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
by Christopher Tahir, Gregorius Airlangga & K Hendrawan

Abstrak. Sebuah sistem pembayaran elektronik versi *peer-to-peer* murni yang membuat pembayaran secara daring dapat terjadi langsung dari satu pihak ke pihak lainnya tanpa melalui sebuah lembaga keuangan. Tanda tangan digital menjadi salah satu bagian dari solusi tersebut, namun manfaat utamanya akan hilang jika masih membutuhkan pihak ketiga untuk menghindari terjadinya *double-spending*. Kami mengajukan solusi terhadap permasalahan pembelanjaan ganda ini dengan menggunakan jaringan *peer-to-peer*. Jaringan ini akan membuat catatan-waktu (*timestamp*) dari transaksi-transaksi dengan cara melakukan hashing yang akan dimasukkan ke dalam rantai hash-based *proof-of-work*, membentuk catatan yang tidak dapat diubah tanpa mengulang kegiatan *proof-of-work* tersebut. Rantai terpanjang bukan hanya menjadi bukti dari serangkaian kejadian yang disaksikan, namun merupakan bukti bahwa rangkaian tersebut muncul dari kumpulan (*pool*) CPU power (tenaga komputasi) terbesar. Selama mayoritas dari tenaga komputasi ini dikendalikan oleh titik komputasi (*node*) yang tidak bermaksud menyerang jaringan ini, mereka akan menciptakan rantai terpanjang sekaligus menghilangkan kesempatan bagi penyerang. Jaringan ini sendiri membutuhkan struktur yang sederhana. Pesan disiarkan berdasarkan usaha terbaik (*best effort basis*), dan titik-titik dapat dengan bebas berpisah dan bergabung kembali dengan jaringan, menerima rantai *proof-of-work* terpanjang sebagai bukti dari kejadian selama mereka tidak aktif pada jaringan.

1. Pendahuluan

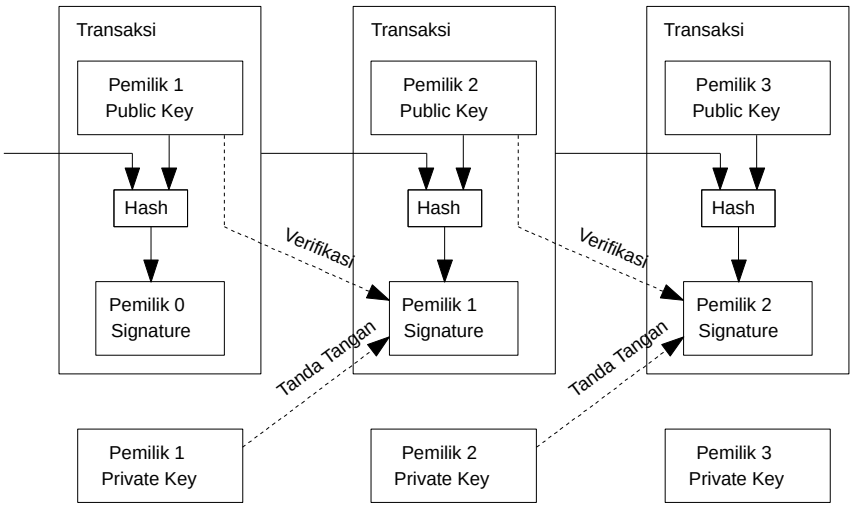
Perdagangan di internet sampai saat ini masih sangat tergantung secara eksklusif pada badan keuangan yang berperan sebagai pihak ketiga yang dipercaya untuk memproses pembayaran elektronis. Walaupun sistem ini berjalan dengan cukup baik untuk hampir semua transaksi, sistem ini masih mempunyai kelemahan bawaan dari sistem berbasis kepercayaan. Transaksi yang bersifat reversibel penuh tidak dimungkinkan karena badan keuangan tersebut tidak dapat menghindari tindakan mediasi perselisihan. Biaya mediasi tersebut mengakibatkan peningkatan biaya transaksi sehingga menyebabkan adanya batas minimum transaksi dan menghilangkan kemungkinan transaksi kecil sederhana sehari-hari, dan karena itu ada juga kemungkinan munculnya biaya dari ketidaktersediaan mengadakan transaksi non-reversibel untuk layanan yang non-reversibel. Dengan kemungkinan untuk melakukan transaksi reversibel, kebutuhan atas kepercayaan meningkat. Para pedagang harus waspada terhadap pelanggan nya, mengorek informasi dari pelanggan nya sampai ke tingkat yang mereka tidak bisa atau butuhkan. Sekian persen dari penipuan bisa dianggap sebagai tidak terhindarkan. Biaya dan ketidakpastian pembayaran ini sebetulnya bisa dihindari secara personal dengan menggunakan mata uang fisik, tapi belum ada mekanisme yang memungkinkan terjadinya pembayaran melalui jalur komunikasi tanpa ada pihak terpercaya.

