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

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Translated in Italian from bitcoin.org/bitcoin.pdf
by @terzim

Sommario. Una versione puramente peer-to-peer di denaro elettronico permetterebbe di spedire direttamente pagamenti online da un'entità ad un'altra senza passare tramite un'istituzione finanziaria. Le firme digitali offrono una soluzione parziale al problema, ma i benefici principali sono persi se una terza persona di fiducia è ancora richiesta per prevenire la doppia spesa. Proponiamo una soluzione al problema della doppia spesa mediante l'utilizzo di una rete peer-to-peer. La rete stampa un marcatore temporale sulle transazioni facendo hashing sulle stesse e incatenandole in una catena di proof-of-work basata sugli hash, formando una registrazione che non può essere modificata senza rifare la proof-of-work. La catena più lunga non solo serve come prova della sequenza di eventi ai quali si è assistito, ma anche come prova che essa proviene dal gruppo più grande di potenza CPU. Fintanto che la maggior parte della potenza CPU è controllata da nodi che non cooperano per attaccare la rete, questi genereranno la catena più lunga e supereranno gli utenti malintenzionati. La rete stessa richiede una struttura minimale. I messaggi sono trasmessi su base best effort, e i nodi possono lasciare e ricongiungersi con la rete a loro piacimento, accettando la catena proof-of-work più lunga come prova di quello che è avvenuto mentre erano non erano presenti.

1. Introduzione

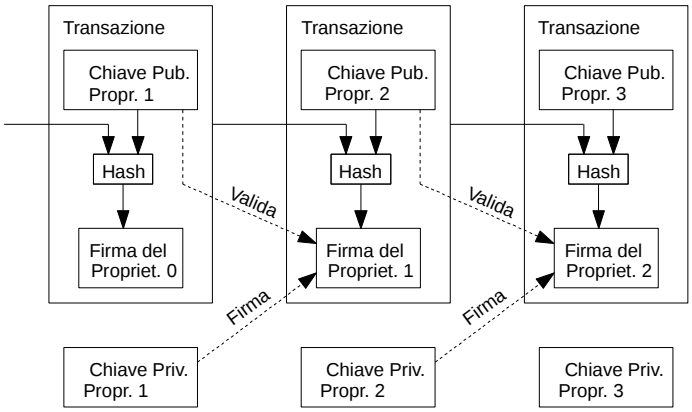
Il commercio su Internet fa affidamento quasi esclusivamente sulle istituzioni finanziarie che servono come terze parti di fiducia per elaborare i pagamenti elettronici. Nonostante il sistema funzioni abbastanza bene per la maggior parte delle transazioni, esso soffre ancora delle debolezze intrinseche di un modello basato sulla fiducia. Transazioni totalmente irreversibili non sono realmente possibili, dal momento che le istituzioni finanziarie non possono evitare le dispute di mediazione. Il costo dell'intermediazione aumenta i costi di transazione, limitando la dimensione minima delle transazioni praticabili ed escludendo la possibilità di piccole transazioni occasionali, e c'è un costo più ampio collegato alla perdita della capacità di effettuare pagamenti irreversibili per quei servizi che sono anch'essi irreversibili. Con la possibilità di reversibilità, si diffonde la necessità di fiducia. I commercianti devono diffidare dei loro clienti, tormentandoli con maggiori richieste di informazioni rispetto a quanto non sarebbe altrimenti necessario. Una certa percentuale di frodi è accettata come inevitabile. Tali costi e le incertezze di pagamento possono essere evitati utilizzando moneta fisica di persona, ma non esiste alcun meccanismo per effettuare pagamenti attraverso un mezzo di comunicazione senza un'entità di fiducia.

