Bitcoin: Sebuah Sistem Uang Tunai Elektronik Peer-to-Peer

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

Translated into Indonesian from bitcoin.org/bitcoin.pdf

Abstrak. Sebuah versi peer-to-peer tunai elektronik murni yang memungkinkan pembayaran online dapat dikirim langsung dari satu pihak ke pihak lain tanpa melalui sebuah institusi keuangan. *Digital signature* dapat menjadi salah satu solusinya, namun manfaat utamanya hilang jika masih membutuhkan peran pihak ketiga untuk bisa mencegah *double-spending*. Kami menawarkan solusi untuk mengatasi masalah double-spending tersebut menggunakan jaringan *peer-to-peer*. Jaringan memberi stempel waktu *(timestamp)* transaksi dengan cara hashing ke dalam sebuah rantai *proof-of-work* bersambung berbasis hash, membuat rekam data yang tidak dapat dirubah tanpa mengulang kembali proof-of work. Rantai yang terpanjang tidak hanya berfungsi sebagai sebuah bukti dari urutan peristiwa yang telah disaksikan, melainkan juga bukti bahwa itu berasal dari *pool* daya CPU terbesar. Selama mayoritas daya CPU tetap dikontrol oleh *node* dan tidak berupaya untuk menyerang jaringan, mereka akan membuat rantai terpanjang dan mampu melampaui penyerang. Jaringan itu sendiri memerlukan struktur yang minim. Broadcast pesan dilakukan secara *best effort basis*, dan *node* dapat datang dan pergi sekehendaknya, menerima rantai proof-of-work terpanjang sebagai bukti atas apa yang telah terjadi pada saat ia meninggalkan jaringan.

1. Pendahuluan

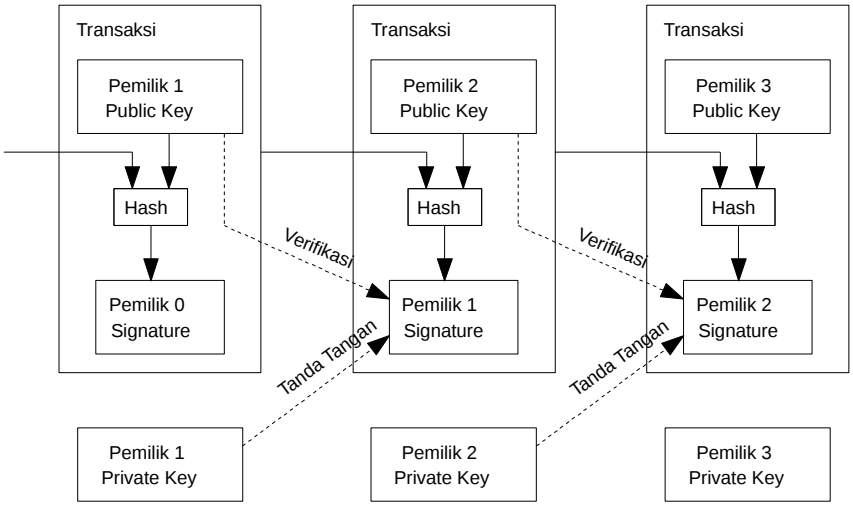
Perdagangan elektronik di Internet sejauh ini hampir seluruhnya berjalan secara ekslusif melalui institusi keuangan sebagai pihak ketiga yang dipercaya dalam memproses pembayaran elektronik. Meski sistem ini berfungsi cukup baik pada kebanyakan transaksi, sistem ini masih memiliki kelemahan bawaan dari model yang berbasis kepercayaan. Transaksi yang bersifat *non-reversible* penuh tidak memungkinkan, sejak institusi keuangan tidak bisa mengabaikan peran mediasi sengketa. Biaya mediasi itu meningkatkan biaya transaksi, membatasi besaran transaksi minimun secara praktis dan memotong kemungkinan transaksi-transaksi kecil secara umum, serta ada tambahan biaya karena ketidakmampuan membuat pembayaran secara *non-reversible* untuk layanan yang bersifat *nonreversible*. Dengan kemungkinan membalikkan transaksi itu, tingkat kepercayaan yang dibutuhkan meningkat. Merchant semestinya berhati-hati kepada pelanggan, merepotkannya untuk menambahkan informasi lebih dari yang tidak mereka perlukan. Sekian prosentase terjadinya penipuan menjadi tidak dapat dihindari. Biaya dari pembayaran yang tidak pasti ini bisa dihindari seseorang dengan menggunakan mata uang fisik, namun tidak ada mekanisme yang memungkinkan melakukan pembayaran melalui channel komunikasi tanpa perantara pihak yang dipercaya.

