Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Matematica

La crittografia nel sistema di moneta digitale Bitcoin

Tesi di Laurea in Crittografia

Relatore:
Prof. Davide Aliffi

Presentata da:
Beatrice Bertani

II sessione

Anno Accademico
2013/2014

Introduzione

Bitcoin è un sistema di critto valuta elettronica peer-to-peer, ideato da un anonimo conosciuto con lo pseudonimo "Satoshi Nakamoto", i cui principi sono descritti in un documento pubblicato a fine 2008 ed ora accessibile in rete. Successivamente viene implementato da Nakamoto il codice del software open source *Bitcoin Core*, con cui è possibile eseguire le transazioni e che tuttora costituisce la spina dorsale della rete Bitcoin.
Una delle motivazioni che spinge Nakamoto all'ideazione del sistema Bitcoin può essere raccolta in questa sua breve citazione:
"Bitcoin is very attractive to the libertarian viewpoint if we can explain it properly."
L'utilizzo di Bitcoin permette di effettuare transazioni con sicurezza da un soggetto ad un altro e, grazie a una tecnologia peer-to-peer che opera senza alcuna autorità centrale e intermediari quali banche o istituzioni governative, sia la gestione delle operazioni sia l'emissione di denaro sono svolte per mezzo di un protocollo P2P.

Cos'è realmente un bitcoin?
Un bitcoin, o più brevemente BTC, è l'unità di valuta del sistema Bitcoin: una stringa digitale alfanumerica e univocamente determinata, che rappresenta una transazione precedentemente avvenuta ed è inserita in un registro pubblico. Quest'ultimo, denominato Block Chain, registra i proprietari della valuta durante una transazione senza rivelare la loro identità, attraverso l'utilizzo di un indirizzo, una sorta di pseudonimo. Ogni bitcoin rappresenta una catena di transazioni, o più precisamente di firme digitali: quando un proprietario trasferisce un certo numero di bitcoins all'acquirente sfrutta un meccanismo di firma digitale che garantisce l'autenticità del pagamento. Ciò che risulta essere firmato è un valore di una particolare funzione, chiamata funzione Hash e inserita nella transazione precedente, e la chiave pubblica del futuro proprietario.

Nel documento Nakamoto suggerisce inoltre una soluzione al problema della spesa-doppia (*double-spendig*) grazie all'utilizzo di un network peer-to-peer distribuito, il quale esegue una marcatura temporale sulle transazioni generando una prova computazionale dell'ordine cronologico di queste. Non c'è alcuna organizzazione che governa i bitcoins. Essi possono essere gestiti da chiunque installi il software sul proprio computer. Il sistema è creato in modo che anche i malintenzionati possono accedere alla rete, ma questi vengono ignorati fino a quando la maggior parte della rete risulta onesta. Se un domani un gruppo di persone o aziende riuscisse ad acquisire la maggioranza della potenza di calcolo nella rete Bitcoin, sarebbe in grado di invertire le proprie transazioni e bloccare nuove transazioni, anche se non sarebbe comunque in grado di rubare direttamente bitcoin dai portafogli altrui.
Nel corso della trattazione si propone un analisi più tecnica del sistema di gestione e creazione dei Bitcoin e degli strumenti crittografici che hanno permesso l'avvento di questa moneta elettronica decentralizzata.

Indice

Elenco delle figure

Capitolo 1

Nel mondo del Bitcoin

1.1 Successo di Bitcoin e suoi vantaggi

Innanzitutto conviene considerare le molteplici ragioni per cui il bitcoin ha incontrato un certo successo.

- **Utilità**: il bitcoin può essere usato per comprare o vendere beni e servizi, anche illegali (come del resto il denaro contante), con il vantaggio di mantenere l'anonimato e di non richiedere alcun trasferimento materiale. Ad esempio il sito Silk Road accetta solo bitcoin.

- **Cambio**: attraverso numerosi siti web è possibile convertire i bitcoin con le altre tipologie di monete, e viceversa.

- **Speculazione**: il valore del bitcoin sta aumentando grazie alla sua sempre maggiore popolarità, per questo motivo gli speculatori acquistano bitcoin nella speranza di ottenere successivamente un profitto (ma rischiando in realtà grosse perdite).

- **Scarsità**: la fornitura dei bitcoin è limitata. La produzione è algoritmicamente limitata e il tetto massimo è di 21 milioni circa.

Per quanto riguarda gli aspetti più vantaggiosi del sistema troviamo sicuramente al primo posto il fatto che nel mondo della critto valuta Bitcoin è una moneta decentralizzata e basata su prove crittografiche, che non risente quindi della debolezza presente in un modello basato sulla fiducia in autorità di garanzia. Ciò a sua volta comporta una notevole riduzione dei costi delle

transazioni rispetto ad altri servizi di pagamento online. E' possibile inoltre trasferire qualsiasi quantità di denaro in maniera relativamente veloce e sicura. Le transazioni avvengono attraverso un indirizzo che permette al proprietario di mantenere l'anonimato. La moneta è irreversibile e non falsificabile: ogni transazione, una volta effettuata, non può essere cancellata, neanche dal mittente, poiché viene direttamente inclusa nella Block Chain, che è pubblica e accessibile a chiunque. Ci sono due principali minacce al successo del Bitcoin: l'intervento dei governi e la competizione con altre critto valute.
Bitcoin è nota per essere una rete che facilita le attività legali, ad esempio la compravendita di droghe online, il gioco d'azzardo o riciclaggio di denaro. Se il Bitcoin diventasse una moneta forte, e venisse usata sempre più in sostituzione del dollaro, le autorità governative potrebbero decidere di proibirne l'utilizzo. Inoltre, dall'introduzione del Bitcoin sulla rete, diverse monete alternative, gestite tuttavia da banche centrali, sono state create e diffuse grazie al web; se una di queste acquisisse una visibilità maggiore, il Bitcoin perderebbe velocemente il suo primato.

1.2 Come gestire i propri bitcoin

E' possibile gestire i propri bitcoin attraverso il cosiddetto *wallet*, anche detto borsellino elettronico: un software, o un file, che contiene una lista di chiavi private. Una chiave privata è un codice crittografico che permette agli utenti di inviare e ricevere bitcoin.
Effettuare un pagamento è semplice come inviare una e-mail: si scrive l'indirizzo del destinatario, si indica l'importo e si esegue l'operazione.
L' indirizzo è la sola informazione che occorre fornire a qualcuno per poter ricevere un pagamento in Bitcoin; esso è composto da 27-34 caratteri alfanumerici, (tranne 0, O, I, L), ad esempio:

1dice8EMZmqKvrGE4Qc9bUFf9PX3xaYDp

Una chiave privata è invece formata da 51 caratteri.
Se qualcuno ottiene l'accesso alle chiavi di un determinato portafoglio, può gestirne, e di conseguenza rubare, i bitcoin all'interno. Una diversa tipologia di portafogli è quella cartacea: è infatti possibile gestire le proprie monete semplicemente scrivendo o stampando su un foglio di carta la chiave privata

(o le chiavi private) e la chiave pubblica, stando naturalmente attenti che nessuno possa spiare o rubare il foglio. Una buona soluzione ai rischi di furto potrebbe essere quella del *borsellino deterministico*, che permette di generare un numero infinito di chiavi da un unico *codice seme*. Se si utilizza il borsellino deterministico è necessario salvare solamente il codice seme, perché le chiavi e gli indirizzi verranno successivamente generati a partire da questo.

1.3 Come procurarsi bitcoins?

Ci sono svariati modi per procurarsi dei bitcoins: il primo consiste nel vendere un bene o un servizio e farsi pagare in bitcoin. Il secondo nell'accedere a qualche servizio di cambio valuta che possa cambiare i dollari, o gli euro, con bitcoin. Il terzo modo per ottenere bitcoin è quello di partecipare al protocollo di validazione e inserimento delle transazioni nella rete, ottenendo una piccola commissione per ogni transazione che viene trattata. I Bitcoin infatti vengono coniati attraverso una semplice regola: ogni 10 minuti devono essere generati, e di conseguenza inseriti nella rete peer-to-peer, un certo numero di Bitcoin nel mondo attraverso un'operazione, detta *mining*, che serve proprio a validare i pagamenti effettuati dagli altri utenti. Chi risolve per primo un opportuno gioco crittografico vince. Approfondirò questo aspetto nella sezione Processo Mining (Cap. 4.1).

1.4 Block Chain e sistema a blocchi

Nel sistema Bitcoin, ogni qualvolta viene eseguita una transazione, questa viene datata utilizzando un server distribuito. Questo server calcola l'hash dell'oggetto, o dell'informazione che si deve datare, e lo pubblica nella **Block Chain**, il registro generale di tutti i pagamenti fatti con i Bitcoin.
Ogni pagamento è registrato attraverso una o più transazioni; le transazioni sono contenute in blocchi e i blocchi formano la Block Chain.
Ogni blocco è in qualche modo collegato ai precedenti poiché contiene un codice di controllo che dipende dal blocco precedente. Risulta quindi impraticabile falsificare un blocco senza falsificare tutti i successivi blocchi, e i codici di controllo determinano in modo univoco l'ordine cronologico del blocchi, dal momento in cui il sistema è stato creato ed attivato.

Definizione 1.1. In crittografia il termine **Nonce** indica un numero casuale utilizzabile una sola volta (*Nonce* è la contrazione delle parole inglesi *number used once*). Un nonce viene utilizzato nei protocolli di autenticazione per assicurare che i dati scambiati nelle vecchie comunicazioni non possano essere riutilizzati successivamente. I codici di controllo inseriti nei blocchi utilizzano dei valori nonce.

