Contesti di applicabilità di Blockchain

Proposta di un sistema di voto elettronico a trust limitato

Matteo Sovilla

Tesi presentata per il conseguimento della
Laurea triennale in Informatica

Dipartimento di Matematica
Università degli studi di Padova
31 Febbraio 2018

Indice

Elenco delle figure

Capitolo 1

Introduzione

Introduzione

Capitolo 2

Origini di Blockchain

2.1 L'obiettivo iniziale: creare asset digitali

Il trasferimento di *asset*, intesi come beni materiali o somme di denaro, è sempre stato un processo delicato e in gran parte manuale, che richiede alle controparti coinvolte procedure di verifica manuali, non automatizzate. Ingenti sforzi si sono profusi nel creare sistemi sempre più veloci e sicuri che permettessero di manipolare tali asset per via informatica, ma i progressi in tal senso non si possono dire pienamente soddisfacenti. Anche solo considerando il semplice trasferimento di valuta tra due conti correnti bancari, infatti, è richiesto all'utente il pagamento di commissioni e il rispetto di tempi di attesa molto più lunghi di quelli a cui il progresso tecnologico ci ha abituato in altri ambiti. Questa difficoltà di digitalizzazione della gestione del possesso di beni è legata ad un problema legato proprio alla facilità di creazione e replicazione di dati informatici.

2.1.1 Il problema del double spending

Immaginiamo di dover creare un asset digitale adatto ad essere trasferito in maniera analoga a quanto viene fatto abitualmente con il denaro contante. Un primo ingenuo passaggio è quello di associare un valore ad un dato, sia esso una stringa, un numero, o un generico file, liberamente accessibile oppure in cifrato. Per loro natura però, i dati digitali sono estremamente facili da replicare: trattandosi in fondo di sequenze di 0 e 1 più o meno lunghe, generarne una copia esatta è un procedimento agevole e sostanzialmente privo di costo. Questo si scontra con il proposito di creare del valore, dal momento che si ha la necessità di rendere *scarso*, ovvero *limitato*, il bene. Prendiamo ad esempio il denaro contante: non è accettabile che la stessa cifra sia trasferita due volte dalla stessa persona a due destinatari differenti semplicemente replicando la banconota.

2.1.2 Una soluzione: Database centralizzati

La soluzione al problema del *double-spending* finora adottata è stata l'affidamento delle transazioni economiche digitali ad un ente esterno fidato (e.g. i sistemi e-banking) che garantisse le identità degli utenti e potesse verificarne l'effettiva disponibilità economica. È il database centrale che traccia la storia delle transazioni effettuate da ogni utente e prima di autorizzare la successiva verifica che non ci siano

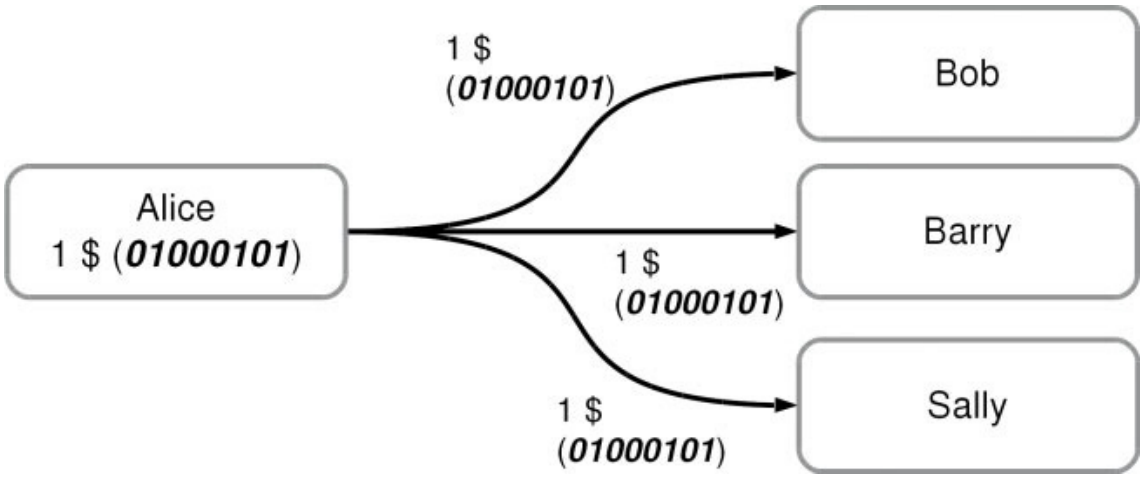

Figura 2.1: Double-spending problem - [11], p. 11

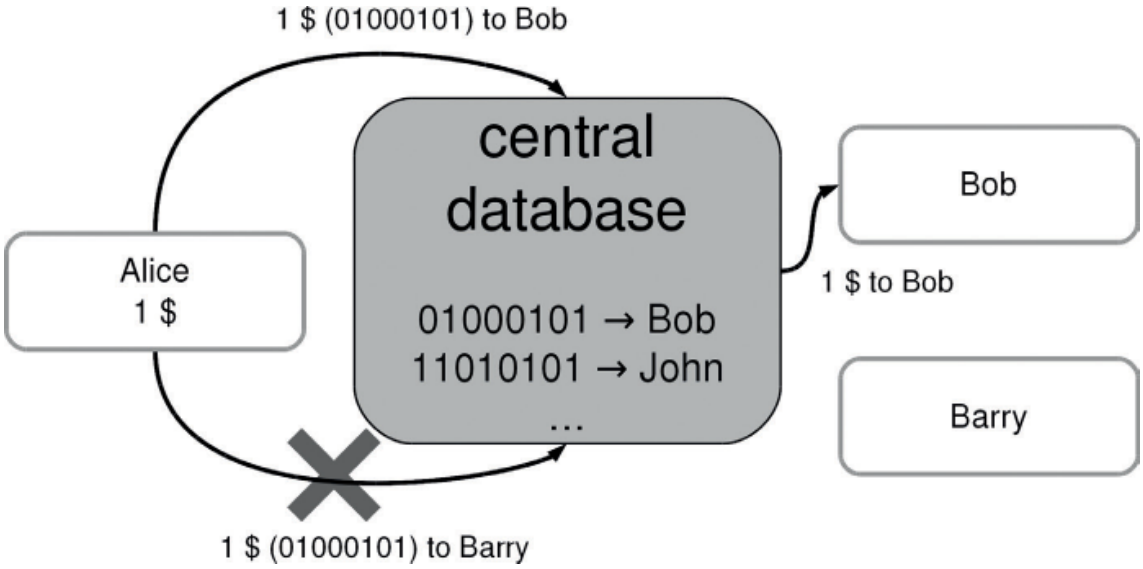

Figura 2.2: Central counterparty holding a centralized database - [11], p. 12

incongruenze.
Affidarsi ad un sistema centralizzato risolve il problema, ma porta con sé una serie di criticità.
Prima di tutto la necessità di *trust*, fiducia, nei confronti dell'intermediario centrale: esso infatti ha libero accesso ai dati degli utenti e alla loro storia transazionale, si occupa dell'affidabilità dell'intero sistema, può autorizzare o negare transazioni e bloccare interi utenti. Si tratta di un nodo cruciale per la sicurezza del sistema, un attacco riuscito nei suoi confronti comporterebbe conseguenze disastrose. Inoltre, il database rappresenta un *single point of failure*: abbattere il nodo centrale significa abbattere l'intera rete.

2.1.3 Difficoltà dei sistemi distribuiti

Un'alternativa ad un sistema gestito da un intermediario centrale è rappresentata da un sistema distribuito decentralizzato. In un sistema distribuito due o più nodi collaborano per svolgere un compito prefissato in maniera trasparente all'utente, che

si interfaccia con un'unica piattaforma logica. Emerge la necessità di coordinare i diversi attori: i nodi possono essere onesti e funzionare in modo corretto oppure difettosi, mal funzionanti o maliziosi. Anche se qualche nodo si rivelasse difettoso o la connessione venisse a mancare, un buon sistema distribuito dovrebbe essere *resiliente*, ovvero assorbire l'imprevisto e continuare a lavorare senza problemi. La complessità di progettazione di sistemi di questo tipo non è indifferente, è stata area di ricerca fertile per molti anni e lo è tuttora. Un risultato importante è il teorema CAP, che dimostra come non possano coesistere tutte le caratteristiche desiderate in uno stesso sistema.

Teorema CAP

Altrimenti noto come Teorema di Brewer, è stato proposto da Eric Brewer nel 1998 e dimostrato poi da Seth Gilbert e Nancy Lynch nel 2002 [12]. L'enunciato è il seguente:

Thm: Un qualsiasi sistema distribuito non può garantire simultaneamente le tre seguenti proprietà:

- **Coerenza (Consistency)** è la proprietà che assicura che tutti i nodi del sistema posseggono la stessa copia aggiornata dei dati;
- **Disponibilità (Availability)** indica che il sistema è attivo, accessibile e pronto a ricevere input e a fornire output corretti in un tempo conforme alle attese;
- **Tolleranza di partizione (Partition tolerance)** assicura che anche nel caso un gruppo di nodi crollasse per qualche motivo, il sistema continuerebbe ad operare correttamente.

Problema dei Generali Bizantini

Si tratta di un dilemma posto da Leslie Lamport (con Marshall Pease e Robert Shostak) nel 1982 [17]:

Un gruppo di generali a capo di diverse sezioni dell'esercito bizantino sta pianificando di attaccare una città. L'unico modo a loro disposizione per comunicare è mediante messaggeri, e *hanno bisogno di concordare il momento dell'attacco per vincere*: raggiungendo un accordo l'attacco riuscirà e la città sarà conquistata, mentre non riuscire a determinare un momento univoco per agire porterebbe ad un attacco frammentato e fallimentare. Potrebbero esserci traditori tra di loro, i quali comunicherebbero messaggi discordanti per minare la riuscita dell'operazione: è quindi necessario stabilire un sistema affidabile per raggiungere il consenso sulle tempistiche di attacco. È facile l'analogia con un sistema informatico distribuito in cui i generali siano rappresentati dai nodi e i messaggeri come un'infrastruttura di rete.
Una possibile soluzione al problema dei generali bizantini in un sistema sincrono è stata proposta da Miguel Castro e Barbara Liskov e pubblicata in *Practical Byzantine Fault Tolerance* [8].
Bitcoin è la prima tecnologia ad implementarne una soluzione pratica basata su Proof-of-Work.

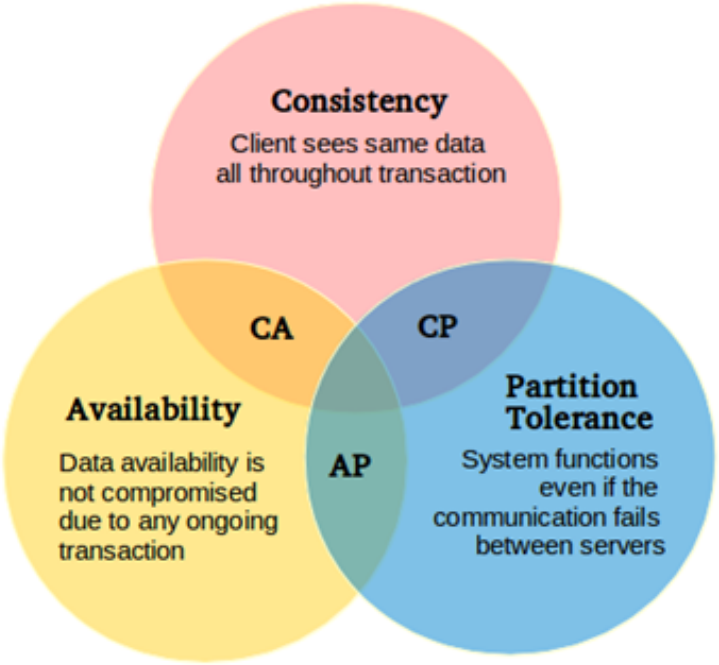

Figura 2.3: Teorema CAP [10]

Algoritmi di consenso

La ricerca sui modi per raggiungere il consenso tra nodi di un sistema distribuito si è sviluppata molto nel periodo successivo all'introduzione di Bitcoin, pertanto sarebbe quantomeno ottimistico fornire un elenco di metodi che volesse essere esaustivo. È possibile tuttavia individuare due categorie principali di algoritmi:

- **Basati sulla fault-tolerance Bizantina**, si affidano ad un insieme di nodi che si scambiano messaggi firmati. Non richiedono impiego di risorse intensivo, sono affidabili fintanto che $N > 3F$ dove N è il numero di nodi e F è il numero di nodi malevoli;
- **Basati sul riconoscimento di un leader**, mettono i nodi in competizione per essere riconosciuti come leader e il nodo vincitore propone un accordo. Tipicamente richiedono un impiego di risorse considerevole come "gara" e la sicurezza delle loro applicazioni spesso si basa proprio sulla non convenienza di un tale investimento per un attaccante.

