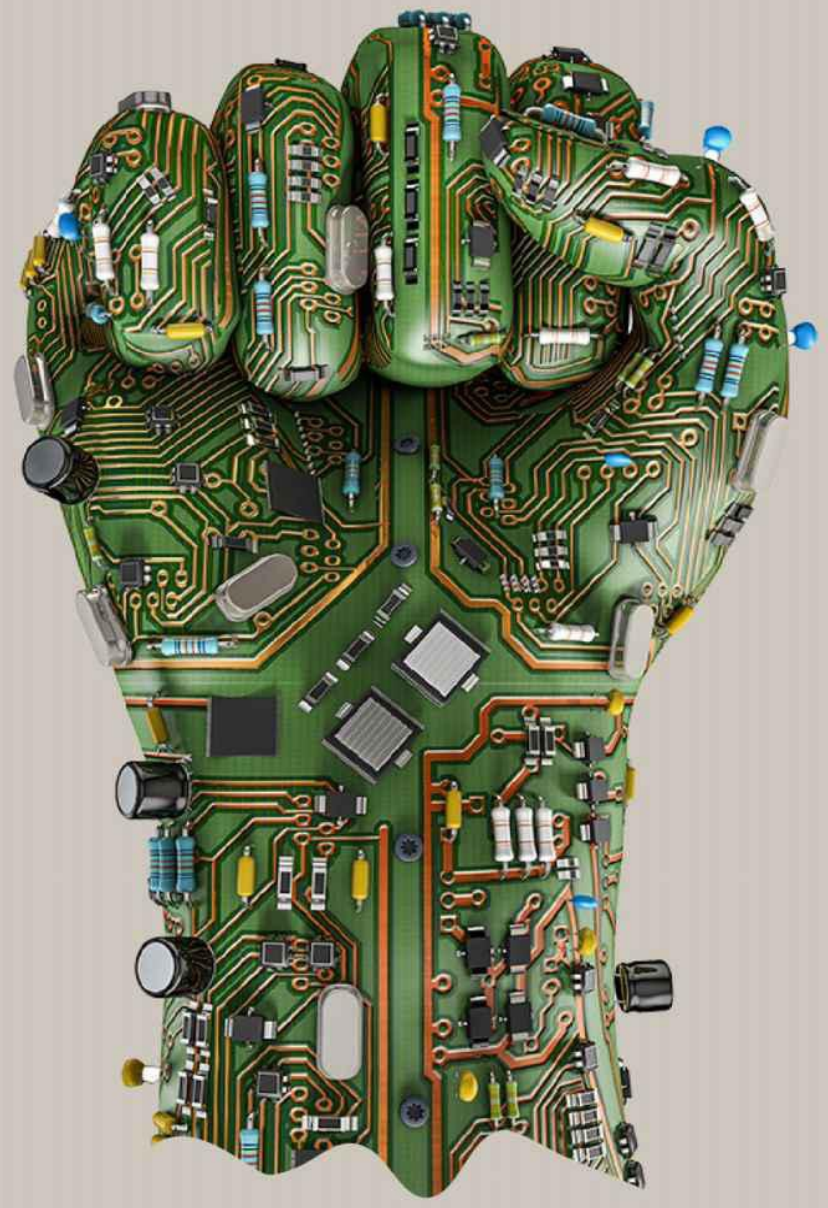
HETERODOXA

BITCOIN MANIFESTO

UNA CPU UN VOTO

Satoshi Nakamoto



ANTONIO TOMBOLINI
EDITORE

SATOSHI NAKAMOTO

Bitcoin Manifesto

Bitcoin Manifesto: UNA CPU UN VOTO

Satoshi Nakamoto

Collana “Heterodoxa”

a cura di Stefano Tombolini

ISBN 978-88-98924-32-5

E-mail:

heterodoxa@simplicissimus.it

www.antoniotombolini.com/heterodoxa

Facebook

Twitter

Immagini e copertina
a cura di Marta D'Asaro

ISBN: 9788898924325

Questo libro è stato realizzato con
StreetLib Write (http://write.streetlib.com)
un prodotto di Simplicissimus Book Farm

Sommario

Prefazione del curatore

Bitcoin: un sistema peer-to-peer di moneta elettronica

Appendice

AttackerSuccessProbability in JavaScript

Note

PREFAZIONE DEL CURATORE

Il presente lavoro si propone di divulgare presso un pubblico quanto più ampio possibile l'opera geniale del misterioso pensatore e innovatore

noto sotto lo pseudonimo di *Satoshi Nakamoto*.

Il lettore forse è già consapevole della complessità della materia: terminologia informatica e commerciale s'intrecciano indissolubilmente. Le note dovrebbero, o quanto meno vorrebbero, agevolare la comprensione ai non addetti ai lavori.

L'autore è a dir poco oscuro, non è stato possibile contattarlo per

chiarimenti… come forma di rispetto, la traduzione è quanto più letterale possibile.

La scelta della forma dell'articolo scientifico per la pubblicazione dei dettagli tecnico-economici dell'innovazione finanziaria rappresentata da *Bitcoin* è stata molto probabilmente dettata dalla formazione culturale e professionale del misterioso autore.

All'atto pratico, il futuristico antimanifesto tecnico che state per

leggere si rivelerà non meno potente e rivoluzionario dei classici manifesti politici, da cui il titolo (un po' ruffiano) di questa edizione italiana.

Satoshi Nakamoto pubblicò *Bitcoin: A Peer-to-Peer Electronic Cash System* sabato 1 novembre 2008 su *The Cryptography and Cryptography Policy Mailing List* di *metzdowd.com*, dove giovedì 8 gennaio 2009 annunciò anche il rilascio della prima versione

sperimentale del programma informatico che implementa l'ambizioso progetto.

Di conseguenza, l'implementazione attuale di Bitcoin può differire, e in effetti differisce, dalla lettera dell'articolo-manifesto in alcuni aspetti pratici non trascurabili (cosa del tutto normale nell'evoluzione di un sistema complesso basato su incentivi), ma i fondamenti tecnico-economici delineati con chiarezza da Satoshi Nakamoto rimangono

essenzialmente inalterati.

Nonostante la difficoltà dei pur interessanti dettagli tecnici, il lettore attento non perderà di vista la differenza chiave tra il sistema finanziario tradizionale, così come lo conosciamo da quasi un millennio, e l'alternativa rivoluzionaria di Bitcoin, coraggiosamente proposta e avviata da Satoshi Nakamoto.

Tradizionalmente, il potere politico centrale, divenuto tale dopo avere guadagnato consenso in

maniera più o meno democratica o autoritaria, autorizza l'attività del sistema bancario e affida a esso la gestione e la conservazione della cronologia delle transazioni finanziarie.

Bitcoin, dal canto suo, non è semplicemente l'ennesima moneta elettronica o un nuovo mezzo di pagamento: Bitcoin rappresenta una piattaforma tecnologica abilitante con incentivi economici tali da permettere a utenti che hanno bisogno

di raggiungere un accordo tra loro (e quindi anche una qualche forma di consenso politico su una determinata materia) di appoggiarsi a una rete decentralizzata di certificatori computazionali.