Capitolo 2

Come la crittografia consente lo sviluppo del Bitcoin

L'uso della crittografia risulta essere di importanza cruciale nel sistema Bitcoin.
Oltre al compito di mantenere la segretezza dei dati, nel caso del Bitcoin la crittografia è impiegata per rendere impossibile a chiunque spendere del denaro dal portafogli di un altro utente o alterare la Block Chain.
Può essere anche utilizzata per criptare un portafogli, in modo che non possa essere usato senza una password. Inoltre il sistema Bitcoin utilizza uno schema di firma digitale chiamato Algoritmo di Firma Digitale su Curve Ellittiche (ECDSA) per garantire che le valute possano essere spese solo dai loro legittimi proprietari.
Un sistema basato sulla crittografia a chiave pubblica è inoltre affidabile e sicuro poiché non richiede al venditore di utilizzare informazioni private dell'acquirente, come il suo numero di carta di credito. Le transazioni avvengono attraverso lo scambio di informazioni firmate. Nell' ambito del Bitcoin ci interessa soprattutto che la crittografia usata garantisca le proprietà di autenticazione, integrità e non ripudiabilità. Con autenticazione si intende la sicurezza dell'identità del venditore, con integrità la validità del pagamento ricevuto e infine la non ripudiabilità proibisce al mittente di negare la transazione effettuata.

2.1 Funzioni Hash Crittografiche

Definizione 2.1. Un funzione **hash crittografica** è una funzione

$$h : \{0,1\}^* \longrightarrow \{0,1\}^n$$

che a una stringa m di lunghezza * qualsiasi, data in input, associa in output un **digest di messaggio** $h(m)$ di lunghezza fissata.

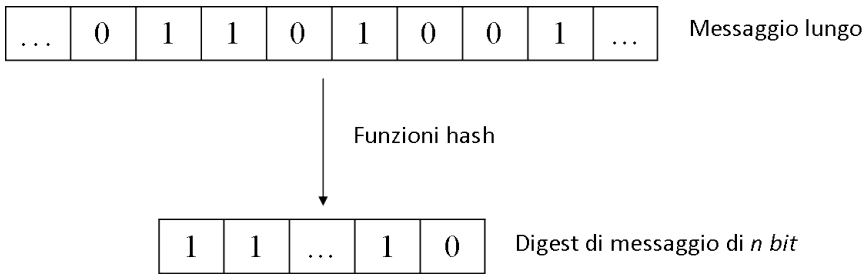

Figura 2.1: Esempio di funzione hash crittografica

Proposizione 2.1.1. *Per le funzioni Hash crittografiche valgono le seguenti proprietà:*

1. *Dato un qualunque m, $h(m)$ deve essere calcolabile in maniera efficace e veloce.*

2. *Unidirezionalità o resistenza alle controimmagini (preimage resistant): dato un qualunque y deve essere computazionalmente impossibile trovare un n tale che: $h(m) = y$.*

3. *Resistenza forte alle collisioni: è impossibile trovare m_1, m_2 tale che $h(m_1) = h(m_2)$ con $m_1 \neq m_2$*

Quando viene trasmessa una informazione attraverso una rete non sicura, è importante che l'informazione non venga corrotta durante la trasmissione. Le funzioni hash crittografiche aiutano a risolvere questo problema, esse infatti tra le varie proprietà hanno quella di non essere iniettive ma, a livello computazionale, non permettere di trovare due controimmagini diverse per una stessa immagine. Uno degli usi principali delle funzioni hash è dunque quello di oscurare alcuni dati in modo da limitare l'uso che se ne può fare. Molti siti web, ad esempio, non memorizzano le password degli utenti nel proprio database, ma

un hash della password. In questo modo il server può verificare che un utente inserisca la password corretta controllando l'hash della password in maniera veloce, ma chiunque tenti di accedere al database delle password visualizza solamente gli hash, inutili allo scopo di carpire le credenziali. Le funzioni hash crittografiche vengono inoltre utilizzate dal sistema Bitcoin per controllare che le transazioni siano eseguite correttamente, poiché esse sono progettate in modo che una minima differenza nell'input cambi in maniera radicale il risultato. Come sarà approfondito nel prossimo capitolo, durante ogni transazione viene firmato digitalmente l'hash della transazione precedente, quindi chi riceve il pagamento può controllare i vari passaggi di proprietà della moneta trasferita verificando le firme presenti in ogni transazione. La modifica di una transazione così non è possibile, poiché gli hash non corrisponderebbero più.
Un altro uso fondamentale delle funzioni hash crittografiche nel sistema Bitcoin è quello fatto nella generazione di nuove monete. Infatti i bitcoin vengono generati continuamente e vengono dati in proprietà a chi riesce a risolvere il problema matematico del calcolo di una controimmagine, descritto poco sopra.

Definizione 2.2. Con il termine SHA, Secure Hash Algorithm, si intende una famiglia di funzioni crittografiche di hash sviluppate dalla *National Security Agency* a partire dal 1993.

Esempio 2.1. Come si può notare in questo esempio esadecimale una minima variazione nel messaggio iniziale produce un cambiamento radicale nell'hash ottenuto in output:

sha256('Bitcoin')=
`b4056df6691f8dc72e56302ddad345d65fead3ead9299609a826e2344eb63aa4`

sha256('bitcoin')=
`6b88c087247aa2f07ee1c5956b8e1a9f4c7f892a70e324f1bb3d161e05ca107b`

Nel sistema Bitcoin vengono utilizzate due funzioni crittografiche hash per la creazione degli indirizzi: SHA-256 e RIPEMD-160. A differenza di SHA-256, RIPEMD-160 è un algoritmo crittografico di hashing che produce un output di un hash di 160 bit, risultando quindi più corto di un hash di 256 bit. Gli algoritmi Ripemd sono stati sviluppati in contrasto a quelli ideati dalla National Security Agency (famiglia SHA), tuttavia Bitcoin li utilizza

entrambi per rendere più stabile e sicuro il sistema. Ripemd viene utilizzato perché produce un hash più breve, mantenendo un elevato livello di sicurezza. La combinazione dei due sembrerebbe quindi essere più forte e resistente agli attacchi sul sistema. In Bitcoin normalmente i calcoli hash sono svolti in due fasi: la prima con SHA-256, e la seconda con SHA-256 o RIPEMD-160, a seconda della lunghezza desiderata.

Esempio 2.2. sha256('bitcoin')=
`6b88c087247aa2f07ee1c5956b8e1a9f4c7f892a70e324f1bb3d161e05ca107b`

sha256(sha256('bitcoin'))=
`a23b7f87e4250b3a64b737f349c06422f752f419cbb25ae9169a6cf1e23f4462`

ripemd160(sha256('bitcoin'))=
`fa00880fdd65c06180c840a41ce88dc22a7fe5a8`

2.2 Gli alberi di Merkle

Definizione 2.3. Un **albero di Merkle** (*Merkle Tree*)o albero di hash è un albero per lo più binario, le cui ramificazioni sono due a ogni passo, in cui ogni nodo è etichettato con l'hash dei suoi nodi figli.

Merkle propose nel 1979 l'idea di un albero di hash che permettesse la verifica efficiente e sicura dei contenuti in esso presente.
La struttura ad albero permette la verifica di un blocco attraverso una quantità di dati proporzionale al logaritmo del numero dei nodi dell'albero. Come si può osservare nella figura sottostante gli hash di tutti i blocchi, presi da una lista, vengono accoppiati ogni volta fino ad ottenere un solo hash, chiamato *top hash.*
Hash 0 è l'hash del risultato della concatenazione *hash 0-0* e *hash 0-1.* Più precisamente, *hash 0 = hash(hash 0-0 + hash 0-1).*
Per verificare l'integrità di un solo blocco è quindi necessario considerare un sottoinsieme dell'albero. Se ad esempio si vuole verificare l'integrità del blocco 1, si devono considerare i nodi *Hash 0-0*, *Hash 0-1*, *Hash 0*, *Hash 1* e *Top Hash*, ma non i nodi *Hash 1-0* e *Hash 1-1.*
Nel particolare sistema Bitcoin ogni foglia dell'albero è costituita da una doppia funzione hash SHA-256, ovvero dall'hash SHA-256 dell'hash SHA-256 di un

determinato oggetto, il block header.

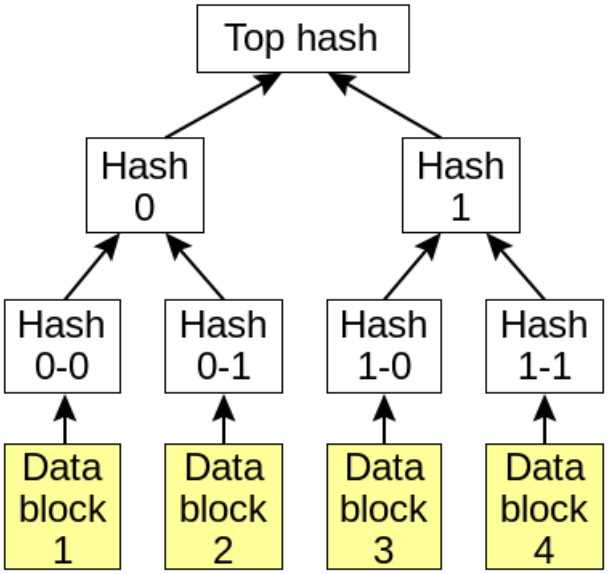

Figura 2.2: Esempio di albero di Merkle

Il vantaggio principale degli alberi di Merkle è che quando il dato di un blocco cambia, non è necessario calcolare l'hash sopra ogni altro dato, ma si calcolano solamente i nodi lungo il ramo, fino al nodo radice. Così facendo il numero di calcoli hash richiesto diminuisce logaritmicamente sul numero di blocchi di dati totali. Tale processo risulta essere particolarmente efficiente quando vi sono molte ramificazioni dell'albero.

2.3 Crittografia a chiave pubblica

I metodi di cifratura e decifrazione si possono suddividere in due grandi categorie: a chiave simmetrica e a chiave asimmetrica, anche detti a **chiave pubblica**. Nella crittografia a chiave simmetrica il mittente invia al destinatario un messaggio cifrato su un canale insicuro e una chiave di decifrazione, attraverso un canale sicuro. La validità della crittografia simmetrica è perciò riposta nella segretezza della chiave utilizzata e nella sicurezza del canale usato per scambiarla.
Dal 1976 comincia dunque a svilupparsi la tecnica della crittografia a chiave pubblica, per ovviare ad alcuni problemi frequenti come ad esempio lo scambio di messaggi fra persone che si trovano a grandi distanza l'una dall'altra, e che non possono scambiarsi direttamente una chiave segreta.
Supponiamo che Alice voglia inviare un messaggio a Bob che si trova in un'altra città e che non possiede una chiave di decifrazione. Bob decide di inviare ad Alice una scatola e un lucchetto, aperto, della cui chiave è l'unico proprietario. Alice mette il proprio messaggio nella scatola, la chiude con il lucchetto e rimanda la scatola a Bob. Supponiamo vi sia poi Eva, che vuole intercettare i messaggi di Alice e Bob: Eva potrebbe tentare di intercettare la prima trasmissione e sostituire il lucchetto con un altro di cui lei stessa ha la chiave; al ritorno può quindi aprire il lucchetto e leggere il messaggio di Alice. Questo problema si può risolvere grazie alla firma digitale, che approfondiremo nella sezione successiva.