Yang dibutuhkan adalah sistem pembayaran elektronik berbasis bukti kriptografi untuk menggantikan kepercayaan, memungkinkan dua belah pihak untuk bertransaksi secara langsung satu sama lain tanpa membutuhkan pihak ketiga yang dipercaya. Transaksi yang dilakukan melalui

komputasi tidak dapat dibalikkan sehingga dapat melindungi penjual dari penipuan, dan mekanisme *escrow* pada umumnya dapat cukup mudah diimplementasikan untuk melindungi pembeli. Di dalam makalah ini, kami menawarkan solusi untuk mengatasi masalah *double-spending* menggunakan server *timestamp* yang didistribusikan secara peer-to-peer untuk membuat bukti melalui proses komputasi berdasarkan urutan kronologis transaksi. Sistem ini aman selama *node* jujur secara kolektif mengendalikan daya CPU lebih besar ketimbang sekelompok *node* penyerang.

2. Transaksi

Kami mendefinisikan sebuah koin elektronik sebagai sebuah rantai *digital signature*. Setiap pemilik mentransfer koin kepada pemilik berikutnya dengan menandatangani *hash* transaksi sebelumnya dan *public key* pemilik berikutnya dan menambahkannya di bagian ujung koin. Seorang *payee* dapat memverifikasi signature untuk memastikan rantai kepemilikan. Kami mendefinisikan sebuah koin elektronik sebagai sebuah rantai *digital signature*. Setiap pemilik mentransfer koin kepada pemilik berikutnya dengan menandatangani *hash* transaksi sebelumnya dan *public key* pemilik berikutnya dan menambahkannya di bagian ujung koin. Seorang *payee* dapat memverifikasi signature untuk memastikan rantai kepemilikan.



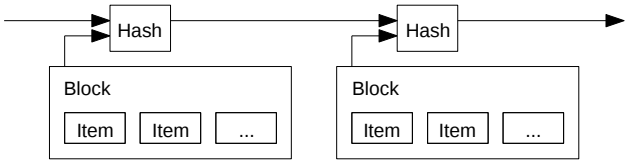
Masalahnya tentu saja adalah karena *payee* tidak dapat memverifikasi salah satu pemilik yang tidak melakukan *double-spend* koin itu. Solusi umum yang digunakan adalah menggunakan otoritas pusat yang dipercaya, atau penerbit koin, yang bertugas memeriksa setiap transaksi dari potensi *double-spend.* Setelah tiap transaksi, koin itu harus dikembalikan ke penerbit untuk membuat sebuah koin baru lagi, dan koin yang berasal langsung dari penerbit itulah yang dipercaya tidak akan melakukan *double-spent.* Masalah dengan solusi ini adalah keyakinan pada seluruh sistem uang yang bergantung pada perusahaan dibalik penerbit uang itu, dengan setiap transaksi yang berjalan melaluinya, sama seperti sebuah bank.