È dunque necessario un sistema di pagamento elettronico basato su prova crittografica invece che sulla fiducia, che consenta a due controparti qualsiasi negoziare direttamente tra loro senza la necessità di una terza parte di fiducia. Le transazioni che sono computazionalmente impraticabili da invertire proteggerebbero i venditori dalle frodi, e meccanismi consuetudinari di deposito di garanzia potrebbero essere facilmente implementati per proteggere gli acquirenti. In questo

lavoro, proponiamo una soluzione al problema della doppia spesa utilizzando un server di marcatura temporale distribuito peer-to-peer per generare la prova computazionale dell'ordine cronologico delle transazioni. Il sistema è sicuro fintanto che i nodi onesti controllano collettivamente più potenza CPU rispetto a qualsiasi gruppo collaborativo di nodi attaccanti.

2. Le Transazioni

Definiamo come valuta elettronica una catena di firme digitali. Ciascun proprietario trasferisce valuta al successivo firmando digitalmente un hash della transazione precedente e la chiave pubblica del proprietario successivo e aggiungendo le stesse alla fine della valuta. Colui che riceve un pagamento può verificare le firme digitali per validare la catena di proprietà.



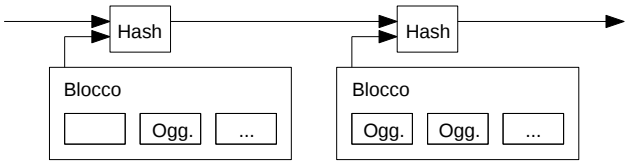
Il problema, naturalmente, è che il beneficiario non può verificare che ciascuno dei proprietari precedenti non abbia speso due volte lo stesso ammontare di valuta. Una soluzione comune è quella di introdurre un'autorità fiduciaria centrale, o zecca, che controlli tutte le transazioni. Dopo ogni transazione, la moneta deve essere restituita alla zecca, la quale emette una nuova moneta, e si crede che solo le monete emesse direttamente dalla zecca non siano state spese due volte. Il problema di questa soluzione è che il destino di tutto il sistema monetario dipende dalla società che gestisce la zecca, e ogni transazione deve passare attraverso di essa, proprio come una banca.

Abbiamo bisogno di un modo per far sì che il beneficiario sappia che i precedenti proprietari non abbiano firmato alcuna transazione precedente a quella che lo riguarda. Per i nostri scopi, la prima operazione è quella che conta, e non ci importa nulla dei tentativi successivi di doppia spesa. L'unico modo per confermare l'assenza di una transazione è di essere a conoscenza di tutte le transazioni. Nel modello basato sulla zecca, questa era a conoscenza di tutte le transazioni e decideva quale era avvenuta per prima. Per fare lo stesso ma senza un'autorità di fiducia, le transazioni devono essere annunciate pubblicamente [1], e abbiamo bisogno di un sistema attraverso il quale i partecipanti concordino su un singolo passato dell'ordine in cui esse sono state ricevute. Il beneficiario ha bisogno di una prova che, al momento di ogni transazione, la maggior parte dei nodi è d'accordo che essa è la prima ricevuta.

3. Server di marcatura temporale

La soluzione proposta parte da un server di marcatura temporale. Un server di marcatura temporale agisce facendo hash di un blocco di oggetti in modo che siano marcati temporalmente e poi pubblica l'hash, ad esempio su un quotidiano o in un post su Usenet. La marcatura temporale prova ovviamente che i dati devono essere esistiti in quella determinata data, visto che sono finiti nell'hash. Ogni marcatura temporale comprende quella precedente nel suo hash,

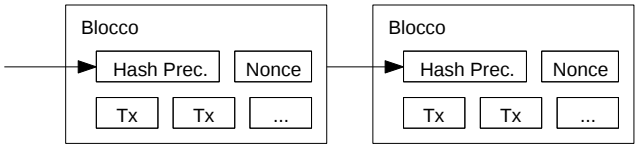
formando una vera e propria catena, e ogni marcatura temporale ovviamente rafforza quelle precedenti.



4. Proof-of-Work

Per implementare un server di marcatura temporale distribuito su base peer-to-peer, avremo bisogno di usare un sistema simile a quello di Hashcash di Adam Back [6], piuttosto che basarci sui messaggi di quotidiani o Usenet. La proof-of-work comporta la ricerca di un valore che, una volta sottoposto ad hash (ad esempio con SHA-256), restituisca un hash che inizia con un numero di zero bit. Il lavoro medio richiesto è esponenzialmente proporzionale al numero di zero bit richiesti e può essere verificato eseguendo un unico hash.

