inseriti sono stati reperiti da Ganache:

- Nome rete: assegniamo il nome che vorremmo che abbia la nuova rete (esempio: *Ganache*);
- Nuovo URL RPC: è l'indirizzo http di Ganache (HTTP://127.0.0.1:7545);
- Chain ID: si deve inserire 1337 che corrisponde all'id di Ethereum;
- Currency Symbol: si deve inserire "ETH" perchè abbiamo una Blockchain Ethereum.

Ciò ha permesso alla DApp di poter comunicare con la Blockchain e di effettuare le varie transazioni richiamando le funzioni scritte nello smart contract.

# Capitolo 1

# Introduzione

## 1.1 Il Web

Internet, la rete di telecomunicazioni che collega i computer in tutto il mondo, è nata nel 1969 e da allora ha subito numerose modifiche tecnologiche e infrastrutturali per raggiungere ciò che è oggi. Lo scopo iniziale di Internet come mezzo di condivisione delle informazioni è andato ben oltre nel corso degli anni ed è diventato una parte essenziale della nostra vita. L'introduzione del World Wide Web da parte di Tim Berners-Lee ha svolto un ruolo importante nel trasformare le nostre vite in modi nuovi.

Il web, elaborato come World Wide Web, è una raccolta di siti web costruiti sulla base di Internet. Questi siti web contengono informazioni sotto forma di pagine di testo, immagini digitali, video, audio ecc., che gli utenti possono recuperare da qualsiasi parte del mondo. Nella Figura 1.1 possiamo notare uno schema semplificato per accedere ad una risorsa presente sul Web: tramite il nostro browser, effettuiamo una richiesta che sarà ricevuta dal server tramite Internet; il server elabora la richiesta ed invia la risposta al client resa visibile dal nostro browser.

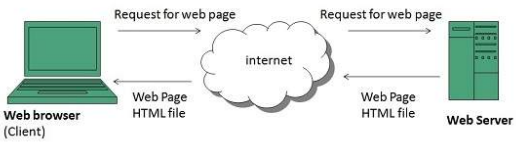


Figura 1.1: Schema Word Wide Web

Per cui, mentre Internet sta avanzando verso la sua terza iterazione molti cambiamenti tecnologici e accese discussioni sono in corso, causati dalla confuzione sulla differenza tra Web3 e Web 3.0. Sebbene la maggior parte delle discussioni che ruotano attorno alla terza generazione del web implichino che Web3 e Web 3.0 siano gli stessi, entrambi sono fondamentalmente diversi. Mentre Web3 è una versione decentralizzata del web basata su Blockchain, Web 3.0 mette in evidenza il concetto di Tim Berners-Lee di web semantico.

### 1.1.1 Le tre versioni del Web

Il web ha subito molte fasi di evoluzione sin dal suo inizio.

Il Web 1.0, la versione iniziale del World Wide Web, è stata sviluppata da Tim Berners-Lee nel 1989 ed è durata fino al 2004. Solitamente indicato come il Web di sola lettura, i siti web di quest'epoca erano esclusivamente informativi e comprendevano semplicemente contenuti statici. Non avevano alcun contenuto interattivo ed erano principalmente collegati tramite collegamenti ipertestuali. Nel complesso, Web 1.0 (mostrato in Figura 1.2) era una rete di distribuzione di contenuti (CDN) che consentiva la visualizzazione di informazioni su siti Web in cui gli utenti consumavano materiali passivamente senza avere la possibilità di lasciare recensioni, commenti o altri tipi di feedback.

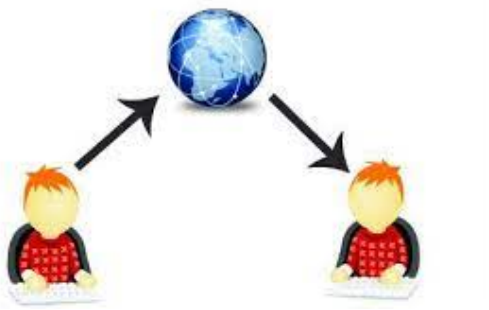

Figura 1.2: Web 1.0

Il Web 2.0, noto anche come la seconda generazione del Web, è il Web prevalente della nostra era emerso nel 2004 ed è ancora fiorente. È considerato il Web in lettura e scrittura che facilita l'interazione dell'utente (come mostra la Figura 1.3), il che rappresenta un enorme miglioramento rispetto alla comunicazione unidirezionale consentita dal Web 1.0. La connettività e l'interattività sociale del Web 2.0 hanno portato allo sviluppo di piattaforme di social media come Facebook, Twitter, YouTube o Discord, in cui gli utenti possono caricare contenuti che altri utenti possono visualizzare e fornire feedback. Tutto ciò ha portato Internet a estendersi ai dispositivi mobili come iPhone e Android, portando al predominio di app come WhatsApp, Instagram, Uber e Paytm.

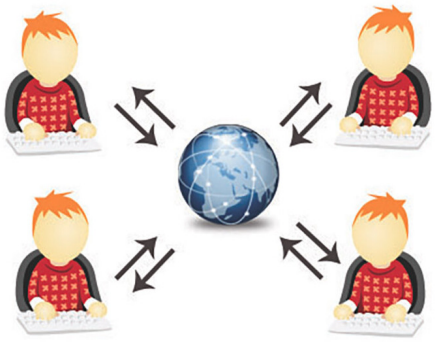

Figura 1.3: Web 2.0

La prossima generazione del Web è il Web 3.0 noto anche come Web Semantico ed è un'estensione del World Wide Web che utilizza gli standard stabiliti dal World Wide Web Consortium (W3C). Mira a rendere Internet più intelligente gestendo le informazioni con un'intelligenza simile a quella umana. Tim Berners-Lee ha coniato il termine Semantic Web (Web Semantico), che si riferisce a una versione del web in grado di connettere tutto a livello di dati. Ha affermato che con l'emergere del Web Semantico,

*"i meccanismi quotidiani del commercio, della burocrazia e della nostra vita quotidiana saranno gestiti da macchine che dialogano con le macchine. Gli "agenti intelligenti", che le persone hanno propagandato per secoli, si materializzeranno finalmente".*

### 1.1.2 Che cos'è il Web3?

Il Web3 è un web decentralizzato e aperto basato sulla tecnologia Blockchain. Coniata dal cofondatore di Ethereum Gavin Wood nel 2014, l'idea alla base di Web3 è quella di creare una versione decentralizzata di Internet rimuovendo il predominio del potere centralizzato dei giganti del Web 2.0 come Amazon e Facebook e restituendo il controllo agli utenti.

Vuole essere un network in cui i contenuti e i servizi non risiedono più su server e piattaforme che appartengono a multinazionali e aziende, ma sono disseminati in maniera omogenea sulla rete. Grazie alla partecipazione democratica, gli utenti potrebbero finalmente monetizzare la condivisione dei dati. I contenuti prodotti dagli utenti resterebbero, infatti, nelle mani dei legittimi autori e non finirebbero più sotto l'ombrello delle piattaforme, come Tik Tok, Instagram e YouTube.

Il tutto viene reso possibile da portafogli di criptovaluta come MetaMask, Venly o TrustWallet, in cui gli utenti archiviano le chiavi di tutti i loro dati e identità. L'utilizzo di un portafoglio cifrato per accedere ad altre app è simile all'utilizzo di un account Facebook, tranne per il fatto che i dati sono conservati e gestiti direttamente dall'utente.

### 1.1.3 Differenze tra Web3 e Web3.0

Il Web Semantico, noto come Web 3.0, riutilizza e collega i dati tra i vari siti web. Il web decentralizzato o Web3, tuttavia, pone una forte enfasi sulla sicurezza e sull'empowerment restituendo agli utenti il controllo dei dati e dell'identità.

Il Web 3.0 memorizza tutti i dati in un'unica posizione centrale chiamata Solid pod consentendo agli utenti di controllare quali persone e quali applicazioni possono accedere ai propri dati. In cima a quel pod, si aggiunge un Web ID in modo che l'utente possa identificarsi sul Web. Nel Web3, basato su Blockchain, gli utenti possono archiviare i propri

dati in un portafoglio cifrato, a cui possono accedere utilizzando le proprie chiavi private.

Inoltre, entrambi utilizzano tecnologie diverse per implementare il loro scopo di sicurezza dei dati. Web3 utilizza la tecnologia Blockchain, mentre nel Web 3.0 vengono utilizzate alcune tecnologie di interscambio di dati come RDF, SPARQL, OWL e SKOS.

I dati in Web3 sono difficili da modificare o eliminare poiché sono sparsi su più nodi; tuttavia, i dati in Web 3.0 possono essere modificati senza sforzo. Infatti, i dati archiviati nel solid pod sono centralizzati, mentre le chiavi archiviate nei portafogli cifrati forniscono l'accesso ai dati degli asset che risiedono su una Blockchain.

### 1.1.4 Somiglianze tra Web3 e Web 3.0

Sebbene sia Web3 che Web 3.0 siano simili nei nomi, c'è un'enorme differenza nei loro concetti e nell'approccio. Tuttavia, entrambi hanno uno scopo comune. Sia Web3 che Web 3.0 mirano a creare una versione migliore di Internet mantenendo il controllo degli utenti sui propri dati. La differenza fondamentale sta nell'approccio adottato per raggiungere questo scopo. Mentre i dati sono archiviati in un solid pod nel Web 3.0, Web3 utilizza tecnologie decentralizzate per lo stesso.

### 1.1.5 Architettura del Web Semantico

Il Web Semantico è, come l'XML, un ambiente dichiarativo, in cui si specifica il significato dei dati e non il modo in cui si intende utilizzarli. La semantica dei dati consiste nel dare alla macchina delle informazioni utili in modo che essa possa utilizzare i dati nel modo corretto, convertendoli eventualmente. Riassumendo, il Web Semantico si compone di tre livelli fondamentali:

1. i dati;
2. i metadati riportano questi dati ai concetti di uno schema;
3. nello schema si esprimono le relazioni fra concetti, che diventano classi di dati.

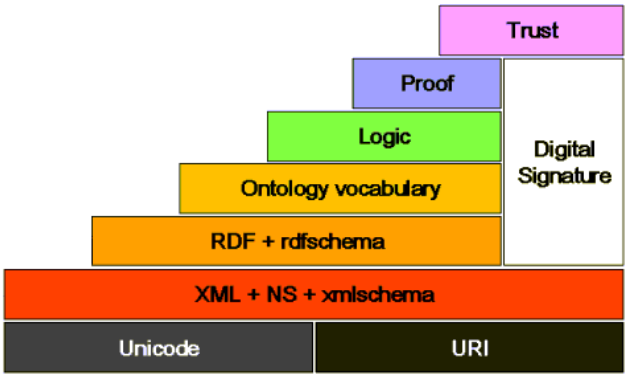


Figura 1.4: Livelli del Web Semantico

Nella Figura 1.4 possiamo notare l'architettura a livelli del Web Semantico, che però non è stata ancora sviluppata completamente:

- al primo livello troviamo gli standard **URI** e **Unicode**. Unicode è un sistema di codifica di caratteri che consente di utilizzare tutte le lingue umane sul web assegnando un numero univoco ad ogni carattere. URI, invece, è una stringa che consente di identificare le risorse (es. documenti) in modo univoco.

- Al livello superiore si trova **XML** (*eXtensible Markup Language*) che modella la realtà che si considera. Per realtà si intende qualsiasi risorsa che sia possibile identificare sulla rete con un indirizzo univoco, mentre per descrizione si indica l'insieme delle proprietà, degli attributi e delle relazioni con altre realtà.

- **RDF** (*Resource Description Framework*) e **RDF Schema** costituiscono il linguaggio per descrivere le risorse e i loro tipi. Derivano da XML.

- Il **livello ontologico** permette di descrivere le relazioni tra i tipi di elementi senza fornire informazioni su come utilizzare queste relazioni dal punto di vista computazionale.

- La **firma digitale** è importante nei diversi livelli dell'architettura del Web Semantico poiché viene utilizzata per stabilire la provenienza delle ontologie e delle deduzioni, oltre che dei dati.

- Il **livello logico** è il livello in cui le asserzioni esistenti sul Web possono essere utilizzate per derivare nuova conoscenza. Tuttavia, i sistemi deduttivi non sono interoperabili, per cui invece di progettare un unico sistema onnicomprensivo per supportare il ragionamento, si potrebbe pensare di definire un linguaggio universale per rappresentare le dimostrazioni. I sistemi potrebbero quindi autenticare con la firma digitale queste dimostrazioni ed esportarle ad altri sistemi che le potrebbero incorporare nel Web Semantico.

## 1.2 Self Sovereign Identity

Oggigiorno, il Web è basato sul principio di offrire servizi in cambio dei dati degli utenti. Questo aspetto genera grandi problemi di sicurezza che possono essere risolti restituendo il controllo dei dati agli utenti tramite il concetto della Self Sovereign Identity: un modello di identità digitale che restituisce all'utente che la crea il pieno controllo della sua identità e delle informazioni da condividere. Le possibilità date dalla SSI sono molteplici e il suo mercato è destinato a crescere esponenzialmente nei prossimi anni.

Quotidianamente distribuiamo i nostri dati e le nostre informazioni a fornitori di servizi, creando molteplici e numerose "identità digitali" su cui il nostro controllo è minimo. Gli ultimi sviluppi in tema digital identity hanno però l'obiettivo di riportare il pieno controllo dell'identità all'utente che la crea, grazie al modello della Self Sovereing Identity.

L'obiettivo della SSI è permettere all'utente di essere *"sovrano"* della sua identità in modo da poter controllare e scegliere quali informazioni personali condividere e con chi. Alla base vi è il concetto che l'identità debba sempre essere sotto l'esclusivo controllo dell'individuo che ne è rappresentato, il quale può disporne in modo indipendente, senza la necessità di affidarsi a intermediari terzi.

La SSI nasce per evitare sia l'identificazione "diretta" tra utente e service provider (si crea un account su un sito per usufruire dei suoi servizi), sia l'identificazione cosiddetta "3-corner" (si usa l'account di un servizio per iscrivermi a un servizio terzo).
Il modello della Self Sovereign Identity, mostrato nella Figura 1.5, si basa invece sul paradigma dell'Identità Decentralizzata: è l'utente stesso a disporre di un'unica identità, a cui vengono associati una serie di attributi, detti "claim", da altre entità, dette "issuer", sempre verificabili dai "verifier". In poche parole, ogni identità digitale disporrà di una serie di claim personali (ad esempio, l'università assocerà l'attributo "laurea" all'identità digitale del laureato, il quale potrà avere nel suo wallet anche l'attributo "residente a Milano" associatogli dal comune di Milano).

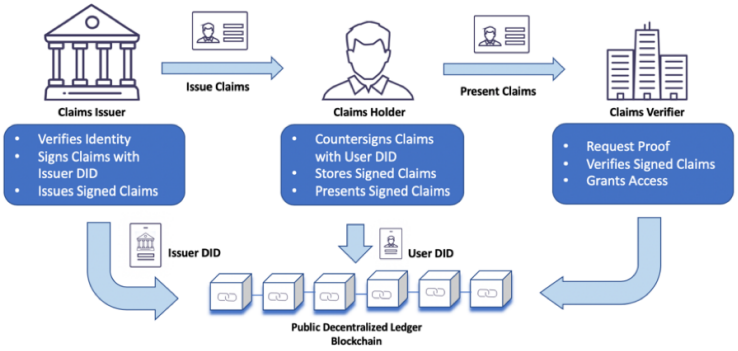


Figura 1.5: Modello SSI

Ancora più interessanti sono i risvolti dell'utilizzo della Blockchain nel modello SSI. La Blockchain, infatti, oltre a rendere ogni claim verificabile e immutabile, darebbe la possibilità al proprietario dell'identità di presentare e far verificare singolarmente i claim. Ciò vuol dire che, grazie alla SSI, si può dimostrare l'originalità senza dover mostrare informazioni aggiuntive ad un servizio terzo.

### 1.2.1 Vantaggi della SSI

Grazie al modello SSI si può utilizzare il nostro "wallet" (portafoglio digitale) unico per accedere alle nostre banche, account personali, servizi pubblici, senza doversi ricordare diverse credenziali. E da qui i possibili risvolti del modello SSI si moltiplicano da ogni punto di vista: basti pensare alle possibilità in ambito digital banking o nel mondo dell'e-voting (le elezioni online), ma anche cosa si sarebbe potuto fare durante il lockdown da Covid-19 per sostituire le autodichiarazioni cartacee. In sintesi, la Self Sovereign Identity consentirà:

- maggiore proprietà e controllo dell'identità;
- maggiore proprietà e controllo dei dati;
- riduzione tentativi di manipolazione delle password (uso di parametri biometrici);
- riduzione del numero di frodi legate all'identità e di «data breach».

### 1.2.2 Attori in una credenziale verificabile

La Figura 1.6 mostra i vari attori che partecipano nella verifica di una credenziale.
Un emittente (issuer) è un'entità autorizzata a rilasciare una credenziale. Questi emittenti sono tipicamente organizzazioni governative, centri sanitari, banche e istituzioni finanziarie, scuole e università, startup, ecc.

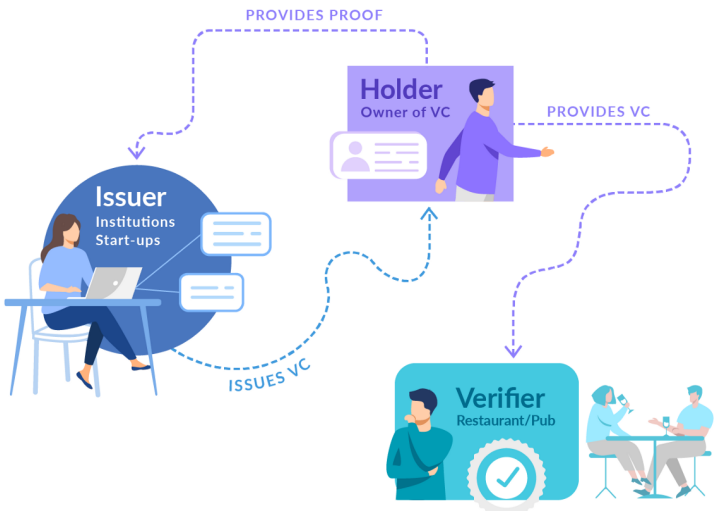


Figura 1.6: Issuer, Holder e Verifier

Un titolare (holder) è qualcuno che è il titolare della credenziale e ha il controllo completo su come può essere gestita, condivisa o revocata. I titolari sono in genere individui o organizzazioni.

