UNIVERSITÀ DEGLI STUDI DI PADOVA

SCUOLA DI INGEGNERIA
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

Studio della tecnologia Blockchain e sue possibili applicazioni in ambito sanitario

Laureando: *Davide Bellemo*

Relatore: *Prof. Mauro Migliardi*

11 Settembre 2018

Anno Accademico: 2017/18

A Martina e alla mia famiglia

Prefazione

La nostra epoca è caratterizzata da forti innovazioni tecnologiche, alcune delle quali rappresentano delle vere e proprie sfide a sistemi esistenti e consolidati. Degno di nota in questo senso è l'avvento della tecnologia blockchain che grazie ai suoi meccanismi ha permesso di introdurre il concetto di criptovaluta, una moneta digitale che può essere scambiata alla pari di quella cartacea. Il successo avuto dalla regina delle criptovalute, bitcoin, ha portato alla nascita di numerose altre monete digitali e ha favorito l'idea che la tecnologia blockchain potesse essere impiegata anche in ambiti diversi da quello monetario. Il sistema sanitario italiano, ad esempio, presenta carenze informatiche notevoli dal punto di vista della gestione delle informazioni sanitarie degli utenti e richiederebbe quindi uno sviluppo tecnologico in linea con le esigenze digitali odierne. L'implementazione di un sistema basato su blockchain potrebbe rappresentare una possibile soluzione al problema, anche se l'infrastruttura necessaria sarebbe notevolmente più complicata rispetto a quella di un sistema classico. Con questo lavoro si intende dare un quadro il più possibile dettagliato del funzionamento della blockchain e quindi delle principali criptovalute esistenti, mostrando i loro punti di forza e le loro debolezze. Nei capitoli successivi verranno discusse possibili applicazioni di tale tecnologia in ambito sanitario. In conclusione si proporrà un'analisi dei possibili vantaggi e svantaggi di un tale approccio.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Tecnologia Blockchain

Introduzione

In contabilità un libro mastro è un registro in cui vengono raccolti tutti i dati che interessano le transazioni di una certa azienda. La blockchain analogamente al libro mastro è una raccolta di operazioni, in gergo transazioni, che vengono effettuate tra gli utenti di una rete. Tecnicamente è un database distribuito in una rete di nodi peer to peer che ha delle caratteristiche che lo contraddistinguono da uno tradizionale. Come dice la parola stessa, la blockchain è una catena di blocchi concatenati tra loro, ciascuno dei quali è costituito da un insieme di transazioni. Il legame tra blocchi è di tipo crittografico, nel senso che ogni blocco contiene uno speciale riferimento a quello precedente. Tale riferimento non è altro che una stringa alfanumerica, detta hash, calcolata tramite una funzione matematica crittografica. Quindi ogni blocco contiene l'hash di quello precedente ad eccezione del blocco genesi, il primo mai creato. Anche all'interno delle transazioni sono presenti elementi crittografici, dato che devono comparire una o più firme digitali. Lo scopo di questa artificiosa architettura è rendere difficilmente modificabile la blockchain e perciò sicura. I dettagli implementativi verranno forniti nei paragrafi successivi. Ora passiamo a quello che è stato uno dei principali casi d'uso della blockchain: Bitcoin.

1.1 Il protocollo Bitcoin

1.1.1 Un po' di storia

Bitcoin è stato introdotto nel 2008 in seguito alla pubblicazione di un paper scientifico intitolato "Bitcoin: A Peer-to-Peer Electronic Cash System" [1]. La data ufficiale della nascita di Bitcoin invece è il 3 gennaio del 2009, giorno in cui viene creato il blocco genesi. Il suo inventore è Satoshi Nakamoto, pseudonimo di una o più persone il cui obbiettivo era quello di creare un sistema di pagamento online, sicuro e indipendente dalle autorità centrali quali possono essere le banche. Bitcoin quindi è una moneta digitale decentralizzata scambiata in una rete di nodi peer to peer, ovvero nodi paritari, che non formano gerarchie di tipo client-server ma agiscono al contempo da client e da server verso gli altri nodi terminali della rete. Una precisazione: bitcoin con la b minuscola è la moneta digitale, Bitcoin con la b maiuscola è il protocollo che la governa. La decentralizzazione è garantita dal fatto che la blockchain di Bitcoin è pubblica e quindi ogni partecipante della rete può consultarla e contribuire ad estenderla, di conseguenza non è appannaggio di un singolo ente ma è tenuta dalla totalità degli utenti; su questa affermazione in seguito si faranno delle precisazioni. Implementare un sistema di questo tipo significa scontrarsi con il problema del "double spending", della doppia spesa. Quando un utente fa una transazione ci deve essere la garanzia che i soldi appena spesi non possano essere utilizzati una seconda volta per compierne un'altra. La moneta fisica risolve la questione alla radice perché la stessa banconota cartacea non può essere fisicamente in due posti allo stesso tempo. Per quanto riguarda i pagamenti digitali, in un sistema di fiducia centralizzato il problema è gestito da una terza parte che fa controlli su ogni operazione effettuata dagli utenti. Satoshi ha affrontato il problema ideando un meccanismo che coinvolge crittografia e compartecipazione degli utenti e verrà trattato in seguito. Il valore economico assunto dal bitcoin è determinato dalla legge di mercato della domanda e offerta. Bitcoin è quotato su siti appositi chiamati Exchange. Questi siti permettono di scambiare Bitcoin con Euro, Dollaro americano o altre valute emesse dai governi, dette anche valute fiat. Il primo Exchange è andato online nel marzo del 2010 e quotava bitcoin a soli 0.003$. Il 22 maggio 2010 vengono acquistate due pizze in Florida per 10 mila bitcoin. Meno di un anno dopo la criptomoneta raggiunge il valore di 1$. Nel 2013 la valutazione subisce alti e bassi arrivando a toccare

un massimo di 1200$ per poi scendere di nuovo a 500$. Con l'apertura di nuovi exchange e grazie alla speculazioni da parte di un numero sempre maggiore di utenti, in particolare cinesi, il prezzo sale fino a 5000$ ad inizio settembre del 2017 e si attesta a 10000$ a novembre. Oggi (agosto 2018) ha un valore di 7500$ circa. Di seguito un grafico che ne rappresenta l'andamento nel corso degli anni. Verranno ora analizzati gli elementi di base costituenti la blockchain.

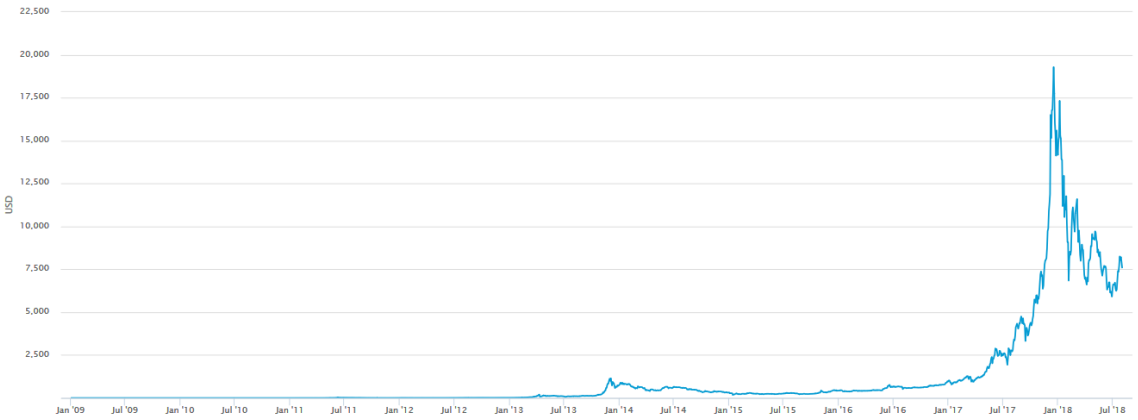

Figura 1.1: Andamento del prezzo di Bitcoin, fonte Blockchain.com

1.1.2 Transazioni

Una transazione non è altro che uno scambio di informazione tra utenti. Nel caso di Bitcoin l'informazione è un asset monetario che dopo una transazione subisce un cambio di proprietà da un utente inviante A ad un utente ricevente B. Le transazioni come detto precedentemente vanno a formare i blocchi della blockchain, il libro mastro che tiene traccia di tutti gli scambi avvenuti tra gli utenti, dal blocco genesi fino ad oggi. Un utente che intende inviare o ricevere bitcoin deve dotarsi di un wallet, cioè di un software che fornisce un indirizzo pubblico visibile a tutti o, meglio, una chiave pubblica, grazie alla quale è possibile ricevere moneta, e una chiave privata, conosciuta solo dal proprietario, con la quale invece la si può inviare. Supponiamo che Alice voglia spedire 1 BTC a Bob. Quest'ultimo fornirà il proprio indirizzo pubblico ad Alice, la quale firmerà un messaggio con la propria chiave privata in cui dichiarerà di voler spedire 1 BTC a Bob. Quando il messaggio viene spedito nel network chiunque può verificare che sia stato firmato da Alice e che la quantità di moneta che intende inviare sia

effettivamente in suo possesso. Questa verifica è possibile farla consultando tutte le transazioni che sono state fatte da e verso l'indirizzo pubblico di Alice e quindi il suo saldo è facilmente ricavabile. In breve, una transazione comunica al network che il proprietario di un certo numero di bitcoin ha autorizzato il trasferimento di una parte di essi ad un altro proprietario. Il nuovo proprietario può ora spendere questi bitcoin creando un'altra transazione che autorizza il trasferimento a un nuovo proprietario, e così via, formando una catena di passaggi di proprietà. Le transazioni quindi sono legate tra loro da una serie di input e di output, dove per input si intende un invio verso un certo account e per output una ricezione di moneta da un certo account. In figura 1.2 è illustrato un esempio. Nel primo

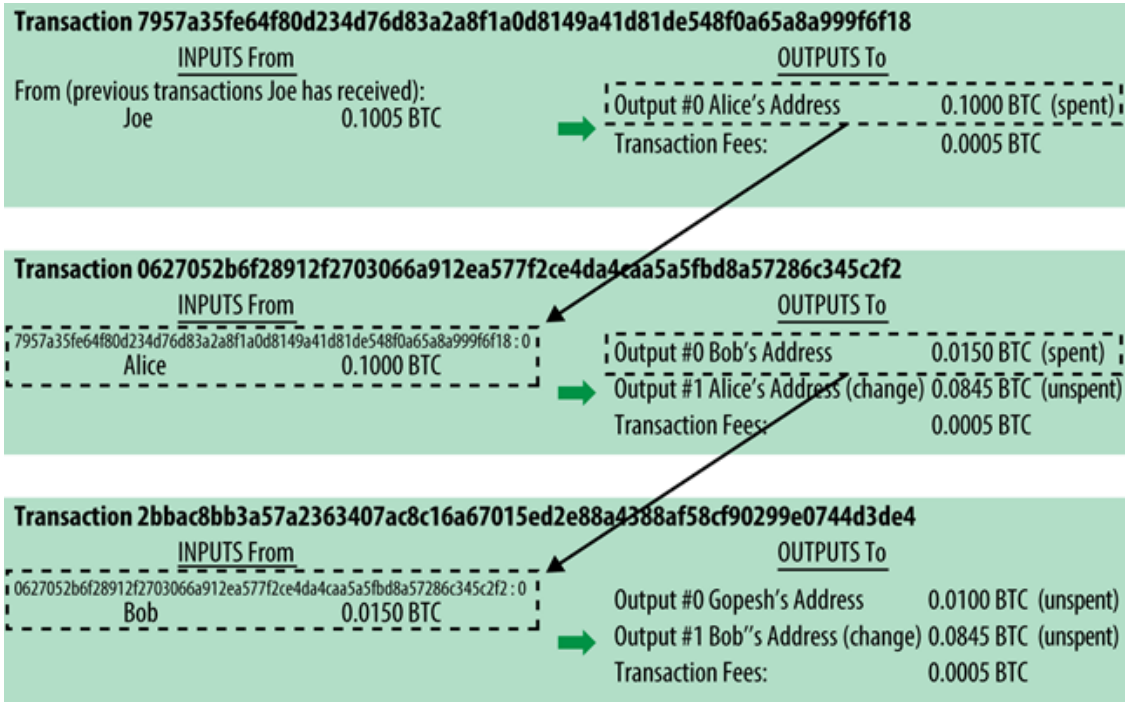

Figura 1.2: Transazione tra utenti, fonte Mastering Bitcoin [2]

riquadro il wallet di Joe crea una transazione (identificata da un codice hash) che vede l'invio di 0.1005 bitcoin, di cui 0.1000 all'indirizzo di Alice e 0.0005 in transaction fee, di fatto una commissione a beneficio di colui che aggiungerà la transazione in un blocco della blockchain. Questi utenti speciali vengono chiamati miner. Gli input provengono da output di una precedente transazione in cui Joe è stato il beneficiario e vanno a finire negli output riportati a destra. Nella seconda transazione Alice invia 0.1000 bitcoin di cui 0.0150 a Bob, 0.0845 a se stessa (come resto) e 0.0005 in transaction fee. La regola è che la somma

degli input deve sempre pareggiare quella degli output, ecco il motivo per cui Alice invia a se stessa un resto. In questo caso l'input fa riferimento all'output della transazione in cui Alice ha ricevuto bitcoin da Joe (prima freccia diagonale) e va a finire nell'output di destra. Infine, nell'ultima transazione, con lo stesso principio, Bob invia della moneta all'indirizzo di Gopesh, della moneta a se stesso e una transaction fee. Il nuovo input quindi farà riferimento all'output della transazione precedente (seconda freccia diagonale). Una forma comune di transazione è quella che aggrega input multipli in un singolo output. Transazioni come queste sono talvolta generate da wallet per "far pulizia" di quelle somme di piccolo valore che sono state ricevute come resto dei precedenti pagamenti. Infine, l'ultima combinazione possibile è una transazione che distribuisce un input a più di un output i quali rappresentano destinatari multipli, figura 1.3.

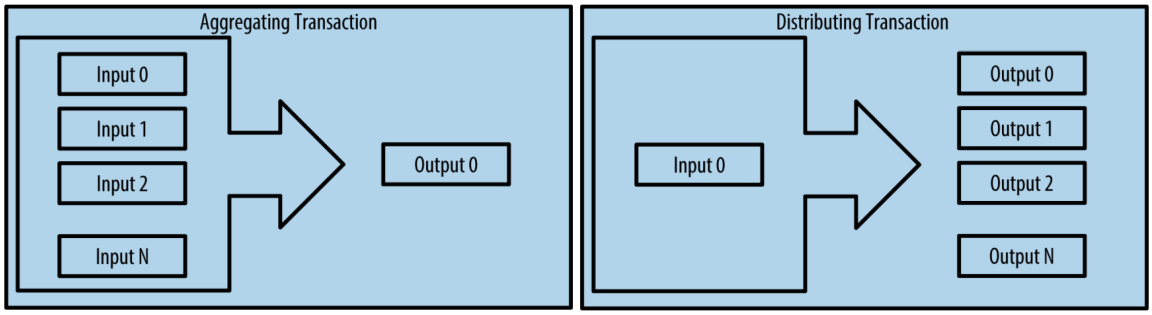

Figura 1.3: Tipologie di transazioni

Una volta che la transazione è creata, il nodo creatore la invia ai nodi circostanti connessi alla rete bitcoin per essere validata. Se il processo di validazione va a buon fine, viene ulteriormente propagata ai prossimi nodi vicini e restituito un messaggio di conferma al creatore. Il processo di inoltro di informazioni tra nodi circostanti si chiama flooding (inondazione). Per prevenire spam o attacchi di Denial of Service (DoS) ogni nodo valuta in modo indipendente ogni transazione prima di divulgarla ulteriormente. Una transazione corrotta infatti non andrà oltre un nodo. Una volta che la transazione è stata validata e propagata nella rete, viene aggiunta all'insieme delle transazioni non ancora registrate nella blockchain, chiamato memory pool, e sarà compito dei miner sceglierne alcune (maggiore priorità a quelle con transaction fee più alte) per formare i blocchi.

Struttura transazione

I mattoni fondamentali di una transazione bitcoin sono gli output non spesi della transazione (unspent transaction outputs), o UTXO. Il wallet calcola il saldo dell'utente scansionando la blockchain e aggregando tutti gli UTXO appartenenti a quell'utente. Perciò una transazione deve essere creata a partire da un UTXO. Si passerà ora all'analisi dettagliata della struttura interna di una transazione.

Dimensione	**Campo**	**Descrizione**
4 bytes	Versione	Specifica quali regole segue la seguente transazione
1–9 bytes (VarInt)	Input Counter	Quanti input sono inclusi
Variabile	Inputs	Uno o più input di transazione
1–9 bytes (VarInt)	Output Counter	Quanti output sono inclusi
Variabile	Outputs	Uno o più output di transazione
4 bytes	Locktime	Tempo che deve passare prima che la transazione venga eseguita

Tabella 1.1: Struttura interna di una transazione

Come mostrato in tabella e in precedenza discusso, una transazione è formata da uno o più input e da uno o più output, gli elementi costituenti. Con il nuovo concetto di UTXO poco fa introdotto ci si rende conto che gli UTXO consumati da una transazione sono gli input e gli UTXO creati gli output. Di seguito è presentata nel dettaglio la struttura degli output.

Dimensione	**Campo**	**Descrizione**
8 bytes	Importo	Valore bitcoin rappresentato in satoshi (10^{-8} bitcoin)
1-9 byte (VarInt)	Dimensione del Locking-Script	Lunghezza Locking-Script in byte
Variabile	Script di Locking	Uno script che definisce le condizioni necessarie per spendere l'output

Tabella 1.2: Struttura di un output

In tabella 1.2 l'output è costituito essenzialmente da due campi: il valore dell'importo espresso in satoshi e lo Script di Locking. Il bitcoin è partizionabile fino alla centomilionesima unità che in gergo viene chiamata satoshi, in onore dell'inventore. Quindi 100 milioni di satoshi formano 1 bitcoin. Il campo Locking Script chiamato anche ScriptPubKey è una condizione che viene imposta e deve essere soddisfatta al fine di poter spendere l'output in futuro. Di fatto costituisce un blocco dell'output. Si vedrà ora l'input.

Dimensione	**Campo**	**Descrizione**
32 bytes	ID della Transazione	Puntatore alla transazione contenente l'UTXO che andrà speso
4 bytes	Indice dell'Output	Numero indice UTXO speso
1-9 bytes (VarInt)	Dimensione Unlocking-Script	Lunghezza Unlocking-Script in bytes
Variabile	Unlocking-Script	Uno script che completa le condizioni del locking script dell'UTXO

Tabella 1.3: Struttura di un input

La prima parte di un input è un puntatore a una transazione passata che contiene un UTXO. La seconda parte è uno script di sblocco ovvero l'Unlocking Script detto anche scriptSig per il fatto che contiene una firma. Il wallet lo utilizza per soddisfare le condizioni di spesa impostate nell'UTXO dal Locking Script.

Script di blocco e sblocco

Il software di convalida delle transazioni di Bitcoin si basa sui due tipi di script presentati in precedenza: lo script di blocco e quello di sblocco. Con questa coppia di script si possono ottenere vari algoritmi che implementano diversi tipi di transazione: P2PKH, P2SH, multisig. La stragrande maggioranza delle transazioni però viene fatta con l'algoritmo Pay-to-Public-Key-Hash (P2PKH). Ora si vedrà come sono strutturati gli script di lock e unlock con l'implementazione P2PKH:

- Locking Script: OP_DUP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG

- Unlocking Script: <sig> <pubKey>

Le voci che cominciano con OP sono le operazioni, le restanti voci i dati. Il linguaggio utilizzato, Script, è basato su stack e permette di compiere operazioni limitate per evitare loop nel codice che causerebbero problemi di esecuzione. I linguaggi di questo tipo si dicono essere Turing incompleti proprio perché hanno una complessità limitata e tempi di esecuzione prevedibili. Uno stack è una struttura dati che può essere vista come una pila di carte. Consente due operazioni: push e pop. Push aggiunge una carta in cima al mazzo, pop la rimuove. Nel nostro caso le carte sono i valori. Date queste premesse, ora verrà fornita una prima un'idea intuitiva di come funziona il processo di verifica della transazione. Chiunque voglia mandare moneta a qualcuno, deve dimostrare di possedere effettivamente quella moneta. Se Alice vuole spedire 1 BTC a Bob costruirà una transazione in cui metterà i riferimenti di come ha ottenuto quel bitcoin, quindi riferimenti a transazioni passate. Gli altri nodi della rete verificheranno che in quelle transazioni passate Alice era la beneficiaria e che ha ottenuto importi sufficienti a formare 1 BTC. Per fare questa verifica basta comparare l'indirizzo di output della transazione passata con l'indirizzo di input della nuova transazione, ovvero quello di Alice, e vedere se corrispondono; una volta fatto ciò si deve verificare che ad autorizzare la transazione, ovvero firmarla, sia stata proprio Alice. Ora si passerà ai dettagli facendo riferimento alla figura 1.4. L'algoritmo nelle due prime iterazione inserisce nello stack i dati del Locking Script, quindi la firma e la chiave pubblica del mittente. Alla terza iterazione comincia l'Unlocking Script con un'operazione di duplicazione dell'elemento che sta in cima alla pila, la chiave pubblica. Successivamente viene calcolato l'hash160 della chiave pubblica. Alla quinta iterazione viene inserito nello stack l'hash della chiave pubblica del beneficiario della precedente transazione e con la successiva operazione vengono confrontati i due elementi in cima. Un esito positivo fa scattare anche l'ultimo controllo che consiste nel verificare che la firma sia stata fatta dal mittente. Per farlo basta applicare la chiave pubblica sulla firma, un esito positivo restituirà il valore TRUE. Con questo meccanismo i nodi della rete testano le transazioni scartando quelle che non verificano le condizioni.

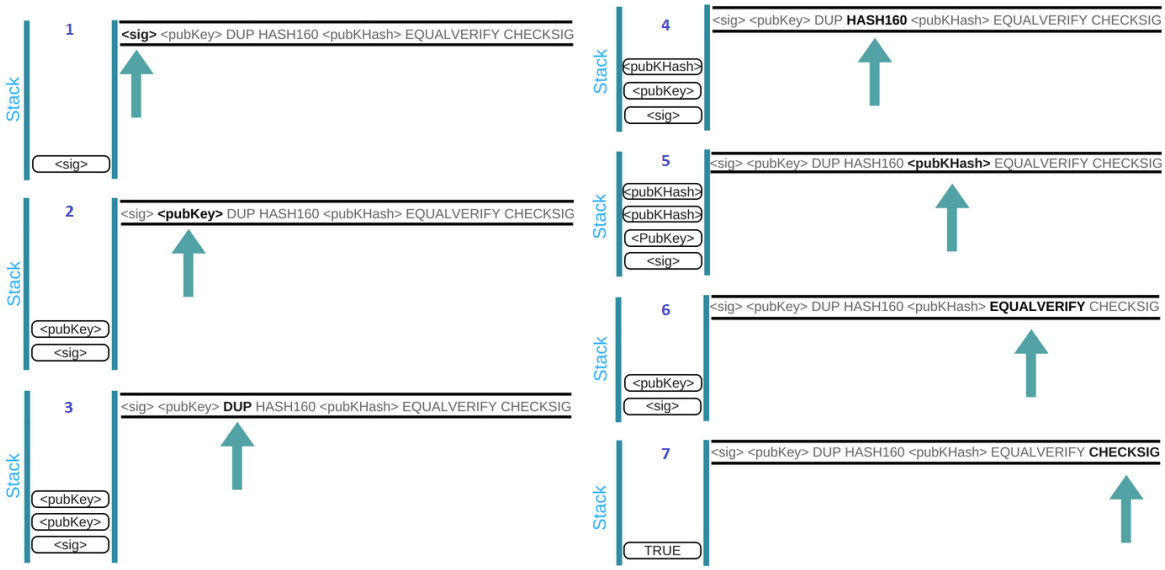

Figura 1.4: Funzionamento P2PKH

I passaggi appena visti coinvolgono molta crittografia tra cui hashing e firme digitali.