2.2 Bitcoin

Il 31 ottobre del 2008, uno sconosciuto (o un gruppo) sotto lo pseudonimo di Satoshi Nakamoto annuncia la pubblicazione di un suo paper [19] in cui presenta un nuovo sistema di moneta elettronica: Bitcoin. Non si tratta del primo esperimento di moneta elettronica, ma la notizia fa scalpore poiché Bitcoin viene presentato come completamente peer-to-peer, senza necessità di mediazione da parte di un intermediario fidato.
Il sistema Bitcoin si basa, in estrema sintesi, su di un registro distribuito di transazioni. Chiunque voglia usufruire del sistema per ricevere pagamenti o effettuarli

deve possedere una coppia di chiavi pubblica e privata. La chiave privata serve a firmare le nuove transazioni, mentre la chiave pubblica viene diffusa e utilizzata per verificare la firma apposta sul pagamento. L'utente è identificato all'interno del registro tramite un hash della sua chiave pubblica, ovvero il suo *indirizzo Bitcoin*. Nella proposta di transazione, l'utente deve dichiarare:

- l'indirizzo del destinatario;
- l'importo da trasferire;
- il riferimento alla transazione tramite cui ha ottenuto il possesso dei bitcoin che sta spendendo.

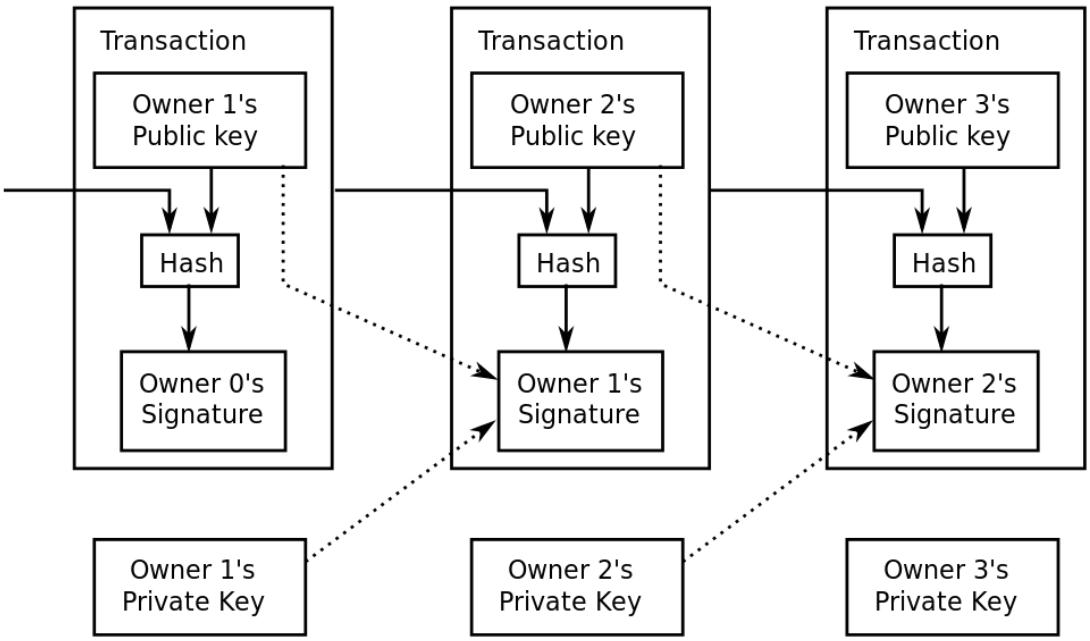

Figura 2.4: Struttura di una transazione

Deve inoltre fornire prova della sua identità firmando la proposta tramite la sua chiave privata.

2.2.1 Evitare il double-spending: la nascita di Blockchain

La crittografia asimmetrica e in particolare i concetti di firma digitale su sui si fonda il sistema descritto fino a questo punto sono ben noti da anni e largamente utilizzati molto tempo prima della nascita di Bitcoin. Non sono sufficienti però ad evitare il problema del double spending: per fare ciò è necessario assicurare l'immutabilità del registro e prevenire la creazione di transazioni non valide sfruttando la mancanza di "accordi" tra i nodi della rete. La tecnologia adottata da Bitcoin per questo scopo è nota come *Blockchain*.
Il registro distribuito di Bitcoin prende la forma di una serie di "blocchi" di transazioni legati fra di loro tramite hash crittografico: ogni blocco contiene, oltre alle transazioni che lo compongono, anche l'hash del blocco precedente permettendo

quindi di individuare un ordine totale nell'insieme dei blocchi che può prendere quindi il nome di *catena*. La catena è mantenuta da ogni partecipante: *ciascun nodo ne conserva una copia locale*, che tiene aggiornata sulla base di quanto accade nella rete.

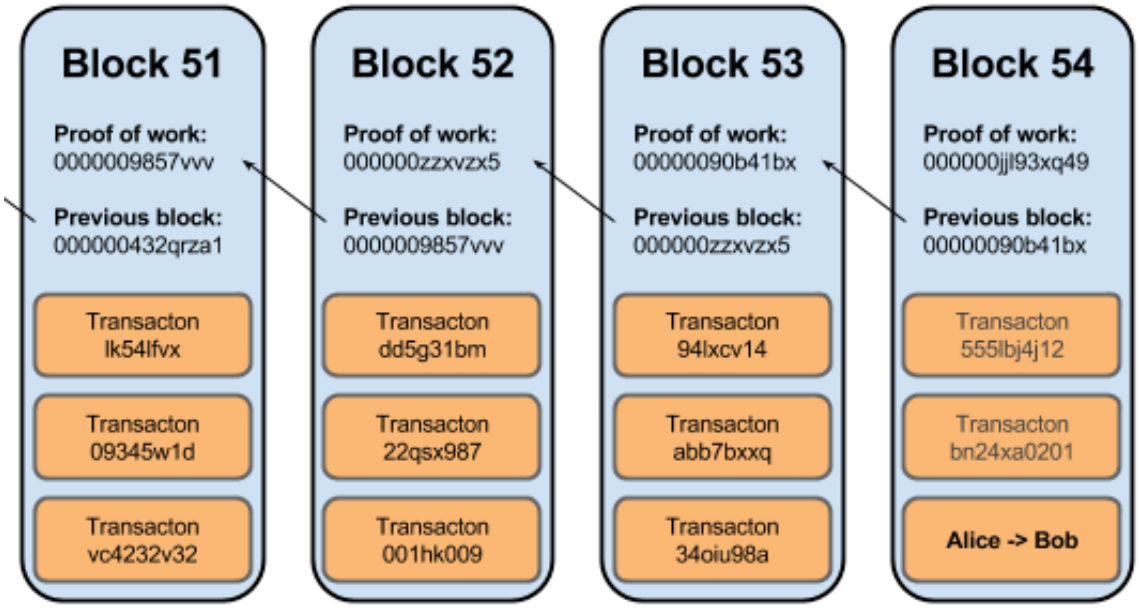

Figura 2.5: Struttura a blocchi del registro

Consenso tramite proof-of-work

Rendere ordinato il registro permette di determinare in maniera univoca quale transazione sia avvenuta per prima, e quindi nel caso di doppia spesa di una particolare somma rifiutare le transazioni successive alla prima. Ciò non impedisce però ad un utente malevolo di creare una nuova catena che riassegni arbitrariamente certe somme (ad esempio modificando il destinatario di una transazione da lui effettuata) e presentarla come corretta al resto della rete. Bitcoin affronta questa eventualità rallentando la produzione dei blocchi attraverso la richiesta di una *proof-of-work*. Ciascun nodo che volesse validare un nuovo blocco e proporlo al resto della rete (chiamato *miner*) deve prima risolvere un problema computazionalmente molto oneroso (risolvibile solamente a forza bruta) basato su di un hash del blocco in "lavorazione" [6]. In particolare, nel blocco è inserito un numero detto *nonce* che deve essere determinato dal nodo proponente affinché l'hash del blocco stesso cominci con un determinato numero di 0 iniziali. Il numero di 0 richiesti influenza la difficoltà del problema: viene determinato algoritmicamente sul tempo medio di creazione degli ultimi blocchi e ciò fa in modo che il tempo necessario si assesti non sotto i 10 minuti. Questo *sforzo* necessario da parte dei miner fa sì che un nodo malevolo che volesse creare nuova catena alterata dovrebbe ricalcolare la soluzione a tutti i problemi matematici relativi a tutti i blocchi successivi a quello alterato. Dal momento che la rete accetta come corretta solamente la catena più lunga, tale attaccante dovrebbe validare i blocchi più velocemente del resto della rete, per poter ad un certo punto superare la lunghezza della blockchain "ufficiale" e veder riuscire il suo attacco.

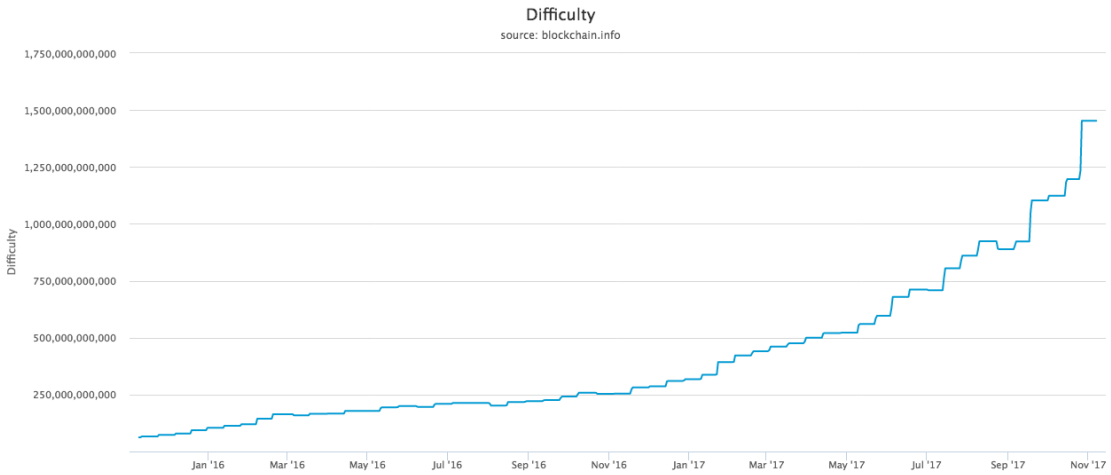

Figura 2.6: Difficoltà di creazione di un nuovo blocco

Algoritmo in dettaglio Per il calcolo della difficoltà Bitcoin si serve di un valore obiettivo detto *Target*. Formalmente, affermare che un valore binario k deve cominciare con un determinato numero di 0 equivale a dire che $k < T$ per un target T. Il target iniziale è stato fissato a 0x00000000FFFF * 2^{208}, ed in seguito ricalcolato ogni 2016 blocchi creati. Poiché ogni valore hash è un numero di 256 bits, la probabilità che un hash sia minore di T è:

$$p(T) = \frac{T}{2^{256}}$$

richiedendo quindi un numero medio di tentativi

$$N(T) = \frac{1}{p(T)} = \frac{2^{256}}{T}$$

per trovare la soluzione. Se gli ultimi 2016 blocchi risultano avere un tempo medio di creazione t_m per un singolo blocco, possiamo stimare la velocità dei minatori come

$$V_{est} = \frac{N(T)}{t_m}.$$

Il tempo previsto per creare un blocco con difficoltà T e velocità di hash v è allora

$$t(T, v) = \frac{N(T)}{v}.$$

Il valore di *Target* deve essere regolato ad un valore T' tale che $t(T', v_{est}) = 600s = 10min$. Si ricava quindi l'equazione

$$600s = t(T', v_{est}) = \frac{N(T')}{v_{est}} = \frac{N(T')}{\frac{N(T)}{t_m}} = t_m * (\frac{T}{T'})$$

da cui

$$T' = \frac{t_m}{600}T.$$

Il guadagno dei miner

L'introduzione di un simile onere pesa tanto su di un ipotetico attaccante quanto sul resto della rete: per fornire un adeguato incentivo per il mantenimento della rete, la prima transazione di ogni blocco è *autorizzata a creare nuovi bitcoin* in favore del miner che l'ha validato. Questo è l'unico modo in cui nuovi bitcoin possono essere creati, e l'ammontare di questa ricompensa è anch'esso algoritmicamente determinato. La presenza di questa ricompensa fa in modo che sia più profittevole, da parte di un nodo con sufficiente potenza di calcolo da impensierire la stabilità della rete, investire queste risorse nel mantenimento della rete stessa svolgendo il ruolo di miner invece che in attacchi come quello descritto sopra.