Un verificatore (verifier) è un'entità che verifica una credenziale e garantisce che provenga da un emittente competente, sia a prova di manomissione ed è ancora rilevante (non scaduta o revocata). Un verificatore prende la presentazione verificabile dal titolare per determinarne l'autenticità.

### 1.2.3 Modello SSI

Sono tre gli elementi tecnici alla base di un sistema di SSI: i Decentralized Identifiers (DID), le Verifiable Claim (VC) e i DID Document.

1. **Decentralized Identifier** (DID): una stringa alfanumerica che identifica univocamente un'entità (Figura 1.7). Per esempio, alcuni identificativi per un individuo possono essere il nome e cognome o il codice fiscale, ovvero dei codici che consentono di riconoscere univocamente questo individuo. Allo stesso modo in un modello SSI i DID sono codici alfanumerici basati su un sistema a doppia chiave crittografica, memorizzato su Blockchain, che consentono di identificare univocamente un'entità online.


Figura 1.7: Identificatore Decentralizzato (DID)

2. **Verifiable Credential** (VC): un qualsiasi tipo di attributo collegato a un'entità. Alcuni corrispettivi fisici di una VC sono per esempio la patente di guida o il diploma di laurea. Nel modello SSI, però, le VC sono digitali, immodificabili e verificabili in maniera indipendente in ciascuna interazione per cui è richiesto quello specifico attributo.

3. **DID Document**: aggiunge ulteriori informazioni ad uno specifico DID. Quest'ultimo, infatti, essendo un elemento atomico (solo un codice alfanumerico), ha necessità di un elemento aggiuntivo contenente ulteriori informazioni sull'entità identificata. Per esempio, un individuo in possesso di un DID potrebbe inserire nel suo DID Document la lista dei DID "fidati" nel caso la sua chiave crittografica fosse compromessa, oppure potrebbe voler indicare dei delegati a firmare dei documenti in sua vece.

### 1.2.4 Come verificare una credenziale?

Con le credenziali fisiche, la verifica si ottiene attraverso una prova di autenticità incorporata direttamente nella credenziale stessa come un chip o un ologramma. Può anche essere fatta verificando direttamente con l'emittente che le credenziali siano valide, accurate e aggiornate. Ma questo processo di verifica manuale può essere difficile e richiedere molto tempo (la Figura 1.8 mostra il processo di verifica digitale di una credenziale).

Questo porta ad uno dei vantaggi fondamentali delle credenziali verificabili: utilizzando la crittografia e Internet, possono essere verificate digitalmente in pochi secondi. Questo processo di verifica può rispondere alle seguenti quattro domande:

1. La credenziale è in un formato standard e contiene i dati necessari al verificatore?
2. Include una firma digitale valida dell'emittente?
3. La credenziale è ancora valida, cioè non scaduta o revocata?
4. Se applicabile, la credenziale (o la sua firma) fornisce la prova crittografica che il titolare della credenziale è l'oggetto della credenziale?

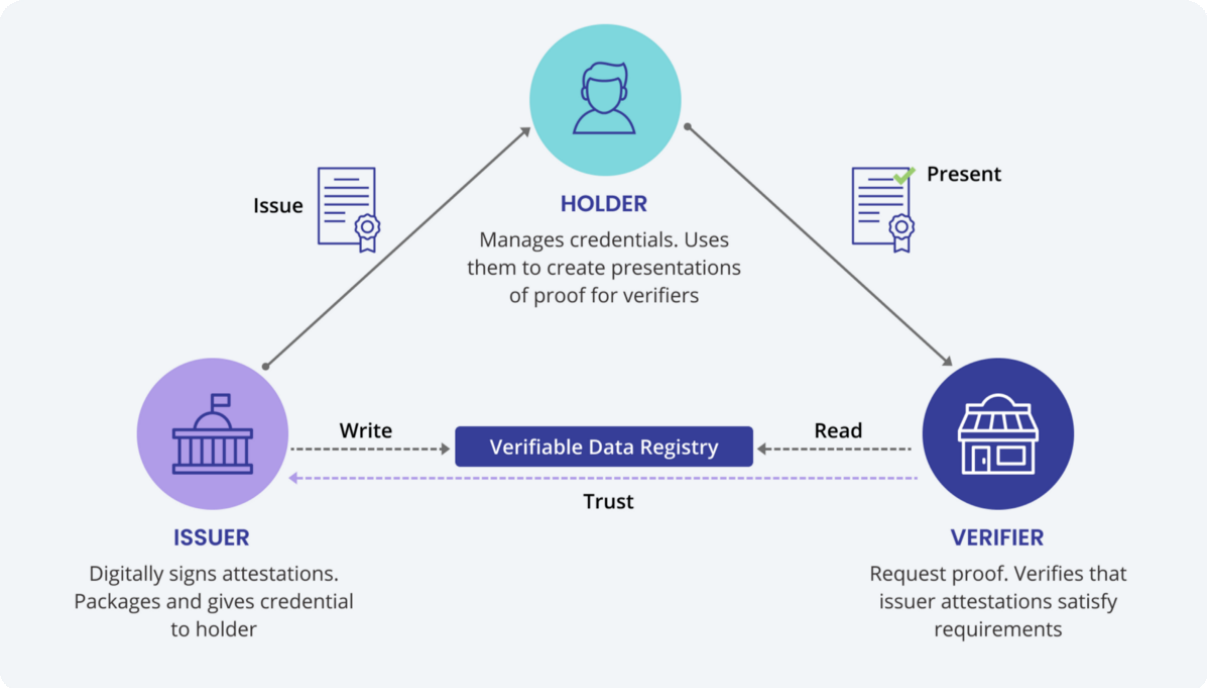


Figura 1.8: Processo di verifica

### 1.2.5 Portafoglio decentralizzato

Avere chiavi private significa avere il pieno controllo sul contenuto del portafoglio (wallet) di un utente. Un wallet decentralizzato può essere in grado di trattenere cryptovalute o l'identità digitale di un utente, in genere sull'hardware (cioè nessun server centrale che effettua operazioni CRUD sui dati). Tuttavia, molte persone non conoscono il concetto di wallet, la tecnologia Blockchain e il concetto di chiavi pubblico-private. Per questo motivo esiste un Custodial Wallet, definito come un portafoglio in cui le chiavi private sono detenute da una terza parte (nella Figura 1.9 abbiamo un esempio di Custodial Wallet, MetaMask). Ciò significa che la terza parte ha il pieno controllo sul contenuto del portafoglio mentre l'utente deve solo consentire le operazioni CRUD sul contenuto del wallet. In questo modo non si presenta il pericolo di perdere la chiave privata.

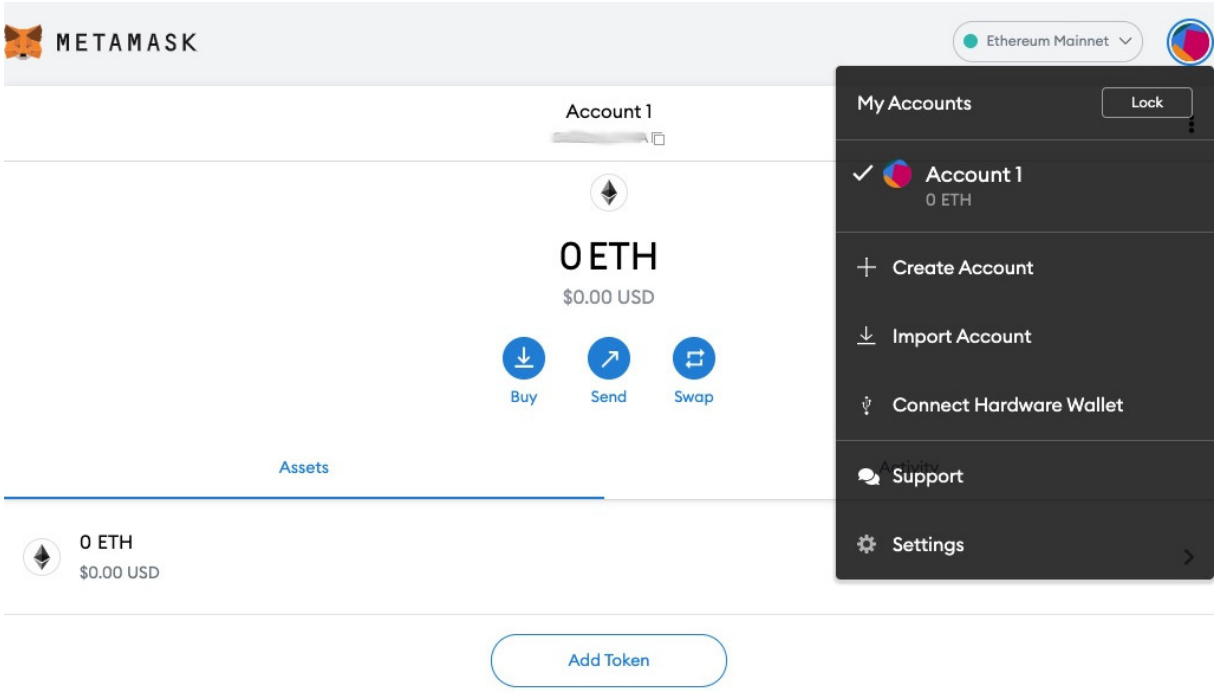


Figura 1.9: MetaMask (crypto wallet)

## 1.3 Sistemi Centralizzati, Decentralizzati, Distribuiti

Il software e i sistemi centralizzati al giorno d'oggi sono i più diffusi. In ogni rete i componenti interagiscono tra di loro al fine di raggiungere un obiettivo comune.

Nei sistemi centralizzati, le decisioni per il raggiungimento di un obiettivo vengono prese da un'entità centrale e fissa e quindi trasmesse ai vari componenti, come mostrato nella Figura 1.10.

In un sistema distribuito i dati non risiedono in un unico nodo centrale, ma sono distribuiti su più nodi.

Un sistema centralizzato potrebbe essere progettato per essere anche distribuito. In questo caso, tutti i nodi saranno controllati da un'autorità centrale che potrà decidere e garantire il corretto funzionamento.

In un sistema decentralizzato invece ogni nodo ha la stessa importanza e non esiste un'entità centrale dominante con poteri di decisione.

Le applicazioni decentralizzate sono applicazioni eseguite su una rete peer to peer (P2P) di computer, anziché su un unico computer. Possono essere considerati una forma di software progettato per funzionare su Internet senza essere controllato da una singola entità.

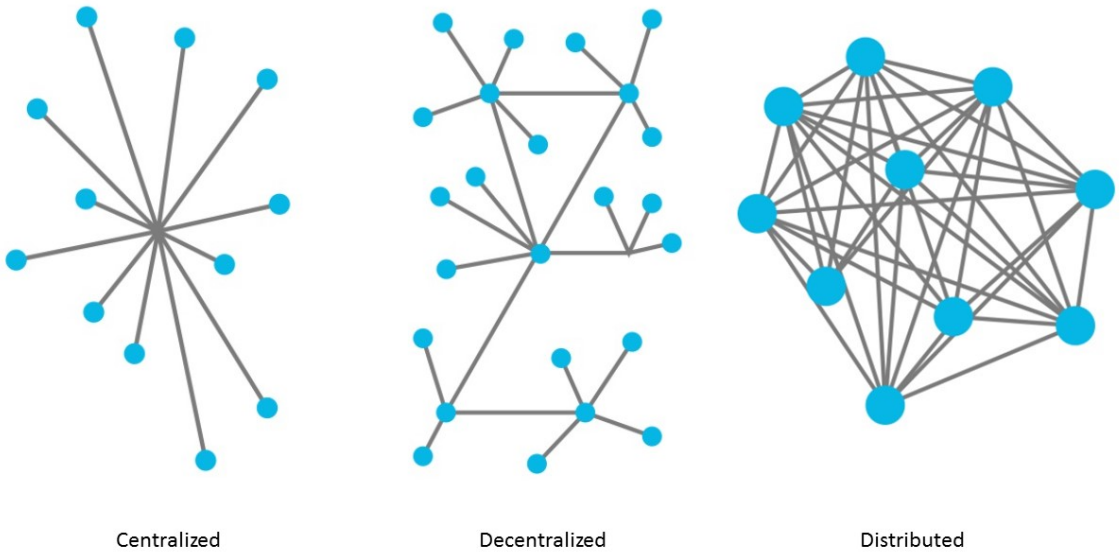


Figura 1.10: I diversi tipi di rete

## 1.4 Le DApps

Le DApps sono applicazioni simili alle app tradizionali, con la differenza fondamentale che al posto di appoggiarsi su server centralizzati vengono eseguiti su un sistema informatico decentralizzato, Blockchain o altro sistema di contabilità distribuita. Parliamo, quindi, di una rete in cui i suoi utenti hanno il pieno controllo del suo funzionamento.

Le prime DApps conosciute sono state viste nei protocolli di condivisione file come *BitTorrent* o *DC++*. Entrambe le applicazioni sono sistemi di condivisione di file peer-to-peer con un'elevata resistenza alla censura.

### 1.4.1 Differenze tra una DApp e un'app tradizionale

Le DApps e le app tradizionali hanno molti elementi in comune, tuttavia, la loro differenza sta nel modo in cui interagiscono con questi elementi. Entrambi i tipi di applicazioni posseggono tre strutture di base che sono: il frontend, il backend e il livello di archiviazione dei dati.

**Frontend**
Il primo livello, il frontend, è l'interfaccia che gli utenti utilizzano per interagire con l'applicazione. In questo caso, sia le DApps che le App tradizionali possono utilizzare le vaste risorse grafiche esistenti. Lo scopo di questo livello è quello di dare all'utente la possibilità di interagire, ricevere e inviare informazioni all'applicazione che sta utilizzando (Figura 1.11).

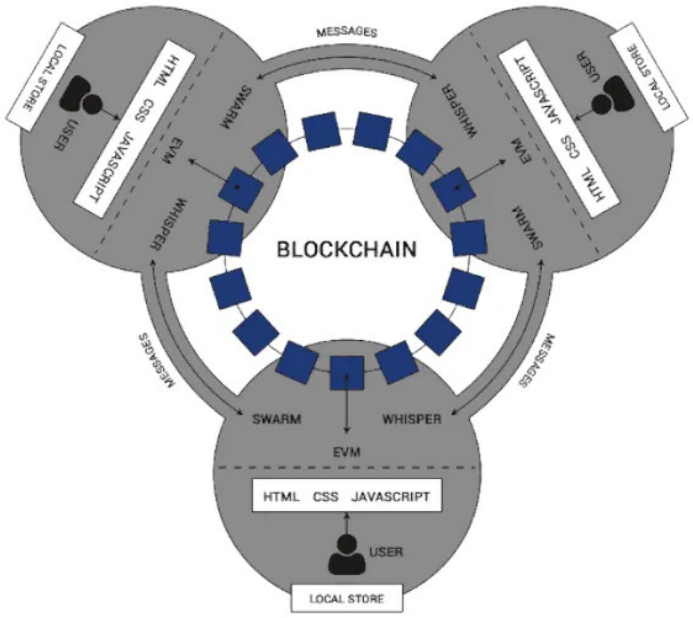


Figura 1.11: Frontend

**Backend**

Questo secondo livello si riferisce alla logica principale dell'applicazione. In un'app tradizionale, questa logica è centralizzata, a differenza delle DApps in cui è decentralizzata. Nelle DApps, il backend è correlato ad uno smart contract (contratto intelligente) che gira su una Blockchain, ad esempio Ethereum. Poiché i contratti intelligenti sono visibili e pubblici, ciò garantisce un elevato livello di trasparenza e sicurezza. Gli utenti possono essere certi che la DApp non farà nient'altro che ciò che specifica lo smart contract.

**Archivio dati**

Infine, c'è il livello d'archiviazione. In un'applicazione tradizionale anche questo livello è centralizzato. Normalmente i dati vengono conservati sul computer dell'utente o su server controllati da terze parti. Questo modo di lavorare ha molti punti deboli. Un utente, ad esempio, potrebbe perdere le informazioni sull'applicazione se il suo computer viene danneggiato. Può anche accadere che i server si interrompano o vengano bloccati. Queste azioni impedirebbero all'utente di utilizzare correttamente l'applicazione o addirittura di perdere informazioni.

In una DApp, l'archiviazione dei dati è completamente decentralizzata. Ogni utente della DApp memorizza una cronologia completa delle azioni eseguite sulla rete DApp, inoltre, le interazioni vengono memorizzate nella Blockchain all'interno di blocchi. Tutto questo avviene in modo crittograficamente sicuro, impedendo l'accesso non autorizzato da parte di terzi. In questo modo, se il computer o lo smartphone di un utente dovessero essere danneggiati, basterebbe utilizzare la DApp su un nuovo dispositivo per recuperare tutte le sue informazioni immagazzinate fino a quel preciso momento.

### 1.4.2 Classificazione delle DApps

Le DApps possono essere classificate in tre categorie, che sono:

1. **Tipo I**: sono tutte quelle DApps che hanno la propria Blockchain. Cioè, quelli che hanno la propria infrastruttura e non dipendono da alcuna Blockchain per essere eseguiti.

2. **Tipo II**: in questa classificazione troviamo quelle DApps che dipendono da una Blockchain e dalle sue caratteristiche per funzionare. In questo caso, le DApps possono funzionare utilizzando i propri token o la Blockchain su cui girano.

3. **Tipo III**: le DApps di tipo III utilizzano i token delle DApps di tipo II per svolgere le loro operazioni. Un esempio è *Safe Network* che fa affidamento a *Omni Layer*, una DApp di tipo II, per generare *Safecoin*, la propria criptovaluta.

### 1.4.3 Vantaggi e svantaggi delle DApps

Le DApps presentano, quindi, alcuni vantaggi che possono potenzialmente risolvere alcuni problemi delle applicazioni centralizzate: sono resistenti, non esiste un single point of failure; sono immutabili, nessuno può modificare quello che avviene tramite l'applicazione; gli asset o le informazioni personali possono essere totalmente di proprietà degli utenti.

Sfruttando queste caratteristiche, le opportunità di sviluppo sono molteplici. Alcuni esempi potrebbero essere: siti di informazione resistenti alla censura, social network in cui i dati non possano essere censurati e rimangano di proprietà degli utenti, piattaforme per la registrazione di copyright e proprietà di asset digitali o anche app di messaggistica decentralizzate.