Per la nostra rete di marcatura temporale, implementiamo la proof-of-work incrementando un nonce nel blocco fino a quando è trovato un valore che dà all'hash del blocco gli zero bits necessari. Una volta che l'impegno della CPU è stato speso per soddisfare la proof-of-work, il blocco non può essere modificato senza rifare il lavoro. Poiché i blocchi successivi sono incatenati dopo di esso, il lavoro necessario per cambiare il blocco dovrebbe includere il rifacimento di tutti i blocchi successivi.



La proof-of-work risolve anche il problema della determinazione della rappresentatività in un sistema di decisioni prese a maggioranza. Se la maggioranza fosse basata sul principio “un indirizzo IP-un-voto”, potrebbe essere sovvertita da chiunque fosse in grado di allocare molti IPs. La proof-of-work invece segue essenzialmente il principio “una CPU-un voto”. La decisione di maggioranza è rappresentata dalla catena più lunga, su cui è stato speso il massimo sforzo di proof-of-work. Se la maggioranza di potenza della CPU è controllata da nodi onesti, la catena onesta crescerà più velocemente e supererà eventuali catene concorrenti. Per modificare un blocco passato, un utente malintenzionato dovrebbe rifare la proof-of-work del blocco e di tutti i blocchi successivi ad esso e poi raggiungere e superare il lavoro dei nodi onesti. Mostreremo in seguito che la probabilità che un utente malintenzionato raggiunga il lavoro dei nodi onesti diminuisce in modo esponenziale a mano a mano che vengono aggiunti blocchi successivi.

Per compensare l'aumento della velocità dell'hardware e il variare dell'interesse dei nodi operativi col tempo, la difficoltà della proof-of-work è determinata da una media mobile che ha come obiettivo la creazione di un numero medio di blocchi ogni ora. Se i blocchi vengono generati troppo velocemente, la difficoltà aumenta.

5. La Rete

I passi per far girare la rete sono i seguenti:

1) Le nuove transazioni sono trasmesse a tutti i nodi.
2) Ogni nodo immagazzina le nuove transazioni in un blocco.
3) Ogni nodo lavora per trovare una proof-of-work difficile per il suo blocco.
4) Quando un nodo trova un proof-of-work, trasmette il blocco a tutti gli altri nodi.
5) I nodi accettano il blocco solo se tutte le transazioni in esso sono valide e non sono già state spese.
6) I nodi esprimono l'accettazione del blocco mediante il tentativo di creare il prossimo blocco nella catena, utilizzando l'hash del blocco accettato come hash precedente.

I nodi considerano sempre come corretta la catena più lunga e continueranno a lavorare per allungarla. Se due nodi trasmettono diverse versioni del blocco successivo contemporaneamente, alcuni nodi possono ricevere l'una o l'altra prima. In tal caso, questi lavorano sul primo che hanno ricevuto, ma salvano l'altra ramificazione nel caso diventi più lunga. Questo impasse sarà risolto quando la proof-of-work successiva viene trovata e una delle ramificazioni diventa più lunga; i nodi che stavano lavorando sull'altra ramificazione a quel punto si sposteranno su quella più lunga.

Le trasmissioni delle nuove transazioni non devono necessariamente raggiungere tutti i nodi. Fintanto che raggiungono numerosi nodi, non ci vorrà molto affinchè vengano inserite in un blocco. Le trasmissioni dei blocchi tollerano anche i messaggi troncati. Se un nodo non riceve un blocco, lo richiederà quando riceverà il blocco successivo e capirà di averne saltato uno.

