Bitcoin: un sistema di contanti elettronico peer-to-peer

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

Abstract. Una versione puramente peer-to-peer di denaro elettronico permetterebbe di effettuare pagamenti online direttamente da un soggetto ad un altro senza passare attraverso un istituto finanziario. Le firme digitali forniscono una parte della soluzione, ma i principali vantaggi si perdono se è ancora necessario un terzo garante per evitare la possibilità di utilizzare la stessa moneta per due spese diverse. Si propone una soluzione al problema della spesa doppia utilizzando un network peer-to-peer Il network esegue una marcatura temporale sulle transazioni sottoponendole ad hash in una continua catena di prove basate sugli hash stessi, che formano un risultato che non può essere cambiato senza rifare tutto il lavoro che lo prova. La catena più lunga non serve solo come prova della sequenza di eventi che ha testimoniato, ma anche come prova che è venuta dal gruppo più esteso di potenza di calcolo. Finchè la maggioranza del potere di calcolo è controllata da nodi che non si alleano per attaccare il network stesso, saranno essi a generare la catena più lunga, lasciando indietro eventuali attaccanti. Il network in sè richiede una struttura minima. I messaggi sono pubblicati secondo le possibilità di ciascuno, e i nodi possono unirsi o lasciare il network quando vogliono, e accettare la catena di risultati più lunga come prova di quanto è avvenuto mentre erano disconnessi.

1. Introduzione

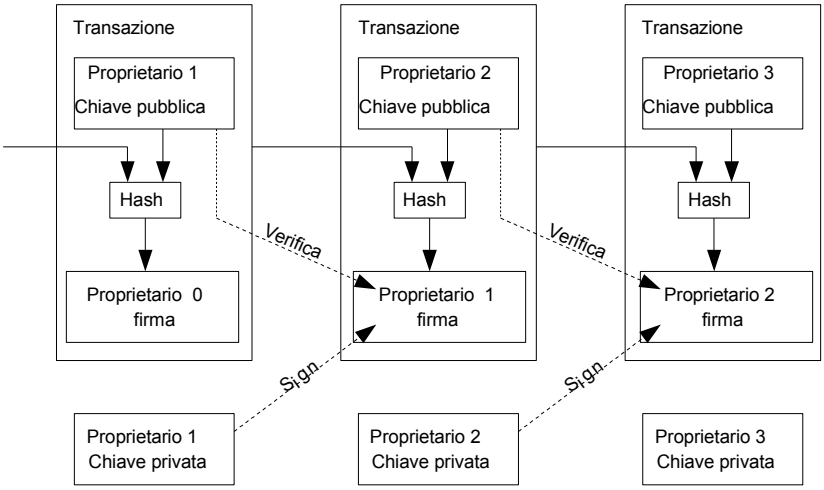
Il commercio online si appoggia quasi esclusivamente su istituzioni finanziarie che fanno da terzi garanti nel processo di pagamento elettronico. Anche se il sistema funziona bene per la maggior parte delle transazioni, soffre della debolezza intrinseca in un modello basato sulla fiducia. Siccome le istituzioni finanziarie non possono esimersi dal cercare una mediazione in caso di dispute, non è veramente possibile generare transazioni non annullabili. Il costo di tale mediazione si aggiunge a quello della transazione, limitando la dimensione pratica delle transazioni ed escludendo la possibilità di transazioni di scarsa entità; e c'e' una spesa ulteriore, visto che i servizi non si possono "annullare", ma è impossibile creare pagamenti non annullabili. A causa della possibilità di richiedere un annullamento del pagamento, la necessità di fiducia aumenta. I negozianti devono essere cauti con i clienti e seccarli richiedendo più informazioni di quante ne dovrebbero servire. Una determinata quota di frodi è accettata come inevitabile. Questi costi, e le incertezze nel pagamento, possono essere evitati negoziando di persona e utilizzando fisicamente del contante, ma non esiste alcun meccanismo per eseguire pagamenti tramite semplice comunicazione senza un terzo che faccia da garante.