Kita butuh sebuah cara agar *payee* mengetahui pemilik mana yang belum membubuhkan tandatangan transaksi apapun sebelumnya. Untuk tujuan kami, transaksi paling awal adalah satu-satunya yang diperhitungkan, jadi kita tidak perlu peduli lagi dengan upaya *double-spend* nantinya. Satu-satunya cara untuk mengkonfirmasi ketiadaan dalam sebuah transaksi adalah dengan mengawasi keseluruhan transaksi. Dalam model yang berbasis penerbitan mata uang, penerbit harus mewaspadai seluruh transaksi dan memutuskan transaksi mana yang datang lebih dulu. Untuk mencapai hal ini tanpa menggunakan pihak yang dipercayai, transaksi haruslah

diumumkan secara terbuka [1], dan kita membutuhkan sebuah sistem agar para partisipan menyetujui satu urutan sejarah pertama yang telah mereka terima. *Payee* memerlukan bukti di setiap saat pada setiap transaksi, mayoritas *node* menyetujui bahwa itu yang pertama kali diterima.

3. Server Timestamp

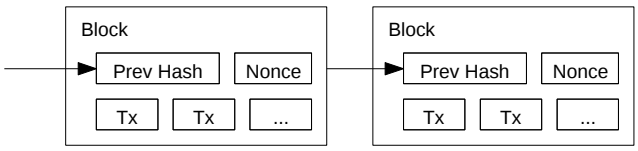
Solusi yang kami usulkan dimulai dengan server *timestamp*. Server timestamp ini bekerja dengan mengambil hash dari item sebuah block untuk diberi stempel waktu dan mempublikasikan hash itu secara luas, seperi halnya sebuah surat kabar atau posting di Usenet [2-5]. *Timestamp* membuktikan bahwa data itu harus ada saat itu, jelas agar bisa mendapatkan hash tersebut. Setiap *timestamp* menyertakan juga *timestamp* sebelumnya yang ada di dalam hash, menyusun sebuah rantai, dengan masing-masing tambahan *timestamp* memperkuat *timestamp* sebelumnya.



4. Proof-of-Work

Untuk mengimplementasikan server timestamp yang terdistribusi berbasis peer-to-peer, kita akan perlu untuk menggunakan sebuah sistem *proof-of-work* yang mirip dengan HashCash dari Adam Back [6], ketimbang sebuah surat kabar atau post di Usenet. *Proof-of-work* mencakup scanning nilai saat melakukan hashing, seperti dengan menggunakan SHA-256, hash dimulai dengan sebuah angka nol dalam bilangan biner. Rata-rata upaya yang diperlukan akan sebanding secara eksponensial dengan jumlah angka nol bit yang diperlukan dan dapat diverifikasi dengan sekali eksekusi hash.

Dalam jaringan timestamp kita, kami mengimplementasikan *proof-of-work* dengan menambahkan sebuah *nonce* di dalam block sampai ditemukan sebuah nilai yang dapat memberikan nol bit hash block yang dibutuhkan. Setelah melakukan upaya dengan menggunakan CPU untuk memenuhi *proof-of-work*, block tersebut tidak dapat dirubah tanpa mengulang kembali kerjanya. Karena block berikutnya akan dirantai tepat setelahnya, upaya untuk merubah block haruslah merubah seluruh block berikutnya juga.



Proof-of-work juga memecahkan masalah dalam menentukan representasi pengambilan keputusan mayoritas. Jika mayoritas tersebut didasarkan pada satu-IP-address-satu-suara, maka hal itu bisa digulingkan oleh siapapun yang mampu mengalokasikan IP terbanyak. Proof-of-work pada dasarnya merupakan satu-CPU-satu-suara. Keputusan mayoritas ditentukan dari rantai terpanjang, yang memiliki upaya proof-of-work terbaik yang telah ditanamkan ke dalamnya. Jika mayoritas daya CPU dikendalikan oleh *node* yang jujur, maka rantai yang jujur pula akan berkembang pesat melebihi rantai kompetitor lain. Untuk mengubah block lama, seorang penyerang harus pula mengulang proof-of-work block tersebut beserta seluruh block berikutnya dan kemudian berusaha

untuk terus melampaui upaya node yang jujur. Kami akan tunjukkan nantinya bahwa probabilitas penyerang yang pada akhirnya akan lamban dan melemah secara eksponensial ketika block-block baru berikutnya telah ditambahkan.