6. L'Incentivo

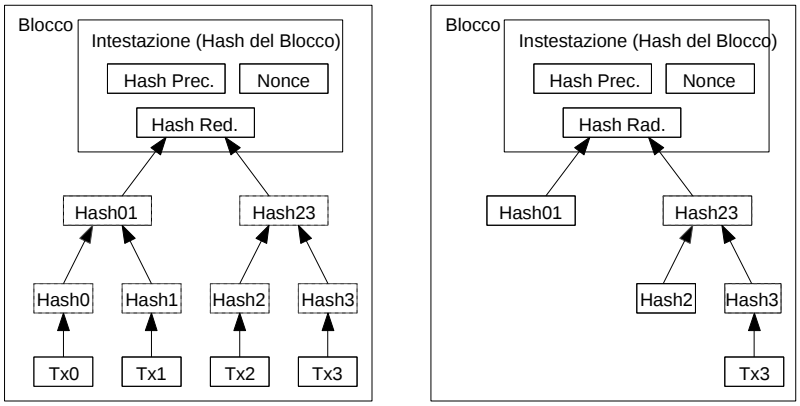
Per convenzione, la prima transazione in un blocco è una transazione speciale che “conia” una nuova moneta di proprietà del creatore del blocco. Questo fornisce un incentivo ai nodi affinché sostengano la rete, e fornisce un modo per la distribuzione iniziale di monete in circolazione, dato che non vi è alcuna autorità centrale che possa emetterle. L'aggiunta costante di una data quantità di nuove monete è analoga al processo dei minatori d'oro, che spendono risorse per incrementare la quantità di oro in circolazione. Nel nostro caso, viene spesa potenza CPU e viene consumata energia elettrica.

L'incentivo può essere anche finanziato con costi di transazione. Se il valore di uscita di una transazione è inferiore al suo valore di ingresso, la differenza è una tassa di transazione che viene aggiunta al valore di incentivazione del blocco contenente la transazione. Nel momento in cui sia entrato in circolazione un ammontare predeterminato di monete, l'incentivo può migrare interamente ai costi di transazione e essere completamente privo di effetti inflazionari.

L'incentivo può contribuire ad incoraggiare i nodi a rimanere onesti. Se un utente malintenzionato fosse avidamente in grado di assemblare più potenza di CPU rispetto a tutti i nodi onesti, dovrebbe scegliere tra un utilizzo truffaldino (ritornando al mittente tutti i suoi pagamenti), o un utilizzo volto a coniare nuove monete. Dovrà necessariamente trovare più redditizio giocare secondo le regole, dato che tali regole lo favoriscono con più monete nuove di tutti gli altri messi insieme, piuttosto che minare la sicurezza del sistema e la validità della propria ricchezza.

7. Rivendicare Spazio sul Disco

Una volta che l'ultima transazione in una moneta è sepolta sotto un numero sufficiente di blocchi, le transazioni spese prima di essa possono essere tralasciate al fine di risparmiare spazio sul disco. Per facilitare ciò senza rompere l'hash del blocco, le transazioni vengono sottoposte ad hash in un albero di Merkle [7] [2] [5], la cui sola radice è inclusa nella hash del blocco. I vecchi blocchi possono a quel punto essere compattati mediante la rimozione di ramificazioni dell'albero. Non c'è bisogno di immagazzinare in memoria gli hash interni.