Nel sistema Bitcoin viene utilizzata la crittografia in chiave pubblica per creare un paio di chiavi che controlla l'accesso ai bitcoins. Le chiavi consistono in una chiave privata, da cui si ricava un' unica chiave pubblica. La chiave pubblica serve per ricevere bitcoins, la chiave privata viene utilizzata per firmare le transazioni che permettono di spendere quei bitcoins. Tali chiavi vengono generate insieme, quella pubblica può essere diffusa pubblicamente, mentre quella privata deve rimanere a conoscenza del solo proprietario. La loro principale caratteristica è che ciò che viene cifrato con la chiave pubblica può essere decifrato solo con la chiave privata, e viceversa. In una transazione Bitcoin le monete appartengono a chi possiede la chiave privata, mentre la chiave pubblica corrisponde a un hash dell'indirizzo del ricevente. La chiave privata K_{priv} è un numero che viene generato casualmente, da essa, attraverso

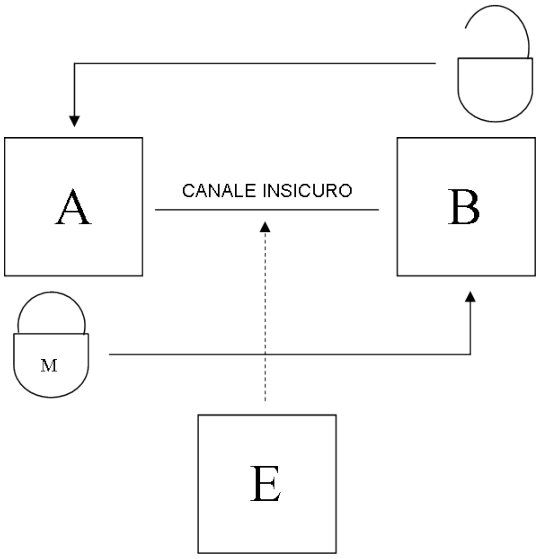

Figura 2.3: Scambio di informazioni attraverso un canale insicuro

la moltiplicazione su una specifica curva ellittica, una operazione crittografica unidirezionale, viene generata una chiave pubblica K_{pub}. Infine dalla chiave pubblica si genererà un indirizzo A, sfruttando le funzioni unidirezionali hash crittografiche.

$$K_{priv} \xrightarrow{\texttt{Moltiplicazione su curva ellittica}} K_{pub} \xrightarrow{\texttt{Funzione hash crittografica}} A.$$

Supponiamo ancora che vi siano Alice e Bob che vogliono scambiarsi informazioni in modo sicuro, possedendo entrambi un algoritmo che genera la coppia di chiavi pubblica e privata. Entrambi devono mantenere segreta la propria chiave privata ma possono condividere pubblicamente la chiave pubblica. Le chiavi hanno una speciale relazione matematica che permette ad Alice di cifrare un messaggio M usando la chiave pubblica di Bob K_{pub}, che solo la chiave privata di Bob, K_{priv}, può decifrare. Se C è quindi il messaggio cifrato, risulta:

$$C = encrypt(M, K_{pub})$$
$$M = decrypt(C, K_{priv})$$

Questo meccanismo si ottiene utilizzando funzioni matematiche che sono computazionalmente impossibili da invertire, come ad esempio le funzioni hash crittografiche. Uno fra i più conosciuti algoritmi di cifratura e decifrazione in chiave pubblica è chiamato RSA, dai suoi inventori Rivest, Shamir e Adleman. Tale algoritmo non viene però utilizzato nel sistema Bitcoin, a favore invece

dell' ECDSA (*Elliptic Curve Digital Signature Algorithm*), che offre gli stessi livelli di sicurezza di RSA, ma utilizzando minor spazio e chiavi più brevi.

2.4 Firma digitale

Generalmente nei processi crittografici la chiave pubblica viene utilizzata per cifrare e la chiave privata per decifrare.
Se avviene l'inverso, ovvero la chiave privata viene utilizzata per cifrare un messaggio che solo la chiave pubblica corrispondente può decifrare, si ottiene un algoritmo che può essere utilizzato per creare una firma digitale, che dimostra che un particolare individuo ha inviato il messaggio.
L'idea della firma digitale nasce infatti dal bisogno di verificare pagamenti e transazioni che avvengono durante il commercio o lo scambio di documenti elettronici.

Definizione 2.4. Con **Firma Digitale** si intende un coppia di numeri (r, s), di cui r rappresenta un nonce, e l'altro è generato con la chiave privata del firmatario. Attraverso la chiave pubblica è possibile verificare, grazie ad un algoritmo matematico definito a partire dal sistema con cui si lavora, se la firma digitale è stata correttamente prodotta.

Osservazione 1. Come una firma manuale, una firma digitale per essere tale deve soddisfare le seguenti caratteristiche:

1. **Autenticità**: il destinatario di un messaggio firmato può verificare che il messaggio non è stato inviato da un impostore.

2. **Integrità**: il messaggio non è stato manomesso.

3. **Non ripudiabilità**: una volta inviato il messaggio, non si può negare di averlo fatto.

Lo schema di firma digitale utilizzato dal sistema Bitcoin è l'ECDSA, che risulta essere una implementazione del più generico algoritmo **DSA**, Digital Signature Algorithm, sviluppato nel 1991 dal National Institute of Standards and Technology.
La firma è applicata a un digest di messaggio: prodotto da una funzione hash con un output di 160 bit, a partire dal messaggio, che indichiamo con m. Per generare le chiavi si applica il seguente algoritmo:

1. Il mittente trova un numero primo q, lungo 160 bit, e sceglie un primo p tale che $q|(p-1)$
2. Sia g una radice primitiva mod p e sia $\alpha \equiv g^{(p-1)/q} \pmod{p}$, così che $\alpha^q \equiv 1 \pmod{p}$.
3. Il mittente sceglie un numero segreto a tale che $1 \leq a < q-1$, successivamente calcola $\beta \equiv \alpha^a \pmod{p}$.
4. Il mittente pubblica (p, q, α, β) e tiene segreto a.

A questo punto il mittente firma il messaggio m in questo modo:

1. Sceglie un numero casuale k tale che $0 < k < q-1$.
2. Calcola $r \equiv (\alpha^k \pmod{p}) \pmod{q}$.
3. Calcola $s \equiv k^{-1}(m + ar) \pmod{q}$.
4. Ottiene la firma (r, s), che invia al destinatario insieme a m.

Il destinatario può quindi verificare la firma digitale svolgendo le seguenti operazioni:

1. Scarica la chiave pubblica del mittente (p, q, α, β).
2. Calcola $u_1 \equiv s^{-1}m \pmod{q}$ e $u_2 \equiv s^{-1}r \pmod{q}$.
3. Calcola $v = (\alpha^{u_1}\beta^{u_2} \pmod{p}) \pmod{q}$.
4. Verifica che $v = r$.

Osservazione 2. La procedura di verifica funziona per definizione di s.

Dimostrazione.

$$s \equiv k^{-1}(m + ar) \pmod{q}$$

da cui ricavo che

$$m \equiv (ks - ar) \pmod{q}$$

e quindi moltiplicando per s^{-1}

$$s^{-1}m \equiv (k - ars^{-1}) \pmod{q}.$$

Ne viene

$$\begin{aligned} k &\equiv s^{-1}m + ars^{-1} \pmod{q} \\ &\equiv u_1 + au_2 \pmod{q}. \end{aligned}$$

Quindi

$$\begin{aligned} r &\equiv (\alpha^k \pmod{p}) \pmod{q} \\ &\equiv (\alpha^{u_1+au_2} \pmod{p}) \pmod{q} \\ &\equiv (\alpha^{u_1}\beta^{u_2} \pmod{p}) \pmod{q} \equiv v \end{aligned}$$

□

Per quanto riguarda il sistema Bitcoin, per evitare che durante una transazione si subisca una frode viene eseguito questo algoritmo:

Per inviare un messaggio firmato M:

1. Si esegue l'hash del messaggio:

$$H = sha256(M)$$

2. Si cifra H con la chiave privata, per ottenere la firma:

$$S = encrypt(H, K_{priv})$$

3. Si invia la firma S insieme al messaggio M.

Per verificare che la firma S sia valida per il messaggio M:

1. Si esegue l'hash del messaggio M:

$$H = sha256(M)$$

2. Si decifra S con la chiave pubblica:

$$H' = decrypt(S, K_{pub})$$

3. Si verifica che $H = H'$. Se risultano uguali la firma è valida.

Come si può osservare, i messaggi nel sistema Bitcoin vengono inviati non criptati. Ciò deriva dal fatto che Bitcoin rende pubblica nella BlockChain ogni transazione e le informazioni che la riguardano.

2.5 Curva ellittica Secp256k1

Nel sistema Bitcoin lo schema di firma digitale utilizzato è l' **Elliptic Curve Digital Signature Algorithm**, o più semplicemente **ECDSA**.

Definizione 2.5. Una **curva ellittica** è un insieme del tipo $E = \{(x, y) \in K^2 t.c.\ f(x, y) = 0\} \cup \{\theta\}$, dove θ rappresenta il punto all'infinito e la funzione è della forma $y^2 - x^3 + ax + b$, che varia al variare di a e b.