1.1.3 Crittografia

Crittografia in greco significa "scrittura segreta" ed è la tecnica di rendere incomprensibile un messaggio a chi non è autorizzato a leggerlo. Nel caso della blockchain la crittografia viene invece utilizzata non tanto per nascondere i dati bensì per rendere sicuro il sistema in modo da non permettere manomissioni degli stessi. La forza della blockchain risiede nel fatto che i dati, ovvero le transazioni, non sono nascosti ma pubblici e condivisi tra tutti i partecipanti della rete. È proprio con ciò che si ottiene la decentralizzazione. In Bitcoin sono due gli schemi crittografici utilizzati: funzioni di hash e firme digitali.
Una funzione di hash crittografica è un algoritmo che prende in input una stringa di lunghezza arbitraria m e ne restituisce in output una di lunghezza fissa h(m). Cambiando anche solamente un carattere di m si otterrebbe un output diverso e quindi, siccome i blocchi della blockchain, come detto in precedenza, sono legati da hash, una modifica di un certo blocco invaliderebbe tutti quelli successivi dal momento che gli hash calcolati non corrisponderebbero più. In questo modo è possibile verificare facilmente l'integrità della blockchain per capire se è tutto in ordine. Una buona funzione di hash ha una serie di proprietà.

- velocità di calcolo: dato un qualunque m, la funzione h(m) è calcolabile in modo veloce ed efficiente
- unidirezionalità o resistenza alle controimmagini: dato h(m) è computazionalmente molto difficile tornare a m
- resistenza alle collisioni: è molto difficile trovare m_1 e m_2 tali che $h(m_1) = h(m_2)$

Queste proprietà, come si vedrà tra poco, sono fondamentali per implementare il sistema crittografico di firme digitali. Le funzioni di hash utilizzate in Bitcoin sono due: SHA-256 (restituiscono un output a 256 bit che corrispondono a 64 caratteri esadecimali) e RIPEMD-160.
Le transazioni in Bitcoin sono possibili grazie alla crittografia asimmetrica detta anche a chiave pubblica/privata. Introdotta nel 1976, la crittografia asimmetrica prevede una coppia di chiavi in possesso a mittente e destinatario per cifrare e decifrare un messaggio scambiato, evitando così il problema legato alla necessità da parte del mittente di comunicare in modo sicuro al destinatario, l'unica chiave utile alla cifratura/decifratura presente invece nella crittografia simmetrica. La crittografia asimmetrica è impiegata per due scopi: cifrare un messaggio o verificarne l'autenticità. Nel primo caso il mittente cifra il messaggio con la chiave pubblica del destinatario, in modo tale che solo quest'ultimo grazie alla propria chiave privata, possa decifrarlo e leggerlo. Nel secondo caso, quello che interessa Bitcoin, il messaggio viene cifrato dal mittente con la chiave privata in suo possesso e il destinatario può verificarne l'autenticità decifrandolo con la chiave pubblica del mittente. L'algoritmo utilizzato da Bitcoin per generare questa coppia di chiavi è ECDSA (Elliptic Curve Digital Signature Algorithm), un algoritmo di firma digitale. Una firma digitale è uno schema matematico che serve a dimostrare l'autenticità di un messaggio o di un documento digitale. Una firma digitale valida dà al destinatario una ragione per credere che il messaggio sia stato creato da un mittente conosciuto, che il mittente non possa negare di aver inviato il messaggio e che il messaggio non sia stato alterato durante il transito (figura 1.5). Quindi con le firme digitali in Bitcoin si vogliono ottenere tre cose:

- autorizzazione alla spesa: chi possiede la chiave privata può usarla per spendere i propri fondi

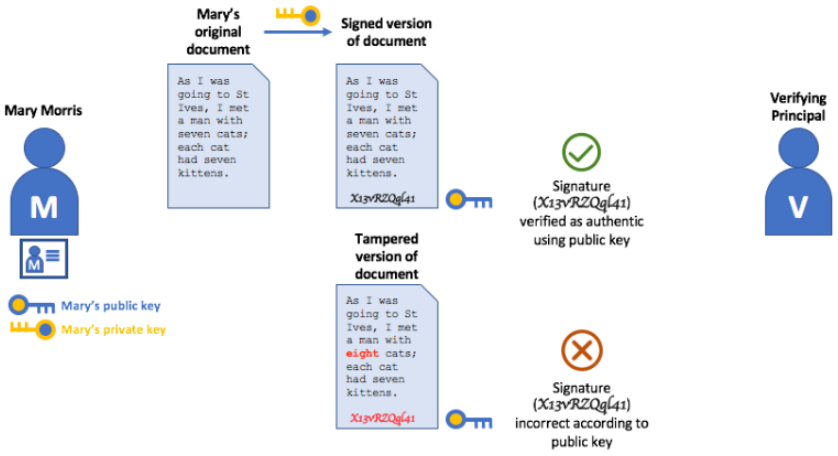

Figura 1.5: Firma digitale su un documento, fonte Doc HF

- non ripudiabilità: le parti che intervengono in un determinato scambio non possono poi negare di aver preso parte allo stesso

- non modificabilità: una volta che una transazione o parte di essa è firmata, non la si può modificare senza invalidarla

L'algoritmo che genera la coppia di chiavi è implementato nel wallet degli utenti; come prima cosa produce una chiave privata k che è un numero casuale. Dalla chiave privata, tramite una procedura che utilizza una specifica curva ellittica (EC) cioè una funzione crittografica a una via (seconda proprietà delle funzioni di hash vista prima), e un insieme di costanti matematiche, viene generata la corrispondente chiave pubblica K. Di quest'ultima viene dapprima calcolato un doppio hash ottenendo A, poi A viene convertito nell'indirizzo pubblico usato nelle transazioni. Più in dettaglio, per ottenere k il software del wallet applica SHA-256 su di una stringa generata casualmente e se il valore esadecimale dell'output ottenuto è minore di 2^{256}-1, tale stringa di 256 bit è la chiave privata. Si possono quindi generare circa 10^{77} chiavi private. La chiave pubblica è ottenuta moltiplicando k per un predeterminato punto della curva ellittica chiamato punto generatore G e quindi è univocamente determinata: K = k*G. Questa espressione in una curva ellittica viene chiamata "funzione trappola", in quanto è molto facile da fare in una direzione (moltiplicazione tra k e G) ma estremamente complicata da fare nella direzione opposta (divisione tra K e G). Conoscendo quindi la chiave privata è facile ricavare quella pubblica, mentre è estremamente difficile ricavare quella privata dalla pubblica. L'hash della chiave pubblica si ottiene applicando

a K due funzioni di hash nel modo seguente: A = RIPEMD160(SHA256(K)). L'indirizzo pubblico si ottiene infine con la traduzione di A in Base58Check encode, un sistema di codifica in base 58 che serve ad aumentare la leggibilità dell'indirizzo. In figura 1.6 è rappresentato uno schema riassuntivo.

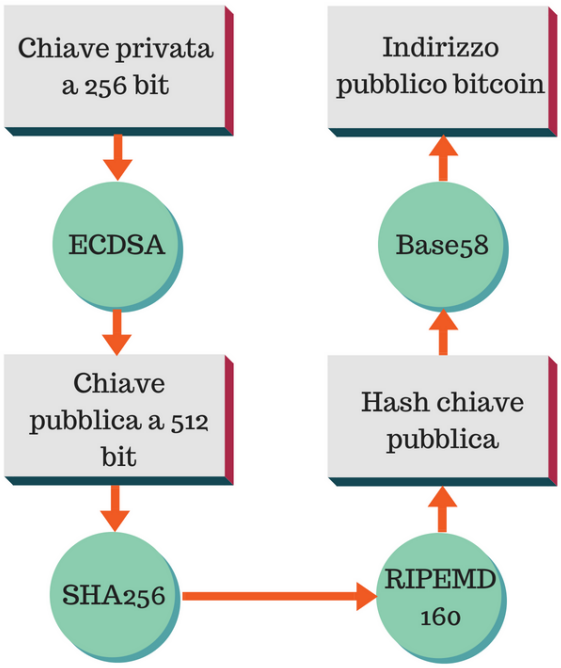

Figura 1.6: Generazione chiave pubblica/privata

Fin qui si è visto come viene generata una coppia di chiavi pubblica/privata. Ora si esporrà l'algoritmo di impiego di questa coppia di chiavi nelle transazioni bitcoin. Una firma digitale è costituita da due parti. La prima corrisponde a un algoritmo di creazione della firma e usa una chiave privata (la chiave di firma) su un messaggio (la transazione). La seconda parte è un algoritmo che consente a chiunque di verificare la firma dato il messaggio e la chiave pubblica. La firma è così composta: Sig = $F_{sig}(F_{hash}(m), dA)$ dove:

- dA è la chiave privata che fa la firma
- m è la transazione o parte di essa
- F_{hash} è la funzione di hash
- F_{sig} è l'algoritmo di firma

La funzione F_{sig} produce la firma Sig dell'hash di un messaggio criptato con dA ed è composta da due valori che solitamente si indicano con R e S: Sig =

(R,S). L'algoritmo di verifica, senza entrare troppo nel dettaglio, utilizza la chiave pubblica e il numero S come parametri di una certa equazione, la risolve e se il valore risultante è pari ad R allora la firma è valida. Per come è costruita questa complessa equazione matematica, non è possibile ricavare il valore della chiave privata a partire da quella pubblica. La correttezza della firma quindi deriva dal fatto che solo chi possiede la chiave privata che ha generato la chiave pubblica può aver prodotto tale firma della transazione e per verificarla basta impiegare la chiave pubblica su di essa (settimo passaggio dell'algoritmo P2PKH, figura 1.4). Trattato l'aspetto crittografico, nel paragrafo successivo verrà affrontata più da vicino la struttura della blockchain in termini di blocchi.

1.1.4 Blocchi

La blockchain, come già constatato, è una catena di blocchi concatenati tra loro. In quella di Bitcoin, ogni blocco è identificato da un hash, generato usando l'algoritmo di hash crittografico SHA-256 sull'intestazione o header del blocco stesso. Ciascun blocco si riferisce a quello precedente, chiamato genitore, attraverso questo hash. Quindi, in altri termini, ogni blocco ha nel proprio header l'hash del blocco precedente. L'unico blocco che non ha genitore è il primo mai creato, il genesis block. In realtà può verificarsi una situazione temporanea in cui un blocco abbia più di un figlio, esiste cioè una moltitudine di blocchi che ha nel proprio header l'hash dello stesso genitore. Ciò accade quando i miner scoprono quasi contemporaneamente nuovi blocchi e li propongono alla rete. Ciò dà luogo a un fork, ovvero a una diramazione. Il fork si risolve con la scelta di un figlio tra i tanti che andrà quindi ad incrementare di una unità la catena. Maggiori dettagli verranno forniti in seguito. Nello specifico un blocco è formato dai campi riportati in tabella 1.4.

Dimensione	**Campo**	**Descrizione**
4 bytes	Dimensione blocco	Dimensione del blocco in bytes
80 bytes	Header del blocco	Vari campi
1–9 bytes (VarInt)	Contatore transazioni	Numero transazioni contenute
Variabile	Transazioni	Lista transazioni contenute

Tabella 1.4: Struttura del blocco

L'header del blocco comprende a sua volta tre insiemi di metadati. Il primo

riguarda la concatenazione dei blocchi e quindi contiene il riferimento all'hash del blocco precedente; il secondo è formato dai campi difficoltà, timestamp e nonce che riguardano il concetto di mining che verrà affrontato più avanti; il terzo contiene la radice del merkle tree, una struttura dati che consente di verificare la presenza delle transazioni nei blocchi in modo più efficiente. In tabella 1.5 si ha la struttura dell'header del blocco. Il numero di transazioni contenute in un blocco

Dimensione	**Campo**	**Descrizione**
4 bytes	Versione	Versione del software usato
32 bytes	Hash del blocco precedente	Riferimento al blocco genitore
32 bytes	Merkle Root	Hash della radice del Merke Tree di questo blocco
4 bytes	Timestamp	Orario di creazione del blocco
4 bytes	Target di difficoltà	Il target di difficoltà della Proof of Work
4 bytes	Nonce	Valore trovato per risolvere la Proof of Work

Tabella 1.5: Struttura dell'header del blocco

in genere è maggiore di 500 e ciascuna occupa in media 225 bytes. Considerando che l'header pesa 80 bytes, un blocco completo di tutte le transazioni è più di 1000 volte superiore al proprio header. Informazioni relative alla blockchain di Bitcoin si possono reperire facilmente consultando i blockchain explorer, siti che mettono a disposizione numeri, grafici e statistiche. Uno famoso per Bitcoin è Blockchain.com. Oltre che per fini statistici, può essere utilizzato ad esempio per controllare l'esito delle transazioni effettuate da e verso il proprio indirizzo. Infatti, dal momento che la blockchain è pubblica, tutto è in chiaro ed esplorabile dal primo blocco della catena. Attualmente la blockchain di Bitcoin contiene più di mezzo milione di blocchi e ha una capienza di oltre 200 GB. Questi dati verranno ampiamente discussi nelle sezioni successive.

Merkle Tree

Ora verrà approfondito un aspetto molto importante della blockchain di Bitcoin, il Merkle tree. Ogni blocco contiene nel proprio header il Merkle root che è

la radice di un Merkle tree e rappresenta un riassunto di tutte quelle che sono le transazioni contenute all'interno del blocco stesso. Anche conosciuto come albero binario di hash, il Merkle tree è un albero in cui ogni foglia è etichettata con l'hash di un certo dato. Le etichette dei genitori sono a loro volta ottenute dall'hash della concatenazione degli hash dei figli. I dati alle foglie dei quali vengono calcolati gli hash, nel caso di Bitcoin sono le transazioni. Quindi l'albero è costruito ricorsivamente bottom-up, cioè dal basso verso l'alto. In figura 1.7 c'è una rappresentazione dell'albero per quattro transazioni. Come prima cosa si ricavano le etichette delle foglie con un doppio hash SHA-256 delle transazioni del blocco. Ad esempio H1 = SHA256(SHA256(Transazione 1)). Successivamente i genitori si ottengono dal calcolo dell'hash della concatenazione degli hash dei figli. Ad esempio H5 = SHA256(SHA256(H1+H2)). Procedendo ricorsivamente si

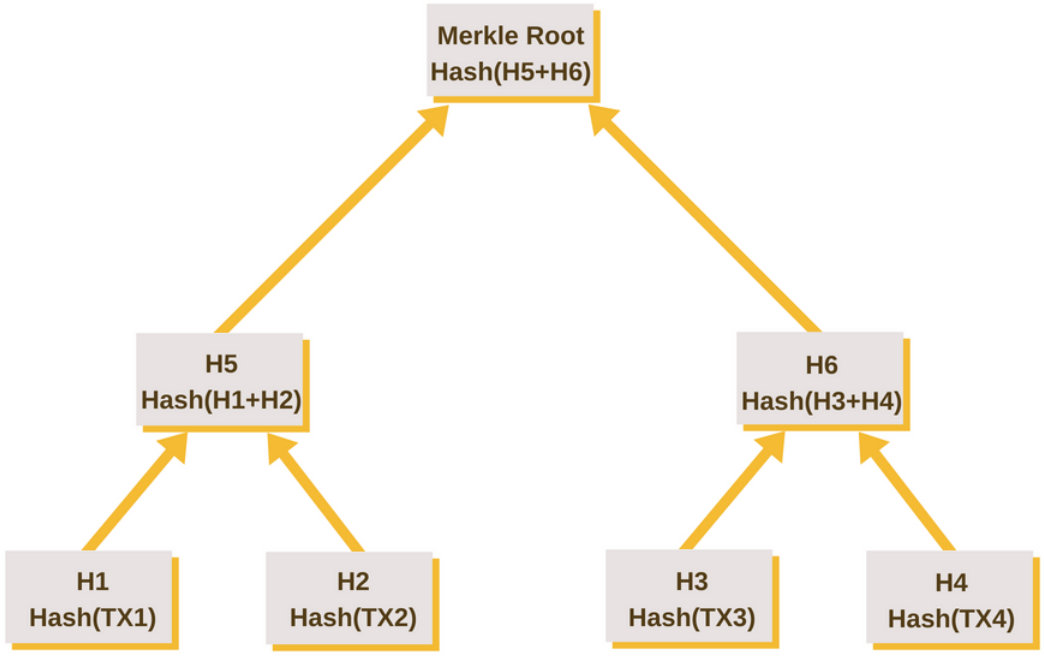

Figura 1.7: Merkle Tree di quattro transazioni

arriverà a calcolare il nodo dell'albero che sta più in alto, la radice del Merkle Tree. L'hash di 32 byte della radice viene salvato nell'header del blocco e rappresenta il riassunto delle transazioni contenute nel blocco. Lo scopo di questa struttura dati è favorire un rapido controllo circa la presenza di una data transazione in un blocco. Per effettuare una verifica, infatti, basta considerare un ramo dell'albero che va dalla foglia che rappresenta la transazione interessata, alla radice, chiamato Merkle path. Dalle proprietà di un albero binario è noto che tale percorso è lungo

$\log_2$(N) dove N è il numero di foglie. Per verificare quindi la presenza di una transazione basta produrre un numero logaritmico di hash a 32 byte. Il Merkle tree è molto utilizzato da una certa tipologia di nodi della rete di cui si discuterà nel prossimo paragrafo, i nodi SPV.

1.1.5 Rete Bitcoin

Bitcoin è basato su un'architettura di rete peer-to-peer (abbreviato in P2P). I nodi che vi partecipano sono dunque paritari a differenza di una rete client-server, in cui i client si rivolgono ai server per ottenere un certo servizio. In Bitcoin non c'è alcun server, nessun servizio centralizzato e nessuna gerarchia. Grazie a questo schema di rete si ottengono decentralizzazione e resilienza, per cui il sistema riesce a resistere all'usura, continuando a garantire la disponibilità dei servizi erogati. Sebbene i nodi siano tutti uguali, svolgono differenti ruoli a seconda di alcune funzionalità: routing, mantenimento della blockchain, mining e servizi del wallet. Tutti i nodi svolgono la funzione di routing, cioè di instradamento delle informazioni attraverso la rete e di validazione e propagazione di transazioni e blocchi. Alcuni nodi, chiamati *full node*, mantengono anche una copia intera della blockchain che, come detto in 1.1.4, è attualmente di oltre 200 GB. È grazie a loro che la blockchain risulta decentralizzata perché, come si vedrà più avanti quando verrà discusso il concetto di consenso distribuito, i full node contribuiscono al consenso, dato che mantengono l'intera copia della blockchain. Altri nodi, chiamati *SPV* (simplified payment verification) o *lightweight node*, invece mantengono solo un sottoinsieme della blockchain. Per questo motivo non contribuiscono al processo di consenso distribuito, in quanto non possiedono le informazioni per intero. Questi nodi generalmente sono costituiti da utenti che installano il wallet nel cellulare e quindi per motivi di spazio di archiviazione non possono scaricare l'intera blockchain. Gli SPV tengono solo gli header dei blocchi che pesano 80 byte ciascuno per un totale di circa 40 MB, cifra molto più bassa rispetto ai 200 GB dell'intera blockchain attuale. Quando un SPV ha bisogno di scoprire la quantità di moneta spendibile di un certo indirizzo bitcoin (UTXO), deve ottenere dai full node tutte le transazioni che coinvolgono quell'indirizzo. Una volta che il nodo light ha ottenuto una certa transazione, deve verificare che essa faccia parte di un blocco. A questo punto si fa inviare dai full node il Merkel path che connette tale transazione al Merkle root. In questo modo, tramite un numero logaritmico

di calcoli di hash, può velocemente controllare la presenza della transazione nel blocco. Gli SPV inoltre utilizzano l'header per connettere il blocco al resto della blockchain. Quindi la combinazione di questi due collegamenti, la transazione con il blocco e il blocco con la blockchain, provano che la transazione è registrata nella blockchain. Altri tipi di nodi sono quelli che effettuano il mining, il processo con cui vengono scoperti nuovi blocchi della blockchain e possono essere sia full che light node.
Quando un nodo decide di entrare a far parte della rete, deve scoprire almeno un nodo che già vi partecipa e collegarsi ad esso. Viene stabilita una connessione TCP tra i nodi, solitamente attraverso la porta 8333, quella utilizzata da Bitcoin, e il nodo remoto inizia un "handshake", trasmettendo un messaggio di "version" al nodo entrante, contenente una serie di informazioni base riportate nella tabella 1.6.

Campo	**Descrizione**
nVersion	Versione del protocollo P2P
nLocalService	Lista dei servizi locali supportati dal nodo
nTime	Orario corrente
addrYou	Indirizzo IP del nodo remoto visto dal nuovo nodo
addrMe	Indirizzo IP del nodo locale
subver	Versione del wallet Bitcoin utilizzato
BestHeight	Altezza dell'ultimo blocco della blockchain del nodo locale

Tabella 1.6: Messaggio di version

Il nodo entrante analizza i dati del messaggio di version del nodo remoto e se risulta compatibile con esso, accetta la connessione. Per trovare nodi peer remoti, il nodo entrante può o interrogare dei server DNS che forniscono un elenco statico di indirizzi IP o in alternativa collegarsi all'indirizzo IP di almeno un nodo Bitcoin noto. Una volta stabilite una o più connessioni, il nuovo nodo invia un messaggio "addr" contenente il proprio indirizzo IP ai suoi vicini. I vicini a loro volta inoltrano il messaggio addr ai loro vicini, assicurando che il nuovo nodo connesso diventi noto. Dopo l'avvio, un nodo registra le proprie connessioni più recenti che hanno avuto successo, in modo che in seguito a un eventuale riavvio, possa ristabilire rapidamente le connessioni con la sua precedente rete di peer. Se nessuno dei peer precedenti risponde alla sua richiesta di connessione, il nodo

può usare di nuovo i server DNS per effettuare l'avvio. Un full node che si collega tramite questa procedura alla rete P2P come prima cosa proverà a costruire la blockchain. In partenza possiede solo il blocco genesi che è già incluso nel software client. I restanti blocchi invece dovrà scaricarli dai peer connessi tramite un messaggio di richiesta "getblocks". Il peer che riceve la richiesta trasmette i primi 500 blocchi usando un messaggio di inv (inventario). Il processo continua fino a che il full node entrante non riceve tutta la blockchain. Un nodo che rimane offline per un certo periodo di tempo, una volta tornato online si fa mandare dai suoi peer i blocchi mancanti e per comunicare a che punto della blockchain sia arrivato, utilizza il campo BestHeight del messaggio di version in cui è riportata l'altezza dell'ultimo blocco posseduto.
Le transazioni che vengono effettuate dagli utenti della rete e sono in attesa di essere incluse nella blockchain, vengono temporaneamente incluse nel cosiddetto *transaction pool* o *memory pool*. Quasi tutti i nodi della rete mantengono questo insieme di transazioni in memoria locale, a patto che risultino valide e verificate. Esiste anche un ulteriore insieme che viene mantenuto ed è quello delle transazioni *orfane*. È costituito da quelle transazioni che non fanno riferimento a transazioni padre presenti nella blockchain o nel memory pool. Ogni volta che un nodo riceve una transazione destinata al memory pool, controlla se nell'insieme di quelle orfane è presente la transazione figlia. In quel caso sposta anch'essa nel memory pool. Infine alcuni nodi mantengono anche un database di UTXO, cioè un insieme di tutti gli output non spesi a partire dal blocco genesi. A differenza degli altri due insiemi che vengono inizializzati vuoti, quest'ultimo parte già popolato.
Data una panoramica dell'architettura di rete utilizzata in Bitcoin, si passerà ora a descrivere quelli che sono i concetti fondamentali della blockchain, con cui si concretizza l'idea di decentralizzazione.