Untuk mengimbangi meningkatnya kecepatan hardware dan minat yang berbeda-beda dalam menjalankan node dari waktu ke waktu, tingkat kesulitan *(difficulty) proof-of-work* ditentukan oleh rata-rata pergerakan angka target block per jam. Jika mereka dibuat dalam waktu yang terlalu cepat, tingkat kesulitan akan bertambah.

5. Jaringan

Berikut beberapa langkah untuk menjalankan jaringan:

1) Transaksi baru dibroadcast kepada seluruh *node*.
2) Setiap *node* mengumpulkan transaksi baru ke dalam sebuah block.
3) Setiap node bekerja untuk menemukan proof-of-work yang rumit untuk block tersebut.
4) Ketika satu *node* menemukan proof-of-work, lalu menyiarkan block tersebut kepada seluruh node.
5) *Node* menerima block hanya jika seluruh transaksi yang ada didalamnya valid dan belum digunakan.
6) *Node* menyatakan menerima block dengan mengerjakan block berikutnya pada rantai block, dengan menggunakan block yang telah diterima itu sebagai hash sebelumnya.

Node selalu cenderung untuk menganggap rantai terpanjang menjadi yang paling benar dan terus mengejakan rantai block tersebut. Jika dua *node* menyiarkan versi yang berbeda untuk block berikutnya secara bersamaan, beberapa *node* lain mungkin akan menerima satu atau yang lain pertama kali. Dalam hal ini, mereka akan mengerjakan block pertama yang telah diterima, namun menyimpan yang lain di cabang lain jika nantinya akan diperpanjang. Ikatan cabang itu akan putus jika telah ditemukan proof-of-work dan hanya satu cabang yang akan diperpanjang, *node* yang mengerjakan di cabang berbeda selanjutnya akan pindah ke cabang yang terpanjang tersebut.

Broadcast transaksi baru tidak perlu harus mencapai seluruh node. Selama broadcast tersebut sudah menjangkau cukup banyak node, mereka akan memasukkan ke dalam sebuah block tidak lama kemudian. Broadcast block juga akan cukup bertoleransi terhadap terputusnya pesan. Jika satu *node* belum menerima sebuah block, dia bisa meminta itu bersamaan dengan block baru berikutnya, dan menyadari jika dia telah melewatkan satu block tersebut.

6. Insentif

Berdasarkan ketentuan, transaksi pertama pada sebuah block baru merupakan sebuah transaksi spesial untuk memulai sebuah koin baru yang dimiliki oleh pencipta block tersebut. Penambahan ini merupakan sebuah insentif untuk *node* karena telah berpartisipasi di dalam jaringan, dan membuka jalan untuk dapat mendistribusikan koin ke dalam sirkulasi, karena tidak ada otoritas terpusat yang bertindak untuk menerbitkan koin itu. Penambahan sejumlah koin secara konstan dianalogikan seperti halnya para penambang emas yang telah mengerahkan sumber dayanya untuk menambah emas ke dalam sirkulasi. Dalam kasus ini, yang dikerahkan adalah berupa CPU time dan daya energi listrik.