Yang dibutuhkan adalah sistem pembayaran elektronik berbasis bukti kriptografi untuk menggantikan sistem kepercayaan, memungkinkan dua pihak dengan sukarela melakukan transaksi secara langsung tanpa memerlukan pihak ketiga sebagai saksi. Transaksi yang secara

komputasi tidak mungkin untuk dibalikkan bisa melindungi penjual dari tindakan penipuan dan mekanisme *escrow* seperti biasa akan bisa dengan mudah melindungi pembeli. Dalam tulisan ini, kami mengajukan solusi atas masalah *double-spending* dengan menggunakan server *timestamp* yang terdistribusi secara *peer-to-peer* yang digunakan untuk membuat bukti urutan kronologis transaksi. Sistem ini akan aman selama lebih banyak titik komputasi yang 'jujur' secara kolektif mengendalikan kekuatan CPU dibandingkan dengan kelompok titik komputasi penyerang.

2. Transaksi

Kami mendefinisikan satu koin elektronik sebagai satu rangkaian tandatangan digital. Setiap pemilik dari koin mentransfer ke pemilik selanjutnya dengan membubuhkan tandatangan digital pada 'hash' dari transaksi sebelumnya dan public key dari pemilik selanjutnya pada ujung akhir dari koin. Penerima pembayaran dapat memverifikasi tandatangan untuk memastikan rantai kepemilikan.

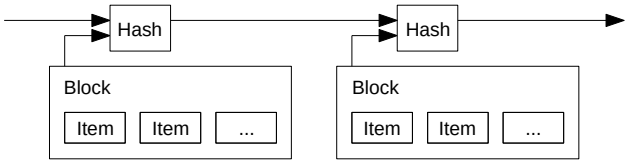


Masalah utamanya, penerima pembayaran tidak bisa memastikan bahwa salah satu pemilik tidak melakukan *double-spending* pada koin nya. Solusi umum atas masalah ini adalah dengan melibatkan otoritas pusat atau pencetak koin yang memeriksa setiap transaksi dari kemungkinan *double-spending*. Setiap melakukan transaksi, koin harus dikembalikan ke pencetak koin untuk kemudian dikeluarkan koin baru, dan hanya koin yang dikeluarkan langsung oleh pencetak koin yang bisa dipercaya tidak melakukan *double-spending*. Masalah dengan solusi ini adalah nasib dari seluruh sistem keuangan bergantung pada perusahaan percetakan koin, dengan keharusan untuk melewatkan setiap transaksi kepada mereka seperti halnya sebuah bank.