Il presupposto alla base dell'idea di Satoshi Nakamoto è che un consenso raggiunto secondo il criterio "una CPU un voto" è meno sabotabile rispetto alle modalità di formazione del consenso politico conosciute prima dell'avvento di

Bitcoin.

La "corsa al riarmo" computazionale non solo offre una maggiore trasparenza *all'interno* del sistema, ma garantisce anche la competizione *tra* sistemi: in seguito all'introduzione di Bitcoin è fiorito un intero *ecosistema* di monete elettroniche basate sullo stesso principio ma con variazioni innovative sul tema, le cosiddette *altcoin* (abbreviazione di "monete alternative").

Se Bitcoin mette in discussione il sistema finanziario tradizionale, l'opera intellettuale di Satoshi Nakamoto minaccia di ridefinire ciò che intendiamo normalmente come significato pratico del termine "pubblico": da "autorizzato dal potere politico centrale" a "certificato da una rete computazionale decentralizzata".

Non è solo la cronologia delle transazioni finanziarie a rischiare di essere *disrupted*, ma qualsiasi tipo di

registro informativo pubblicamente rilevante: anagrafe, registro automobilistico, contrattualistica, nomi di dominio, standard *ISO*, marchi di origine, listini azionari, ecc.

I miei più sentiti ringraziamenti per i consigli e il sostegno vanno al mio editore e a Marco Barulli, cofondatore di *Clipperz*, per il supporto sugli aspetti più tecnici di Bitcoin. Clipperz è un servizio web

con cui gestire password e dati sensibili, disponibile qui: http://clipperz.is.

Buona lettura,

Stefano Tombolini

BITCOIN: UN SISTEMA PEER-TO-PEER DI MONETA ELETTRONICA

Abstract

Una variante puramente *peer-to-peer* [1] di valuta elettronica renderebbe possibile l'invio di pagamenti online da un soggetto a un altro senza l'intermediazione di un'istituzione finanziaria. Le *firme digitali* [2] costituiscono parte della soluzione, ma i vantaggi principali verrebbero meno se fosse ancora richiesta la presenza di un terzo fiduciario per prevenire il *double spending* [3]. Il presente lavoro propone

una soluzione al problema del double spending che sfrutta una rete peer-to-peer. La rete marca temporalmente le transazioni applicandovi una funzione *hash* [4] all'interno di una catena continua di *proof-of-work* [5] basate su hash, creando un record che non può essere modificato senza rieseguire la proof-of-work. La catena più lunga non serve solo a provare la sequenza degli eventi certificati, ma garantisce anche la sua stessa provenienza dal maggiore gruppo di potere computazionale.

Fintantoché la quota maggiore di potere computazionale è controllata da nodi che non cooperano in un attacco alla rete, tali nodi genereranno la catena più lunga e precederanno gli attaccanti. La rete in sé richiede un'infrastruttura minima. I messaggi sono trasmessi su base *best effort* [6], e i nodi possono lasciare la rete e riunirvisi in seguito a piacimento, accettando la catena di proof-of-work più lunga come prova di ciò che è avvenuto in loro assenza.

1. Introduzione

Il commercio su Internet ha finito per basarsi quasi esclusivamente su istituzioni finanziarie che fungono da terzi fiduciari per processare i pagamenti elettronici. Anche se il sistema funziona abbastanza bene per la maggior parte delle transazioni, esso soffre ancora delle debolezze intrinseche al modello fiduciario. Transazioni totalmente irrevocabili sono di fatto impossibili, dal momento che le

istituzioni finanziarie non possono evitare di mediare le controversie. Il costo della mediazione aumenta i costi di transazione, ponendo un limite al minimo importo di transazione utilizzabile e impedendo di effettuare piccole transazioni saltuarie, e c'è anche un costo più diffuso in termini di impossibilità di effettuare pagamenti irrevocabili in cambio di servizi irripetibili. Con la possibilità di revoca, la necessità di fiducia si propaga. I commercianti sono costretti a essere

sospettosi nei confronti dei propri clienti, scocciandoli per ottenere più informazioni di quelle di cui avrebbero bisogno altrimenti. Una certa percentuale di truffe è accettata come inevitabile. Questi costi e le incertezze sul pagamento possono essere evitati scambiando valuta fisica di persona, ma non esiste alcun meccanismo per effettuare pagamenti attraverso un mezzo di comunicazione in mancanza di un soggetto fiduciario.

