Università degli Studi di Torino

Facoltà di Scieze della Natura
Corso di Laurea in Informatica

Tesi di laurea

Studio e realizzazione di un prototipo di un sistema basato su blockchain per il mobility as a service

Candidato:
Giorgio Mecca
Matricola 880847

Relatore:
Prof. Claudio Schifanella

Anno Accademico 2020-2021

Sommario

Il seguente elaborato descrive lo studio effettuato sulla tecnologia blockchain durante il tirocinio presso la 2+consulting. Lo studio è stato svolto ponendo particolare attenzione sulle blockchain private, la differenza e i vantaggi con le tecnologie pubbliche. In seguito ho dedicato la mia concentrazione allo studio di come e se fosse possibile avviare una blockchain privata scegliendo come tecnologia il client Besu, client facente parte del gruppo Hyperlegder che utilizza la tecnologia Etherium. Una volta definiti e studiati i parametri di essa, come ad esempio il meccanismo del consenso da utilizzare scegliendo IBFT 2.0 basato su PoA, ho studiato il modello di gestione e utilizzo di blockchain con gli Smart Contract, che permettono inoltre di inserire e memorizzare dati su blockchain includendo dati sulle transazioni pubbliche da certificare. In questo modo si è svolto uno studio sui dati utilizzati e come gestire il modello di Blockchain Ibrida, cioè l'utilizzo concatenato di una blockchain privata utilizzata per le sue proprietà sulla sicurezza e quindi come ente certificatore decentralizzato e un DataBase relazionale così da poter interrogare e accedere con facilità ai dati. Il passo finale è stato sviluppare una Web app con tipologia Single Page Application utilizzando come server la tecnologia Node Js con l'utilizzo del framework Web3 per l'interfacciamento alla blockchain. La Web app è sviluppata con l'ideologia della possibile integrazione nel Mobility as Service. Questa caratteristica è stata sviluppata con l'ideologia di fornire ai vari enti partecipanti un diverso account(coppia di chiavi pubblica e privata) che si utilizzerà per l'interfacciamento e gli fornirà determinati accessi. La WebApp ha quindi permesso la distinzione dei vari utenti che, in base al loro ruolo, avranno accessi e possibilità diverse sull'app. Queste possibilità riguardano chi e quali dati dovrà inserire e chi invece può interrogare le informazioni a cui ha accesso.

Indice

Capitolo 1

Introduzione

1.1 Descrizione del Progetto

1.2 Descrizione dell'azienda

Capitolo 2

Blockchain

2.1 Problema dei generali bizantini

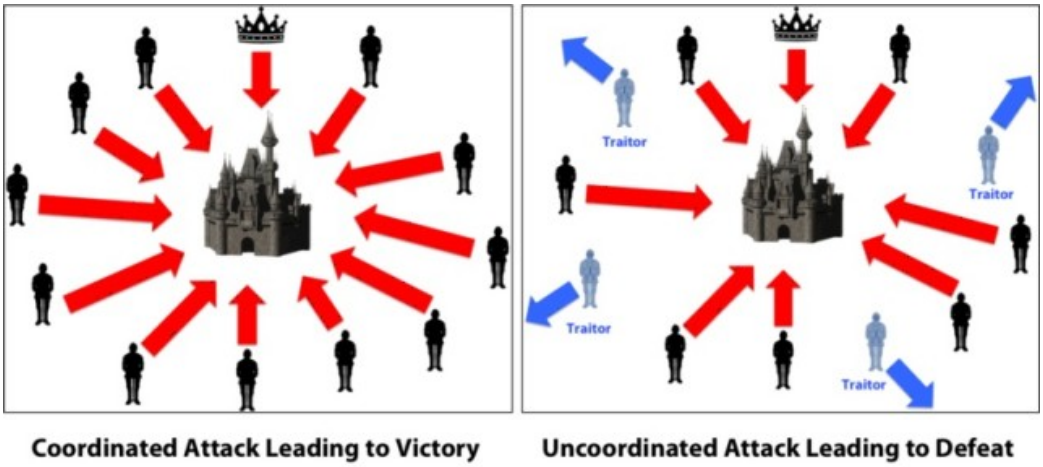

Figura 2.1: Problema dei generali bizantini

Il problema dei generali bizantini è un problema informatico su come raggiungere il consenso in situazioni in cui è possibile la presenza di errori. Il problema consiste nel trovare un accordo, comunicando solo tramite messaggi, tra componenti diversi nel caso in cui siano presenti informazioni

discordanti. Il problema è stato teorizzato dai matematici Leslie Lamport, Marshall Pease e Robert Shostak nel 1982, i quali crearono la metafora dei generali, caso di studio molto utilizzato nei sistemi basati o che comunque utilizzano una network. La metafora si basa su diversi generali che durante un assedio sono sul punto di attaccare una città nemica. Essi sono dislocati in diverse aree strategiche e possono comunicare solo mediante messaggeri al fine di coordinare l'attacco decisivo (Figura 2.1). I generali possono attaccare o ritirarsi, l'importante e che ci sia una decisione unanime, l'utilizzo di sola metà forza bellica porterebbe ad una sconfitta o una perdita. Il problema risiede quindi nell'alta probabilità che tra questi vi sia un generale traditore che mandi messaggi che vanno contro la strategia dell'esercito. La possibile soluzione punta al trovare un meccanismo secondo il quale un generale non traditore che riceva più messaggi sappia riconoscere quello veritiero. Secondo l'articolo di Lamport, Shostak e Pease non esiste una soluzione se il numero di processi non corretti è maggiore o uguale a un terzo del numero totale di processi. Una soluzione proposta è quella di Nakamoto che in una sua stesura sulla blockchain descrive un meccanisco per arrivare al consenso chiamato PoW Proof of Work.