1.1.6 Mining e consenso

Con il termine mining, tradotto in italiano con "estrazione", nel linguaggio delle blockchain si intende un processo che consente di raggiungere vari obiettivi, tra cui quello di immettere nuova moneta nel sistema. Nel caso di Bitcoin la metafora del minatore che estrae faticosamente minerali preziosi da una miniera è opportuna, in quanto il miner prima di riuscire ad introdurre nuova moneta,

deve far fronte a complicati e dispendiosi calcoli computazionali previsti da un algoritmo di consenso, la Proof of Work (PoW). La nuova moneta viene emessa sotto forma di ricompensa al miner che riesce ad aggiungere un nuovo blocco alla blockchain. Il meccanismo di mining però non ha questo solo scopo, ma anche e soprattutto di implementare due caratteristiche fondamentali della blockchain, decentralizzazione e sicurezza. Il lavoro del miner consiste nel pescare dal memory pool nuove transazioni che necessitano di essere inserite nella blockchain e nel creare un blocco. In media è previsto che un blocco venga inserito nella catena ogni 10 minuti. Si vedrà in seguito la ragione di questa cadenza. Le transazioni che diventano parte di un blocco aggiunto alla blockchain, sono considerate confermate e quindi la moneta ricevuta dai destinatari di quelle transazioni diviene spendibile.

Premi del miner

I miner ricevono due premi da questo lavoro: le *fee* contenute in ciascuna transazione del blocco estratto e il *block reward*. La *coinbase transaction* è la prima transazione che viene inserita in ciascun blocco e consente a chi lo genera di indicare l'indirizzo di destinazione a cui inviare i due tipi di premio. Il miner per riscattarli basterà che inserisca nell'output un indirizzo in suo possesso. Il block reward segue una legge per cui ogni quattro anni si dimezza, precisamente ogni 210.000 blocchi. Nel gennaio del 2009 era di 50 BTC mentre attualmente è di 12,5 BTC. I premi ottenuti dal block reward perciò decrescono esponenzialmente simulando i rendimenti decrescenti ottenuti dall'estrazione di metalli preziosi che sono in quantità finita. Il numero totale di bitcoin previsto è di 21 milioni e l'ultimo satoshi (un milionesimo di bitcoin) si prevede verrà estratto nel 2140. In figura 1.8 è rappresentato il grafico dei bitcoin estratti fino ad oggi e si può notare che sono oltre 17 milioni quindi più dell'80% del totale.
Dopo il 2140 i miner guadagneranno solamente dalle fee che gli utenti pagano per effettuare le transazioni. Ad oggi le commissioni rappresentano una piccola frazione del block reward, ma con l'aumentare del numero di transazioni per blocco, aumenterà anche il loro valore. Le commissioni sono calcolate in base alla dimensione della transazione in kilobyte, non dal valore dei bitcoin scambiati durante la transazione. I miner quando devono formare una blocco, danno la priorità alle transazioni con fee più alte in modo da ottenere guadagni maggiori.

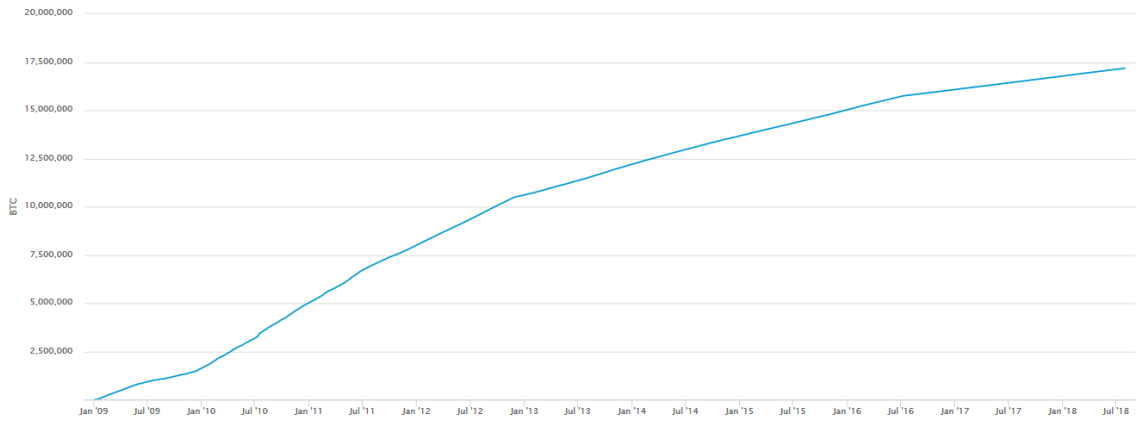

Figura 1.8: Bitcoin estratti fino ad oggi, fonte Blockchain.com

Le fee non sono obbligatorie ma se un utente non le include, con ogni probabilità non vedrà la propria transazione finire nella blockchain. I wallet in genere usano un algoritmo di stima delle commissioni che calcola la tariffa appropriata in base ai valori delle commissioni concorrenti e offre la possibilità di scegliere tariffe ad alta, media e bassa priorità. Molte applicazioni di wallet invece utilizzano servizi di terze parti per i calcoli delle commissioni. Un servizio noto è Bitcoinfees. Il grafico 1.9 mostra la stima delle fee per 13 intervalli di satoshi per byte e il

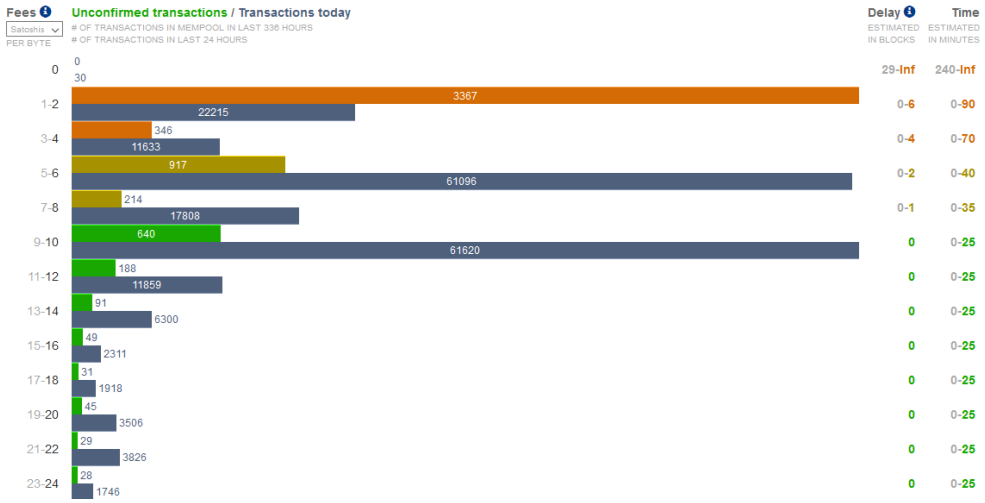

Figura 1.9: Stima delle fee di Bitcoin, fonte Bitcoinfees

relativo tempo di conferma delle transazione previsto in numero di blocchi e minuti. Per ogni intervallo sono presenti due barre orizzontali che mostrano il numero di transazioni non confermate e quello di transazioni totali nelle ultime 24 ore. Ad esempio per l'intervallo di 5-6 satoshi ci sono 917 transazioni non

confermate su 61096 totali e il tempo stimato per la conferma di una transazione va da 0 a 40 minuti. Le commissioni ad alta priorità, in base al grafico, sono quelle dell'intervallo 9-10 satoshi/byte. Considerando che una transazione in media occupa 225 byte, il costo al valore di 10 satoshi/byte è di 2250 satoshi, che corrisponde a 0,0000225 BTC e quindi 0,17$ (con il valore attuale di quasi 7500$ per 1 BTC). Le fee fungono anche da meccanismo di sicurezza contro attacchi di Denial of Service, perché rendono costoso, per gli aggressori, inondare la rete di transazioni.
Il lavoro del miner, come si diceva prima, serve a raggiungere gli obiettivi previsti dal protocollo: decentralizzazione e sicurezza. Ciò è possibile grazie al concetto di *consenso distribuito*.

Consenso distribuito

Raggiungere il consenso distribuito in una rete di nodi consiste nel far sì che i nodi convergano su un valore comune a fronte di fallimenti di tipo crash o arbitrari. Questo problema è stato teorizzato nel 1982 dai matematici Leslie Lamport, Marshall Pease e Robert Shostak con la metafora dei generali bizantini [3]. Si immagini lo scenario in cui diverse divisioni dell'esercito bizantino siano accampate fuori da una città nemica e che ciascuna divisione sia comandata dal proprio generale. I generali possono comunicare tra loro solo tramite messaggeri. Dopo aver osservato il nemico, devono decidere un piano d'azione comune: attaccare o ritirarsi. Tuttavia, alcuni generali traditori potrebbero cercare di impedire ai generali leali di adottare la stessa tattica, mandando l'esercito incontro a una sconfitta. Gli autori dell'articolo hanno provato che il problema non ha soluzione se i generali traditori sono almeno un terzo. Nonostante la rete Bitcoin non si presti bene a questo tipo di modello per via di alcune particolarità proprie del protocollo, il problema dei generali bizantini può essere utilizzato per descrivere alcune situazioni analoghe che possono verificarsi. Si supponga che un generale traditore comunichi a due generali di attaccare e ad altri due di ritirarsi, come in figura 1.10. In Bitcoin uno schema di questo tipo in cui si mandano informazioni diverse a diverse parti della rete corrisponde a un *attacco di doppia spesa*. Tale attacco consiste nel voler spendere la stessa moneta due volte, ne verrà ora illustrato un esempio. Gli attori in gioco sono Alice, l'attaccante, e Bob, un commerciante. Alice vuole acquistare un prodotto da Bob

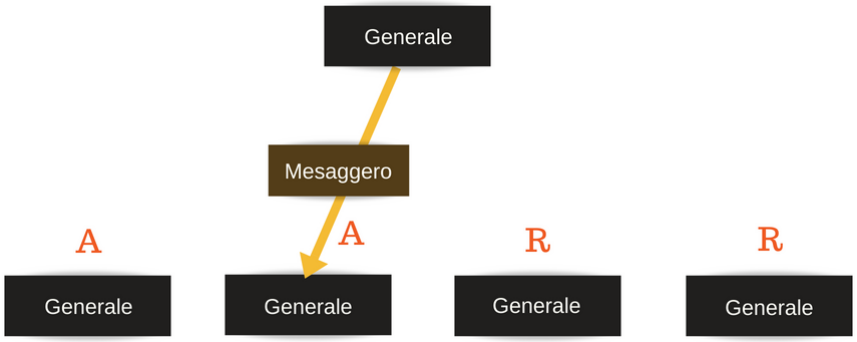

Figura 1.10: Attacco di doppia spesa

dal valore di 1 BTC ed effettua a suo favore una transazione (T1). Una volta che T1 è stata trasmessa alla rete, un miner crea un blocco che la contiene e lo diffonde alla rete. Bob a questo punto spedisce il prodotto ad Alice. Alice fa partire l'attacco di doppia spesa creando una seconda transazione (T2) in cui invia ad un indirizzo in suo possesso lo stesso bitcoin che aveva mandato a Bob. Un nodo da lei controllato crea un blocco in cui è presente T2 e lo diffonde nella rete. Se la catena più lunga che si forma in rete, cioè quella costituita da più blocchi, contiene il blocco in cui è presente T2 anziché quello in cui è presente T1, Alice di fatto avrà ottenuto il prodotto gratis. È evidente che entrambi i blocchi, quello contenente T1 e quello contente T2, non possono coesistere nella blockchain in quanto la transazione più recente delle due non verrebbe accettata perché la moneta risulterebbe già spesa nella transazione più vecchia. Un altro attacco di doppia spesa si può verificare se Bob, ingenuamente, spedisce il prodotto prima ancora che T1 venga inserita in un blocco; si ha così una transazione a conferma zero. È consigliabile allora attendere un certo numero di conferme, dove per conferma si intende un blocco inserito nella catena, prima di considerare accettata una transazione; Nakamoto nel suo paper con una serie di calcoli probabilistici individua questo numero in sei blocchi, in seguito si capirà perché. Dal problema dei generali bizantini è possibile ottenere un'altra configurazione di attacco; si pensi al caso in cui un comandante fa pervenire numerosi messaggi ad un altro comandante tramite diversi messaggeri, come in figura 1.11, allo scopo di ottenere maggior consenso. In un sistema di rete anonimo come Bitcoin si potrebbero utilizzare un numero arbitrario di indirizzi IP per impersonare più nodi. L'attacco si chiama Sybil Attack ed eredita il nome dalle sibille, esseri mitologici immortali che vivevano in diverse parti del mondo

e in diversi periodi e si presentavano con differenti identità. Bitcoin risolve questo

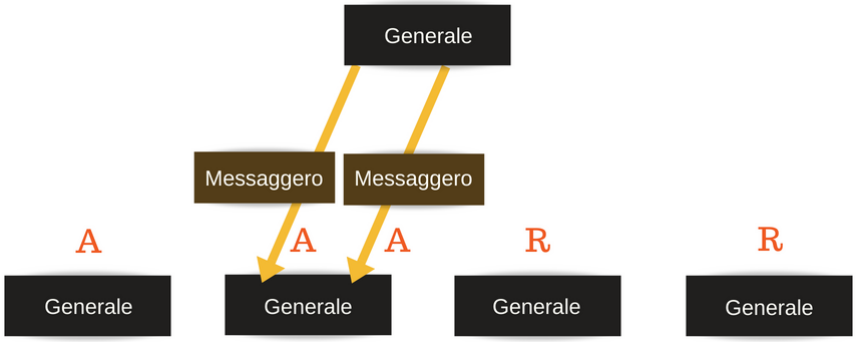

Figura 1.11: Sybil Attack

tipo di problemi con un meccanismo particolare: la Proof of Work. Esistono dei problemi difficili da risolvere ma facili da verificare. Ad esempio un cubo di Rubik può richiedere parecchio tempo prima di essere risolto ma una volta che si arriva alla soluzione, basta un colpo d'occhio per verificarla. L'idea di Nakamoto è stata la seguente: ad ogni messaggio scambiato dai generali deve essere associata la soluzione di un problema difficile da risolvere. Quindi se un comandante vuole mandare un messaggio di attacco o ritirata deve prima risolvere un problema difficile e poi allegare la soluzione che sarà facilmente verificata da chi riceve il messaggio. In questo modo risulta molto oneroso per un nodo mandare numerosi messaggi malevoli attraverso la rete. Date queste premesse, il consenso in Bitcoin si raggiunge con un processo diviso in quattro fasi indipendenti che avvengono tra i nodi della rete:

1. verifica indipendente, da parte di ogni full node, della lista di criteri che ogni transazione deve rispettare

2. raggruppamento in nuovi blocchi delle transazioni che rispettano i requisiti con annessa dimostrazione del calcolo effettuato attraverso la PoW

3. verifica indipendente da parte di ciascun nodo dei nuovi blocchi proposti e inserimento di essi nella catena

4. selezione indipendente da parte di ciascun nodo della catena con la maggior spesa computazionale in termini di PoW

Per quanto riguarda il primo punto, si era già accennato in 1.1.2 che i nodi prima di inoltrare una transazione ai vicini, verificano che tale transazione rispetti dei requisiti che sono di seguito elencati:

- la sintassi e la struttura dati devono essere corrette
- né la lista degli input né quella degli output devono essere vuote
- la dimensione in byte della transazione deve essere minore di MAX_BLOCK_SIZE
- ciascun output deve essere all'interno del range consentito (meno di 21 milioni di bitcoin)
- la dimensione deve essere maggiore o uguale di 100 byte
- la transazione deve essere nel pool o in un blocco del ramo della catena principale
- per ogni input, se l'output referenziato esiste in qualsiasi altra transazione nel pool, la transazione deve essere rifiutata; se non esiste, va a finire nel pool delle transazioni orfane
- per ogni input, se la transazione di output referenziata è un output coinbase, deve avere almeno 100 conferme
- per ogni input, l'output referenziato deve esistere e non può essere già speso
- rifiutare la transazione se la somma dei valori di input è inferiore alla somma dei valori di output
- gli script di sblocco per ciascun input devono essere convalidati rispetto agli script di blocco degli output corrispondenti

Per quanto riguarda il terzo punto, i criteri di verifica della correttezza di un blocco sono:

- la struttura dati del blocco è sintatticamente valida
- l'hash del block header è minore di un certo target
- il timestamp del blocco è minore di due ore nel futuro

- la dimensione del blocco è all'interno di limiti accettabili
- la prima transazione è la coinbase
- tutte le transazioni rispettano i criteri del punto precedente

Riguardo al quarto punto, quando un miner produce un blocco, lo propone alla rete; i nodi riceventi mantengono tre insiemi di blocchi: quello connesso alla catena principale, quello che forma diramazioni della blockchain (catene secondarie) e quello dei blocchi orfani. La catena principale è quella per cui è stato speso il quantitativo maggiore di potenza computazionale, quindi la catena più lunga. La costruzione di un blocco può quindi essere vista come una gara tra miner. Una volta che un miner ha costruito un nuovo blocco e risolto la PoW, lo invia alla rete. La gara termina quando la maggior parte dei nodi accetta quel blocco che andrà ad incrementare la blockchain di una unità; a quel punto tutti i miner cominceranno una nuova competizione che li vedrà impegnati a costruire il blocco successivo per guadagnare i premi. Si vedrà ora nel dettaglio in cosa consiste l'algoritmo di consenso.

Proof of Work

La Proof of Work è un espediente nato prima di Bitcoin con cui si impone a chi richiede un servizio di eseguire dei "compiti", in genere dei calcoli computazionali per scoraggiare attacchi alla rete di tipo Denial of Service o Spam. Nakamoto utilizza la Proof of Work per implementare il meccanismo con cui la rete Bitcoin raggiunge il consenso e si ispira ad Hashcash, ideata da Adam Back nel 1997 [4]. La Pow di Bitcoin consiste quindi in una gara matematica i cui partecipanti, i miner, competono per ottenere la coinbase e così facendo contribuiscono alla realizzazione della blockchain. Con la PoW si ottiene un sistema democratico di elezione casuale di chi dovrà fare il blocco successivo e ciò garantisce, assieme al fatto che la blockchain sia pubblica, la proprietà di decentralizzazione. Il consenso si concretizza con la scelta da parte dei nodi della rete della catena con la maggiore spesa computazionale, punto 4) del paragrafo precedente. La PoW risolve anche il problema della determinazione della rappresentatività in un sistema di decisioni prese a maggioranza. Se la maggioranza fosse basata sul principio "un indirizzo IP-un-voto", potrebbe essere sovvertita da chiunque fosse in grado di allocare molti IP (Sybil Attack). La proof-of-work invece segue

il principio “una CPU-un voto”. Nakamoto nel suo paper parla anche di come con la PoW si riesca ad implementare un sistema di marcatura temporale per evitare il problema della doppia spesa, rifacendosi al concetto di "servizio di timestamping" già discusso nel 1995 da Bruce Schneier [5]. Un server di marcatura temporale agisce facendo l’hash di un blocco di oggetti in modo che siano marcati temporalmente e poi lo pubblica, ad esempio su un quotidiano o in un post su Usenet. La marcatura temporale prova che i dati devono essere esistiti in quella determinata data, visto che sono finiti nell’hash. Ogni marcatura temporale comprende quella precedente nel suo hash, formando una catena. In questo modo si dà un ordine temporale alle transazioni per cui non ci possono essere doppie spese. Anziché basarsi su quotidiani o Usenet, in Bitcoin viene appunto utilizzata la PoW. Per quanto riguarda la sicurezza, essa è data dal fatto che una volta che un blocco viene inserito nella blockchain, per manometterlo escludendo o includendo certe transazioni (modificarle è impossibile senza invalidare le firme), occorrerebbe rifare la PoW di quel blocco e di tutti quelli successivi ad esso e poi raggiungere e superare il lavoro dei nodi onesti, tutte operazioni estremamente onerose. A questo proposito la PoW scoraggia i miner malintenzionati a barare per via delle scarse probabilità di successo e per gli ingenti costi di energia elettrica da impiegare. La PoW è utile anche per come è stata pensata in origine, ovvero come meccanismo anti-spam perché impedisce ai miner di inondare la rete con continue proposte di blocchi.

L’algoritmo di PoW utilizza la funzione di hash SHA-256 che prende in input l’hash dell’header del blocco candidato, concatenato a un numero variabile, il *nonce*. L’obiettivo di questa operazione è ottenere una stringa di output che abbia una certa quantità di zeri all’inizio, chiamata *target*. L’algoritmo usa l’approccio brute-force poiché per soddisfare il target, è necessario variare il nonce ripetutamente prima di individuare la soluzione; questa attività si chiama mining. La difficoltà del problema varia esponenzialmente con il numero di bit a zero richiesti ed è tale per cui un miner prima di riuscire a trovare la soluzione, impiega mediamente 10 minuti. Nel caso in cui la potenza di calcolo dei miner cresca e quindi si riduca il tempo di mining dei blocchi, il target viene aggiustato accrescendone la difficoltà e viene perciò ripristinato il tempo di 10 minuti per trovare la soluzione. In genere ciò accade ogni circa due settimane, precisamente ogni 2016 blocchi. La difficoltà del problema si misura in TeraHash al secondo e

negli anni ha subito notevoli balzi. Nei primi anni era possibile estrarre i blocchi in proprio con il semplice computer di casa, oggi invece è necessario utilizzare hardware specializzato e costoso e mettere in condivisione con altri utenti la propria potenza computazionale formando dei pool di mining. Il premio che si acquisisce in caso di vincita del pool, è proporzionato alla potenza messa a disposizione al pool durante la Proof of Work. Le macchine maggiormente utilizzate per fare mining sono ASIC (application specific integrated circuit), dispositivi specializzati che integrano la funzione di hash SHA-256 direttamente nei chip di silicone.