Ciò che serve è un sistema di pagamento elettronico basato su prove crittografiche, invece che sulla fiducia, che consenta a soggetti consenzienti di negoziare direttamente

tra loro senza la necessità di un garante terzo. Creando delle transazioni che, a livello di potenza di calcolo, risultassero impraticabili da decrittare e annullare, si proteggerebbero i venditori da frodi, e per proteggere gli acquirenti si potrebbero facilmente implementare meccanismi di deposito a garanzia. In questo documento si propone una soluzione al problema della spesa doppia, utilizzando un server per la marcatura temporale distribuito in maniera peer-to-peer che generi una prova computazionale dell'ordine cronologico delle transazioni. Il sistema è sicuro finchè i nodi onesti controllano collettivamente più potere di calcolo di qualsiasi gruppo di nodi che cerchi di attaccarlo contemporaneamente.

2. Transazioni

Si definisce un gettone elettronico come una catena di firme digitali. Ogni proprietario trasferisce il gettone al seguente proprietario firmando digitalmente l'hash della transazione precedente e la chiave pubblica del futuro proprietario, e aggiungendo queste informazioni al termine del gettone. Chi riceve un pagamento piò verificare le firme per verificare la catena di possesso.



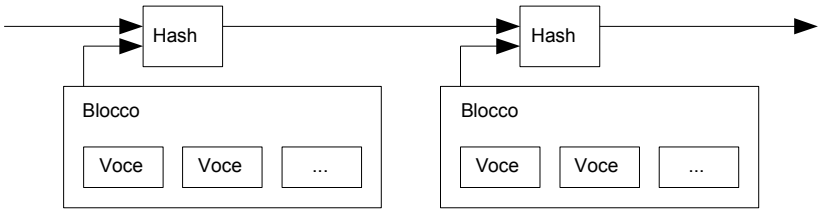
Chiaramente il problema è che chi riceve il pagamento non può verificare che chi spedisce il denaro non lo abbia già speso. Una soluzione comune è quella di istituire un'autorità centrale degna di fiducia, detta Zecca, che controlla ciascuna transazione per evitare la spesa doppia di una singola moneta. Dopo ogni transazione, il gettone deve essere restituito alla Zecca per generare un nuovo gettone, e solo per i gettoni che escono direttamente dalla Zecca ci si può fidare che non siano stati spesi due volte. Il problema di questa soluzione è che il destino dell'intero sistema monetario dipende dalla società che gestisce la Zecca, visto che ogni transazione deve passare attraverso di essa, proprio come in una banca.

E' quindi necessario che chi riceve un pagamento sia certo che il precedente proprietario della moneta non abbia già firmato un'altra transazione. Per lo scopo in

questione, la prima transazione è quella che conta, in modo da non doversi preoccupare dei tentativi di doppia spesa successivi. L'unico modo di confermare l'assenza di una transazione è conoscerle tutte. Nel modello basato sulla Zecca, la Zecca era a conoscenza di tutte le transazioni e decideva quale era la prima. Per ottenere ciò senza un terzo garante, le transazioni devono essere pubblicamente annunciate e sarà necessario un sistema che consenta ai partecipanti di concordare su un'unica evoluzione storica dell'ordine in cui le transazioni sono state ricevute. Chi riceve il pagamento deve avere una prova che, al momento di ciascuna transazione, la maggioranza dei nodi concordasse che era effettivamente la prima ricevuta.

3. Server di marcatura temporale

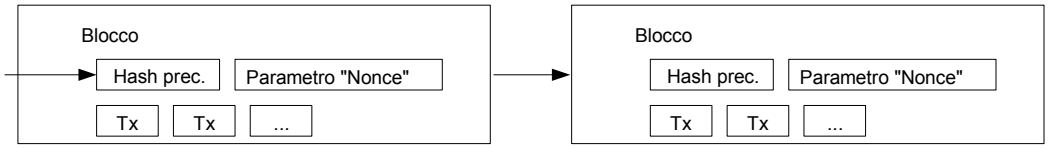
La soluzione proposta comincia con un server di marcatura temporale. Questo funziona sottoponendo ad hash un blocco di dati da marcare e pubblicando l'hash stesso a tutti, esattamente come fa un quotidiano o un post su Usenet [2-5]. La marcatura prova che i dati erano ovviamente già esistenti al momento dell'operazione, dato che sono stati inclusi nell'hash. Ogni marcatura include la precedente nel proprio hash, formando dunque una catena in cui ciascuna marcatura rinforza le precedenti.