La curva usata dal sistema Bitcoin pone $a = 0$ e $b = 7$, divenendo così l'equazione $y^2 = x^3 + 7$, chiamata **Secp256k1**, secondo la classificazione *Standards for Efficient Cryptography* (SEC). Tale curva sul piano reale è rappresentata nel seguente modo: Il sistema Bitcoin utilizza Secp256k1 definita sul

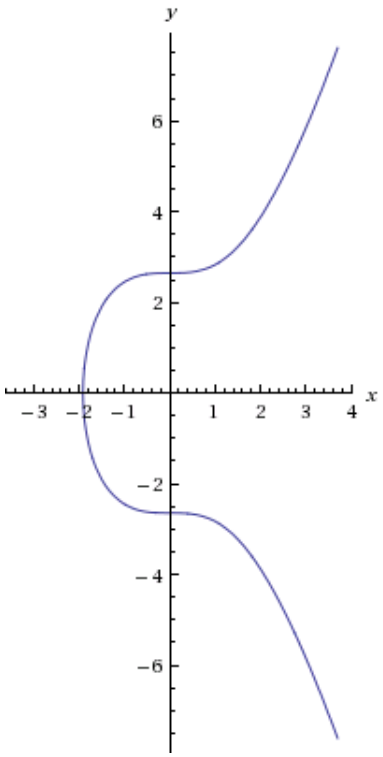

Figura 2.4: Curva Secp256k1 rappresentata sul piano reale

campo $\mathbb{Z}_{2^{256}-2^{32}-977}$, nel quale le coordinate x e y sono interi a 256-bit modulo un primo molto grande.
La curva Secp256k1 non è quasi mai stata utilizzata prima dell'avvento del Bitcoin, ma ora sta guadagnando popolarità grazie alle sue diverse proprietà. A differenza della maggior parte delle curve ellittiche comunemente usate, che

hanno una struttura "casuale", Secp256k1 è stata costruita in modo da consentire uno svolgimento più efficiente dei calcoli. Se l'implementazione è sufficientemente ottimizzata risulta essere più veloce del 30% rispetto alle altre curve, inoltre, a differenza delle curve proposte dal NIST, le costanti a e b di secp256k1 sono state selezionate in modo da ridurre significativamente la possibilità che il creatore della curva possa aver inserito qualche tipo di backdoor. Oltre a dei parametri a e b ve ne sono altri quali il modulo (primo) del campo, un punto base che si trova sulla curva e l'ordine del punto base, che non è scelto casualmente ma è una funzione dipendente dagli altri parametri. Bitcoin utilizza dei parametri molto grandi, la sicurezza dell'algoritmo si basa su questi valori che risultano difficili da attaccare attraverso un attacco a forza bruta. Nel caso del Bitcoin:

- **Modulo (primo)**: $2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$ = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFC2F (in esadecimale)

- **Punto Base G**:
  x = 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798
  y = 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8

- **Ordine**: FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141

2.5.1 Generazione delle chiavi pubblica e privata con ECDSA

Nell'algoritmo ECDSA la chiave privata è un numero, impossibile da prevedere, scelto casualmente fra 1 e $n - 1$, dove n è l'ordine di G. Tale numero deve avere lunghezza 256 bit, per cui viene eseguito l'hash SHA-256 che ha come output una stringa di lunghezza 256 bit. Se il risultato è inferiore a $n-1$ abbiamo una chiave privata adeguata, altrimenti si cerca di nuovo con un altro numero casuale. La chiave pubblica si deriva dalla seguente equazione:

$$K_{pub} = K_{priv} * G$$

e questo implica che il numero delle possibili chiavi private è uguale all'ordine.

Definizione 2.6. Si definisce **somma fra due punti** P, Q su una curva ellittica il punto: $P + Q = R$ dove, dati $P = (x_P, y_P)$, $Q = (x_Q, y_Q)$, $R = (x_R, y_R)$, vale che:

$$x_R = m^2 - x_P - x_Q$$
$$y_R = m(x_P - x_R) - y_P$$

con

$$m = \frac{(y_Q - y_P)}{(x_Q - x_P)}$$

Definizione 2.7. Si definisce punto doppio il punto $2P = P + P = R$ dove, dati $P = (x_P, y_P)$ e $R = (x_R, y_R)$, vale che:

$$x_R = m^2 - 2x_P$$
$$y_R = m(x_P - x_R) - y_P$$

con

$$m = \frac{(3x_P^2 + a)}{2y_P}$$

In pratica, la creazione della chiave pubblica è suddivisa in una serie di operazioni raddoppio e somma di punti, partendo dal punto base.

Esempio 2.3. Per capire meglio come avviene la generazione delle chiavi si propone di seguito un esempio sulla curva Secp256k1, scegliendo:
Modulo: 67
Punto base G: (2,22)
Ordine: 79
Chiave privata: 2
poiché è stata scelta una chiave privata fra le più semplici è richiesta una sola operazione punto doppio a partire dal punto base. Per prima cosa si cerca la chiave pubblica cominciando a calcolare il valore m:
$m = \frac{3*2^2+0}{2*22} \bmod 67$
$m = \frac{12}{44} \bmod 67$
con $44^{-1} = 32 \bmod 62$; dunque risulta:
$m = 12 * 32 \bmod 67 = 384 \bmod 67 = 49$
Con m ricavo:
$x_R = (49^2 - 2 * 2) \bmod 67 = (2401 - 4) \bmod 67 = 2397 \bmod 67 = 52$
$y_R = (49 * (2 - 52) - 22) \bmod 67 = (-2450 - 22) \bmod 67 = -2472 \bmod 67 = 7$
La chiave pubblica risulta quindi essere $(52, 7)$.

A partire dalla chiave pubblica risulta essere computazionalmente impossibile dedurre la chiave privata, soprattutto se vengono utilizzati dei valori iniziali molto grandi.

2.5.2 Firmare un documento con la chiave privata

Una volta ottenute le chiavi privata e pubblica si può procedere con la firma del documento. Solitamente il primo step è quello di calcolare l'hash del documento, per generare un numero della stessa lunghezza (in bit) dell'ordine della curva; ora, per semplicità, omettiamo questo passaggio e firmiamo il documento z, dopo aver assegnato G punto base, n l'ordine e d chiave privata.

- Si sceglie un $k \in 1, \ldots, n-1$.
- Si calcola $(x, y) = k * G$.
- Si trova $r = x \bmod n$: se $r = 0$ si deve scegliere un nuovo k.
- Si trova $s = \frac{(z+r*d)}{k} \bmod n$: se $s = 0$ si cambia k.
- La firma risulta quindi essere è la coppia (r, s).

Esempio 2.4. A partire dai dati calcolati nell'esempio precedente, si calcola ora la firma digitale, ponendo:
$z = 17$
$n = 79$
$G = (2, 22)$
$d = 2$ (chiave privata)
Si sceglie $k = rand(1, 79-1)$, sia $k = 3$, allora risulta:
$(x, y) = 3G = G + 2G = (2, 22) + (52, 7) = (62, 63)$.
Si trova r =62 mod 79 = 62.
Si calcola $s = \frac{(17+62*2)}{3} \bmod 79 = 47 \bmod 79 = 47$. La firma digitale risulta quindi essere la coppia (r, s)=$(62, 47)$.

2.5.3 Verifica della firma digitale con la chiave pubblica

Una volta venuti in possesso del documento e della firma su esso, una terza parte che possiede la chiave pubblica del mittente può verificare l'autenticità del mittente nel seguente modo, chiamando Q la chiave pubblica e scegliendo le altre variabili come prima:

si verifica che r, s siano compresi fra 1 e $n-1$.
Si calcola $w = s^{-1} \bmod n$.
Si calcola $u = z * w \bmod n$.
Si calcola $v = r * w \bmod n$.
Si calcola il punto $(x, y) = uG + vQ$.
Si verifica che $r = x \bmod n$, in caso negativo la firma non risulta valida.

Dimostrazione. Si dimostra la correttezza dell'algoritmo.
Posto G punto base, Q chiave pubblica, d chiave privata:

$$\begin{gathered}(x, y) = uG + vQ = uG + v(dG) = \\ = uG + vdG = (u + vd)G = \\ = (zs^{-1} + rs^{-1}d) = (z + rd)s^{-1}G = \\ = (z + rd)(z + rd)^{-1}k^{-1}G = kG\end{gathered}$$

□

Esempio 2.5. Continuando con l'esempio precedente in cui:
$z = 17$
$(r, s) = (62, 47)$ (firma digitale)
$n = 79$ (ordine)
$G = (2, 22)$ (punto base)
$Q = (52, 7)$ (chiave pubblica)
verificato che r ed s siano compresi fra 1 e $n-1$, si calcola:
$w = s^{-1} \bmod n = 47^{-1} \bmod 79 = 3$
$u = zw \bmod n = 17 * 37 \bmod 79 = 629 \bmod 79 = 76$
$v = rw \bmod n = 62 * 37 \bmod 79 = 2294 \bmod 79 = 3$ da cui si ricava il punto $(x, y) = uG + vQ$, infatti:
$uG = 76G = 2(2(G + 2(G + 2(2(2G))))) = (62, 4)$
$vQ = 3Q = Q + 2Q = (11, 20)$
da cui ricavo: $(x, y) = uG + vQ = (62, 4) + (11, 20) = (62, 63)$.
Poiché $62 = 62 \bmod 79$ si conclude che la firma è autentica.

Capitolo 3

Nella struttura del Bitcoin

Bitcoin è costruito come una rete peer-to-peer di computer, chiamati *nodi*. I nodi sono responsabili dell'elaborazione delle transazioni e del mantenimento delle loro registrazione nel registro elettronico, la Block Chain.
Chiunque può scaricare dalla rete il software open-free Bitcoin e diventare un nodo. Tutti i nodi sono trattati in modo equivalente ma, come già accennato, al fine del corretto funzionamento del sistema la maggioranza della potenza dei computer deve rimanere sotto il controllo di nodi onesti.
Le transazioni vengono registrate cronologicamente nella Block Chain, un registro pubblico online che, a differenza di un tipico registro bancario, non contiene i bilanci di ogni account, ma registra un proprietario per ogni somma di denaro scambiata. Per spendere Bitcoin il proprietario preleva i soldi da una transazione precedente, di cui è stato beneficiario, e li assegna a qualcun altro.

