بیت‌کوین: یک سیستم پول الکترونیکی فرد به فرد

Satoshi Nakamoto
mailto:satoshin@gmx.com
http://www.bitcoin.org

Translated in Persian from https://bitcoin.org/bitcoin.pdf
by Zahra Amini

چکیده:

یک نسخه‌ی کاملا فرد به فرد از پول الکترونیکی، ارسال مستقیم پرداخت‌های آنلاین از یک شخص به شخص دیگر را بدون نیاز به گذر از موسسه مالی مقدور می‌سازد. امضاهای دیجیتالی بخشی از راه‌حل هستند، اما اگر همچنان به وجود یک واسطه معتمد برای جلوگیری از خرج دوباره پول نیاز باشد، مزایای اصلی آن از بین می‌رود. ما یک راه‌حل برای مشکل خرج دوباره پول با استفاده از شبکه‌ای فرد به فرد، ارائه می‌دهیم. این شبکه تراکنش‌ها را با استفاده ازهش کردن‌شان در قالب یک زنجیره‌ی پیوسته براساس اثبات انجام کار بر پایه‌ی هش، برچسب زمان‌دار می‌کند که بدون انجام دوباره‌ی کاری که انجام شده، قابل تغییر نخواهد بود. بلندترین زنجیره نه تنها به عنوان اثباتی از ترتیب وقایع محسوب می‌شود، بلکه اثبات خروج آن از بزرگ‌ترین مجموعه قدرت CPU است. تا زمانی که غالب قدرت CPU شبکه توسط نودهایی که در حمله به شبکه مشارکت نمی‌کنند، کنترل می‌شود، بلندترین زنجیره توسط آنها ایجاد شده و از مهاجمان پیشی می‌گیرند. خود شبکه به حداقل شکل ساختاری نیازمند است. همچنین پیام‌ها بر پایه‌ی توافقات در شبکه مخابره می‌شوند و نودها می‌توانند به تصمیم خود شبکه را در حالی که با بلندترین زنجیره PoW، به عنوان تایید چیزی که به هنگام رفتن آنها رخ داده، ترک کرده و یا به آن بپیوندند.

۱. مقدمه

تجارت در اینترنت تا به امروز برای پردازش پرداخت‌های الکترونیکی تا حد زیادی منحصرا به موسسات مالی، به عنوان یک واسط معتمد، وابسته بوده‌است. اگرچه این سیستم نسبتا به خوبی پاسخگوی بیشتر تراکنش‌هاست، اما همانند سایر روش‌هایی که بر پایه‌ی اعتماد هستند، ذاتا دارای نقاط ضعف است. انجام تراکنش‌های قطعی وغیرقابل‌بازگشت مقدور نیست؛ چراکه موسسات مالی باید درهنگام اختلاف، وساطت کنند. این واسطه‌گری هزینه‌های تراکنش را افزایش می‌دهد، اندازه حداقلی قابل استفاده تراکنش را محدود می‌کند و انجام تراکنش‌های کوچک مرسوم را نامقدور می‌سازد. عدم امکان انجام تراکنش‌های غیرقابل‌بازگشت در ازای خدمات غیرقابل‌بازگشت بهای سنگین‌تری‌ست که پرداخت میشود چراکه با امکان بازگشت، نیاز به اعتماد افزایش می‌یابد. فروشندگان باید نسبت به مشتریان خود محتاطانه عمل کرده و از آنها اطلاعات بیش از حد مورد نیاز دریافت کنند. به علاوه، همواره درصد مشخصی از خطا به عنوان عاملی اجتناب‌ناپذیر پذیرفته شده است. این هزینه‌ها و نگرانی‌ها در پرداخت‌های حضوری با استفاده از ارزهای مادی قابل پیشگیری است؛ اما در حال حاضر مکانیسمی برای پرداخت با استفاده از راه ارتباطی بدون واسطه موجود نیست.