4. Prova del Lavoro

Per implementare un server di marcatura temporale su base peer-to-peer sarà necessario un sistema di "prova del lavoro svolto" simile a "Hashcash" [6] di Adam Back, piuttosto che ricorrere ad un quotidiano o ai post su Usenet. La "prova del lavoro" consiste nel cercare un valore tale che, quando sottoposto ad hash (ad esempio con SHA-256), faccia risultare un hash che cominci con un certo numero di bit a zero. In media, il lavoro necessario al risultato è esponenziale rispetto al numero di bit a zero richiesti e si può verificare eseguendo un singolo hash.

Per il network di marcatura temporale, si implementa la prova del lavoro aumentando un apposito parametro detto "nonce" nel blocco finchè l'hash del blocco stesso ha il numero necessario di bit a zero. Una volta che i processori sono stati impegnati per soddisfare la prova del risultato, il blocco non può essere cambiato senza rifare tutto il lavoro. Visto che i blocchi successivi sono legati dopo di esso, il lavoro necessario per cambiare il blocco dovrebbe includere il rifacimento di tutti i blocchi dopo di esso.



Il sistema di prova del lavoro risolve anche il problema di determinare la maggioranza nel processo di decisione. Se la maggioranza fosse basata su "un IP, un voto", potrebbe essere falsificata da chiunque abbia la possibilità di allocare molti IP. La prova del risultato è essenzialmente: "un processore, un voto" La decisione della maggioranza è rappresentata dalla catena più lunga, che ha richiesto l'investimento di sforzo maggiore nella prova del lavoro. Se la maggior parte delle potenzialità di calcolo è controllata da nodi onesti, la catena onesta crescerà più velocemente e supererà qualsiasi catena concorrente. Per modificare un blocco passato, sarebbe necessario rifare la prova del lavoro del blocco stesso e di tutti quelli seguenti, e poi raggiungere e superare il lavoro dei nodi onesti. Si dimostrerà in seguito che la probabilità che un attacco lento raggiunga la catena di maggioranza diminuiscono esponenzialmente man mano che ulteriori blocchi vengono aggiunti alla catena stessa.

Per compensare l'aumento della velocità degli strumenti di calcolo e incentivare l'interesse a tenere operativi i nodi nel tempo, la difficoltà di provare i risultati è determinata da una media mobile orientata a un numero costante di blocchi all'ora. Se i blocchi vengono generati troppo velocemente, la difficoltà aumenta.

5. Rete

I passi per far funzionare il network sono i seguenti:

1. le nuove transazioni sono pubblicate a tutti i nodi.
2. ogni nodo raccoglie le nuove transazioni in un blocco.
3. ogni nodo lavora per trovare una "prova del lavoro" difficile per il proprio blocco.
4. quando un nodo trova una prova del lavoro, pubblica il blocco a tutti gli altri nodi.
5. i nodi accettano il blocco solo se tutte le transazioni in esso sono valide e non ono già state spese una volta.
6. i nodi esprimono la loro accettazione del blocco lavorando per creare il blocco seguente nella catena, utilizzando l'hash del blocco accettato come hash precedente.

I nodi considerano corretta sempre la catena più lunga e continueranno a lavorare per espanderla. Se due nodi pubblicano differenti versioni del prossimo blocco contemporaneamente, alcuni nodi riceveranno per primo l'uno o l'altro. In quel caso, lavoreranno sul primo che hanno ricevuto, ma salveranno l'altro nel caso dovesse diventare più lungo. L'equilibrio si romperà quando verrà trovata la seguente prova del

risultato e uno dei rami diventerà più lungo; i nodi che stavano lavorando sull'altro passeranno quindi a quello più lungo.