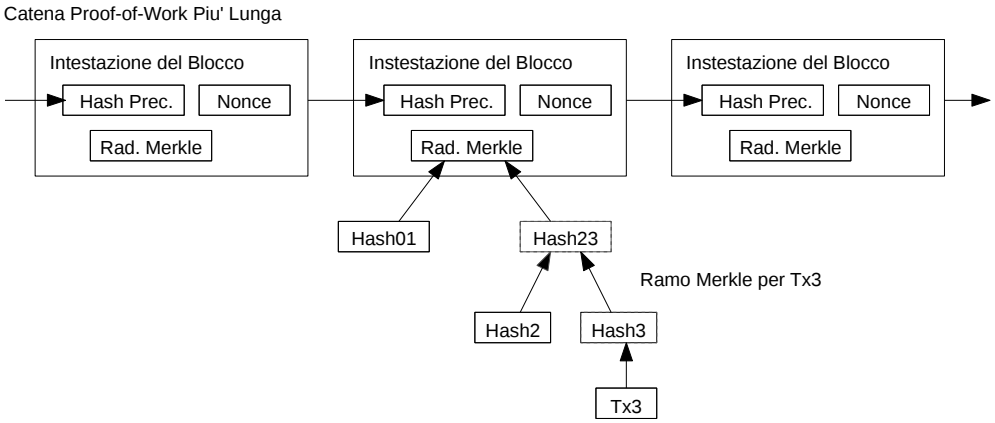
Transazioni Hashed in un Albero Merkle

Dopo aver rimosso Tx0-2 dal Blocco

La grandezza dell'intestazione di un blocco senza transazioni dovrebbe essere di circa 80 bytes. Se supponiamo che i blocchi siano generati ogni 10 minuti, 80 bytes * 6 * 24 * 365 = 4.2MB all'anno. Dato che nel 2008 sono in vendita computer con circa 2GB di RAM, e la Legge di Moore predice una crescita di 1.2GB per anno, lo spazio di archiviazione non dovrebbe rappresentare un problema anche se le intestazioni dei blocchi devono essere immagazzinate in memoria.

8. Verifica Semplificata dei Pagamenti

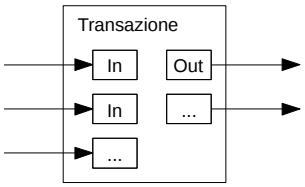
È possibile verificare i pagamenti senza far girare un nodo dell'intera rete. Un utente ha solo bisogno di conservare una copia delle intestazioni dei blocchi della catena proof-of-work più lunga, che può ottenere interrogando i nodi di rete finché è convinto di avere la catena più lunga, e ottenere il ramo Merkle che collega la transazione al blocco in cui è marcata temporalmente. L'utente non può controllare autonomamente la transazione, ma collegandola ad un posto nella catena, può vedere che un nodo di rete l'ha accettata, e i blocchi aggiunti successivamente ad essa confermano ulteriormente che la rete l'ha accettata.



Come tale, la verifica è affidabile fintanto che nodi onesti controllano la rete, ma è più vulnerabile se la rete è sopraffatta da un utente malintenzionato. Mentre i nodi di rete sono in grado di verificare le transazioni autonomamente, il metodo semplificato può essere ingannato da transazioni fabbricate da un utente malintenzionato fintanto che questi riesce a continuare a superare in potenza la rete. Una strategia per la protezione contro questo sarebbe accettare allarmi da nodi di rete quando rilevano un blocco non valido, spingendo il software dell'utente a scaricare il blocco completo e le transazioni dubbie per confermarne l'inconsistenza. Le aziende che ricevono pagamenti frequenti probabilmente vorrebbero ancora far girare i propri nodi per una sicurezza più indipendente e una verifica più veloce.

9. Aggregare e Suddividere Valore

Anche se fosse possibile movimentare le monete individualmente, non sarebbe saggio effettuare una transazione separata per ogni centesimo coinvolto in un trasferimento. Per far sì che il valore possa essere aggregato e suddiviso, le transazioni contengono molteplici input ed output. Normalmente ci sarà un singolo input a partire da una grande transazione precedente o più input che raggruppano quantità più piccole, e al massimo due output: uno per il pagamento, e uno per dare un eventuale resto al mittente.

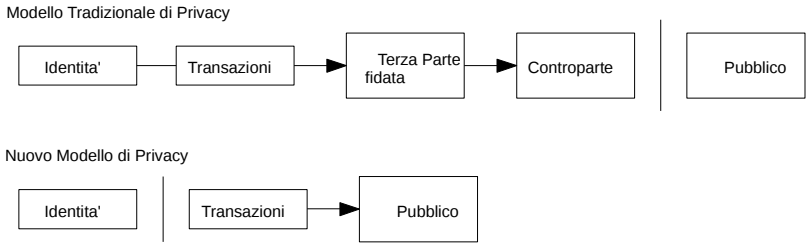


Notiamo a questo punto che il "fan-out", laddove una singola transazione dipende da molte altre, e queste altre dipendono ancora da altrettante, non è un problema. Non c'è mai bisogno di estrarre una copia completa della storia di una transazione.