Offerta anelastica e scenario futuro

È necessario un piccolo appunto sulla possibile evoluzione di questo equilibrio: la ricompensa per la validazione di un blocco è infatti destinata a ridursi con il tempo. L'algoritmo prevede infatti che questa venga dimezzata circa ogni 4 anni fino a raggiungere una quantità complessiva di bitcoin immessi che si aggira attorno ai 21 milioni. L'ultima frazione di bitcoin generata sarà creata nel 2141: a quel pun-

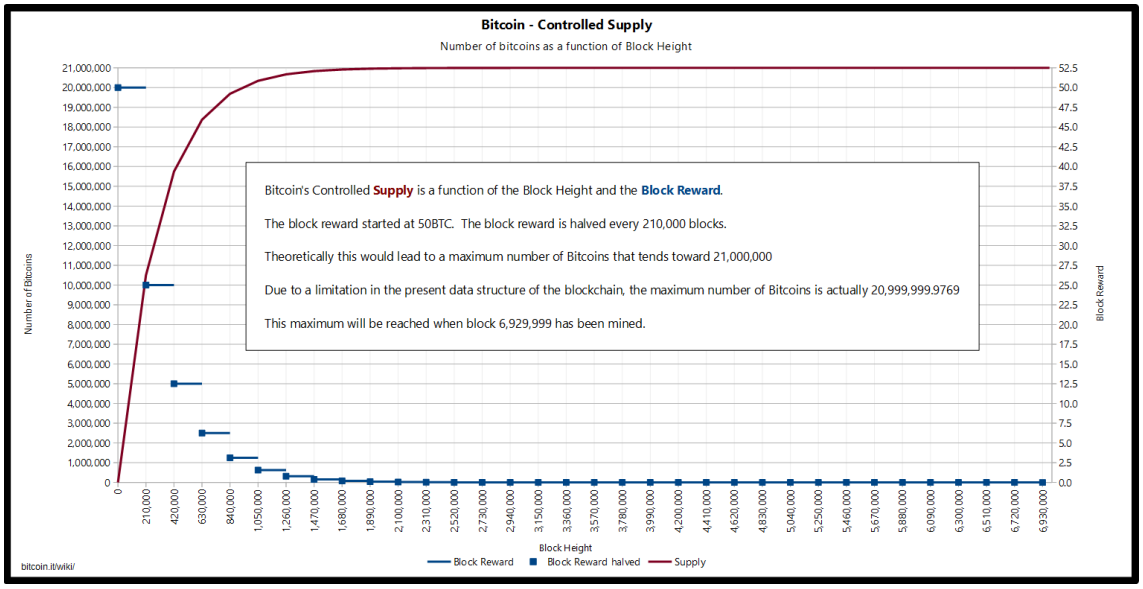

Figura 2.7: Emissione fissata di bitcoin

to verrà meno, apparentemente, l'incentivo al mining "onesto". Nonostante questo possa sembrare uno scenario remoto per via della data ancora sufficientemente distante nel tempo, il problema è più attuale di quanto sembri per via degli enormi costi delle attrezzature per il mining. La competizione in quest'ambito si è accesa e ha dato il via ad una corsa continua per accumulare la maggior potenza di calcolo possibile, alzando moltissimo la quantità di risorse necessarie in termini di hardware ed energia. Il momento in cui questi costi supereranno il valore della ricompensa è quindi molto più vicino della data di emissione dell'ultimo bitcoin.
A sostenere i costi affrontati dai miner quando non sarà più sufficiente la quantità di bitcoin estratta saranno delle somme donate volontariamente associate ad ogni transazione. Il protocollo permette infatti di includere ad ogni transazione una *fee*, una "tassa" pagata dal richiedente ed indirizzata al miner che registrerà la transa-

zione sulla blockchain.
Finché l'incentivo economico sarà sufficiente, il lavoro onesto dei miner sarà sempre conveniente rispetto all'utilizzo malevolo della propria potenza di calcolo. La stabilità di questo equilibrio determinerà l'effettiva sopravvivenza del sistema Bitcoin negli anni a venire.

Capitolo 3

Potenzialità e limiti

3.1 Generalizzazione

In seguito all'esordio di Blockchain assieme a Bitcoin sono state presentate numerose proposte per sviluppi ed implementazioni della tecnologia. Nel seguito di questa sezione si presenta quello che vuole essere uno schema delle caratteristiche salienti di un sistema basato su Blockchain: sono riportati le principali scelte di progettazione che lo caratterizzano e alcune delle soluzioni ad oggi adottate. La ricerca è estremamente attiva a riguardo, perciò questo elenco non sarà e non vuole essere esaustivo: lo scopo è quello di chiarire a che tipologie di sistema si fa riferimento parlando genericamente di *Blockchain*.

3.1.1 Definizione tecnica

Una buona definizione generica di Blockchain è quella proposta da Imran Bashir in *Mastering Blockchain* [5]: essenzialmente si tratta di un registro distribuito peer-to-peer, crittograficamente sicuro, append-only, immutabile (o meglio, estremamente difficile da modificare) e aggiornabile solo previo consenso da parte dei peer. Esistono ovviamente molte altre definizioni, ciascuna con enfasi diversa sui vari aspetti di Blockchain, tuttavia l'estrema sintesi di quella di Bashir permette di concentrarsi sugli aspetti imprescindibili che la caratterizzano, lasciando alle scelte di progettazione successive il compito di determinarne i particolari.

3.1.2 Elementi costitutivi di una Blockchain

Permessi di accesso

La prima grande suddivisione in macro categorie dei sistemi basati su Blockchain ne riguarda le modalità di accesso. La Blockchain può essere:

- **Pubblica** nel momento in cui il sistema è aperto al pubblico e chiunque può installarne un client, scaricare il registro e partecipare attivamente ai processi di validazione dei blocchi. Tipicamente ogni utente scarica e mantiene una copia locale del registro e tramite un sistema di consenso distribuito si raggiunge un accordo con il resto della rete sull'effettivo stato "ufficiale" dei dati. Tipicamente si affidano ad algoritmi di consenso basati su proof-of-work o proof-of-stake. Sono anche dette *permissionless ledgers*, e l'esempio più noto è Bitcoin;

- **Privata** nel caso l'accesso sia ristretto ad un limitato insieme di utenti o organizzazioni che abbiano deciso di condividere dei registri comuni. In questa accezione di Blockchain acquista importanza l'autenticazione degli utenti, e si ottiene una maggiore riservatezza dei dati a costo di rinunciare alla natura completamente "*trustless*" delle blockchain pubbliche. Ciò porta a differenze sostanziali nella gestione del consenso: è possibile prescindere da prove onerose come la proof-of-work, e soprattutto viene a mancare la necessità di rendere vantaggioso a ciascun peer il mantenimento del sistema. Il grado di chiusura di questo tipo di blockchain può variare anche molto: è possibile ad esempio permettere a ciascun utente di osservare il registro ma limitarne ad un insieme ristretto la modifica. Sono anche dette *permissioned ledgers* e esempi possibili sono Ripple e i sistemi costruiti su Hyperledger.

Meccanismi di consenso

La tipologa di consenso adottata caratterizza fortemente l'intero sistema Blockchain, ed è strettamente legata alla tipologia di accesso scelta.

- **Proof of work**: è il meccanismo di consenso adottato da Bitcoin. Richiede la soluzione ad un problema matematico computazionalmente impegnativo (e.g. *Hashcash*) come prova delle risorse investite per far accettare la propria proposta dalla rete;

- **Proof of stake**: è il meccanismo di consenso usato da Peercoin (e si parla di adottarlo prossimamente in Ethereum [7]). Sbilancia la probabilità di far accettare una proposta di cambiamento del registro in favore dei nodi di maggiore "importanza" all'interno della rete. In una Blockchain che rappresenti una valuta, ad esempio, l'importanza consiste nella quantità di valuta posseduta. Nella variante dei consensi *deposit-based* invece si acquista importanza versando quella che è in pratica una caparra. L'idea di base è che un attaccante dovrebbe investire talmente tante risorse nell'oggetto del suo attacco da renderlo non profittevole, in quanto le ricadute sul suo investimento sarebbero troppo pesanti. Una variante è la *Delegated Proof of Stake* in cui i nodi importanti delegano la validazione di ciascuna transazione tramite votazione;

- **Proof of elapsed time**: è un meccanismo di consenso introdotto da Intel [14] allo scopo di limitare l'enorme dispendio di energie dei sistemi proof-of-work. Utilizza hardware proprietario certificato per garantire l'affidabilità di un timer, ottenendo di fatto lo stesso effetto dei PoW con irrisorio consumo di energia;

- **Reputation based**: è una variante dei sistemi proof-of-stake dal nome autoesplicativo. Il leader viene eletto sulla base della reputazione che si è costruito nel tempo all'interno della rete, che può basarsi su lavoro effettuato o su espressione degli altri nodi;

- **Federated (Byzantine) agreement**: è un sistema non-trustless, e pertanto adatto a blockchain di tipo permissioned. I nodi tengono traccia di un gruppo di peer pubblicamente riconosciuti come affidabili e propaga solo le transazioni validate dalla maggioranza dei nodi affidabili. È implementato nello Stellar Consensus Protocol [18];

- **Practical Byzantine Fault Tolerance**: è la soluzione originale al problema dei generali bizantini proposta da Castro e Liskov nel 1999 [8].

Struttura e organizzazione dei dati

La scelta è in questo caso tra:

- Tokenized blockchain;
- Tokenless blockchain.

Blockchain è nata come supporto ad un sistema di scambio di token, e le applicazioni al momento più diffuse rispecchiano questo legame nei confronti di un'unità minima di informazione trasferibile. Soprattutto nel caso di blockchain permissioned, però, è possibile prescindere dal concetto di token, condividendo informazioni analogamente a quello che si può fare mediante un database tradizionale. Hyperledger permette di implementare sistemi di questo tipo, basati su un database condiviso (lo *state*) in cui ogni modifica è registrata in una blockchain che funge da "log certificato".

Indipendenza da altre catene

Un ultima distinzione degna di nota è quella tra blockchain stand-alone e sidechain [4]. Finora, le implementazioni di sidechain riguardano esclusivamente le tokenized blockchain. Una sidechain si appoggia ad una stand-alone e permette i trasferimenti di token dall'una all'altra. Si appoggia su varianti di proof-of-stake in cui è necessario impegnare dei coin di una catena per generarne nell'altra. I trasferimenti possono essere unidirezionali o bidirezionali. L'interesse sulle sidechain è estremamente alto poiché queste abilitano la creazione di monete che si appoggiano a criptovalute diffuse come Bitcoin e ne vanno a colmare delle lacune. Interessante a proposito il punto di vista di Ferdinando Ametrano sull'uso di sidechain per ottenere la stabilità dei prezzi con una criptovaluta, riportata nel suo paper del 2014 "*Hayek Money: the Cryptocurrency Price Stability Solution*" [1].

3.1.3 Teorema CAP e Blockchain: il concetto di Aging

Può sembrare che le varie implementazioni di Blockchain illustrate violino il teorema CAP presentato in 2.1.3. Ciò è inesatto: in particolare Blockchain persegue disponibilità e tolleranza di partizione in maniera istantanea, mentre la coerenza è sacrificata e ottenuta solo attraverso un lasso di tempo. Si parla in questo caso di *coerenza eventuale*, ed è il motivo per cui il protocollo Bitcoin è stato di fatto artificialmente rallentato mediante un problema a difficoltà dinamica, che assicurasse un impegno per la proof-of-work di circa 10 minuti. Un qualsiasi sistema Blockchain può dirsi coerente solo facendo riferimento a transazioni con una certa "età", che siano state accettate e validate col tempo dai vari peer.

3.2 Benefici e limitazioni *[development]*

Spunto di riflessione: blockchain innovazione sia in ambito economico che in ambito informatico: rivoluziona nel primo caso, innova nel secondo.
LINK: Ending the bitcoin vs blockchain debate

Nella gran quantità di materiale prodotto su Blockchain dalla sua presentazione ad oggi si contrappongono le posizioni di chi la considera la base della "quarta rivoluzione industriale" [21], o il quinto "disruptive computing paradigm" [24], a chi ne critica fortemente l'efficacia sostenendo che sia applicabile unicamente ai sistemi di cripto valute [3].
L'analisi della questione è complicata dall'intrinseca interdisciplinarità dell'argomento. Ferdinando Ametrano parla di Bitcoin [2] come argomento all'incrocio di quattro discipline: crittografia, reti di calcolatori e trasmissione dati, teoria dei giochi e teoria economica e monetaria. Anche affrontando solamente l'aspetto tecnologico di Blockchain intesa in senso più generale, è fondamentale rendersi conto della moltitudine di sfaccettature possibili: la maggior parte dei confronti tende a semplificare e ad ignorarne alcune in favore di altre, il che conduce a conclusioni completamente diverse. Un altro punto delicato riguarda le enormi differenze tra i diversi tipi di blockchain: sebbene l'affermarsi di questa tecnologia porti indubbiamente *innovazione*, non sempre si può parlare di *rivoluzione* vera e propria. Per comprenderne benefici e limitazioni in maniera equilibrata, è necessario distinguere almeno le due grandi famiglie di blockchain: *permissionless* e *permissioned*.