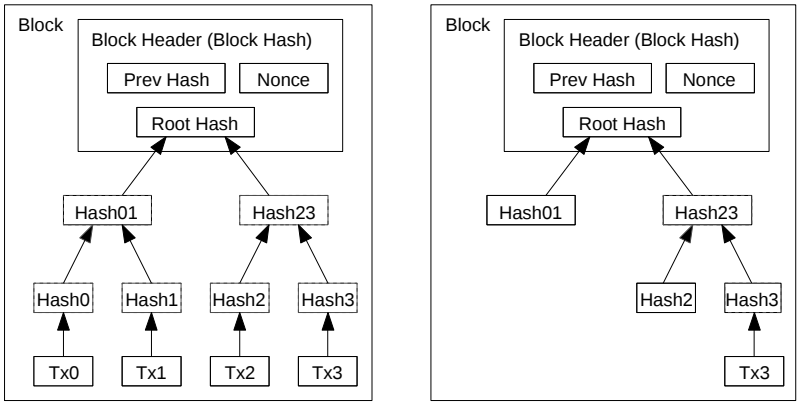
Pembiayaan untuk insentif ini juga dapat dilakukan dari biaya transaksi. Jika nilai output sebuah transaksi lebih sedikit daripada nilai input, selisih nilai itu adalah biaya transaksi yang dapat ditambahkan sebagai nilai insentif dari block yang didalamnya berisi transaksi. Setelah sejumlah koin yang telah ditentukan sebelumnya masuk di dalam sirkulasi, insentif tersebut dapat sepenuhnya digantikan dengan biaya transaksi dan sepenuhnya bebas inflasi.

Insentiff dapat mendorong *node* untuk terus berlaku jujur. Jika seorang penyerang yang rakus mampu menghimpun daya CPU lebih besar daripada seluruh node jujur, dia harus memilih

menggunakannya untuk menipu orang dengan mencuri kembali pembayaran transaksinya, atau menggunakannya untuk memproduksi koin baru. Dia sepatutnya dapat memahami bahwa lebih menguntungkan untuk tetap mengikuti aturan yang ada, aturan semacam itu akan menguntungkan dirinya dengan memperoleh lebih banyak koin dari kebanyakan orang lain jika digabungkan, daripada untuk merusak sistem dan keabsahan untuk kekayaannya sendiri.

7. Reclaiming Ruang Penyimpanan

Setelah transaksi terbaru pada sebuah koin terkubur dibawah sejumlah block yang cukup, transaksi yang telah dipergunakan sebelumnya dapat diabaikan untuk menghemat ruang peyimpanan. Untuk memfasilitasi hal ini dapat dilakukan tanpa harus merusak hash block transaksi yang dihashing dalam sebuah Merkle Tree [7][2][5], dengan hanya menyertakan *root* saja dalam hash block. Block lama dapat diringkas dengan cara memotong cabang pohon tersebut. Bagian-bagian dalam hash tidak perlu untuk disimpan.



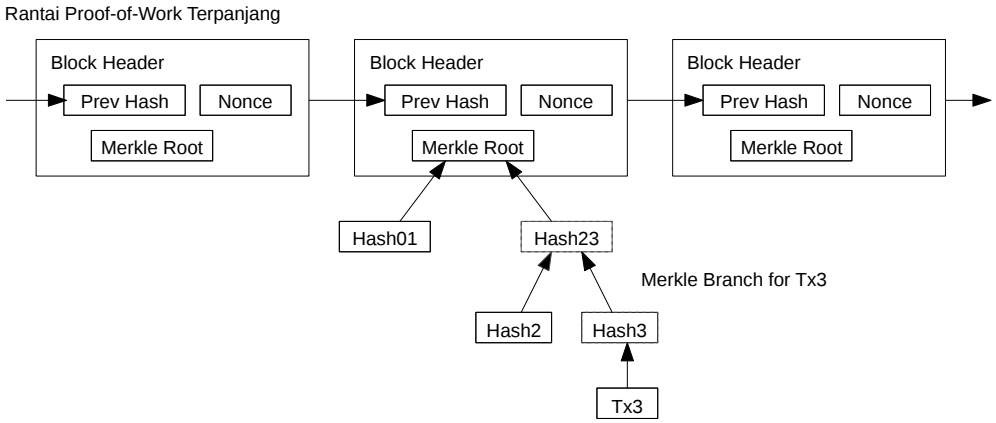
Transaksi yang di Hashed dalam Merkle Tree