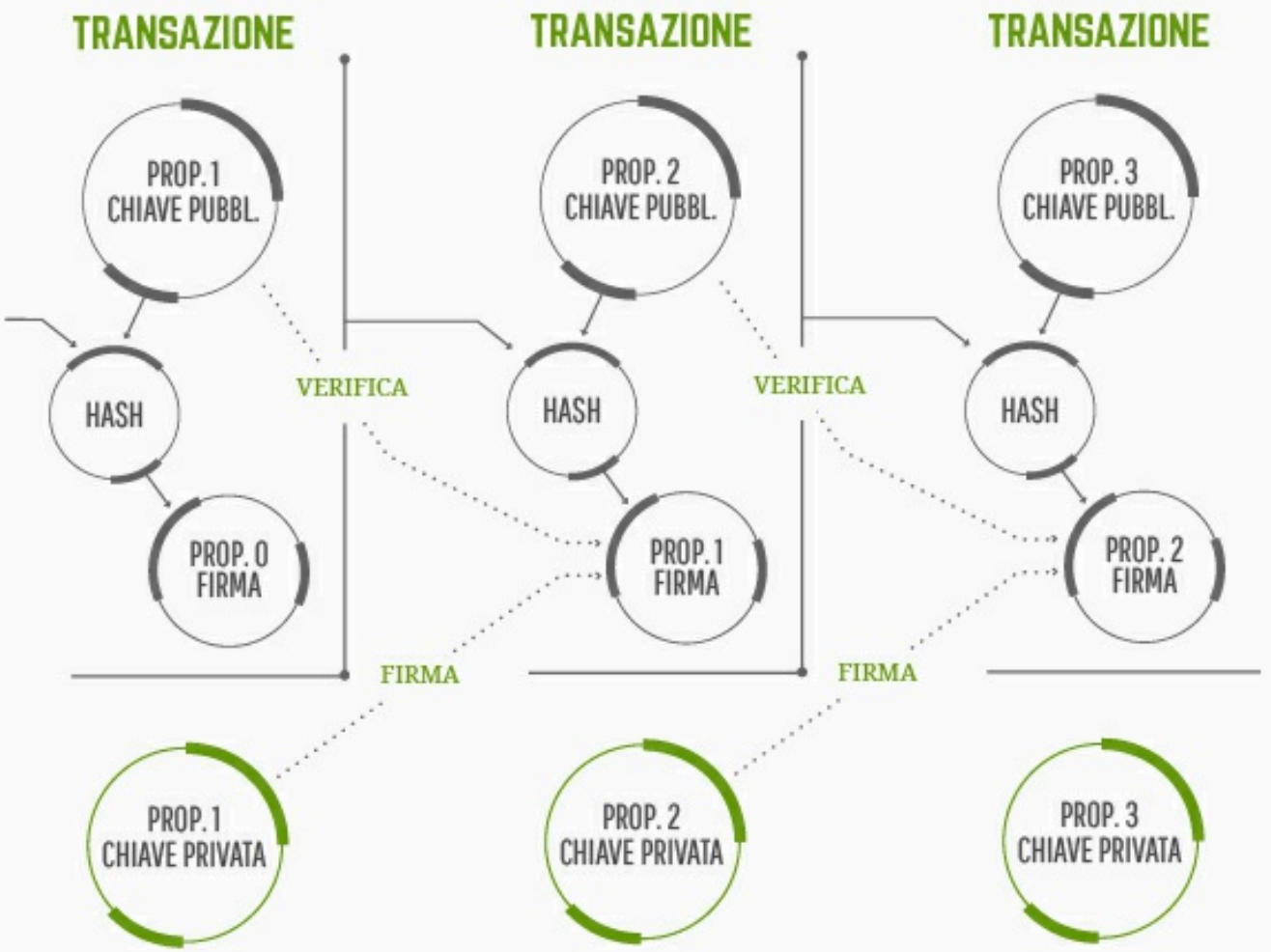
Ciò di cui vi è bisogno è un sistema di

pagamento elettronico basato su certificazione crittografica invece che fiduciaria, che permetta a due soggetti in comune accordo di effettuare transazioni dirette l'uno verso l'altro senza bisogno di un terzo fiduciario. Transazioni computazionalmente esose da annullare proteggerebbero da truffe i venditori, e meccanismi automatici di deposito in garanzia potrebbero essere implementati facilmente per proteggere i compratori. Il presente lavoro propone una soluzione al problema del double spending che

utilizza un server di marcatura temporale, distribuito in peer-to-peer, per certificare computazionalmente l'ordine cronologico delle transazioni. Il sistema è sicuro fintantoché i nodi onesti controllano, collettivamente, più potere computazionale di qualsiasi gruppo cooperante di nodi attaccanti.

2. Transazioni

Definiamo una moneta elettronica come una catena di firme digitali. Ogni proprietario trasferisce la moneta al successivo firmando digitalmente un hash della transazione precedente e la chiave pubblica del proprietario successivo, e aggiungendo entrambe queste informazioni [7] alla fine della moneta. Il beneficiario del pagamento può verificare le firme per verificare la catena di proprietà.


TRANSAZIONE
TRANSAZIONE
TRANSAZIONE
PROP. 1
CHIAVE PUBBL.
PROP. 2
CHIAVE PUBBL.
PROP. 3
CHIAVE PUBBL.
HASH
HASH
HASH
VERIFICA
VERIFICA
PROP. 0
FIRMA
PROP. 1
FIRMA
PROP. 2
FIRMA
FIRMA
FIRMA
PROP. 1
CHIAVE PRIVATA
PROP. 2
CHIAVE PRIVATA
PROP. 3
CHIAVE PRIVATA

Naturalmente il problema è che il beneficiario del pagamento non può verificare che uno dei proprietari non abbia speso due volte la moneta. Una soluzione comune è introdurre un'autorità fiduciaria centrale, o "zecca", che controlli tutte le transazioni. Dopo ogni transazione, la moneta deve essere restituita alla zecca per l'emissione di una nuova moneta, e solo le monete emesse direttamente dalla zecca garantiscono contro il double spending. Il problema di questa

soluzione è che il destino dell'intero sistema monetario dipende dalla società che controlla la zecca: ogni transazione deve passare al suo vaglio, proprio come una banca.

Abbiamo bisogno di un metodo per cui il beneficiario del pagamento sappia che i proprietari precedenti non hanno firmato alcuna transazione antecedente. Per i nostri scopi, la prima transazione è l'unica che conti, perciò non ci interessano tentativi successivi di double spending. L'unico modo di

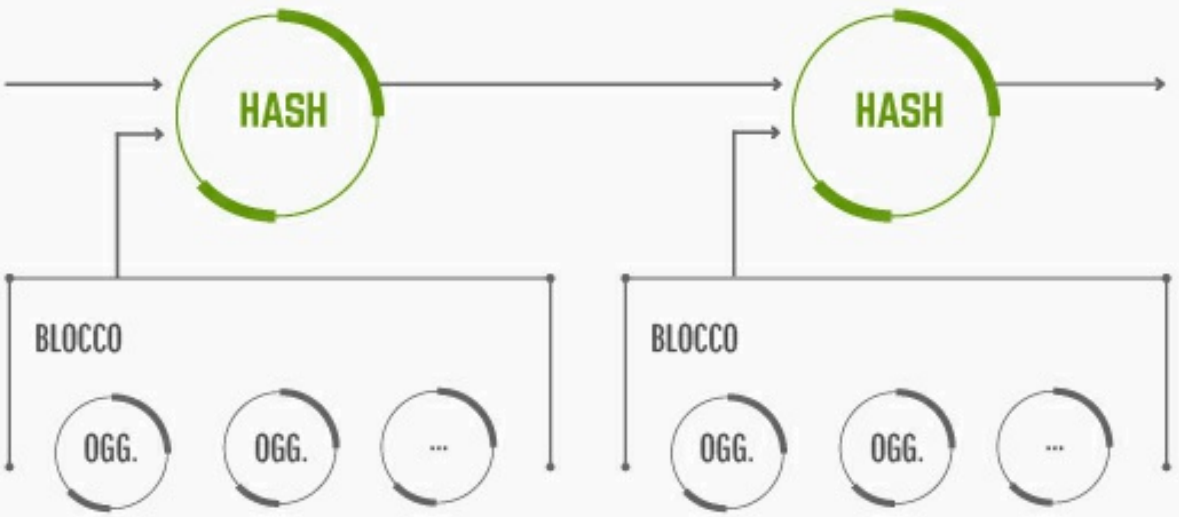
confermare l'assenza di una transazione è essere al corrente di tutte le transazioni. Nel modello "zecca", la zecca era al corrente di tutte le transazioni e stabiliva l'ordine di precedenza. Per ottenere lo stesso risultato in mancanza di un soggetto fiduciario, le transazioni devono essere dichiarate pubblicamente (Dai 1998), e abbiamo bisogno di un sistema per cui i partecipanti raggiungano il consenso su un'unica sequenza cronologica di recepimento delle transazioni. Il

beneficiario di un pagamento ha bisogno della prova che, per la maggioranza dei nodi, quella sia la sequenza cronologica valida.

3. Server di marcatura temporale

La soluzione proposta parte da un server di marcatura temporale. Un server di marcatura temporale applica una funzione hash a un blocco di oggetti da marcare temporalmente e rende pubblicamente noto l'hash, per esempio su un giornale o in un messaggio *Usenet* [8] (Massias et al. 1999; Haber e Stornetta 1991; Bayer et al. 1993; Haber e Stornetta 1997). La marcatura

temporale prova che in quel momento i dati devono essere esistiti, come ovvio, in modo da potere essere inclusi nell'hash. Ogni marcatura temporale include all'interno del suo hash la marcatura temporale precedente, formando una catena in cui ogni marcatura temporale aggiuntiva consolida quelle precedenti.