2.2 Struttura di una blockchain

Una Blockchain come suggerisce l'etimologia della parola è una catena di blocchi o DLT - Distributed Ledger Technology. È una struttura dati formata da un insieme di blocchi(struttura prioritaria) collegati univocamente 1 ad 1 così da creare una metaforica catena. Una blockchain è considerata una struttura condivisa e immutabile in quanto il suo contenuto una volta scritto non è più né modificabile né eliminabile, a meno di non invalidare l'intera struttura. Questa tecnologia fà parte dei Distributed Legder cioè dei "libri mastro distribuiti" o registri condivisi infatti tutti i partecipanti della blockchain, detti anche nodi, posseggono lo stesso registro cioè le stesse

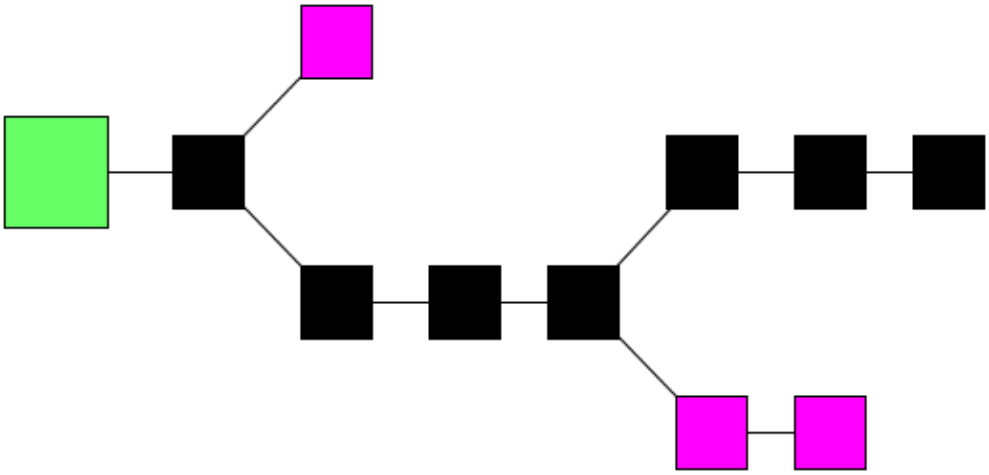

Figura 2.2: Rappresentazione struttura di una blockchain

informazioni andando a costruire il contrapposto di una struttura centralizzata come un Database, quindi una struttura Decentralizzata in cui ogni nodo ha la possibilità di leggere autonomamente le informazioni contenute. Nella figura 2.2 viene visualizzata una semplice blockchain in cui sono presenti tre tipologie di blocchi quali, il blocco verde visto come il blocco di genesi, i blocchi neri che vanno a costituire la catena principale e i blocchi viola considerati blocchi orfani. L'aggiunta di un nuovo blocco è globalmente regolata da un protocollo condiviso e se autorizzata ogni nodo aggiorna la propria copia privata del registro così da evitare manipolazioni future. In una blockchain i nodi partecipanti vengono anche chiamati minatori- miner o validatori, riferendosi al loro compito nella rete rispetto ai blocchi.

2.3 Hashing

Un codice hash è una qualunque sequenza di caratteri alfanumerici generati da una particolare funzione di hash. Questa funzione prende in input un qualunque tipo di informazione e restituisce una stringa di lunghezza prefis-

sata, questo rende la funzione one-way o non invertibile in quanto conoscendo il digest(codice hash restituito) non è possibile risalire all'informazione che lo ha generato. In una blockchain l'Hash viene utilizzato per la costruzione della catena, viene calcolato l'hash di un blocco e il blocco che lo succederà avrà come parametro questo hash. In questo modo ogni blocco è legato univocamente al blocco precedente e siccome il codice hash di un blocco viene calcolato utilizzando anche il codice hash precedente modificando un singolo blocco verrà invalidata tutta la struttura blockchain immediatamente successiva.

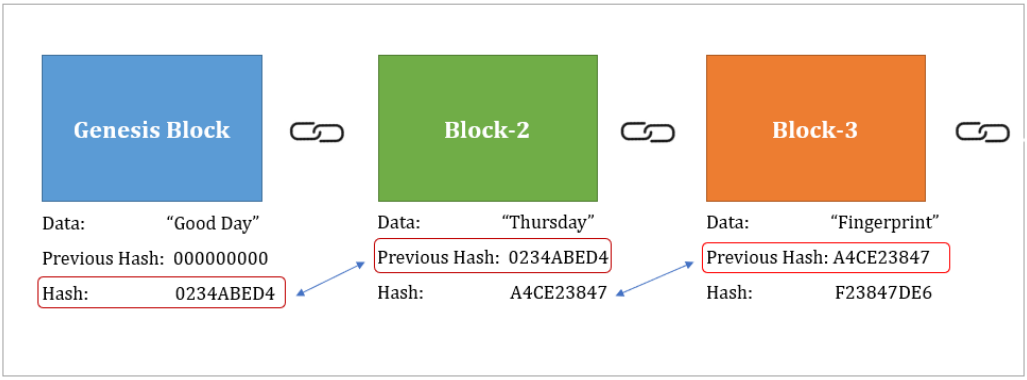

Figura 2.3: Catena di una blockchain

2.4 Transazioni

In una blockchain i dati vengono scritti sotto forma di Transazioni in seguito contenute in vari blocchi. L'uso più comune delle transazioni è l'invio di denaro o in qualche modo una moneta equivalente. Le transazioni devono quindi avere un mittente, un destinatario, un 'value' cioè il valore trasmesso, vengono quindi considerate come un cambio di stato riferendosi alle informazioni nella blockchain e saranno identificate da un Transaction Hash. Una volta inviata, la transazione entra in un transaction pool, da dove i miner an-

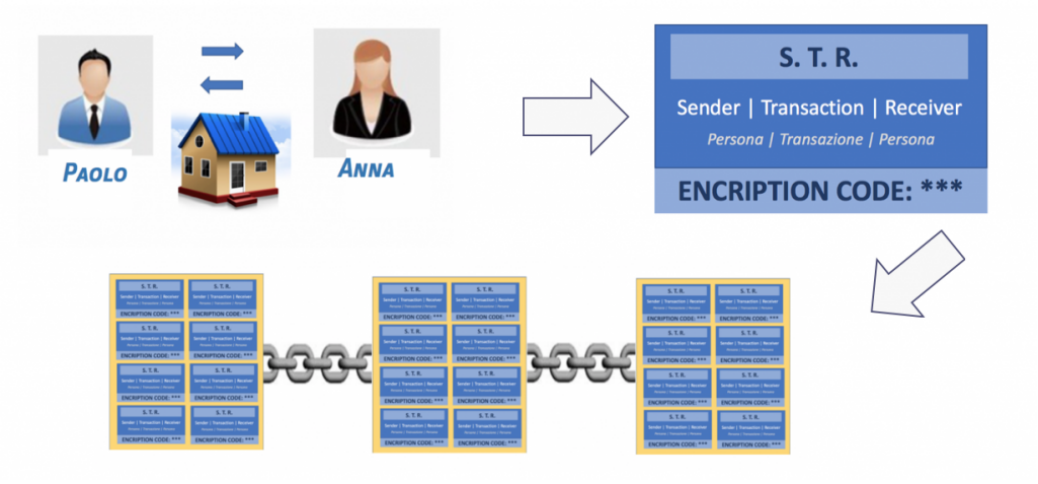

Figura 2.4: Flusso di esecuzione di una Transazione

dranno a selezionare randomicamente transazioni da includere nel prossimo blocco. Una transazione per essere considerata valida deve essere accettata da un nodo che la inserirà nel blocco che sta minando, non è certo che due nodi che minano lo stesso blocco la inseriscano nella stessa posizione.