Kita membutuhkan suatu cara dimana penerima pembayaran bisa mengetahui bahwa pemilik-pemilik sebelumnya tidak menandatangani transaksi sebelumnya. Untuk kepentingan ini, transaksi yang paling awal adalah yang berlaku, jadi kita tidak perlu memperdulikan usaha untuk melakukan *double-spending* di belakangnya. Satu-satunya cara untuk memastikan ketiadaan suatu transaksi adalah dengan mengawasi transaksi secara keseluruhan. Dalam model percetakan koin, percetakan mengetahui semua transaksi dan memutuskan mana yang terjadi lebih dahulu. Untuk melakukan tanpa pihak terpercaya, transaksi harus diumumkan secara publik, dan kita membutuhkan sebuah sistem dimana partisipan dapat menyetujui suatu urutan sesuai dengan saat diterimanya. Penerima pembayaran membutuhkan bukti pada saat terjadinya setiap transaksi, mayoritas titik-titik komputasi tersebut menyetujui bahwa transaksi tersebut diterima pertama kali.

3. Timestamp Server

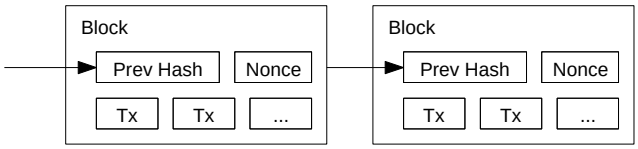
Solusi yang kita ajukan dimulai dengan *timestamp* server. *Timestamp* server bekerja dengan cara mengambil hash dari blok setiap item untuk di-*timestamp* dan mempublikasikan secara luas hash tersebut, sama seperti pada newspaper atau usenet post [2-5]. *Timestamp* membuktikan bahwa data ada pada saat itu, secara jelas, agar bisa dimasukkan ke dalam hash. Setiap *timestamp* melingkupi *timestamp* sebelumnya dalam hash, membentuk suatu rantai, dengan setiap *timestamp* tambahan memperkuat *timestamp* sebelumnya.



4. Proof-of-Work

Untuk bisa mengimplementasikan *timestamp* server terdistribusi secara *peer-to-peer*, kita perlu menggunakan sistem *proof-of-work* yang sama dengan Adam Back's Hashcash [6], daripada newspaper atau Usenet posts. *Proof-of-work* melibatkan proses scanning untuk nilai yang jika melalui proses hashing, contohnya dengan menggunakan SHA-256, hash akan dimulai dengan sejumlah bit nol. Jumlah rata-rata usaha yang diperlukan berbanding eksponensial dengan jumlah bit nol yang diperlukan dan bisa diverifikasi dengan mengeksekusi sebuah hash.

Untuk jaringan *timestamp* kita, *proof-of-work* diimplemetasikan dengan deret inkremental nonce dalam blok sampai didapat suatu nilai yang dapat memberikan hash dari suatu blok banyaknya zero bit yang diperlukan. Setelah usaha CPU dikeluarkan untuk memenuhi kebutuhan *proof-of-work*, blok tersebut tidak dapat dirubah tanpa mengulang proses. Jika kemudian blok-blok disambungkan di belakangnya, usaha untuk mengubah blok tersebut mengharuskan untuk mengulang pembuatan blok-blok dibelakangnya juga.



Proof-of-work juga mengatasi masalah dalam menentukan pembuat keputusan mayoritas. Jika keputusan mayoritas dibuat berdasarkan kepada satu IP address untuk satu suara (one-IP-address-one-vote), ini dapat ditumbangkan oleh siapapun yang dapat mengalokasikan banyak IP. *Proof-of-work* pada dasarnya adalah satu CPU satu suara (one-CPU-one-vote). Keputusan mayoritas ditentukan oleh rantai terpanjang, yang mempunyai usaha *proof-of-work* terbanyak. Jika mayoritas kekuatan CPU dikendalikan oleh titik-titik yang jujur, rantai kejujuran dapat tumbuh lebih cepat dan mengalahkan rantai kompetitor. Untuk mengubah blok yang sudah ada, penyerang perlu membuat ulang *proof-of-work* blok tersebut dan semua blok sesudahnya dan kemudian mengejar dan menyusul kerja titik-titik komputasi yang jujur. Kita akan perlihatkan nanti kemungkinan penyerang yang lebih lambat yang berusaha untuk mengejar menghilang secara eksponensial ketika blok-blok berikutnya ditambahkan.