HASH
HASH
BLOCCO
OGG.
OGG.
...
BLOCCO
OGG.
OGG.
...

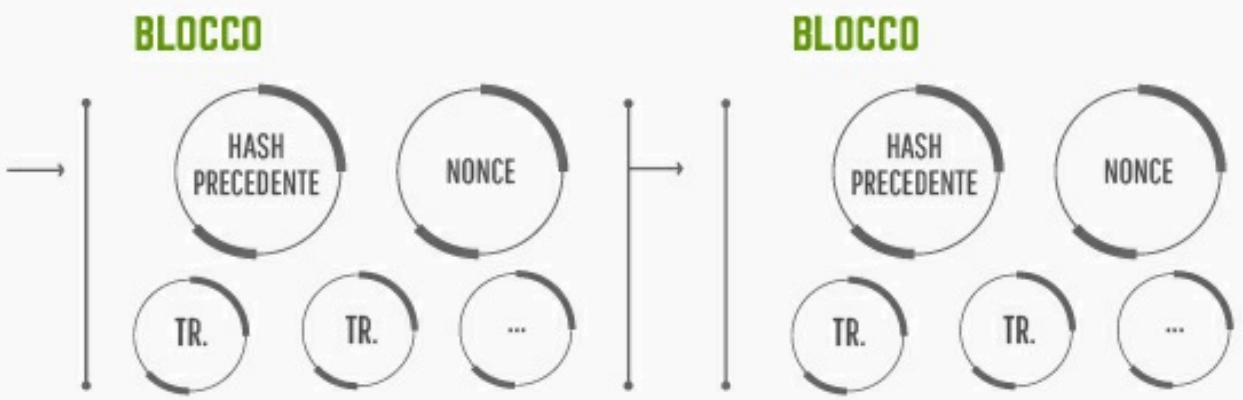
4. Proof-of-work

Per implementare un server di marcatura temporale distribuito in peer-to-peer, abbiamo bisogno di usare un sistema proof-of-work simile ad *Hashcash* [9] di Adam Back (Back 2002), piuttosto che un giornale o dei messaggi Usenet. La proof-of-work consiste nel trovare un valore che fornito come input a una funzione hash, per esempio *SHA-256* [10], generi un hash con un certo numero di *bit zero* [11]

iniziali. Il lavoro medio richiesto cresce esponenzialmente col numero di bit zero richiesti e può essere verificato eseguendo un singolo hash.

Per la nostra rete di marcatura temporale, la proof-of-work è implementata incrementando un *nonce* [12] nel blocco finché non viene trovato un valore tale che l'hash del blocco contenga i bit zero richiesti. Una volta che il lavoro computazionale è stato speso per soddisfare la proof-of-work, il blocco non può essere modificato

senza rieseguire il lavoro. Infatti, poiché il blocco seguente nella catena includerà un hash di questo blocco, una sua modifica implicherà il ricalcolo della proof-of-work per tutti i blocchi che seguono.


BLOCCO
HASH PRECEDENTE
NONCE
TR.
TR.
...
BLOCCO
HASH PRECEDENTE
NONCE
TR.
TR.
...

La proof-of-work risolve anche il problema della determinazione della rappresentanza nelle decisioni prese a maggioranza. Se la maggioranza fosse basata su "un *indirizzo IP* [13] un voto", potrebbe essere sovvertita da chiunque fosse in grado di allocare molti IP. La proof-of-work equivale essenzialmente a "una *CPU* [14] un voto". La maggioranza è rappresentata dalla catena più lunga, che incorpora il maggiore sforzo computazionale. Se la quota maggiore di potere

computazionale è controllata da nodi onesti, la catena onesta crescerà più rapidamente di qualsiasi altra catena concorrente. Per modificare un blocco precedente, un attaccante dovrebbe rieseguire la proof-of-work del blocco e di tutti i blocchi successivi a esso, per poi raggiungere e superare il lavoro dei nodi onesti. Più avanti verrà dimostrato che la probabilità che un attaccante più lento recuperi il ritardo diminuisce esponenzialmente man mano che vengono aggiunti blocchi successivi.

Per compensare la crescente velocità dell'hardware e la variabilità nel tempo della presenza di nodi attivi nella rete peer-to-peer, la difficoltà della proof-of-work varia anch'essa nel tempo ed è calcolata come una media mobile in grado di garantire la produzione di un numero di blocchi predeterminato ogni ora. Se i blocchi vengono generati più velocemente, la difficoltà aumenta.

5. Rete

Le azioni che la rete deve compiere sono le seguenti:

1. Le nuove transazioni sono trasmesse a tutti i nodi.
2. Ogni nodo raggruppa le nuove transazioni in un blocco.
3. Ogni nodo lavora per produrre una proof-of-work per il blocco che ha assemblato.
4. Quando un nodo trova una proof-

of-work, esso trasmette il blocco a tutti i nodi.

5. I nodi accettano il blocco solo se tutte le transazioni contenute in esso sono valide e non sono state già spese.
6. I nodi esprimono l'accettazione del blocco lavorando alla creazione del blocco successivo nella catena, usando l'hash del blocco accettato come hash precedente.

I nodi considerano sempre la catena

più lunga come quella corretta e continueranno a lavorare alla sua estensione. Se due nodi trasmettono simultaneamente versioni differenti del blocco successivo, alcuni nodi potrebbero ricevere per prima o l'una o l'altra. In questo caso, essi lavorano alla prima che hanno ricevuto, ma conservano l'altro ramo nell'eventualità che diventi più lungo. Il legame è spezzato quando viene trovata la proof-of-work successiva e un ramo diventa più lungo; i nodi che stavano lavorando

all’altro ramo passeranno quindi al ramo più lungo.

La diffusione sulla rete delle nuove transazioni non deve necessariamente raggiungere tutti i nodi. Fintantoché una nuova transazione raggiunge un numero elevato di nodi, è garantito che presto confluirà in un blocco. La trasmissione dei blocchi tollera anche la perdita di messaggi. Se un nodo non riceve un blocco, esso lo richiederà quando avrà ricevuto il blocco successivo e si sarà reso conto che gliene manca uno.

6. Incentivo

Per convenzione, la prima transazione in un blocco è una transazione speciale che genera nuova moneta di proprietà del creatore del blocco. Ciò aggiunge un incentivo per i nodi a supportare la rete, e fornisce un modo per mettere nuove monete in circolazione, dal momento che non esiste nessuna autorità centrale che le emetta. L'aggiunta continua di una quantità costante di nuove monete è paragonabile all'opera dei minatori

d’oro, che spendono risorse per mettere nuovo oro in circolazione. Nel nostro caso, a essere spesi sono *tempo di CPU* [15] ed elettricità.

Un ulteriore incentivo può essere rappresentato da commissioni di transazione. Se il valore in uscita di una transazione è inferiore al suo valore in entrata, la differenza consiste in una commissione di transazione che è aggiunta al valore di incentivo del blocco contenente la transazione. Una volta che un numero predeterminato di

monete è entrato in circolazione, l'incentivo può essere costituito interamente da commissioni di transazione e risultare in un sistema monetario completamente privo di inflazione.