3.2.1 Blockchain permissioned

Rappresentano l'innovazione più mite. Il controllo operato sui partecipanti al sistema permette di ridurre molti aspetti negativi della tecnologia, ma ne limita certamente il potenziale. In particolare, si può pensare a questi sistemi come a database distribuiti con l'aggiunta di alcune desiderabili proprietà. Tutti i punti riportati di seguito sono applicabili anche ai sistemi permissionless e pertanto si possono considerare come caratteristiche "generali" di Blockchain.

Vantaggi:

- **Semplificazione del paradigma attuale**: al momento è possibile ottenere gran parte dei vantaggi con altri sistemi, usare blockchain serve a semplificare e uniformare in un'unica architettura coerente il tutto;
- **Trasparenza**: ogni peer può controllare ogni transazione all'occorrenza, ideale negli scenari in cui sia necessario verificare il rispetto di norme condivise;
- **Condivisibilità**: la natura distribuita del sistema porta con se come diretta conseguenza la condivisione dei suoi contenuti, pur mantenendone pieno controllo sulle modifiche;
- **Immutabilità**: estrema difficoltà di modificare dati scritti in precendenza (impossibilità di fatto). Nel caso particolare dei sistemi permissioned, bisogna fare attenzione a non minare la sicurezza del sistema generale dando per scontata l'affidabilità degli attori;
- **Sicurezza**: tutte le transazioni in una blockchain sono crittograficamente protette e contribuiscono ad assicurare l'integrità del sistema;
- **Resilienza**: basarsi su una rete peer-to-peer permette di garantire la disponibilità del servizio anche nel caso un certo numero di peer non fosse disponibile.

Le policy di consenso influenzano anche questo punto molto pesantemente: se stabilisco che una transazione debba ricevere il consenso da tutti gli altri peer, ne basta uno non funzionante e il sistema risulta non disponibile;

Svantaggi:

- **Inefficienza rispetto ai database tradizionali**: la natura ridondante e cifrata di Blockchain mal si concilia con le necessità di alta velocità in lettura/scrittura;

- **Scalabilità**: la necessità di mantenere tracciata ogni modifica effettuata al registro distribuito provoca l'enorme aumento nel tempo della dimensione del database. Nel caso di sistemi trusted sarebbe possibile effettuare un *pruning* ovvero ridurre periodicamente la dimensione del database considerando affidabile la situazione in un certo momento ed eliminando la storia delle transazioni relative ai dati precedenti. Questo procedimento andrebbe però a minare la condizione di sicurezza e tracciabilità totale dei sistemi Blockchain, rendendoli violabili;

- **Complessità e carenza di know-how**: allestire un sistema Blockchain richiede competenze in reti e sistemi distribuiti, gestione dei database, crittografia a livelli sufficienti da poter garantirne la sicurezza di insieme. L'argomento è naturalmente complesso e più delicato rispetto ad un'architettura centralizzata, il che rende difficoltoso trovare personale sufficientemente esperto o formarlo allo scopo;

- **Tecnologia relativamente immatura**: anche i framework più stabili presenti al momento della stesura di questo documento (e.g. Hyperledger sez. 4.1.1) presentano variazioni più che considerevoli nell'arco di poche settimane. Uno scenario di questo tipo è estremamente ostile per un'applicazione in ambito enterprise della tecnologia, che dovrà attendere che le soluzioni attuali raggiungano una sufficiente stabilità prima di basare su di esse un'infrastruttura. Inoltre, una tecnologia tanto giovane non può essere di provata e testata affidabilità quanto i ben più rodati sistemi centralizzati: è necessario un certo periodo di sperimentazione prima che si affermi come effettiva concorrente.

3.2.2 Blockchain pubbliche

Rappresentano l'aspetto rivoluzionario di Blockchain. L'idea originaria di Nakamoto ha permesso per la prima volta di generare un'entità digitale *scarsa*, ovvero non arbitrariamente replicabile, senza la necessità della supervisione di un intermediario centrale. Inoltre, il sistema proposto da Nakamoto deve la sua stabilità all'impossibilità di fatto di alterare il registro condiviso: questo lo rende ideale nel contesto delle pratiche di notariato, che possono affidarsi ad una rete per la validazione prescindendo da un certificatore apposito. Una naturale applicazione della tecnologia si può individuare nella gestione delle proprietà (che possono diventare *smart properties*) o nel timestamping di documenti. In questa tipologia si trovano i vantaggi più dirompenti, assieme ai limiti più vincolanti.

Vantaggi:

- **No single point of failure**: la replicazione in ogni nodo e il sistema di consenso totalmente distribuito permettono il corretto funzionamento della rete anche nel caso in cui uno o più nodi non fossero raggiungibili, o presentassero comportamenti anomali;
- **Non censurabilità o revocabilità del servizio**: il sistema è indipendente dal controllo di un intermediario centrale, pertanto non esiste un partecipante alla rete che abbia la possibilità di negare arbitrariamente l'accesso al servizio ad un altro elemento.
- **Velocità di validazione**: prescindere da intermediari può portare ad un'accorciamento dei tempi necessari allo svolgimento di operazioni come quelle finanziarie. Paragonata ad altri sistemi informatici un'infrastruttura di questo tipo è sicuramente più lenta, ma se si considerano i tempi attualmente necessari a seguire le procedure legali in caso sia necessario validare e approvare documenti si nota come Blockchain permetta di effettuare operazioni verificabili in maniera estremamente veloce.
- **Risparmio sulle transazioni** analogamente a quanto detto per la velocità di validazione, eliminare la necessità di controllo da parte di terzi permette una riduzione considerevole dei costi per ogni procedura;

Svantaggi:

- **Scalabilità**: in questo caso la situazione è estremamente più drastica rispetto a quella dei sistemi permissioned. Non è infatti possibile stabilire un momento in cui si considera il sistema affidabile, e non esiste nemmeno un ente o un supervisore che possa essere investito di tale incarico. L'immagine riporta la situazione della blockchain di Bitcoin aggiornata al 05/11/2017, e lascia intuire come questo aspetto rischi facilmente di andare fuori controllo e risultare estremamente limitante per il sistema.

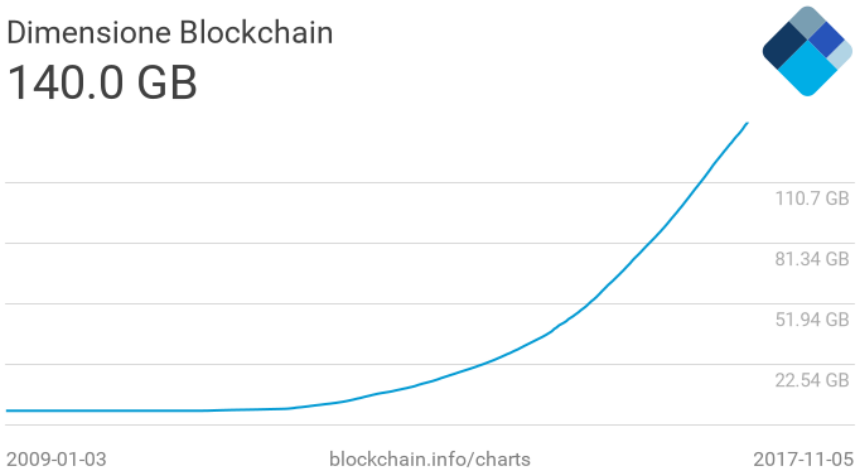

Figura 3.1: Dimensione della blockchain di Bitcoin

- **Adattabilità**: al momento la sola implementazione largamente diffusa della tecnologia la si trova in Bitcoin. Sebbene le sperimentazioni a riguardo siano innumerevoli, la necessità di fornire una ricompensa appetibile ha finora vincolato il successo delle proposte all'uso di tokenized blockchain. Nonostante queste si adattino straordinariamente bene all'ambito economico o alle operazioni di notariato, finora una flessibilità come quella della piattaforma Hyperledger rimane prerogativa dei sistemi non trustless;

- **Regolamentazione**: la rapida ascesa di Blockchain non è seguita altrettanto rapidamente da una regolamentazione efficace dal punto di vista normativo. Tale regolamentazione si presenta come particolarmente problematica per via della natura libera, indipendente e incensurabile dei sistemi in questione;

- **Privacy**: diffondere copie dei registri attraverso la rete pone problemi non indifferenti per la riservatezza delle informazioni, su diversi livelli. Un primo aspetto, proprio di Bitcoin, è la pseudonimità del protocollo. Gli indirizzi con cui si opera nella rete sono infatti pubblici, ed è possibile risalire alla completa storia delle transazioni di un indirizzo. Esistono strategie più o meno efficaci per offuscare questo genere di informazioni, come sfruttare più indirizzi generati algoritmicamente invece di utilizzarne solamente uno o usare servizi come CoinJoin che permettono di accorpare transazioni di più utenti in maniera da rendere difficilmente tracciabile il flusso singolo di pagamento. Tuttavia, nel momento in cui si riuscisse ad associare l'indirizzo all'ente a cui è legato (impresa tutt'altro che proibitiva, dal momento che l'ente stesso deve diffondere l'indirizzo per utilizzarlo) la riservatezza delle transazioni sarebbe completamente compromessa. Un altro punto delicato riguarda la presenza di dati on-chain: anche se cifrati, questi sono infatti bloccati dall'immutabilità della blockchain. Ne consegue direttamente che un attacco a forza bruta su questi dati sarebbe facilitato di molto dall'impossibilità di modificare la chiave di cifratura dopo un certo periodo di tempo, vincolando la privacy su tali dati ad una inevitabile "data di scadenza";

- **Impossibilità di interventi correttivi**: è un aspetto fortemente legato alla regolamentazione dei sistemi Blockchain. L'immutabilità del registro, così come la completa assenza di governance da parte di un ente verificatore, comporta il fatto che una volta immessi dati o stabiliti dei contrati questi siano completamente fuori dal controllo di chiunque. È celebre l'incidente avvenuto con "The DAO" sulla piattaforma Ethereum [23], in cui uno smart contract mal formulato ha portato ad un attacco per un valore di circa 50 milioni di dollari. È estremamente difficile proteggere il sistema da evenienze di questo tipo senza rompere le fondamenta stesse su cui si basa: la protezione dell'utente in tali scenari è responsabilità esclusiva dell'utente stesso. Un esempio analogo lo si trova in Bitcoin, dove nel caso di perdita della chiave privata associata al proprio portafoglio si perde irrevocabilmente l'accesso ad esso e, dualmente, un malintenzionato che entrasse in possesso di un portafoglio ne potrebbe disporre a sua assoluta discrezione;

- **Rischio di Lock-in**: la resistenza alle modifiche dei sistemi basati su Blockchain comporta delle situazioni indesiderabili per la sua adozione in produzione

a livello aziendale. Un esempio è l'elevata dipendenza dalla specifica tecnologia adottata: la migrazione dei dati tra diverse blockchain è molto difficoltosa e, ricordando che ciò che garantisce la correttezza dei dati è la loro completa tracciabilità, interrompere la catena migrando da un sistema all'altro avrebbe gli stessi effetti descritti parlando del *pruning* (sez. 3.2.1) rendendo violabile il sistema. Una soluzione possibile sarebbe permettere alla catena in cui si migra di "puntare" alla catena precedente, tuttavia ciò frammenterebbe le garanzie di affidabilità del sistema e un fallimento di un singolo frammento di catena la comprometterebbe nella sua interezza;

- **Protezione dei wallet**: i problemi di scalabilità a cui si fa riferimento precedentemente comportano che l'utente del sistema si carichi di un onere considerevole in termini di memoria e risorse computazionali. Una soluzione spesso adottata è quella di affidarsi a wallet esterni: si tratta di nodi completi, gestiti da un intermediario, che espongono le funzionalità di base agli utenti esterni. Questo si rivela un enorme rischio per la sicurezza in quanto ogni client deve necessariamente riporre un forte trust nei confronti del wallet, rendendo spesso completamente superfluo l'impiego di un sistema Blockchain. Inoltre, nell'analizzare la sicurezza generale del sistema bisogna considerare che il punto più fragile potrebbe essere proprio questo intermediario. È necessario perciò disincentivare quanto più possibile tale pratica, preoccupandosi di fornire agli utenti informazioni chiare a riguardo e dotandoli di un sistema non eccessivamente oneroso da sostenere;

- **Sybil attack**: un possibile attacco ad un sistema blockchain vede un eventuale attaccante tentare di saturare il network con nodi malevoli, in modo che sia altamente probabile che un utente si connetta solo a peer compromessi. Questa situazione può essere sfruttata in diversi modi, tra cui i seguenti:

  - L'attaccante può bloccare la propagazione delle informazioni da parte del resto della rete, isolando i nodi che dipendono da quelli compromessi;
  - L'attaccante può inoltrare solo blocchi da lui creati, creando un fork artificiale della blockchain non verificabile da parte dei nodi isolati e aprendo alla possibilità di double-spending;
  - Anche senza passare per il punto illustrato al passo precedente, se il client fa affidamento su transazioni senza richiedere più livelli di "profondità" del blocco per ritenerle confermate è vulnerabile a double-spending;
  - Sistemi di anonimizzazione a bassa latenza come quelli utilizzati tipicamente dai client Bitcoin (e.g. Tor) possono essere rotti facilmente con dei *timing attack* qualora l'attaccante avesse un controllo sulla rete come quello descritto;

- **Majority attack**: qualora un attaccante avesse il controllo della maggior parte del "potere di consenso" nella rete (potenza di calcolo nel caso di proof-of-work, valuta nel caso di proof-of-stake, etc.) potrebbe effettuare double spending creando un fork della blockchain. In seguito potrebbe far crescere la catena compromessa a velocità maggiore di quella ufficiale fino a renderla la catena più lunga, e quindi quella accettata dalla rete;

- **Flood attack**: un attaccante potrebbe inviare ripetutamente transazioni a se stesso, saturando la capienza massima del blocco e impedendo alle altre transazioni di essere elaborate. Bitcoin contrasta attacchi di questo tipo limitando le transazioni gratuite per blocco a 50KB, per cui tale attaccante esaurito lo spazio gratuito dovrebbe pagare una commissione che renderebbe l'attacco costoso, tuttavia in un sistema generico che volesse essere completamente gratuito questa potrebbe essere una vulnerabilità presente;

- **Timejacking attack** [9]: questo tipo di vulnerabilità è peculiare di Bitcoin e dipende da alcune sue caratteristiche specifiche, tuttavia è citata per far comprendere quanto delicati siano gli algoritmi sottostanti ai sistemi blockchain. Ogni nodo del sistema mantiene un contatore interno che rappresenta il *tempo di rete*, necessario per determinare fattori come la validità dei blocchi o il tempo medio di validazione necessario al calcolo della "difficoltà" del blocco successivo. Un attaccante potrebbe velocizzare o rallentare questo contatore attraverso ripetute connessioni al nodo comunicando timestamp alterati. Grazie alla discrepanza su questo contatore e al fatto che Bitcoin scarta automaticamente blocchi con timestamp che diverge troppo da quello interno, è possibile provocare un fork della catena e aprire a possibilità di double spending, come spiegato in dettaglio nell'articolo in nota.

Capitolo 4

Proposta di use case

Alla luce di quanto scritto finora, una possibile applicazione della tecnologia si può individuare nell'e-voting. Nei sistemi di voto attualmente in uso, la correttezza del voto e l'anonimato del votante sono totalmente affidati al controllo di un ente o di un sistema centrale in cui si ha fiducia come può essere lo stato o l'organo che indice la votazione. Questo vincola gli eleggibili e gli elettori ad una mancanza di trasparenza finora ritenuta inevitabile, e il sistema elettorale a dover investire risorse considerevoli nel gestire la votazione a partire da quando viene indetta fino al termine degli scrutini. L'avvento della tecnologia Blockchain permette un cambio di paradigma in favore di sistemi decentralizzati e trasparenti in cui non sia necessario riporre fiducia nell'operato di alcuno degli agenti coinvolti. Si propone quindi un modello che permetta di applicare queste caratteristiche desiderabili ad una votazione elettronica.

4.1 Tecnologie e algoritmi utilizzati

4.1.1 Hyperledger Fabric v.1.0

Hyperledger è un progetto collaborativo della Linux Foundation che mira a creare un framework open source per lo sviluppo di sistemi Blockchain in ambito aziendale. Fabric è uno dei sottoprogetti che fanno parte di Hyperledger, in particolare rappresenta il principale contributo di IBM alla causa. Nella visione di Hyperledger Fabric un sistema Blockchain aziendale deve essere basato su di un'architettura modulare in cui i diversi elementi, come algoritmi di consenso, sistema di verifica delle identità, protocolli di cifratura o tipo di database ausiliari, siano intercambiabili e sostituibili liberamente. Permette anche lo sviluppo di smart contract, qui chiamati chaincode, attraverso l'uso di un qualsiasi linguaggio di programmazione: caratteristica che lo rende estremamente più flessibile del limitato set di istruzioni previsto da Bitcoin o dal linguaggio dedicato sviluppato da Ethereum. Per fare ciò l'esecuzione dei chaincode è isolata in un container dedicato, creato attraverso Docker, contenente un sistema operativo sicuro, il linguaggio del chaincode e gli SDK che permettono di interfacciarsi con il sistema. Fabric è un sistema permissioned, tutti i partecipanti sono tenuti ad autenticarsi attraverso un servizio dedicato (anche questo intercambiabile) per avere accesso alla rete.

Componenti di Fabric

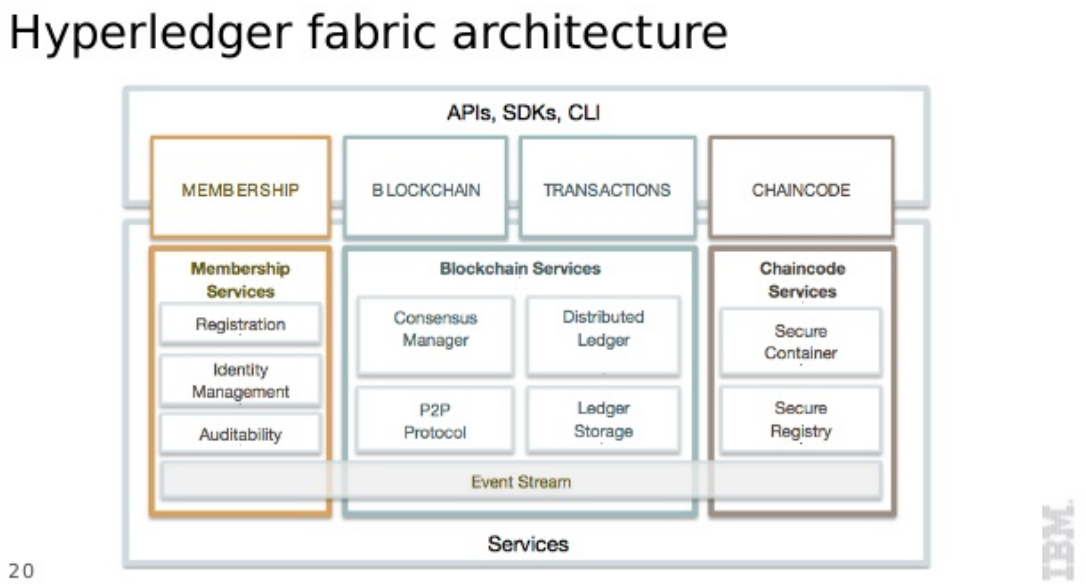

Figura 4.1: L'architettura di hyperledger come descritta nel whitepaper

- **Client**: sono gli enti partecipanti alla rete che invocano un chaincode o propongono una transazione. Rappresentano gli utenti finali, non mantengono una copia della blockchain ma devono connettersi con un peer per interagire con il sistema;
- **Peer**: sono gli enti che effettuano le transazioni e mantengono il database condiviso e la blockchain. I peer sono coordinati dall'orderer secondo delle policy specificate a priori;
- **Orderer**: è l'ente (eventualmente distribuito) che sovrintende alla comunicazione tra i nodi della rete. Fornisce garanzie di consegna dei messaggi immessi nella rete, tra cui le proposte di transazione da parte dei peer. È il responsabile del consenso all'interno del sistema distribuito;
- **Membership services**: sono l'astrazione di tutti i meccanismi crittografici e i protocolli che lavorano alla validazione dei certificati e al riconoscimento degli utenti;
- **Organizzazioni**: rappresentano ciascuna entità distinta che possiede un certificato univoco per la rete. I nodi come i peer o i client devono essere legati ad un'organizzazione per poter essere riconosciuti e autenticati;
- **Channel**: è l'overlay della blockchain privata di Hyperledger. Ciascun canale ha un registro specifico, condiviso tra i peer registrati al canale, e ciascun ente coinvolto nella transazione deve essere autenticato ad uno specifico channel affinché questa abbia luogo;

- **State**: abbreviazione per "State Database", è il database in cui è salvato lo stato attuale del sistema per effettuare query sul registro in maniera efficiente. È modellato come un database chiave/valore con versionamento. Può essere completamente ricostruito dalla storia delle transazioni salvata in blockchain, il che ne garantisce il controllo sull'affidabilità;

- **Chaincode**: è il nome dato ai programmi salvati nella blockchain di Hyperledger. Sono invocati dalle transazioni, e possono inizializzare componenti nello state o modificarli. La loro esecuzione avviene in un container Docker isolato per garantirne la riservatezza. Incapsula la componente di business logic del sistema, ha corrispondenza diretta con quelli che in altri sistemi basati su Blockchain vengono chiamati *smart contract*.

- **Transazioni**: sono la rappresentazione di ogni azione effettuata sul registro. Possono essere *deploy transactions* che creano nuovo chaincode e lo "installano" in un canale oppure *invoke transactions* che invocano una funzione presente in un chaincode installato per effettuare letture o modifiche al ledger. Ciascuna transazione deve essere approvata e gestita dall'orderer, e viene registrata nella blockchain per la futura tracciabilità.

4.1.2 Ring signature

Una ring signature, o firma ad anello, è un tipo di firma digitale legata ad un gruppo di utenti ciascuno in possesso di una chiave. Questo tipo di firma permette di verificare che l'autore appartiene a quel dato gruppo di utenti rendendo però computazionalmente infattibile determinare quale specifico utente esso sia. Riporto di seguito (tradotta) una possibile definizione formale [25]:
Si supponga che un gruppo di entità possieda le coppie di chiavi pubbliche/private $(P_1, S_1), (P_2, S_2), ..., (P_n, S_n)$. Il soggetto i può apporre una ring signature σ sul messaggio m avendo in input $(m, S_i, P_1, ..., P_n)$. Chiunque può verificare la validità della firma dati σ, m e le chiavi pubbliche coinvolte $P_1, ..., P_n$. Una *Linkable* ring

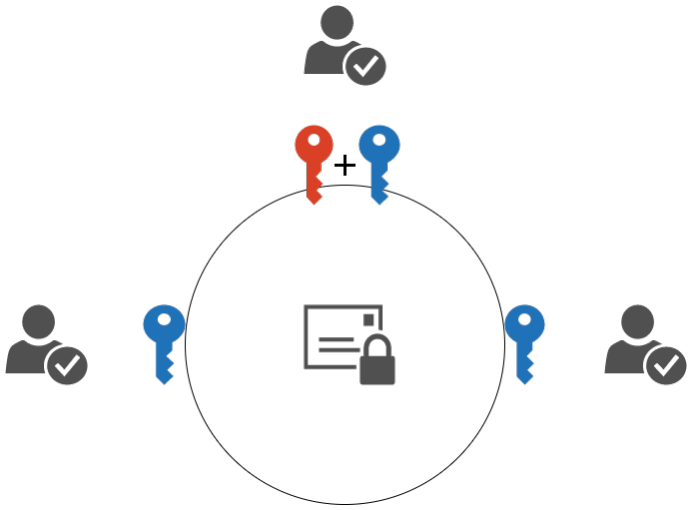

Figura 4.2: Ring signature

signature (LRS) permette inoltre di determinare se due ring signature sono state

computate dallo stesso membro del gruppo, pur senza poter risalire alla sua identità. La maggior parte degli algoritmi per la costruzione di Ring Signature ha complessità $O(n)$, anche se ne esistono di più efficienti come quello proposto da Tsang e Wei in *Short Linkable Ring Signatures for E-voting, E-cash and Attestation* [20], basato su un lavoro di Dodis del 2004.

4.2 Architettura

Il sistema si articola in due blockchain separate, la blockchain di firma e la blockchain di voto (da ora rispettivamente Signature chain o SC e Vote chain o VC). Le due chain sono isolate, per garantire la non collegabilità del voto al votante in fase di scrutinio. Le due chain hanno scopi diversi: SC assicura che il voto avvenga in maniera controllata e abilita al voto il client, inoltre contiene le informazioni per dividere in raggruppamenti i votanti similmente a quanto avviene con le sezioni elettorali; VC raccoglie i voti e li registra per effettuare poi il conteggio, ma deve garantire l'anonimato del votante.

4.2.1 Preparazione

Viene indetta una votazione. Si prevede che l'architettura sia già predisposta, in particolare devono essere noti ed affidabili, completi di chiavi pubbliche:

- L'elenco dei votanti;
- L'elenco dei peer;
- L'elenco degli indirizzi dei candidati per la Vote chain;
- L'elenco dei programmi client verificati.