3.1 Indirizzi e chiavi

Chiunque voglia ricevere un pagamento, o effettuarne uno, deve per prima cosa creare una coppia di chiavi pubblica e privata.
Ogni utente che partecipa alla rete Bitcoin crea, grazie al client, una chiave privata casuale di 256 bit. Questa servirà in seguito per firmare la transazione e trasferire bitcoins al nuovo proprietario. E' di fondamentale importanza mantenere segreta la propria chiave privata, poiché chiunque ne venisse in possesso la potrebbe usare per spendere denaro.
Da questa chiave si genera successivamente la chiave pubblica di 512 bit, applicando l'algoritmo ECDSA a 512 bit. Tale chiave viene usata da chi riceve

il pagamento per verificare che la firma digitale del mittente sia autentica. E' importante notare che nel sistema Bitcoin, a differenza della maggior parte dei sistemi in cui la chiave pubblica è resa tale appena viene creata, non viene rivelata fino a che la transazione non è stata firmata dal mittente. La chiave pubblica del mittente rappresenta anche il suo indirizzo, che viene condiviso con gli altri utenti della rete per eseguire transazioni. La chiave che viene creata con ECDSA risulta però molto lunga e ciò può causare problemi nella sua trascrizione o nell'utilizzo. Per accorciarla e renderla più facilmente accessibile o memorizzabile (nel caso in cui si utilizzi il metodo *paper-wallet*), si esegue un doppio hash: per prima cosa si calcola *SHA-256(PubKey)*, e successivamente si esegue *RIPEMD-160(SHA-256(PubKey))*; infine ogni chiave viene codificata in formato $ASCII$ utilizzando l'algoritmo $Base58Check$ che manda in output un indirizzo del tipo:

$$1NDpZ2wyFekVezssSXv2tmQgmxcoHMUJ7u$$

costituito da caratteri standard, e si può notare che risulta impossibile determinare la chiave pubblica o la chiave privata a partire da esso. Gli indirizzi degli utenti Bitcoin sono solamente degli hash delle loro chiavi pubbliche. Ci si potrebbe chiedere perché non usare direttamente la chiave pubblica stessa, tuttavia non solo gli indirizzi generati sono più corti delle chiavi pubbliche, ma risultano essere anche più sicuri contro gli attacchi crittografici.

3.2 Transazioni

Definizione 3.1. Una **transazione** è un trasferimento di denaro da un indirizzo A ad un altro B.

Ogni volta che un proprietario di Bitcoin invia a qualcuno delle monete crea un messaggio, la *transazione*, in cui è presente la chiave pubblica del nuovo proprietario e l'importo da inviare, poi firma digitalmente tale messaggio con la propria chiave privata. Si può quindi dire che quando un utente A trasferisce della moneta all'utente B rinuncia alla sua proprietà aggiungendo l'indirizzo di B sulle monete in oggetto e trasmette al nuovo proprietario la proprietà delle monete attraverso la rete peer-to-peer. Il resto dei nodi della rete validano le firme crittografiche e l'ammontare delle cifre coinvolte prima

di accettarla definitivamente. Poiché ogni transazione è inserita nella Block Chain, è sempre possibile stabilire chi effettua una transazione, verso chi e in quale momento.
Vi sono due tipi di transazioni: la transazioni **P2PKH**, Pay-to-PubKey-Hash, e le transazioni **Coinbase**. Le prime permettono il trasferimento da un indirizzo all'altro; d'altra parte, grazie alle transazioni Coinbase, nuovi Bitcoin vengono introdotti nel sistema, fino al raggiungimento di circa 21 milioni. Le strutture di entrambe le transazioni sono le stesse, tuttavia vedremo che si differenziano per alcune importanti caratteristiche.

3.2.1 Transazioni P2PKH

Supponiamo che Alice, dopo aver ricevuto un pagamento da qualcuno, voglia inviare un pagamento a Bob, che a sua volta invierà il denaro a un terzo utente, usando semplicemente un paio di chiavi crittografiche. Questo è un tipico esempio di transazione P2PKH. In una transazione Pay-to-PubKeyHash il mittente trasferisce bitcoins a un indirizzo P2PKH.
Per prima cosa Bob deve generare il paio di chiavi (di cui una pubblica e una privata), come specificato nella sezione precedente. Successivamente Bob invia l'hash della chiave pubblica (ovvero il suo indirizzo) ad Alice, che può così creare la transazione. Alice crea un messaggio contenente le istruzioni che permettono a Bob, e a chiunque possiede la chiave privata corrispondente all'hash della chiave pubblica di Bob, di spendere la somma indicata. Una volta che Alice trasmette la transazione, questa viene inserita nella Block Chain e, non appena sarà stata convalidata dalla rete, Bob sarà in grado di spendere a sua volta quella somma. La transazione è costituita da due componenti principali:

- **Input**: le informazioni che si riferiscono alla transazione precedente, da cui è prelevata la somma.
- **Output**: le informazioni che riguardano il destinatario del trasferimento di bitcoins.

Una transazione può comprendere più input e output, che vengono rispettivamente registrati in due vettori *vin*, per le transazioni in ingresso, e *vout*, per quelle in uscita. Ogni output viene usato come input nelle transazioni future, in particolare quando una transazione viene usata come input, si conviene che

la transazione stessa sia "spesa", poiché l'intera somma viene mandata all'indirizzo destinatario. Per non perdere alcun Bitcoin, nel caso ad esempio in cui si debba effettuare un pagamento di entità minore rispetto a quella presente nella transazione input, il client genera un nuovo indirizzo Bitcoin del proprietario, e invia la differenza a questo indirizzo. Questo processo viene anche chiamato *change*, ossia "resto".
I proprietari possono inviare anche solo una porzione del valore di una transazione da loro ricevuta: è infatti possibile suddividere 1 BTC in 100.000.000 di parti.
La più piccola unità, 1/100000000, è chiamata *Satoshi*, dal nome del suo inventore.

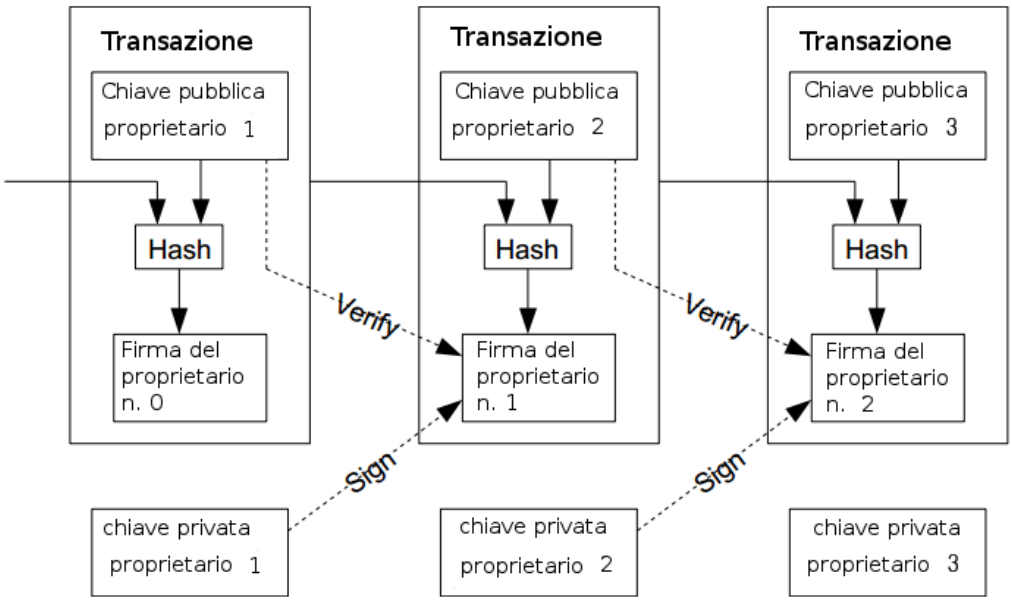

Figura 3.1: Esempio di transazione fra più utenti

Come mostrato in figura, durante ogni transazione il possessore della moneta trasferisce l'importo al destinatario firmando digitalmente un hash della transazione precedente, assieme alla chiave pubblica del prossimo possessore e aggiunge queste firme alla fine della transazione così da provare che è il vero possessore dell'importo. In questo modo solamente il proprietario della chiave privata può creare una firma valida, e ciò garantisce che è l'unico in grado di spendere soldi. Questo meccanismo permette a chi viene pagato di controllare le firme in maniera tale da verificare la catena di proprietà, ma non consente di

garantire che uno dei precedenti possessori della moneta non abbia commesso alcuna azione di double-spending. Questo problema verrà risolto con l'introduzione del sistema di marcatura temporale, che permette di ordinare in modo univoco le transazioni.

3.2.2 Struttura di una transazione

Il diagramma seguente descrive in dettaglio la struttura di una transazione dove A (Alice), invia bitcoins a B (Bob), dopo averli ricevuti da due differenti debitori. La transazione sulla sinistra descrive due inputs (CTxIn) posti nel vettore *vin* e due output (CtxOut), nel vettore *vout*. Il messaggio della transazione crea una lista di input *vin*, in cui ogni posizione ($n = 0$, $n = 1$) si riferisce ad una transazione precedente indirizzata all'attuale proprietario; inoltre viene creata una lista di outputs *vout* che specificano gli indirizzi di uno o più destinatari.