L'incentivo potrebbe aiutare i nodi a restare onesti. Se un attaccante avido fosse in grado di accumulare più potere computazionale di tutti i nodi onesti, dovrebbe scegliere se usarlo per defraudare gli altri rubando i suoi stessi pagamenti oppure usarlo per generare

nuove monete. Egli dovrebbe trovare più redditizio stare alle regole del gioco, regole tali da beneficiarlo con più monete di qualsiasi altro insieme di nodi, piuttosto che sabotare il sistema e il valore della sua stessa ricchezza.

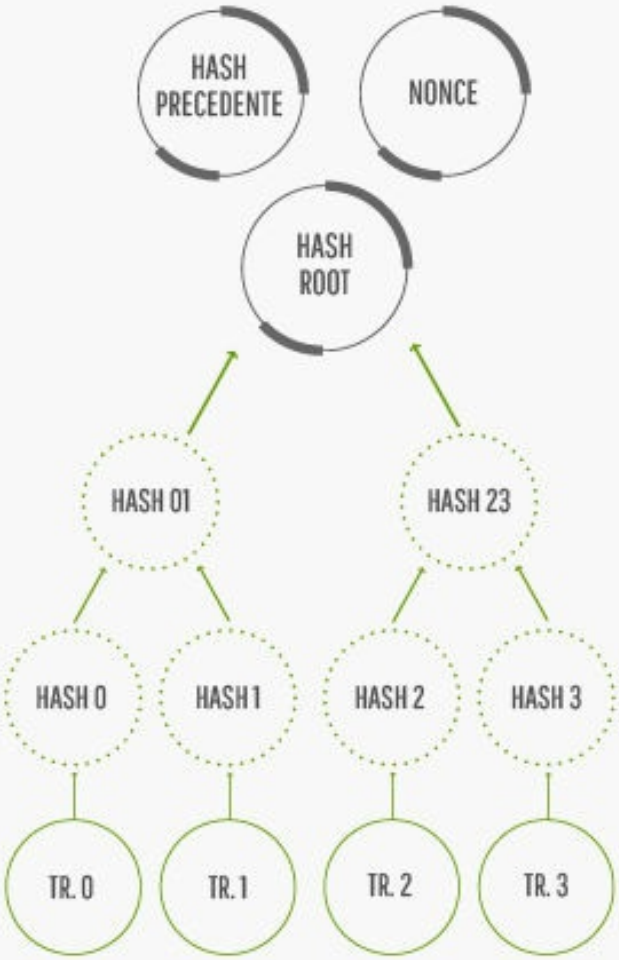
7. Liberare spazio su disco

Una volta che la transazione più recente di una moneta è sepolta sotto un numero sufficiente di blocchi, le transazioni di spesa che la precedono possono essere scartate per risparmiare spazio su disco. Per facilitare l'operazione senza compromettere l'hash del blocco, le transazioni precedenti vengono rappresentate in maniera sintetica tramite un *Merkle tree*

[16] (Merkle 1980; Massias *et al.* 1999; Haber e Stornetta 1997), includendo solo la root del Merkle tree nell'hash del blocco. I vecchi blocchi possono poi essere compattati potando i rami dell'albero. Non c'è bisogno di memorizzare gli hash interni.

BLOCCO

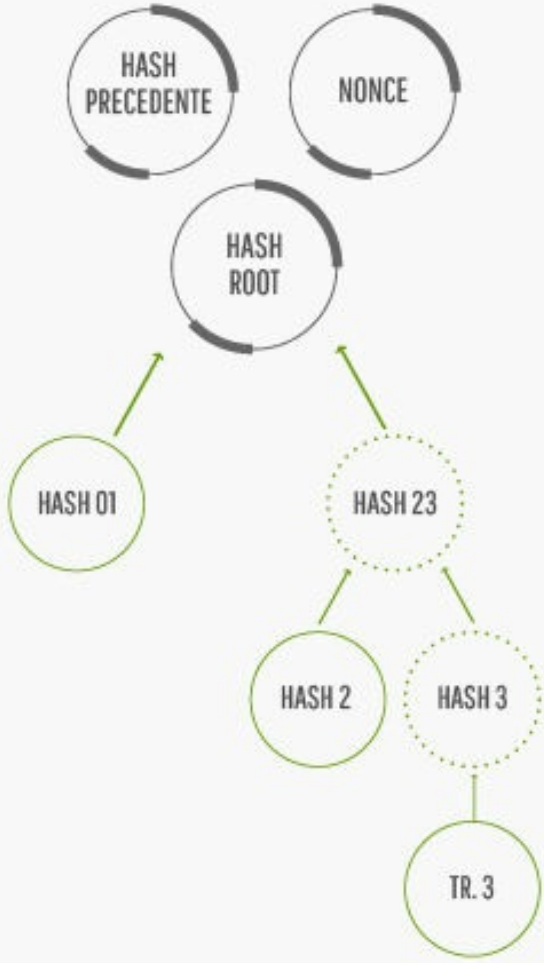
INTESTAZIONE BLOCCO (HASH)



TRANSAZIONI INSERITE
NEL MERKLE TREE

BLOCCO

INTESTAZIONE BLOCCO (HASH)