2.5 Blocchi

La blockchain è una sequanza di blocchi che contengono una collezione di Transazioni. il numero di transazioni all'interno di ognuno di questi blocchi varia in base alla dimensione della transazione stessa.I blocchi sono prodotti dai nodi validators e vengono generati in un lasso di tempo definito dalle regole della blockchain(Es: 15 secondi per Ethereum, 10 min per BitCoin), nel momento in cui il blocco viene completato i dati contenuti diventano verificati.

Un Blolcco è diviso in due parti, l'header e il body. Le transazioni sono racchiuse nel body del blocco e nell'header sono presenti i campi di gestione del blocco stesso come descritto nella figura 2.5.

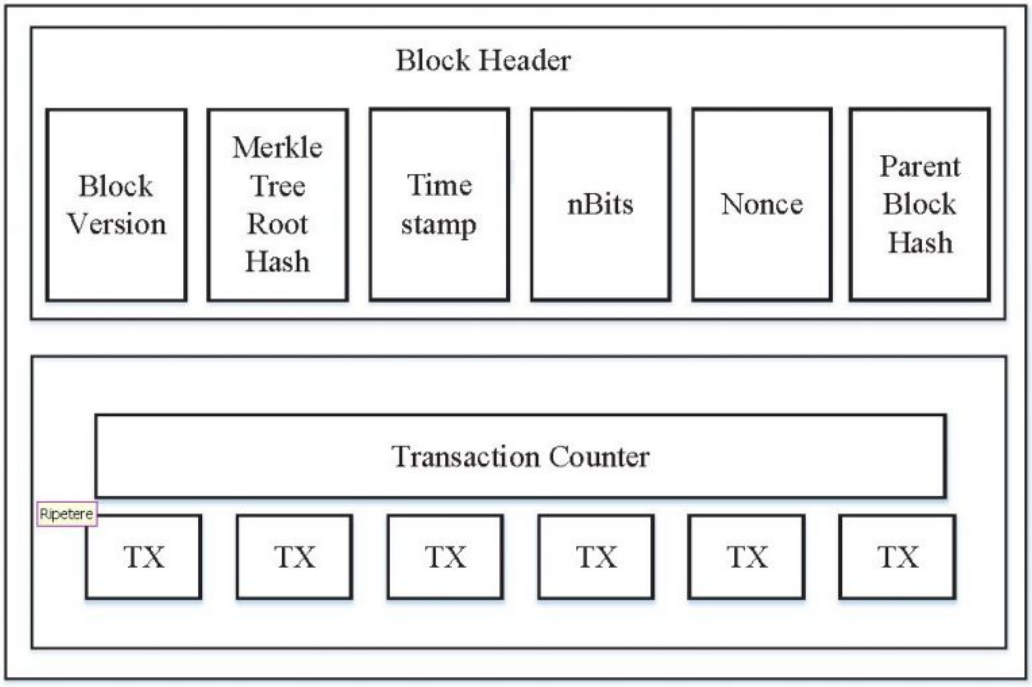

Figura 2.5: Struttura di un Blocco

- Versione del blocco: indica le regole di validazione del blocco da rispettare
- Merkle Tree Root Hash valore della radice del Merkle Tree in cui sono salvate le transazioni del blocco
- TimeStamp: Marca temporale salvata come Timestamp UNIX che indica l'inserimento del blocco nella blockchain
- nBits: è la soglia target di un hash di blocco valido
- Nonce: è un campo il cui valore è settato dai miner così che l'hash del blocco calcolato sia minore o uguale al target attuale della rete(difficoltà). Dato che non è possibile prevedere la combinazione di bit che risulterebbero nell'hash voluto, numerosi valori di nonce sono

calcolati fino a quando l'hash risultante rispetti i requisiti attuali della rete.

- Parent Block Hash: segna l'hash del blocco a cui verrà agganciato

Il Body è composto da un contatore di transazioni (transaction counter) e dalle transazioni (TX), queste vengono memorizzate e organizzate tramite un Merkle Tree cioè una struttura ad albero in cui le foglie contengono i digest hash delle informazioni mentre i nodi contengono i digest hash dei nodi sottostanti(figli)

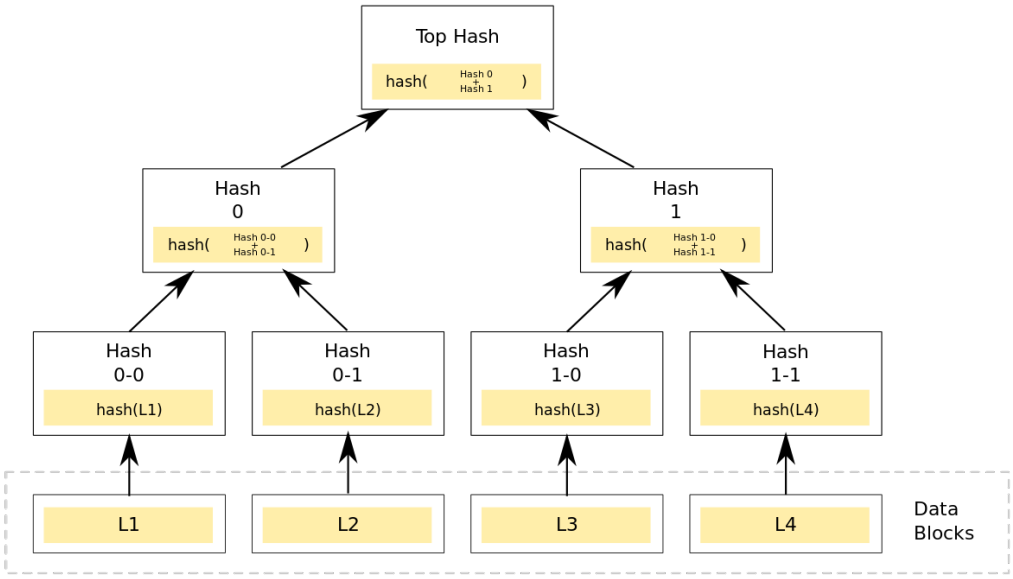

Figura 2.6: Merkle Tree

2.6 Mining e Meccanismi del consenso

I miner sono i nodi partecipanti alla rete che ascoltano le transazioni inviate, verificano che non siano malevole e compongono un blocco organizzando le