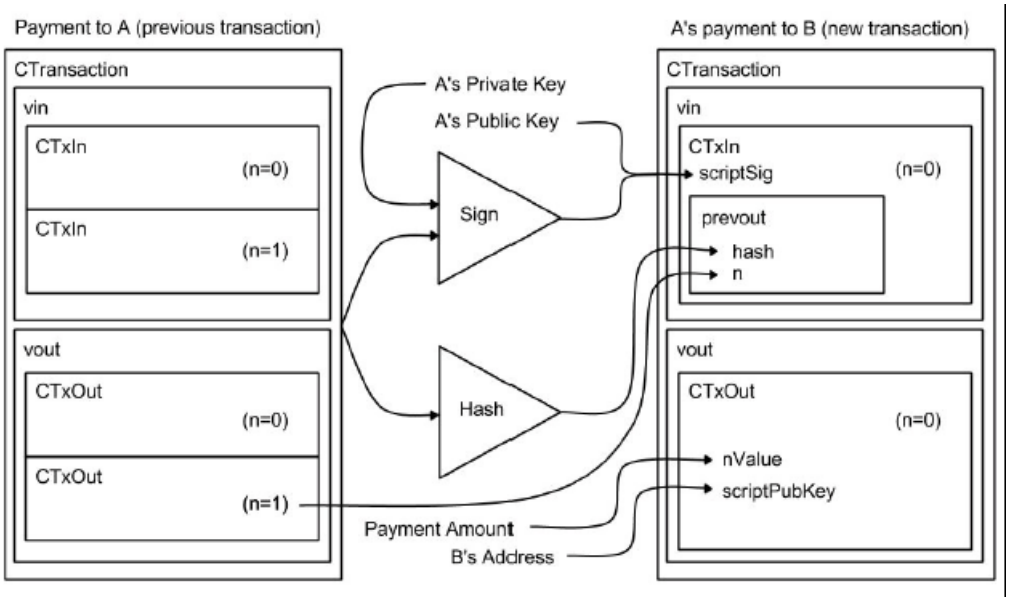

Figura 3.2: Struttura di una transazione

Come si può osservare dalla figura quando Alice decide di spendere i soldi, ricevuti da un precedente debitore, crea un nuovo input *CTxIn* in cui inserisce uno script di firma, **ScriptSig**, contenente la chiave pubblica di Alice e la firma digitale delle transazioni da cui Alice riceve i soldi, generata a partire dalla sua chiave privata. La chiave pubblica permette a Bob, che riceverà il pagamento, di decodificare la transazione ricevuta da Alice, assicurandosi così

che possieda effettivamente il denaro. Inoltre viene inserito nel campo *prevout* un riferimento alla transazione precedente: una tupla $(hash, n)$ in cui il valore *hash* rappresenta il doppio-SHA256 (o SHA-256^2) della transazione precedente. Dunque, mentre una transazione è univocamente identificata dal suo hash, l'output specifico all'interno di quella transazione è identificato dall'indice n. L'output *CTxOut* contiene due valori: nValue e **ScriptPubKey**. *nValue* indica la somma da inviare all'indirizzo ricevente il pagamento, *ScriptPubKey* specifica l'indirizzo del destinatario, ricordando che tale indirizzo è un hash della chiave pubblica che Bob ha provveduto a creare in precedenza.

3.2.3 Verifica di una transazione da parte di un nodo

Nel protocollo Bitcoin, durante la convalida di una transazione, vengono effettuati i seguenti passaggi:

- Si controlla che la sintassi sia corretta.
- Viene rifiutata la transazione se è già presente nella Block Chain.
- Ad ogni input vengono concatenati gli script *scriptSig* e *scriptPubKey* dell'output a cui si riferisce ed è verificata la transazione se il risultato è *Vero*, in altre parole si verifica la firma di Alice sulla transazione precedente.
- Viene rifiutata se alcuni degli input specificati nella trasanzione sono già stati inseriti in altre transazioni (problema del double-spending).

Nel codice Bitcoin si trovano i seguenti script:

```
   scriptPubKey =
OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG

   scriptSig =
<sig> <pubKey>
```

In particolare OP_CHECKSIG considera il primo valore, della lista su cui agisce, una chiave pubblica, il secondo una firma digitale, e verifica che la firma sia autentica utilizzando la chiave pubblica.
Il compito degli script è quello di assicurare che l'hash della chiave pubblica inserita nella transazione che si sta per eseguire corrisponda all'indirizzo di destinazione della transazione precedentemente avvenuta, e questa stessa chiave

appartenga all'utente che firma la transazione.
Affinché sia valida la transazione lo script scriptSig della transazione eseguita deve soddisfare le condizioni dello scriptPubKey nella precedente transazione. Più precisamente lo script scriptSig dato in input e lo scriptPubKey dell'output di riferimento sono valutati (in questo ordine), con scriptPubKey che utilizza i valori restituiti da scriptSig. La transazione in input è autorizzata se scriptPubKey restituisce *True*.
La figura successiva rappresenta come avviene la procedura di verifica di una transazione P2PKH.

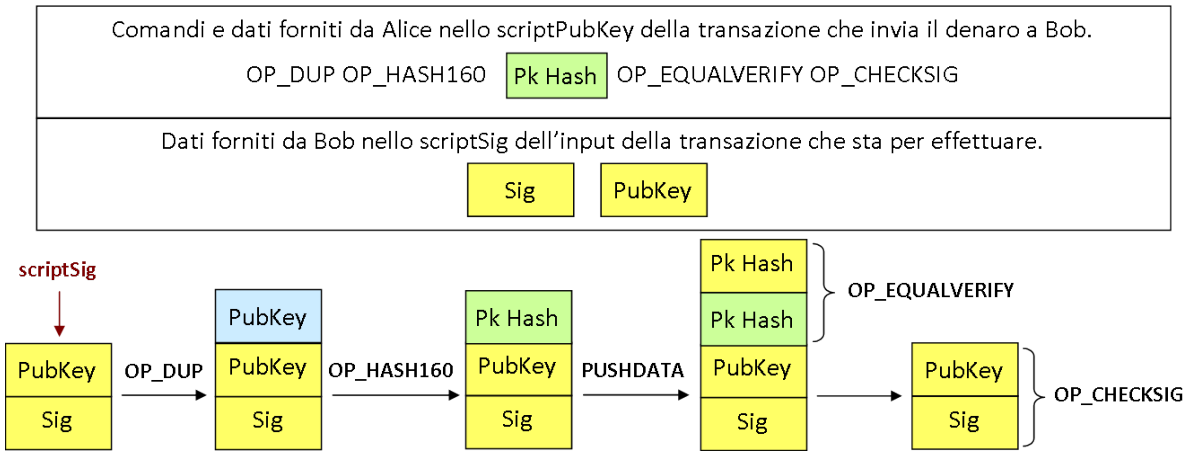

Figura 3.3: Procedura di verifica di una transazione P2PKH

Sig indica la firma digitale della transazione, *pubKey* è la chiave pubblica del mittente Alice, *pubKeyHash* è l'indirizzo da cui giungono i bitcoins (rispetto alla transazione precedente).
Per prima cosa lo script scriptSig crea una pila vuota, successivamente inserisce prima la firma del mittente (di un pagamento da verificare) e sopra questa la chiave pubblica con cui è giunta al destinatario. A questo punto entra in gioco lo script pubKeyHash, che fa riferimento alla transazione precedente, e che esegue il comando OP_DUP, il quale crea una copia della chiave pubblica del mittente (in azzurro) e la pone in cima alla lista; poi esegue OP_HASH160 che effettua un hash della chiave pubblica copiata (in verde) e sopra ad essa, attraverso il comando PUSHDATA, inserisce l'hash della chiave pubblica corrispondente all'indirizzo. Infine attraverso il comando OP_EQUALVERIFY viene controllato che i due hash delle chiavi pubbliche risultino uguali: in caso negativo il comando restituisce *False* e la transazione viene bloccata, in

caso positivo viene eseguito OP_CHECKSIG che convalida definitivamente la transazione se ritorna, a sua volta, *True*. Sostanzialmente viene verificata l'integrità della transazione e l'autenticità della firma che contiene.
Ci si potrebbe domandare perché Alice, quando invia un pagamento a Bob, firma solamente l'assegno ricevuto e non quello che sta per inviare. Il sistema Bitcoin è infatti programmato in modo tale che, una volta inviata una transazione, questa viene inserita in un blocco all'estremità della catena; in altre parole viene distribuita in rete peer-to-peer così che ogni utente che partecipa al protocollo viene a conoscenza di tale transazione. Alice non deve quindi firmare anche lo scriptPubKey (ovvero i dati in output), poiché se Eva, un terzo utente truffatore, volesse rubare i soldi destinati a Bob, dovrebbe andare a modificare lo scriptPubKey inserendo il proprio indirizzo al posto di quello di Bob. "Possedere" i soldi significa infatti, poterli spendere, ovvero poter creare una transazione che trasferisce il denaro a un nuovo proprietario. E tale transazione, per essere ritenuta valida, dovrebbe contenere come input uno scriptSig che soddisfi le condizioni di scriptPubKey, ovvero contenga una chiave pubblica il cui hash sia esattamente l'indirizzo presente in scriptPubKey. In caso contrario infatti la transazione verrebbe considerata invalida e il blocco in cui è inserita non può essere accettato dalla catena. A tal proposito Eva per modificare scriptPubKey dovrebbe creare un nuovo blocco, all'estremità della catena, dove è inserita la transazione fasulla di Eva, e andrebbe così a creare una fork, forchetta, di cui la rete sarebbe l'unica a decidere su quale blocco continuare la catena. Ogni minatore (utente) onesto considererà valido il blocco contenente la transazione (reale) inviata da Alice poiché è stata la prima ad arrivare, tuttavia, poiché la validità di un blocco è stabilita dalla maggioranza della potenza di calcolo, e considerato che ad ogni livello della catena vi è un solo blocco considerato valido, Eva per far accettare dalla catena il nuovo blocco creato con la transazione fasulla dovrebbe possedere il 51%, o più, della potenza di calcolo totale della rete. Se ciò avvenisse però il sistema di scambio valuta perderebbe di significato. Tale attacco viene chiamato "Attacco al 51%" e verrà spiegato nel capitolo 4.