Setelah dipangkas Tx0-2 dari Block

Block header tanpa transaksi di dalamnya berkisar sekitar 80 byte. Jika kita menghendaki bahwa block akan dihasilkan setiap 10 menit, 80 byte * 6 * 24 * 365 = 4,2MB per tahun. Dengan sistem komputer yang umumnya dijual dengan RAM 2GB di tahun 2008, prediksi berdasarkan hukum Moore pertumbuhannya 1,2GB per tahun, ruang penyimpanan tidak akan menjadi masalah bahkan jika header block tetap harus di simpan di memori.

8. Penyederhanaan Verifikasi Pembayaran

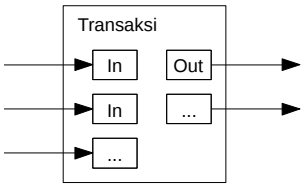
Cukup memungkinkan untuk memverifikasi pembayaran tanpa harus menjalankan *full node* di jaringan. User cukup hanya menyimpan salinan block header dari rantai proof-of-work terpanjang, yang dapatt diperoleh melalui *query* node di dalam jaringan sampai dia yakin bahwa dia mendapatkan rantai terpanjang, dan mengetahui bahwa cabang Merkle yang berelasi dengan transaksi ke block tersebut telah dibubuhi dengan stempel waktu. Dia tidak dapat memeriksa transaksi untuk dirinya sendiri, namun dengan mentautkannya di sebuah tempat di dalam rantai, dia dapat melihat node di dalam jaringan telah menerimanya, dan block akan ditambahkan kemudian setelah jaringan mengkonfirmasi telah menerimanya.



Dengan demikian, verifikasi akan dapat diandalkan selama *node* jujur mengontrol jaringan, namun akan lebih rentan jika jaringan berhasil dikuasai oleh penyerang. Ketika *node* di jaringan dapat memverifikasi transaksinya sendiri, metode sederhana ini dapat dengan mudah ditipu oleh transaksi dari penyerang selama dia masih menguasai jaringan. Salah satu strategi untuk melindugi dari hal ini adalah dengan menerima peringatan dari *node* di jaringan saat mereka mendeteksi adanya block yang tidak valid, menuntun piranti lunak pengguna untuk mengunduh seluruh block dan mengkonfirmasi adanya ketidakkonsistenan pada transaksi tersebut. Bisnis yang kerap menerima pembayaran mungkin dapat menjalankan node nya sendiri agar lebih independen dengan verifikasi yang lebih cepat.

9. Penggabungan dan Pemisahan Nilai

Meski memungkinkan untuk menangani koin secara individual, akan cukup berat untuk memisahkan setiap sen transaksi dalam sebuah transfer. Untuk bisa memisah dan menggabungkan nilai, transaksi terdiri dari beberapa input dan output. Normalnya akan ada satu input dari transaksi besar sebelumnya atau beberapa input yang berasal dari gabungan beberapa jumlah kecil, dan paling banyak terdiri dari dua output: satu untuk pembayaran, satu lagi sebagai kembalian, jika ada, akan dikembalikan pada pengirim.

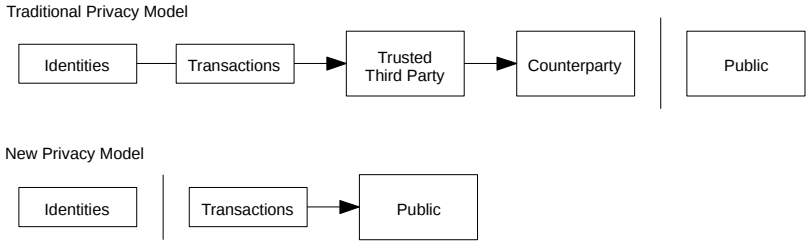


Perlu dicatat bahwa *fan-out*, ketika sebuah transaksi bergantung pada beberapa transaksi, dan transaksi tersebut bergantung juga pada beberapa transaksi lain, bukanlah menjadi persoalan disini. Tidak pernah dibutuhkan untuk ekstrak salinan penuh kesejarahan transaksi secara mandiri.