Untuk mengkompensasi peningkatan kecepatan perangkat keras dan minat menjalankan titik komputasi yang berubah-ubah seiring perkembangan waktu, tingkat kesulitan dari *proof-of-work* ditentukan oleh target rata-rata jumlah blok per jam yang selalu berubah-ubah juga. Jika mereka dibuat terlalu cepat, tingkat kesulitan akan meningkat.

5. Jaringan

Langkah untuk menjalankan jaringan adalah sebagai berikut:

1) Transaksi-transaksi baru disiarkan ke semua titik-titik komputasi.
2) Setiap titik menggabungkan transaksi-transaksi baru ke dalam blok.
3) Setiap titik berupaya untuk menemukan *proof-of-work* yang sulit untuk blok nya.
4) Ketika sebuat titik menemukan *proof-of-work*, titik tersebut menyiarkan blok tersebut ke semua titik-titik.
5) Titik menerima blok tersebut hanya jika semua transaksi di dalam nya valid dan belum pernah digunakan.
6) Titik-titik menyatakan bahwa blok diterima dengan bekerja menambahkan blok berikutnya pada rantai, dengan hash dari blok yang diterima sebegai hash sebelumnya.

Titik-titik komputasi selalu menganggap rantai terpanjang adalah rantai yang benar dan akan terus bekerja memperpanjang rantai tersebut. Jika dua titik menyiarkan versi berbeda dari blok berikutnya secara bersamaan, beberapa titik-titik akan menerima satu atau lainnya terlebih dahulu. Pada kejadian seperti itu, titik tersebut akan mengerjakan blok yang pertama diterima, tapi tetap menyimpan cabang yang satunya lagi barangkali rantai itu bertambah panjang. Posisi seri ini akan dipecahkan ketika *proof-of-work* berikutnya ditemukan dan satu cabang bertambah panjang; titik-titik yang bekerja pada cabang yang lain akan berpindah pada cabang yang lebih panjang.

Penyiaran transaksi baru tidak harus mencapai keseluruhan titik. Selama bisa mencapai banyak titik, mereka akan segera tergabung dalam suatu blok. Penyiaran blok juga mempunyai toleransi atas pesan yang terputus (*dropped message*). Jika satu titik tidak menerima sebuah blok, maka titik tersebut akan meminta pada saaat menerima blok berikutnya dan mengetahui bahwa sebuah blok telah terlewatkan.

6. Insentif

Berdasarkan kesepakatan, transaksi pertama pada suatu blok adalah transaksi khusus yang memulai koin baru yang dimiliki oleh kreator dari blok tersebut. Ini menjadi insentif bagi titik-titik untuk mendukung jaringannya, dan menyediakan jalan untuk mulai mendistribusikan koin ke dalam sirkulasi, karena tidak ada otoritas pusat yang mengeluarkannya. Penambahan jumlah koin secara konstan bisa dianalogikan dengan penambang emas yang mengerahkan sumber daya untuk menambahkan emas pada sirkulasi. Dalam hal ini, sumber daya CPU time dan daya listrik yang dikerahkan.