3.3 Il problema del Double-Spending

Tutti i sistemi di pagamenti elettronici fanno grande affidamento sulla crittografia e, più in particolare, sulla firma digitale, sfortunatamente però questa

non risolve il problema del **Double-Spending**.
Il Double-Spending è un tipo di frode digitale nella quale un utente cerca di spendere la stessa cifra due volte, ad esempio inviando contemporaneamente lo stesso pagamento a due destinatari differenti. Per un sistema decentralizzato, come Bitcoin, risulta ancora più difficile da risolvere poiché, a differenza dei sistemi centralizzati, non vi è un ente, ad esempio una banca, che garantisce i pagamenti effettuati.
La soluzione più immediata ed efficace risulta essere quella di considerare valida la sola transazione che per prima viene verificata. La Block Chain non è solo una semplice lista di transazioni, ma è destinato a risolvere il problema del Double-Spending per un network peer-to-peer di nodi potenzialmente non attendibili. Il problema di coordinare un gruppo di nodi con possibili sabotatori è conosciuto come *Problema dei Generali Bizantini.* [1] La Block Chain risolve il problema utilizzando la potenza computazionale dei nodi come sistema di voto.
Per prima cosa i nodi raggruppano le transazioni in *blocchi* che sono connessi fra loro per formare la Block Chain; successivamente i blocchi vengono trasmessi a tutta la rete. Come dimostra la figura sottostante, ogni blocco contiene un hash del blocco precedente nella catena di cui fa parte, il quale deve già essere stato verificato poiché, in caso contrario, il blocco appena aggiunto viene rifiutato dal sistema.

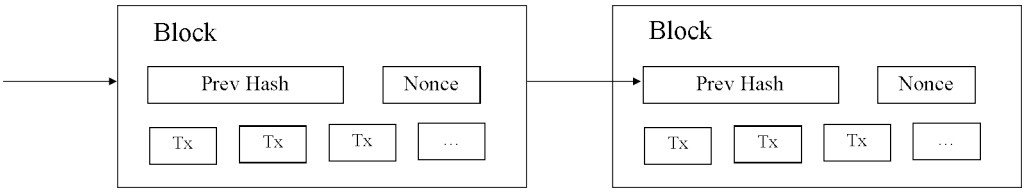

Figura 3.4: Blocchi che formano una catena

Questo permette di stabilire e verificare l'ordine cronologico dei blocchi (attraverso la verifica del valore hash), ma, poiché è possibile a chiunque ag-

[1] Il *Problema dei Generali Bizantini* è un protocollo che permette ai Generali Bizantini, notoriamente non affidabili, di prendere una decisione corretta nonstante la presenza di generali sabotatori. Questo protocollo prevede che la decisione della maggioranza venga ascoltata da tutti i generali, anche quando non tutti sono in grado di comunicare direttamente con gli altri.

giungere un nuovo blocco, un attaccante potrebbe cercare di creare una nuova Block Chain che riassegna i vari proprietari arbitrariamente.
Per prevenire questo problema, in altre parole per rallentare la produzione dei blocchi, i nodi Bitcoin richiedono che una certa *Proof-Of-Work*, prova di lavoro, debba essere risolta per aggiungere un nuovo blocco ad una catena.
Ciò implica che per duplicare una catena, si dovrebbe ripetere la prova di lavoro per ogni blocco al suo interno, richiedendo una quantità eccessiva di potenza computazionale. Un attaccante potrebbe anche tentare di creare un blocco che sostituisce il blocco più recente in una catena, facendo sì che vi siano due blocchi provenienti dallo stesso blocco precedente; può accadere anche perché i diversi nodi ricevono i nuovi blocchi in istanti diversi, a seconda del ritardo introdotto dalla rete: se questo accade, ogni nodo sceglie il blocco che ha ricevuto per primo e lavora cercando di estendere la catena a partire da esso. Il sistema Bitcoin sceglie sempre la catena più lunga, poiché la lunghezza definisce la difficoltà computazionale totale, perciò: dopo un certo numero di nuovi blocchi, la biforcazione viene eliminata.

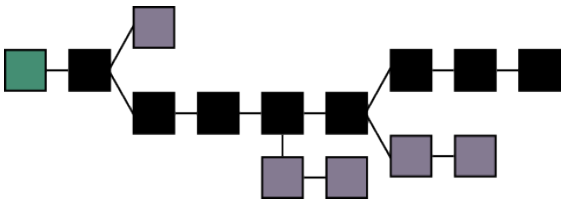

Figura 3.5: Esempio di catena di blocchi

In figura i *blocchi principali*, in nero, rappresentano il ramo accettato da tutta la rete, che è sempre il più lungo. I *blocchi orfani*, in grigio, vengono ignorati perché non sono nel ramo più lungo, ma verranno comunque inseriti in blocchi successivi. Infine il primo blocco della catena, in verde, è chiamato *blocco genesi*.

3.3.1 Conferme

Al fine di evitare il problema del Double-Spending, è necessario eseguire un processo, la conferma della transazione, così che quando i bitcoins vengono ricevuti da un utente, non possono essere spesi immediatamente.

Definizione 3.2. Il processo di inserimento di una transazione in un blocco viene chiamato **conferma della transazione**. Una transazione si dice **confermata** quando possiede 6 o più conferme.

Il client base del Bitcoin visualizza una transazione come"*n/unconfirmed*" (non confermata), fino a che non è stata inserita in 6 blocchi successivi, tuttavia ogni utente può impostare a propria discrezione, e a proprio rischio, il numero di conferme richieste per le transazioni che riceve.
Poiché un blocco viene prodotto ogni 10 minuti circa, l'importo di una transazione che richiede 6 conferme non potrà essere speso prima di un'ora dalla creazione della transazione.
Questo implica che, ad esempio, un negoziante che accetta un pagamento in bitcoin non possa essere certo che il cliente è effettivamente in possesso della somma spesa prima che sia trascorsa circa un'ora.

Capitolo 4

Creare nuovi Bitcoins: Mining e Proof of Work

4.1 Processo di Mining

Definizione 4.1. Il **Mining** è il processo che crea nuovi blocchi, convalidati, e li aggiunge alla Block Chain; i nodi che svolgono il processo di mining sono chiamati **Miners**, minatori.

Ogni miner esegue una serie di passaggi che portano a produrre un nuovo blocco:

1. Raccoglie e decide quali informazioni, raccolte dalla rete peer-to-peer, includere in un nuovo blocco.
2. Verifica che tutte le transazioni che ha incluso nel blocco siano valide.
3. Seleziona il blocco più recente nel ramo più lungo della catena di blocchi e inserisce un hash di tale blocco nel nuovo blocco che sta creando.
4. Cerca di risolvere il proof of work per il nuovo blocco e contemporaneamente osserva quali altri nuovi blocchi vengono creati da altri nodi.
    - Se trova una soluzione al problema del proof of work, il nuovo blocco viene aggiunto alla catena scelta e condiviso in rete.
    - Se un altro nodo risolve prima il problema della prova di lavoro, il suo blocco viene sottoposto ai controlli di validità. Se risulta valido viene aggiunto ad una copia locale della Block Chain e condiviso in rete, altrimenti viene scartato.

Poiché i minatori cercano di creare nuovi blocchi simultaneamente, con la maggior parte del loro tempo impiegato per risolvere il problema del proof of work, accade che ci siano versioni differenti della Block Chain ad un determinato istante. Questo accade perché vengono creati nuovi blocchi con regolarità ogni 10 minuti circa, quindi durante questo periodo di tempo tutti i nodi cercano di risolvere il proof of work.
Quando un nodo trova una soluzione valida guadagna un certo numero di bitcoin in premio, previsti dal protocollo, condivide il blocco in rete e i nodi che ricevono il nuovo blocco, dopo averlo verificato, lo aggiungono alla loro catena, ricominciando a lavorare su un nuovo blocco. In questo modo i nodi si sincronizzano ogni 10 minuti circa.

4.2 Proof of Work

Il problema del Proof of Work è un problema crittografico che richiede un enorme numero di tentativi e può essere risolto solamente tramite il metodo di **forza bruta**[1], il quale rende impossibile prevedere quale utente troverà la soluzione prima degli altri.

Definizione 4.2. Un **Proof Of Work**, è un dato difficile da calcolare poiché deve soddisfare determinati requisiti ed è costoso dal punto di vista del tempo di calcolo richiesto. Inoltre è calcolato con un processo di ricerca casuale che ha bassa probabilità di successo.

Il sistema Bitcoin utilizza una versione della prova di lavoro **Hashcash**, proposta da Adam Back nel 1997, e sfrutta l'apparente casualità delle funzioni hash crittografiche. Un algoritmo converte un dato arbitrario in un numero esadecimale e, se il dato iniziale viene incrementato e risottoposto al calcolo dell'hash, viene prodotto un risultato completamente differente.
Questa operazione viene eseguita fino a che si trova un risultato minore di un valore T, fissato dalla rete.
Più piccolo è tale valore, più difficile e costosa è l'operazione di generazione di un nuovo blocco.

[1]Con metodo *forza bruta* si intende un algoritmo di risoluzione di un problema che consiste nel provare tutte le soluzioni teoricamente possibili fino a trovare quella funzionante.

La soluzione del problema del Proof Of Work si ottiene trovando:

$$x \text{ tale che } H(x) \leq T,$$

dove H rappresenta una funzione hash crittografica.
Il protocollo seguito dai minatori per risolvere il problema è il seguente:

1. Si incrementa (aumentandolo ogni volta di 1) un numero casuale, presente nella intestazione del blocco, che è detto *Nonce*.

2. Si esegue un doppio hash SHA256 di alcune parti del blocco (chiamati, nell'insieme, *block header*[2]) che si vuole aggiungere:

$$\text{SHA256}^2(\textit{block header}).$$

3. Si verifica che l'hash calcolato $H \leq T$:

   - Se il valore è minore l'utente ha risolto il problema e il suo blocco viene aggiunto alla catena.
   - In caso negativo il blocco viene rifiutato dal sistema e ricomincia la prova dal punto 1).

Esempio 4.1. Come si può osservare in questo esempio per ottenere un hash che abbia 000 come prime tre cifre, e sia perciò minore di 2^{244}, a partire da un valore "Hello, world!0", si deve incrementare tale valore 4250 volte per raggiungere l'obiettivo.

```
SHA256(SHA256(Hello, world!0)) =
1312af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64
SHA256(SHA256(Hello, world!1)) =
e9afc424b79e4f6ab42d99c81156d3a17228d6e1eef4139be78e948a9332a7d8
⋮
SHA256(SHA256(Hello, world!4250)) =
0000c3af42fc31103f1fdc0151fa747ff87349a4714df7cc52ea464e12dcd4e9
```

[2]Fra i dati contenuti nel Block Header sono presenti i seguenti valori: *nVersion*, numero della versione di Bitcoin utilizzata, *HashPrevBlock*, hash del blocco precedente, *HashMerkleRoot*, la radice dell'albero di Merkle a cui fa riferimento il blocco, *nNonce*, valore che viene incrementato ad ogni nuovo tentativo.