10. Privasi

Model perbankan tradisional meraih level privasinya dengan membatasi akses informasi untuk pihak-pihak yang terlibat dengan pihak ketiga yang dipercaya. Kebutuhan untuk mengumumkan transaksi secara publik menjadi terhambat pada metode ini, namun privasi tetap dapat dipertahankan dengan memutus aliran informasi di tempat lain: dengan cara menjaga *public key* tetap anonim. *Public key* dapat melihat seseorang sedang mengirim sebagian jumlah kepada orang

lain, namun tanpa mentautkan informasi transaksi tersebut pada siapapun. Hal ini mirip dengan level informasi yang dirilis oleh bursa saham, dimana waktu dan ukuran perdagangan secara individual, “tape”, dibuat secara publik, namun tanpa memberitahukan siapa saja pihak didalamnya.



Sebagai firewall tambahan, sepasang *key* baru harus digunakan untuk setiap transaksi agar transaksi itu tidak ditautkan pada pemiliknya secara umum. Beberapa tautan itu masih tidak dapat dihindarkan dengan transaksi yang memiliki beberapa input, karena terpaksa harus mengungkap input yang dimiliki oleh pemilik yang sama tersebut. Resikonya adalah jika *key* pemilik terungkap, tautan itu juga akan membuka transaksi lain yang dimiliki oleh pemilik yang sama.

11. Kalkulasi

Kami mempertimbangkan skenario seorang penyerang yang berupaya untuk generate rantai alternatif yang lebih cepat dibandingkan ranti yang jujur. Bahkan jika ini berhasil, itu tidak akan membuat sistem secara terbuka untuk dapat merubah sekehendaknya, seperti membuat nilai dari ketiadaan atau mengambil uang yang tidak pernah dimiliki oleh penyerang, dan *node* jujur tidak akan pernah menerima block mereka. Seorang penyerang hanya akan dapat merubah satu transaksinya sendiri agar bisa mengambil uang pengeluarannya kembali.

Perlombaan antara rantai jujur dengan rantai penyerang dapat dikategorikan sebagai sebuah *Binomial Random Walk*. Event yang berhasil adalah rantai jujur akan memperpanjang satu block, berhasil memimpin dengan +1, dan event yang gagal adalah rantai penyerang setelah memperpanjang satu block, namun kemudian dikurangi dari adanya gap -1.

Probabilitas penyerang yang berupaya mengejar defisit itu dianalogikan seperti sebuah *Gambler's Ruin problem*. Misalkan seorang penjudi dengan nilai kredit yang tidak terbatas memulai di posisi defisit ketertinggalan dan memainkan percobaan dengan jumlah yang tidak terbatas agar mampu memperoleh titik impas, atau penyerang pernah menyamai rantai yang jujur, sebagai berikut [8]:

p = probabilitas node jujur mendapatkan blok berikutnya
q = probabilitas penyerang mendapatkan blok berikutnya
q_z = probabilitas penyerang akan menyusul dari posisi z blok sebelumnya

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

Dengan asumsi kami bahwa $p > q$, probabilitas turun secara eksponensial karena jumlah block yang harus dikejar penyerang juga akan bertambah. Dengan kondisi yang tidak menguntungkan untuk penyerang, jika dia tidak dapat membuat keberuntuknган besar di tahap awal, maka peluangnya menjadi semakin kecil ketika dia makin jauh tertinggal.