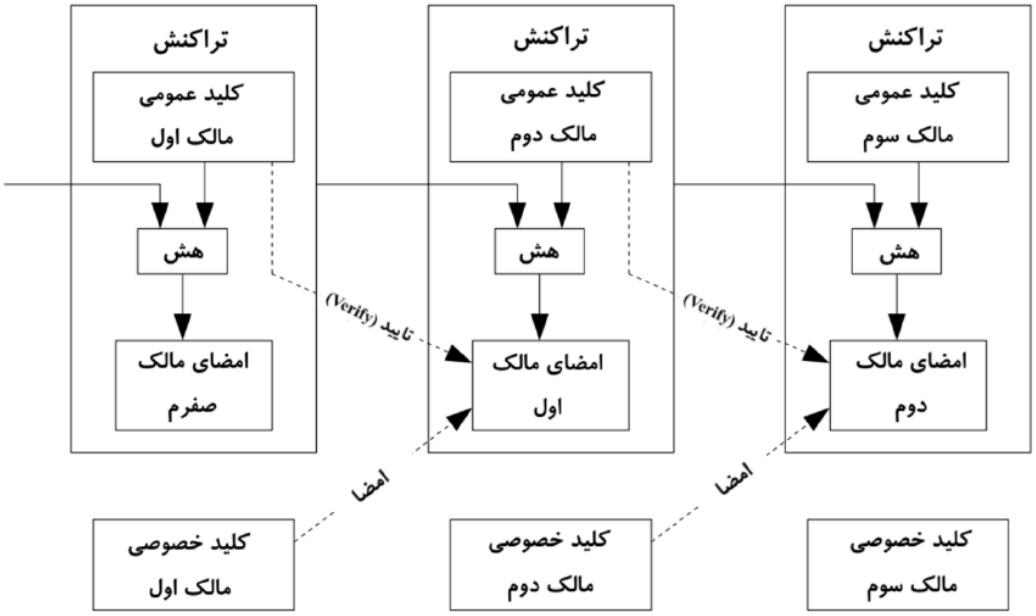
جایگزینی که برای اعتماد مورد نیازست، یک سیستم پرداخت الکترونیکی بر پایه‌ی اثبات رمزنگاری‌ست؛ تا طرفین بدون نیاز به اعتماد به واسطه بتوانند به صورت مستقیم تراکنش انجام دهند. تراکنش‌هایی که از لحاظ محاسباتی غیرقابل‌بازگشت هستند از فروشندگان در برابر جعل و تقلب محافظت می‌کنند و مکانیسم پرداخت تضمینی می‌تواند به راحتی اعمال شده و خریدار را در برابر موارد مشابه مصون دارد.

در این مقاله ما راه‌حلی برای مشکل خرج دوباره‌ی پول با استفاده از یک سرور برچسب زمان‌دار توزیع‌شده‌ی فرد به فرد[1] پیشنهاد می‌کنیم که این سرور اثبات کامپیوتری از ترتیب زمان قرارگیری تراکنش‌ها ایجاد می‌کند. سیستم مذکور تا زمانی که نودهای معتبر[2] جمعا قدرت CPU بیشتری را نسبت به نودهای مهاجم کنترل می‌کنند، امن است.

[1] a peer-to-peer distributed timestamp server
[2] Honest nodes

۲. تراکنش

ما یک سکه‌ی الکترونیکی را به عنوان زنجیره‌ای از امضاهای دیجیتالی توصیف می‌کنیم. هر مالک، به منظور انتقال سکه به فرد دیگری، هش[۳] تراکنش قبلی به علاوه‌ی کلید عمومی مالک بعدی را به صورت دیجیتالی امضا کرده و به انتهای سکه‌ی مذکور ضمیمه می‌کند. مالک جدید می‌تواند از طریق تصدیق این امضاها زنجیره‌ی مالیکت را تایید کند.



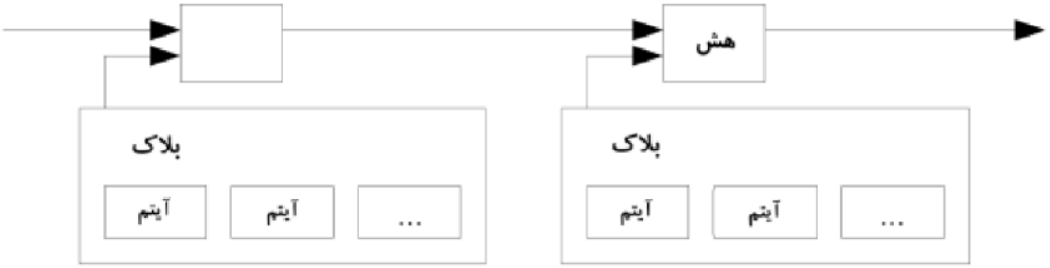
البته مشکل اینست که دریافت‌کننده نمیتواند خرج نشدن دوباره پول توسط مالکین قبلی سکه را تصدیق کند. راه حل معمول برای این مشکل در نظر گرفتن یک واسط معتمد، یا ضرابخانه، است که هر تراکنش را بررسی می‌کند. پس ازهر تراکنش، سکه بایستی به ضرابخانه بازگردد تا یک سکه جدید تولید شود؛ و تنها سکه‌هایی که مستقیما از ضرابخانه منتشر شده‌اند مصون از خرج دوباره محسوب می‌شوند. مشکل این راه‌حل اینست که با اجبار گذر تراکنش‌ها از ضرابخانه، سرنوشت کل سیستم پولی به گروهی که آن را اداره میکنند وابسته است؛ درست مثل سیستم بانکی.