L'uso delle DApps è ancora nelle fasi iniziali, quindi è sperimentale e soggetto a determinati problemi e incognite. Ci sono domande sul fatto che le applicazioni saranno in grado di scalare in modo efficace, in particolare nel caso in cui un'app richieda calcoli significativi e sovraccarichi una rete, causando una congestione della stessa.

La sfida di apportare modifiche al codice è un'altra limitazione delle DApps. Una volta distribuita, una DApp avrà probabilmente bisogno di modifiche continue allo scopo di apportare miglioramenti o correggere bug o rischi per la sicurezza. Secondo Ethereum, può essere difficile per gli sviluppatori apportare gli aggiornamenti necessari alle DApps perché i dati e il codice pubblicati sulla Blockchain sono difficili da modificare.

# Capitolo 2

# Autenticazione sicura

Il processo di autenticazione consiste nel confermare l'identità di un utente che richiede l'accesso ad un sistema. L'autenticazione è importante perché impedisce ad utenti non autorizzati di poter accedere ad informazioni sensibili, di conseguenza c'è bisogno di sistemi di autenticazione sicuri che non mettano a rischio i dati degli utenti e delle organizzazioni. Tale processo si basa sull'inserimento di dati che si riferiscono ad una categoria o ad una combinazione di più categorie, in questo caso si parla di autenticazione a più fattori.

Le categorie interessate sono 3:

1. Conoscenza: qualcosa che l'utente conosce, ad esempio username e password.
2. Possesso: qualcosa che l'utente possiede, ad esempio una smartcard.
3. Identità: qualcosa che l'utente è, ad esempio l'impronta digitale.

Il processo di autenticazione base, confronta i dati che vengono immessi dall'utente con ciò che è memorizzato nel sistema a cui si tenta di accedere.

Ogni processo di autenticazione si suddivide in vari sotto-processi che riguardano:

- l'identificazione: l'utente che tenta di accedere deve rispondere alla domanda: *"chi sei tu?"*, inserendo i dati richiesti al fine dell'identificazione, ad esempio l'username;
- l'autenticazione: il sistema ha bisogno di trovare un metodo per comprendere se l'utente è chi dice di essere, in questo caso si deve rispondere alla domanda: *"come puoi dimostrare che sei tu?"*, per rispondere a questa domanda c'è bisogno di inserire un dato legato all'identificazione, ad esempio una password.
- l'autorizzazione: una volta eseguiti i passaggi precedenti bisogna stabilire l'utente cosa può fare, ciò rientra nella procedura di controllo degli accessi.

Esistono svariati meccanismi di autenticazione che verranno trattati nei successivi paragrafi.

## 2.1 Autenticazione basata su password

L'autenticazione basata su password è uno dei metodi di autenticazione più semplice, rientra nella categoria "conoscenza" in quanto consiste nel verificare l'identità di un utente nel sistema attraverso l'utilizzo dei dati che possiede; in questo caso la combinazione di username e password associata (Figura 2.1).

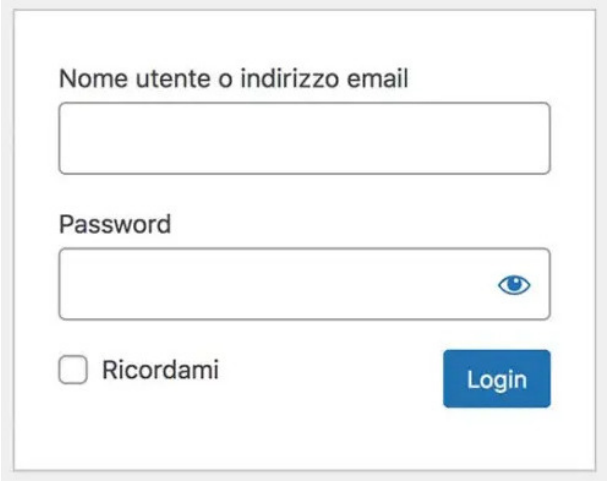


Figura 2.1: Username e Password

Le problematiche relative a questo metodo di autenticazione riguardano principalmente la scelta delle password da parte degli utenti, in quanto:

- la scelta di password semplici comporta diversi rischi perché sono facili da indovinare, infatti, i malintenzionati potrebbero impossessarsi facilmente della password utilizzando metodi di forza bruta (brute force);
- password riutilizzate in svariati siti aiuterebbero un hacker nel violare un server di un sito, rubare le password e se l'utente ha utilizzato la stessa password in più siti potrebbe avere l'accesso anche a quest'ultimi.
- condivisione delle password, questo comporta rischi per la sicurezza, in quanto potrebbero essere intercettate;
- le password che sono difficili da ricordare possono essere dimenticate facilmente.

La solidità di una password si può valutare attraverso tre fattori: lunghezza, cardinalità ed entropia. Ad esempio, nella maggior parte dei siti si prevede l'inserimento di una password che rispetti uno standard preciso: lunghezza di almeno 8 caratteri, presenza di caratteri maiuscoli, minuscoli e speciali.

| Password Length | Numerical 0-9 | Upper & Lower case a-Z | Numerical Upper & Lower case 0-9 a-Z | Numerical Upper & Lower case Special characters 0-9 a-Z %$ |
|---|---|---|---|---|
| 1 | instantly | instantly | instantly | instantly |
| 2 | instantly | instantly | instantly | instantly |
| 3 | instantly | instantly | instantly | instantly |
| 4 | instantly | instantly | instantly | instantly |
| 5 | instantly | instantly | instantly | instantly |
| 6 | instantly | instantly | instantly | 20 sec |
| 7 | instantly | 2 sec | 6 sec | 49 min |
| 8 | instantly | 1 min | 6 min | 5 days |
| 9 | instantly | 1 hr | 6 hr | 2 years |
| 10 | instantly | 3 days | 15 days | 330 years |
| 11 | instantly | 138 days | 3 years | 50k years |
| 12 | 2 sec | 20 years | 162 years | 8m years |
| 13 | 16 sec | 1k years | 10k years | 1bn years |
| 14 | 3 min | 53k years | 622k years | 176bn years |
| 15 | 26 min | 3m years | 39m years | 27tn years |
| 16 | 4 hr | 143m years | 2bn years | 4qdn years |
| 17 | 2 days | 7bn years | 148bn years | 619qdn years |
| 18 | 18 days | 388bn years | 9tn years | 94qtn years |
| 19 | 183 days | 20tn years | 570tn years | 14sxn years |
| 20 | 5 years | 1qdn years | 35qdn years | 2sptn years |

Figura 2.2: Tempo di violazione di una password

Nella Figura 2.2 sono presenti i dati relativi al tempo impiegato per violare una password con un attacco di forza bruta, che consiste nell'iterare le varie combinazioni possibili fino a trovare la chiave corretta. Come si evince in figura, la combinazione di numeri, caratteri maiuscoli, minuscole e speciali appare essere la combinazione più efficace.

L'entropia permette di calcolare la forza di una password in bit, basandosi sul set di caratteri utilizzati e sulla lunghezza della password. Una password, per essere sicura, dovrebbe essere lunga almeno 8 caratteri, mista, non avere un senso ed essere diversa in ogni sito su cui si è registrati. Anche se l'utente ha rispettato tutte le linee guida sulla scelta di una password robusta, ci sono comunque problemi che non sono direttamente imputabili all'utente come nel caso di violazioni di siti che potrebbero comportare il furto di password. Questo accade soprattutto nel caso di sistemi centralizzati, quindi sono necessari ulteriori meccanismi di autenticazione che saranno trattati nei prossimi paragrafi.

## 2.2 Autenticazione a 2 fattori e verifica in 2 passaggi

Come già descritto precedentemente, in un processo di autenticazione c'è bisogno di scegliere quali sono i dati da chiedere all'utente tra le 3 categorie proposte. Nel caso in cui l'autenticazione avviene con la sola password: si tratta di autenticazione ad un fattore.
Si parla, invece, di autenticazione a due fattori (2FA) se si usano almeno due delle tre categorie elencate (conoscenza, possesso, identità). Ma non basta: la condizione affinché si possa definire "autenticazione a due fattori" si verifica solo quando i due fattori utilizzati sono di matrice differente: in altre parole se, per esempio, si usa "*Una cosa che sai*" + "*Una cosa che hai*", come mostrato nella Figura 2.3.

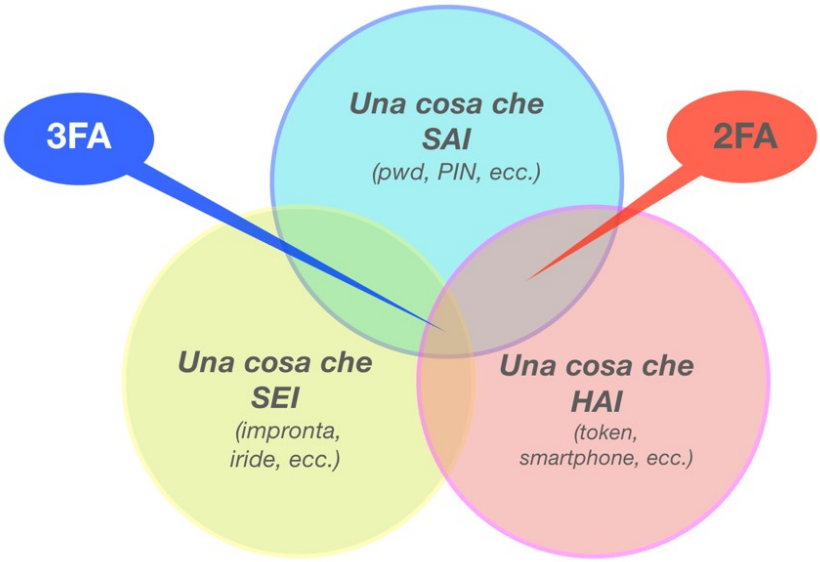


Figura 2.3: Autenticazione a due fattori

Per soddisfare la prima parte dell'**autenticazione a due fattori** bisognerà sfruttare qualcosa che si conosce, come ad esempio un password, oppure un codice OTP. Per soddisfare la seconda parte, invece, (ovvero il secondo fattore) bisognerà utilizzare qualcosa che si possiede (come un'applicazione di autenticazione, una keycard o un secondo dispositivo), oppure la verifica di un elemento unico e inimitabile (come la scansione dell'iride, dell'impronta digitale o altro).

La **verifica in due passaggi**, invece, prenderà in considerazione soltanto qualcosa che si conosce che sarà verificata due volte (due fattori della stessa categoria). Ad esempio: l'inserimento di una password e l'utilizzo di un codice OTP ricevuto per SMS. La verifica a due passaggi è generalmente utilizzata per aumentare la sicurezza di account online relativi a social network, app bancarie e altro, mentre l'autenticazione a due fattori viene perlopiù sfruttata per account aziendali, o comunque per account contenenti informazioni molto sensibili.

L'autenticazione a due fattori si può ottenere in vari modi.

- **Attraverso un SMS**. In questa tipologia di autenticazione potrebbe essere utilizzato nel primo fattore l'utilizzo dell'username/password e successivamente l'invio di un SMS sul proprio smartphone che rientra nella categoria "possesso". Questo approccio è il meno sicuro ed è soggetto a truffe come la SIM Swap Fraud (Figura 2.4) dove un malintenzionato può ottenere l'accesso completo al numero di telefono del legittimo proprietario con la conseguente ricezione degli SMS tra i quali i codici per l'autenticazione a due fattori.

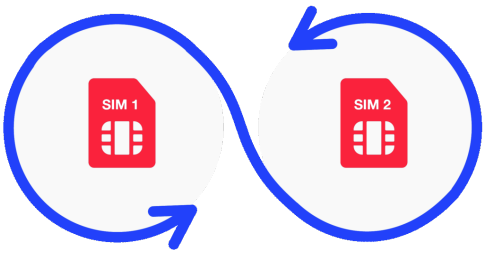


Figura 2.4: Sim Swapping Attack

- **Attraverso l'utilizzo di un'App Authenticator**, un metodo che permette di sostituire l'utilizzo degli SMS. Sullo smartphone dell'utente è installata questa tipologia di applicazione (la Figura 2.5 mostra un esempio di processo a due fattori utilizzando l'applicazione Google Authenticator); quando l'utente tenta di accedere al servizio online o in generale tenta di eseguire una transazione, il servizio online richiede all'app che è associata all'utente la conferma della transazione. L'utente deve confermare la transazione accedendo all'app tramite PIN, impronta digitale o riconoscimento facciale. Questa tipologia di autenticazione rientra in un livello di autenticazione a 3 fattori in quanto utilizza:

  1. il servizio a cui tenta di accedere attraverso i dati ("*qualcosa che si sa*)";
  2. lo smartphone ("*qualcosa che si ha*");
  3. l'accesso all'autenticazione biometrica ("*qualcosa che si è*").

  I problemi legati a questo metodo riguardano alcune limitazioni sull'uso dello smartphone quali la perdita di quest'ultimo o l'esaurimento della batteria.

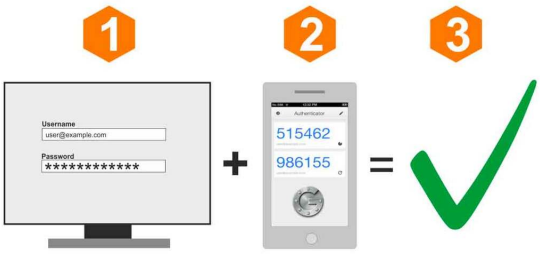


Figura 2.5: Autenticazione a due fattori con Google Authenticator

- **Attraverso lo standard FIDO2**. Il processo di autenticazione utilizza una Security Key (Figura 2.6), i costi di questi dispositivi variano tra i 20 e i 60 dollari. La Security Key permette di eseguire due azioni: registrazione e autenticazione.

Figura 2.6: Security Key

Nella fase di registrazione, si potrebbe ipotizzare l'utilizzo di un sistema basato su username/password con la successiva registrazione della Security Key. Per registrare la Security Key c'è bisogno di inserirla nel dispositivo che si sta utilizzando per accedere al servizio online. Il servizio online invia una stringa pseudo-casuale al browser che identifica la Security Key ed inoltra la stringa, successivamente avviene la creazione da parte della Security Key di una chiave pubblica + privata: la chiave privata servirà per firmare i documenti, che verranno inviati insieme alla chiave pubblica al servizio online attraverso il browser.

Nella fase di autenticazione ci sarà l'invio della stringa pseudo-casuale e l'id della chiave pubblica associata all'utente. La Security Key verificherà, dopo la pressione dell'utente sul bottone, l'identificativo e firmerà i dati ricevuti con la chiave privata inviandoli nuovamente. A questo punto il servizio online verificherà attraverso la chiave pubblica che la firma effettuata sia corretta e abiliterà la transazione. In questa tipologia di autenticazione potrebbero sussistere problemi in quanto un malintenzionato potrebbe cercare di simulare il servizio online.

## 2.3 Sistema di riconoscimento biometrico

I sistemi di riconoscimento biometrico utilizzano caratteristiche fisiche per poter identificare l'utente come: le impronte digitali, riconoscimento vocale, rilevamento del volto e scansione della retina (Figura 2.7). Il sistema di riconoscimento biometrico viene eseguito memorizzando in una fase iniziale il "template" dell'utente che consiste nel registrare la caratteristica presa in esame per la fase di autenticazione. Una volta eseguita la fase di registrazione del "template", l'utente per accedere al sistema ha bisogno di mostrare la caratteristica biometrica presa in esame che il sistema confronterà con il template registrato.

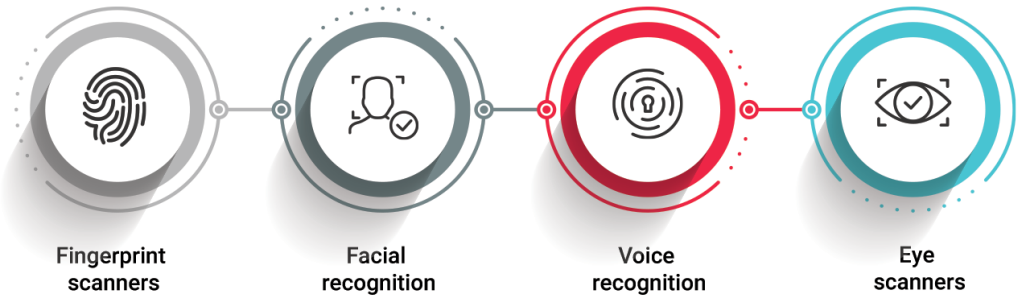


Figura 2.7: Tipi di autenticazione biometrica

Le problematiche relative a questo metodo di autenticazione riguardano i falsi positivi: il dato preso in esame potrebbe essere associato al "template" di un utente ma, invece, appartiene ad un'altra persona. Infatti, in alcuni casi possono essere generati da manomissioni come la realizzazione di impronte digitali con gelatina. Per questo motivo, la questione relativa alla privacy è di fondamentale importanza in quanto essa potrebbe essere messa a rischio da Data Breach dei template biometrici.

Per quanto riguarda i falsi negativi, si verifica che un utente tenta di accedere al suo profilo e non viene riconosciuto, infatti, per riconoscere i dati biometrici vengono utilizzati dei dispositivi che potrebbero generare delle problematiche se acquisiscono informazioni in condizioni differenti da quelle registrate nel template (ad esempio, se l'utente ha registrato il suo volto utilizzando una luce frontale e durante le successive autenticazioni l'illuminazione varia di orientazione o intensità, questo potrebbe dar luogo ad errori di riconoscimento).

## 2.4 Sistemi a chiave pubblica e privata

Questo sistema di autenticazione permette di fornire un meccanismo alternativo a quello delle password (mostrato nella Figura 2.8). L'utente disponde di una chiave privata e di una chiave pubblica, queste due chiavi sono legate da un algoritmo crittografico che permette dalla chiave privata di ricavare la chiave pubblica ma non il viceversa. Nel sistema sul quale l'utente tenta di accedere sono conservati il nome utente e la chiave pubblica. L'utente tenta di accedere al sistema attraverso l'username, a questo punto il sistema invierà una stringa pseudocasuale. L'utente utilizza questa stringa cifrandola insieme alla chiave privata, una volta eseguito questo passaggio, verrà inviata al sistema che, attraverso la chiave pubblica, decifrerà la stringa per verificare se il risultato corrisponde alla stringa inviata.