La pubblicazione di una nuova transazione non deve necessariamente raggiungere tutti i nodi. Quando le trasmissioni raggiungono vari nodi, in breve tempo saranno inserite in un blocco. La pubblicazione dei blocchi tollera anche la possibilità di messaggi persi. Se un nodo non riceve un blocco, lo richiederà appena riceve il successivo e scoprirà che ne ha perso uno.

6. Incentivi

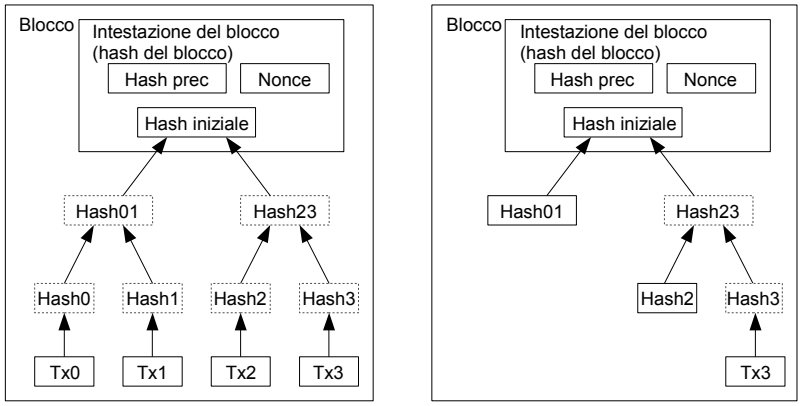
Per convenzione, la prima transazione di un blocco e' una transazione speciale che da' inizio ad un nuovo gettone posseduto dal creatore del blocco stesso. Ciò consente un ulteriore incentivo ai nodi che supportano il network, e fornisce un metodo per la distribuzione iniziale delle monete in circolazione, visto che non c'e' un'autorità centrale ad emetterle. L'aumento continuo di un numero costante di nuove monete è analogo allo sforzo dei minatori d'oro, che spendono risorse per aumentare la quantità d'oro in circolazione. In questo caso, si tratta della spesa in tempo macchina ed elettricità.

Gli incentivi si possono anche trovare nelle commissioni. Se il valore d'uscita di una transazione è minore di quello d'entrata, la differenza è considerata una commissione che viene inclusa all'incentivo del blocco che include la transazione stessa. Una volta che un determinato numero di monete è stato messo in circolazione, l'incentivo si trasferirà interamente nelle commissioni e diventerà del tutto privo di inflazione.

Gli incentivi dovrebbero incoraggiare i nodi a rimanere onesti. Se un attacco esterno fosse capace di radunare maggiore capacità di calcolo di tutti gli altri nodi, dovrebbe decidere se utilizzare tale capacità per frodare gli altri utenti riprendendosi i propri pagamenti, o piuttosto per generare nuovi gettoni. Anche in questo caso sarebbe più conveniente giocare secondo le regole, visto che esse avvantaggiano l'attaccante con nuovi gettoni in quantità maggiore rispetto a chiunque altro, piuttosto che distruggere il sistema e quindi la validità delle proprie monete.

7. Necessità di spazio su disco

Quando l'ultima transazione in un gettone è sepolta sotto un numero sufficiente di blocchi, le transazioni spese in precedenza possono essere scartate in modo da risparmiare spazio su disco. Per facilitare ciò senza interrompere l'hash del blocco, le transazioni vengono sottoposte a hash secondo una struttura di Merkle ad albero [7][2][5], della quale viene inserita nell'hash del blocco solo la radice. I blocchi vecchi possono essere compattati tagliando i rami dell'albero. Non serve memorizzare gli hash interni.