ما نیازمند روشی هستیم که دریافت‌کننده بتواند از امضا نشدن تراکنش‌های پیشین توسط صاحبان قبلی آگاه شود. برای دست‌یابی به هدف‌مان، ما آخرین تراکنش را تراکنش موثق تلقی می‌کنیم و برای ما تلاش‌های بعدی برای خرج دوباره پول حائز اهمیت نیست. آگاهی از تمامی تراکنش‌ها، تنها راه تصدیق عدم حضور یک تراکنش است. در روشی که ضرابخانه اساس قرار داده می‌شود، این مرکز با آگاهی از تمامی تراکنش‌ها، ترتیب آنها را تشخیص می‌دهد. به منظور پیاده‌سازی چنین مدلی بدون نیاز به یک واسط معتمد، لازم است تراکنش‌ها به صورت عمومی مخابره شوند [۱] و همچنین نیاز به سیستمی است که اعضای آن بر سر یک تاریخچه‌ی واحد از ترتیبی که تراکنش‌ها دریافت شده‌است، توافق کنند. از طرفی دریافت‌کننده به اثباتی نیاز دارد تا ثابت کند در زمان هر تراکنش، اکثریت نودها بر سر دریافت تراکنش مذکور پیش از سایرین اتفاق نظر داشته‌اند.

[3]Hash Function

۳. سرور برچسب زمان‌دار

راه‌حلی که ما ارائه می‌دهیم با سرور برچسب زمان‌دار آغاز می‌شود. یک سرور برچسب زمان‌دار هش بلوکی از داده‌ها را گرفته و به آنها برچسب زمانی الحاق می‌کند؛ سپس هش بدست آمده را به طور گسترده منتشر می‌کند؛ چیزی شبیه به انتشار یادداشت در روزنامه و یا یوزنت [۲-۵]. بدیهی‌ست که به منظور محاسبه هش نیاز به داده است، پس وجود این برچسب زمانی در هر لحظه وجود داده را اثبات می‌کند. هر برچسب زمانی، هش برچسب زمانی قبلی خود را نیز در بر می‌گیرد که در مجموع یک زنجیره را شکل می‌دهند و هر برچسب زمانی جدید، برچسب‌های زمانی قبل از خود را تقویت می‌کند.



۴. اثبات انجام کار

به منظور اعمال یک سرور برچسب زمان‌دار توزیع‌شده در سطح فرد به فرد، به جای استفاده از یادداشت‌های روزنامه و یا یوزنتی، ما نیاز به یک سیستم اثبات انجام کار[۴] همانند سیستم هش‌کش Adam Back داریم [۶]. سیستم اثبات انجام کار شامل وارسی برای مقداری است که اگر هش آن (برای مثال با استفاده از SHA-256) محاسبه گردد، این عدد هش‌شده با تعدادی از بیت‌های صفری آغاز شود. متوسط کار مورد نیاز برای دست‌یابی به این مقدار از لحاظ بیت‌های صفری مورد نظر به صورت نمایی است و با انجام یک هش قابل تصدیق می‌باشد.

برای اعمال PoW در شبکه‌ی برچسب زمان‌دار، ما از افزودن یک نانس در بلوک استفاده می‌کنیم تا مقداری یافت شود که به هش بلوک، بیت‌های صفری مورد نیازش را بدهد. زمانی که تلاش CPU برای یافت مقدار مورد نظر PoW نتیجه دهد، بلوک مذکور بدون بازگردانی کار انجام‌شده، قابل تغییر نیست. همین طور که بلوک‌های بعدی به صورت زنجیره‌وار به آن متصل می‌شوند، برای تغییر بلوک مذکور باید کار انجام‌شده روی تمامی بلوک‌ها مجدد انجام گیرد.

[4] Proof-of-work (PoW)

همچنین سیستم اثبات انجام کار، مشکل تشخیص نظر غالب در تصمیم‌گیری‌های جمعی را حل می‌کند. اگر اکثریت بر اساس یک آدرس IP - یک رای بود، کسی که می‌توانست به بیش از یک IP دسترسی داشته باشد، می‌توانست در این تصمیم‌گیری اخلال ایجاد کند. در سیستم PoW هر CPU در نهایت یک رای دارد. تصمیم اکثریت توسط بلندترین زنجیره مشخص می‌شود که این زنجیره دارای بیش‌ترین اثبات انجام کار صرف‌شده است. چنانچه غالب قدرت CPU توسط نودهای معتبر کنترل شود، زنجیره موثق سریع‌تر رشد کرده و از زنجیره‌های رقیب پیشی می‌گیرد. به منظور تغییر یک بلوک قدیمی، فرد مهاجم باید علاوه بر انجام دوباره PoW بلوک، اثبات انجام کار تمامی بلوک‌های پس از آن را نیز دوباره انجام دهد و سپس به نودهای معتبر رسیده و از آن‌ها سبقت بگیرد. در ادامه نشان خواهیم داد که احتمال اینکه یک مهاجم به این نقطه برسد با اضافه شدن بلوک‌های بعدی به صورت نمایی کاهش می‌یابد.
به منظور جبران افزایش سرعت سخت‌افزار و تعداد نودهای فعال در بازه‌های زمانی متفاوت، سختی PoW با متوسط متغیری غیرثابت معین می‌شود که متوسط تعداد بلوک‌ها به ازای یک ساعت را مدنظر قرار می‌دهد. اگر بلوک‌ها با سرعت زیادی ایجاد شوند، سختی افزایش می‌یابد.