RIMOZIONE TR. 0-2
DAL BLOCCO

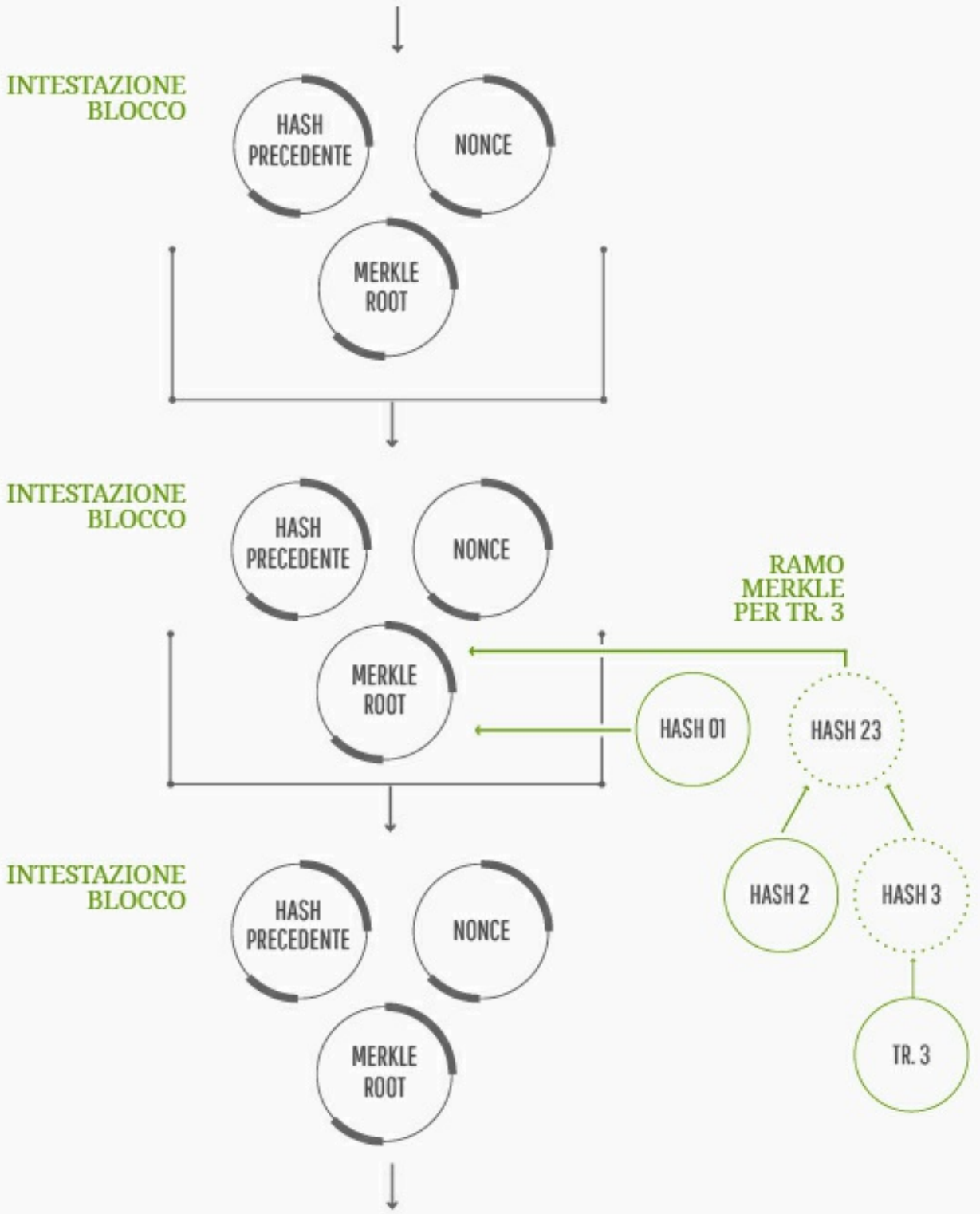
L’intestazione di un blocco privo di transazioni peserebbe 80 byte circa. Supponendo che i blocchi siano generati ogni 10 minuti, allora 80 byte * 6 * 24 * 365 = 4.2MB/anno. Coi 2GB di RAM normalmente inclusi nei computer in vendita nel 2008, e con la *legge di Moore* [17] che prevede una crescita di 1.2GB/anno, lo spazio non dovrebbe rappresentare un problema nemmeno se le intestazioni di blocco dovessero essere conservate in memoria.

8. Verifica dei pagamenti semplificata

È possibile verificare i pagamenti senza disporre di tutte le funzionalità di un nodo completo della rete Bitcoin. Un utente ha infatti solo bisogno di disporre delle intestazioni di blocco della catena di proof-of-work più lunga, che può ottenere interrogando gli altri nodi finché non è sicuro di possedere la catena più lunga, e ottenere il ramo del Merkle tree che collega la transazione al

blocco in cui la transazione da verificare è stata inserita e marcata temporalmente. L'utente non può controllare la transazione da solo, ma, collegandola a un punto della catena, può vedere che un nodo della rete l'ha accettata, e i blocchi aggiunti dopo il blocco che la contiene confermano ulteriormente che la rete l'ha accettata.

CATENA PROOF-OF-WORK PIÙ LUNGA



In questo modo, la verifica è affidabile fintantoché i nodi onesti controllano la rete, ma è più vulnerabile se la rete è dominata da un attaccante. Mentre i nodi della rete possono verificare le transazioni da soli, il metodo semplificato può essere ingannato da transazioni artefatte fintantoché l'attaccante riesce a continuare a dominare la rete. Una strategia di protezione da questo attacco potrebbe consistere nell'accettare avvisi dai nodi della rete quando questi

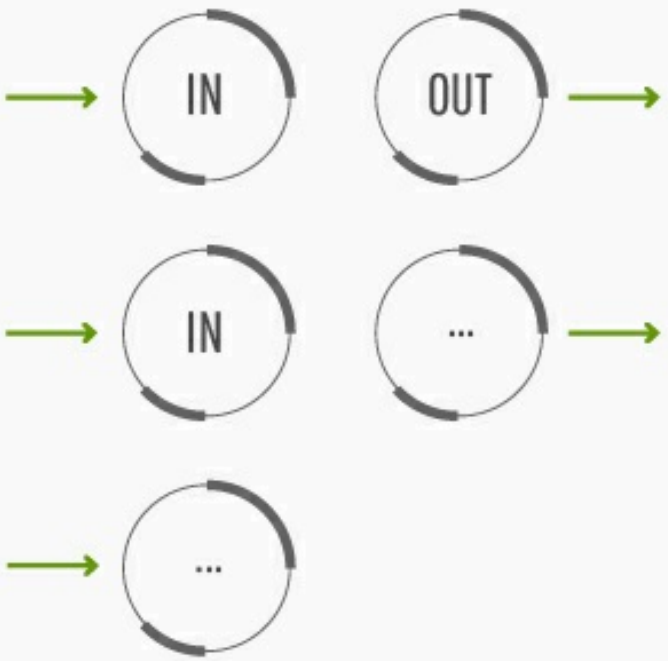
individuino un blocco non valido, inducendo il software dell'utente a scaricare l'intero blocco e le transazioni segnalate per confermare l'incongruenza. Probabilmente le attività commerciali che ricevono pagamenti frequenti vorranno comunque disporre di loro nodi al fine di verificare le transazioni con maggiore sicurezza e velocità.

9. Combinare e dividere il valore

Sebbene sia possibile gestire le monete singolarmente, sarebbe scomodo effettuare una transazione distinta per ogni centesimo parte del trasferimento. Per permettere che il valore sia diviso e combinato, le transazioni contengono input e output multipli. Normalmente ci sarà o un singolo input da una transazione precedente più grande o input multipli che combinano importi

minori, e al più due output: uno per il pagamento e uno che ritorna il resto, se presente, al datore.

TRANSAZIONE



Si noti che la ramificazione per cui una transazione dipende da varie transazioni e queste transazioni dipendono a loro volta da molte altre, non è un problema. Infatti, per quanto esposto sopra, non è mai necessario estrarre una copia completa della cronologia di una transazione.

10. Riservatezza

Il modello bancario tradizionale raggiunge un certo livello di riservatezza limitando l'accesso alle informazioni alle parti coinvolte e al terzo fiduciario. La necessità di annunciare pubblicamente tutte le transazioni preclude questo metodo, ma la riservatezza può essere ancora mantenuta spezzando il flusso delle informazioni in un altro punto: rendendo anonime le chiavi pubbliche. Chiunque

potrà vedere che qualcuno sta inviando una somma a qualcun altro, ma senza informazioni che colleghino la transazione a nessuno. Ciò è simile al livello di informazioni comunicate dai mercati azionari, dove l'ora e la dimensione degli scambi individuali, il *tape* [18], sono rese pubbliche, ma senza dichiarare chi siano le parti.