1.1.7 Fork

A causa dei tempi di latenza della rete, la blockchain, essendo decentralizzata, non sempre è consistente. Blocchi diversi potrebbero arrivare a nodi diversi in momenti diversi. In conseguenza di ciò possono verificarsi delle temporanee inconsistenze della blockchain chiamate fork che si risolvono dopo un certo numero di blocchi. Accade che due miner risolvono la PoW a distanza di poco tempo uno dall'altro. Una volta scoperta la soluzione del loro blocco candidato, i miner lo trasmettono ai loro vicini che a loro volta lo diffondono in rete. Un nodo che riceve un blocco valido X, lo aggiunge alla propria blockchain estendendola di una unità. Se poco dopo quel nodo riceve un altro blocco candidato Y che estende lo stesso parente di X, lo aggiunge ad una catena secondaria. Quello che accade è che per alcuni nodi nella catena principale c'è X e in quella secondaria Y, mentre per altri il viceversa e ciò comporta una disputa su quale delle due versioni della blockchain accettare. I miner che hanno ricevuto prima X, cominceranno a estrarre un blocco per estendere la loro catena principale; lo stesso discorso vale per quelli che hanno ricevuto prima Y. I miner di fatto stanno votando con la loro potenza di hashing la catena interessata. La competizione si conclude quando un gruppo di miner trova e trasmette un blocco in rete prima dell'altro gruppo e ciò, con buone probabilità, accade dopo un blocco. Di conseguenza, quella che per alcuni nodi era la catena secondaria, diventa la principale e si ottiene così una versione comune a tutti i nodi; le transazioni presenti nel blocco della catena sconfitta tornano disponibili nel memory pool. La formazione dei fork dipende essenzialmente da un fattore: la difficoltà della PoW. Maggiore è la difficoltà del problema da risolvere, maggiore sarà il tempo impiegato per trovare

la soluzione. Il tempo di mining di 10 minuti costituisce quindi un compromesso tra velocità di generazione di un blocco e probabilità di un fork. Se il tempo di mining fosse inferiore a 10 minuti, la probabilità di avere fork sarebbe maggiore in quanto i miner troverebbero la soluzione più rapidamente. In questo modo la rete dovrebbe gestire molti blocchi candidati. Viceversa se il tempo di mining fosse superiore a 10 minuti, la probabilità di avere fork diminuirebbe. Tuttavia, ciò comporterebbe un'attesa maggiore per gli utenti prima di vedere le loro transazioni finire nella blockchain.

Soft fork e Hard fork

Esiste un altro scenario in cui la rete può formare dei fork: un cambiamento delle regole di consenso. Può accadere che nel software che implementa il protocollo siano presenti dei bug oppure che si vogliano effettuare delle migliorie generali. Gli sviluppatori in questo caso possono proporre un aggiornamento tramite fork. Ci sono due tipologie di variazione del protocollo di consenso: *soft fork* e *hard fork*. Un soft fork avviene quando le modifiche apportate sono retro-compatibili, nel senso che non aggiungono nulla in più alle regole già esistenti, piuttosto generano delle restrizioni. Ne consegue che un blocco che si basa sulle nuove regole viene accettato anche dai nodi non aggiornati, mentre un blocco basato sulle vecchie regole viene rifiutato dai nodi aggiornati. Concretamente ciò che avviene è una temporanea biforcazione della blockchain che vedrà da una parte la catena con le vecchie regole e dall'altra quella con le nuove. Se la maggioranza dei nodi deciderà di considerare come catena principale la seconda, il protocollo effettuerà un aggiornamento. In figura 1.12 è rappresentata una situazione di soft fork. Per

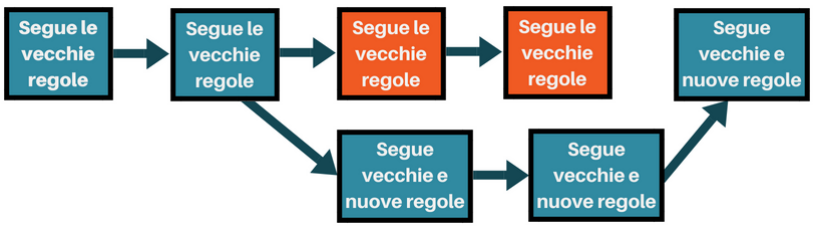

Figura 1.12: Soft fork

quanto riguarda un hard fork, le modifiche proposte non sono retro-compatibili e quindi necessariamente vengono a formarsi due catene principali indipendenti. I nodi che accettano le nuove regole, a differenza di un soft fork, non potranno più

accettare quelle vecchie. Questi tipi di fork sono molto meno frequenti rispetto ai soft fork e vengono effettuati quando si vogliono apportare modifiche importante alle regole del consenso. In figura 1.13 è rappresentata una situazione di hard fork. Un esempio di hard fork è Bitcoin Cash, verificatosi il primo agosto 2017 al

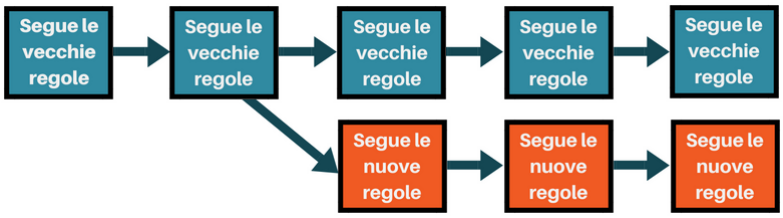

Figura 1.13: Hard fork

blocco 478558, e di cui si parlerà in seguito.

Conferme blocchi

Dire che un blocco ha un numero z di conferme equivale a dire che dopo di lui si susseguono z blocchi nella blockchain. Maggiore è z, più il blocco in questione è difficilmente modificabile. Il numero di conferme consigliato prima di considerare un blocco appartenente in modo definitivo alla blockchain, senza possibilità di fork, è sei (circa un'ora di tempo). I calcoli effettuati da Nakamoto nel suo paper per trovare questo risultato, fanno affidamento sulla teoria della probabilità. Si consideri lo scenario in cui un utente malintenzionato cerchi di generare una catena alternativa più veloce di quella onesta allo scopo di condurre o un attacco di doppia spesa o di Denial of Service. La probabilità P che l'utente malintenzionato possa raggiungere e superare la catena onesta partendo da z blocchi indietro varia in funzione di z e di un altro parametro, q, che rappresenta la probabilità che l'utente malintenzionato generi un blocco. Dai risultati ottenuti nella sezione 11 di [1], si osserva che ad esempio per q=0.1 e z=6, P=0.0002428. La probabilità di successo del malintenzionato è quindi irrisoria nel caso in cui abbia probabilità di generare un blocco del 10%. Per avere certezza di conseguire con successo un attacco di questo genere, un gruppo di miner collusi dovrebbe possedere la maggioranza della potenza totale di hash della rete. L'attacco in questione viene chiamato *51% Attack*. Uno scenario di questo genere è improbabile ma non impossibile. Attualmente esistono infatti dei server di pool di mining che ricoprono notevoli percentuali dell'hash-rate

della rete e che potrebbero sferrare potenzialmente simili attacchi. È anche vero, d'altra parte, che una manovra di questo tipo eroderebbe la fiducia in Bitcoin in breve tempo, causando probabilmente un significativo calo del suo valore. La figura 1.14 mostra la distribuzione della capacità di hash tra i principali gruppi di minatori.

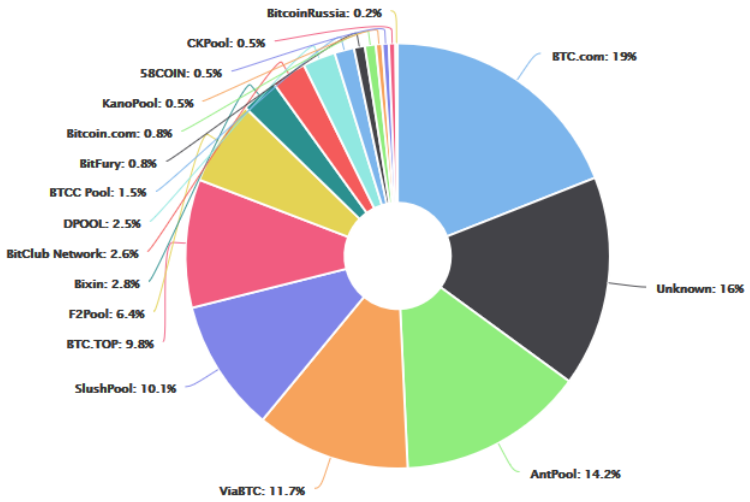

Figura 1.14: Distribuzione della capacità di hash, fonte Blockchain.com

1.1.8 Riflessioni su Bitcoin

La blockchain di Bitcoin, come si è potuto constatare dalle precedenti sezioni, è costituita da una struttura articolata e complessa. Le numerose proprietà che la caratterizzano fanno sì che presenti dei punti critici su cui è opportuno fare delle riflessioni. Le due proprietà principali che caratterizzano quest'innovativa tecnologia sono decentralizzazione e sicurezza. La prima è data dalla condivisione, tra i nodi trustless della rete, della stessa copia pubblica del registro delle transazioni e dal consenso ottenuto con la PoW. La sicurezza invece è data dalle firme digitali e dalla difficoltà con cui un malintenzionato può cambiare il passato, merito ancora una volta della PoW. Se da un lato l'algoritmo di consenso di Bitcoin è fondamentale per il suo funzionamento, dall'altro è estremamente inefficiente in termini di costo per via del suo approccio brute-force. Verranno presentate ora una serie di problematiche relative alla blockchain a partire proprio dai costi della PoW.

Consumi del mining

La potenza computazionale delle grandi farm di mining aumenta ogni giorno di più con il risultato che la difficoltà della PoW, al fine di rispettare il tempo di mining di 10 minuti, subisce allo stesso modo un incremento. Questo circolo vizioso porterà a consumi sempre più elevati nel tempo. Gli analisti stimano che l'estrazione di bitcoin arriverà a consumare più di 125 terawattora di elettricità entro un anno, pari allo 0,6% del consumo mondiale. Lo scorso sono stati consumati 36 terawatt di energia, tanto quanto uno stato come il Qatar. Le sei più grandi farm di mining al mondo, da Antpool a Btcc, sono tutte in Cina dove i costi energetici sono più economici rispetto a nazioni come il Regno Unito o gli Stati Uniti. Secondo il sito Bitcoinmagazine.com circa il 70% dei principali pool minerari bitcoin si trovano in Cina o sono di proprietà di società cinesi con un impatto rilevante in termini di emissioni nocive dovuto alla bassa qualità degli impianti di fornitura elettrica. Uno studio dell'università di Cambridge ha stimato che la Cina produrrebbe circa un quarto di tutta la potenza computazionale necessaria per generare criptovalute con conseguenze dannose sull'ecosistema. Bitmain Technologies gestisce la più grande server farm al mondo a Erdors, in Mongolia. È costituita da otto magazzini in metallo lunghi 100 metri con oltre 25.000 computer dedicati alla risoluzione dei calcoli di hash, che generano quasi il 4% della potenza di elaborazione nella rete bitcoin globale. Da questi dati è evidente come la PoW di Bitcoin risulti estremamente dispendiosa; nel paragrafo 1.2.3 verrà presentata la Proof of Stake, un algoritmo di consenso più ecologico.

Costo delle fee

Un'altra questione che merita una riflessione è il costo variabile delle fee. Attualmente il prezzo medio delle fee per transazione si aggira attorno ai 0,50€, un costo accettabile di commissione considerando quelli di svariate decine di euro per effettuare i bonifici esteri con le banche. Nel maggio del 2017 le fee medie sono salite di oltre un dollaro per la prima volta nella storia di Bitcoin. All'inizio di giugno toccavano in media 5,66$. Il tetto massimo è stato toccato nel dicembre del 2017 con la crescita del valore dei bitcoin e ha raggiunto cifre di quasi 60$ per transazione. In figura 1.15 è rappresentato l'andamento dell'ultimo anno. Il motivo dell'aumento del costo delle fee verso la fine del 2017 è da attestare all'alto

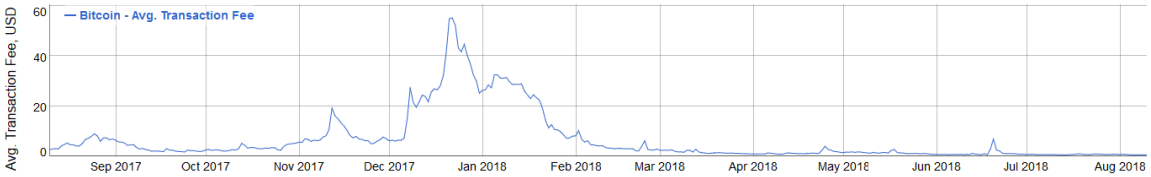

Figura 1.15: Costo medio delle fee per transazione, fonte Bitinfocharts.com

numero di transazioni scambiate in rete in quel periodo. Maggiore è il numero di transazioni che vanno a finire nel memory pool, più alta sarà la competizione che gli utenti faranno per vedersi confermare dai miner le transazioni. Questo si traduce in aumento delle fee per guadagnare priorità. Il memory pool in questi mesi si è invece ridotto notevolmente favorendo un abbassamento del costo delle fee. I motivi che hanno permesso questo risultato sono tre: il calo del volume delle transazioni, un accorgimento adottato dagli Exchange chiamato *batching* e una miglioria del software apportata dagli sviluppatori di Bitcoin chiamata *SegWit*. Il batching è un raggruppamento di più transazioni in una sola e ha lo scopo di aumentare il numero delle stesse per blocco, pagando così commissioni minori. Ad esempio 10 output aggregati nella stessa transazione occupano meno di 10 transazioni indipendenti. Questo implica meno competizione e fee più basse. SegWit (Segregate Witness) è un aggiornamento del formato della transazione avvenuto in seguito al soft fork del 24 agosto 2017. Lo scopo primario di SegWit era correggere un bug presente da sempre nel protocollo Bitcoin, la transaction malleability. Il bug consentiva a un malintenzionato di modificare il transaction id (Txid) di una transazione. Il Txid è l'hash ottenuto da un doppio SHA-256 di tutti i campi della transazione tra i quali compariva, prima di SegWit, anche l'unlocking script, detto Witness (sig e pubKey, vedi 1.1.2). Dal momento che i dati del Witness sono modificabili, un malintenzionato poteva condurre attacchi DoS facendo variare il Witness e ottenendo così diversi Txid; la transazione in questo modo rimaneva sintatticamente valida ma si presentava diversa da quella precedente. SegWit separa il Witness dalla transazione ponendolo al di sotto di essa, così che il Txid non si possa più calcolare su dati variabili e divenga immutabile. Con SegWit si è ottenuto anche un altro risultato. Essendo il Witness la parte che occupa più spazio in una transazione, circa i tre quarti, lo spostamento al di sotto di essa ha fatto sì che il blocco Bitcoin possa raggiungere il limite teorico massimo di 4 MB di transazioni pur mantenendo fisso a 1 MB il blocksize,

la parte che contiene i dati. Quindi facendo una transazione con SegWit le fee hanno un costo minore. Quest'affermazione giustifica il grafico 1.16 che mostra come le transazioni con SegWit siano in aumento. Riassumendo, la combinazione

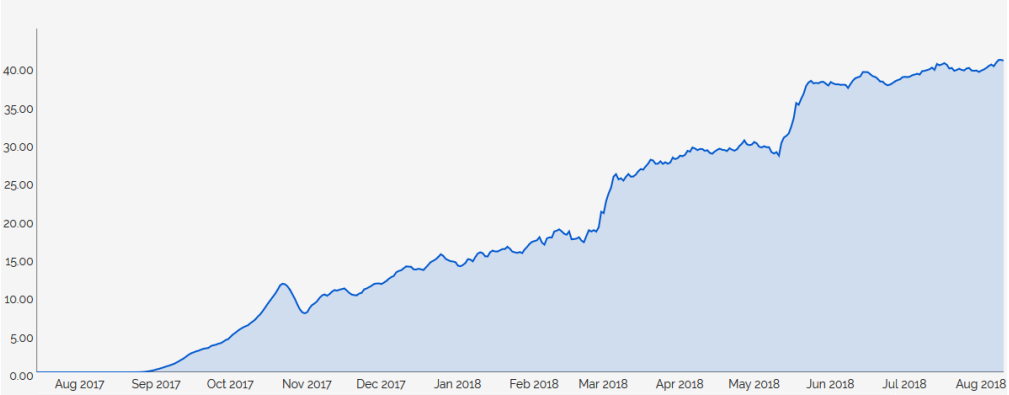

Figura 1.16: Percentuale di transazioni che usano SegWit, fonte Transaction-fee.info

di batching e SegWit oltre che a una diminuzione del volume generale delle transazioni degli ultimi mesi, ha portato ad un abbassamento del costo delle fee rispetto a fine 2017.

Scalabilità

La scalabilità è la capacità di un sistema di adattarsi a un aumento del carico di lavoro. Un incremento può riguardare, ad esempio, il numero delle richieste fatte dagli utenti al sistema oppure la mole di dati che il sistema deve gestire. Bitcoin presenta dei problemi di scalabilità che riguardano le transazioni. Un sistema di pagamento digitale come Visa è in grado di elaborare fino a 2000 transazioni al secondo, Bitcoin è molto distante da queste cifre. Ciò dipende principalmente da due fattori: tempo di mining e dimensione del blocco. Il tempo di mining è fissato a 10 minuti e la dimensione del blocco a 1 MB; sapendo che una transazione in media è di circa 225 byte, risulta che la rete Bitcoin può sostenere massimo circa 7 transazioni al secondo. Quindi al crescere delle transazioni effettuate dagli utenti, la velocità con cui vengono confermate non aumenta. Come visto nelle sezioni precedenti, il tempo di mining è un trade-off tra velocità di generazione di un blocco e la probabilità di un fork. Abbassando la quota di 10 minuti si otterrebbe una velocità delle transazioni maggiore a discapito però di un'inconsistenza della rete più alta dovuta ai numerosi fork e ciò comporterebbe anche più vulnerabilità

agli attacchi di doppia spesa e DoS. Questo è un risultato che la comunità di Bitcoin non intende conseguire. Per quanto riguarda invece la questione della dimensione del blocco, bisogna fare delle osservazioni. Aumentare il blocco comporta due problemi: maggior latenza nella trasmissione in rete e aumento della dimensione della blockchain. Il primo problema è dovuto alla larghezza di banda e al tempo di validazione del blocco. Blocchi più pesanti verrebbero trasmessi con maggiore difficoltà in rete e ciò potrebbe causare la formazione di fork, problema analogo all'abbassamento del tempo di mining. Abbassare il tempo di mining o rallentare la velocità di propagazione dei dati sono due problematiche strettamente collegate. Il primo agosto 2017 però, la rete di Bitcoin si è divisa con un hard fork che ha portato alla nascita di Bitcoin Cash. Con questo fork si è ottenuto l'ampliamento della dimensione del blocco che è passato da 1 MB a 8 MB. L'aumento della dimensione della blockchain fa sì inoltre che molti meno nodi possano permettersi di scaricarla tutta e agire da full node e ciò causerebbe un problema di centralizzazione. Una soluzione "parziale" al problema della crescente capienza della blockchain era stata proposta già da Nakamoto e implementata nel software ufficiale Bitcoin Core: il *pruning*. Un nodo dopo aver scaricato tutta la blockchain e formato il pool delle UTXO, può decidere di rilasciare la blockchain salvando gli header block e le UTXO. Essendo la dimensione dell'header di un blocco di circa 80 bytes e sapendo che un blocco viene generato ogni 10 minuti, in un anno si accumulerebbero solo 80 bytes * 6 * 24 * 365 = 4.2 MB di spazio. È una soluzione parziale perché un full node deve necessariamente scaricare tutta la blockchain prima di fare il pruning per essere sicuro di ottenere il corretto pool di UTXO, evitando quindi di fidarsi di un nodo inviante. La proposta che ha riscosso maggior successo per far fronte al problema della scalabilità di Bitcoin è Lightning Network.

Payment Channel e Lightning Network

L'idea della Lightning Network è stata proposta per la prima volta da Joseph Poon e Thadeus Dryja nel febbraio del 2015 e si basava sui *canali di pagamento* ovvero canali trusteless al di fuori della blockchain (off-chain) fra due utenti che instauravano un rapporto continuativo di pagamenti. Lo scopo del canale di pagamento quindi è permettere lo scambio di un alto numero di transazioni tra due utenti in modo veloce. Per ottenere questo risultato è necessario agire

al di fuori della blockchain, evitando la lenta procedura di conferma dei miner, e registrare al suo interno solamente lo stato iniziale del rapporto di scambio e quello finale. Questi due stati si chiamano: *funding transaction* F (transazione iniziale) e *settlement transaction* S (transazione finale). Le transazioni intermedie off-chain si chiamano *commitment transaction* C. Si supponga che Alice voglia usufruire di un servizio di film in streaming gestito da Bob. Il canale è instaurato dall'interazione tra il software presente nel browser di Alice e quello nel server di Bob ed entrambi i software implementano le funzionalità di un wallet. Ogni secondo di video costa 0,0000002 BTC quindi se Alice volesse vedere un film della durata di un'ora e mezza dovrebbe pagare 0,001 BTC (circa 6€). Alice crea una F di 0,001 BTC verso un indirizzo a doppia firma (2-2 multisig) condiviso tra Alice e Bob, il cui output è riscattabile perciò con le chiavi private di entrambi. Una volta che F è confermata, Alice può cominciare a vedere il video. Dopo un secondo di video, il wallet di Alice crea e firma una C che parte da F e manda 0.0000002 BTC a un indirizzo di Bob e 0,0009998 ad Alice come resto. Quando anche Bob mette la seconda firma a F, il server concede la visione di un secondo ad Alice. Al successivo secondo di video, Alice crea e firma un'altra C che parte da F e manda 0.0000004 BTC a Bob e 0,0009994 BTC a se stessa. Bob firma e concede un altro secondo, e così via. Questo scambio avviene off-chain. Quando Alice interrompe la visione, sia lei che Bob possono trasmettere alla blokchain l'ultima transazione fatta: S. I problemi di questa implementazione sono due: Bob potrebbe sparire bloccando i soldi di Alice in F (ci vogliono entrambe le firme per sbloccarli); Alice potrebbe trasmettere alla blockchain la prima C anziché l'ultima con il risultato che pagherebbe solo 0.0000002 BTC per un'ora di film. Questi problemi si possono risolvere con un contatore chiamato *nLocktime*. Prima di trasmettere F, Alice si fa anticipatamente firmare da Bob la prima C, inserendovi un nLocktime di un certo numero di blocchi (300 blocchi corrispondono a 60 ore) al termine del quale i soldi di F tornano ad Alice. Dopo che Bob firma C, Alice può trasmettere F alla blockchain e cominciare uno scambio di C off-chain. Ogni successiva C avrà un nLocktime minore di quella precedente. Vengono così risolti i due problemi di cui sopra: se Bob sparisce, dopo un certo nLocktime Alice recupera i soldi; Alice non potrà trasmettere una C a suo piacimento poiché l'ultima C avrà un contatore più basso di tutte le altre e quindi sarà trasmissibile prima di esse. Tuttavia la soluzione al primo problema costringe Alice, nel caso Bob sparisca, ad

aspettare nLocktime prima di riavere i propri bitcoin e potrebbe essere un tempo molto lungo. Inoltre il contatore decrescente stabilisce un numero massimo di transazioni che si possono effettuare. Per risolvere questi problemi sono stati introdotti due espedienti: transazioni simmetriche e *revocation key* R. La R è una chiave segreta che ha la funzione di riscattare tutto il contenuto di F nel momento in cui uno dei due partecipanti tenta di imbrogliare trasmettendo una C conveniente. Si supponga che Alice e Bob creino una F inviando ciascuno 5 BTC, per un totale di 10 BTC. Ora, anziché creare una singola C, ne vengono create due simmetriche, una da parte di Alice e una da parte di Bob. Alice ne crea una che Bob dovrà firmare e viceversa. Le due C si presentano come in figura 1.17, C1A è quella creata da Alice, C1B quella creata da Bob. C1A è costruita in modo

Figura 1.17: Transtaction commitment simmetriche

tale per cui nel momento in cui Alice la firmerà e la trasmetterà alla blockchain, 5 BTC presenti in F torneranno subito a disposizione di Bob e i restanti 5 BTC presenti in F andranno ad Alice dopo 1000 blocchi oppure a chi esibisce R1A. La chiave segreta R1A è creata da Alice e verrà comunicata a Bob prima di un avanzamento di stato, ovvero prima di creare C2A e C2B. Si supponga quindi che Alice comunichi R1A a Bob e poi gli mandi 2 BTC per pagare un certo servizio, come rappresentato in figura 1.18. Se a questo punto Alice volesse trasmettere

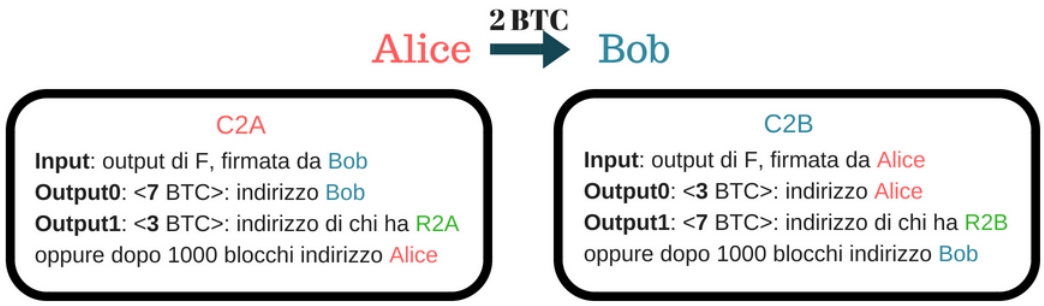

Figura 1.18: Fase successiva e nuove transtaction commitment

alla blockchain C1A perché più conveniente per lei, avendo Bob a disposizione

R1A, avrebbe tempo 1000 blocchi per anticipare Alice e riscattare anche i suoi 5 BTC! Questo meccanismo assicura che nessuno possa trasmettere una transazione passata, pena la perdita dei propri fondi.
Lightning Network (LN) a differenza dei canali di pagamento che avvengono solo tra due nodi, permette scambi veloci tra una rete di nodi; è una routed payment channel. La revocation key in LN è sostituita dagli *Hash Time Lock Contracts* (HTLC), un meccanismo che consente ai partecipanti di riscattare fondi bloccati o attraverso un codice segreto o allo scadere di un contatore. Il destinatario di un pagamento crea un segreto R e invia l'hash H = Hash(R) al mittente. Il mittente utilizza H per creare uno script chiamato HTLC, in cui inserisce della moneta riscattabile solo da chi possiede il valore x tale per cui Hash(x) = H, quindi R. In questo modo solo il destinatario può riscattare la moneta. HTLC implementa anche un timeout che fa tornare la moneta al mittente se non viene riscattata in tempo. Per capire il funzionamento della LN si farà ora un esempio. Una rete è composta da 5 partecipanti: Alice, Bob, Carol, Diana ed Eric. Ognuno di essi ha aperto un canale di pagamento con il vicino in cui sono stati depositati 2 BTC da parte di ogni partecipante, quindi 4 BTC per canale. In figura sono illustrati i passi necessari per consentire ad Alice di trasferire 1 BTC ad Eric.