Inizializzazione della Signature chain

La SC viene predisposta con il chaincode che permette al client di interrogare la chain per conoscere le chiavi pubbliche dei votanti inseriti nello stesso raggruppamento, registrando contemporaneamente la propria firma. I dati sui raggruppamenti dei votanti sono inseriti nella SC. Questa dovrà essere accessibile solamente ai peer deputati al controllo della votazione, a causa della riservatezza delle informazioni sui raggruppamenti.

Inizializzazione della Vote chain

La VC contiene inizialmente delle transazioni che registrano l'associazione di ogni indirizzo numerico ad un nome intelligibile del candidato corrispondente, i saldi dei voti dei candidati inizializzati a 0 e il chaincode necessario al client per incrementare di 1 il saldo del candidato scelto. Un altro chaincode utile è quello che permette ai peer di ricevere il conteggio finale e quindi l'esito della votazione, così come il chaincode che permette ad un servizio web di interrogare la blockchain e rendere pubblici i dati in essa contenuti per permettere al votante la verifica del voto espresso.

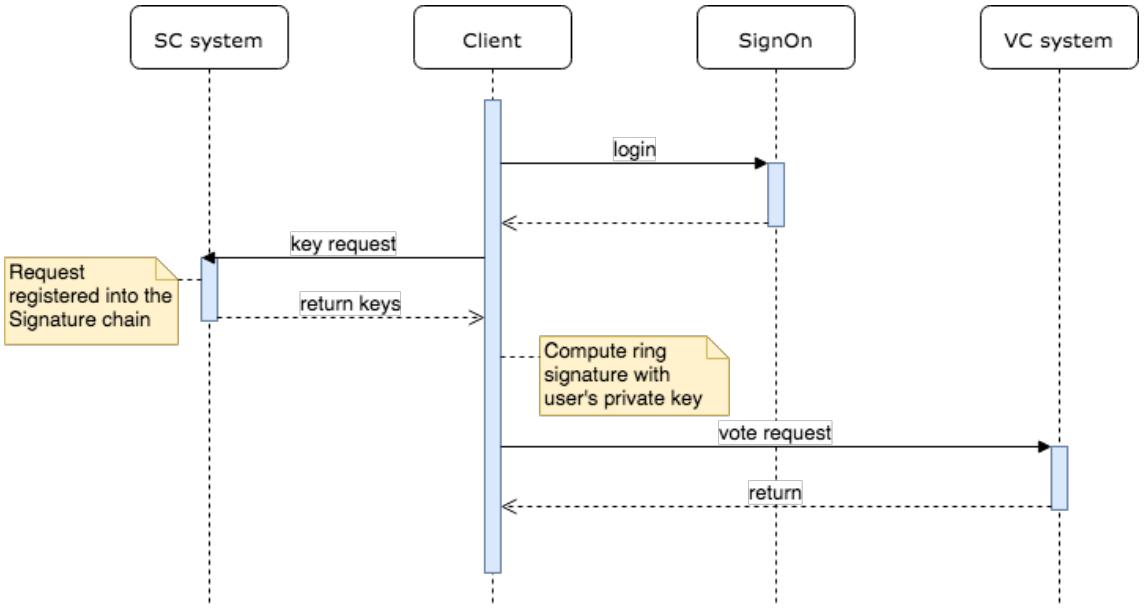

Figura 4.3: Workflow della votazione

4.2.2 Workflow di una votazione

Il diagramma in figura 4.3 illustra il workflow tipico di una votazione.

Signature chain: Firma del registro votanti

Il client autentica l'utente attraverso un sign-on affidabile. I controlli di accesso possono essere rinforzati o meno da altri passaggi affidati a sistemi di intelligenza artificiale o al controllo di un operatore. A utente autenticato, il client invoca il chaincode di query su SC ed ottiene l'elenco delle chiavi pubbliche del suo raggruppamento. Questa richiesta viene registrata in blockchain, per assicurarsi che lo stesso utente non possa ripetere la procedura di voto due volte.

(a) Firma tradizionale (b) Firma di SC

Figura 4.4: Firma del registro votanti

Vote chain: In cabina elettorale

Il client si trova ora in possesso delle chiavi pubbliche della sua sezione. In locale permette al votante di esprimere la propria preferenza di voto e richiede la transazione corrispondente nella VC firmandola con la ring signature del proprio raggruppamento. La votazione è ora conclusa.

(a) Voto tradizionale (b) Voto in VC

Figura 4.5: Votazione

Scrutinio e verifica

Le operazioni di scrutinio e verifica possono ora essere completamente automatizzate, facendo riferimento ad una base di dati sicura e virtualmente inalterabile come la blockchain.

4.3 Analisi

In questa sezione si analizzano le scelte architetturali fatte argomentandole, concludendo con delle possibilità di miglioramento che si potranno presentare con l'avanzamento dello scenario tecnologico attuale.

4.3.1 Qualità del voto

Il voto è definito dall'art. 48 [15] della Costituzione Italiana come "personale ed eguale, libero e segreto".

Personalità

La personalità del voto, intesa come la necessità di esercitarlo di persona e non tramite terzi, è delegata in questa proposta al servizio di sign-on. Al momento attuale, è possibile fare affidamento su diversi sistemi di autenticazione: per il corretto funzionamento del sistema è previsto che ogni utente abbia una coppia di chiavi pubblica e privata distribuita precedentemente, sotto verifica di un operatore. Ad esempio, è possibile distribuire queste chiavi contestualmente alla consegna della tessera elettorale. La natura certificata di queste chiavi le rende ideali come metodo di autenticazione ma è possibile aggiungere ulteriori controlli come una verifica tramite dati biometrici.

Eguaglianza

L'eguaglianza del voto consiste nell'avere ogni voto lo stesso valore di tutti gli altri. Si articola nell'impedire, in particolare, il voto plurimo e il voto multiplo. Nel sistema proposto, il voto plurimo è scongiurato dal codice del chaincode della Vote chain, il quale permette solo incrementi unitari al saldo dei candidati. Uno scenario di voto multiplo è invece controllato dalla Signature chain e dalla struttura del client: è infatti indispensabile che la procedura di voto sia strettamente successiva alla procedura di firma in SC, e non invocabile in maniera indipendente. In tal modo è possibile garantire che una stessa persona non reiteri l'esecuzione del chaincode di voto. Pur ammettendo il caso in cui un attaccante riesca a violare il client, questo potrà reiterare voti firmati dalla stessa ring signature: un controllo automatico può

facilmente accorgersi della non coerenza del numero di votanti nel raggruppamento e il numero di voti espressi dal raggruppamento, invalidando di fatto le transazioni ma mantenendo confinato l'evento disastroso: si possono applicare le procedure invocate attualmente qualora venissero riscontrate irregolarità che compromettano la validità dei voti di una sezione elettorale.

Libertà

La libertà del voto viene garantita dal negare la possibilità di voto qualora il votante fosse stato costretto con mezzi illeciti a esprimere una certa preferenza. Nel sistema di voto attuale è garantita dal controllo degli operatori di seggio. Nei sistemi di voto remoto l'argomento risulta spinoso: è infatti estremamente difficile stabilire una situazione di particolare pressione dell'elettore, o l'interferenza di esterni. Nel caso di voto elettronico, forzare il client ufficiale a poter essere installato solo su dispositivi muniti di telecamera frontale potrebbe risolvere il problema. Il controllo può essere svolto da appositi addetti eventualmente aiutati da un'intelligenza artificiale simile a quella utilizzata da Unilever [22] per i colloqui di lavoro. Questo non scongiura ogni possibile manipolazione, ma va evidenziato che il controllo sul sistema proposto non lo rende meno sicuro del sistema già in essere: permette anzi un livello di garanzia estremamente più alto di quello dei sistemi di voto per corrispondenza, già accettati in Italia in caso di votanti dall'estero e adottato in maniera diffusa in Svizzera.

Segretezza

La segretezza del voto nel sistema proposto ha come fulcro la completa separazione delle due blockchain. Affidando al client il compito di mantenere la continuità dell'operazione di voto, l'accoppiamento tra votante e voto espresso non può essere ricostruito con certezza in alcun modo. L'unica informazione che può trapelare riguarda il momento in cui viene registrata la votazione nelle due chain, tuttavia questa informazione è molto meno affidabile di quanto possa sembrare a causa della consistenza eventuale della blockchain. Sebbene qualche informazione trapeli, l'elevato numero di transazioni contemporanee previsto e l'incertezza nel determinare l'esatto momento di voto rende di fatto inservibili queste informazioni.

4.3.2 Necessità dei raggruppamenti

La suddivisione in sezioni elettorali è assolutamente necessaria nel sistema di voto attualmente adottato a causa di problemi logistici: è impensabile infatti raggruppare tutte le schede e svolgere un'unica operazione di scrutinio, soprattutto considerando il livello di sorveglianza che sarebbe richiesto in ogni momento dell'operazione. In un sistema automatico come quello proposto, esente da queste problematiche, sono stati introdotti raggruppamenti di elettori per differenti ragioni.
In primo luogo, la complessità computazionale degli algoritmi più noti e affermati per la ring signature è lineare: questo preclude l'uso significativo di questa tecnologia (centrale nel modello proposto) per insiemi di chiavi sufficientemente grandi. Anche implementando i più recenti algoritmi sublineari, la divisione in raggruppamenti permette di limitare ulteriormente il carico ai nodi del sistema, migliorandone l'efficienza e limitandone i costi in caso di affidamento a piattaforme come Ethereum, in cui il costo della transazione scala in base alla complessità del task da eseguire.

In secondo luogo, nel caso un attacco permettesse di replicare lo stesso voto più volte la divisione in raggruppamenti permette di contenere i danni al solo raggruppamento interessato.
È da sottolineare come, nel caso di un sistema elettronico, l'indipendenza dai problemi logistici dei sistemi tradizionali permetta di variare di volta in volta la composizione dei raggruppamenti, rendendo inutile qualsiasi informazione trapelata dalle votazioni precedenti. Questa è una condizione certamente più sicura di quella attuale, dove una semplice osservazione sul posto permette di identificare gli appartenenti a ciascuna sezione e far trapelare quindi informazioni (per quanto parziali) sulle preferenze di voto espresse: essendo i risultati delle votazioni per seggio pubblici, il trapelare di qualsiasi informazione sulla composizione del seggio è da considerarsi quantomeno indesiderato.

4.3.3 Prevedibili miglioramenti futuri al sistema

Al momento passaggi critici come l'autenticazione utente e il passaggio dalla SC alla VC devono essere svolti off-chain, ma future scoperte potrebbero aprire la strada verso implementazioni più complete. Portare l'autenticazione utente su Blockchain è un progetto che si lega con la creazione di un sistema di identità digitale decentralizzata. Immaginando un futuro (ad ora remoto) in cui siano disponibili documenti affidabili on-chain attraverso qualche tecnologia, l'autenticazione utente potrebbe svolgersi in questo modo guadagnando le caratteristiche di decentralità e sicurezza proprie della Blockchain.
Più vicino alla situazione attuale è un meccanismo che permetta di abilitare alla votazione su VC direttamente da SC, senza rinunciare all'anonimato del votante. Al momento, ogni chaincode di Hyperledger è eseguito in un ambiente completamente isolato per ragioni di sicurezza. Sono già previste successive implementazioni che permettano a diversi chaincode di interagire tra di loro. Questo aprirebbe ad una modifica del sistema di voto proposto: sarebbe possibile implementare su VC un sistema di token: questi token sarebbero generati quando viene indetta la votazione in base al numero di aventi diritto al voto, e versati dai chaincode verso dei "portafogli" temporanei da loro generati casualmente. Questi sarebbero poi passati al client già cifrati in maniera da mantenere chiunque, eccetto il software del chaincode, all'oscuro dell'accoppiamento votante-portafoglio. In questo modo sarebbe possibile prescindere dai raggruppamenti affidando la sicurezza del sistema unicamente alla solidità della Blockchain. L'aspetto negativo è che si perderebbe la capacità dei raggruppamenti di confinare un eventuale attaccante.

Capitolo 5

Scenario attuale e sfide future

DISCLAIMER

Questa sezione è un cantiere aperto. NON è definitiva in nessuna sua parte, NON nei contenuti, NON nella struttura, NON nel tono generale del discorso. Si tratta di una serie di appunti che sto prendendo e che saranno la base su cui scriverò poi la versione definitiva, li inserisco per mantenerne traccia e per avere ben chiara la direzione che sta intraprendendo il discorso.

In questo capitolo si vuole presentare il quadro attuale della situazione attorno ai sistemi blockchain. Nuove tecnologie sono sviluppate continuamente, perciò dopo una breve panoramica sui progetti più interessanti già in stadio avanzato di sviluppo si illustreranno le sfide e gli ambiti di ricerca più ferventi, fino ad arrivare ad elencare alcune delle applicazioni che potrebbero conseguire dai risultati di tali ricerche.