transazioni in un Merkle Tree. I minatori che 'calcolano' un blocco valido vengono premiati con un incentivo (Es: criptovaluta) È anche possibile che più miner producano lo stesso blocco e la scelta del blocco da seguire per la catena spetta al meccanismo del consenso cioè il set di regole che permette la finalizzazione delle transazioni e il funzionamento del sistema così che tutti i nodi della rete convergano ad una sola versione condivisa della catena, questo andrà a creare una biforcazione o Fork da cui si avranno una sequenza valida(quella che verrà continuata) e una sequenza(anche un singolo nodo) orfana(Figura 2.2).

2.6.1 Consenso Trustless

La blockhain è considerata Trustless, letteralmente "senza fiducia" poiché a differenza di un sistema centralizzando non esiste nessun ente centrale in cui riporre la fiducia(come ad esempio una Banca,governi o istituti finanziari). Utilizzando un sistema decentralizzato la fiducia non viene eliminata definitivamente ma viene suddivisa tra tutti i partecipanti, in questo modo più sarà alto il numero dei partecipanti meno fiducia si dovrà affidare ad ogni singolo nodo. La fiducia viene quindi riposta nei singoli nodi e che questi rispettino le regole considerate valide.

2.6.2 Proof of Work

L'algoritmo del consenso più famoso ed utilizzato(come da Bitcoin, Ethereum e Monero) nonché il primo algoritmo del consenso mai creato è il PROOF OF WORK abbreviato PoW. Questo si basa sulla "Prova del Lavoro" svolto dai miner che dovranno risolvere unauna serie di operazioni considerate u puzzle matematico per poter creare un blocco valido. Il primo miner che costruisce il blocco lo aggiunge alla catena, notifica in broadcast il resto della rete e di conseguenza tutte le transazioni in esso presenti vengono validate, infine il miner viene ricompensato con un incentivo come una moneta che sarà relati-

va quello che gli utenti pagano per effettuare le transazioni, cioè le "tasse" - fees. I miner quindi utilizzano la loro potenza e le risorse computazionali per provare che hanno svolto del lavoro(questo produce anche un massivo consumo di elettricità), talvolta è possibile che due o più nodi producano lo stesso blocco creando un fork e quindi due differenti catene. Il sistema è incentivato a scrivere nuovi blocchi sulla catena più lunga così da eliminare la biforcazione orfana e ricondurre tutta la rete a un'unica catena, questo implica che se si voglia creare un blocco "falso" e farlo accettare dalla blockchain bisognerà possedere una potenza di calcolo maggiore di tutta la rete. Il mining diventa sempre più competitivo con la partecipazione di sempre più gente e con l'aumentare della difficoltà, perciò si è vista la creazione delle Mining Pool cioè un insieme di persone che raggruppa la propria potenza di calcolo per il mining di criptomoneta, in questo modo la probabilità di costruire un blocco valido aumenta e di conseguenza aumenta anche il guadagno suddiviso tra i partecipanti

2.6.3 Proof of Stake

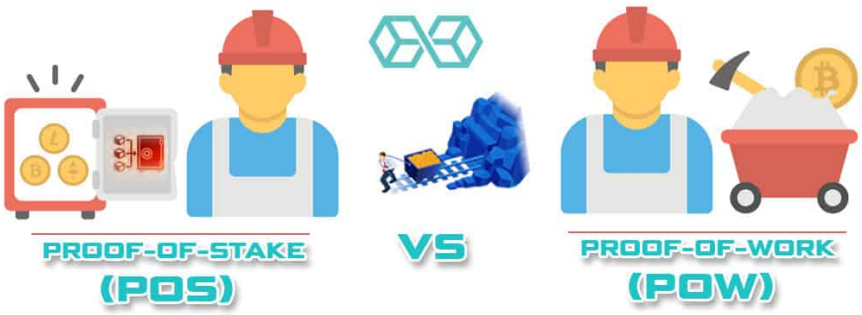

Figura 2.7: PoW-PoS

Nel 2011 basandoci sui problemi del PoW come il dispendio di energia elettrica o la creazione di grosse Mining Pool che elimina la decentralizzazione si è sviluppata l'idea del PoS Proof of Stake. Il PoS sostituisce i miner con i validatori o coniatore, questi per far si che il loro blocco venga considerato valido devono depositare della moneta come "cauzione" che viene appunto chiamata Stake. In questo modo se con il PoW era possibile inserire transazione fraudolente qui si andrebbe a perdere la somma congelata. Il criterio di scelta dei validatori si basa sulla quantità di moneta bloccata la durata del blocco, con il secondo parametro si va in contro al problema generato dal primo parametro cioè che solo i più ricchi possono essere scelti e quindi diventare più ricchi. L'elezione del blocco leader (da aggiungere) avviene tramite l'algoritmo VFR – Verifiable random function che utilizza l'algoritmo "follow-the-coin" - "più denaro blocchi più hai fiducia". Il PoS porta molteplici vantaggi rispetto al PoW come un minimo dispendio energetico e una maggiore sicurezza e decentralizzazione dovuta all'assenza di mining pool.

2.6.4 Proof of Authority

Un ultimo algoritmo del consenso è il PoA - Proof of Autority. Qui viene meno il concetto di decentralizzazione in quanto si basa sull'utilizzo di nodi validatori noti. Il PoA viene spesso utilizzato in ambito privato o militare e si utilizza, come intuibile dal nome, il concetto di Autorità che quindi avrà il potere di decidere i nodi validatori, cioè gli unici nodi che potranno produrre blocchi, mentre gli altri nodi avranno soltanto la possibilità di lettura. Il PoA fa si che i nodi validatori mettano in "Stake" la loro reputazione a differenza di una

moneta. Il modello Proof of Authority consente alle imprese di mantenere la propria privacy e allo stesso tempo avvalersi dei vantaggi della tecnologia blockchain. Inoltre il modello PoA riduce il problema del consumo in quanto diventa inutilizzabile il concetto di concorrenza nel mining e di conseguenza le mining pool. Riferendosi al Trilemma della scalabilità PoA rinuncia alla decentralizzazione a favore della sicurezza.