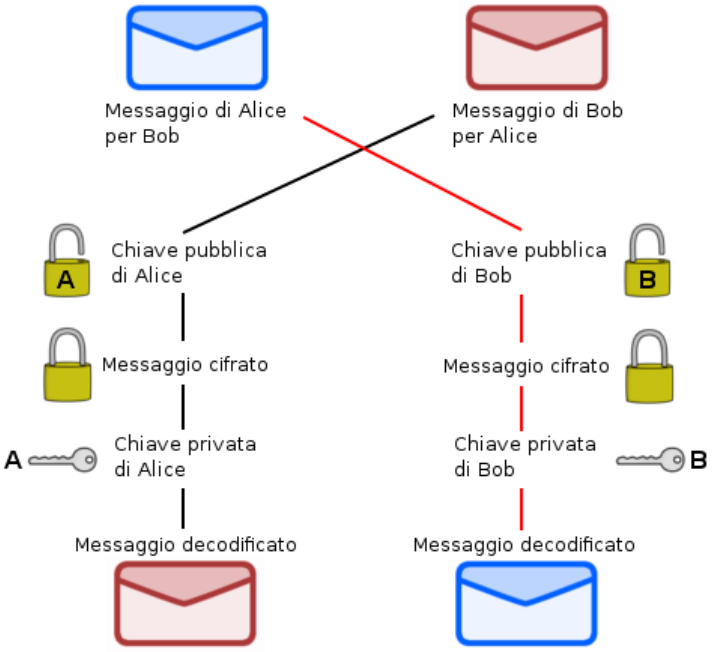


Figura 2.8: Schema crittografia asimmetrica

I vantaggi di questo metodo risiedono nell'esposizione ai dati privati, in quanto, il sistema non archivia nulla di segreto. Le problematiche riguardano la chiave privata, rappresentata da una stringa molto lunga e che l'utente deve conservare con cura.

Le soluzioni per gestire le chiavi private convergono sull'utilizzo di smartcard. Nella smartcard è presente una chiave privata che non può essere estratta, in questa circostanza l'utente deve prestare attenzione a non smarrire la smartCard che potrebbe essere motivo di furto. In alcuni contesti esiste l'introduzione di un ulteriore fattore di autenticazione, come nel caso delle carte di credito dove viene aggiunto un PIN che funge da password e che l'utente deve digitare per poter eseguire operazioni bancarie una volta inserita la carta nell'apposito dispositivo.

## 2.5 CAPTCHAs

I CAPTCHAs (Figura 2.9) nascono per verificare se chi sta tentando di accedere ad una risorsa sia umano o meno. Questo processo è stato introdotto in quanto gli hacker utilizzano programmi sempre più sofisticati. Gli utenti devono inserire ciò che vedono attraverso un'immagine distorta di lettere e numeri. Le problematiche rilevative a questo metodo riguardano principalmente le persone affette da disabilità (ad esempio visive), in quanto potrebbero non riuscire a superare il controllo CAPTCHA. Molti servizi web utilizzano i CAPTCHA, tra i quali spicca Google che integra i CAPTCHA nelle situazioni come la registrazione ad un nuovo servizio e la modifica della password di un account esistente.

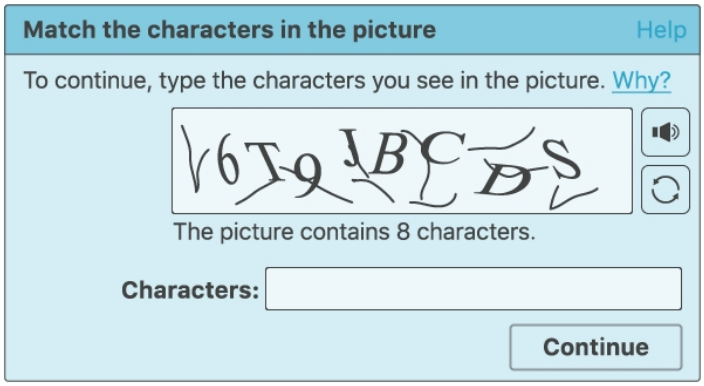


Figura 2.9: CAPTCHA

## 2.6 OAuth

OAuth è un protocollo aperto nato nel 2006 che ambisce a sostituire i meccanismi di autenticazione tradizionali. Un server autorizzativo (gestito da un soggetto terzo come Google, Microsoft, Facebook e così via) fornisce un token di accesso al sito o all'applicazione web che lo richiede previa esplicita approvazione dell'utente. Quando si effettua l'accesso con OAuth, il gestore del server autorizzativo (service provider) mostra l'elenco delle informazioni che sono state richieste dal client ovvero dal sito web sul quale l'utente desidera autenticarsi.

Dopo aver effettuato l'autenticazione sul service provider, in maniera totalmente indipendente rispetto all'applicazione web alla quale si vuole accedere senza procedere con una registrazione di tipo tradizionale, il server autorizzativo fornirà all'applicazione web di terze parti un token di accesso (access token) che può essere utilizzato per riconoscere l'utente e autenticarlo.

I principali service provider usano oggi OAuth 2.0 ovvero la più recente versione del protocollo approvata come standard a ottobre 2012. Rispetto al predecessore, OAuth 2.0 riduce di default la scadenza del token con la possibilità comunque di aumentarne la durata e implementa un mediatore tra client e server. Il mediatore gestisce i token di accesso e può essere lo stesso soggetto che offre le risorse alle quali l'utente intende accedere.

Il permesso Authorization Code (Authorization Code grant), utilizzato per accedere ai vari siti web con OAuth 2.0, prevede che il service provider assegni all'applicazione di terza parte un identificativo e un segreto (client ID e client secret) che verranno scambiati durante il processo di autorizzazione.

L'applicazione, al momento della registrazione presso il service provider, fornisce un nome e un redirect URI che sarà utilizzato dal server autorizzativo per informare circa l'avvenuta approvazione della richiesta da parte dell'utente (Figura 2.10).

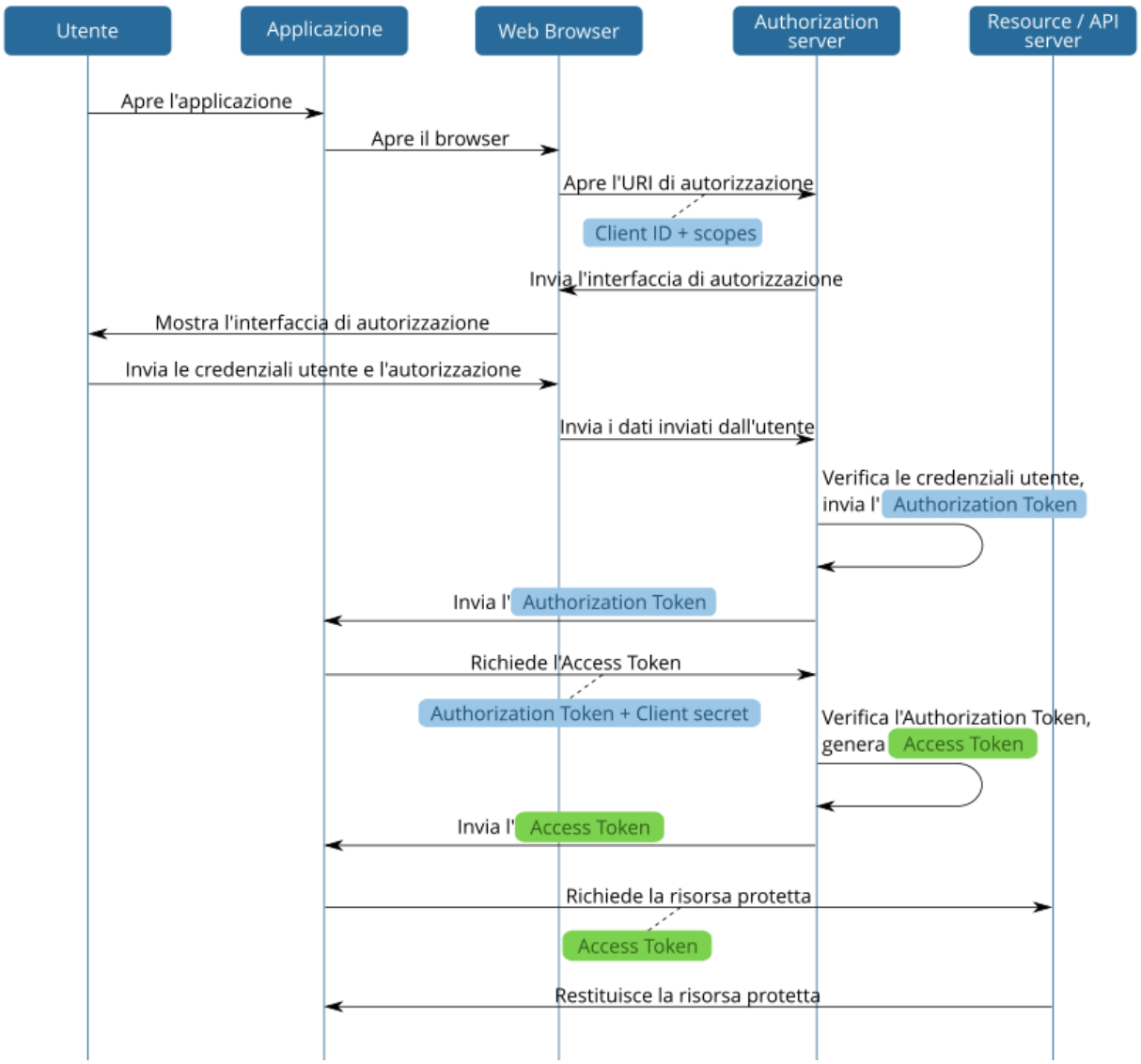


Figura 2.10: Flusso Authorization Code grant

### 2.6.1 Attori in OAuth

In OAuth vi sono i seguenti attori principali:

- l'Utente o Resource Owner: la persona che può dare l'accesso ad una qualche porzione del proprio account. È la persona che può garantire l'accesso alle risorse protette presenti sul "Resource Server".
- il Client o Consumer: è l'applicazione che cerca di ottenere l'accesso all'account dell'utente e che richiede il permesso dell' "Utente" prima di poter accedere alle risorse protette.
- il Resource Server o Service Provider: è il server utilizzato per accedere alle risorse protette dell'utente.
- l'Authorization Server: è il server che presenta l'interfaccia dove l'Utente approva o nega la richiesta di accesso. In piccoli ambienti può coincidere con il Resource Server.

Il Client interagisce con il Resource Server per ottenere delle credenziali temporanee. Per accedere alle risorse protette, il Client richiede all'Utente il permesso, detto token di accesso. Una volta ottenuto il token di accesso può accedere e interagire con le risorse stabilite per un breve periodo.

## 2.7 OpenID

OpenID è un protocollo di autenticazione decentralizzato ideato per autenticarsi in un servizio (service provider) riutilizzando account già esistenti da altri siti web (identity provider). Lo scopo è di evitare che ogni utente necessiti di registrare un account in ogni fornitore di servizi. L'identificativo di un utente OpenID è rappresentato da un URL.

La **procedura di registrazione** consiste di un modulo di registrazione in un sito web che richieda informazioni di base del profilo come l'indirizzo e-mail, il nome, la data di nascita, ecc. Può permettere all'utente di pre-compilare questi campi ricevendoli da un sito web in cui l'utente sia già registrato (identity provider). Ciò abbatte il tempo che intercorre fra l'inizio di una registrazione e la fruizione del servizio.

In fase di **autenticazione** l'utente non inserisce alcuna combinazione di nome utente e password dedicata a tale servizio, dato che l'autenticazione è delegata all'identity provider scelto dall'utente. OpenID è stato ideato per evitare di ricordare molteplici combinazioni di nome utente e password, sia per evitare frustrazione per gli utenti, sia per evitare che si sia indotti a riutilizzare una stessa password per ogni servizio, comportando un rischio per la sicurezza. La verifica dell'identità avviene sul sito OpenID scelto dall'utente.

Per quanto riguarda la **raccolta delle informazioni**, l'utente è nella posizione di decidere quali informazioni personali l'identity provider debba trasmettere al singolo fornitore di servizi.

### 2.7.1 OpenID Connect

OpenID Connect (OIDC) è un protocollo che combina le funzionalità di OpenID e OAuth 2.0, quindi l'autenticazione e l'autorizzazione. Il flusso di esecuzione di OpenID Connect in generale comprende: una richiesta iniziale da parte di un utente o del proprietario della risorsa. Questa richiesta viene istanziata, attraverso un client, per accedere ad una risorsa protetta. Il client invia una richiesta al server che si occupa dell'autorizzazione. Per accedere, l'utente individua l'identity provider dove effettuare l'accesso e viene reindirizzato alla pagina di autenticazione. Una volta effettuato l'accesso viene reindirizzato alla risorsa originale con un codice temporaneo. Il Client invia un'altra richiesta al server di autorizzazione con l'ID Client e il codice di autorizzazione, il server verifica le informazioni e invia un Access Token e ID Token. A questo punto da parte del client si può richiedere la risorsa allegando il token di accesso.

# Capitolo 3

# Autenticazione in KryptoAuth

Nel seguente capitolo andremo ad analizzare le tecnologie e l'implementazione che hanno reso possibile lo sviluppo della Web DApp KryptoAuth. L'obiettivo del progetto è stato di realizzare un'applicazione web che permettesse all'utente di autenticarsi in modo sicuro tramite la tecnologia Blockchain Ethereum, realizzando in tal modo un sistema decentralizzato dove l'utente ha il pieno controllo dei suoi dati.

## 3.1 La Blockchain

La Blockchain è un libro mastro pubblico distribuito (distribuited public ledger) di transazioni o eventi digitali eseguiti e condivisi tra i partecipanti. Come si può vedere in Figura 3.1, la Blockchain è costituita da una catena di blocchi. I blocchi rivestono un ruolo importante in quanto al loro interno sono immagazzinate le transazioni che avvengono sulla rete. Una volta memorizzate, le informazioni non possono essere né cancellate né modificate. Ogni blocco presenta al suo interno il valore hash del blocco precedente, questo consente di collegare i vari blocchi della catena. Gli hash permettono di rendere la Blockchain immutabile, in quanto, una modifica di un blocco comporterebbe la modifica del valore dell'hash. L'aggiunta dei blocchi viene stabilita attraverso dei meccanismi di consenso stabiliti dai nodi della rete e che permettono anche di verificare le transazioni.

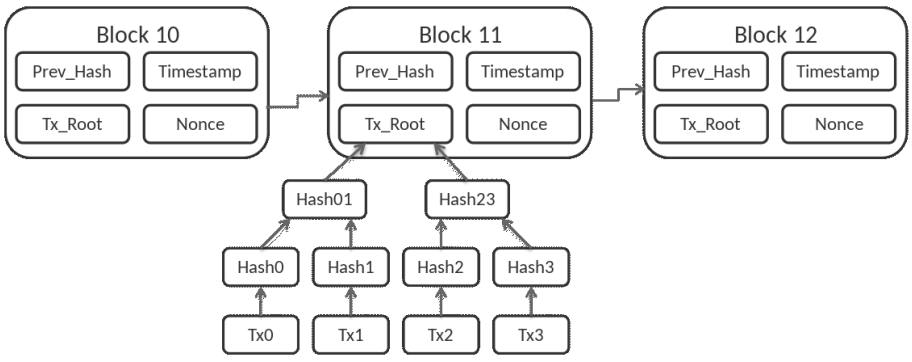


Figura 3.1: Struttura blocchi Blockchain

### 3.1.1 Caratteristiche della Blockchain

Una Blockchain presenta le seguenti caratteristiche:

- **Decentralizzazione**: è una delle caratteristiche principali. Si parla di decentralizzazione dell'infrastruttura quando non esiste un singolo controllo dell'infrastruttura che comporterebbe un singolo punto di failure.
- **Trasparenza**: ogni nodo mantiene la stessa copia del ledger.
- **Immutabilità** dei dati.
- **Consenso**: per poter essere valide, le transazioni devono essere validate dai partecipanti.
- **Provenienza**: i partecipanti conoscono la provenienza degli assets.

Partendo da questi principi, la Blockchain è diventata la declinazione in digitale di un nuovo concetto di fiducia al punto che alcuni ritengono che la Blockchain possa assumere anche un valore per certi aspetti di tipo "sociale e politico". In questo caso la Blockchain è da vedere come una piattaforma che consente lo sviluppo e la concretizzazione di una nuova forma di rapporto sociale, che grazie alla partecipazione di tutti è in grado di garantire a tutti la possibilità di verificare, di "controllare", di disporre di una totale trasparenza sugli atti e sulle decisioni, che vengono registrati in archivi che hanno caratteristica di essere inalterabili, immodificabili e dunque immuni da corruzione.

### 3.1.2 Come funziona la Blockchain

Ogni volta che avviene una transazione, questa viene registrata come un "blocco" di dati (Figura 3.2). Queste transazioni rappresentano il movimento di un asset che può essere tangibile (un prodotto) o intangibile (intellettuale). Questo blocco di dati può riportare le informazioni che si desiderano: chi, cosa, quando, dove, quanto e persino delle condizioni (come la temperatura di una spedizione di cibo).


Figura 3.2: Blocchi della Blockchain

Ogni blocco è collegato a quelli che lo precedono e che lo seguono. Questi blocchi formano una catena di dati man mano che un asset si sposta da un luogo all'altro o cambia il proprietario. I blocchi attestano l'ora e la sequenza esatte delle transazioni e i blocchi si collegano in modo sicuro tra loro per evitare che uno di essi venga alterato o inserito tra due blocchi esistenti.

Le transazioni sono bloccate tra loro in una catena irreversibile: una Blockchain. Ogni blocco aggiuntivo rafforza la verifica del blocco precedente e quindi dell'intera Blockchain. Questo fa sì che la Blockchain sia a prova di manomissione, offrendo l'elemento chiave dell'immutabilità. Questo elimina la possibilità di manomissioni da parte di malintenzionati e crea un registro di transazioni affidabile.

### 3.1.3 Blockchain e Distributed Ledger Technology

La Blockchain può essere considerata una tecnologia che appartiene alla categoria delle tecnologie *Distributed Ledger* (archivi distribuiti). Le Distributed Ledger Technology o DLT possono essere definite come un insieme di sistemi caratterizzati dal fatto di fare riferimento a un registro distribuito, governato in modo da consentire l'accesso e la possibilità di effettuare modifiche da parte di più nodi di una rete.

Qualunque transazione, ovvero i dati che la rappresentano, è sottoposta ad un meccanismo di firma a doppia chiave asimmetrica che, pur non dotata di certificati rilasciati da certificatori accreditati, funziona con un meccanismo simile a quello della firma digitale. Le DLT prevedono l'utilizzo di algoritmi crittografici che abilitano l'utente all'utilizzo del sistema mettendogli a disposizione una chiave pubblica ed una privata che viene usata per sottoscrivere le transazioni o per attivare gli smart contract o altri servizi collegati alla Blockchain.