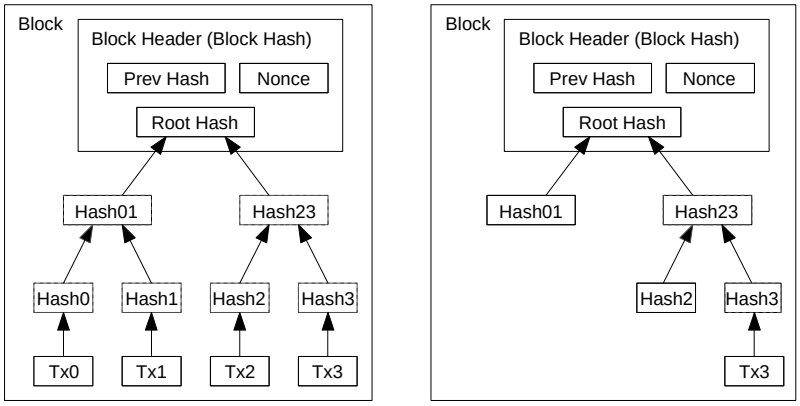
Insentif dapat juga dibiayai dari biaya transaksi. Jika nilai keluaran dari satu transaksi kurang dari nilai masukan, selisih nya adalah biaya transaksi yang ditambahkan pada besaran insentif dari blok yang mengandung transaksi tersebut. Setelah sejumlah koin yang sudah ditentukan memasuki sirkulasi, insentif dapat beralih secara keseluruhan menjadi biaya transaksi dan sama sekali menjadi bebas inflasi.

Insentif dapat membantu menjaga titik-titik agar tetap jujur. Jika ada penyerang yang serakah berhasil membangun kekuatan CPU lebih tinggi daripada semua titik jujur, dia hanya akan bisa memilih menggunakannya untuk menipu orang dengan merebut kembali pembayarannya, atau untuk menbuat koin baru. Dia seharusnya akan menyadari bahwa mengikuti aturan akan lebih menguntungkan, seperti aturan yang akan menguntungkannya dengan koin baru lebih banyak daripada gabungan beberapa orang, daripada merusak sistem dan keabsahan dari kekayaannya sendiri.

7. Memulihkan Ruang Penyimpanan

Setelah transaksi terakhir dalam sebuah koin terbenam di bawah blok yang cukup, transaksi yang dilakukan sebelumnya dapat diabaikan untuk menghemat ruang penyimpanan. Untuk memfasilitasi ini tanpa memutuskan hash blok, transaksi di hash kan dalam Merkle Tree [7][2] [5], dengan hanya root yang disertakan dalam hash blok. Blok lama dapat diringkaskan dengan

mematikan cabang dari pohon tersebut. Hash-hash yang berada di bagian dalam tidak perlu disimpan lagi.



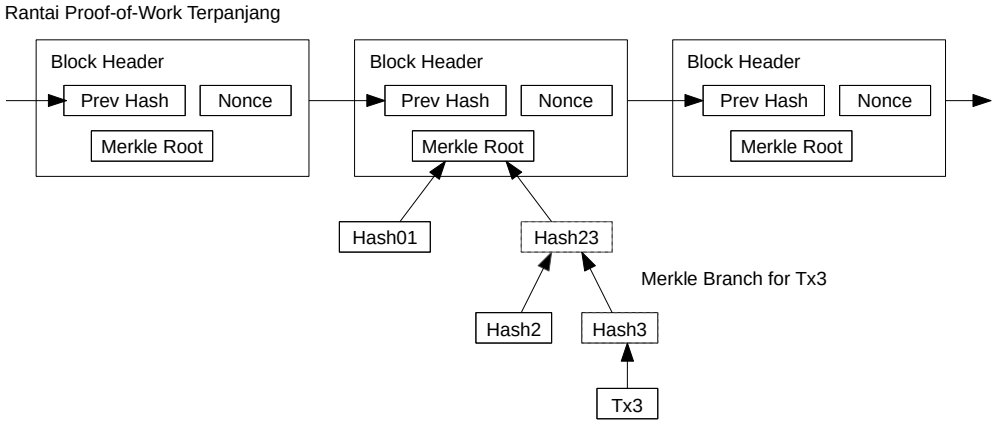
Transaksi yang di Hashed dalam Merkle Tree

Setelah dipangkas Tx0-2 dari Block

Sebuah kepala blok (block header) tanpa transaksi kurang lebih akan berukuran 80 bytes. Jika kita beranggapan bahwa blok dihasilkan setiap 10 menit maka, 80 bytes * 6 * 24 * 365 = 4.2MB per tahun. Dengan sistem komputer yang umumnya dijual dengan 2GB RAM pada tahun 2008, dan hukum Moore yang memprediksi pertumbuhan 1.2GB per tahun, ruang penyimpanan seharusnya tidak akan menjadi masalah bahkan jika kepala blok harus disimpan dalam memory.