۵. شبکه

شبکه را می‌توان طبق مراحل زیر اجرا کرد:

۱. تراکنش‌های جدید به تمامی نودها مخابره می‌شوند.
۲. هر نود تراکنش‌های جدید را در یک بلوک ذخیره می‌کند.
۳. هر نود سعی می‌کند تا جواب یک PoW سخت را برای بلوک خود بیابد.
۴. زمانی که یک نود جواب PoW را می‌یابد، بلوک خود را به تمامی نودها مخابره می‌کند.
۵. سایر نودها بلوک مذکور را تنها در شرایطی می‌پذیرند که کل تراکنش‌های آن صحیح با شند و قبلا خرج نشده باشند.
۶. سایر نودها موافقت خود را با بلوک مورد نظر با ایجاد بلوک بعدی در زنجیره اعلام می‌دارند و به این منظور از هش بلوک پذیرفته‌شده به عنوان هش قبلی استفاده می‌کنند.

نودها همواره بلندترین زنجیره را به عنوان زنجیره صحیح در نظر می‌گیرند و برای گسترش آن تلاش می‌کنند. اگر دو نود دو نسخه‌ی متفاوت از بلوک بعدی را به صورت هم‌زمان مخابره کنند، ممکن است برخی نودها یک نسخه و برخی نودها نسخه‌ی دیگر را دریافت کنند. در این شرایط آن‌ها بر روی اولین نسخه‌ای که دریافت کرده‌اند کار می‌کنند؛ اما نوع دیگر را در شاخه‌ای متفاوت ذخیره خواهند کرد تا اگر آن شاخه بلندتر شد، آن را ادامه دهند. این اتصال زمانی که جواب PoW بعدی یافت شود قطع شده و تنها یک شاخه طویل‌تر خواهد بود؛ در این حالت سایر نودهایی که بر روی نسخه‌ی دیگر بلوک کار انجام داده بودند به شاخه بلندتر تغییر مسیر می‌دهند.
نیازی نیست که تراکنش‌های جدید منتشرشده الزاما به همه‌ی نودها برسند. تا زمانی که به تعدادی از نودها برسد، به زودی در یک بلوک قرار خواهند گرفت. همچنین بلوک‌های منتشرشده نسبت به پیام‌های ازقلم‌افتاده مصون‌اند؛ به این معنی که اگر یک نود بلوکی را دریافت نکند،هر زمان که در حال دریافت بلوک بعدی باشد متوجه عدم حضور بلوک خواهد شد و درخواست دریافت آن را نیز می‌کند.

۶. انگیزه

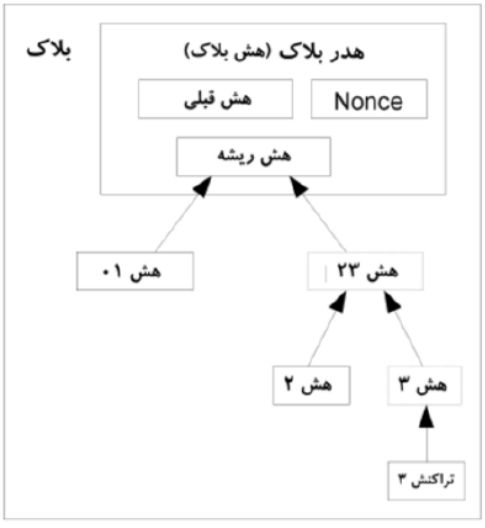
به صورت قراردادی، اولین تراکنش در یک بلوک، تراکنش خاصی محسوب می‌شود که سکه‌ی جدیدی را ایجاد کرده و مالک آن تولیدکننده‌ی آن بلوک است. این مساله یک انگیزه برای نودها ایجاد می‌کند تا از شبکه پشتیبانی کنند و روشی برای توزیع سکه‌ها به منظور به گردش درآوردن آنها مهیا کنند؛ چراکه هیچ مرکزیتی برای چاپ آن‌ها موجود نیست. می‌توانیم این رویه‌ی متدوام تولید مقداری از سکه‌های جدید را به استخراج‌کنندگان طلا تشبیه کنیم که با صرف منابع، طلا را به گردش درمی‌آورند. در مقوله‌ی ما، این منابع زمان CPU و برق مصرفی است.