10. Privacy

Il modello bancario tradizionale consegue un certo livello di privacy limitando l'accesso alle informazioni alle parti coinvolte e alla terza controparte fiduciaria. La necessità di annunciare pubblicamente tutte le transazioni preclude tale metodo, ma la privacy può essere ancora mantenuta rompendo il flusso di informazioni in un altro luogo: ovvero mantenendo anonime le chiavi pubbliche. Il pubblico può vedere che qualcuno sta inviando un certo importo a qualcun

altro, ma senza informazioni che collegano la transazione ad un utente specifico. Questo è simile al livello delle informazioni rilasciate dai mercati azionari, dove il tempo e la dimensione dei singoli scambi - il "nastro" - è reso pubblico, ma senza che si riveli l'identità delle controparti.



Come ulteriore schermo protettivo, una nuova coppia di chiavi deve essere utilizzata per ogni singola transazione, in modo da impedire che essa sia ricondotta ad un singolo proprietario. Qualche collegamento è ancora inevitabile nelle transazioni multi-input, che necessariamente rivelano che i loro input erano di proprietà dello stesso individuo. Il rischio è che, se il proprietario di una chiave fosse rivelato, si possa risalire ad altre transazioni effettuate dallo stesso.

11. Calcoli

Consideriamo ora lo scenario in cui un utente malintenzionato cerchi di generare una catena alternativa più veloce di quella onesta. Anche se ciò avvenisse, questo non rende il sistema vulnerabile a modifiche arbitrarie, come la creazione di valore dal nulla o l'appropriazione di denaro che non apparteneva all'utente malintenzionato. I nodi non accetterebbero una transazione non valida come forma di pagamento, e i nodi onesti non accetterebbero mai un blocco che la contiene. Un utente malintenzionato può solo cercare di cambiare una delle sue transazioni in modo da recuperare soldi che ha recentemente speso.

La gara tra la catena onesta e la catena di un utente malintenzionato può essere caratterizzata come una Passeggiata Aleatoria Binomiale (Binomial Random Walk). L'evento “successo” è che la catena onesta sia estesa di un blocco, aumentando il suo vantaggio di +1, e l'evento “fallimento” è che la catena dell'utente malintenzionato sia estesa di un blocco, riducendo il divario di -1.

La probabilità che un utente malintenzionato recuperi un determinato deficit è analoga al problema della “Rovina del Giocatore” (Gambler's Ruin). Supponiamo che un giocatore d'azzardo con credito illimitato parta da una situazione di deficit e giochi potenzialmente un numero infinito di puntate per cercare di raggiungere una situazione di pareggio. Siamo in grado di calcolare la probabilità che egli possa mai raggiungere il pareggio, o che un utente malintenzionato possa mai raggiungere la catena onesta, come [8] segue:

p = probabilità che un nodo onesto generi il prossimo blocco
q = probabilità che un utente malintenzionato generi il prossimo blocco
q_z = probabilità che un utente malintenzionato possa mai recuperare partendo da z blocchi dietro

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

Data la nostra ipotesi che p > q, la probabilità cala esponenzialmente man mano che aumenta il numero di blocchi che l'utente malintenzionato deve raggiungere. Con tutte le probabilità contro di lui, se non fa un affondo fortunato nella fase iniziale, le sue possibilità diventano irrisorie tanto più rimane indietro.

Adesso consideriamo quanto tempo il destinatario di una nuova transazione deve attendere prima di essere sufficientemente certo che il mittente non possa cambiare la transazione. Assumiamo che il mittente sia un utente malintenzionato che vuole fare credere per un po' al destinatario di averlo pagato, e quindi cambiare la transazione per ripagare sé stesso dopo che sia trascorso un certo lasso di tempo. Il beneficiario sarà avvisato quando ciò accade, ma il mittente spera che a quel punto sia troppo tardi.