2.7 Attacchi

Nonostante la blockchain sta avendo molteplici riscontri positivi negli ultimi anni la sua caratteristica di decentralizzazione la rende si più affidabile rispetto ad un sistema centralizzato ma risultano comunque possibili e attuabili degli Attacchi ad essa.

2.7.1 Selfish Mining Attack

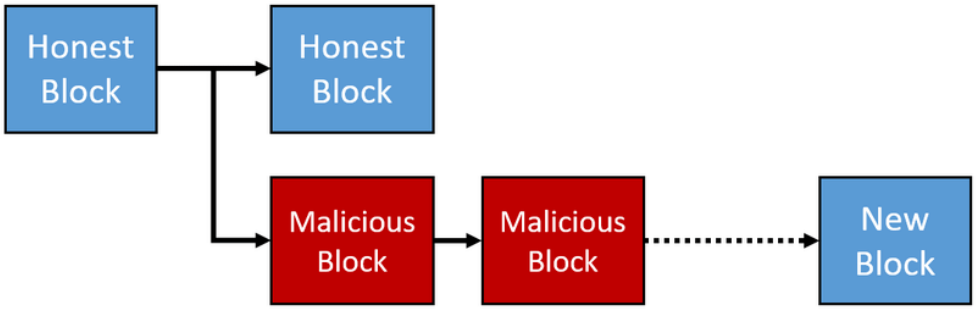

Figura 2.8: Selfish-mining-attack

Il Selfish Mining Attack si basa su un sistema PoW e sfrutta la sua debolezza nel momento in cui si genera un Fork. Nel momento in cui si genera una biforcazione solo la catena più lunga viene considerata valida e le transazioni nella catena orfana vengono rese nulle. Per l'attuarsi del Selfish Mining

Attack si ha bisogno che l'attaccante cioè il Faul-Miner generi un blocco che potrebbe creare un Fork (in cui si ha la possibilità di inserire transazioni fraudolente) ma lo tenga segreto, senza aggiungerlo alla Blockchain. Il Fault-Miner dovrà continuare a generare blocchi seguendo la sua catena e quando questa sarà più lunga di quella che attualmente gli altri miner stanno seguendo, pubblicherà il suo Fork e la sua catena che essendo più lunga secondo l'algoritmo del PoW verrà considerata valida e sarà quella che gli altri miner seguiranno da quel momento in poi. Per l'attuazione del Selfish Mining Attack si ha bisogno quindi che un singolo ente possegga più del 51

2.7.2 Double Spending Attack

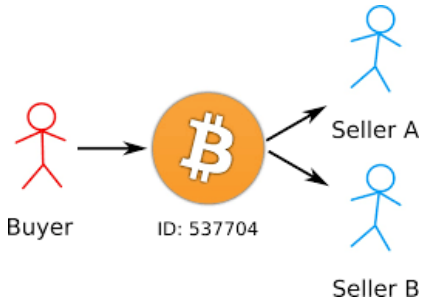

Figura 2.9: Double Spending Attack

Il Double Spending Attack viene concepito per la possibilita di spedere due volte la stessa moneta. Questo potrebbe essere attuabile con la creazione di due diverse transazioni. Le due transazioni però dovranno appartenere a due blocchi diversi, e se due miner che competono nel minare un blocco utilizzano ognuno una transazione diversa? Allora verranno prodotti due blocchi con entrambe le transazioni MA verrà prodotto un Fork quindi sono una delle due transazioni verrà poi validata mentre la seconda sarà annullata, ricordando che una transazione o meglio un blocco ha una validità pari al numero di blocchi minati in seguito (In BitCoin un blocco è validato se sono stati prodotti 6 blocchi successivi).

2.8 Blockchain Pubbliche/Private

Una blockchain pubblica solitamente è anche detta permissionless, letteralmente "senza permessi" cioè un nuovo nodo non ha bisogno di speciali permessi per partecipare quindi minare o effettuare transazioni nella rete che quindi viene definita pubblica. Le reti permissionless sono quindi decentralizzate in quanto nessuno ha il controllo della rete.

Le blockchain private sono conosciute come permissioned, sono caratterizzate dalla presenza di un'autorità centrale che decide chi può accedere e assegna loro un ruolo nella rete che determinerà cosa il nuovo nodo avrà il permesso di fare, se ha la possibilità di partecipare alla rete o se potrà essere un nodo validatore o sola lettura . La sicurezza di una blockchain privata fa maggiormente perno sull'affidabilità dei singoli partecipanti.

2.9 Ethereum

Figura 2.10: Etherium Logo

Ethereum è una piattaforma decentralizzata ideata nel 2013 (in seguito pubblicata nel 2015) come sostituto a BitCoin. Ethereum è una piattaforma basata su blockchain che permette quindi la gestione degli smart contract e come Bitcoin mette a disposizione una moneta che viene analogamente chiamata Ether e abbreviata con ETH. La moneta viene utilizzata per le varie transazioni ma viene anche usata per pagare le "tasse" - fee, nello specifico

viene utilizzato un sottomultiplo dell'ETH chiamato wei che corrisponde a circa 10^{-18} ETH (1 ETH = 10^{18} wei). Per lo sviluppo e interfacciamento viene messa a disposizione da Ethereum la EVM - Ethereum Virtual Machine che funge da macchina turing completa in grado di eseguire byte code. La sua funzione è quella di consentire l'esecuzione di programmi o smart contract al fine di implementare una serie di funzionalità aggiuntive su detta blockchain; la Evm utilizza per gli smart contract un linguaggio di alto livello specializzato chiamato Solidity.

2.9.1 Smart Contract

Uno smart Contract, o contratto intelligente è un programma messo a disposizione da ethereum ed è una collezione di codice(funzioni) e di dati(stato). Lo smart Contract è identico ad un contratto reale ma grazie alla tecnologia blockchain non necessita di un terzo ente verificatore. Sono programmi scritti in un linguaggio di programmazione ad alto livello come Solidity (simil-JavaScript) o Vyper (simil-Python), compilati in bytecode e distribuiti sulla blockchain tramite speciali transazioni inviate ad un generico indirizzo 0x0 pagando un determinato gas. Nonostante siano caricati da un utente gli smart contract non hanno un proprietario ma appartengono alla rete, anche la loro sicurezza deriva da essa, vengono infatti definiti come programmi "on-chain" cioè programmi caricati su rete che mantengono uno stato sicuro e inviolabile al di fuori delle regole del contratto stesso infatti una volta caricato questo non può essere modificato. Ogni operazione gestita da uno smart contract che si vuole effettuare comporta una transazione che avrà come destinatario l'indirizzo del contratto e che come una qualunque transazione sarà considerata valida sono una volta che la rete l'avrà validata.