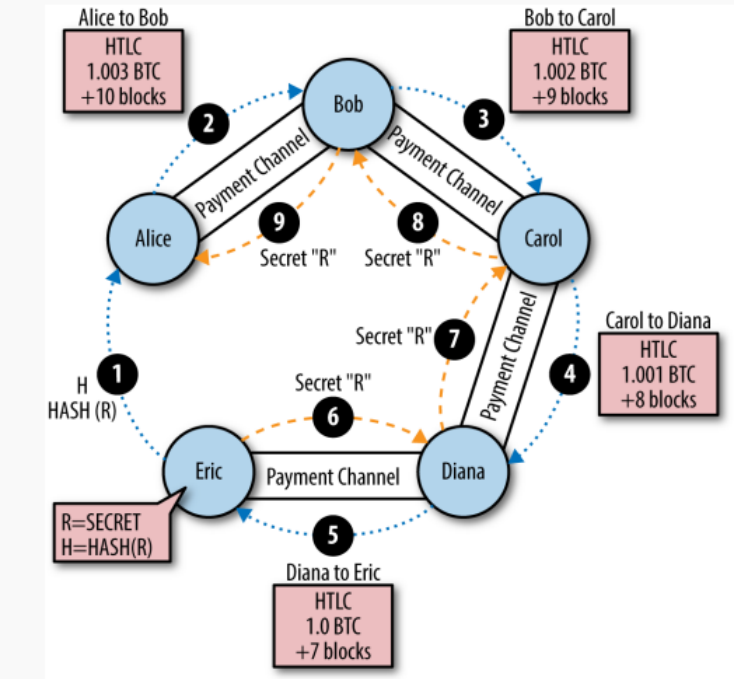

Figura 1.19: Instradamento dei pagamenti attraverso Lightning Network, fonte Mastering Bitcoin [2]

1. Eric crea un segreto R, ne calcola l'hash H e spedisce H ad Alice;
2. Alice crea un HTLC relativo all'hash H con un timeout di 10 blocchi e un valore di 1.003 BTC e lo invia a Bob; il valore di 0.003 sono le fee per gli intermediari;
3. Bob crea un HTLC relativo all'hash H con un timeout di 9 blocchi e un valore di 1.002 BTC e lo invia a Carol;
4. Carol crea un HTLC relativo all'hash H con un timeout di 8 blocchi e un valore di 1.001 BTC e lo invia a Diana;
5. Diana crea un HTLC relativo all'hash H con un timeout di 7 blocchi e un valore di 1.000 BTC e lo invia a Eric;
6. Eric invia il segreto R a Diana per dimostrargli che può riscattare lo script HTLC e quindi ricevere 1 BTC;
7. Ora anche Diana conosce il segreto R e lo usa per riscattare HTLC inviatole da Carol guadagnando 0.001 BTC;
8. Carol riscatta l'HTLC inviatole da Bob guadagnando 0.001 BTC;
9. Bob riscatta l'HTLC inviatogli da Alice guadagnando 0.001 BTC. Oltre alla scalabilità delle transazioni e a costi di commissioni più bassi, Lighting Network consente di ottenere un buon livello di privacy, garantita dal protocollo di onion-routing. Un nodo, quando fa da tramite per un pagamento infatti, non ha altre informazioni ad eccezione di quelle sul nodo precedente e su quello successivo.

1.2 Ethereum

La seconda blockchain ad aver riscosso un grande successo dopo Bitcoin è senza dubbio Ethereum. È nata nel 2013 [6] come alternativa a Bitcoin, proponendo una sostanziale differenza nel linguaggio utilizzato. In Bitcoin, come già constatato, è presente un linguaggio di scripting che consente di fare un numero limitato di operazioni. La novità di Ethereum risiede nella possibilità di costruire applicazioni decentralizzate grazie al proprio linguaggio Turing completo. Tali applicazioni

prendono il nome di *smart contract*. Ad introdurre per primo il concetto di smart conctract fu l'ingegnere informatico Nick Szabo nel 1994 [7] che lo definì come un protocollo di transazione informatizzato che esegue i termini di un contratto. Con il termine "smart" si intende che il contratto in questione, inteso come una qualsiasi azione, è auto-eseguibile al verificarsi di certe condizioni. Il codice utilizzato per scrivere le applicazioni appartiene ad un linguaggio ad alto livello: Solidity. Per evitare che i programmi abbiano loop infiniti di esecuzione del codice, gli sviluppatori hanno implementato un meccanismo di esaurimento delle risorse. La risorsa in questione è chiamata *gas* e agisce come se fosse il carburante del contratto. Prima di approfondire tale meccanismo, verrà introdotto il modello con cui vengono registrati i record nella blockchain.

1.2.1 Modello degli Account

In Bitcoin ciascuna transazione spende gli output delle transazioni precedenti generando nuovi output, UTXO, che possono essere spesi in transazioni future. La somma posseduta da un utente è calcolata dal suo wallet aggregando tutti gli output non spesi delle transazioni riferite al suo indirizzo. Il modello degli Account di Ethereum invece aggiorna di volta in volta lo stato degli account degli utenti, riportando il valore di Ether spendibili dallo stesso. Gli account si possono suddividere in due tipologie: *externally owned accounts* e *contract accounts*. I primi sono controllati da chi possiede la chiave privata associata all'indirizzo pubblico; i secondi sono governati dal codice interno, quello che caratterizza lo smart contract. Una volta compilato, il codice di un contratto viene convertito dalla *EVM* (Ethereum Virtual Machine), una macchina virtuale radicata all'interno del software di Ethereum e in esecuzione nei full node, in una sequenza di operazioni codice chiamate opcodes come ad esempio ADD (addizione), MUL (moltiplicazione) e altre. Ad ogni operazione è associato un costo in gas. Ad esempio ADD richiede 3 gas, SHA256 ne richiede 60 e così via; la lista completa è nello yellowpaper di Ethereum [8]. Il codice di uno smart contract può essere innescato in seguito a una transazione che viene fatta verso il suo indirizzo pubblico da un account esterno o da un altro smart contract. Chi crea la transazione deve pagare un quantitativo di gas dato dalla somma delle operazioni presenti nel contratto; il gas quindi ha la duplice funzione di essere un meccanismo anti-loop e una fee. Il gas viene tradotto in ether, la moneta ufficiale

di Ethereum, ed il quantitativo minimo che bisogna pagare per una transazione qualsiasi è di 21000 gas. Quando si effettua una transazione è possibile impostare il *LimitGas*, il quantitativo massimo di gas che si intende pagare. Se il valore impostato non è sufficiente, si perde il gas, viceversa, se la quantità è superiore, il gas in eccesso viene restituito. Altro parametro impostabile è il *gasPrice* che indica il prezzo che si intende pagare per ogni singola unità di gas. L'unità di misura del gasPrice è il Giga Wei che corrisponde a un miliardesimo di Ether. Analogamente a Bitcoin, anche per Ethereum ci sono servizi di terze parti che forniscono una stima del valore delle commissioni in base al tempo che si intende aspettare prima che una transazione venga inserita nella blockchain. Ad esempio attualmente il sito Ethgasstation.info calcola che pagando 17 Gwei/gas per 21000 gas, ovvero 0.00036 ETH, quindi 0.131$, una transazione finisce nella blockchain in circa un minuto. Ethereum, allo stesso modo di Bitcoin, impiega come algoritmo di consenso la PoW. Il tempo di mining è notevolmente più basso, mediamente un blocco viene estratto ogni 15 secondi, con il risultato che le transazioni degli utenti vanno a buon fine in tempi molto più rapidi. Con un tempo di mining così basso però i fork rischierebbero di compromettere la sicurezza della rete, per cui gli sviluppatori hanno pensato di ricorrere al protocollo GHOST, proposto per la prima volta da due sviluppatori di Bitcoin nel 2013, Yonatan Sompolinsky e Aviv Zohar [9].

1.2.2 GHOST

Prima di introdurre GHOST è bene precisare che il protocollo attuale di Ethereum è in continuo divenire e privo di una documentazione sistematica per cui il suddetto protocollo verrà presentato come inizialmente definito.
Lo scopo di GHOST (Greedy Heaviest Observed Subtree) è far fronte alle problematiche conseguenti a un tempo di mining molto basso come quello di Ethereum. A causa della latenza di trasmissione dei blocchi nella rete, se un miner A trova un blocco e successivamente un miner B ne trova un altro prima che quello di A sia trasmesso a B, il blocco di B andrà sprecato e non contribuirà alla sicurezza della rete. Si avrebbe inoltre un problema di centralizzazione. Si supponga che il pool di mining A abbia una potenza di hash del 40% e il pool B una del 15%. A avrà un rischio di produrre un blocco ritardatario pari al 60% delle volte (durante l'altro 40% del tempo è A che produce blocchi) mentre B lo avrà per 85% dei casi,

rendendo sconveniente la competizione. In questo modo la blockchain correrebbe il rischio di subire una centralizzazione verso il pool A. GHOST risolve il primo dei due problemi, la perdita di sicurezza, tramite l'inclusione dei blocchi ritardatari nel calcolo di quella catena che presenta la maggior potenza computazionale. Perciò, nel calcolo, non vengono considerati solo i parenti e gli antenati del nuovo blocco estratto, come accade in Bitcoin, ma anche i discendenti dei blocchi "orfani" degli antenati del nuovo blocco estratto che in gergo si chiamano *uncles* (zii). In figura 1.20 sono rappresentati i metodi con cui viene scelta la catena principale in Bitcoin e in Ethereum. Per quanto riguarda Bitcoin, l'arrivo del blocco B3 fa sì che il ramo sottostante diventi la catena più lunga e quindi quella principale. Nel caso di Ethereum invece, grazie a C2, prodotto in ritardo rispetto al fratello B2, si ha che la catena principale sia quella sottostante poiché la più pesante (Heaviest). Per risolvere il secondo problema, quello relativo alla centralizzazione, vengono

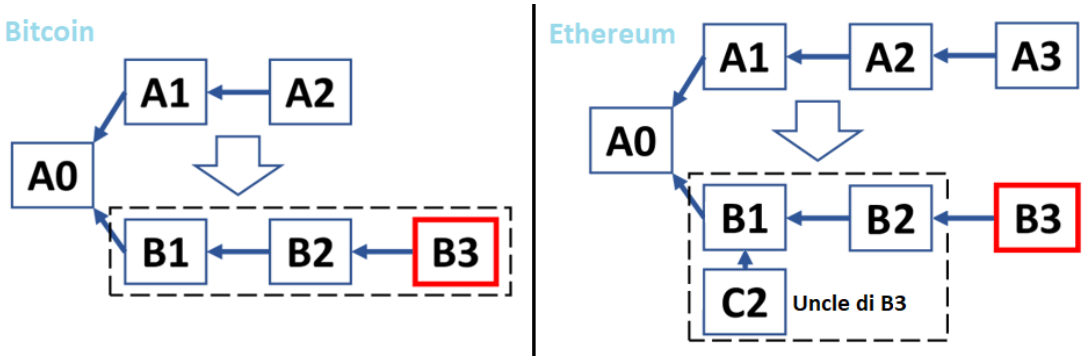

Figura 1.20: Catena principale Bitcoin/Ethereum

forniti reward ai miner anche per gli uncle. Chi estrae un blocco uncle riceve infatti 7/8 della block reward che in Ethereum è costante nel tempo e pari a 5 ETH. Chi estrae il nipote del blocco uncle invece riceve 1/8 della block reward. Le fee degli uncle e dei nipoti però non vengono assegnate ai miner. Un uncle può essere discendente, fino alla settima generazione, di uno dei suoi fratelli diretti. GHOST di Ethereum si ferma alla settimana generazione per ragioni di semplicità di calcolo delle reward. Concedendo reward per gli uncle, si incoraggiano anche i piccoli miner a contribuire alla sicurezza del sistema, evitando di centralizzare il potere. Ethereum implementa una versione semplificata di GHOST che prevede che:

- Un blocco debba specificare un genitore e zero o più uncle

- Un uncle incluso nel blocco B debba avere le seguenti proprietà:
  - essere un figlio diretto di un antenato di K-esima generazione di B, dove 2 <= k <= 7
  - non possa essere un antenato di B
  - avere un header block valido, ma non essere necessariamente stato verificato o validato in precedenza
  - essere diverso da tutti gli uncle inclusi nei blocchi precedenti o nello stesso blocco (nessuna doppia inclusione)
- Per ogni uncle U nel blocco B, il minatore di B ottenga un ulteriore 3,15% in aggiunta alla sua reward e il miner di U ottenga il 93,75% di una block reward base.

1.2.3 Proof of Stake

Ethereum attualmente utilizza come algoritmo di consenso la Proof of Work. I miner prima di risolvere il puzzle matematico devono impiegare molte risorse in termini di energia elettrica per effettuare i calcoli. Gli sviluppatori di Ethereum per far fronte a questo dispendioso meccanismo di consenso hanno pensato di adottare un nuovo algoritmo: la Proof Of Stake (PoS). Viene proposta per la prima volta nel 2011 nel famoso forum Bitcointalk da un utente. L'idea di base è sostituire il meccanismo di voto dei blocchi: con la PoW i miner esprimono il loro voto con la potenza di hashing; con la PoS invece i validatori, anziché i miner, votano mettendo in palio (stake) parte della moneta posseduta. Il processo di scelta del validatore non può basarsi solamente sulla quantità di moneta messa in palio altrimenti solo gli utenti più ricchi potrebbero validare blocchi e guadagnare le fee. Sono stati proposti vari metodi nel tempo per far fronte a questo problema. *Peercoin* si basa su PoS e utilizza il concetto di *coin age* come metodo di selezione. La coin age è numero ottenuto dal prodotto tra la moneta posseduta e l'anzianità della moneta ovvero il tempo in cui la si detiene senza che sia stata spesa, periodo che deve essere di almeno 30 giorni. Una volta che un soggetto spende la moneta, la coin age va a zero ed è necessario attendere 30 giorni prima di metterla in palio. La probabilità di creare un blocco quindi aumenta con la coin age. Per evitare che una moneta molto anziana possa dominare la rete, il tetto massimo di

giorni di anzianità viene fissato a 90. Peercoin usa una combinazione di PoW e PoS per la creazione dei blocchi. La PoS nei blocchi si manifesta con una speciale transazione chiamata *coinstake*. In maniera analoga alla coinbase transaction, nella coinstake il nodo indirizza il pagamento a se stesso della propria coin age consumata, acquistando così il diritto di generare e ricevere un compenso. Il primo input di tale transazione è chiamato *kernel* e contiene un hash che deve soddisfare un certo target. Maggiore è il coin age, meno impegnativo è il target da soddisfare. I target in Peercoin richiedono meno lavoro prima di essere soddisfatti il che rende Peercoin meno inefficiente di Bitcoin in termini di consumi. In caso di fork, la catena principale è scelta in base al punteggio dei blocchi ottenuto dalla somma delle coin age presenti in ciascuno di essi. Si è visto che Bitcoin considera come catena principale quella in cui viene fatto il maggior lavoro computazionale e per questa ragione, potrebbe teoricamente essere soggetto ad attacchi del 51%. Peercoin, secondo gli sviluppatori, è meno a rischio da questo punto di vista perché sferrare un attacco del 51% vorrebbe dire possedere più di metà delle monete in circolazione, il cui valore sarebbe più elevato del costo di una potenza di mining del 51%. Il problema maggiore secondo i critici della PoS è quello del cosiddetto *nothing at stake* (nessuna posta in gioco). Nel caso di fork della blockchain un nodo potrebbe votare per entrambe le varianti, perché ha delle poste in gioco in ciascuna di esse. In Peercoin infatti questa procedura non comporta perdite, perché se la somma messa in gioco riguarda un blocco orfano, la stake non viene accettata. In questo modo si potrebbe verificare un attacco di doppia spesa all'1%. Si supponga che tutti i validatori votino contemporaneamente per entrambe le catene, basterebbe che un utente possieda l'1% delle monete per sferrare un attacco di doppia spesa sulla catena malevola. La contromisura di Peercoin consiste in un checkpoint di un blocco prima del quale la blockchain non è modificabile. A impostarlo però sono gli sviluppatori stessi e ciò comporta una centralizzazione del sistema.
Ethereum a breve adotterà *Casper*, un nuovo protocollo basato su PoS. Il problema del nothing at stake viene affrontato con il concetto di deposito. I validatori per poter firmare i blocchi devono mandare il proprio stake allo smart contract di Casper. Successivamente possono votare una certa catena inviando transazioni speciali. Se vengono identificati due voti dello stesso validatore per due blocchi diversi che si trovano alla stessa altezza, il contratto di Casper lo punisce tagliando

parte dello stake presente nel deposito. Inoltre anche in Casper come in Peercoin verranno implementati i checkpoint per garantire che la storia non possa essere modificata. A differenza di Peercoin, però, i checkpoint vengono votati dai validatori. Il protocollo è implementato secondo il modello dei Generali Bizantini, per cui non è possibile che vengano accettati due checkpoint contemporaneamente su catene diverse senza che un terzo o più dei validatori sia disonesto. Se un validatore si comporta in modo disonesto, la prova della violazione può essere inclusa nella blockchain sotto forma di transazione; a quel punto l'intero deposito del validatore viene sequestrato e una piccola "commissione di ricerca" fornita al soggetto che ha scoperto il tentativo di frode.

1.3 Tipologie di blockchain

Le blockchain viste sinora si presentano come dei libri mastri di transazioni accorpate in blocchi distribuiti in una rete di nodi paritari. I partecipanti alla rete, non avendo barriere all'entrata, possono eseguire transazioni, creare blocchi e, se agiscono da full node, validarli. Questa tipologia di blockchain prende il nome di *Permissionless*, termine che si traduce in italiano con "senza permessi", e implica appunto la libertà di esecuzione, creazione e validazione da parte dei nodi; Bitcoin ed Ethereum ne rappresentano due esempi. Esiste un'altra tipologia di blockchain in cui invece l'ingresso ai nodi entranti è limitato e quindi gli stessi nodi possono partecipare alla rete solo in in seguito alla concessione di un permesso da parte degli amministratori: la *Permissioned* blockchain. Questa seconda specie di blockchain è in genere utilizzata da un'organizzazione o da un loro consorzio per scambiare informazioni in maniera privata e più efficiente rispetto all'altra soluzione. Una blockchain oltre ad avere o meno barriere all'ingresso per i partecipanti, può essere di due tipi: *pubblica* o *privata*. Quando è pubblica, i nodi possono consultarla senza necessariamente essere parte attiva della rete, ovvero senza effettuare transazioni o creare blocchi. Nel caso di Bitcoin ad esempio, come visto in 1.1.4, chiunque può esplorare la blockchain blocco per blocco tramite un blockchain explorer. Nel caso invece sia privata, possono consultarla solo i nodi che hanno il permesso d'accesso. Le blockchain permissionless sono pubbliche per definizione mentre le permissioned possono essere sia pubbliche che private ma in genere prevale la seconda specie. Hyperledger Fabric, un'iniziativa della

IBM, è un esempio di private permissioned blockchain e verrà trattata in dettaglio nel capitolo successivo. Un'altra caratteristica fondamentale che contraddistingue una permissioned da una permissionless riguarda il protocollo di consenso. La Proof of Work di Bitcoin usa un algoritmo costoso e appositamente impegnativo per raggiungere il consenso in rete e ottenere decentralizzazione e sicurezza. In un sistema aperto e trustless, non essendovi fiducia tra i partecipanti, il processo di elezione del nodo creatore deve essere competitivo e la gara matematica della PoW a questo proposito è una soluzione efficace a questo proposito. Il prezzo da pagare però sono le basse prestazioni del sistema. Nelle permissioned invece non c'è necessità di escogitare un meccanismo artificioso per eleggere il nodo che dovrà formare un nuovo blocco, in quanto i nodi validatori e creatori vengono scelti dagli amministratori. Perciò, quando di conseguenza gli utenti scambiano transazioni, si affidano a questi nodi fidati, i trusted peer, che raggiungono il consenso utilizzando algoritmi più efficienti di quelli usati per le permissionless blockchain. Questo comporta un miglioramento delle prestazioni in termini di velocità delle transazioni. La fiducia che è necessario riporre nei trusted peer determina anche una decentralizzazione meno naturale di quella ottenibile con una permissionless blockchain. In un contesto trustless, la distribuzione del libro mastro tra i nodi della rete P2P fa sì che quest'ultimo sia decentralizzato e privo di un'autorità che lo gestisca. In un contesto trusted il libro mastro talvolta non è affidato a tutti i partecipanti, ma a una ristretta cerchia di nodi fidati selezionata dagli amministratori; questo fa sì che la decentralizzazione del libro mastro di fatto consista in realtà in una distribuzione dello stesso tra pochi e controllati attori. Alcuni studiosi della blockchain, tra cui il professor Arvind Narayanan di Stanford in un suo articolo del 2015 [10], ritengono che le blockchain private siano dei semplici database distribuiti in quanto, essendovi pochi validatori che fanno parte della rete, la PoW non è necessaria e quindi la decentralizzazione e la sicurezza che si ottengono con essa perdono di valore. Per quanto riguarda la sicurezza, non essendovi la PoW, in un sistema privato è data dalla fiducia che si ripone nei nodi validatori e quindi non ci sono garanzie matematiche circa l'irreversibilità delle transazioni. Questi aspetti verranno discussi in modo più approfondito nel capitolo successivo.