5.1 Innovazioni vicine

Al momento si assiste ad una crescita rapidissima di Bitcoin, tale da portare l'argomento al di fuori degli ambiti specialistici e farlo diventare un vero e proprio fenomeno di massa. Dietro a Bitcoin si sta sviluppando un'enorme quantità di ap-

Figura 5.1: Valore di mercato di bitcoin

plicazioni e sperimentazioni che riguardano Blockchain, alcune delle quali hanno

raggiunto ormai una certa stabilità. Molte di queste riguardano l'ambito economico e prendono il nome di *criptomonete*, altre invece si basano sulla blockchain per costruire sistemi più complessi.

5.1.1 Criptomonete

- **Ripple**: è il più famoso sistema di scambio di valuta basato su blockchain permissioned, adottato tra gli altri da MUFG, RBC, Santander, Unicredit, BBVA (https://ripple.com/);
- **Litecoin**: è una criptomoneta molto simile a Bitcoin da cui differisce per alcune caratteristiche chiave, su tutte il tempo necessario all'elaborazione di un blocco (2 minuti e mezzo contro i 10 minuti di Bitcoin) e il sistema di consenso. Litecoin infatti usa *scrypt* per la sua proof-of-work, una funzione gravosa sulla memoria piuttosto che sul processore che punta a evita il predominio delle server farm con hardware dedicato che controllano le operazioni di mining di Bitcoin;
- **Peercoin**: è una criptomoneta alternativa che adotta un algoritmo di consenso ibrido tra proof-of-work e proof-of-stake, allo scopo di portare un'alternativa solida all'enorme consumo energetico di Bitcoin;
- **Monero**: fork di Bitcoin che utilizza CryptoNote, un protocollo basato sulla Ring Signature orientato a rafforzare l'anonimato nella blockchain;
- **ZCash e ZCoin**: come Monero si pongono l'obiettivo di garantire la privacy in un sistema Blockchain, usano algoritmi di tipo zero-knowledge sebbene con piccole differenze tra di loro [13];

Un'altra applicazione interessante a livello economico è quella legata all'uso di criptomonete per il crowd-funding. Questa trova una semplice implementazione direttamente in Bitcoin attraverso l'uso di quello che si chiama *assurance contract*. Si tratta di una transazione con un singolo output rappresentante l'obiettivo del finanziamento, a cui si unisce poi ciascun volontario partecipante tramite hash di tipo *SIGHASH_ALL* e *SIGHASH_ANYONECANPAY*, che permettono di modificare gli input di transazione ma non gli output (si rimanda alla documentazione ufficiale per approfondimento). La transazione può essere registrata in blockchain, e quindi essere effettiva, solo quando il valore di output è coperto da corrispondente valore in input, ovvero al raggiungimento dell'obiettivo della campagna di crowd-funding.

5.1.2 Non-Criptomonete

- **Ethereum**: *DA ESPANDERE E PROMUOVERE A SEZIONE ASSIEME AD HYPERLEDGER* si tratta del principale sistema basato su Blockchain dopo Bitcoin. Il suo scopo è quello di permettere la creazione di applicazioni decentralizzate, che vengono eseguite come smart contract distribuiti tra i peer che partecipano alla rete. È il primo sistema blockchain ad aver introdotto un linguaggio Turing-completo (*Solidity*) e il concetto di macchina virtuale su blockchain, nel 2013. Si basa su una propria moneta virtuale che viene utilizzata per "alimentare" le applicazioni distribuite (in ambito Ethereum si

parla di *gas*, abbreviazione di *gasoline*). Questa si chiama *Ether*, e al momento si trova saldamente al secondo posto come market cap dietro a Bitcoin;

- **OpenTimestamps**: servizio che mira a diventare uno standard per il timestamping via blockchain. Concatena hash crittografici dei dati di cui gli utenti richiedono il timestamping e ne registra il risultato in una transazione bitcoin, rendendolo quindi pubblico e immutabile;

- **Namecoin**: servizio basato su blockchain che si propone di potenziare decentralizzazione, sicurezza, resistenza alla censura, privacy e velocità di componenti alcune dell'infrastruttura di Internet come i DNS;

- **Everledger**: una nicchia promettente riguarda i sistemi ad-hoc con utilizzo ben preciso. Ne è un esempio Everledger, che si propone di tracciare e gestire storia e possesso di beni di lusso come i diamanti; *Filament* per reti wireless sicure in IoT;

- **Filament**: la natura distribuita di Blockchain la rende ideale per implementazione di applicazioni IoT che richiedano un adeguato livello di sicurezza. Lavora in questo ambito Filament, startup nata recentemente, nel 2017, che si pone come obiettivo la gestione di reti wireless sicure in ambito IoT;

- **Blockchain a livello enterprise**: per la creazione di applicazioni aziendali basate su Blockchain è necessario appoggiarsi ad opportuni framework. Questi devono rispettare requisiti imprescindibili per l'ambito business, tra cui fornire opportuna documentazione, supporto tecnico e rendere agevoli le operazioni di testing, integrazione, controlli di sicurezza. Esempi di piattaforme simili in sviluppo sono Hyperledger, Bloq, Chain, sebbene ciascuno di questi soffra al momento di poca stabilità dovuta alla loro ancora recente nascita;

- **Hawk**: è un sistema di smart contract che affronta il problema della poca privacy nelle transazioni delle blockchain più diffuse. Hawk fornisce uno strumento per la generazione di un "protocollo crittografico efficiente in cui le controparti del contratto interagiscono con la blockchain usando primitive crittografiche come gli algoritmi *zero-knowledge proof*." [16];

5.2 Sfide e ambiti di ricerca

5.2.1 Efficienza e scalabilità

- Algoritmi di consenso

- Problemi crittografici nuovi

Il problema della scalabilità dei sistemi blockchain è molto dipendente dall'algoritmo di consenso implementato. ... PoW bello ma costoso in termini di risorse

5.2.2 Standardizzazione e interoperabilità

Blockchain non è ancora una tecnologia sufficientemente matura da permettere una piena ed agevole integrazione con i sistemi esistenti. Emettere degli standard adeguati aiuterà a migliorare l'integrazione dei sistemi blockchain tanto fra di loro quanto con l'infrastruttura circostante.
Dal punto di vista applicativo lo sviluppo di framework come Hyperledger aiuta a creare una base solida e delle sottostrutture chiare per la progettazione di reti distribuite interoperanti, e maggiore sarà il numero di utenti che abbracceranno soluzioni di questo tipo minori saranno le differenze critiche tra sistema e sistema permettendo adattamenti più agevoli. D'altra parte, è necessario che tali piattaforme si sviluppino e allarghino la gamma di scenari modellabili venendo incontro alle necessità di gruppi di utenti sempre più grandi ed eterogenei.
Dal punto di vista normativo invece si sono visti passi in avanti con la creazione nel 2016 del comitato tecnico ISO/TC 307 "Blockchain and distributed ledger technologies", che sta sviluppando il primo standard ISO ufficiale sull'argomento.

5.2.3 Privacy

Algoritmi zero-knowledge

- **Zero-Knowledge Password Proof**
  `https://en.wikipedia.org/wiki/Zero-knowledge_password_proof`
- **Piattaforma Enigma**
  `https://www.enigma.co/enigma_full.pdf`
- **zk-SNARK**
  `https://z.cash/technology/zksnarks.html`

Obfuscation

vedi pag 244 di Understanding Bitcoin

5.3 Cosa si potrà fare

- Reti IoT sicure basate su Blockchain condivise
- Condivisione dati sensibili (cartelle cliniche?)
- Digital Identity
- Immigrazione e controlli doganali
- Cronotachigrafi
- Gestione DRM (digital rights management)
- Digital assets e smart properties
- Moneta basata su Bitcoin per transazioni efficaci tra privati

Capitolo 6

Conclusioni

Appendice A

Implementazione

A.1 Costruzione della rete

```
#!/bin/sh
#
# Copyright IBM Corp All Rights Reserved
#
# SPDX-License-Identifier: Apache-2.0
#
export PATH=$GOPATH/src/github.com/hyperledger/fabric/build/bin:${PWD}/../bin:${PWD
    }:$PATH
export FABRIC_CFG_PATH=${PWD}
export CHANNEL_NAME=sigchannel

# remove previous crypto material and config transactions
rm -fr config/*
rm -fr crypto-config/*

# generate crypto material
cryptogen generate --config=./crypto-config.yaml
if [ "$?" -ne 0 ]; then
  echo "Failed␣to␣generate␣crypto␣material..."
  exit 1
fi

# generate genesis block for orderer
configtxgen -profile OneOrgOrdererGenesis -outputBlock ./config/genesis.block
if [ "$?" -ne 0 ]; then
  echo "Failed␣to␣generate␣orderer␣genesis␣block..."
  exit 1
fi

# generate channel configuration transaction
configtxgen -profile OneOrgChannel -outputCreateChannelTx ./config/sigchannel.tx -
    channelID $CHANNEL_NAME
if [ "$?" -ne 0 ]; then
  echo "Failed␣to␣generate␣channel␣configuration␣transaction..."
  exit 1
fi

# generate anchor peer transaction
configtxgen -profile OneOrgChannel -outputAnchorPeersUpdate ./config/
    StatoMSPanchors.tx -channelID $CHANNEL_NAME -asOrg StatoMSP
if [ "$?" -ne 0 ]; then
  echo "Failed␣to␣generate␣anchor␣peer␣update␣for␣StatoMSP..."
  exit 1
fi

#-------------------------------------------------------------------------
#
#   Start the network
#
#-------------------------------------------------------------------------
```

```
# Exit on first error, print all commands.
set -ev

# don't rewrite paths for Windows Git Bash users
export MSYS_NO_PATHCONV=1

docker-compose -f docker-compose.yml down

docker-compose -f docker-compose.yml up -d

# wait for Hyperledger Fabric to start
# incase of errors when running later commands, issue export FABRIC_START_TIMEOUT=<
    larger number>
export FABRIC_START_TIMEOUT=10
#echo ${FABRIC_START_TIMEOUT}
sleep ${FABRIC_START_TIMEOUT}

# Create the channel
docker exec -e "CORE_PEER_LOCALMSPID=StatoMSP" -e "CORE_PEER_MSPCONFIGPATH=/opt/
    gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/stato.
    eurasia.com/users/Admin@stato.eurasia.com/msp" -e "CORE_PEER_ADDRESS=peer0.
    stato.eurasia.com:7051" cli peer channel create -o orderer.eurasia.com:7050 -c
    sigchannel -f /opt/gopath/src/github.com/hyperledger/fabric/peer/config/
    sigchannel.tx
# Join peer0.stato.eurasia.com to the channel.
docker exec -e "CORE_PEER_LOCALMSPID=StatoMSP" -e "CORE_PEER_MSPCONFIGPATH=/opt/
    gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/stato.
    eurasia.com/users/Admin@stato.eurasia.com/msp" -e "CORE_PEER_ADDRESS=peer0.
    stato.eurasia.com:7051" cli peer channel join -b sigchannel.block
```

A.2 Configurazione crypto-tool

```
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#

# ---------------------------------------------------------------------------
# "OrdererOrgs" - Definition of organizations managing orderer nodes
# ---------------------------------------------------------------------------
OrdererOrgs:
  # ---------------------------------------------------------------------------
  # Orderer
  # ---------------------------------------------------------------------------
  - Name: Orderer
    Domain: eurasia.com
    # ---------------------------------------------------------------------------
    # "Specs" - See PeerOrgs below for complete description
    # ---------------------------------------------------------------------------
    Specs:
      - Hostname: orderer
# ---------------------------------------------------------------------------
# "PeerOrgs" - Definition of organizations managing peer nodes
# ---------------------------------------------------------------------------
PeerOrgs:
  # ---------------------------------------------------------------------------
  # Stato
  # ---------------------------------------------------------------------------
  - Name: Stato
    Domain: stato.eurasia.com
    # ---------------------------------------------------------------------------
    # "Specs"
    # ---------------------------------------------------------------------------
    # Uncomment this section to enable the explicit definition of hosts in your
    # configuration.  Most users will want to use Template, below
    #
    # Specs is an array of Spec entries.  Each Spec entry consists of two fields:
    #   - Hostname:   (Required) The desired hostname, sans the domain.
    #   - CommonName: (Optional) Specifies the template or explicit override for
```

```
    #                 the CN.  By default, this is the template:
    #
    #                              "{{.Hostname}}.{{.Domain}}"
    #
    #                 which obtains its values from the Spec.Hostname and
    #                 Org.Domain, respectively.
    # ---------------------------------------------------------------------------
    # Specs:
    #   - Hostname: foo # implicitly "foo.stato.eurasia.com"
    #     CommonName: foo27.org5.eurasia.com # overrides Hostname-based FQDN set
        above
    #   - Hostname: bar
    #   - Hostname: baz
    # ---------------------------------------------------------------------------
    # "Template"
    # ---------------------------------------------------------------------------
    # Allows for the definition of 1 or more hosts that are created sequentially
    # from a template. By default, this looks like "peer%d" from 0 to Count-1.
    # You may override the number of nodes (Count), the starting index (Start)
    # or the template used to construct the name (Hostname).
    #
    # Note: Template and Specs are not mutually exclusive.  You may define both
    # sections and the aggregate nodes will be created for you.  Take care with
    # name collisions
    # ---------------------------------------------------------------------------
    Template:
      Count: 1
      # Start: 5
      # Hostname: {{.Prefix}}{{.Index}} # default
    # ---------------------------------------------------------------------------
    # "Users"
    # ---------------------------------------------------------------------------
    # Count: The number of user accounts _in addition_ to Admin
    # ---------------------------------------------------------------------------
    Users:
      Count: 1
```