همچنین این انگیزه می‌تواند با کارمزد تراکنش‌ها نیز تامین گردد. اگر مقدار خروجی یک تراکنش از مقدار ورودی آن کم‌تر باشد، مابه‌تفاوت یک کارمزد تراکنش است که به میزان انگیزه بلوک حاوی تراکنش می‌افزاید. زمانی که تعداد از پیش تعیین‌شده‌ی سکه‌ها به گردش درآمد، انگیزه تنها می‌تواند متمرکز کارمزد تراکنش‌ها شود و کاملا عاری از تورم باشد.

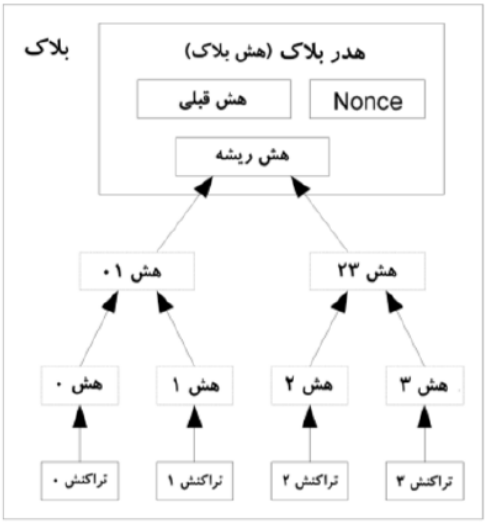
این انگیزه می‌تواند نودها را تشویق به درست‌کاری کند. اگر یک مهاجم آزمند بتواند قدرت CPU بیشتری از تمامی نودهای درست‌کار جمع کند، باید بین استفاده‌ی این قدرت در جهت سرقت پرداخت‌های مردم و کلاه‌برداری از آن‌ها و یا در جهت تولید سکه‌های جدید، انتخاب کند. برای این شخص پیروی از قوانین باید سودمندتر باشد؛ قوانینی که به او از مجموع سایرین، سکه‌های جدید بیشتری بدهد تا او نخواهد سیستم و ارزش سرمایه خود را زیر پا بگذارد.

۷. بازیابی فضای دیسک

به محض اینکه آخرین تراکنش در یک سکه زیر بلوک‌های کافی قرار گرفت، تراکنش‌های خرج‌شده‌ی قبل از آن می‌توانند نادیده گرفته شوند تا فضای دیسک محفوظ باشد.به منظور انجام این کار بدون نیاز به شکستن هش بلوک، تراکنش‌ها در یک درخت مرکل هش می‌شوند [۷][۲][۵] که تنها ریشه در هش بلوک دخیل است. بلوک‌های قدیمی متعاقبا به این روش می‌توانند با استفاده از کوتاه کردن شاخه‌های درخت فشرده شوند. نیازی به ذخیره‌ی هش‌های داخلی نیست.



بعد از هرس کردن تراکنش های ۰ تا ۲ از بلاک

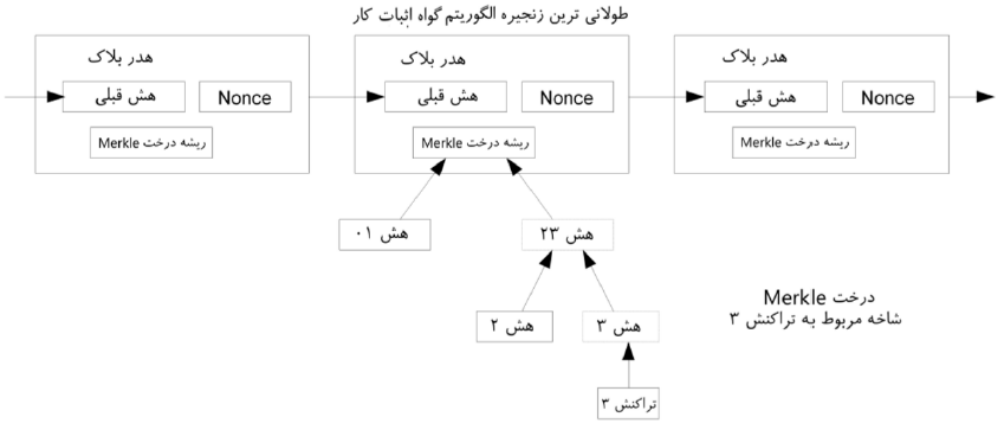


تراکنش های هش شده در درخت **Merkle**

حجم سربرگ بلوکی که حاوی تراکنش نیست تقریبا حدود ۸۰ بایت خواهد بود. چنانچه زمان تولید بلوک‌ها را هر ۱۰ دقیقه در نظر بگیریم، ۸۰ بایت *۶*۲۴*۳۶۵= ۴/۲ مگابایت بر سال خواهد بود. سیستم‌های کامپیوتری که در سال ۲۰۰۸ به فروش می‌رسند عموما دارای RAM دو گیگابایتی هستند و قانون مور رشد فعلی را سالانه ۱/۲ گیگابایت تخمین می‌زند؛ درنتیجه فضای ذخیره‌ای حتی درصورت لزوم به نگه‌داری سربرگ‌های بلوک در حافظه، نباید مشکل‌ساز باشد.