Capitolo 2

Caso d'uso: sistema sanitario

Il successo che ha riscosso Bitcoin dal 2009 ad oggi ha portato alla nascita di numerosi altri progetti basati su tecnologia blockchain. Il codice open source di Bitcoin è stato scaricato e modificato varie volte per creare sistemi che potessero scambiare e mettere al sicuro vari tipi di informazioni. Recentemente, ingegneri e aziende hanno indagato come questa tecnologia possa essere applicata a casi d'uso non finanziario, tra cui le certificazioni di identità, la gestione della supply chain e la gestione del materiale sanitario. In questo capitolo si presenteranno in un primo momento alcuni problemi legati al settore sanitario italiano in campo informatico, poi, come possibile soluzione, un progetto partito all'inizio di febbraio 2018, basato su una permissioned private blockchain. Infine verrà proposta un'implementazione alternativa alla blockchain e si farà un'analisi critica sui presunti vantaggi che si otterrebbero con una soluzione di tipo blockchain piuttosto che con una del secondo tipo.

2.1 Carenze del sistema informatico sanitario

Il servizio sanitario nazionale (in acronimo SSN), nell'ordinamento giuridico italiano, identifica il complesso delle funzioni, delle attività e dei servizi assistenziali gestiti ed erogati dallo Stato italiano [Wiki]. La macchina della sanità in Italia muove circa 140 miliardi di euro, pari al 9% del PIL nazionale, dati che mettono in risalto l'importanza del settore e quindi quanto sia importante per lo Stato che funzioni in maniera efficiente in termini di costi e di tempistiche nell'erogazione dei servizi. In un recente convegno di presentazione dei dati dell'Osservatorio

Innovazione Digitale in Sanità del Politecnico di Milano, il responsabile scientifico, Mariano Corso, ha affermato che il livello di qualità dei servizi sanitari sia già in declino e non debba essere ulteriormente peggiorato. Secondo l'ultimo rapporto Euro Health Consumer Index, l'Italia è passata tra il 2010 e il 2017 dal 14° al 21° posto delle 35 nazioni censite a livello europeo, rispetto alle performance del sistema sanitario ed è tra i peggiori paesi per accessibilità ai servizi e tempi di attesa, gestione dei pazienti anziani sul territorio e possibilità di offrire cure di nuova generazione. Il 2° Rapporto sulla sostenibilità del Servizio Sanitario Nazionale – GIMBE stima che la spesa del servizio sanitario nazionale raggiungerà i 210 miliardi nel 2025, con ulteriori 60 miliardi a carico delle famiglie per coprire il fabbisogno di cure. C'è il rischio che molte persone si troveranno nell'impossibilità di permettersi le cure, con il conseguente aggravarsi di diseguaglianze e degrado sociale in un Paese dove, già oggi, oltre il 20% della popolazione ha difficoltà da elevata a moderata nel potervi accedere (dati Eurostat). Se il divario tra risorse disponibili e bisogni è quindi destinato ad aumentare, mettendo a rischio la sostenibilità del sistema sanitario e la qualità delle cure, l'innovazione digitale può essere la strada per far fronte a questi problemi. "Il sistema non funzionerà se il cittadino non assume un ruolo attivo, diventando uno degli attori del coordinamento della propria salute" afferma Corso e in questo il digitale può assumere un ruolo cruciale consentendo l'accesso a informazioni autorevoli attraverso Internet, l'accesso ai propri dati clinici tramite il *Fascicolo Sanitario Elettronico* (FSE), il monitoraggio del proprio stile di vita con le App e una continua comunicazione con il proprio medico anche tramite strumenti digitali. Attraverso un'indagine svolta in collaborazione con Doxapharma su 2.030 cittadini italiani, l'Osservatorio ha stimato che per il ritiro dei referti clinici, i cittadini che si recano di persona presso la struttura sanitaria sono circa il 60% della popolazione (metà della quale per conto di altre persone) e che il tempo medio per ritirare il referto è pari a 45 minuti. Ebbene, se questi cittadini ritirassero i referti in farmacia, il tempo si ridurrebbe a 20 minuti e se ciò avvenisse con lo scaricamento del documento via Web, si ipotizza un tempo di 5 minuti. Se la metà di quel 60% ritirasse online i referti, il 25% presso farmacia e un altro 25% lo facesse di persona, l'impatto economico sarebbe di 1120 milioni euro, nell'ipotesi che ciascun cittadino che accede al servizio effettuasse cinque ritiri all'anno. Se l'80% dovesse scegliere il ritiro online, il 10% in farmacia e il 10% di persona, arriveremmo a

1630 milioni di euro. Facendo simili stime per quanto riguarda gli altri servizi, l'Osservatorio ipotizza complessivamente risparmi di 150 milioni di euro per l'accesso a informazioni su prestazioni e strutture sanitarie, 430 milioni di euro per la prenotazione di visite ed esami e 980 milioni di euro per il pagamento di visite ed esami. Da questo punto di vista l'assistenza sanitaria è ancora arretrata. I sistemi informatici sono obsoleti, onerosi, lenti, spesso vulnerabili e danno scarso peso al paziente. I dati sulla salute associati a questi sistemi sono difficili da condividere tra le varie realtà ospedaliere a causa di diversi formati e standard. L'attuale panorama dei dati sanitari è frammentato e poco adatto alle esigenze istantanee degli utenti e ciò va ad inficiare anche la qualità delle cure. I medici si basano sui test per diagnosticare le malattie del paziente e successivamente elaborare un possibile piano di trattamento. Tradizionalmente, un'indagine o un test dovrebbero essere richiesti solo nel caso in cui dovessero portare ad una diagnosi diversa o a un piano di trattamento alternativo. Quando un paziente effettua dei test i cui risultati si ripresentano, questi sono raramente condivisi totalmente con tutti i professionisti della salute coinvolti nella cura del paziente e conservati presso l'istituto che li ha richiesti originariamente. Di conseguenza, la qualità dell'assistenza del paziente ne risente. Altre istituti non sono a conoscenza della storia completa del paziente e ciò potrebbe portare a decisioni sbagliate, ritardi e costi non necessari per il paziente o per l'istituto sanitario. Il fascicolo sanitario in formato elettronico consente inoltre di far risparmiare tempo al paziente facendogli evitare code e spostamenti per prenotare, pagare e ritirare i documenti. Una copia della cartella clinica cartacea generalmente costa oltre i 15€: con queste cifre l'ospedale copre le spese relative all'attività del personale dell'archivio clinico e al materiale di cancelleria. Un approccio digitale rappresenta quindi anche un risparmio di denaro per i pazienti. Ricapitolando, un sistema informatico pensato per la gestione dei fascicoli sanitari elettronici dei pazienti, porterebbe a vari benefici:

- raccolta collettiva dei dati sanitari frammentati tra gli ospedali
- abbattimento delle tempistiche di accesso ai dati sanitari per i pazienti e i medici autorizzati
- risparmio di denaro per i pazienti
- migliore qualità delle cure

Si vedrà ora una possibile implementazione basata su blockchain.

2.2 Medicalchain

Medicalchain è una piattaforma sanitaria basata su blockchain, nata da una ICO (initial coin offering) il 1° febbraio 2018 e ancora in fase di sviluppo [11]. La ICO è una forma di finanziamento utilizzata da startup o da soggetti che intendono realizzare un determinato progetto realizzato tramite blockchain, in cui vengono creati e poi ceduti dei token, a fronte di un corrispettivo, ai soggetti finanziatori. L'obiettivo di Medicalchain è creare un sistema di gestione delle cartelle cliniche dei pazienti che si fondi sulle caratteristiche proprie della blockchain, quindi decentralizzazione, sicurezza e immutabilità dei dati. Gli utenti che si iscrivono alla piattaforma, possono accedere quando vogliono alla loro cartella sanitaria e dare permessi in lettura a parenti e lettura/scrittura ai propri medici. È possibile perciò monitorare la visione dei propri record, stabilire chi può vedere cosa e per quanto tempo. Di seguito in tabella 2.1 sono riportati alcuni esempi di permessi di lettura/scrittura.

Attori	**Permessi**
Medico	• lettura/scrittura degli FSE consentiti • richiesta di permesso in lettura/scrittura per altri medici/istituzioni
Paziente	• lettura del proprio FSE • permesso ad un medico/istituzione di leggere/scrivere su tutto o parte dell'FSE • revoca di un permesso • permesso ad un parente di leggere/fornire permessi • scrittura di certi attributi: ammontare di tabacco consumato giornalmente, alcol consumato settimanalmente, esercizi settimanali, ecc.
Istituto di ricerca	• permesso in lettura di un FSE

Tabella 2.1: Definizioni dei partecipanti e autorizzazioni

Se un utente è in una situazione di emergenza e c'è necessità di controllare la sua

cartella clinica o si contatta chi ha i permessi d'accesso (medico di base/parenti) oppure, tramite un braccialetto con chip NFC che il paziente deve avere con sé, due medici autorizzati possono accedere ai suoi dati ICE (in case of emergency) che lui deve aver già predisposto (ad esempio gruppo sanguigno, intolleranze, ecc).
Oltre agli FSE, Medicalchain intende sviluppare un servizio di telemedicina e un mercato dei dati sulla salute. L'applicazione di telemedicina consentirebbe agli utenti di consultare un medico da remoto (ad esempio tramite cellulare) a fronte di una tassa pagabile direttamente al medico. Esempi potrebbero essere: telecardiologia, teleradiologia, telepatologia, telepsichiatria, teledermatologia. La moneta utilizzata per i pagamenti è il MedTokens, la criptovaluta di proprietà. Il Marketplace invece consente agli utenti di Medicalchain di negoziare termini commerciali con terzi per usi alternativi o applicazioni dei loro dati sanitari personali, ad esempio fornendo a laboratori scientifici, a fronte di un pagamento, dati che verranno utilizzati nella ricerca medica. Medicalchain è basata su una doppia struttura blockchain. La prima controlla l'accesso alle cartelle cliniche ed è costruita utilizzando Hyperledger Fabric. La seconda è basata su Ethereum e comprende tutte le applicazioni e i servizi della piattaforma (ad esempio telemedicina e marketplace) che verranno realizzati tramite smart contract. I MedTokens, le monete della piattaforma, sono state realizzate con uno speciale smart contract chiamato ERC-20[1], uno standard utilizzato per la creazione intuitiva di token. La netta maggioranza dei token rilasciati sulla blockchain di Ethereum è conforme a questo modello. La blockchain di Hyperledger è basata su autorizzazioni e richiede agli utenti di registrarsi per utilizzarla. Gli accessi sono gestiti dalla piattaforma Civic, una blockchain utilizzata per certificare l'identità degli utenti con l'utilizzo della biometria. Le aziende partner di Civic, tra cui Medicalchain appunto, verificano l'identità di un utente iscritto a Civic direttamente sulla sua blockchain. Per assicurare la privacy degli utenti, i record sono criptati con la crittografia a chiave simmetrica. I record vengono salvati off-chain in data store che fanno parte della giurisdizione dell'utente, ovvero quella del Paese di residenza. Ogni volta che un utente concede a qualcuno un permesso d'accesso ai propri dati, vengono effettuate le seguenti operazioni:

1. Il record viene decriptato con la chiave privata dell'utente proprietario

[1]ERC sta per Ethereum Request for Comment e 20 è il numero assegnato a questa richiesta

2. La chiave simmetrica viene criptata con la chiave pubblica dell'utente autorizzato

Se un utente è autorizzato ad accedere a un record sanitario e richiede l'accesso, viene eseguito il seguente processo:

1. la chiave privata dell'utente richiedente è usata per decriptare la chiave simmetrica

2. la chiave simmetrica viene utilizzata per decriptare i record dell'FSE dell'utente proprietario

Nel caso in cui l'accesso di un partecipante venga rimosso da una cartella clinica:

1. la chiave simmetrica viene decifrata con la chiave privata del proprietario dell'FSE

2. l'FSE viene decriptato utilizzando la chiave simmetrica

3. il record viene criptato con una nuova chiave simmetrica

4. la chiave simmetrica è criptata con tutte le chiavi pubbliche degli utenti autorizzati rimanenti.

In figura 2.1 è illustrato lo schema con cui vengono gestiti i dati degli utenti. Quando un utente crea un nuovo record (riquadro in cima), questo viene salvato off-chain in un data store (a sinistra) e l'hash del dato viene salvato on-chain (a destra); l'hash costituisce un riferimento alla locazione di memoria del dato. Quando un utente vuole leggere un record, questo viene recuperato dal data store, decriptato e il suo hash confrontato con quello presente nel libro mastro. Se gli hash corrispondono, l'utente può leggere il record. Qualsiasi tipo di interazione con le cartelle cliniche viene registrato come una transazione sulla blockchain. Le transazioni sono visualizzabili solo dai partecipanti associati ad essa. Sia i dati off-chain, quindi gli FSE dei pazienti, che i dati on-chain, sono tenuti da nodi fidati (trusted peer) della giurisdizione di appartenenza che sono ospedali, scuole, università, laboratori di ricerca e altre istituzioni statali.
Ci si concentrerà ora sul funzionamento della blockchain utilizzata da Medicalchain per la gestione delle cartelle sanitarie.

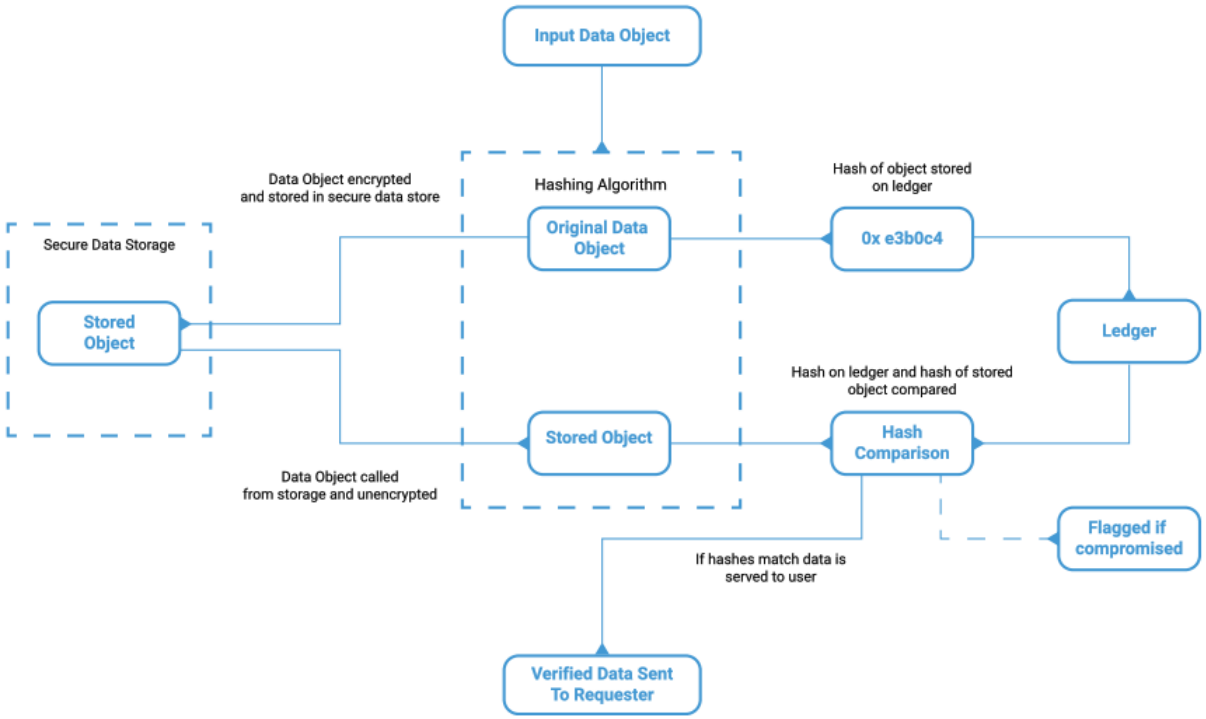

Figura 2.1: Schema di gestione dei dati in Medicalchain [11]

2.3 Hyperledger Fabric

2.3.1 Introduzione

Hyperledger è un progetto open source basato su blockchain avviato dalla Linux Foundation nel 2015. È costituito da dieci differenti piattaforme, una delle quali è Fabric, una permissioned private blockchain promossa da IBM [12]. Hyperledger Fabric (HF) è stata progettata per l'utilizzo in contesti aziendali ed è programmabile a seconda delle esigenze dell'azienda. Gli smart contract, chiamati chaincode, possono essere scritti in linguaggi di programmazione general-purpose come Java, Node.js, GO. Sono supportati vari protocolli di consenso che permettono di personalizzare la piattaforma per adattarsi a casi d'uso particolari. I protocolli non richiedono una criptovaluta nativa per incentivare l'attività di mining o per mandare in esecuzione gli smart contract. Uno dei più noti e utilizzati è Kafka, di cui si parlerà in seguito. Fabric ha un'architettura modulare composta dai seguenti componenti che verranno in seguito approfonditi:

- un servizio di ordinamento configurabile che stabilisce il consenso sull'ordine delle transazioni e trasmette i blocchi ai peer

- un membership service provider che si occupa di dotare le entità della rete di un'identità digitale
- un servizio opzionale peer-to-peer di gossip per diffondere i blocchi, dal servizio di ordinamento agli altri peer
- chaincode programmabili in diversi linguaggi
- il ledger (libro mastro) che può essere configurato per supportare vari tipi di DBMS
- una politica di approvazione e validazione delle transazioni indipendente dall'applicazione

La maggior parte delle blockchain esistenti, tra cui Bitcoin ed Ethereum, segue un'architettura *order-execute* per cui la rete, tramite il protocollo di consenso, convalida e ordina le transazioni e poi le esegue sequenzialmente nello stesso ordine su tutti i nodi peer. Nello specifico ogni peer assembla un blocco contenente transazioni valide pre-eseguite, prova a risolvere la PoW e, se ci riesce, diffonde il proprio blocco alla rete. Ogni peer che riceve il blocco esegue le transazioni in ordine. Perciò si ha una ripetizione della fase di esecuzione. Fabric utilizza un'architettura diversa, di tipo *execute-order-validate*, i cui passaggi, al fine di ottenere un nuovo blocco nella blockchain, sono: esecuzione di una transazione e verifica della correttezza; ordinamento delle transazioni corrette con una politica di approvazione specifica dell'applicazione; formazione del blocco. Questo design si discosta radicalmente dall'altro paradigma, dato che Fabric esegue le transazioni prima di raggiungere un accordo finale sul loro ordine. La fase di execute posta all'inizio serve per filtrare le transazioni inconsistenti prima dell'ordinamento. Si passerà ora a una trattazione dettagliata dell'architettura di HF.

2.3.2 Architettura

La rete di Hyperledger Fabric è costituita da un insieme di componenti: ledger, chaincode, nodi (peer, client), organizzazioni, servizi di ordinamento, canali, autorità di certificazione, Membership Service Provider. Una rete viene creata in seguito alla definizione di un consorzio e dei suoi client, peer, canali e servizi

di ordinamento. Il servizio di ordinamento è il punto di amministrazione della rete poiché contiene la configurazione dei canali. Ogni configurazione descrive la politica del canale e le informazioni relative all'iscrizione per ciascun membro. Un consorzio è composto da due o più organizzazioni che devono essere autenticate su un canale per poter interagire tra loro. Ogni organizzazione possiede dei client, ovvero delle entità che effettuano le transazioni, e dei nodi peer che possiedono il ledger e che, a seconda dei ruoli, approvano le transazioni o le ordinano formando i blocchi. Tutte le entità che fanno parte di una rete devono essere autenticate.

Permessi

La caratteristica principale della blockchain di Hyperledger Fabric è il fatto di essere privata e ad accesso controllato. Per garantire queste proprietà è necessario che gli attori che partecipano alla rete siano dotati di *identità digitale* (ID). Questi ID sono fondamentali in quanto determinano i permessi di accesso alle informazioni e i ruoli degli attori, cioè membro, amministratore, client o peer. L'unione tra ID e gli attributi associati prende il nome di "principale". I principali includono una serie di proprietà quali: l'organizzazione dell'attore, il ruolo, l'identità specifica dell'attore. Sono le proprietà che conferiscono i permessi. Affinché un'identità sia verificabile, deve provenire da un'autorità fidata. Il *Membership Service Provider* (MSP) è il componente che definisce le regole che consentono ad una identità di venir considerata valida per una certa organizzazione. L'implementazione MSP predefinita in Fabric utilizza i certificati X.509 come identità, adottando un'infrastruttura a chiave pubblica (PKI) tradizionale. Una PKI è una raccolta di tecnologie internet che consentono a terze parti fidate di verificare e/o farsi garanti dell'identità di un utente. I quattro elementi di una PKI sono: certificati digitali, chiavi pubbliche e private, autorità di certificazione (CA), liste di revocazione dei certificati. Un *certificato digitale* è un documento che contiene una serie di attributi relativi al titolare del certificato. Il tipo più comune è quello conforme allo standard X.509, che consente la codifica dei dettagli identificativi dell'attore. Gli attributi di un attore sono protetti da crittografia in modo che una manomissione invalidi il certificato. Ad erogare i certificati agli attori sono le *autorità certificate*. Questi certificati sono firmati digitalmente dalla CA e legano assieme l'attore con la sua chiave pubblica (e facoltativamente un elenco completo di proprietà). Di conseguenza, se ci si fida della CA, ci si può fidare anche del

fatto che l'attore specifico sia associato alla chiave pubblica presente nel certificato e che possegga gli attributi inclusi, convalidando la firma della CA sul certificato dell'attore. Dal momento che le CA sono fondamentali per un sistema ad accesso controllato, Fabric fornisce un componente integrato, il *Fabric CA*, che consente di creare CA specifiche per le reti blockchain che si vogliono implementare; tuttavia, le organizzazioni possono implementarne una esterna a loro scelta. Le autorità di certificazione e gli MSP quindi forniscono una combinazione di funzionalità. La CA distribuisce molti tipi diversi di identità digitali verificabili; l'MSP di una certa organizzazione si serve delle CA per assegnare le identità e determinare chi abbia accesso ai canali.

Canali

Un altro componente fondamentale di HF è il *canale*. Si tratta di una sottorete di comunicazione tra due o più membri della rete che ha lo scopo di condurre transazioni private e confidenziali. Un canale è definito da: organizzazioni, anchor peer per organizzazione, libro mastro condiviso, applicazioni di chaincode e nodi del servizio di ordinamento. L'anchor peer è il mezzo di comunicazione tra i peer di organizzazioni diverse che operano all'interno di uno stesso canale. Ogni transazione della rete viene eseguita in un canale, in cui ciascun attore deve essere autenticato e autorizzato ad effettuare transazioni. Per creare un nuovo canale, il client SDK invoca il sistema di configurazione chaincode, ottenendo la creazione di un blocco genesi per il ledger del canale. Tale blocco memorizza le informazioni di configurazione relative alle politiche adottate, ai membri e agli anchor peer. Sebbene qualsiasi di questi peer possa appartenere a più canali e pertanto mantenere più ledger, nessun dato di un certo ledger può passare da un canale ad un altro. Questa separazione è definita ed implementata dal chaincode di configurazione, dall'MPS e dal protocollo di gossip di diffusione dei dati. Questo isolamento dei dati di peer e ledger consente ai membri della rete di interagire tra loro scambiando transazioni private e confidenziali.