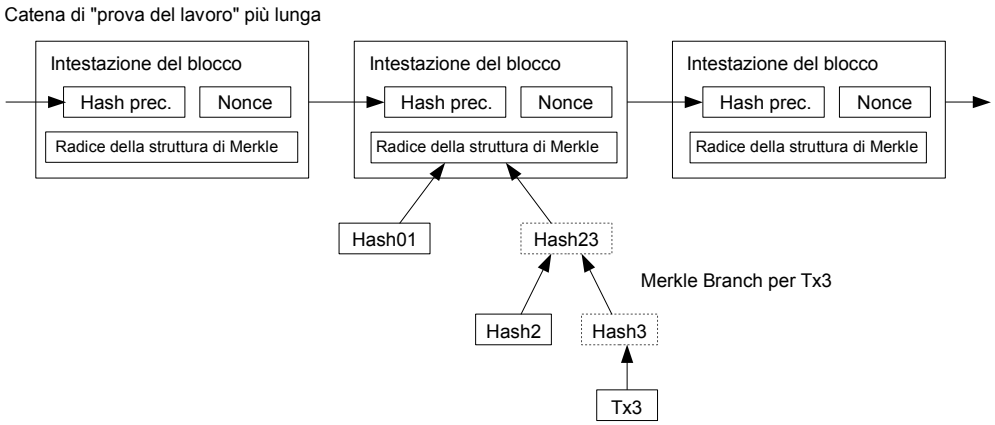


Transazioni sottoposte ad Hash in una struttura di Merkle dopo la potatura di Tx0-2 dal blocco

L'intestazione di un blocco senza transazione sarebbe di circa 80 byte. Se si suppone che i blocchi vengono generati ogni 10 minuti, 80 byte * 6 * 24 * 365 = 4,2 MB all'anno. Dato che il computer medio nel 2008 si vende con 2GB di RAM, e la Legge di Moore prevede la crescita attuale di 1,2 GB per ogni anno, lo spazio su disco non dovrebbe essere un problema anche se le intestazioni blocco devono essere mantenute in memoria.

8. Verifica di pagamento semplificata

E' possibile verificare i pagamenti senza operare un intero nodo della rete. L'utente ha bisogno solo di conservare una copia delle intestazioni blocco della catena di prova del risultato più lunga, che può ottenere interrogando i nodi del network finchè è convinto che sia effettivamente la più lunga, e di ottenere il ramo Merkle che collega la transazione al blocco su cui è marcata temporalmente. Non può controllare la transazione da solo, ma collegandola ad una posizione nella catena, può vedere che un nodo del network l'ha accettata, e i blocchi aggiunti dopo di essa confermano ulteriormente che l'intero network l'ha accettata.

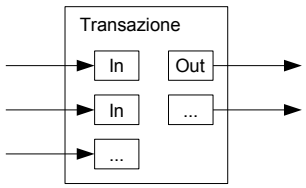


In quanto tale, la verifica è affidabile finchè il network è controllato da nodi onesti, ma è più vulnerabile se esposta ad un attacco con potenza di calcolo superiore. Se i nodi del

network possono verificare le transazioni da soli, il metodo semplificato si può ingannare con transazioni falsificate finchè chi attacca il sistema riesce a superare la potenza del network. Una strategia per proteggersi da ciò sarebbe accettare allarmi dai nodi del network che individuano un blocco invalido, così da spingere l'utente a scaricare l'intera catena dei blocchi e delle transazioni sotto allarme per confernarne l'inconsistenza. Le aziende che ricevono pagamenti frequentemente probabilmente vorranno gestire i propri nodi per una maggiore sicurezza indipendente e per verifiche veloci.

9. La combinazione e la divisione degli importi

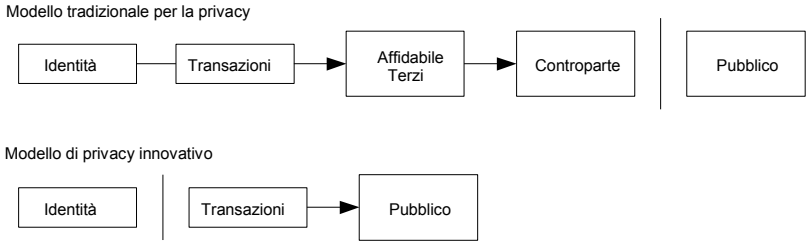
Anche se è possibile gestire singole monete, sarebbe faticoso fare una transazione separata per ogni centesimo di un pagamento. Per consentire ad un valore di essere diviso e combinato con altri, le transazioni contengono diversi input e output. Normalmente ci sarà o un unico input da una transazione precedente più grande, o più ingressi che includono piccoli importi, e al massimo due output: uno per il pagamento, ed uno di ritorno al mittente con l'eventuale resto.