8. Verifikasi Pembayaran yang Disederhanakan

Verifikasi pembayaran memungkinkan untuk dilakukan tanpa perlu menjalankan jaringan titik-titik komputasi secara penuh. Seorang pengguna hanya perlu menyimpan salinan dari kepala blok dari rantai terpanjang *proof-of-work*, yang mana bisa didapat dengan melakukan kueri atas titik-titik jaringan sampai dia teryakinkan bahwa rantai yang diperolehnya adalah rantai yang terpanjang, dan mendapatkan cabang Merkle Tree yang menghubungkan transaksi tersebut kepada blok dengan *timestamp* yang diperuntukkannya. Dia tidak dapat memeriksa transaksi tersebut untuk dirinya sendiri, tapi dengan menghubungkannya pada suatu tempat pada rantai, dia dapat melihat apakah sebuat titik jaringan telah menerimanya, dan blok-blok telah ditambahkan setelahnya mengkonfirmasi lebih lanjut bahwa jaringan telah menerima transaksi tersebut.

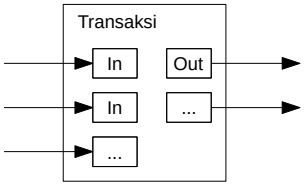


Dengan demikian, verifikasi bisa diandalkan selama titik-titik yang jujur mengendalikan jaringan, tapi akan lebih rentan jika jaringan dikuasai oleh penyerang. Ketika titik-titik jaringan dapat memverifikasi transaksi untuk mereka sendiri, metoda yang disederhanakan dapat dikelabui

oleh transaksi palsu si penyerang selama penyerang bisa menguasai jaringan. Satu strategi untuk mencegah kejadian ini adalah dengan menerima peringatan dari titik-titik jaringan ketika mereka mendeteksi blok yang tidak valid, memberitahu software sang pengguna untuk mengunduh blok secara penuh dan meminta transaksi untuk mengkonfirmasi ketidak-konsistensiannya. Bisnis yang sering menerima pembayaran kemungkinan akan tetap menginginkan untuk menjalankan titik-titik nya sendiri untuk kemanan yang lebih independen dan verifikasi yang lebih cepat.

9. Menggabungkan dan memisahkan nilai

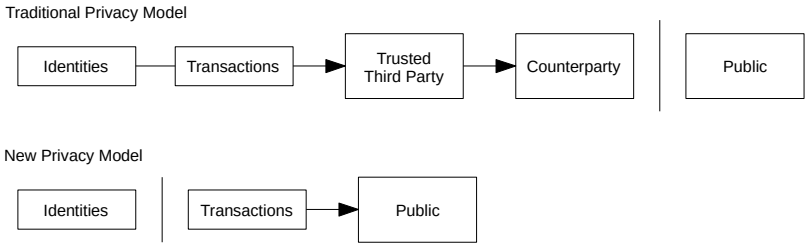
Walaupun memungkinkan untuk menangani setiap koin secara individual, akan sangat berat untuk membuat transaksi terpisah bagi setiap sen dalam suatu transfer. Untuk memungkinkan suatu nilai bisa di pisahkan dan digabungkan, transaksi mempunyai beberapa masukan dan keluaran. Biasanya akan ada masukan tunggal yang berasal dari transaksi lebih besar sebelumnya atau beberapa masukan yang menggabungkan jumlah yang lebih kecil, dan paling tidak ada dua keluaran: satu untuk pembayaran dan satu lagi untuk kembalian, kalau ada, ke pihak pengirim.