2.9.2 Solidity

Solidity è il linguaggio più diffuso nel contesto Ethereum ed è un linguaggio orientato agli oggetti, compilato, staticamente tipato e di alto livello usato per implementare smart contract eseguibili sull'Ethereum Virtual Machine. È stato sviluppato per essere simile alla sintassi ECMAScript per renderlo familiare agli sviluppatori web. Solidity è un linguaggio fortemente tipato, i tipi delle variabili sono gestiti in modo statico, cioè il tipo va dichiarato nel momento in cui questa viene creata. Solidity gestisce i tipi di dato in tre suddivisioni:

- Tipi Valore: possono essere utilizzati per lo storage di comuni valori come un INT oppure UINT per un unsigned int, ma vengono considerati tali anche i bool o le string;
- Tipi indirizzo: come specificato dal nome indicano gli address di altre variabili/oggetti;
- Tipi mapping: rappresentano strutture dati di tipo chiave/valore. Durante la creazione tutti i valori vengono inizializzati con i byte a zero, il loro uso è analogo ad un "array" visto da altri linguaggi o meglio ad una Hash map.

Le funzioni rappresentano il codice eseguibile di uno smart contract, possono prendere vai parametri in input e possono restituire più argomenti come output che andranno specificati nella firma. Queste posseggono una visibilità che può essere public, private, external, internal. Un contratto di esempio :

```
1 pragma solidity >=0.4.0 <0.6.0;
2 contract SimpleStorage {
3  uint storedData;
4
5  function set(uint x) public {
6   storedData = x;
7  }
```

```
8   function get() public view returns (uint) {
9     return storedData;
10  }
11 }
```

Nella prima riga si nota la dicitura pragma solidity che indica la versione del linguaggio utilizzata per scrivere il contratto che quindi indica al compilatore la versione da utilizzare per una giusta compilazione. In seguito inizia la definizione del contratto. Questo possiede una variabile storedData di tipo Unsegned Int di 256bit, e due funzioni entrambi con visibilità pubblica, la prima set ha un argomento di tipo uint enessun return mentre nella seconda viene (nella firma) segnato il return e il tipo restituito (uint).

2.9.3 Gas

Quando si effettua una transazione o si carica uno smart contract la VM richiede un pagamento calcolato in gas. Il gas è una frazione dell'ether e viene richiesto per un'operazione come l'invio di una transazione infatti viene comunemente chiamato "transaction fee". Il gas viene anche richiesto quando si effettua una qualunque operazione tramite uno smart contract ed è calcolato in base alla complessità di questa e da quanta memoria va a utilizzare.

2.9.4 DApps

Le DApp - Decentralized APPlication sono applicazioni simili alle app tradizionali, con la differenza fondamentale che al posto di appoggiarsi su server centralizzati sfruttano le piattaforme blockchain e il loro network distribuito, in questo modo possono essere utilizzati da interfaccia con gli smart contract. Le DApp sono immutabili, quindi nessuno può modificare quello che avviene tramite l'applicazione.

Capitolo 3

Tecnologie utilizzate

3.1 Besu

Hyperledger è un open source creato per far progredire le cross-industry blockchain technologies.È una collaborazione globale, ospitata da The Linux Foundation, che include leader in finanza, banche, IoT, supply chain, manufacturing e tecnologia.

Figura 3.1: Hyperledger Besu logo

Hyperledger Besu è un Ethereum client - open source sviluppato sotto Apache 2.0 e scritto utilizzando Java. Può essere utilizzato per la rete pubblica Ethereum oppure per una rete permissioned privata, viene anche utilizzata per le reti di test come Rinkeby, Ropsten, and Görli. Hyperledger Besu include molti alcoritmi del consenso tra cui PoW, PoA(IBFT, IBFT 2.0, Etherhash, and Clique).

Hyperledger Besu offre molte proprietà come una EVM completa (Ethereum Virtual Machine) che permette l'invio e l'esecuzione di smart contract con le transazioni su Ethereum blockchain.

Besu usa un RocksDB key-value database per la persistenza locale dei dati della rete. I dati si dividono in due categorie:

- Blockchain: I dati della blockchain sono composti dal Block Header che che form la "catena" di dati utilizzata per verificare crittograficamente lo stato blockchain; block bodies che contengono la lista delle transazioni ordinate comprese in ciascun blocco; e ricevute di transazione che contengono metadati relativi all'esecuzione della transazione, inclusi i transaction logs.

- World State: Il world state è un mapping da addresses to accounts

Hyperledger Besu implementa Ethereum's devp2p network protocols per l'inter-client communication eun'altro sotto-protocollo per l'IBFT 2.

Hyperledger Besu mette a disposizione del programmatore le API di EEA, JSON-RPC utilizzando protocolli HTTP e WebSocket.

Hyperledger Besu ti permette di monitorare i nodi e le performance della rete, i nodi sono monitorati usando Prometheus o le metriche di debug JSON-RPC API method, invece la performance della rete viene monitorata con un qualunque Block Explorer

3.1.1 IBFT

Un meccanismo di consenso messo a disposizione da Besu (e sarà anche quello utilizzato nello sviluppo della blockchain) è l'IBFT versione 2.0 basato su PoA(Proof-of-Authority), questo è un protocollo è utilizzabile solo dalle reti private. Nelle reti IBFT 2.0 solo i nodi pre-approvati, conosciuti come validatori, possono validare transazioni e blocchi. I validatori prendono un turno per creare il prossimo blocco; prima che questo venga inserito sulla catena la maggioranza dei validatori(maggiorre del 66,6%) deve prima firmare il blocco. Per aggiungere o rimuovere un validatore si ha anche qui bisogno della maggioranza dei voti; IBFT 2.0 ha bisogno di minimo 4 nodi peressere

Byzantine fault tolerant, cioè l'abilità per una rete blockchain di funzionare correttamente e di raggiungere il consenso nonostante i nodi falliscano o propaghino informazioni errate ai peer.