Il beneficiario genera una nuova coppia di chiavi e dà la chiave pubblica al mittente poco prima di firmare. Questo impedisce al mittente di preparare una catena di blocchi in anticipo lavorando su di essa in continuazione fino a che non ha la fortuna di andare abbastanza avanti, quindi eseguire l'operazione in quel momento. Una volta che la transazione viene inviata, il mittente disonesto inizia a lavorare in segreto su una catena parallela contenente una versione alternativa della sua transazione.

Il destinatario attende che la transazione sia stata aggiunta ad un blocco e che z blocchi siano stati successivamente collegati ad esso. Non conosce l'ammontare esatto del progresso che l'utente malintenzionato ha compiuto, ma assumendo che ci sia voluto il tempo medio atteso per un blocco onesto, il progresso possibile dell'utente malintenzionato sarà una Distribuzione di Poisson con valore atteso:

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

Per ottenere la probabilità che un utente malintenzionato possa ancora recuperare in quel punto, moltiplichiamo la funzione di densità di probabilità Poisson per ciascun ammontare di progresso che egli può avere compiuto per la probabilità che possa recuperare da quel punto:

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

Riordinando per evitare di sommare la coda infinita della distribuzione...

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

Scritto in linguaggio C ...

```
#include <math.h>
double ProbabilitaSuccessoMalintenzionato(double q, int z)
{
    double p = 1.0 - q;
    double lambda = z * (q / p);
    double somma = 1.0;
    int i, k;
    for (k = 0; k <= z; k++)
    {
        double poisson = exp(-lambda);
        for (i = 1; i <= k; i++)
            poisson *= lambda / i;
        somma -= poisson * (1 - pow(q / p, z - k));
    }
    return somma;
}
```

Dai risultati, vediamo come la probabilità tenda esponenzialmente a zero all'aumentare di 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%...

```
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 non basato sulla fiducia. Abbiamo iniziato con il framework abituale delle valute basate su firme digitali, che prevede un forte controllo sulla proprietà, ma è incompleto non avendo modo di prevenire la doppia spesa. Per risolvere questo problema, abbiamo proposto una rete peer-to-peer che utilizza la proof-of-work

per registrare una storia pubblica delle transazioni, la cui modifica diventa rapidamente computazionalmente impraticabile per un utente malintenzionato se i nodi onesti controllano la maggioranza della potenza della CPU. La rete è robusta nella sua semplicità non strutturata. I nodi lavorano tutti insieme con poca coordinazione. Non hanno bisogno di essere identificati, dal momento che i messaggi non vengono instradati in qualche direzione particolare ma vengono solo consegnati su base best effort. I nodi possono lasciare e ricongiungersi con la rete a piacimento, accettando la catena proof-of-work come prova di quello che è successo mentre erano assenti. Essi votano con la loro potenza di CPU, esprimendo la loro accettazione di blocchi validi mediante il lavoro che compiono sulla loro estensione e respingendo i blocchi non validi tramite il rifiuto di lavorare sugli stessi. Tutte le regole e gli incentivi necessari possono essere applicati mediante questo meccanismo di consenso.

Riferimenti

[1] W. Dai, "b-money," http://www.weidai.com/bmoney.txt, 1998.

[2] 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.

[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In *Journal of Cryptology*, vol 3, no 2, pagine 99-111, 1991.

[4] D. Bayer, S. Haber, W.S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," *In Sequences II: Methods in Communication, Security and Computer Science*, pagine 329-334, 1993.

[5] S. Haber, W.S. Stornetta, "Secure names for bit-strings," In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pagine 28-35, Aprile 1997.

[6] A. Back, "Hashcash - a denial of service counter-measure," http://www.hashcash.org/papers/hashcash.pdf, 2002.

[7] R.C. Merkle, "Protocols for public key cryptosystems," In Proc. *1980 Symposium on Security and Privacy*, IEEE Computer Society, pagine 122-133, Aprile 1980.

[8] W. Feller, "An introduction to probability theory and its applications," 1957.