E' opportuno notare che la distribuzione multipla, ossia che una transazione dipende da diverse transazioni, che a loro volta dipendono da molte altre ancora, non costituisce un problema in questo caso. Non sarà mai necessario estrarre una copia completa ed indipendente dello storico della transazione.

10. Privacy

Il modello bancario tradizionale riesce ad assicurare una certa privacy limitando l'accesso alle informazioni ai soli soggetti coinvolti, oltre che al terzo garante. La necessità di annunciare tutte le transazioni pubblicamente impedisce questo metodo, tuttavia si può ancora mantenere la privacy interrompendo il flusso di informazioni in un altro punto: mantenendo anonime le chiavi pubbliche. Tutti possono vedere che qualcuno sta spedendo un importo a qualcun altro, ma senza poter legare le informazioni della transazione a un soggetto particolare. Ciò è simile al livello di informazione pubblicato dalle borse valori, dove il momento e la dimensione delle singole transazioni, il "nastro", viene pubblicato senza dire chi erano i soggetti coinvolti.



Come ulteriore protezione, si dovrebbe utilizzare una nuova coppia di chiavi per ciascuna transazione, in modo da evitare che diverse chiavi siano riconducibile ad un unico proprietario. Alcuni collegamenti sono ancora inevitabili nel caso di transazioni molteplici, che necessariamente rivelano che i loro punti di partenza erano posseduti dallo stesso proprietario. Il rischio è che se viene rivelato il proprietario di una chiave, i collegamenti possano rivelare altre transazioni fatte dallo stesso proprietario.

11. Calcoli

Si consideri lo scenario di un attacco che cerchi di generare una catena alternativa più velocemente di quella onesta. Anche se ciò venisse attuato, non precipiterebbe il sistema in una situazione di modifiche arbitrarie, come creare valore da zero o rubare denaro che non è mai stato posseduto da chi attacca il sistema. I nodi non accetteranno una transazione invalida come pagamento, e i nodi onesti non accetteranno mai un blocco che ne contenga. Un attacco può solo cercare di cambiare le proprie transazioni per riprendersi il denaro che ha spedito recentemente.

La gara tra la catena onesta e quella attaccante si può descrivere come un Random Walk Binomiale. L'evento "successo" consiste nell'aggiunta di un blocco alla catena onesta, che aumenta il suo distacco di +1, mentre l'evento "fallimento" è l'estensione di un blocco nella catena attaccante, che riduce il proprio ritardo di -1.

La probabilità che attacco raggiunga la prova del lavoro colmando un determinato distacco è analoga al problema detto Gambler's Ruin (dilemma del giocatore d'azzardo). Si supponga che un giocatore d'azzardo con credito illimitato cominci in perdita e giochi un numero di tentativi potenzialmente infinito, per cercare di raggiungere un pareggio. E' possibile calcolare la probabilità che un attacco raggiunga il pareggio, o che riesca mai a raggiungere una catena onesta, come segue [8]:

p = Probabilità che un nodo onesto trovi il prossimo blocco
q = Probabilità l'attaccante trovi il prossimo blocco
q_z = Probabilità l'attaccante potrà mai raggiungere da un z blocchi indietro

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

Supponendo che $p > q$, la probabilità diminuisce esponenzialmente rispetto all'aumentare del numero dei blocchi che l'attacco deve raggiungere. Con le probabilità avverse, se non

si ottiene un avanzamento fortunato molto presto, le possibilità si annullano man mano che si rimane sempre più indietro.