Per usare IBFT 2.0, Besu richiede la scrittura di un genesis file (file contenente le configurazioni necessarie alla rete)

```
"config": {
  "chainId": 1981,
  "muirglacierblock": 0,
  "ibft2": {
    "blockperiodseconds": 2,
        "epochlength": 30000,
        "requesttimeoutseconds": 4,
        "blockreward": "5000000000000000",
        "miningbeneficiary": "0xfe3b557e8fb62b89f491
           6b721be55ceb828dbd73"
    }
}
```

Riferendosi al codice di esempio sopracitato (genesis file) si vedono i dati di config di una rete ibft2 identificando:

- blockperiodseconds: anche chiamato block time è il tempo alla cui scadenza il protocollo propone un nuovo blocco;
- epochlength: numero di blocchi che indica ogni quanto resettare i voti;
- requesttimeoutseconds: il timeout in secondi di un round per il cambio di validatore;
- blockreward: Importo della ricompensa opzionale in Wei per premiare il beneficiario;
- miningbeneficiary: Il beneficiario opzionale del blockreward.

IBFT Methods / Besu API

Besu per la gestione dell'IBFT offre delle API svivuppate tramite metodi utilizzabili tramite chiamate post. Tra i più utilizzati / quelli necessari c'è *ibft_discardValidatorVote* che prende in input un indirizzo di un nodo e restituisce un booleano secondo se l'operazione è andata a buon fine, con questo metodo è possibile rimuovere da un nodo la proprietà di validatore se più del 50% della rete effettua questo voto. Analogamente si usa *ibft_proposeValidatorVote* per proporre un nuovo nodo come validatore. infine ci sono molteplici metodi per ottenere delle metriche come *ibft_getValidatorsByBlockHash* e *ibft_getValidatorsByBlockNumber* che restituisce la lista di validatori che hanno firmato un blocco a partire dal numero del blocco o il suo hash.

3.1.2 Free Gas Network

Con Free Gas Network ci riferiamo ad una rete in cui vengono annullate le tasse "fee" delle transazioni. Utilizzando un client ethereum come BESU le tase vengono calcolate tramite il gas e il prezzo che ha 1 unità di esso; il costo di una transazione è quindi gas used * gas price. Utilizzando il comando –*min-gas-price=0* all'avvio su Besu imposteremo il gas price a zero, in questo modo il gas richiesto da una transazione sarà reso nullo in quanto il valore di questo sarà zero (gas x O = 0), quindi tutti i nodi potranno effettuare transazioni senza pagare alcuna tassa.

3.1.3 API Methods

3.2 Truffle

Truffle è il framework più utilizzato per lo svilupo su Etherieum, permette il management degli smart contract per tutto il loro ciclo di vita. L'installazione di truffle avviene tramite il comando

```
npm install truffle -g
```

che fa uso del gestore di pacchetti npm. Una volta installato è possibile inizializzare un progetto tramite

```
truffle init
```

una volta creata la nuova directory avrete la seguente struttura:

```
contracts/: Directory per i contratti sviluppati con Solidity
migrations/: Directory per i file di deploy
test/: Directory per i fiel di test degli  smart contract
truffle-config.js: Truffle configuration file
```

3.2.1 Compile

Con la Truffle suit è possibile compilare i propri smart contract. Con il comando

```
truffle compile
```

Verranno compilati i contratti, quindi si avrà una visione dei possibili errori oppure se compilati correttamente verrà generato per ogni contratto le sue ABI - Application Binary Interface, queste sano scritte tramite modello JSON, avranno lo stesso nome del contratto e saranno nella directory *build/contract*.

3.2.2 Test

Con truffle è possibile scrivere ed eseguire dei test per i contratti. questi andranno scritti in javascript, firmati con estensione *.spec.js* e salvati nella directory */test* Con il comando

```
truffle test
```

vengono eseguiti i vari programmi sulla rete di test e verrà visualizzato su riga di comando quali test sono andati a buon fine e quali hanno generato dei problemi che verranno mostrati.

3.2.3 Deploy

All'avvio di truffle init viene creato un file nominato truffel-config.js - file formato json che contiene le configurazioni di truffle, inizialmente sarà vuoto ma specificando una qualunque rete Trulle ci permette di effettuare il deploy / migration dei nostri smart contract sulla rete selezionata scrivendo il file di migration specificando il contratto da inviare(uno per ogni contratto) ed eseguendo il comando

```
1 truffle migrate --network=//nome-rete
```

3.3 Node js

Node js è una tecnologia open source di sviluppo software, orientata agli eventi per l'esecuzione di codice JavaScript costruito su Google Chrome's V8 JavaScript engine. È un linguaggio event-driven e usa la programmazione

asincrona, non supporta il multithreading e il modello di programmazione si basa sulle funzioni di callback cioè funzioni che vanno in esecuzione solo dopo che è stato lanciato l'evento che che indica che l'elaborazione è terminata e il valore di output è disponibile. Questo ambiente ci permette di utilizzare il linguaggio javascript sia per front-end sia back-end, rendendo possibile la creazione di un server tramite l'uso di pacchetti come Express che sarà in ascolto di default sulla porta 1010:

```
1 const express     = require('express');
2 const app = express();
3 //Home Page
```

```
4 app.get('/', (req, res) => {
5     res.sendFile(path.join(__dirname, '/html/index.html'));
6 });
```

con questo codice definiamo anche la risposta che darà il server alla richiesta get effettuata ('/' - indica la ricerca base del solo server) e sarà il file*index.html*

3.3.1 Web3

Capitolo 4

Caso d'uso

4.1 Problema Iniziale

Il progetto blockchain è stato ideato e sviluppato come proposta di soluzione ai problemi nella gestione del tracciamento, invio, certificazione e mantenimento di dati riguardanti "spostamenti". Questi spostamenti sono informazioni(LOG) inviate da un qualunque ente che metta a disposizione della società o di un qualunque servizio che preveda dei mezzi pubblici o privati quali ad esempio autobus, treni, taxi etc... Queste informazioni vengono ora raccolte, analizzate e utilizzate su di modelli di storage a fogli di calcolo (Ad Esempio EXCEL - Programma Micrsoft). I fogli di calcolo offrono alcuni vantaggi come la semplicità con cui vengo creati, scritti e salvati benché offrano un'interfaccia poco user friendly('facilmente utilizzabile') guardando tutti i possibili attori, mentre il progetto si focalizza sui difetti come la pubblicazione/condivisione delle informazioni o l'interrogazione di queste in quanto utilizzando semplici fogli non vengono proposte alcune regole di struttura e organizzazione,e ci si pone particolare importanza alla sicurezza e all'affidabilità di queste informazioni e che non vengano modificate durante la condivisione, quindi la possibile certificazione di essi.