MODELLO DI RISERVATEZZA TRADIZIONALE

NUOVO MODELLO DI RISERVATEZZA

Al fine di schermare ulteriormente le identità, per ogni nuova transazione dovrebbe essere usato un nuovo paio di chiavi per evitare che queste possano essere collegate a un unico soggetto. Un certo grado di collegamento è comunque inevitabile con transazioni a input multiplo, che rivelano necessariamente che i loro input erano posseduti dallo stesso proprietario. Il rischio è che, se il proprietario di una chiave venisse rivelato, questo potrebbe fare emergere altre transazioni appartenenti allo stesso

proprietario.

11. Calcoli

Consideriamo lo scenario in cui un attaccante tenta di generare una catena alternativa più rapidamente dei nodi onesti. Anche se ciò gli riuscisse, non esporrebbe il sistema a cambiamenti arbitrari, come la creazione di valore dal nulla o il furto di moneta mai appartenuta all'attaccante. I nodi non accetteranno in pagamento una transazione non valida, e i nodi onesti non accetteranno mai un blocco che ne

contenga. Un attaccante può solo modificare una delle sue stesse transazioni per riprendersi le monete appena spese.

La gara tra la catena onesta e la catena di un attaccante può essere rappresentata da una *passeggiata aleatoria binomiale* [19]. L'evento successo è che la catena onesta sia estesa di un blocco, aumentando il suo vantaggio di +1, e l'evento insuccesso è che la catena dell'attaccante sia estesa di un blocco, riducendo il distacco di -1.

La probabilità che un attaccante recuperi un certo svantaggio è analoga al problema della *rovina del giocatore* [20]. Si supponga che un giocatore con credito illimitato parta con uno svantaggio e possa giocare un numero infinito di volte per cercare di raggiungere il pareggio. Possiamo calcolare la probabilità che egli raggiunga il pareggio, ovvero che un attaccante recuperi lo svantaggio dalla catena onesta, come segue (Feller 1957):

p = probabilità che un nodo onesto scopra il blocco successivo

q = probabilità che l'attaccante scopra il blocco successivo

q_z = probabilità che l'attaccante recuperi z blocchi di svantaggio

$$q_z = \begin{cases} 1 & \text{se} \quad p \leq q \\ (q/p)^z & \text{se} \quad p > q \end{cases}$$

Data la nostra ipotesi $p>q$, la probabilità diminuisce esponenzialmente all'aumentare del numero di blocchi che

un attaccante deve recuperare. Con le probabilità contro di lui, se non effettua un balzo fortunato in avanti all'inizio, le sue possibilità divengono piccole al limite dell'evanescente man mano che scivola sempre più indietro.

Consideriamo ora quanto a lungo debba aspettare il beneficiario di una nuova transazione prima di essere sufficientemente sicuro che il datore non possa modificare la transazione. Ipotizziamo che il datore sia un attaccante che voglia fare credere per un

po’ al beneficiario di avere effettuato il pagamento, e poi invertirlo per ripagare sé stesso dopo che è passato un po’ di tempo. Il beneficiario sarà avvertito quando ciò avviene, ma il datore spera che sia troppo tardi.

Il beneficiario genera un nuovo paio di chiavi e fornisce la chiave pubblica al datore poco prima della firma. Ciò impedisce al datore di preparare una catena di blocchi prima del tempo lavorandovi continuamente finché è abbastanza fortunato da portarsi

sufficientemente avanti, per poi eseguire la transazione sul momento. Una volta che la transazione è inviata, il datore disonesto comincia a lavorare in segreto a una catena parallela che contiene una versione alternativa della sua transazione.

Il beneficiario attende finché la transazione è stata aggiunta a un blocco e z blocchi sono stati concatenati dopo di esso. Egli non conosce l'esatto ammontare del progresso fatto dall'attaccante, ma, ipotizzando che i

nodi onesti abbiano impiegato il tempo medio per blocco, il progresso potenziale dell'attaccante seguirà una *distribuzione di Poisson* [21] con valore atteso:

$$\lambda = z\frac{q}{p}$$

Per ottenere la probabilità che l'attaccante possa ancora recuperare, moltiplichiamo la *densità* [22] della Poisson corrispondente a ogni ammontare di progresso che egli potrebbe avere fatto per la probabilità che egli possa recuperare da quel punto:

$$\sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \begin{cases} (q/p)^{(z-k)} & \text{se } k \le z \\ 1 & \text{se } k > z \end{cases}$$

Riordinando per evitare di sommare la coda infinita della distribuzione…

$$1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - (q/p)^{(z-k)}\right)$$

Passando al codice *C* [23]…

```
#include <math.h>
double AttackerSuccessProbability
{
  double p = 1.0 - q;
  double lambda = z * (q / p);
  double sum = 1.0;
  int i, k;
```

```
  for (k = 0; k <= z; k++)
  {
    double poisson = exp(-lambda)
    for (i = 1; i <= k; i++)
      poisson *= lambda / i;
    sum -= poisson * (1 - pow(q ,
  }
  return sum;
}
```

Eseguendo alcune prove, possiamo vedere che la probabilità cala esponenzialmente con z.

q=0.1	

z=0	P=1.0000000
z=1	P=0.2045873
z=2	P=0.0509779
z=3	P=0.0131722
z=4	P=0.0034552
z=5	P=0.0009137
z=6	P=0.0002428
z=7	P=0.0000647
z=8	P=0.0000173
z=9	P=0.0000046
z=10	P=0.0000012

q=0.3	
z=0	P=1.0000000
z=5	P=0.1773523
z=10	P=0.0416605
z=15	P=0.0101008
z=20	P=0.0024804
z=25	P=0.0006132
z=30	P=0.0001522
z=35	P=0.0000379

z=40	P=0.0000095
z=45	P=0.0000024
z=50	P=0.0000006

Risolvendo per P minore di 0.1% [24]…

P<0.001	
q=0.10	z=5
q=0.15	z=8
q=0.20	z=11
q=0.25	z=15

q=0.30	z=24
q=0.35	z=41
q=0.40	z=89
q=0.45	z=340

12. Conclusione

Abbiamo proposto un sistema per le transazioni elettroniche che non si basa sulla fiducia. Siamo partiti dal classico modello di monete create a partire da firme digitali, che fornisce un forte controllo sulla proprietà, ma è incompleto senza una modalità di prevenzione del double spending. Per risolvere questo problema, abbiamo proposto una rete peer-to-peer che usa proof-of-work per registrare una

cronologia pubblica delle transazioni che rapidamente diventa computazionalmente difficile da modificare per un attaccante se i nodi onesti controllano la quota maggiore di potere computazionale. La rete è robusta nella sua semplicità non strutturata.

I nodi lavorano contemporaneamente senza coordinamento. Non hanno bisogno di essere identificati, dato che i messaggi non sono trasmessi a un punto particolare e hanno solo bisogno di essere consegnati su base best effort. I

nodi possono lasciare la rete e riunirvisi in seguito a piacimento, accettando la catena di proof-of-work più lunga come prova di ciò che è avvenuto in loro assenza. Essi votano col proprio potere computazionale: esprimono la loro accettazione dei blocchi validi lavorandone all'estensione e scartano i blocchi non validi rifiutando di lavorarvi. Tutte le regole e gli incentivi necessari possono essere applicati tramite questo meccanismo di consenso.