۸. تایید تسهیل‌شده‌ی پرداخت

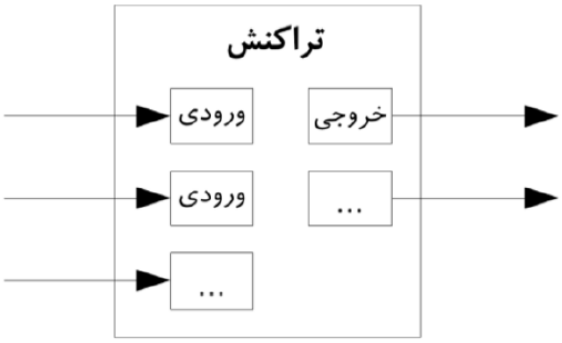
تایید پرداخت‌ها بدون اجرای یک نود شبکه‌ی کامل امکانپذیرست. کاربر تنها باید یک کپی از سربرگ‌های بلوک از طویل‌ترین زنجیره‌ی PoW داشته باشد؛ که می‌تواند این اطلاعات را از نودهای شبکه درخواست کند تا زمانی که از داشتن بلندترین زنجیره و شاخه مرکلی که تراکنش را به بلوکی که در آن برچسب زمان‌دار شده است متصل می‌کند، اطمینان حاصل کند. او به خودی خود، نمی‌تواند تراکنش را بررسی کند؛ اما با اتصال آن به مکانی در زنجیره، می‌تواند پذیرفته شدن آن را توسط یک نود شبکه مشاهده کند و بلوک‌های افزوده‌شده‌ی پس از آن تصدیقی بر اینست که شبکه آن را پذیرفته است.



به این ترتیب تا زمانی که نودهای معتبر شبکه را کنترل می‌کنند، تاییدها قابل اعتماد است؛ اما اگر غالب قدرت شبکه به دست مهاجم باشد، آسیب‌پذیرتر خواهد بود. ضمن اینکه نودهای شبکه می‌توانند تراکنش‌ها را تایید کنند، چنانچه مهاجم بتواند غالب قدرت خود در شبکه را حفظ کند، روش تسهیل‌شده توسط تراکنش‌های ساختگی مهاجم قابل دست‌کاری خواهد بود. یک روش برای جلوگیری از این مساله اینست که امکان پذیرفتن اخطار از نودهای شبکه زمانی که بلوک نامعتبری را مشاهده می‌کنند، موجود باشد. به این صورت که دانلود کل بلوک و تراکنش‌های مشکوک برای تصدیق ناهماهنگی به نرم‌افزار کاربر القا شود. کسب‌وکارهایی که پرداخت‌های مکرر دارند بهترست نودهای خود را اجرا کنند تا امنیت خودگردان بیش‌تر و تایید سریع‌تر داشته باشند.

۹. مقدار ترکیبی و تفکیکی

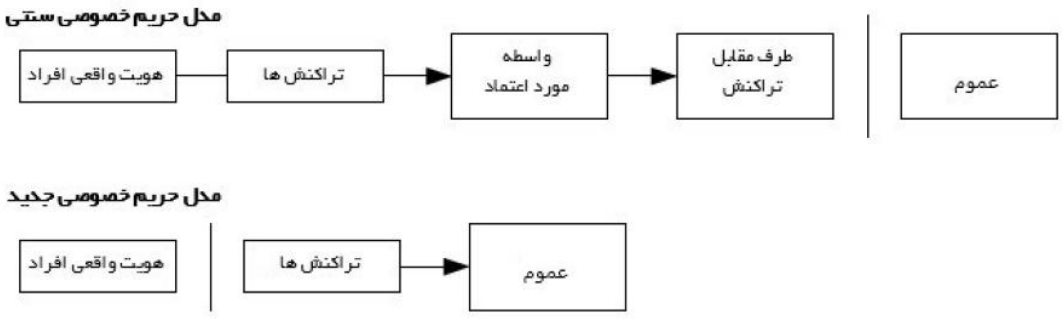
اگرچه مدیریت سکه‌ها به صورت جداگانه ممکن است؛ اما ایجاد یک تراکنش جدا برای انتقال هر سِنت ناکارآمد به نظر می‌رسد. برای اینکه تفکیک و ترکیب هر مقدار ممکن باشد، تراکنش‌ها دارای چندین ورودی و خروجی هستند. معمولا یا یک ورودی از یک تراکنش بزرگ‌تر قبلی خواهد بود و یا چندین ورودی، مقادیر اندک را تلفیق خواهند کرد؛ و در بسیاری از موارد دو خروجی خواهیم داشت: یکی برای پرداخت و دیگری برای بازگشت باقی مانده. اگر باقی‌مانده موجود باشد، به فرستنده بازمی‌گردد.