Le DLT prevedono pertanto un meccanismo di validazione a sua volta distribuito basato sul concetto del *Consenso*. Le modalità di gestione del consenso rappresentano due fra i principali punti qualificanti della carta d'identità delle tecnologie Distributed ledger. Ed è all'interno di questo insieme che trovano la loro collocazione le Blockchain.

Nella fattispecie, si può dire che le Blockchain sono delle Distributed Ledger technology caratterizzate da un registro strutturato in modo da gestire le transazioni all'interno di una Catena di Blocchi. Dal punto di vista delle "regole di gestione", ciascun blocco si "aggiunge" alla catena sulla base di un processo basato sul Consenso distribuito su tutti i nodi della rete, ovvero con la partecipazione di tutti i nodi che vengono chiamati a contribuire alla validazione delle transazioni presenti in ciascun blocco e alla loro "inclusione" nel registro.

### 3.1.4 Blockchain e sistemi centralizzati

La decentralizzazione, come descritto precedentemente, è una delle caratteristiche principali in alcune tipologie di Blockchain. Per comprendere i concetti della decentralizzazione bisogna comprendere le differenze con il concetto di centralizzazione.

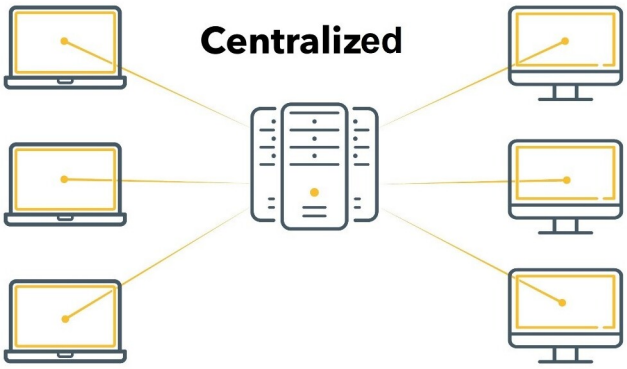


Figura 3.3: Sistema centralizzato

In Figura 3.3 Si può notare l'esempio di un sistema centralizzato, dove il potere è interamente nelle mani di un'autorità centrale. Centralizzare il potere comporta rischi in quanto si è inclini ad attacchi hacker perché un malintenzionato una volta ottenuto il controllo può manipolare i dati ed avere accesso ad informazioni sensibili degli utenti. Inoltre, bisogna garantire che l'autorità centrale sia affidabile ed onesta.

Nel caso della decentralizzazione, ogni cosa è distribuita in un sistema decentralizzato, dalla computazione al processo decisionale. Tutti i partecipanti hanno lo stesso peso e si coordinano tra di loro.

### 3.1.5 Vantaggi della decentralizzazione

I vantaggi legati alla decentralizzazione riguardano:

- **nessun proprietario**: non essendoci un controllo centralizzato si riducono drasticamente i rischi di manipolazione interna da parte di un'autorità centrale;
- **Fault Tolerance**: se un nodo o un gruppo di nodi si spegne questa condizione non impatta sul sistema. Questa è la differenza sostanziale con i sistemi centralizzati dove se, per esempio, un Database riscontra problemi, di conseguenza l'intero sistema riscontrerà dei problemi.
- **Permissionless**: questa condizione è evidente soprattutto nelle Blockchain di tipologia pubblica, dove chiunque può far parte del sistema senza avere permessi da parte di altri membri.

## 3.2 Tipologie di Blockchain

Per far fronte alle specifiche necessità associate ai differenti casi d'uso sono emerse tre caratteristiche fondamentali che agiscono da discriminante per determinare il tipo di una Blockchain: Pubblic, Consortium e Private (Figura 3.4).

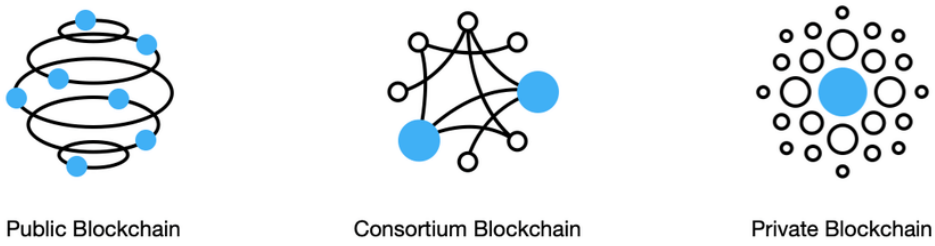


Figura 3.4: Tipologie di Blockchain

Le **Blockchain pubbliche** sono così definite perché non richiedono autorizzazione per l'accesso alla rete, l'esecuzione di transazioni, e la creazione di un nuovo blocco. Gli esempi principali di questo tipo di Blockchain sono senza dubbio Bitcoin ed Ethereum, entrambe catene senza restrizioni di accesso. In questo tipo di Blockchain decentralizzata, come mostra la Figura 3.5, nessun utente ha privilegi sugli altri né facoltà di alterare le informazioni memorizzate sulla catena o di cambiare il protocollo che ne determina il funzionamento. Il problema principale di questo tipo di Blockchain è la sua mancanza di scalabilità: al crescere della quantità di nodi la velocità con cui vengono gestite le transazioni rimane invariata.

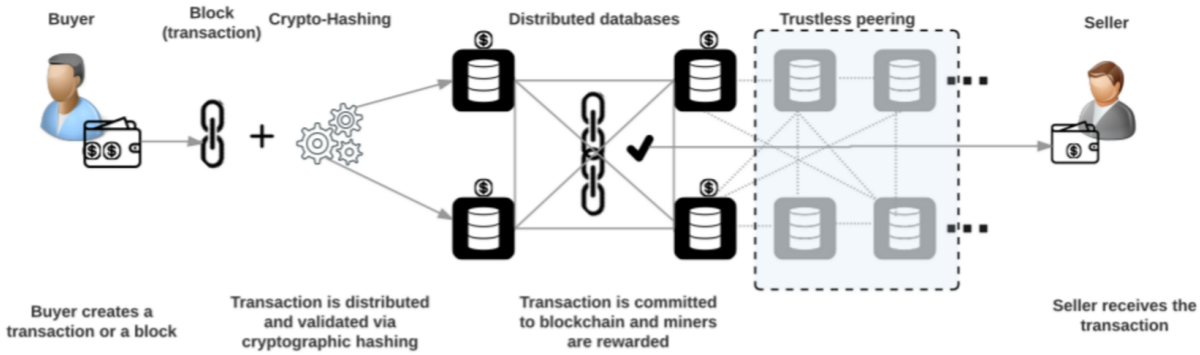


Figura 3.5: Blockchain pubblica

Le **Consortium Blockchain** sono progettate per entità che collaborano tra di loro e che hanno dei rapporti d'affari. Il consenso è determinato da alcuni nodi selezionati, la rete è aperta al pubblico ma non tutti i dati sono accessibili. Un esempio di questa tipologia di Blockchain è presente nel settore finanziario, dove un gruppo di entità decide i ruoli, i permessi ed i partecipanti al network. Questa tipologia di Blockchain è parzialmente decentralizzata e garantisce privacy per quanto riguarda le transazioni. Questo tipo di Blockchain è più veloce, performante, e scalabile rispetto ad una Blockchain pubblica.

Le **Private Blockchain** sono la tipologia di Blockchain maggiormente restrittiva (Figura 3.6). In questa tipologia di Blockchain la privacy riguardante le transazioni e l'immutabilità assumono un ruolo primario a discapito della decentralizzazione. I permessi di scrittura sono centralizzati ad un'unica entità. Si può pensare a questo tipo di Blockchain come ad un tradizionale sistema centralizzato con l'aggiunta di un grado di sicurezza maggiore dettato da componenti crittografiche. I vantaggi offerti dall'ulteriore controllo sulla rete fanno sì che questo tipo di Blockchain sia particolarmente appetibile a società private ed istituzioni finanziarie.

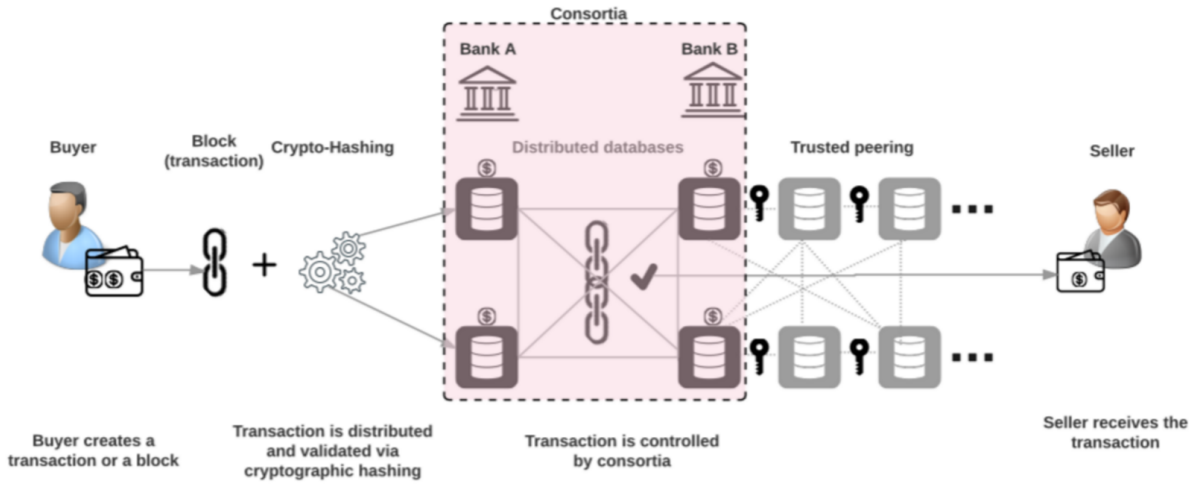


Figura 3.6: Blockchain privata permissioned

### 3.2.1 Blockchain Permissioned e Permissionless

In linea di massima si possono distinguere due tipologie di Blockchain: Permissioned e Permissionless.

In una Blockchain di tipo **Permissionless** (senza autorizzazioni) chiunque può partecipare ed è anonimo, il grado di fiducia verso i partecipanti viene a mancare e per risolvere questa problematica si devono introdurre algoritmi di consenso che incentivino gli utenti ad eseguire i propri compiti.

Nelle Blockchain di tipo **Permissioned** i partecipanti sono noti ed identificati quindi in generale in questa tipologia c'è un grado di fiducia maggiore.

### 3.2.2 Combinazioni tra Blockchain

Queste tre tipologie di Blockchain possono venire assemblate in diversi modi, per rispondere al meglio alle necessità dell'utente. Di seguito vedremo le combinazioni più comuni:

- **Blockchain pubblica permissionless** (senza autorizzazioni), i cui esempi più famosi sono Bitcoin ed Ethereum. Questo tipo di rete permette l'accesso ad ogni utente che decida di connettersi e partecipare, generando nuove transazioni, effettuando il compito di miner o semplicemente leggendo il registro delle transazioni memorizzate. In questo sistema i minatori sono anonimi e, quindi, non vengono considerati individui affidabili.

- **Blockchain pubblica permissioned** (autorizzata), come ad esempio Ripple e Hyperledger Fabric. Si tratta di una rete che opera per conto di una comunità che condivide un interesse comune, dove l'accesso al ruolo di miner è limitato ad un numero esiguo di individui considerati fidati. Il livello di lettura del registro e partecipazione nella generazione di nuove transazioni possono essere soggetti a limitazioni a seconda dell'organizzazione che controlla la Blockchain.

- **Blockchain privata permissioned** (autorizzata), come Chain e Bankchain. In questo caso possono accedere alla rete solo utenti autorizzati ed autenticati poiché la Blockchain in questione opera esclusivamente entro i limiti di una comunità ben definita dove tutti i partecipanti sono noti. Solitamente a capo di questi sistemi si trovano istituti finanziari o agenzie governative che definiscono chi possa accedere o meno alla rete. Questo vuol dire che tutti i miner sono considerati fidati.

## 3.3 Hash

Nella Blockchain i blocchi sono collegati attraverso chiavi hash crittografate. Questa tecnologia viene utilizzata per eseguire un controllo di integrità sui dati rilevando modifiche.

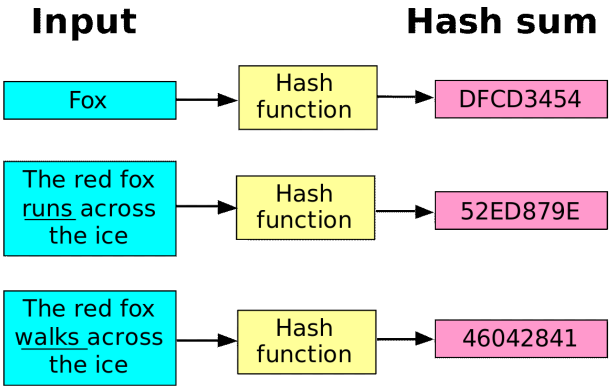


Figura 3.7: Funzione Hash

Le funzioni hash (Figura 3.7) rivestono un ruolo importante in quanto sono funzioni unidirezionali: dall'input si ricava un output ma dall'output non è possibile ricavare l'input. Presentano la caratteristica di essere deterministiche: dallo stesso input si otterrà sempre lo stesso output.

In un blocco ci sono numerose transazioni, tutti gli hash vengono conservati come un *Merkle root tree* che insieme ad altri attributi forma l'unico valore hash per il blocco. Se sussistono cambiamenti di qualche transazione all'interno del blocco ci sarà il cambiamento dell'hash per quella transazione che cambierà di conseguenza il Merkle root hash fino a cambiare l'hash del blocco.

## 3.4 Blocchi

Un blocco è utile perché fornisce un meccanismo per poter collezionare le transazioni. È costituito da due parti: l'intestazione (header) e il corpo (body). Nel body sono presenti tutte le operazioni associate alle transazioni, mentre, l'intestazione è costituita da:

- **Hash del blocco precedente**: permette di creare la catena di blocchi e dare anche un ordine cronologico;
- **Timestamp**: indica la data e l'ora della transazione;
- **Nonce**: un campo che viene incrementato fino a quando il valore dell'hash non è minore del target di difficoltà;
- **Merkle root**: permette di aggregare tutti gli hash delle transazioni presenti nel blocco.

## 3.5 Algoritmi di Consenso

Raggiungere il consenso è una delle tematiche più importanti che interessano la tecnologia Blockchain. Questa tematica rientra nel più ampio caso del problema dei *Generali Bizantini*. Questo problema come il nome suggerisce nasce da un gruppo di generali Bizantini che intendono attaccare una città. Affinché l'attacco abbia un esito positivo, ad attaccare devono essere tutti i generali, quindi la decisione deve essere unanime, o si attacca la città oppure si opta per il ritiro. Di seguito sono elencati alcuni approcci per raggiungere il consenso in ambito Blockchain.

### 3.5.1 Proof-of-Work (PoW)

Uno degli algoritmi più conosciuti è il Proof-of-Work. L'idea di base consiste nell'offrire una ricompensa, in modo da incentivare il lavoro, in cambio della risoluzione di una sorta di enigma matematico che richiede una potenza di calcolo. Il primo nodo che risolve i calcoli necessari riceve la ricompensa. I nodi essendo in competizione tra di loro aumentano la capacità di calcolo della valuta. La problematica riscontrata in questo metodo risiede nei costi elevati in termini energetici.

### 3.5.2 Proof-of-Stake (PoS)

È uno degli algoritmi di consenso introdotti per sopperire alle richieste ingenti di energia che interessano l'algoritmo PoW. Nel PoS vengono scelti degli utenti in modo casuale tra gli utenti che hanno monete: chi ha più monete ha più probabilità di essere eletto. Una volta scelti, ricevono guadagni ed incentivi in cambio della convalida delle transazioni e della creazione di nuovi blocchi. Possono convalidare le transazioni e creare nuovi blocchi all'interno della rete, il che gli consente di ricevere guadagni e incentivi per il lavoro svolto.

## 3.6 Ethereum

Ethereum è la più grande piattaforma software decentralizzata che consente lo sviluppo di smart contracts e applicazioni implementate al di sopra di una Blockchain permissionless. È composta da una rete Blockchain ed una cryptovaluta, chiamata Ether (ETH), usata per pagare gas (un'unità di calcolo utilizzata nelle transazioni e in altre transizioni di stato). Il Wei è la più piccola unità di ETH, e sono necessari 1018 Wei per un ETH (109 Wei compongono un Gwei).

L'algoritmo PoW utilizzato si chiama *Ethash*: un algoritmo di hash appartenente alla famiglia *Keccak*, la stessa di *SHA3*. La Blockchain Ethereum è formata da accounts che interagiscono tra loro tramite messaggi. Ogni account è caratterizzato da uno stato ed un indirizzo a 20 byte che li identifica. Ethereum gestisce due diversi tipi di account:

- **Externally owned accounts** (EOA), accounts in grado di inviare e ricevere ETH ed inviare transazioni agli Smart Contracts;
- **Contract accounts**, sono accounts che, oltre alle funzionalità degli EOA, hanno anche del codice associato che implementano delle azioni triggerate da EOA o altri Contract accounts.

### 3.6.1 Indirizzi Ethereum

Gli **indirizzi della rete** Ethereum incominciano e sono identificati dal prefisso "0x", comune per i numeri numeri in base 16, seguito dai 20 byte più a destra dell'hash Keccak-256 della chiave pubblica ECDSA. Un esempio di indirizzo Ethereum è il seguente: 0x5451bc7Bba9DA0f1cd4d8Dcb863b6857513566A5.

Gli **indirizzi di contratto** sono nello stesso formato, ma sono determinati dal mittente e dal nonce, uno scalare di valore pari al numero di transazione inviato dal mittente. In altre parole, i conti utente sono indistinguibili dai conti-contratto, ai quali viene associato un singolo indirizzo per ciascuno e nessun dato Blockchain. Un utente, al contrario, può avere molteplici livelli di chiave privata, dai quali sono generati altrettanti indirizzi Ethereum. Qualsiasi hash Keccak-256 valido inserito nel formato descritto è valido, anche se non corrisponde a un account con una chiave privata o a un contratto.