4.2 Soluzione

La soluzione proposta si offre di risolvere tutti i problemi sopra elencati come la certificazione, salvataggio e interrogazioni di informazioni. Viene ideata una blockchain privata che avrà funzione di ente (decentralizzato) certificatore, questa non utilizza nessuna moneta creando una Free Gas Network e che con l'ausilio di appositi smart contract(scritti e caricati autonomamente) ci permette di salvare un codice che andrà ad identificare un determinato gruppo di spostamenti come un codice Hash che usufruendo della struttura e utilizzo della blockchain non potrà essere modificato, ciò implica che si potrà sempre verificare la correttezza del gruppo di spostamenti richiesti ricreando e controllando il loro codice.

La memorizzazione dei dati viene invece affidata ad un database relazionale, utilizzando nel progetto il DBMS(DataBase Managenent System) MySql, che ci permette di salvare grandi quantità di dati con una efficiente organizzazione gestita con la creazione di tabelle così da essere facilmente interrogabile in futuro.

L'interfaccia comune è gestita con un server sviluppato tramite tecnologia node js che con una single Page Application avrà la funzione di interfaccia user friendy con funzioni di memorizzazione per i log degli spostamenti su Database, calcolo e salvataggio dei loro codici hash sulla blockchain al tempo stimato e quando necessario, cioè quando e ovunque verranno richiesti dei dati e avverrà l'interrogazione del DataBase sarà reso obbligatorio il controllo di questi con il codice sulla blockchain. Inserendo questo WebServer intermedio o server proxy si andrà ad eliminare il passaggio di dati non propriamente protetto e rende partecipi tutti i singoli attori dell'attività.

4.2.1 Problematiche

Utilizzando delle nuove tecnologie sorgono comunque nuove problematiche che non sono state affrontate nello sviluppo in quanto non inerenti ai fini del

progetto.

Una prima problematica si sviluppa utilizzano un server proxy. Avendo un singolo server di accesso al database a alla blockchain sequesto non viene correttamente protetto e costantemente controllato è soggetto ai classici attacchi come un DDOS - Distributed Denial of Service in cui si utilizzano molteplici messaggi fittizi (come un inizio di HandShake per una connessione TCP) per far si che il server non possa sostenere tutti i servizi e essendo l'unico punto di accesso bloccherebbe l'intero accesso alla rete blockchain.

Una caratteristica che rende sicura la blockchain pubblica è la molteplicità di nodi, questa con una blockchain privata come la nostra va a decadere con il discendere del numero di nodi; utilizzando un meccanisco di conseso basato su PoA(Proof of Autority) si ha infatti bisogno di un minimo di 4 nodi per essere resistente al problema bizantino.

Per evitare l'appesantimento della Blockchain si è pensato di salvare su di essa solo un codice identificativo(codice Hash) per un gruppo di Log. Questo implica che con l'aumentare dei log identificati da un singolo codice hash diminuisca la sicurezza che questo apporta infatti sarà più facilmente utilizzabile un attacco come l'attacco del compleanno che ha come obiettivo quello di generare una collisione cioè di trovare dei dati fittizi ai Log originari che però generano lo stesso codice Hash, questi dati fittizi potranno essere quindi sostituiti nel DB ma verranno comunque considerati certificati dal sistema in quanto produrranno lo stesso codice.

4.3 Attori

Il caso d'uso per il progetto blockchain prevede la partecipazione di diversi attori quali:

Un Terminal User o utente finale è un comune dipendente di un ente che partecipa alla blockchain il quale ha il compito di comunicare i propri spostamenti/Log o qualunque informazione di cui si preveda il salvataggio;

Gli Admin sono dei dipendenti di enti partecipanti che vengono segnati dagli stessi come amministratori che quindi posseggono particolari oneri come il possesso e la trasmissione di una chiave privata;

Il proprietario/gestore della blockchain avrà il compito di gestire l'intera blockchain privata con l'amministrazione che ne segue, come la supervisione dei nodi presenti, il loro funzionamento e la loro caratterizzazione come validatori.

4.4 Scenario di utilizzo

Il Progetto prevede uno scenario di utilizzo diverso seguendo la distinzione degli attori. Per l'utilizzo si prevede che ad ogni ente partecipante al progetto gli venga assegnato un account, cioè una copia di chiavi privata e pubblica che serviranno per interagire con la blockchain, inoltre ogni ente dovrà inserire i propri dipendenti nel Database e specificare il ruolo di essi, se admin o Terminal User.

Un Terminal User, una volta effettuato l'accesso, viene portato ad un'interfaccia in cui può inserire la città che sarà selezionata come Start dello spostamento e in seguito viene spostato in una seconda interfaccia da cui può terminare lo spostamento o annullarlo, se annullato potrà cominciare un nuovo spostamento dalla precedente interfaccia, il completamento di questo avverrà solo se compila i campi necessari quali la città di Termine e la distanza percorsa indicata in Kilometri.

Un admin, una volta effettuato l'accesso, potrà a differenza di un Terminal User effettuare delle query/ interrogazioni riguardo gli spostamenti compiuti, inserendo una data otterrà tutti gli spostamenti che sono stati certificati da una transazione inserita in quella determinata data, da qui potrà anche accedere ai dettagli della transazione o del blocco che la contiene riferendoci alla blockchain, inoltre, quando il sistema lo richiede, ha il compito di

inserire la Private Key dell'utente(ente) che verrà utilizzata per la scrittura su blockchain.

Capitolo 5

Sviluppo

5.1 Schema Progetto

5.2 Blockchain Ibrida

5.3 Smart Contract

5.3.1 Boxing

5.4 WebApp

5.4.1 Single Page Application

5.4.2 Input

Inserimento in un DB

Inserimento nella Blockchain

5.4.3 Output

Report di Controllo

Monitor Blockchain

Capitolo 6

Sviluppi futuri

6.1 Analisi costi

6.2 Immissione nella blockchain pubblica

6.3 Blockchain pubblica come certificazione

6.4 Svilupo full Blockchain

Elenco delle figure