لازم به ذکرست که ظرفیت خروجی، جایی که تراکنش به تراکنش‌های متعددی و آن تراکنش‌ها به تراکنش‌های بیش‌تری وابسته است، در این جا مشکل‌ساز نیست. هیچ گاه نیازی به استخراج یک کپی مستقل کامل از تاریخچه‌ی تراکنش‌ها نخواهد بود.

۱۰. حریم خصوصی

روش سنتی بانک‌داری دسترسی به اطلاعات را به گروه‌های دخیل و گروه‌های واسطه معتمد محدود می‌کند و به این روش به سطحی از حریم خصوصی دست می‌یابد. ضرورت مخابره‌ی تمامی تراکنش‌ها به صورت عمومی مانع این روال می‌شود؛ اما حریم خصوصی همچنان می‌تواند با استفاده از جلوگیری از جریان اطلاعات به مکانی دیگر رعایت شود: بدین صورت که کلیدهای خصوصی به شکل ناشناس نگه‌داری شوند. عموم می‌توانند ارسال مقداری از یک شخص به شخص دیگر را بدون اطلاعاتی که تراکنش را به فردی مرتبط کند، ببینند. این سطح از اطلاعات منتشرشده مشابه صرافی‌های بورس است؛ جایی که زمان و اندازه‌ی معاملات فردی، یا همان tape، بدون افشای هویت افراد، عمومی است.



به عنوان یک لایه محافظتی (دیوارآتش) مازاد، برای هر تراکنش بایستی یک جفت کلید جدید استفاده شود تا قابل ربط به صاحب همیشگی آن نباشد. البته اندازه‌ای از قابلیت ارتباط با وجود تراکنش‌هایی که چند ورودی دارند، اجتناب ناپذیر است؛ چراکه واضح است ورودی آنها الزاما یک مالک دارد. احتمال خطر این مساله اینست که اگر صاحب کلید مشخص شود، ارتباط او با سایر تراکنش‌هایش نمایان خواهد شد.

۱۱. محاسبات

ما سناریویی را متصور می‌شویم که در آن مهاجم سعی می‌کند زنجیره‌ی جایگزینی سریع‌تر از زنجیره‌ی معتبر تولید کند. حتا اگر این سناریو قابل دست‌یابی باشد، سیستم را به سمت تغییرات دلخواه مانند تولید ارزش از هیچ و یا دریافت پولی که متعلق به مهاجم نبوده است، سوق نمی‌دهد. نودها تراکنش‌های نامعتبر را برای پرداخت نمی‌پذیرند و نودهای معتبر بلوک‌های حاوی این تراکنش‌ها را قبول نخواهند کرد. یک مهاجم تنها می‌تواند سعی بر تغییر یکی از تراکنش‌های خود کند تا پولی که اخیرا خرج کرده است را پس بگیرد.

رقابت بین زنجیره‌ی معتبر و زنجیره‌ی مهاجم را می‌توان به عنوان گشت تصادفی دوجمله‌ای توصیف کرد. پیروزی، طویل‌تر شدن زنجیره‌ی معتبر به اندازه‌ی یک بلوک، با افزایش **+۱** و شکست، طویل‌تر شدن زنجیره‌ی مهاجم به اندازه‌ی یک بلوک، با کاهش **-۱**.

احتمال اینکه مهاجم بتواند با این کسری به زنجیره معتبر برسد، مشابه مساله پاکباختگی قمارباز است. تصور کنید یک قمارباز با موجودی نامحدود شروع به زیان می‌کند و با تعداد ذاتا محدودی از آزمایش برای رسیدن به نقطه سربه‌سر بازی می‌کند. ما میتوانیم احتمال اینکه آیا هیچگاه به نقطه سربه سر میرسد و یا اینکه یک مهاجم میتواند به زنجیره معتبر برسد را به شکل زیر محاسبه کنیم[۸]:

p = احتمال پیدا کردن بلوک بعدی توسط یک نود معتبر
q = احتمال پیدا کردن بلوک بعدی توسط مهاجم
qz = احتمال اینکه مهاجم از z بلوک عقب‌تر بتواند به زنجیره معتبر برسد

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

با توجه به فرض اولیه‌ی ما که p > q است، با افزایش تعداد بلوک‌هایی که مهاجم باید بگذراند تا به زنجیره معتبر برسد، احتمال آن به صورت نمایی کاهش پیدا می‌کند. اگر شرایط وفق مراد مهاجم نباشد و نتواند به سرعت پیش برود، هر چه از زنجیره عقب‌تر بیفتد، شانس او بسیار کم و ناچیز خواهد شد.