### 3.6.2 Come funziona Etherium

Ethereum utilizza nodi che vengono gestiti volontariamente per verificare le transazioni nella rete. I nodi possono contenere l'intera o un segmento della cronologia delle transazioni di Ethereum, le informazioni più recenti sullo stato degli smart contracts, i saldi dei conti e altro ancora. Alla base di Ethereum c'è la Ethereum Virtual Machine (EVM), che è l'ambiente eseguibile e senza fiducia per i contratti intelligenti: protocolli informatici che facilitano, verificano e impongono la negoziazione e l'esecuzione di una sorta di accordo digitale. L'EVM esegue un contratto con qualsiasi regola, inizialmente programmata dallo sviluppatore, come l'invio di denaro da Alice a Bob. L'EVM esegue questi programmi attraverso un linguaggio in bytecode.

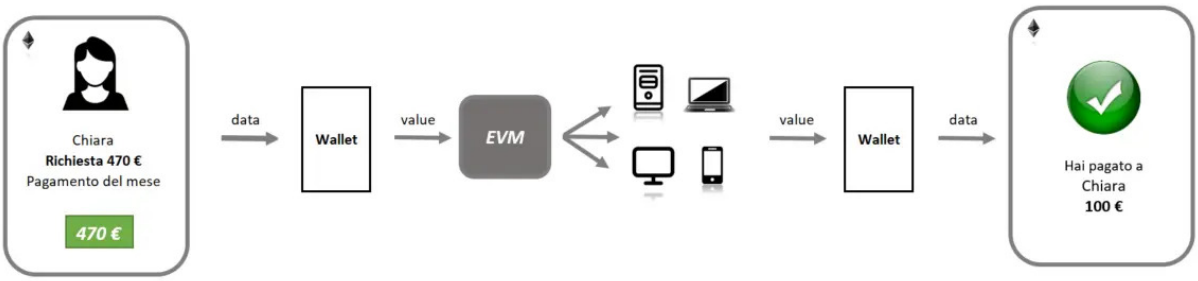


Figura 3.8: Esecuzione di una transazione su Blockchain Ethereum

### 3.6.3 Differenza fra Ethereum e Bitcoin

Ether, la valuta utilizzata per completare le transazioni sulla rete Ethereum, e Bitcoin possiedono molte somiglianze fondamentali. Sono entrambe criptovalute radicate nella tecnologia Blockchain. Ciò significa che i computer indipendenti in tutto il mondo mantengono volontariamente un elenco di transazioni, consentendo di controllare e confermare la cronologia di ciascuna moneta.

Sono entrambe valute virtuali utilizzate attivamente per servizi, contratti e come riserva di valore. La loro natura decentralizzata è un grande cambiamento rispetto alle valute tradizionali, ma esse non sono accettate ovunque. Mentre Bitcoin è accettato più ampiamente e visto come una valuta digitale internazionale, Ether è accettato solo per le transazioni di applicazioni digitali (DApps) eseguite sulla rete Ethereum.

Figura 3.9: Ethereum e Bitcoin

**Bitcoin** rende difficile il furto o la manomissione poiché tutte le macchine sulla rete decentralizzata devono concordare i termini di qualsiasi transazione. Ciò significa principalmente confermare che il beneficiario è il legittimo proprietario della valuta.

La moneta può essere scambiata sul mercato aperto oppure può essere prestata come potenza di calcolo alla rete (mining) e ottenere il pagamento in Bitcoin per l'uso della propria macchina (raccolta, o harvesting).

La quantità massima di Bitcoin che può essere prodotta è di 21 milioni, il che introduce l'argomento della scarsità sul mercato. Al fine di evitare che Bitcoin si esaurisca, nel protocollo sono incorporati eventi di halving per pagare meno Bitcoin ai minatori dopo aver raggiunto un traguardo di raccolta.

**Ethereum** ha esaminato il modo in cui veniva utilizzata la tecnologia Blockchain e ha immaginato come potesse essere usata al di là della semplice valuta. Ether viene estratta allo stesso modo di Bitcoin, ma a differenza di Bitcoin, i miner di Ethereum possono addebitare una commissione per la conferma di una transazione. Inoltre, non c'è limite alla quantità di Ether che può essere rilasciata.

Ethereum e Bitcoin operano su protocolli separati e i loro processi non sono correlati tra loro. Ciò significa che alcune transazioni consentite su una piattaforma potrebbero non essere consentite sull'altra. Questo diventa un quesito fondamentale quando si considerano le transazioni con permessi e senza permessi.

## 3.7 MetaMask

Un wallet memorizza le chiavi pubbliche e private (un utente può avere più indirizzi), che possono essere utilizzati per ricevere o spendere ether. I wallet sono come applicazioni che consentono di interagire con un account Ethereum, analogamente alle app di e-baking e un conto bancario.

Un esempio è MetaMask (Figura 3.10), un portafoglio crittografico e un gateway per le app Blockchain. Disponibile come estensione del browser e come app mobile, MetaMask fornisce un deposito di chiavi, un accesso sicuro, un portafoglio di token e uno scambio di token, tutto ciò di cui l'utente ha bisogno per gestire le sue risorse digitali.

Figura 3.10: Metamask

MetaMask fornisce il modo più semplice ma più sicuro per connettersi ad applicazioni basate su Blockchain. MetaMask genera password e chiavi sul proprio dispositivo, quindi solo l'utente ha accesso ai suoi account e dati: l'utente sceglie sempre cosa condividere e cosa mantenere privato.

A differenza di altri portafogli crittografici, MetaMask è progettato per essere al primo posto nella privacy. Consente di accedere, archiviare e scambiare token, senza che l'utente si preoccupi che le DApps accedano ai dati personali di quelli che l'utente ha acconsentito di fornire. Con MetaMask si ha sempre il controllo completo dei propri dati.

MetaMask richiede che l'utente conservi la sua frase di recupero segreta in un luogo sicuro. È l'unico modo per recuperare i fondi in caso di crash del dispositivo o ripristino del browser. Il metodo più comune è scrivere la frase di 12 parole su un pezzo di carta e conservarla al sicuro in un luogo in cui solo l'utente ha accesso.

## 3.8 Ganache

Ganache è un software open source che ci permette di creare una Blockchain in RAM. La Blockchain che verrà generata sarà una Blockchain realizzata in Javascript, che replica comportamento e caratteristiche della famosa Blockchain di Ethereum.

Ganache (Figura 3.11) è una Blockchain personale per lo sviluppo rapido di applicazioni distribuite su Ethereum e Corda. Si può utilizzare Ganache durante l'intero ciclo di sviluppo; consentendo di sviluppare, distribuire e testare le DApps in un ambiente sicuro e deterministico. Ganache è disponibile in due versioni: interfaccia utente e CLI.

Ganache offre due vantaggi significativi nel garantire un risparmio di tempo e denaro. Durante il corso dello sviluppo, è necessario cercare operazioni efficienti in termini di costi. Tuttavia, si deve notare che si deve pagare commissioni di transazione per ogni smart contract che si desidera valutare sulla Blockchain di Ethereum.

Nella maggior parte dei casi, i prezzi del gas sono imprevedibili e potrebbero portare a perdite in caso di errori in uno smart contract. In questi casi, Ganache può aiutare a testare i contratti intelligenti su una Blockchain locale. Di conseguenza, i vantaggi con Ganache sono chiaramente evidenti nella flessibilità per caricare più velocemente lo smart contract sulla rete Blockchain locale. Nel caso della Blockchain principale di Ethereum e delle reti di test, gli sviluppatori devono attendere un po' di tempo per caricare i loro contratti per i test.

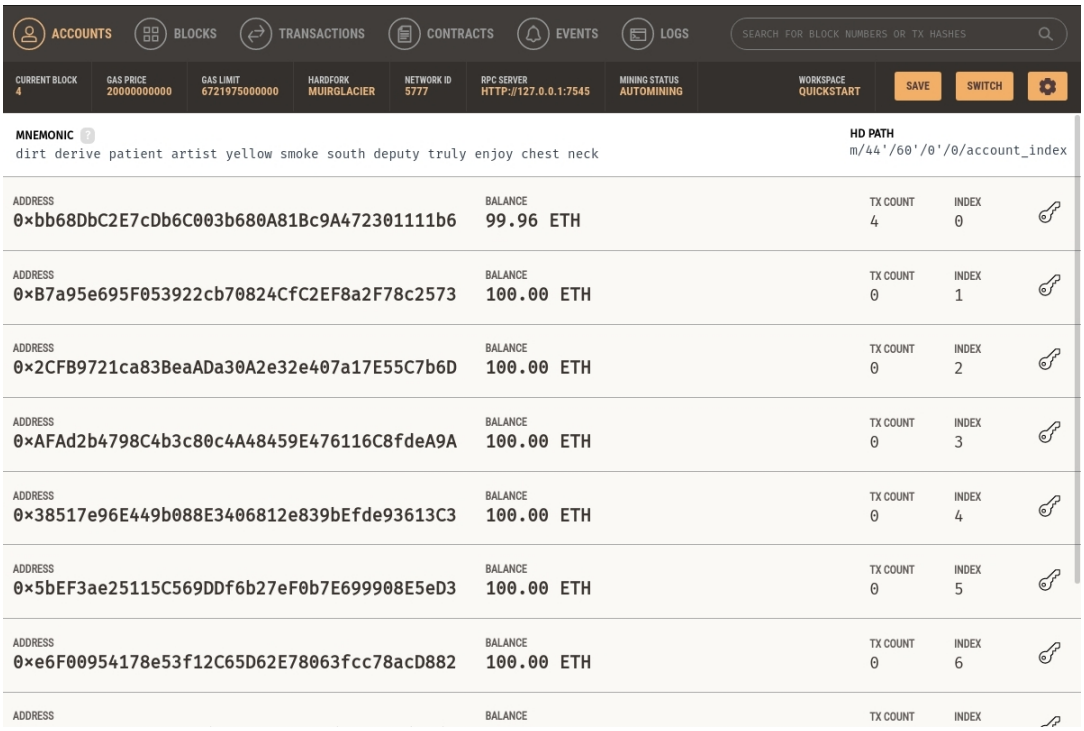


Figura 3.11: Ganache

## 3.9 Smart Contracts

Un contratto è un accordo legalmente vincolante che riconosce e disciplina i diritti ed i doveri delle parte del contratto. Uno smart contract è la trasposizione in codice di un contratto, mediante funzioni *if/then* incorporate in software o protocolli informatici, per verificare in automatico l'avverarsi di determinate condizioni e di auto eseguire azioni quando le condizioni sono raggiunte e verificate.

È un programma deterministico che elabora le informazioni in una Blockchain: a parità di input i risultati restituiti saranno identici. Ciò garantisce alle parti una assoluta certezza di giudizio oggettivo escludendo qualsiasi forma di interpretazione.

Uno smart contract per Blockchain deve soddisfare i seguenti obiettivi: osservabilità, verificabilità, riservatezza e applicabilità. Su una Blockchain, uno smart contract non può essere modificato, ma può essere facilmente osservato, verificato, auto applicato e, a seconda della modalità di accesso della Blockchain, è possibile ottenere la privacy.
Nella Figura 3.12 possiamo notare lo schema di uno Smart Contract che opera su una rete Blockchain.

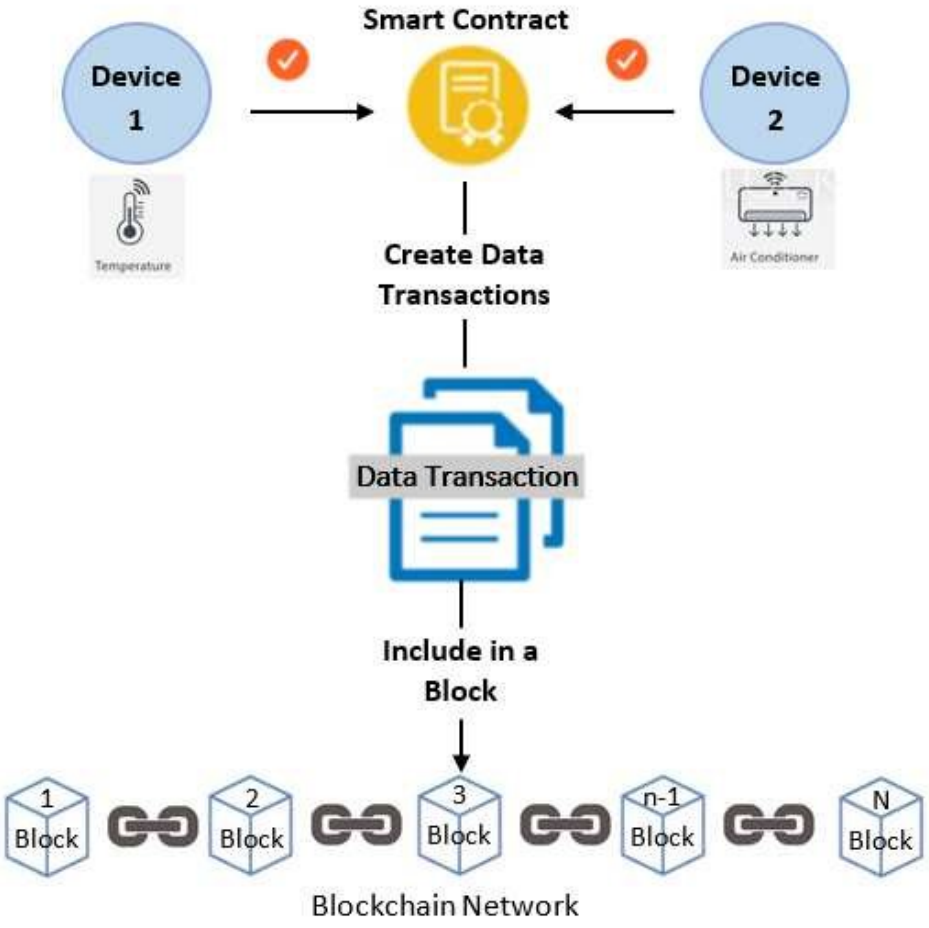


Figura 3.12: Smart Contract

### 3.9.1 Processo di vita di uno Smart Contract

Un processo graduale di vita di uno smart contract è il seguente:

1. Gli sviluppatori scrivono la logica, lo compilano ed ottengono una sua rappresentazione in bytecode;

2. Viene pubblicato sulla Blockchain ed archiviato. Una volta pubblicato, sarà di sola lettura o scrittura. Nel caso in cui sia di sola lettura, per fornire un aggiornamento gli sviluppatori dovranno pubblicare una nuova versione dello smart contract e reindirizzare gli utenti ad essa. Una volta caricato, lo smart contract è al suo stato iniziale, pari ai valori iniziali delle variabili interne.

3. L'accesso ad un programma di smart contract dipende dalla Blockchain: come un indirizzo restituito quando lo smart contract è caricato nella piattaforma. Tale indirizzo può essere utilizzato per interagire con lo smart contract, inviando le transazioni contenenti la funzione che si desidera utilizzare e gli argomenti della funzione stessa. Se è necessaria una quantità di valuta della piattaforma per avviare l'esecuzione della funzione, tale importo sarà trasferito insieme alla transazione. La transazione verrà archiviata nel pool di transazioni della piattaforma Blockchain che attendono di essere eseguite e convalidate.

4. La piattaforma Blockchain selezionerà le transazioni da eseguire e convalidare. Durante l'esecuzione, le funzioni nella transazione verranno eseguite dai nodi. Durante la validazione, i nodi confronteranno i propri risultati e selezioneranno quello da mantenere secondo un protocollo di consenso.

5. Una volta selezionato il risultato valido, verrà inserito in un blocco da aggiungere alla Blockchain. Se una transazione validata ha alterato le variabili interne di uno smart contract, i nuovi valori saranno considerati come valori iniziali per le transazioni future sullo smart contract. L'implementazione di smart contracts non è standardizzata e ogni piattaforma Blockchain propone una propria soluzione.

### 3.9.2 Solidity

Ai fini del progetto, lo smart contract realizzato è stato scritto in solidity: un linguaggio di alto livello orientato agli oggetti per l'implementazione di smart contracts. Solidity è un linguaggio influenzato da C++, Python e JavaScript. E' tipizzato staticamente, supporta l'ereditarietà, le librerie e i complessi tipi definiti dall'utente. Con Solidity si possono creare contratti per usi comuni come votazioni, crowdfunding, aste e portafogli multi-firma.

## 3.10 Scelta Progettuale

Il progetto implementato ha l'obiettivo di fornire un sistema di autenticazione attraverso l'utilizzo della Blockchain Ethereum. Nella fase iniziale del progetto è stato scritto uno smart contract in Solidity reso sicuro e robusto tramite la libreria *OpenZeppelin* per la definizione dei ruoli assunta dagli utenti. Successivamente, lo smart contract è stato tradotto in una classe Java, tramite la libreria Web3j, in modo da poter essere richiamato dal framework Spring Boot.

Infine, è stata configurata la rete di test Ganache per effettuare l'interfacciamento a MetaMask. Ciò ha permesso alla DApp di poter comunicare con la Blockchain e di effettuare le varie transazioni richiamando le funzioni scritte nello smart contract.

Inoltre, è stata adottata la tipologia username/passaword in quanto facile da utilizzare, offre garanzie in ottica di testing della funzionalità e la possibilità di automatizzare il processo perché non è presenta una richiesta manuale che, invece, avviene nella tipologia JSON token dove ad ogni autenticazione deve essere eseguita la richiesta di un nuovo token.

### 3.10.1 Sistema proposto

Per garantire una sicurezza maggiore nel processo di autenticazione, KryptoAuth è realizzata come una Web DApp sulla Blockchain Ethereum. Il primo passo sarà quello di collegarsi all'estensione MetaMask inserendo una password per caricare i vari account presenti sulla Blockchain Ganache (come mostrato in Figura 3.13).