4.3 Difficoltà di Proof of Work

La difficoltà del Proof Of Work è regolata in modo da limitare la velocità con cui nuovi blocchi possono essere generati dalla rete, e quindi aggiunti alla Block Chain: è importante che i nuovi blocchi non siano creati troppo velocemente, né troppo lentamente.
Se i blocchi fossero creati troppo velocemente la Block Chain conterrebbe molte biforcazioni, che renderebbero difficoltosa ai nodi la scelta del ramo su cui lavorare. D'altra parte se la creazione fosse troppo lenta ci vorrebbe molto tempo per confermare le transazioni. I nodi del sistema Bitcoin si servono del *target* T per regolare automaticamente la velocità di creazione dei blocco, in modo che occorra ogni volta una media di 10 minuti affinché ne venga creato uno nuovo.

Definizione 4.3. Il **target** è un numero a 256-bit condiviso da tutti i client. Il target massimo usato da Bitcoin è
`0x00000000FFFF0000000000000000000000000000000000000000000000000000`,
questo valore rappresenta la più bassa difficoltà di creazione di un blocco che si può riscontrare.

Definizione 4.4. La **difficoltà** D è una misura di quanto sia difficile, dal punto di vista computazionale, trovare un valore di hash minore di un fissato valore target. E' definita come il rapporto fra il massimo target possibile e il target corrente:

$$D = \frac{T_{max}}{T}$$

dove $T_{max} = 65535 * 2^{208} = \texttt{0x00000000FFFF} * 2^{208}$, da cui:

$$D = \frac{65535 * 2^{208}}{T}.$$

La difficoltà viene modificata ogni 2016 blocchi creati (circa 2 settimane), e si basa sul tempo medio usato per creare l'ultimo blocco.
Poiché ogni valore hash è un numero di 256 bits, vi sono 2^{256} possibili valori hash e dunque la probabilità di trovare un valore hash minore di T in una sola volta è:

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

quindi il numero di prove necessarie a trovarlo sarà in media:

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

Se gli ultimi 2016 blocchi sono stati generati in un tempo di circa t_m, con un target T, allora possiamo stimare che i minatori collettivamente hanno creato ad una velocità di:

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

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

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

Il valore di target deve quindi essere regolato ad un valore T' così che $t(T', v_{est})$ $= 600s \equiv 10min$.
Si ricava quindi che:

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

Da cui:

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

4.4 Ricompense

Risolvere il problema del Proof Of Work non comporta solo una spesa di tempo, ma anche di elettricità, che permettere ai computer di elaborare il problema e svolgere continuamente dei calcoli. Per invogliare gli utenti a utilizzare i propri calcolatori per validare i blocchi, il sistema Bitcoin elargisce una certa quantità di BTC a chi produce un blocco valido.

Definizione 4.5. Una **transazione Coinbase**, anche detta *transazione di Generazione*, è la prima transazione che viene inclusa in un blocco, alla sua creazione, ed è utilizzata per pagare una ricompensa ai minatori che per primi risolvono il Proof Of Work.

Tale transazione ha come input un valore casuale che il minatore può utilizzare come valore nonce, come output il valore *null*, nel campo `prevout`, perché non fa riferimento ad alcuna transazione precedente, e uno o più indirizzi a cui,

nel caso in cui il blocco sia accettato nella catena di blocchi, viene assegnata l'intera somma o parte di essa. Una transazione Coinbase non trasferisce bitcoins, bensì ne crea di nuovi: questo è anche l'unico modo per far entrare in circolazione nella rete nuovi bitcoins.
Inizialmente la ricompensa era di 50 bitcoin per blocco, ma se tale ricompensa rimanesse sempre la stessa, la moneta in circolazione aumenterebbe infinitamente nel tempo e si verificherebbe una continua inflazione. Per evitare questo problema il sistema è programmato per generare moneta secondo una serie geometrica, fino a che il numero totale di Bitcoin non giunge a 21 milioni. Bitcoin dimezza la ricompensa ogni 210.000 blocchi minati, ovvero ogni 4 anni circa; così facendo si ottiene che:

$$210.000 * 50 * \sum_{k=0}^{\infty} (\frac{1}{2})^{k} = 210.000 * 100 = 21.000.000$$

Attualmente il totale di BTC prodotti è quasi 14 milioni.

Figura 4.1: Totale dei Bitcoin in circolazione (Dicembre 2014)

4.5 Attacchi al sistema Bitcoin

Il sistema Bitcoin utilizza degli algoritmi crittografici, fra i quali SHA-256 ed ECDSA, considerati fra i più forti al momento. Tuttavia, come ogni software, anche nel Bitcoin vi sono attacchi, sia al protocollo sia alla sua implementazione, che potrebbero compromettere la funzionalità dell'intera rete di scambio.
In particolare Bitcoin è esposto al rischio del double-spending, nel momento in cui una transazione viene creata, e tale rischio che diminuisce sempre più man mano che la transazione guadagna delle conferme.

4.5.1 Attacco al 51%

Uno fra i più conosciuti e potenti attacchi che può subire il sistema è chiamato **Attacco al 51%**. L'attacco al 51% è una forma di attacco al protocollo Bitcoin che consentirebbe, a chi possiede il 51% di tutta la potenza di calcolo (*hashrate*) suddivisa nel sistema, di falsificare e gestire le transazioni a proprio piacimento.

Definizione 4.6. Hashrate è la velocità con cui un calcolatore completa un'operazione nel codice Bitcoin; è calcolata come il numero di hash che un computer, o una rete di computer, può calcolare in un secondo. Esprime la potenza di calcolo che mettiamo al servizio dell'inserimento delle transazioni, quindi esprime anche la probabilità di riuscire a produrre blocchi validi.

Come è stato mostrato precedentemente, la validità di un blocco, contenente più transazioni, dipende solamente dalla rete, poiché, quando appaiono due blocchi contrastanti, è la rete a decidere quale è quello valido.
Ogni minatore "vota" il proprio blocco contenente le transazioni che ha ricevuto, continuando a concatenare nuovi blocchi. Controllando più della metà della potenza di elaborazione della rete, potrebbe decidere quale blocco alla fine della catena viene accettato come vero. Poiché la rete è libera e aperta, se un attaccante utilizzasse durante questa operazione una potenza di calcolo superiore a tutte le altre, nel complesso, riuscirebbe a generare blocchi più veloci rispetto al resto della rete, inserendoli così nella propria forchetta privata. Questo procedimento continuerebbe fino a che la nuova forchetta non diventa più lunga del ramo costruito dagli utenti onesti, e, poiché non vi è alcuna autorità centrale nel sistema Bitcoin, nessuno potrebbe impedire all'attaccante

di farlo, se non superandolo in potenza di calcolo. Tuttavia, se un tale attacco dovesse avvenire con successo, probabilmente la fiducia nel sistema andrebbe perso, e il valore della moneta diminuirebbe rapidamente.

4.5.2 Attacchi a collisione

Un **attacco a collisione** avviene con la ricerca di due stringhe che, date in input, producono lo stesso risultato hash. Questo è teoricamente possibile, dato che le funzioni hash crittografiche prendono messaggi di qualunque lunghezza e li trasformano in stringhe alfanumeriche di lunghezza finita e predeterminata, e perciò non possono essere iniettive.
Quando due ingressi producono lo stesso risultato, si dice che vi è una **collisione**. Se un attaccante cercasse di generare un indirizzo con un hash uguale a quello dell'indirizzo di un altro utente, sia il proprietario dell'indirizzo originale, sia il proprietario dell'indirizzo contraffatto potrebbero spendere le stesse somme di denaro. Tuttavia nella pratica ciò risulta essere computazionalmente molto improbabile, infatti creare una collisione, significherebbe calcolare, in media, $2^{\frac{160}{2}}$ valori hash. Un computer in grado di calcolare 10^{12} indirizzi al secondo (*1 terahash*), tenendo presente che questo valore è teorico ed è molto maggiore di qualsiasi calcolatore attualmente in uso, per calcolare tutti i possibili valori hash servirebbero 1.48×10^{20} secondi, ovvero circa 38309 anni.

Bibliografia

[1] Nakamoto Satoshi. *Bitcoin: A peer-to-peer electronic cash system*, 2009.

[2] Clark Chris. *Bitcoin Internals: A Technical Guide to Bitcoin* [Kindle Edition], 2013.

[3] Carboni Davide. *Dizionario Bitcoin - Italiano: Glossario ragionato sul mondo Bitcoin per i meno esperti* [Kindle Edition], 2014.

[4] Trappe, Wade, and Lawrence C. Washington. *Crittografia: Con Elementi Della Teoria Dei Codici*, Milano: Pearson Education Italia, 2009.

[5] Okupski Krzysztof. *Bitcoin Developer Reference*,
URL: `http://enetium.com/resources/Bitcoin.pdf`, 2014.

Sitografia

[1] Harding David, Sanders Greg. *Bitcoin Developer Guide*,
URL: `https://bitcoin.org/en/developer-guide`.

[2] Rykwalder Eric. *The Math Behind Bitcoin*,
URL: `http://blog.chain.com/post/95218566791/the-math-behind-bitcoin`,
Ottobre 2014.

[3] *Block Chain Info*, URL: `https://blockchain.info/it`.

[4] *Bitcoin Wiki*, URL: `https://en.bitcoin.it/wiki/`.

[5] Perry David. *Bitcoin Attacks in Plain English*,
URL: `http://codinginmysleep.com/bitcoin-attacks-in-plain-english/`,
Ottobre 2012.

[6] *Learn Cryptography*, URL: `http://learncryptography.com/bitcoin/`.

[7] Shirriff Ken. *Bitcoins the hard way: Using the raw Bitcoin protocol*,
URL: `http://www.righto.com/2014/02/bitcoins-hard-way-using-raw-bitcoin.html`,
Febbraio 2014.