Sekarang kami mempertimbangkan berapa lama waktu yang dibutuhkan penerima transaksi untuk menunggu sebuah transaksi baru sebelum meyakinkan dirinya bahwa pengirim tidak akan dapat mengubah transaksi tersebut. Kami beranggapan bahwa pengirim adalah seorang penyerang yang ingin membuat penerima mempercayai bahwa dia telah melakukan pembayaran untuk sementara waktu, lalu mengubahnya untuk membayar kembali untuk dirinya sendiri setelah beberapa saat kemudian. Penerima akan diperingatkan ketika hal itu terjadi, namun pengirim berharap agar peringatan itu datang terlambat.

Penerima membuat *key pair* baru dan memberikan *public key* tersebut pada pengirim tepat sebelum menandatanganinya. Hal ini mencegah pengirim untuk mampu menyiapkan rantai block didepannya agar bisa punya waktu lebih untuk mengerjakan itu secara terus-menerus sampai dia beruntung dan bisa mencapai lebih jauh ke depan, lalu mengeksekusi transaksi tersebut. Setelah transaksi terkirim, pengirim yang tidak jujur itu mulai bekerja secara rahasia pada sebuah rantai paralel yang mengandung versi alternatif transaksinya.

Sedangkan penerima masih harus menunggu sampai transaksi itu berhasil ditambahkan pada sebuah block dan *z* block berhasil ditautkan setelahnya. Penerima itu tidak mengetahui jumlah pasti dari perkembangan yang dilakukan penyerang, namun asumsi block jujur mengambil rata-rata waktu yang diharapkan adalah per block, perkembangan potensial penyerang akan menjadi *Poisson distribution* dengan nilai yang diharapkan:

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

Untuk mendapat probabilitas penyerang saat masih berupaya mengejar saat ini, kita mengalikan densitas *Poisson* di setiap progress yang telah dicapai dengan probabilitas yang memungkinkan dikejar penyerang pada satu titik:

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

Penataan ulang untuk menghindari penjumlahan yang tak terhingga pada bagian akhir distribusi…

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

Konversi menjadi code 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;
}
```

Menjalankan pada beberapa hasil, kita dapat melihat penurunan probabilitas secara exponensial dengan *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
```

Penyelesaian untuk *P* kurang dari 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. Kesimpulan

Kita telah menawarkan sebuah sistem untuk transaksi elektronik tanpa harus bergantung pada pihak yang bisa dipercaya. Kami mulai dengan kerangkan koin biasa yang dibuat dari *digital signature*, yang memberikan kontrol kepemilikan penuh, namun tidak lengkap tanpa sebuah cara yang mampu mencegah *double spending*. Untuk mengatasi hal ini, kami menawarkan jaringan *peer-to-peer* menggunakan *proof-of-work* untuk merekam kesejarahan transaksi publik yang cukup cepat menjadi tidak praktis bagi penyerang secara komputasi untuk merubahnya jika *node* jujur mengkontrol sebagian besar daya CPU. Jaringannya kuat dengan kesederhanaan yang tidak terstruktur. Seluruh *node* bekerja serentak dengan sedikit koordinasi. Mereka tidak perlu teridentifikasi, karena pesan tidak disalurkan kepada tempat tertentu dan dikirimkan pada sebuah *best effort basis*. *Node* dapat pergi dan datang kembali sesukanya, menerima rantai *proof-of-work* sebagai bukti atas apa yang terjadi saat mereka meninggalkan jaringan. Mereka memberikan voting melalui daya CPU mereka, mewakili kesepakatan untuk menerima block yang valid dengan mengolah dan memperpanjang block tersebut serta menolak block yang tidak valid dengan tidak mengolahnya. Setiap aturan dan insentif yang diperlukan dapat diselenggarakan melalui mekanisme konsensus ini.

Referensi

[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 trust requirements," In *20th Symposium on Information Theory in the Benelux*, May 1999.

[3] S. Haber, W.S. Stornetta, "How to time-stamp a digital document," In *Journal of Cryptology*, vol 3, no 2, pages 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*, pages 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*, pages 28-35, April 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, pages 122-133, April 1980.

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