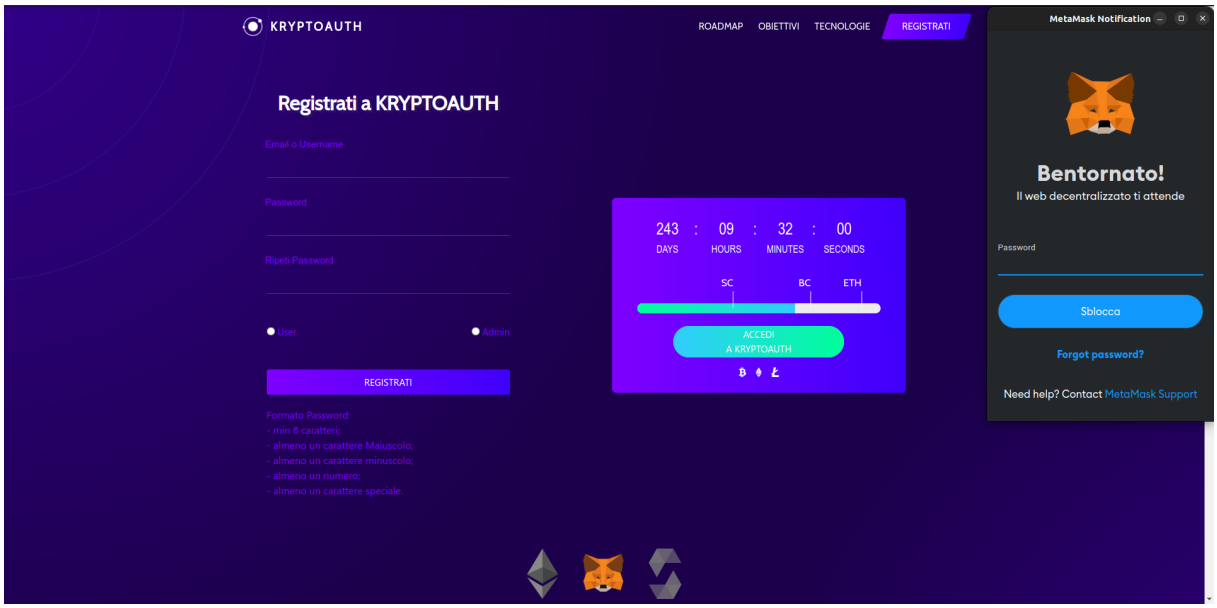


Figura 3.13: Login in MetaMask

L'utente si registrerà inserendo all'interno di un form i suoi dati, quali: username o e-mail, password, un secondo campo di inserimento password (per controllare che l'utente non abbia commesso errori nella digitazione) e due pulsanti per scegliere il ruolo che vorrà avere l'utente all'interno del sistema (Figura 3.14).

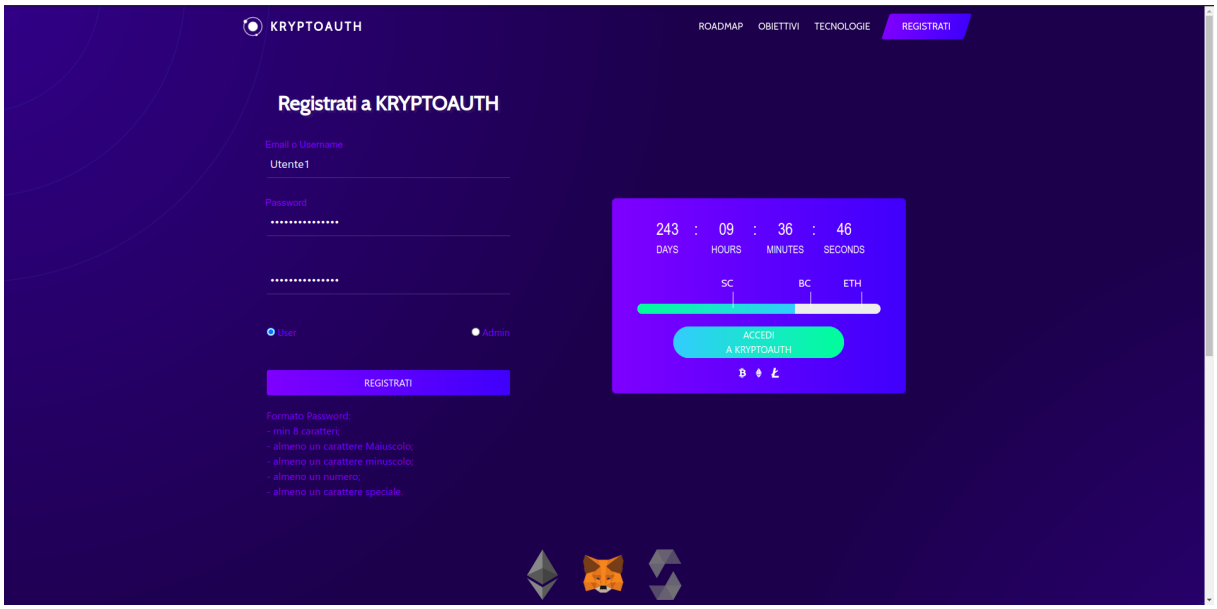


Figura 3.14: Registrazione in KryptoAuth

La registrazione sarà confermata soltanto dopo aver inserito la private key associata al seguente account (Figura 3.15 e Figura 3.16): la private key è una chiave personale e segreta conservata in maniera sicura su Metamask e dovrà essere inserita solo una volta, quando l'utente effettua la prima transazione e di conseguenza carica il contratto sulla Blockchain.

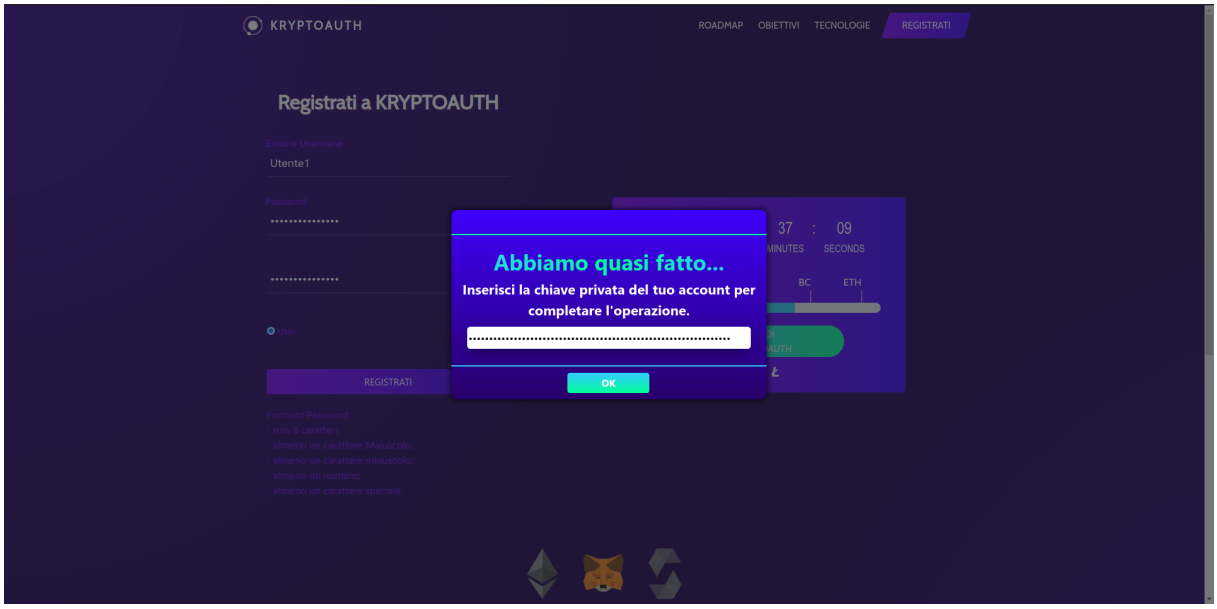


Figura 3.15: Campo Private Key

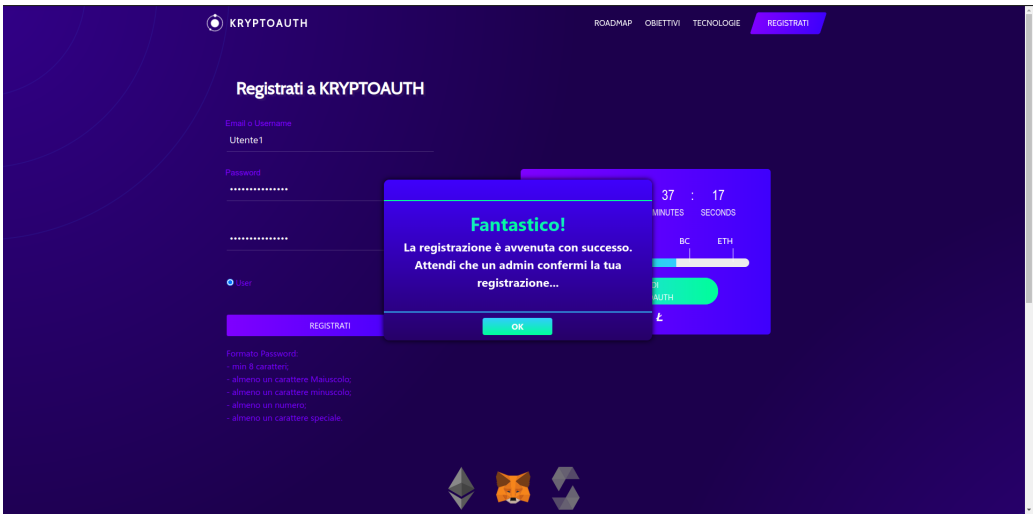


Figura 3.16: Popup di successo

Una volta effettuata la registrazione, l'utente potrà eseguire l'operazione di Login soltanto dopo l'approvazione dell'amministratore.

Gli amministratori possono attivare, cioè attribuire il ruolo di "Admin" o di "User", a qualsiasi account (ancora non attivo) di qualsiasi utente registrato al sistema. Inoltre, per gli account registrati e già attivi con il ruolo di "User", qualsiasi amministratore può promuoverli al ruolo di "Admin", mentre, solo l'amministratore proprietario, cioè colui che ha effettuato il login al sito in quel momento, può rinunciare al suo diritto di essere "Admin" e quindi diventare "User", oppure potrà disattivare il suo account e di conseguenza perdere qualsiasi ruolo. Nella Figura 3.17 l'amministratore attiva l'account dell'utente "*Utente1*" tramite il pulsante "Attiva" in corrispondenza del suo account. Dopodiché inserisce la sua private key (come mostrato in Figura 3.18) eseguendo la transazione sulla Blockchain Ganache. Se l'operazione ha avuto esito positivo verrà mostrato un popup con un messaggio di successo (Figura 3.19).

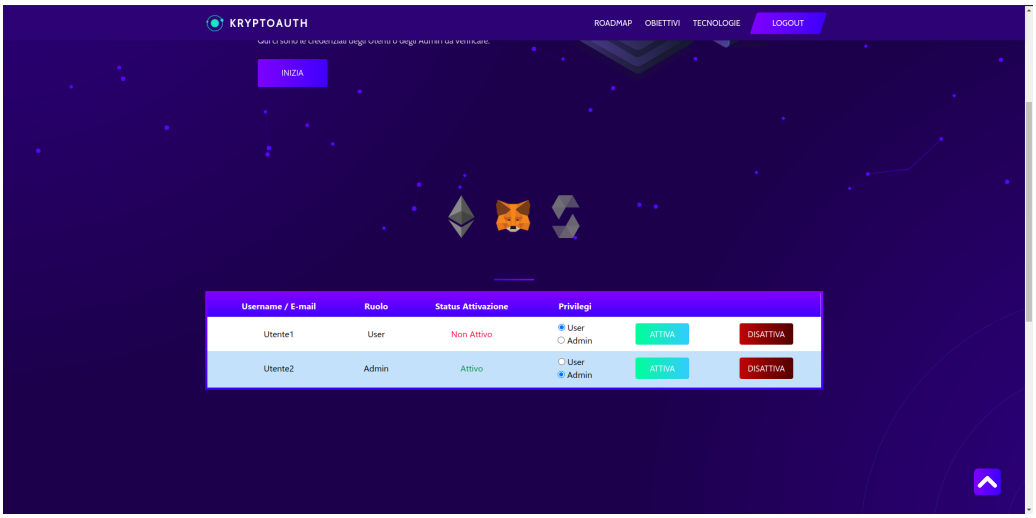


Figura 3.17: Area di amministrazione

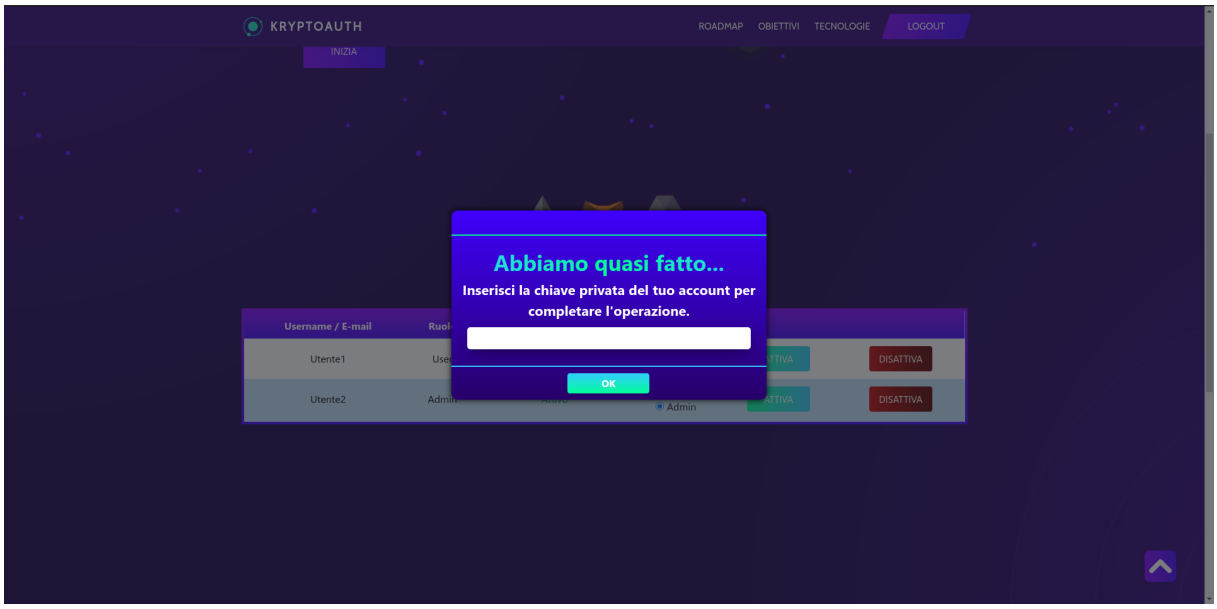


Figura 3.18: Campo Private Key Admin

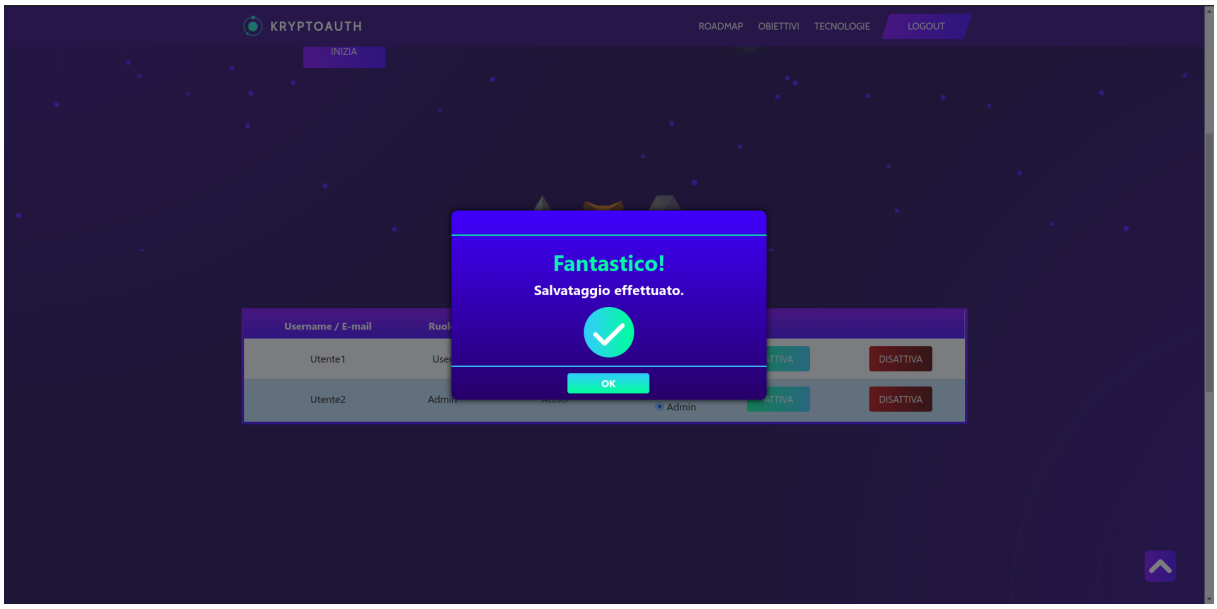


Figura 3.19: Popup di successo

La Figura 3.20 mostra l'operazione di revoca dei privilegi da parte dell'admin stesso, infatti, appare un popup che gli notifica se è sicuro della scelta che vuole effettuare. Una volta revocati i privilegi, l'amministratore sarà reindirizzato all'homepage (Figura 3.21) perdendo qualsiasi ruolo e l'account "Non Attivo".