A.3 Configurazione configtxgen

```
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#

---
################################################################################
#
#   Profile
#
#   - Different configuration profiles may be encoded here to be specified
#   as parameters to the configtxgen tool
#
################################################################################
Profiles:

    OneOrgOrdererGenesis:
        Orderer:
            <<: *OrdererDefaults
            Organizations:
                - *OrdererOrg
        Consortiums:
            VotingConsortium:
                Organizations:
                    - *Stato
    OneOrgChannel:
        Consortium: VotingConsortium
        Application:
            <<: *ApplicationDefaults
            Organizations:
```

```
                - *Stato

################################################################################
#
#   Section: Organizations
#
#   - This section defines the different organizational identities which will
#   be referenced later in the configuration.
#
################################################################################
Organizations:

    # SampleOrg defines an MSP using the sampleconfig.  It should never be used
    # in production but may be used as a template for other definitions
    - &OrdererOrg
        # DefaultOrg defines the organization which is used in the sampleconfig
        # of the fabric.git development environment
        Name: OrdererOrg

        # ID to load the MSP definition as
        ID: OrdererMSP

        # MSPDir is the filesystem path which contains the MSP configuration
        MSPDir: crypto-config/ordererOrganizations/eurasia.com/msp

    - &Stato
        # DefaultOrg defines the organization which is used in the sampleconfig
        # of the fabric.git development environment
        Name: StatoMSP

        # ID to load the MSP definition as
        ID: StatoMSP

        MSPDir: crypto-config/peerOrganizations/stato.eurasia.com/msp

        AnchorPeers:
            # AnchorPeers defines the location of peers which can be used
            # for cross org gossip communication.  Note, this value is only
            # encoded in the genesis block in the Application section context
            - Host: peer0.stato.eurasia.com
              Port: 7051

################################################################################
#
#   SECTION: Orderer
#
#   - This section defines the values to encode into a config transaction or
#   genesis block for orderer related parameters
#
################################################################################
Orderer: &OrdererDefaults

    # Orderer Type: The orderer implementation to start
    # Available types are "solo" and "kafka"
    OrdererType: solo

    Addresses:
        - orderer.eurasia.com:7050

    # Batch Timeout: The amount of time to wait before creating a batch
    BatchTimeout: 2s

    # Batch Size: Controls the number of messages batched into a block
    BatchSize:

        # Max Message Count: The maximum number of messages to permit in a batch
        MaxMessageCount: 10

        # Absolute Max Bytes: The absolute maximum number of bytes allowed for
        # the serialized messages in a batch.
        AbsoluteMaxBytes: 99 MB

        # Preferred Max Bytes: The preferred maximum number of bytes allowed for
```

```
        # the serialized messages in a batch. A message larger than the preferred
        # max bytes will result in a batch larger than preferred max bytes.
        PreferredMaxBytes: 512 KB

    Kafka:
        # Brokers: A list of Kafka brokers to which the orderer connects
        # NOTE: Use IP:port notation
        Brokers:
            - 127.0.0.1:9092

    # Organizations is the list of orgs which are defined as participants on
    # the orderer side of the network
    Organizations:

################################################################################
#
#   SECTION: Application
#
#   - This section defines the values to encode into a config transaction or
#   genesis block for application related parameters
#
################################################################################
Application: &ApplicationDefaults

    # Organizations is the list of orgs which are defined as participants on
    # the application side of the network
    Organizations:
```

A.4 Compose

```
#
# Copyright IBM Corp All Rights Reserved
#
# SPDX-License-Identifier: Apache-2.0
#
version: '2'

networks:
  basic:

services:
  ca.eurasia.com:
    image: hyperledger/fabric-ca
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca.eurasia.com
    ports:
      - "7054:7054"
    command: sh -c 'fabric-ca-server start --ca.certfile /etc/hyperledger/fabric-ca
        -server-config/stato.eurasia.com-cert.pem --ca.keyfile /etc/hyperledger/
        fabric-ca-server-config/
        a22daf356b2aab5792ea53e35f66fccef1d7f1aa2b3a2b92dbfbf96a448ea26a_sk -b
        admin:adminpw -d'
    volumes:
      - ./crypto-config/peerOrganizations/stato.eurasia.com/ca/:/etc/hyperledger/
          fabric-ca-server-config
    container_name: ca.eurasia.com
    networks:
      - basic

  orderer.eurasia.com:
    container_name: orderer.eurasia.com
    image: hyperledger/fabric-orderer
    environment:
      - ORDERER_GENERAL_LOGLEVEL=debug
      - ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
      - ORDERER_GENERAL_GENESISMETHOD=file
      - ORDERER_GENERAL_GENESISFILE=/etc/hyperledger/configtx/genesis.block
      - ORDERER_GENERAL_LOCALMSPID=OrdererMSP
```

```
    - ORDERER_GENERAL_LOCALMSPDIR=/etc/hyperledger/msp/orderer/msp
  working_dir: /opt/gopath/src/github.com/hyperledger/fabric/orderer
  command: orderer
  ports:
    - 7050:7050
  volumes:
      - ./config/:/etc/hyperledger/configtx
      - ./crypto-config/ordererOrganizations/eurasia.com/orderers/orderer.eurasia
          .com/:/etc/hyperledger/msp/orderer
      - ./crypto-config/peerOrganizations/stato.eurasia.com/peers/peer0.stato.
          eurasia.com/:/etc/hyperledger/msp/peerStato
  networks:
    - basic

peer0.stato.eurasia.com:
  container_name: peer0.stato.eurasia.com
  image: hyperledger/fabric-peer
  environment:
    - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
    - CORE_PEER_ID=peer0.stato.eurasia.com
    - CORE_LOGGING_PEER=debug
    - CORE_CHAINCODE_LOGGING_LEVEL=DEBUG
    - CORE_PEER_LOCALMSPID=StatoMSP
    - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/msp/peer/
    - CORE_PEER_ADDRESS=peer0.stato.eurasia.com:7051
    # # the following setting starts chaincode containers on the same
    # # bridge network as the peers
    # # https://docs.docker.com/compose/networking/
    - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=${COMPOSE_PROJECT_NAME}_basic
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb:5984
    # The CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME and
        CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD
    # provide the credentials for ledger to connect to CouchDB.  The username and
        password must
    # match the username and password set for the associated CouchDB.
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  working_dir: /opt/gopath/src/github.com/hyperledger/fabric
  command: peer node start
  # command: peer node start --peer-chaincodedev=true
  ports:
    - 7051:7051
    - 7053:7053
  volumes:
      - /var/run/:/host/var/run/
      - ./crypto-config/peerOrganizations/stato.eurasia.com/peers/peer0.stato.
          eurasia.com/msp:/etc/hyperledger/msp/peer
      - ./crypto-config/peerOrganizations/stato.eurasia.com/users:/etc/
          hyperledger/msp/users
      - ./config:/etc/hyperledger/configtx
  depends_on:
    - orderer.eurasia.com
    - couchdb
  networks:
    - basic

couchdb:
  container_name: couchdb
  image: hyperledger/fabric-couchdb
  # Populate the COUCHDB_USER and COUCHDB_PASSWORD to set an admin user and
      password
  # for CouchDB.  This will prevent CouchDB from operating in an "Admin Party"
      mode.
  environment:
    - COUCHDB_USER=
    - COUCHDB_PASSWORD=
  ports:
    - 5984:5984
  networks:
    - basic

cli:
```

```
container_name: cli
image: hyperledger/fabric-tools
tty: true
environment:
  - GOPATH=/opt/gopath
  - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
  - CORE_LOGGING_LEVEL=DEBUG
  - CORE_PEER_ID=cli
  - CORE_PEER_ADDRESS=peer0.stato.eurasia.com:7051
  - CORE_PEER_LOCALMSPID=StatoMSP
  - CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/
      crypto/peerOrganizations/stato.eurasia.com/users/Admin@stato.eurasia.com/
      msp
  - CORE_CHAINCODE_KEEPALIVE=10
working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
command: /bin/bash
volumes:
    - /var/run/:/host/var/run/
    - ./../chaincode/:/opt/gopath/src/github.com/
    - ./crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto
        /
    - ./config:/opt/gopath/src/github.com/hyperledger/fabric/peer/config/
networks:
    - basic
#depends_on:
#  - orderer.eurasia.com
#  - peer0.stato.eurasia.com
#  - couchdb
```

Bibliografia

[1] Ferdinando M. Ametrano. Hayek money: the cryptocurrency price stability solution. Technical report, Milan Bicocca University, 2016. `https://papers.ssrn.com/abstract=2425270` [Online; acceduto 30/10/2017].

[2] Ferdinando M. Ametrano. Bitcoin and blockchain technology: An introduction, 2017. [Slides online, acceduto 02/11/2017].

[3] Saifedean Hisham Ammous. Blockchain technology: What is it good for? Technical report, Lebanese American University, 2016. `https://papers.ssrn.com/abstract=2832751` [Online; acceduto 02/11/2017].

[4] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timòn, and Pieter Wuille. Enabling blockchain innovations with pegged sidechains, 2014. [Online, acceduto 30/10/2017].

[5] Imran Bashir. *Mastering Blockchain*. Packt, 2017.

[6] BitcoinWiki. Hashcash, 2017. [Online, acceduto 08/11/2017].

[7] Vitalik Buterin. Casper version 1 implementation guide, 2017. [Online; acceduto 30/10/2017].

[8] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. Technical report, MIT, 1999. `http://pmg.csail.mit.edu/papers/osdi99.pdf` [Online; acceduto 23/10/2017].

[9] corbixgwelt. Timejacking & bitcoin, the global time agreement puzzle, 2011. [Online, acceduto 10/11/2017].

[10] Manoj Debnath. Getting started with mongodb as a java nosql solution. [Online, acceduto 07/11/2017].

[11] Pedro Franco. *Understanding Bitcoin*. Wiley, 2015.

[12] Seth Gilbert and Nancy A. Lynch. Perspectives on the cap theorem. Technical report, MIT, 2002.

[13] Poramin Insom. Zcoin and zcash: Similarities and differences, 2016. [Online, acceduto 09/11/2017].

[14] Rick Echevarria (Intel). 2017: A summer of consensus, 2017. [Online; acceduto 30/10/2017].

[15] Repubblica Italiana. Costituzione, art. 48, 1948.

[16] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. Technical report, University of Maryland and Cornell University, 2015. `https://eprint.iacr.org/2015/675.pdf` [Online; acceduto 10/11/2017].

[17] Leslie Lamport, Marshall Pease, and Robert Shostak. The byzantine generals problem. Technical report, ACM, 1982. `http://lamport.azurewebsites.net/pubs/pubs.html#byz` [Online; acceduto 23/10/2017].

[18] David Mazieres. The stellar consensus protocol: A federated model for internet-level consensus, 2016. [Online; acceduto 30/10/2017].

[19] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, The Cryptography Mailing List, 2008. `https://bitcoin.org/bitcoin.pdf` [Online; acceduto 07/11/2017].

[20] Patrick P.Tsang and Victor K. Wei. Short linkable ring signatures for e-voting, e-cash and attestation. Technical report, The Chinese University of Hong Kong, 2004. `https://eprint.iacr.org/2004/281.pdf` [Online; acceduto 20/11/2017].

[21] DJ Qian. The fourth industrial revolution: Blockchain tech and the integration of trust, 2017. [Online, acceduto 02/11/2017].

[22] Jon-Mark Sabel. How AI is transforming pre-hire assessments - Hirevue blog, 2017. [Online; acceduto 23/10/2017].

[23] David Siegel. Understanding the dao attack, 2016. [Online, acceduto 07/11/2017].

[24] Melanie Swan. *Blockchain, Blueprint for a New Economy.* O'Reilly, 2015.

[25] Wikipedia. Ring signature. [Online, acceduto 06/11/2017].