حالا بررسی میکنیم که دریافت‌کننده‌ی یک تراکنش جدید چه مدت باید صبر کند تا بتواند به قدر کافی از عدم امکان تغییر تراکنش توسط فرستنده اطمینان حاصل کند. ما فرض می‌کنیم که فرستنده یک مهاجم است که قصد دارد دریافت‌کننده را در رابطه با پرداخت برای مدتی فریب دهد؛ سپس بعد از

گذشت مدتی پرداخت را برای خود بازگشت بزند. در این حالت به گیرنده اخطار داده خواهد شد اما فرستنده امیدوارست که کار از کار گذشته باشد.
گیرنده یک جفت کلید جدید تولید می‌کند و کلید عمومی را قبل از امضای آن در اختیار فرستنده قرار می‌دهد. این کار از آماده‌سازی زنجیرهای از بلوک‌ها پیش از زمان مقرر توسط فرستنده جلوگیری می‌کند و او باید برای این کار به طور مداوم تلاش کند تا زمانی که به اندازه کافی خوش‌شانس بوده و به اندازه مورد نیاز جلو بیفتد؛ سپس در آن زمان تراکنش را اعمال کند. وقتی که تراکنش ارسال شد، فرستنده‌ی متقلب مخفیانه شروع به تلاش در زنجیره‌ی موازی که حاوی نسخه‌ی جایگزین تراکنش اوست، می‌کند. گیرنده تا وقتی که تراکنش به یک بلوک اضافه شود و تعداد z بلوک به آن متصل شود، منتظر می‌ماند. او از پیشرفت دقیق مهاجم بی‌اطلاع است؛ اما تخمین میزند که بلوک‌های معتبر زمان متوسط مورد انتظار بر حسب هر بلوک را طی کرده‌اند؛ پتانسیل پیشرفت مهاجم با توزیع پواسون، مقدار مورد انتظار زیر خواهد بود:

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

حالا به منظور دستیابی به احتمال رسیدن مهاجم، ما تراکم پواسون را در هر مقدار پیشرفتی که مهاجم ممکن است با احتمالی که از آن نقطه به بعد به‌دست آورده باشد، ضرب می‌کنیم:

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

معادله را برای جلوگیری از جمع‌بندی انتهای متناهی از توزیع مجدد مرتب می‌کنیم:

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

و به زبان C تبدیل می‌کنیم:

```
#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;
}
```

نتایج را اجرا می‌کنیم؛ کاهش احتمال به شکل نمایی با متغیر z مشاهده می‌شود:

```
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
```

حل Pهای کمتر از ۰/۱ % ...

```
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
```

۱۲. نتیجه

ما سیستمی برای تراکنش‌های الکترونیکی پیشنهاد کردیم که به اعتماد متکی نیست. ما با چهارچوب معمولی که از امضاهای دیجیتالی سکه‌ها ساخته می‌شوند و کنترل بسیاری برای مالک مهیا می‌کنند، شروع کردیم؛ اما این شیوه بدون روشی برای جلوگیری از خرج دوباره پول تکمیل نبود. به منظور حل این مشکل ما یک شبکه فرد به فرد را با استفاده ازسیستم اثبات انجام کار پیشنهاد کردیم که تاریخچه‌ی همگانی از تراکنش‌ها را ثبت و ضبط کند که تا زمانی که قدرت غالب CPU توسط نودهای معتبر کنترل شود، تغییر در آن سریعا از لحاظ محاسباتی برای یک مهاجم غیرعملی باشد. شبکه با سادگی غیرساختاری خود پایدارست. نودها همگی به شکل هم‌زمان و با حداقل نیاز به هماهنگی کار می‌کنند. آن‌ها نیاز به شناسایی ندارند چرا که پیغام‌ها به یک مکان مشخصی هدایت نمی‌شوند و تنها نیاز است تا بر اساس توافق به مقصد برسند. نودها می‌توانند شبکه را در حالی که با زنجیره‌ی PoW به عنوان اثباتی از وقایع رخ‌داده زمانی که از شبکه رفته‌اند، ترک کنند و یا به آن بپیوندند. آن‌ها با استفاده از قدرت CPU رای خواهند داد و پذیرش خود را با قبول بلوک‌های معتبر به صورت تلاش برای گسترش آنها و عدم پذیرش خود را با رد بلوک‌های نامعتبر و عدم انجام کار بر روی آن‌ها، اعلام می‌کنند. هر گونه قانون و انگیزه‌هایی میتواند توسط مکانیسم توافق جمعی[5] اعمال گردد.

منابع

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

۲. 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.

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

۴. 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.

۵. 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.

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

۷. R.C. Merkle, "Protocols for public key cryptosystems," In Proc. *1980 Symposium on Security and Privacy*, IEEE Computer Society, pages 122-133, April 1980.

۸. W. Feller, "An introduction to probability theory and its applications," 1957.

[5] Consensus mechanism