Si considera ora quanto tempo sia necessario attendere perchè chi riceve una transazione sia sufficientemente sicuro che il mittente non possa cambiarla. Si ipotizza che il mittente stia attaccando il ricevente facendogli credere che l'ha pagato per un certo periodo, e poi rispedendosi indietro il pagamento dopo che è passato un po' di tempo. Il ricevente sarà avvisato quando ciò accade, ma il mittente spera che sarà troppo tardi.

Il ricevente genera una nuova coppia di chiavi e fornisce quella pubblica al mittente poco prima di firmare. Ciò impedisce al mittente di preparare una catena di blocchi in anticipo, lavorandoci continuamente finchè non è abbastanza fortunato di arrivare abbastanza avanti, e poi di eseguire la transazione nel momento adatto. Una volta inviata la transazione, il mittente disonesto comincerà a lavorare segretamente su una catena di blocchi parallela che contenga una versione differente della transazione stessa.

Il destinatario attende finche' la transazione è stato aggiunta a un blocco e *z* i blocchi che sono stati collegati dopo di esso Non conosce l'esatto progresso raggiunto dall'attacco, ma supponendo che i nodi onesti abbiano usato il tempo medio necessario per ciascun blocco, il potenziale progresso dell'attacco sarà una distribuzione di Poisson con valore atteso:

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

Per calcolare la probabilità che un attacco possa ancora raggiungere la prova del risultato attuale, si moltiplica la densità di Poisson per ciascun tipo di progresso che potrebbe aver fatto per la probabilità che possa raggiungere la catena partendo da esso.

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

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

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

Conversione al codice C ...

```
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
	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 / p, z - k));
	}
	return sum;
}
```

Calcolando alcuni risultati, si vede che la probabilità diminuisce esponenzialmente rispetto a 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 inferiore a 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. Conclusioni

E' stato proposto un sistema per transazioni elettroniche che non si basa sulla fiducia. Si è cominciato con una normale struttura di monete a base di firme digitali, che consente un forte controllo sulla proprietà, ma è incompleta senza un modo per evitare la doppia spesa. Per risolvere ciò, si propone un network peer-to-peer basato sul sistema "prova di lavoro" che registri l'elenco storico pubblico delle transazioni, e che diventi in breve tempo impraticabile da calcolare e cambiare da parte di un attacco esterno, semprechè la maggioranza della potenza di calcolo sia controllata dai nodi onesti. Il network è robusto grazie alla sua semplicita' destrutturata. I nodi lavorano contemporaneamente senza necessità di gran coordinazione. Essi non hanno bisogno di essere identificati, poichè i

messaggi non sono inoltrati a nessun indirizzo particolare e devono essere consegnati solo secondo possibilità. I nodi possono unirsi o lasciare il network quando vogliono, e accettano la catena della prova di lavoro come prova di quanto è successo mentre erano disconnessi. Questi votano tramite la propria potenza di calcolo, esprimendo l'accettazione di blocchi validi e lavorando all'estensione di essi, espellendo quelli invalidi rifiutandosi di elaborarli. Eventuali regole e incentivi si possono imporre con il meccanismo del consenso.

Riferimenti

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

[2] H. Massias, X.S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal requisiti fiducia," In *20 Simposio sulla teoria dell'informazione nel 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, WS Stornetta, "Migliorare l'efficienza e l'affidabilità delle marcature temporali digitali" In *Sequenze II: Metodi di Scienze della Comunicazione, Sicurezza e Computer*, Pagine 329-334, 1993.

[5] S. Haber, WS Stornetta, "Secure names for bit-strings" In *Atti della 4a Conferenza ACM sul computer e delle comunicazioni di sicurezza*, Pagine 28-35, aprile 1997.

[6] Torna A., "HashCash - una contromisura al denial of service", http://www.hashcash.org/papers/hashcash.pdf, 2002.

[7] RC Merkle, "Protocolli per sistemi a chiave pubblica:" In *Atti 1980 Convegno sulla Sicurezza e la Privacy*, IEEE Computer Society, pagine 122-133, aprile 1980.

[8] W. Feller, "Introduzione alla teoria della probabilità e le sue applicazioni", 1957.