Perlu dicatat bahwa dalam model *fan-out*, dimana sebuah transaksi bergantung pada beberapa transaksi-transaksi, dan transaksi-transaksi tersebut bergantung pada lebih banyak lagi transaksi, tidak menjadi masalah di sini. Tidak pernah diperlukan untuk mengekstrak sebuah salinan sejarah transaksi yang benar-benar berdiri sendiri.

10. Privasi

Model perbankan tradisional menerapkan suatu tingkatan privasi dengan membatasi akses informasi kepada pihak-pihak yang terkait dan pihak ketiga yang dipercaya. Kebutuhan untuk mengumumkan semua transaksi secara publik menghalangi metoda ini, tapi privasi tetap bisa diperoleh dengan memutuskan aliran informasi di tempat lain: dengan menjaga anonimitas dari public key. Publik bisa menyaksikan bahwa seseorang mengirimkan sejumlah kepada seseorang lainnya, tapi tanpa informasi yang menghubungkan transaksi kepada siapapun. Ini sama dengan tingkat informasi yang disiarkan oleh bursa efek, dimana waktu dan ukuran dari perdagangan individual, the "tape", dibuka secara publik, tapi tanpa memberitahukan siapa saja yang terkait.



Sebagai pengaman tambahan, sepasang kunci baru harus digunakan pada setiap transaksi untuk menghindari dihubungkannya dengan pemilik biasa. Beberapa penghubungan masih tidak dapat dihindari untuk transaksi multi-input, yang secara terpaksa menampilkan bahwa masukan mereka dimiliki oleh pemilik yang sama. Resiko bahwa pemilik dari suatu kunci bisa diketahui, penghubungan bisa menampakkan transaksi-transaksi lainnya yang dimiliki oleh pemilik yang sama.

11. Perhitungan

Kami meninjau suatu skenario dimana penyerang berusaha menciptakan rantai alternatif yang lebih cepat daripada rantai yang jujur. Walaupun ini berhasil dilakukan, tidak akan menyebabkan sistem menjadi terbuka atas perubahan yang sewenang-wenang, sebagai contoh membuat nilai secara sekonyong-konyong atau mengambil uang yang bukan merupakan hak sang penyerang. Titik-titik tidak akan menerima transaksi yang tidak sah sebagai pembayaran, dan titik-titik jujur akan menerima blok yang mengandung nya. Penyerang hanya akan bisa mencoba merubah salah satu transaksi miliknya untuk mengambil kembali uang yang sudah dia belanjakan.

Adu cepat antara rantai yang jujur dan rantai penyerang bisa dikarakteristikkan sebagai Binomial Random Walk. Keberhasilan terjadi ketika rantai jujur diperpanjang sebanyak satu blok, menambahkan keunggulan nya sebanyak + 1, dan kegagalan adalah ketika rantai penyerang diperpanjang sebanyak satu blok, mengurangi jarak sebanyak -1.

Kemungkinan dari seorang penyerang mengejar defisit ini sama dengan permasalahan Gambler's Ruin. Anggap seorang penjudi dengan kredit yang tidak terbatas memulai dari defisit dan mencoba bermain sebanyak tidak terhingga sampai mencapai titik impas. Kita dapat menghitung probabilitas dia mencapai titik impas, atau saat si penyerang dapat mengejar rantai yang jujur dengan perhitungan sebagai berikut [8]:

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

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

Dengan asumsi kita bahwa p > q, nilai kemungkinan akan turun secara eksponensial seiring dengan jumlah blok yang harus dikejar oleh penyerang meningkat. Dengan kondisi yang tidak menguntungkan ini, jika si penyerang tidak mendapatkan keberuntungan dengan melakukan sergapan lebih awal, maka kesempatannya akan menjadi sangat kecil karena dia akan tertinggal jauh di belakang.