Ledger e Chaincode

Il ledger è formato da due componenti: il *world state* che descrive l'ultimo stato del ledger e la *blockchain* che registra tutti i blocchi concatenati di transazioni che hanno portato al valore corrente del world state (figura 2.2). LevelDB è il database predefinito utilizzato per implementare il world state e rappresenta i dati come coppie chiave-valore. Il world state cambia frequentemente in quanto gli stati sono determinati dalla lista di transazioni della blockchain e vengono continuamente creati, aggiornati e cancellati. Ogni canale ha un ledger diverso e ogni peer della rete mantiene il ledger del canale. Un peer che appartiene a canali diversi conterrà ledger diversi.

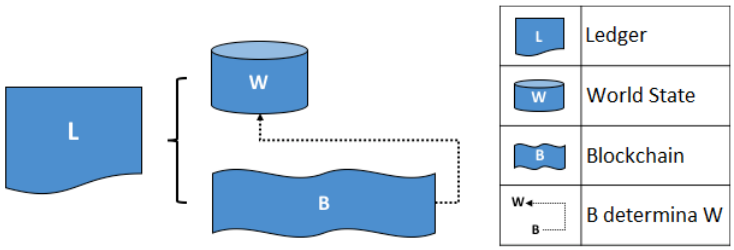

Figura 2.2: Ledger, fonte Doc HF

I *chaincode* rappresentano gli smart contract di HF. Sono dei programmi che vengono installati nei nodi peer della rete da membri autorizzati e definiscono asset modificabili tramite istruzioni rappresentate dalle transazioni; costituiscono quindi la business logic della blockchain. Vengono invocati dai client, applicazioni esterne alla blockchain, per generare transazioni di interrogazione o aggiornamento del ledger. I chaincode quindi, una volta invocati, leggono o modificano le coppie chiave-valore del world state. I chaincode di sistema sono invece software particolari che definiscono le regole per il canale. Le transazioni possono essere di due tipi:

- deploy transaction: creano un nuovo chaincode e prendono un programma come parametro. Quando una deploy transaction viene eseguita correttamente, il chaincode viene installato nei peer del canale
- invoke transaction: eseguono una funzione fornita da un chaincode preesistente andando a leggere o a modificare il world state

Le deploy transaction sono casi speciali di invoke transaction, in quanto la creazione di un nuovo chaincode corrisponde a invocare una transazione sul chaincode

di sistema.
Si passerà ora a descrivere il percorso seguito dalle transazioni che va dall'invocazione fino alla registrazione nella blockchain.

Flusso delle transazioni

I ruoli all'interno della rete di HF sono vari. I nodi si distinguono in:

- *Client*: rappresentano le entità che agiscono per conto degli utenti finali. Devono connettersi a dei peer per comunicare con la blockchain. I client creano e quindi invocano le transazioni
- *Peer*: mantengono una copia del libro mastro. Ce ne sono di due tipi.
  - *Endorser* (approvatore): simulano e approvano una transazione
  - *Committer* (registratore): verificano le approvazioni e convalidano i risultati delle transazioni prima di registrarle sulla blockchain
- *Orderer* (ordinatore): fanno parte del servizio di ordinamento che accetta le transazioni approvate, le ordina in blocchi e spedisce i blocchi ai committer

Il consenso è ottenuto in tre step:

1. approvazione delle transazioni
2. ordinamento
3. validazione e registrazione

La prima fase del processo di registrazione delle transazioni nella blockchain consiste nella proposta di una transazione da parte di un client agli endorsing peer a cui è collegato. Le applicazioni client consentono agli utenti di comunicare con la rete blockchain. Si connettono ai peer ogni qual volta hanno necessità di accedere al ledger e di invocare i chaincode per generare transazioni.
Nella seconda fase gli approvatori, una volta ricevuta la transazione proposta, verificano la firma del client e poi simulano la transazione invocando il chaincode corrispondente e la copia del world state che hanno in locale. Come risultato dell'esecuzione, il peer approvatore calcola le dipendenze della versione letta (readSet) e gli aggiornamenti di stato (writeSet) senza però aggiornare il ledger. Il

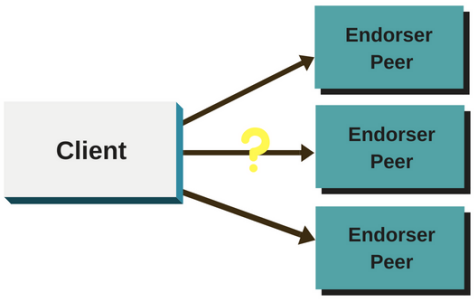

Figura 2.3: Fase 1: proposta transazione

readSet contiene quindi le coppie chiave-valore delle chiavi lette sulla transazione, il writeSet le coppie delle chiavi modificate. Se la proposta rispetta le logiche di approvazione, viene firmata dal peer e spedita al client come approvata, altrimenti viene rifiutata. Se quella del client è un transazione di interrogazione del ledger, una volta ricevuta la risposta dall'endorser, il flusso termina; se è una transazione di aggiornamento del ledger, il flusso continua con altre due fasi. La politica di approvazione (endorsement) specificata nel chaincode di sistema tipicamente prevede che venga accumulato un numero minimo di firme da parte degli endorser affinché la transazione venga considerata valida.

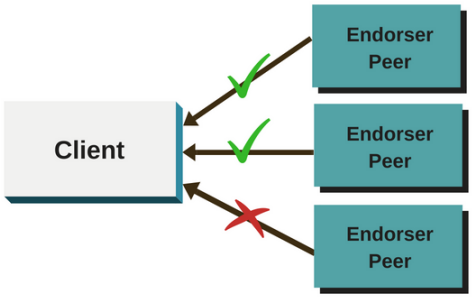

Figura 2.4: Fase 2: endorsement transazione

Nella terza fase, se il client riceve un numero sufficiente di approvazioni, trasmette la propria transazione al servizio di ordinamento. La transazione conterrà il readSet, il writeSet, le firme degli endorser e l'ID canale. Il servizio di ordinamento non ha bisogno di ispezionare il contenuto di una transazione, il suo compito è quello di ricevere le transazioni da tutti i canali della rete, ordinarle cronologicamente per canale e creare dei blocchi.
Nell'ultima fase il servizio di ordinamento consegna i blocchi creati ai peer leader. Il peer leader si occupa di mantenere la connessione con il servizio di ordinamen-

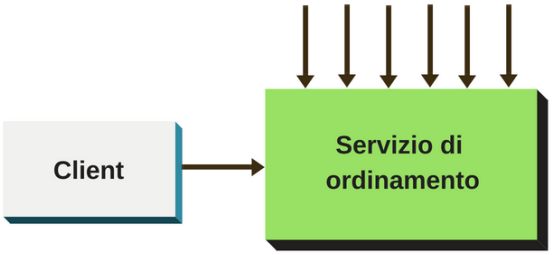

Figura 2.5: Fase 3: broadcast della transazione al servizio di ordinamento

to e avviare la distribuzione dei blocchi in arrivo (protocollo di gossip) verso i peer della propria organizzazione. Una volta che i peer hanno ricevuto i blocchi, le transazioni vengono convalidate per garantire che la politica di approvazione sia soddisfatta e che il readSet ottenuto durante l'endorsement sia identico al world state. Ogni peer aggiunge il blocco alla catena del proprio ledger sia che le transazioni siano valide, sia che non lo siano. Per ciascuna transazione valida il writeSet corrispondente è utilizzato per aggiornare il world state. Infine viene notificato al client se la transazione aggiunta alla catena è stata convalidata o meno. Ricapitolando, HF segue un'architettura di tipo execute-order-validate in cui le

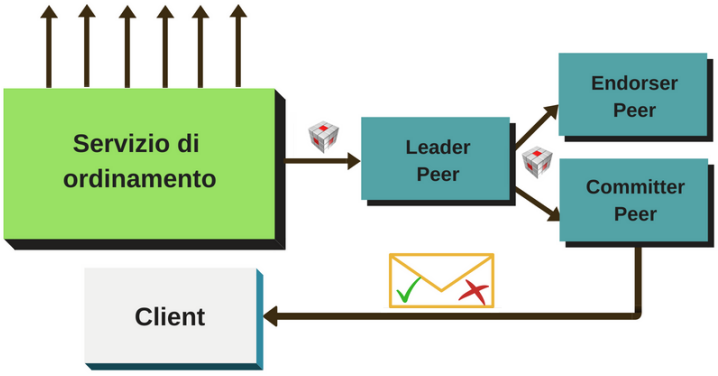

Figura 2.6: Fase 4: invio blocchi ai leader peer, commit nel ledger e notifica al client

transazioni vengono dapprima simulate dai peer approvatori, successivamente vengono ordinate dal servizio di ordinamento e raggruppate in blocchi per canale, infine i blocchi vengono spediti a tutti i nodi peer dei canali che implementano quel servizio di ordinamento e accodati alla blockchain dei loro ledger. Tutto il processo del flusso delle transazioni è illustrato in figura 2.7.

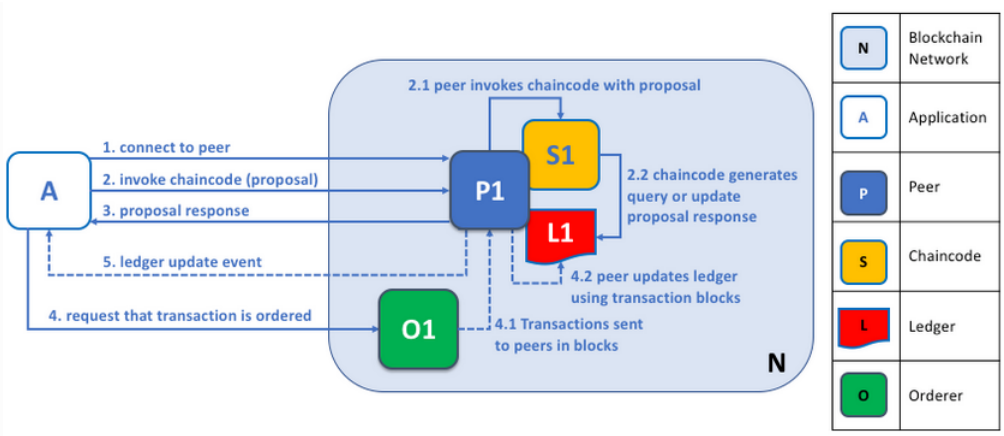

Figura 2.7: Flusso delle transazioni, fonte Doc HF

Kafka

Il servizio di ordinamento più utilizzato in Fabric è quello implementato con Kafka, una piattaforma di streaming distribuita dell'Apache Foundation [13]. Tale servizio consiste in un cluster Kafka e un insieme di nodi (OSN, ordering service node) che si frappongono tra i client e Kafka. Gli OSN autenticano i client, consentono loro di scrivere nella blockchain o di leggere da essa, filtrano e validano le transazioni di configurazione che o riconfigurano un chaincode esistente o ne creano uno nuovo. In Kafka i messaggi, chiamati anche record, vengono scritti nelle partizioni dei topic. Ci possono essere più topic e ciascuno può avere più partizioni. Ognuna di esse è composta da una sequenza ordinata di record che vengono aggiunti in coda uno dopo l'altro. Nel caso di HF i record sono le transazioni, le partizioni i canali e i topic insiemi di canali che rispettano certe politiche di rete. A seguito dell'autenticazione del client da parte di un OSN al quale è connesso, la transazione trasmessa dal client viene inserita nella corrispondente partizione e accodata alle altre. Gli OSN in seguito "consumano" quella partizione ottenendo una lista di transazioni comune a tutti gli OSN. Le transazioni vengono raggruppate in un blocco o al raggiungimento del numero limite di transazioni (batchSize) o allo scadere di un timer (batchTimeout). I blocchi vengono poi salvati in locale dai peer e notificati ai client. In figura 2.8 è illustrato un esempio. Siano OSN0 e OSN2 i nodi connessi ai client che fanno il broadcast delle transazioni e OSN1 il nodo che notificherà i client. Vengono inviate tre transazioni al cluster Kafka. La prima è la transazione "foo" ed è trasmessa da OSN0. Successivamente OSN2 trasmette la transazione "baz" e

infine OSN0 trasmette "bar". Alle transazioni viene assegnato un offset crescente a seconda dell'ordine di ricezione, quindi la prima ricevuta da Kafka ha l'offset più basso. Al verificarsi di una condizione tra batchSize e batchTimeout, viene creato un blocco dai nodi che leggono le partizioni di Kafka. OSN1 forma il blocco 4 che contiene foo, zoo, bar e manda una notifica al client che ne aveva fatto richiesta.

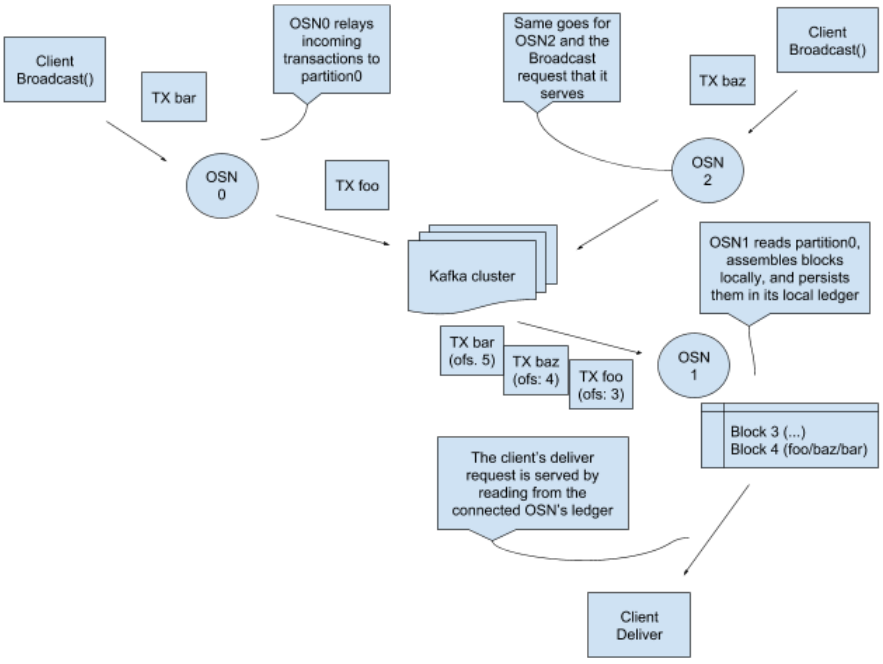

Figura 2.8: Servizio di ordinamento con Kafka, fonte A Kafka-based Ordering Service for Fabric

Per una migliore comprensione di Hyperledger Fabric, si vedrà ora un esempio grafico di una possibile rete. La figura 2.9 illustra una rete N che una certa politica NP1 e un servizio di ordinamento O. Il canale C1 è governato dalla politica CP1 ed è frequentato da due organizzazioni, RA e RB, che formano un consorzio. La politica CP1 è implementata all'interno del servizio O, il quale contiene le configurazioni dei canali C1 e C2 e i certificati dei membri di ciascun canale. I peer P1 e P2 appartenenti rispettivamente a RA e RB, mantengono il ledger L1 del canale C1. A1 e A2 sono i client di RA e RB che hanno l'autorizzazione ad effettuare transazioni nel canale C1. CA1 è l'autorità di certificazione di RA. Il peer P2 oltre a L1 mantiene anche il ledger L2 del canale C2 e fa uso di due diverse chaincode a seconda del canale: S4 e S5. I nodi ordinatori di O appartengono all'organizzazione RD.

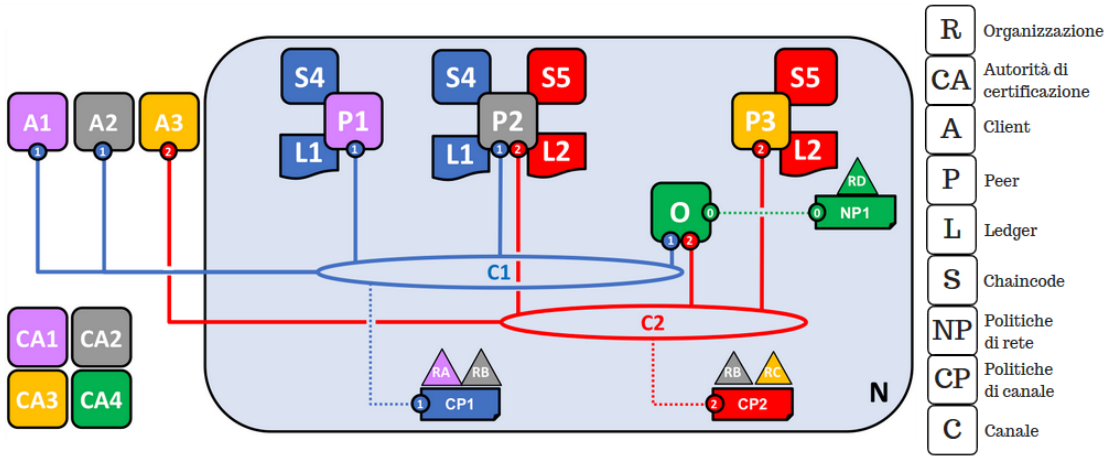

Figura 2.9: Esempio di rete in Fabric, fonte Doc HF

Nel caso di Medicalchain, le organizzazioni sono rappresentate da ospedali, università, laboratori e altre istituzioni statali. Queste strutture gestiscono sia i nodi che hanno il compito di mantenere i dati off-chain, sia i vari tipi di nodi peer che conservano il ledger. Ogni canale è formato da un utente e da coloro ai quali tale utente dà i permessi di lettura/scrittura riguardo alla propria cartella sanitaria. In questo modo i partecipanti di un canale possono consultare solamente le transazioni che li riguardano poiché ogni canale ha il proprio ledger ed è privato. La rete formata da canali e consorzi rappresenta una giurisdizione. La politica di rete sarebbe implementata in base alle regole legislative previste dalla giurisdizione. Una soluzione di questo tipo, basata su una permissioned private blockchain, presenta però degli aspetti critici che verranno discussi in seguito, dopo la presentazione di un'altra possibile implementazione di un sistema per la gestione delle cartelle sanitarie.

2.4 Event Sourcing

Il concetto di blockchain visto come un elenco immutabile di record che vengono accodati uno dopo l'altro è molto simile ad un approccio ad eventi applicabile a un database, introdotto nel 2005 da Martin Fowler [14] e che si sta affermando recentemente sempre di più, denominato event sourcing. L'approccio tipico relativo alla gestione dei dati di un database da parte di un'applicazione prevede che essa mantenga lo stato corrente, aggiornandolo ogni volta che gli utenti vi apportano modifiche. Ad esempio, nel modello tradizionale di creazione, lettura, aggiornamento ed eliminazione (CRUD), è possibile leggere i dati del database, fare alcune modifiche ed aggiornare lo stato corrente con i nuovi valori. Questo approccio ha varie limitazioni:

- le operazioni di aggiornamento eseguite direttamente nell'archivio dati causano un sovraccarico di elaborazione che può rallentare le prestazioni e la velocità di risposta e limitare la scalabilità
- in un dominio collaborativo con molti utenti concorrenti, è più probabile che si verifichino conflitti perché le operazioni di aggiornamento avvengono su singoli dati
- a meno che non esista un meccanismo di controllo aggiuntivo, che registra i dettagli di ciascuna operazione in un registro separato, la cronologia viene persa

Event sourcing (origine degli eventi) è un pattern che utilizza gli eventi come strumento per modellare la logica di business. Lo stato dell'applicazione e i suoi cambiamenti vengono rappresentati come una sequenza di eventi salvati nella base di dati, chiamata event store. Non esistono operazioni di modifica o cancellazione, la modalità è append-only e tutto quello che è successo nel dominio è rappresentato da questa sequenza. Con tale logica è possibile esplorare il passato a partire dallo stato corrente ricostruendo la sequenza di eventi che lo ha generato, cioè la sua storia. Tornando al sistema tradizionale, un'applicazione processa sia le richieste che modificano lo stato del sistema (i comandi), sia le query. L'applicazione lavora in maniera sincrona con un processo di request-response e tutti i comandi vanno ad agire su un database tipicamente relazionale, dove ciascuno di essi aggiorna lo stato dell'aggregato (figura 2.10). L'aggregato

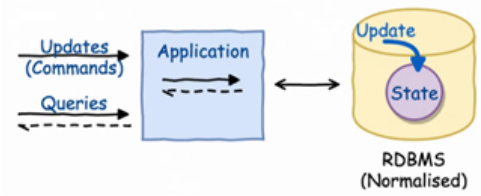

Figura 2.10: Sistema tradizionale sincrono, fonte [15]

è un concetto dell'approccio DDD (domain driven design) e rappresenta una collezione di oggetti legati assieme da un'entità padre, nota come radice dell'aggregato, la quale garantisce la consistenza dei cambiamenti compiuti ai membri interni. L'aggregato ha uno stato che è l'insieme dei valori di queste entità. Un esempio di aggregato è quello dell'ordine di un prodotto (radice) che comprende indirizzo di consegna, scorte di magazzino, stato della richiesta (figli). Quindi i comandi aggiornano lo stato dell'aggregato ma se si devono servire più richieste e si vuole mantenere bassa la latenza del sistema per non far attendere troppo i client, bisogna scalare il sistema. Lo si può scalare orizzontalmente aumentando il numero dei nodi o verticalmente aumentandone la potenza. Se vengono aggiunti nodi il sistema diventa distribuito ma è comunque sincrono e request-response quindi il client deve aspettare che la scrittura sia confermata nel database prima di essere rilasciato. Gli aggiornamenti quando coinvolgono aree comuni causano contention e hanno bisogno dei lock, che a loro volta bloccano il sistema. Il client a monte deve aspettare la conferma e intanto i comandi si impilano, per cui scalare orizzontalmente non dà vantaggi, il collo di bottiglia è il database. Il processo sincrono è nemico della scalabilità. Si vedrà ora un'implementazione asincrona che utilizza un message broker. Il client anziché dialogare direttamente con il database, lo fa con un broker, un intermediario. In questo modo i comandi, anche se non ancora registrati nel database, vengono intanto salvati e il client viene rilasciato (figura 2.11).