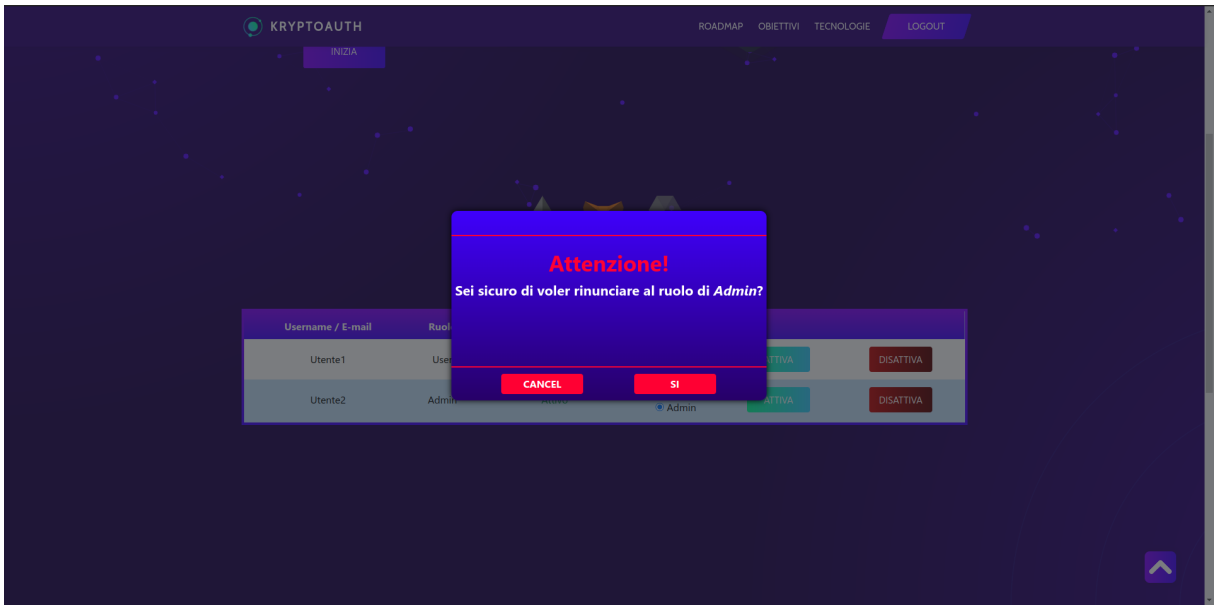


Figura 3.20: Revoca dei privilegi

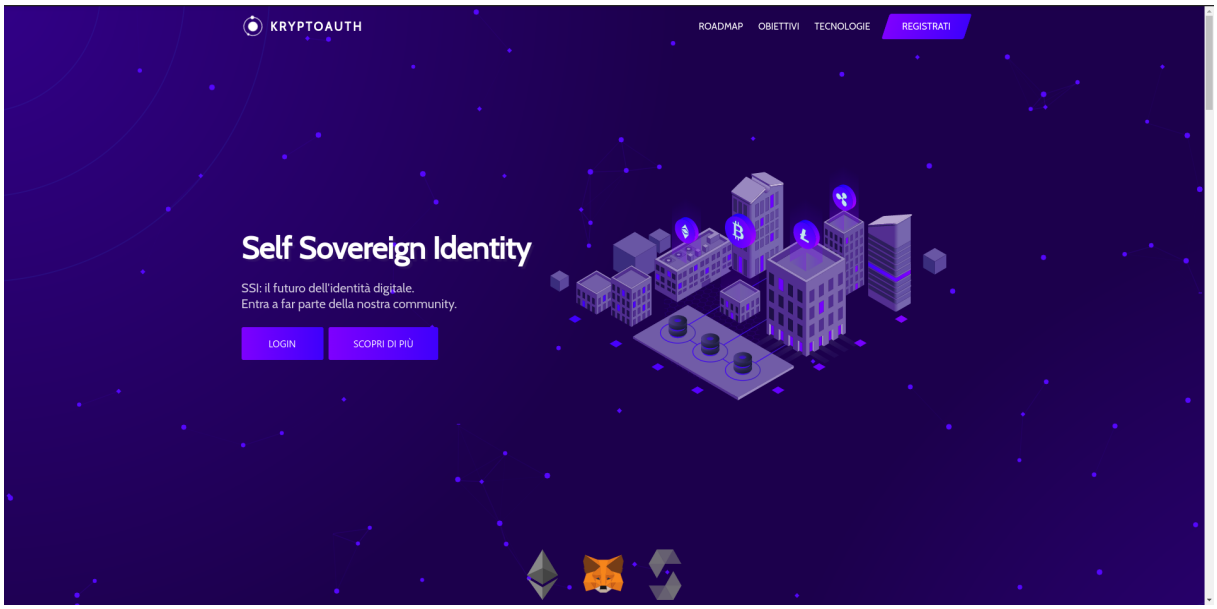


Figura 3.21: HomePage

### 3.10.2 Implementazione e installazione Smart Contract

Per implementare lo Smart Contract si è scelto il linguaggio Solidity incorporato della libreria OpenZeppelin per la definizione dei ruoli degli utenti: "Admin" o "User".

La Figura 3.22 mostra i vari ruoli che possono assumere gli utenti che accedono alla Web DApp; in particolare abbiamo una struct "User" in cui sono memorizzate le informazioni di ogni utente e un costruttore che assegna il ruolo di "Admin" all'account che carica il contratto sulla Blockchain. Le operazioni di assegnazione o revoca dei ruoli effettuate solo dall'account che possiede come ruolo "`DEFAULT_ADMIN_ROLE`" all'interno dello smart contract. Di conseguenza, tutti gli altri utenti sono impossibilitati nel modificare o nell'accedere a quei servizi che potranno essere eseguiti dall'amministratore.

Gli account, invece, che hanno come ruolo "`USER_ROLE`" potranno effettuare solamente il login al sito oltre che a navigare tra le varie pagine ma non potranno mai accedere alla sezione di gestione account riservata agli amministratori.

```
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/access/AccessControl.sol";

contract Authentication is AccessControl {
    bytes32 public constant USER_ROLE = keccak256("USER");

    struct User {
        address addr;
        string name;
        bytes32 password;
    }

    mapping(address => User) user;

    constructor() {
        _setupRole(DEFAULT_ADMIN_ROLE, msg.sender);
        _setRoleAdmin(USER_ROLE, DEFAULT_ADMIN_ROLE);
    }

    /* Ristretto ai membri che hanno l'admin role. */
    modifier onlyAdmin(){
        require(isAdmin(msg.sender), "Restricted to admins.");
        _;
    }

    /* Ristretto ai membri che hanno l'user role. */
    modifier onlyUser(){
        require(isUser(msg.sender), "Restricted to users.");
        _;
    }
```

Figura 3.22: Definizione dei ruoli con OpenZeppelin

La Figura 3.23 mostra la funzione di registrazione di un nuovo utente, la quale memorizza, all'interno della struct "User", le seguenti informazioni: l'account esadecimale che vuole effettuare la transazione, l'indirizzo e-mail o username e la password che sarà cifrata. La cifratura avviene tramite la funzione *keccak256* ed è previsto un controllo per verificare se l'attuale utente è già registrato all'interno della Web DApp KryptoAuth.

```
function registerUser(
    address _address,
    string memory _name,
    string memory _password
) public returns (bool) {
    require(user[_address].addr != _address, "already registered");

    user[_address].addr = _address;
    user[_address].name = _name;
    user[_address].password = keccak256(abi.encodePacked(_password));
    return true;
}
```

Figura 3.23: Funzione di registrazione per un nuovo utente

La Figura 3.24 mostra la funzione di login che verifica se l'username e la password inviati coincidono con quelli memorizzati nella Blockchain restituendo un booleano.

```
function loginUser(address _address, string memory _name, string memory _password) public view returns (bool) {
    //keccak256(abi.encodePacked(...)) è usato per comparare le stringhe e richiede meno gas
    if (isUser(_address) && user[_address].password == keccak256(abi.encodePacked(_password)) &&
        keccak256(abi.encodePacked(user[_address].name)) == keccak256(abi.encodePacked(_name))) {
        return true;
    }
    return false;
}
```

Figura 3.24: Funzione di login per un utente con il ruolo "User"

Infine, i ruoli vengono assegnati e revocati solo da un account amministratore come illustrato nella Figura 3.25.

```
/* Aggiunge un account con il ruolo di user (lo possono fare solo gli admin). */
function addUser(address account) public virtual onlyAdmin returns (bool){
    if(!hasRole(USER_ROLE, account)){
        grantRole(USER_ROLE, account);
        return true;
    } return false;
}

/* Aumenta i privilegi ad un account assegnandogli il ruolo di admin e revocando
 * quello di user (lo possono fare solo gli admin). */
function addAdmin(address account) public virtual onlyAdmin returns (bool){
    if(!hasRole(DEFAULT_ADMIN_ROLE, account)){
        removeUser(account);
        grantRole(DEFAULT_ADMIN_ROLE, account);
        return true;
    } return false;
}

/* Rimuove un account dal ruolo di user (lo possono fare solo gli admin). */
function removeUser(address account) public virtual onlyAdmin returns (bool){
    if(hasRole(USER_ROLE, account)){
        revokeRole(USER_ROLE, account);
        return true;
    } return false;
}

/* Un admin rinuncia ad esserlo. */
function renounceAdmin(address account) public virtual onlyAdmin returns (bool){
    if(hasRole(DEFAULT_ADMIN_ROLE, account)){
        renounceRole(DEFAULT_ADMIN_ROLE, account);
        return true;
    } return false;
}
```

Figura 3.25: Ruoli assegnati da un account amministratore

Per effettuare il deploy dello Smart Contract bisogna dapprima aprire l'applicazione Ganache, dopodiché ci rechiamo nel package "smart contract" del progetto KryptoAuth e all'interno del terminale digitiamo:

- `truffle compile`, per verificare la presenza di errori sintattici all'interno dello Smart Contract;
- `truffle migrate`, per deployare lo Smart Contract sulla Blockchain Ganache (per effettuare un reset delle connessioni eseguiamo `truffle migrate --reset`).

### 3.10.3 Traduzione Smart Contract in Java

Nel caso in cui si volesse tradurre lo Smart Contract, scritto in Solidity, in una classe Java, bisogna utilizzare:

- il compilatore *solcjs*, per la generazione dei file `Authentication.abi` e `Authentication.bin` (come mostrato nella Figura 3.26);

```
alberto@alberto-ASUS:~$ solcjs /home/alberto/Documents/Github/Blockchain-Authentication/KryptoAuth
/src/main/'smart contract'/contracts/Authentication.sol --bin --include-path node_modules/ --base-
path . --abi --optimize -o /home/alberto/Documents/Github/Blockchain-Authentication/KryptoAuth/src
/main/resources/solidity
```

Figura 3.26: Generazione .abi e .bin

- la libreria *Web3j*, per la creazione della classe `Authentication.java` (come mostrato nella Figura 3.27).

```
alberto@alberto-ASUS:~$ web3j generate solidity -b ./src/main/resources/solidity/Authentication.bi
n -a ./src/main/resources/solidity/Authentication.abi -o ./src/main/java -p it.unisa.KryptoAuth.co
ntracts
```

Figura 3.27: Generazione classe Java

### 3.10.4 Configurazione rete MetaMask - Ganache

Per collegare l'estensione browser Metamask alla Blockchain Ganache i passaggi iniziali da seguire sono di installare e di registrarsi a Metamask; successivamente dalle impostazioni dell'estensione andiamo nella sezione "Aggiungi Rete" per creare una nuova rete.

Le informazioni che devono essere inserite devono essere reperite da Ganache, in particolare:

- Nome rete: assegniamo il nome che vorremmo che abbia la nuova rete (esempio: *Ganache*);
- Nuovo URL RPC: è l'indirizzo http di Ganache (HTTP://127.0.0.1:7545);
- Chain ID: si deve inserire 1337 che corrisponde all'id di Ethereum;
- Currency Symbol: si deve inserire "ETH" perchè abbiamo una Blockchain Ethereum.

# Capitolo 4

# Conclusione

Nell'ambito della tesi si è descritta l'implementazione di un sistema di autenticazione tramite la Blockchain Ethereum e, in particolare, la necessità di scindere i dati dalle applicazioni al fine di fornire un effettivo controllo dei dati. Successivamente, ci si è concentrati sui meccanismi di autenticazione odierni e sulla necessità di introdurre una terza entità fidata, descrivendo la Blockchain.

Nella sezione relativa alla Blockchain sono state evidenziate le caratteristiche che questa tecnologia presenta tra cui l'immutabilità dei dati, che sono registrati all'interno dei blocchi, e la presenza di diverse tipologie di Blockchain che possono essere applicati in contesti differenti. A seguito dell'analisi sull'ambito di esecuzione è stato adottato l'utilizzo di Ganache, una piattaforma Blockchain per lo sviluppo rapido di applicazioni distribuite su Ethereum e Corda. Ganache può aiutare a testare i contratti intelligenti su una Blockchain locale. Di conseguenza, i vantaggi con Ganache sono chiaramente evidenti nella flessibilità per caricare più velocemente lo smart contract sulla rete Blockchain locale.

Quindi, è stata sviluppata una Web DApp in Spring Boot per fornire un'interfaccia al sistema, dove è possibile effettuare transazioni tramite la connessione alla Blockchain Ethereum.

## 4.1 Sviluppi futuri

La Web DApp realizzata è pensata come un forum in cui gli utenti possono chiarire i loro dubbi sulla tecnologia Blockchain, registrandosi al sito. Di conseguenza, viene offerto un luogo virtuale dove chiunque può informasi e pensare a come ampliare o migliorare il progetto. Ad esempio, creando degli NFT e implementandoli, si potrà dare la possibilità agli utenti di usufruire di servizi offerti dalla Web DApp, come sconti e cashback sull'acquisto di prodotti che il sito si predispone a vendere.

Infine, si ricordi che in informatica, tutto è divenire, tutto è migliorabile; è continuo cambiamento, evoluzione. Studiare la storia, gli studi già portati a termine, le app già programmate, è utile per dare una base di formazione in questo ambito mutevole e scorrevole. E' proprio da questo che nasce la passione per un ambito, nel saper guardare con occhi da bambino una sequenza di bit, ed è proprio questo che permette a tutti di andare avanti, di non fermarsi mai e di continuare a immaginare ciò che ancora non c'è.

Del resto, una macchina senza un uomo, è solo una macchina; ma un uomo senza una macchina, è un uomo che inventerà una macchina.

# Ringraziamenti

I primi ringraziamenti vanno a questi tre anni di Università in cui ho avuto modo di crescere, maturare e, soprattutto, di coltivare e ampliare le mie passioni. Sono stati anni impegnativi che, oltre allo studio intensivo e faticoso, mi hanno permesso di iniziare ad approcciarmi ad una nuova realtà, anche se piccola e identificata nel Campus di Fisciano; ho dovuto conquistare 180 CFU per arrivare fin qui e questo lo devo anche a quelle persone fantastiche che ho conosciuto, persone che mi hanno aiutato durante questo lungo viaggio.

Per quanto detto, inizio ringraziando il mio relatore Christiancarmine Esposito, presente, puntuale e disponibile (anche durante le vacanze estive). Grazie al percorso intrapreso insieme, ho ampliato la mia capacità di analisi e di problem solving e mi sono avvicinato maggiormente alla tecnologia Blockchain.

Grazie ai miei amici Josef, Gennaro e Martina per essermi sempre stati vicino, persino durante quest'ultima fase del mio percorso di studi. Grazie per aver ascoltato i miei sfoghi, per tutti i momenti di spensieratezza e per tutti gli esami e i progetti che abbiamo svolto assieme, aiutandoci e sopportandoci (soprattutto) continuamente.

Infine, ringrazio la mia collega, nonché, migliore amica Viviana per tutti i consigli e gli aiuti che mi ha dato in questi tre lunghi anni. Mi hai sopportato e supportato lungo tutto questo percorso, dandomi molto spesso l'epiteto di "scemo" e cercando di mostrarmi la soluzione migliore a tutti quei problemi sciocchi o meno che mi si presentavano davanti.

Oltre all'ambito universitario, non posso non citare quello familiare, importante soprattutto per mantenere la tranquillità e uno stato mentale solido. Devo dire che molto spesso questo aspetto è mancato, tuttavia devo ringraziare i miei genitori che da sempre mi sostengono nella realizzazione dei miei progetti, sia emotivamente che finanziariamente, con tutti i sacrifici che questo comporta. Grazie per avermi insegnato cos'è giusto e cosa non lo è. So che quello che fate, è tutto per proteggermi e permettermi di essere quello che sono oggi, anche se non mi dispiacerebbe, ogni tanto, essere un po' più libero.

Inoltre, voglio ringraziare la mia ragazza Mena per avermi trasmesso parte della sua "immensa" sapienza. Ti ringrazio per essermi stata vicino in questo percorso universitario e già so che farai altrettanto nei prossimi anni in cui intraprenderò gli studi magistrali. Grazie per avermi ascoltato ed essere stata cavia dei miei vari progetti informatici, mi hai dato idee, suggerimenti e tutti gli aiuti possibili quando mi vedevi in difficoltà. Ricordo ai tempi del liceo in cui mi insegnavi a costruire i primi algoritmi e mi introducevi alla logica (cosa che ancora oggi non ho acquisito del tutto, ma ci stiamo lavorando). Grazie anche alla tua famiglia che mi ha accolto come un terzo figlio e mi ha per lo più sfamato per tutte quelle volte che venivo a casa tua (per tutte le pringles e noccioline che ti ho consumato).

Infine, un ringraziamento particolare va a me stesso per la caparbietà e tenacia che metto in tutte le cose che faccio. Forse sarò un po' egocentrico ma, in realtà, sono la mia ambizione e sete di conoscenza che so che mi permetteranno di realizzare tutti i miei sogni o, per lo meno, di provarci, mettendo tutto me stesso e non rimpiangendo nessun giorno.

# Bibliografia


[1] LeewayHertz, *Web3 Vs Web 3.0: How Are They Different?*

[2] Articolo di Antonella Dorati e Stefania Costantini. *Approcci al Web Semantico*

[3] Marek Obitko, *Semantic Web Architecture, 2007*

[4] Intesa a Kyndryl Company, *Self Sovereign Identity: il futuro dell'identità digitale, 2020*

[5] Jake Frankenfield, *Decentralized Applications (dApps), March 19, 2022*

[6] The Cryptonomist, *Dapp: cosa sono e come funzionano le applicazioni decentralizzate*

[7] Bit2Me Academy, *Cosa sono le DApp?*

[8] Valeria Portale, *Cosa sono le DApp e come possono rivoluzionare il mondo delle App, 31 Ottobre 2018*

[9] Roberto Cocciolo, Puntoinformatico. *Autenticazione a due fattori o verifica in due passaggi: qual è la differenza?, 25 Febbraio 2022*

[10] Giorgio Sbaraglia, Cybersecurity360. *Autenticazione a due fattori: cos'è, come e perché usarla, anche alla luce della PSD2, 4 Aprile 2022*

[11] Giorgio Sbaraglia, Cybersecurity360. *Autenticazione a due fattori: cos'è, come e perché usarla, anche alla luce della PSD2, 4 Aprile 2022*

[12] Wikipedia, *OAuth, 4 Agosto 2022*

[13] Nikos Fotiou, Iakovos Pittaras, Vasilios A. Siris, Spyros Voulgaris, George C. Polyzos, *OAuth 2.0 authorization using blockchain-based tokens*

[14] IBM, *Cos'è la tecnologia blockchain?*

[15] Mauro Bellini, Blockchain4Innovation. *Blockchain: cos'è, come funziona e gli ambiti applicativi in Italia, 3 Febbraio 2022*

[16] Valeria Vitale, Spindox S.p.A. *La classificazione delle Blockchain: pubbliche, autorizzate e private, 20 Giugno 2018*

[17] Ethereum, *Cos'è Ethereum?, 18 Agosto 2022*

[18] Consensys, *Blockchain Fundamentals: FAQ*

[19] Wikipedia, *Ethereum, 16 Agosto 2022*

[20] Plus500, *Qual è la differenza fra Ethereum e Bitcoin?*

[21] Truffle Suite, *Ganache, one click Blockchain*

# Elenco delle figure