Riferimenti

- W. Dai, “b-money”, http://bit.ly/1rpNMuD, 1998.
- H. Massias, X. S. Avila e J.-J. Quisquater, “Design of a secure timestamping service with minimal trust requirements”, in *20th Symposium on Information Theory in the Benelux*, Maggio 1999.
- S. Haber e W. S. Stornetta, “How to time-stamp a digital document”, in *Journal of*

Cryptology, vol. 3, n. 2, pagg. 99-111, 1991.

- D. Bayer, S. Haber e W. S. Stornetta, "Improving the efficiency and reliability of digital time-stamping", in *Sequences II: Methods in Communication, Security and Computer Science*, pagg. 329-334, 1993.
- S. Haber e W. S. Stornetta, "Secure names for bit-strings", in *Proceedings of the 4th ACM Conference on Computer and*

Communications Security, pagg. 28-35, Aprile 1997.

- A. Back, “Hashcash - a denial of service counter-measure”, http://bit.ly/1z3bUND, 2002.
- R. C. Merkle, “Protocols for public key cryptosystems”, in *Proc. 1980 Symposium on Security and Privacy*, IEEE Computer Society, pagg. 122-133, Aprile 1980.
- W. Feller, “An introduction to probability theory and its applications”, 1957.

APPENDICE

AttackerSuccessProbabilit in JavaScript

```
function AttackerSuccessProbab
var p = 1.0 - q;
var lambda = z * (q / p);
var sum = 1.0;
for (var k = 0; k <= z; k++) {
  var poisson = Math.exp(-lambd
  for (var i = 1; i <= k; i++)
    var poisson = poisson * lam
  }
  sum -= poisson * (1 - Math.po
}
return sum;
}
```

[1] In una rete peer-to-peer i nodi sono paritari, cioè inviano richieste agli altri nodi (fungendo da client) e ricevono richieste da essi (fungendo da server), senza una gerarchia client/server prestabilita. [NdC]

[2] Le firme digitali sono metodi di certificazione digitale che garantiscono la provenienza, l'integrità e la consistenza dei dati. [NdC]

[3] Nei sistemi di valuta elettronica, il double spending è una truffa che consiste nello spendere la stessa moneta già spesa in una certa transazione anche in una o più altre transazioni. [NdC]

[4] Un algoritmo hash mappa dati di lunghezza arbitraria in dati di lunghezza fissa (più breve del dato originario, nelle principali applicazioni pratiche), il valore restituito è a sua volta chiamato hash. [NdC]

[5] Un sistema proof-of-work rappresenta una misura economica di sicurezza che scoraggia gli abusi di servizio esigendo un certo ammontare di lavoro da parte dei richiedenti. In ambito informatico, si tratta solitamente di lavoro computazionale. [NdC]

[6] La rete “fa del suo meglio”, ma non garantisce che la comunicazione avvenga entro un tempo predeterminato, con un livello di qualità prestabilito o in base a un grado di priorità preassegnato. [NdC]

[7] Ossia, l’hash della transazione precedente e la chiave pubblica del proprietario successivo. [NdC]

[8] Usenet è un’architettura di rete distribuita per la discussione tra utenti Internet che permette di leggere e inviare messaggi in sezioni tematiche a struttura gerarchica (*newsgroup*). I newsgroup Usenet sono stati i precursori dei forum Internet, che però adottano un’architettura client/server centralizzata. [NdC]

[9] Hashcash è un metodo antispam basato su proof-of-work: il mittente deve includere nel messaggio una marcatura che dimostri al destinatario di avere svolto un certo lavoro computazionale prima dell’invio. [NdC]

[10] SHA-256 è un algoritmo hash che restituisce una sequenza di 256 bit. Vedi [4]. [NdC]

[11] I bit zero sono semplicemente bit (valori binari)

valorizzati a zero. [NdC]

[12] Nonce deriva dall'inglese "number used once" ("numero usato una sola volta") e viene impiegato nel sistema proof-of-work al fine di variare i dati forniti come input alla funzione hash durante la ricerca della soluzione al problema crittografico. [NdC]

[13] Un indirizzo IP identifica univocamente un nodo all'interno di una rete informatica basata sul protocollo di comunicazione utilizzato dalla rete Internet (*Internet Protocol*, da cui l'acronimo IP). [NdC]

[14] La CPU è l'unità di elaborazione centrale ("central processing unit") di un computer, qui intesa nella sua capacità di eseguire un certo numero di istruzioni logiche e aritmetiche in una data unità di tempo. [NdC]

[15] Il tempo di CPU è il tempo impiegato dal processore per eseguire un determinato programma. [NdC]

[16] Un Merkle tree è un albero di hash in cui gli

elementi terminali (“foglie”) sono hash dei dati e gli elementi intermedi (“rami”) sono hash dei loro elementi figli (foglie o altri rami). In cima alla gerarchia c’è l’elemento “radice”, l’hash root. [NdC]

[17] La legge di Moore, proposta nel 1965 da Gordon Moore, cofondatore di *Intel*, è la prima e la più celebre di una serie di osservazioni empiriche che prevedono una crescita esponenziale negli sviluppi della microelettronica. [NdC]

[18] Il tape (“nastro”) trasmetteva informazioni sui prezzi azionari attraverso le linee telegrafiche. In uso sin dalla seconda metà del XIX secolo, ha rappresentato il primo mezzo di comunicazione digitale, per essere poi rimpiazzato dalla televisione e dal computer a partire dalla seconda metà del XX secolo. [NdC]

[19] Una passeggiata aleatoria è la rappresentazione formale in matematica di un percorso composto da una sequenza di passi casuali. Una passeggiata aleatoria binomiale è una passeggiata aleatoria semplice composta da una sequenza di passi di lunghezza

identica e indipendenti tra loro in una delle due sole direzioni possibili (+1/-1, destra/sinistra e così via — in questo caso, successo/insuccesso). [NdC]

[20] Il problema della rovina del giocatore consiste nel calcolo della probabilità che un giocatore vinca una serie di scommesse fino all'esaurimento della dotazione iniziale di fondi dell'avversario, date le dotazioni iniziali di entrambi e una probabilità costante di vincita/perdita. [NdC]

[21] La distribuzione di Poisson è una distribuzione di probabilità utilizzata, in genere, per modellare il conteggio di incidenti, ovvero eventi osservati con bassa frequenza in campioni numerosi. [NdC]

[22] La funzione di densità di probabilità di una variabile casuale descrive la probabilità che la variabile casuale assuma, approssimativamente, un certo valore. [NdC]

[23] In informatica, C è un famoso linguaggio di programmazione creato agli inizi degli anni '70 da Dennis Ritchie presso i Bell Laboratories, centro di

ricerca della compagnia telefonica statunitense AT&T. In appendice è riportata l'implementazione dello stesso algoritmo in *JavaScript*, linguaggio di programmazione disponibile su tutti i principali browser web. [NdC]

[24] Ossia, una probabilità piccola a piacere. [NdC]