Sekarang kita pertimbangkan berapa lama penerima suatu transaksi baru perlu menunggu sebelum cukup mendapatkan kepastian bahwa pengirim sudah tidak bisa melakukan perubahan pada transaksi. Kita beranggapan bahwa pengirim adalah seorang penyerang yang bermaksud untuk mengelabui penerima bahwa dia sudah melakukan pembayaran, kemudian mengembalikan pembayaran kepada dirinya sendiri setelah beberapa saat. Penerima akan mendapat peringatan atas kejadian ini, namun pengirim berharap bahwa saat itu sudah terlambat.

Penerima akan menghasikan sepasang kunci baru dan mengirimkan public key kepada pengirim sesaat sebelum menandatanganinya. Ini mencegah pengirim untuk mempersiapkan sebuah rantai blok sebelumnya dengan mengolah secara terus-menerus sampai cukup beruntung mendapatkan posisi cukup jauh di depan, kemudian mengeksekusi transaksi pada saat itu. Segera setelah sebuat transaksi terkirim, sang pengirim yang tidak jujur mulai membuat versi lain dari transaksi tersebut secara paralel dan tersembunyi.

Penerima menunggu sampai transaksi berhasil ditambahkan pada sebuah blok dan z blok telah tertautkan sesudahnya. Dia tidak mengetahui secara pasti perkembangan dari aksi sang penyerang, tapi dengan asumsi titik-titik jujur memerlukan jumlah waktu rata-rata yang diperkirakan per blok, tingkat perkembangan potensial dari sang penyerang akan dalam Poisson distribution dengan nilai yang telah diperkirakan:

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

Untuk mendapatkan kemungkinan sang penyerang masih bisa mengejar saat ini, kita mengalikan Poisson density untuk setiap jumlah perkembangan yang dia sudah dapat lakukan dengan kemungkinan yang dia dapat kejar pada saat itu:

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

Setelah ditata ulang untuk menghindari menjumlahkan ekor tak terhingga pada distribusi...

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

Dikonversi ke kode 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;
}
```

Jika kita jalankan kode berikut, kita dapat melihat kemungkinan penurunan secara exponensial pada nilai 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
```

Dengan menyelesaikan P kurang daripada 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 mengajukan sebuah sistem transaksi elektronik tanpa perlu bergantung pada kepercayaan. Kita memulai dengan kerangka kerja koin seperti biasa yang terdiri dari tandatangan digital, yang menyediakan kendali yang kuat atas kepemilikan, tapi tetap tidak lengkap tanpa suatu cara untuk mencegah *double-spending*. Untuk mengatasi hal ini, kami mengajukan jaringan *peer-to-peer* dengan *proof-of-work* untuk mencatat histori publik dari transaksi-transaksi yang dengan segera secara komputasi akan menjadi tidak mudah untuk dirubah oleh penyerang jika titik-titik komputasi yang jujur mengendalikan mayoritas kekuatan CPU. Jaringan itu sendiri kuat dikarenakan kesederhanaan strukturnya. Semua titik bekerja secara bersamaan dengan sedikit koordinasi. Mereka tidak perlu teridentifikasi, karena pesan tidak disalurkan kepada tujuan tertentu dan hanya perlu disampaikan dengan dasar usaha terbaik (best effort basis). Titik-titik dapat meninggalkan jaringan dan bergabung kembali sesukanya, menerima rantai *proof-of-work* sebagai bukti atas apa yang terjadi saat mereka tidak tergabung dengan jaringan. Mereka melakukan vote dengan kekuatan CPU, mewakili persetujuan atas blok yang sah dengan mengolah memperpanjang blok tersebut dan menolak blok yang tidak sah dengan tidak mengolah nya. Setiap aturan dan insentif yang diperlukan dapat ditegakkan dengan mekanisme yang sudah dimufakati 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.