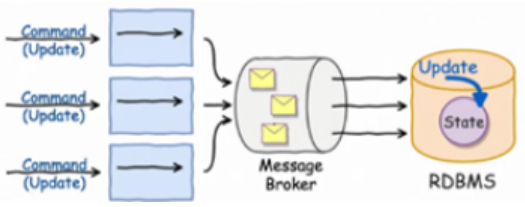

Figura 2.11: Applicazione asincrona

Il sistema ottenuto è distribuito, asincrono e message-driven, ma presenta dei problemi. Non è possibile garantire un ordine dei messaggi a livello globale. Ci possono essere situazioni di failure dei nodi in cui i messaggi vengono consegnati in ordine errato; in questo modo nel database si crea inconsistenza. Ad esempio, se vengono applicati due aggiornamenti che interferiscono tra loro e in ordine inverso, non è più possibile recuperare lo stato originale. La soluzione è salvare i comandi in ordine di arrivo nel broker; anziché applicarli all'interno di un database relazionale e modificarne lo stato, vengono salvati così come sono, uno in fila all'altro. Si ottiene in questo modo un log persistente, un append-only che evita la contention. Questo modus operandi è analogo a quello di una blockchain in cui le transazioni vengono accodate e vanno a formare un elenco; inoltre ciò viene fatto in modo asincrono, proprio come con i blocchi di transazioni (figura 2.12). A questo punto si deve ricostruire lo stato dell'aggregato quindi i comandi

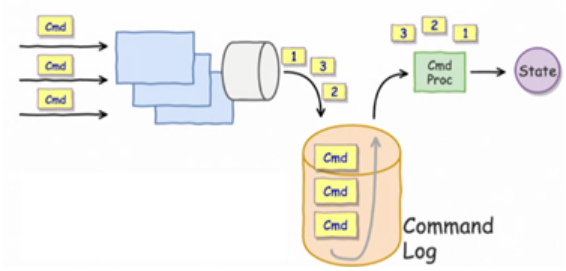

Figura 2.12: Log dei dati

vengono presi e passati in fila attraverso una qualche business logic. I comandi ricevuti possono però essere mal formati o malevoli; spesso inoltre un comando influenza più di un aggregato. Ad esempio il comando di un invio di un ordine nell'e-commerce va a modificare anche l'aggregato del conto dell'utente o dello stato di magazzino. Quindi ciò rende non adatti i comandi a ricostruire lo stato. È opportuno spostare l'attenzione sugli eventi. L'evento è l'effetto di un comando, cioè qualcosa che è successo e che non può essere cambiato, è immutabile; il concetto è analogo a quello di una transazione in una blockchain. È possibile far sì che ogni evento rappresenti un singolo aggregato. Questo rende gli eventi più adatti a ricostruire lo stato degli aggregati, piuttosto che i comandi. Quindi si deve operare una trasformazione dall'idea di command-sourcing a quella di event sourcing (figura 2.13). Ci deve essere perciò un componente che valida i comandi rigettando quelli non validi, in modo da informare il client che il comando non è accettabile. Inoltre il componente può suddividere il comando in

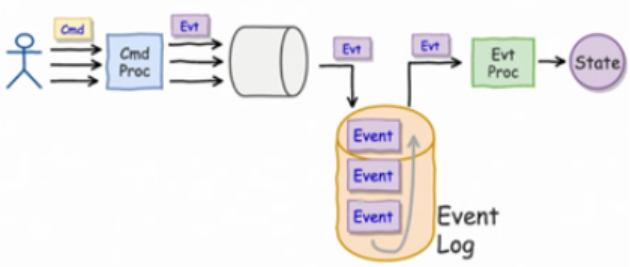

Figura 2.13: Idea di Event-Sourcing

più eventi (figura 2.14). Tale componente può essere implementato secondo due

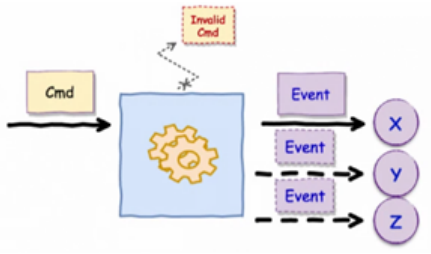

Figura 2.14: Comando suddiviso in vari eventi

approcci. Il primo approccio è stateless, e agisce velocemente in quanto valida il comando ricevuto solo da un punto di vista sintattico e invia al client una risposta positiva o un messaggio d'errore, a seconda dell'esito della validazione. Il secondo è stateful ed è più lento, perché la validazione consiste nel ricostruire l'intero stato dell'aggregato e controllare se è tutto in ordine. Quest'ultimo approccio è quello mainstream, anche se presenta due problemi. Diventa un punto di sincronizzazione nel sistema perché tutti i comandi relativi a quell'aggregato passano da esso e sono processati come single thread, causando rallentamenti. Nel caso in cui i comandi arrivino fuori ordine o vengano persi (tipico dell'IoT) va in fallimento. Quindi è consigliabile il primo approccio. Per concludere la parte relativa alla scrittura, bisogna introdurre il componente che si occupa di garantire la consistenza eventuale di business. Consistenza significa che prima di iniziare una transazione il sistema è in uno stato internamente coerente e quando la transazione termina, tale sistema deve trovarsi in un altro stato coerente, cioè senza violazione di vincoli di integrità della base dati. La consistenza eventuale è meno rigida perché non è richiesta la coerenza immediata del sistema, ma può avvenire a posteriori. La consistenza eventuale di business riguarda i processi esterni del mondo reale anziché le transazioni tra sistemi. Un modo usato dall'e-

vent sourcing per ottenerla è con l'utilizzo del pattern Saga. È una macchina che contiene stati e riceve eventi che riguardano un particolare processo, un ordine ad esempio. Mantiene lo stato, traccia quello che succede e se si verifica qualche errore, fa un'azione correttiva emettendo un evento che va a sistemare lo stato di un certo aggregato. Saga inoltre è "out of band" e lavoro in modo asincrono al di fuori del pattern di persistenza, per cui non blocca la scrittura a monte (figura 2.15).

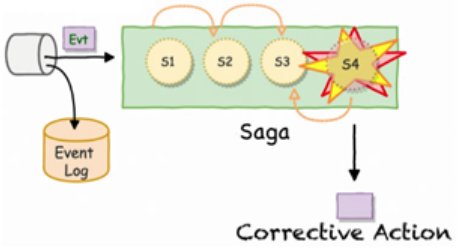

Figura 2.15: Pattern Saga

Con l'approccio event sourcing, l'idea è quella di scrivere prima gli eventi e successivamente, se il componente si accorge di qualche errore, viene effettuato un evento correttivo che ripara lo stato. Si vedrà ora la parte relativa alla lettura dei dati. Per avere lo stato di un ordine, il sistema recupera dall'event log tutti gli eventi relativi a quell'aggregato e li passa attraverso la business logic che ricostruisce lo stato e lo ritorna. Questo meccanismo non è efficiente se si fanno query complesse. Per risolvere questo problema bisogna ricorrere al concetto di CQRS, ovvero alla separazione tra comandi e query (command query responsability segregation). Si ottengono quindi due sistemi separati che lavorano indipendentemente: un message broker, che processa i comandi (C) e li va a scrivere nell'event log, e un sistema di recupero (Q) che riceve gli eventi sia direttamente dal broker, sia dall'event log, e va a servire le query (figura 2.16).

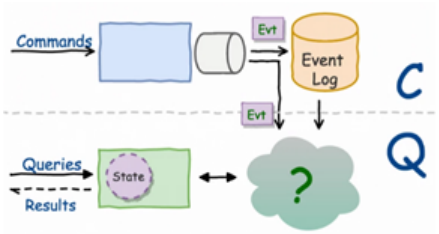

Figura 2.16: CQRS

Il modo più comune per implementare Q è costruire delle viste, cioè eventi

processati e immagazzinati in un data store secondario che ricostruiscono lo stato in modo opportuno. Il data store può essere fatto in vari modi, ad esempio con un KV store, un DB graph, un RDBMS o anche con combinazioni di questi. Con questa implementazione C e Q possono scalare indipendentemente, per aumentare le prestazioni.
A questo punto si immagini che la società fittizia "Medicalevent" implementi un sistema di gestione delle cartelle sanitarie basato su event sourcing. Il sistema, come per Medicalchain, rimane privato, ad accesso controllato e con transazioni dotate di firme digitali apposte dagli utenti ma viene gestito dallo Stato. Le differenze principali sono quindi la modalità con cui vengono salvati e recuperati i dati, il registro delle transazioni e il tipo di gestione (centralizzata/decentralizzata). Nella parte conclusiva si metteranno a confronto le due proposte.

2.5 Confronto implementazioni

Entrambe le soluzioni proposte, nonostante le differenze implementative, conseguono un risultato comune: la creazione di un registro formato da una lista crescente di transazioni che segue un ordine cronologico. Le transazioni, una volta inserite, sia che si tratti della blockchain, che dell'event log, non vengono più rimosse e vanno a costituire lo storico di tutte le azioni intraprese da un utente e da coloro ai quali quest'ultimo ha dato permessi di lettura o modifica della cartella sanitaria. La differenza sostanziale tra i due tipi di registri sta nel modo di raggruppare le transazioni e di legarle crittograficamente con degli hash, caratteristiche presenti nella blockchain. La domanda che ci si pone a questo punto è quali siano i vantaggi derivanti da una struttura a catena rispetto a una senza. Per rispondere al quesito è opportuno riprendere le proprietà e le tipologie di blockchain viste nel capitolo precedente. Si consideri anzitutto l'aspetto della sicurezza. Ciò che garantisce che i dati non vengano falsificati non sono gli hash tra i blocchi ma le firme digitali apposte nelle transazioni. Un utente malintenzionato che intende creare una transazione in cui afferma di aver ricevuto una certa medicina o un trattamento da un medico, non può farlo senza possedere la chiave privata di quest'ultimo. Allo stesso modo non è possibile modificare una transazione esistente perché l'alterazione provocherebbe un'invalidazione della firma e la transazione non verrebbe accettata in fase di validazione. L'unica

operazione che si può fare per modificare il registro è escludere delle transazioni o impedendo che vengano inserite o rimuovendole una volta che sono state inserite. Entrambe le azioni corrispondono ad attacchi di tipo DoS. Si è visto nel caso di Bitcoin che la probabilità di alterare un certo blocco diminuisce man mano che ne vengono aggiunti di nuovi dopo di esso e si calcola che il numero sufficiente di conferme per considerare un blocco immutabile è sei. L'elemento che determina la difficoltà crescente di manomettere un blocco vecchio è la PoW. Un attaccante, per imporre il proprio blocco fraudolento, dovrebbe risolvere una nuova PoW per esso e per tutti quelli successivi fino a raggiungere e poi superare la catena principale esistente. Dal momento che la risoluzione della PoW è un'operazione molto impegnativa, non è possibile condurre attacchi di questo tipo senza avere una potenza di hashing di almeno il 51% di tutta la rete. In una permissioned blockchain come Hyperledger Fabric non c'è un protocollo di consenso che preveda una fase di mining condotta da miner che ricevono degli incentivi per il loro lavoro, in quanto i nodi che validano le transazioni, raggiungono il consenso e mantengono la blockchain, sono nodi fidati. In un contesto trusted il consenso quindi si raggiunge con maggior facilità rispetto a uno trustless in cui è necessaria una strategia per eleggere in maniera democratica il nodo che produrrà un blocco. Ciò consegue che il concetto di immutabilità dei blocchi vada ad indebolirsi. Non essendoci una PoW, data una certa blockchain, un malintenzionato potrebbe alterarne un blocco ricalcolando, con poca fatica, l'hash di quest'ultimo e di tutti quelli successivi a patto di mantenere una coerenza tra le transazioni. Una riduzione di difficoltà per modificare la blockchain implica un indebolimento del ruolo dell'hash tra i blocchi che così non costituisce più un'arma a favore dell'immutabilità, ma un semplice controllo di integrità. In Bitcoin infatti, grazie alla PoW, l'hash tra i blocchi, oltre a rappresentare un controllo di integrità e ad implementare un sistema di marcatura temporale che dà un ordine alle transazioni e risolve così il problema della doppia spesa, serve per rendere la blockchain resistente alle alterazioni. In Fabric l'ordine delle transazioni è dato dal servizio di ordinamento e la resistenza ai cambiamenti è data dalla distribuzione del registro tra i nodi, non dall'hash tra i blocchi. Un attaccante per diffondere il proprio ledger corrotto con poca fatica, dovrebbe attaccare la maggioranza dei nodi, come per un qualsiasi sistema distribuito. Quindi la struttura a catena del registro di Fabric non porta a una sicurezza maggiore rispetto a quella senza

catena del registro di Medicalevent. Un'altra apparente differenza tra le due implementazioni è la decentralizzazione. Una blockchain permissionless è per definizione pubblica e ad accesso libero. In Bitcoin questi due ingredienti, combinati con la PoW, consentono di ottenere un sistema decentralizzato in una rete di nodi trustless. La blockchain di Medicalchain invece basa il proprio funzionamento su una rete di nodi trusted che fanno parte di organizzazioni quali ospedali, università e altri enti. Pur essendo questi enti diversi tra loro rispondono tutti alla stessa giurisdizione. Quindi i nodi peer che mantengono il ledger, anche se appartengono a organizzazioni diverse, sono in realtà controllati da un unico ente centrale: lo Stato. Ciò potrebbe comportare possibili azioni di alterazione della blockchain discusse prima. Lo Stato ad esempio avrebbe il potere di forzare i nodi peer che mantengono il registro per escludere una certa transazione in arrivo. Da queste osservazioni deriva che, allo stesso modo di Medicalevent, anche l'implementazione basata su permissioned blockchain è in realtà governata da un unico attore. Constatato perciò che le due applicazioni non hanno differenze riguardo alla sicurezza e sono entrambe centralizzate, è preferibile adottare un sistema palesemente centralizzato piuttosto di uno "implicitamente centralizzato", per motivi di costi infrastrutturali e di prestazioni. Con una soluzione del primo tipo non c'è bisogno di raggiungere il consenso in quanto il registro è tenuto e controllato da un singolo ente; ne consegue un miglioramento delle performance. Fabric inoltre, a seconda della politica di endorsement, può avere un throughput, ovvero un numero diverso di transazioni al secondo. All'aumentare del numero di approvazioni che un utente deve ottenere prima di poter trasmettere la propria transazione al servizio di ordinamento, si avrà un'inevitabile diminuzione del throughput [16]. Riassumendo, si possono fare due considerazioni, una di carattere generale e una più specifica:

- Considerazione 1: una permissioned blockchain che per sua natura prevede un protocollo di consenso più efficiente di una PoW, non potrà essere resistente ai cambiamenti come lo è Bitcoin e quindi i legami crittografici tra i blocchi servono solo come controllo di integrità.

- Considerazione 2: la permissioned blockchain adattata per questo specifico caso d'uso perde in decentralizzazione.

Riguardo alla prima considerazione, Andreas Antonopoulos, uno dei maggiori esperti di Bitcoin e autore del best-seller Mastering Bitcoin [[2]], in una confe-

renza del 2015 ha espresso con una metafora la differenza tra permissionless e permissioned blockchain. L'autore ha paragonato una permissionless come Bitcoin a un "sewer rat", un topo di fogna newyorkese con un sistema immunitario resistente a qualsiasi tipo di malattia. Mentre le permissioned blockchain delle banche le ha definite come "bubble boy"[2]. Inoltre in un'intervista del 2016 [17] afferma con sarcasmo: "*As long as you trust that nine banks are better than one central clearinghouse and won't cheat as much, then okay*" collegandosi anche alla seconda considerazione. L'implementazione per questo caso d'uso è implicitamente centralizzata perché i nodi fanno parte di organizzazioni statali. Se i nodi invece appartenessero a enti privati come le banche, ciò non rappresenterebbe comunque garanzia di decentralizzazione perché con pochi attori in gioco, potrebbero essere presi accordi comuni. Nell'assunzione che sia necessario avere la maggioranza dei nodi collusi per sferrare un attacco, ad esempio su una dozzina di nodi, il 51% sarebbe un numero risibile e per nulla difficile da ottenere tramite accordi. Nel caso in cui invece il protocollo del consenso prevedesse una tolleranza ai guasti bizantini, basterebbe il 34% dei nodi collusi. In generale si potrebbe quindi affermare che una permissioned private blockchain non sia sicura come Bitcoin e abbia una scarsa, se non nulla, decentralizzazione.

[2]Negli anni '70 venne così definito David Vetter, un ragazzo costretto a vivere in una camera sterile per via della sua malattia al sistema immunitario

Capitolo 3

Conclusioni

In questa tesi sono stati trattati vari argomenti riguardanti la tecnologia blockchain. Nel primo capitolo si è cercato di dare un quadro dettagliato circa il funzionamento del protocollo Bitcoin. Sono stati dapprima presentati i componenti basilari quali le transazioni, i blocchi, la struttura a catena e la crittografia. Poi si è passati all'analisi delle proprietà che si vogliono ottenere con l'implementazione di una blockchain: decentralizzazione, sicurezza e immutabilità. Gli elementi fondamentali che consentono di ottenerle sono: rete di nodi peer-to-peer, registro pubblico e algoritmo di consenso della PoW. I concetti di moneta digitale, struttura a catena e PoW erano già esistenti prima di Bitcoin, l'innovazione apportata da Satoshi Nakamoto è stato il modo con cui è riuscito a combinare gli ultimi due per implementare il primo. L'impiego dell'algoritmo di hashcash di Adam Back come PoW per i miner che intendono creare un blocco è inoltre l'intuizione fondamentale che ha consentito di implementare un meccanismo di consenso democratico in una rete di nodi trustless. Si sono discusse poi le criticità del protocollo quali i consumi del mining, il costo variabile delle fee e la scalabilità. Riguardo a quest'ultima, si è visto come i canali di pagamento e la Lightning Network, siano considerate ad oggi le soluzioni più interessanti per migliorarla. Oltre a Bitcoin è stata presentata anche Ethereum, la seconda blockchain più utilizzata. La novità principale che ha introdotto rispetto a Bitcoin riguarda gli smart contract, applicazioni decentralizzate auto-eseguibili al verificarsi di certe condizioni; realizzate con un linguaggio Touring completo. Un'altra differenza è il modo con cui viene rappresentato il saldo, non più calcolato recuperando le UTXO, ma basato sul modello di account. Come per Bitcoin, anche Ethereum usa una PoW, ma si è visto che a breve verrà adottato *Casper*, un nuovo protocollo

basato su Proof of Stake. Si è concluso il capitolo mostrando le differenze tra permissionless e permissioned blockchain. Nel secondo capitolo si è discusso un caso d'uso in ambito sanitario e due possibili proposte implementative. Dopo aver elencato alcune problematiche relative al panorama informatico sanitario, è stata fatta la scelta di concentrarsi sulla mancanza di un sistema di gestione delle cartelle sanitarie elettroniche. La prima soluzione proposta descrive un progetto partito agli inizi del 2018, Medicalchain. La società prevede di mantenere i dati degli utenti off-chain e salvare on-chain le operazioni svolte sulle cartelle, impiegando Hyperledger Fabric, una permissioned private blockchain. Gli utenti devono autenticarsi per ottenere l'accesso alla piattaforma e per scambiare transazioni con coloro ai quali hanno dato accesso in lettura o scrittura alla propria cartella. Gli scambi avvengono all'interno dei canali e in seguito a una fase di esecuzione, ordinamento e validazione, i ledger vengono aggiornati e mantenuti da nodi fidati appartenenti ad alcune organizzazioni. La seconda implementazione utilizza l'approccio event sourcing, che in modo simile a una blockchain, rappresenta lo stato dell'applicazione e i suoi cambiamenti come una sequenza di eventi immutabile. I client dialogano in modo asincrono con un broker che salva gli eventi in un event store seguendo l'ordine di arrivo. Le query vengono processate da un sistema di recupero separato dall'event log; in questo modo i due processi di scrittura e lettura possono scalare indipendentemente. Ad implementare questo approccio si è immaginata essere Medicalevent, una società fittizia alternativa a Medicalchain, governata dallo Stato. Nel confronto tra le due implementazioni è emerso che la sicurezza di una permissioned blockchain, in termini di immutabilità dei blocchi, non è maggiore di quella di un registro senza una struttura a catena, a causa dell'assenza della PoW. Per quanto riguarda l'aspetto della decentralizzazione della blockchain di Medicalchain, si è constatato che in realtà, essendo i nodi peer appartenenti a enti statali, rispondono tutti a un ente centrale, lo Stato appunto. Quindi si tratta di una centralizzazione implicita. Essendo perciò le due implementazioni simili, è preferibile utilizzare un approccio basato su un sistema palesemente centralizzato per questioni di prestazioni. Riguardo all'aspetto della decentralizzazione, si è inoltre asserito che, anche se i nodi peer fossero mantenuti da enti privati come le banche, essendoci un numero limitato di attori, non sarebbero impraticabili situazioni di comune accordo con conseguente centralizzazione del sistema.

Bibliografia

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

[2] Andreas M Antonopoulos. *Mastering Bitcoin: Programming the open blockchain*. "O'Reilly Media, Inc.", 2017.

[3] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[4] Adam Back et al. Hashcash-a denial of service counter-measure, 2002.

[5] Bruce Schneier. Applied cryptography protocols. chapter 4.1. John Wiley & Sons, 1995.

[6] Vitalik Buterin. Ethereum white paper, 2014. *A Next-Generation Smart Contract and Decentralized Application Platform*, 2014.

[7] Nick Szabo. Smart contracts. *Unpublished manuscript*, 1994.

[8] Gavin Wood. Ethereum yellow paper, 2014.

[9] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 507–527. Springer, 2015.

[10] Arvind Narayanan. *"Private blockchain" is just a confusing name for a shared database*, 2015.

[11] Medicalchain. *Whitepaper Medicalchain 2.1*, 2018.

[12] Hyperledger fabric. *A Blockchain Platform for the Enterprise*.

[13] Apache kafka. *Apache Kafka, a distributed streaming platform*.

[14] Martin Fowler. Event sourcing. *Online, Dec*, page 18, 2005.

[15] Lorenzo Nicora. *A Visual Introduction to Event Sourcing and CQRS*, 2016.

[16] Parth Thakkar, Senthil Nathan, and Balaji Vishwanathan. Performance benchmarking and optimizing hyperledger fabric blockchain platform. *arXiv preprint arXiv:1805.11390*, 2018.

[17] Mark Frauenfelder. *Bitcoin is the Sewer Rat of Currencies, year=2016*.

[18] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 180–184. IEEE, 2015.

[19] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In *International Workshop on Open Problems in Network Security*, pages 112–125. Springer, 2015.

[20] Ghassan Karame, Elli Androulaki, and Srdjan Capkun. Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin. *IACR Cryptology ePrint Archive*, 2012(248), 2012.

[21] Medibloc. *MediBloc: Blockchain-based Healthcare Information Ecosystem* , 2017.

[22] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pages 1–10. IEEE, 2013.

[23] Fergal Reid and Martin Harrigan. An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks*, pages 197–223. Springer, 2013.

[24] Konstantinos Christidis and Michael Devetsikiotis. Blockchains and smart contracts for the internet of things. *IEEE Access*, 4:2292–2303, 2016.

[25] Wei Xin, Tao Zhang, Chengjian Hu, Cong Tang, Chao Liu, and Zhong Chen. On scaling and accelerating decentralized private blockchains. In *Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), 2017 IEEE 3rd International Conference on*, pages 267–271. IEEE, 2017.

[26] Richard Dennis, Gareth Owenson, and Benjamin Aziz. A temporal blockchain: a formal analysis. In *Collaboration Technologies and Systems (CTS), 2016 International Conference on*, pages 430–437. IEEE, 2016.

[27] Asaph Azaria, Ariel Ekblaw, Thiago Vieira, and Andrew Lippman. Medrec: Using blockchain for medical data access and permission management. In *Open and Big Data (OBD), International Conference on*, pages 25–30. IEEE, 2016.

[28] Jae Kwon. Tendermint: Consensus without mining. *Retrieved May*, 18:2017, 2014.

[29] Christian Cachin. Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, 2016.

[30] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, page 30. ACM, 2018.

[31] Patrick Dai, Neil Mahi, Jordan Earls, and Alex Norta. Smart-contract value-transfer protocols on a distributed mobile application platform. *White paper Qtum*, 2017.

[32] Kyrylo Voronchenko. Do you need a blockchain? 2017.

[33] Dominic Betts, Julian Dominguez, Grigori Melnik, Fernando Simonazzi, and Mani Subramanian. Exploring cqrs and event sourcing: A journey into high scalability, availability, and maintainability with windows azure. 2013.