بیت‌کوین: سیستم پول نقد الکترونیکی همتا به همتا

ساتوشی ناکاموتو

satoshin@gmx.com

www.bitcoin.org

مترجم: حامد رحمتی

چکیده. *یک نسخه‌ی کاملاً همتا به همتا از پول نقد الکترونیکی این امکان را فراهم می‌آورد تا پرداخت‌های آنلاین مستقیماً از یک طرف به طرف دیگر و بدون عبور از یک مؤسسه‌ی مالی ارسال شوند. امضای دیجیتال بخشی از راه‌حل را فراهم می‌کند، اما اگر همچنان به یک شخص ثالث قابل اعتماد برای جلوگیری از دوبار خرج کردن نیاز باشد، مزایای اصلی از بین می‌رود. ما راه‌حلی برای مسئله‌ی دوبار خرج کردن با استفاده از شبکه‌ای همتا به همتا پیشنهاد می‌کنیم. شبکه، تراکنش‌ها را با درهم‌سازی آن‌ها در زنجیره‌ای مداوم از اثبات کار مبتنی بر درهم، مُهر زمانی می‌زند و سابقه‌ای ایجاد می‌کند که بدون انجام دوباره‌ی اثبات کار، قابل تغییر نیست. طولانی‌ترین زنجیره نه تنها به‌عنوان اثبات توالی رویدادهای مشاهده‌شده عمل می‌کند، بلکه اثبات می‌کند که از بزرگ‌ترین مجموعه‌ی قدرت پردازش (CPU) آمده است. تا زمانی که اکثریت قدرت پردازش توسط گره‌هایی کنترل می‌شود که برای حمله به شبکه همکاری نمی‌کنند، آن‌ها طولانی‌ترین زنجیره را تولید کرده و از مهاجمان پیشی می‌گیرند. خود شبکه به ساختاری حداقلی نیاز دارد. پیام‌ها بر اساس بهترین تلاش پخش می‌شوند و گره‌ها می‌توانند به‌دلخواه شبکه را ترک کرده و دوباره به آن ملحق شوند و طولانی‌ترین زنجیره‌ی اثبات کار را به‌عنوان اثبات آنچه در غیابشان رخ داده است، بپذیرند.*

۱. مقدمه

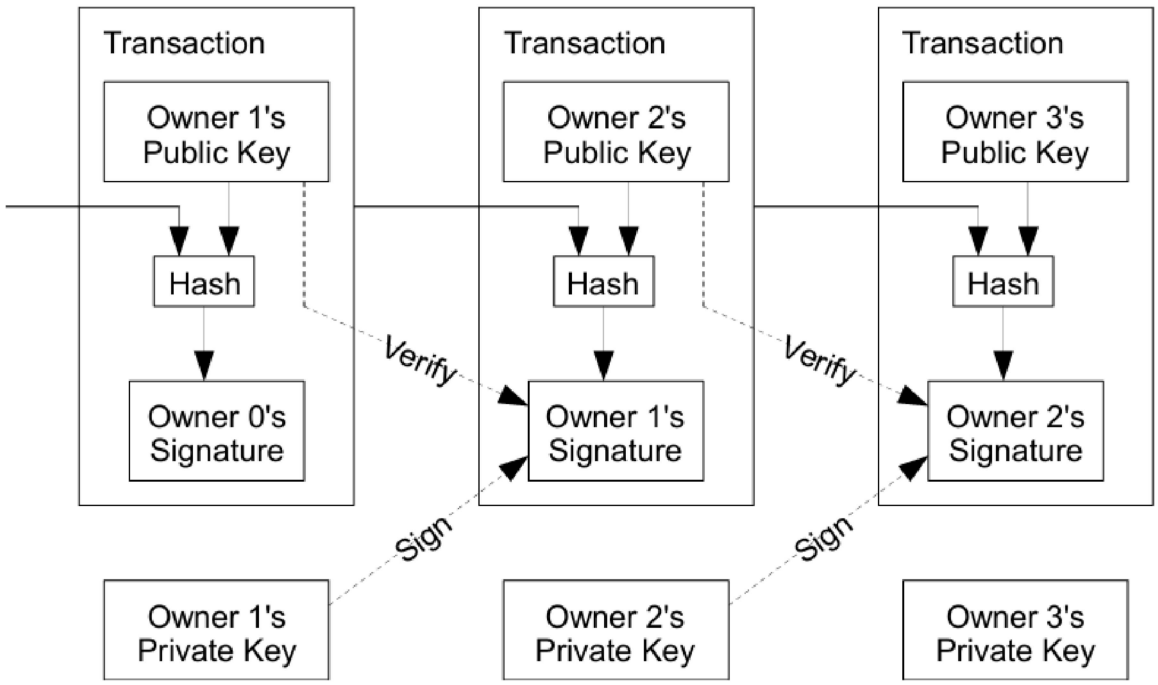
تجارت در اینترنت تقریباً به‌طور انحصاری به مؤسسات مالی به‌عنوان اشخاص ثالث قابل اعتماد برای پردازش پرداخت‌های الکترونیکی متکی شده است. در حالی که این سیستم برای بیشتر تراکنش‌ها به‌خوبی کار می‌کند، همچنان از ضعف‌های ذاتی مدل مبتنی بر اعتماد رنج می‌برد. تراکنش‌های کاملاً غیرقابل بازگشت واقعاً ممکن نیستند، زیرا مؤسسات مالی نمی‌توانند از میانجی‌گری در اختلافات اجتناب

کنند. هزینه‌ی میانجی‌گری، هزینه‌های تراکنش را افزایش می‌دهد و اندازه‌ی حداقلی عملی تراکنش را محدود کرده و امکان تراکنش‌های کوچک و غیررسمی را از بین می‌برد. همچنین هزینه‌ای گسترده‌تر به‌صورت از دست رفتن توانایی انجام پرداخت‌های غیرقابل بازگشت برای خدمات غیرقابل بازگشت وجود دارد. با امکان بازگشت، نیاز به اعتماد گسترش می‌یابد. فروشندگان باید مراقب مشتریان خود باشند و برای کسب اطلاعات بیش از آنچه که در غیر این صورت نیاز داشتند، آن‌ها را به زحمت بیندازند. درصد مشخصی از تقلب به‌عنوان امری اجتناب‌ناپذیر پذیرفته می‌شود. این هزینه‌ها و عدم قطعیت‌های پرداخت را می‌توان به‌صورت حضوری با استفاده از پول فیزیکی دور زد، اما هیچ سازوکاری برای انجام پرداخت از طریق یک کانال ارتباطی بدون وجود یک طرف قابل اعتماد وجود ندارد.

آنچه مورد نیاز است یک سیستم پرداخت الکترونیکی مبتنی بر اثبات رمزنگارانه به جای اعتماد است که به هر دو طرف مایل اجازه دهد مستقیماً با یکدیگر تراکنش کنند، بدون نیاز به شخص ثالث قابل اعتماد. تراکنش‌هایی که بازگشت آن‌ها از نظر محاسباتی غیرعملی است، از فروشندگان در برابر تقلب محافظت می‌کند و سازوکارهای معمول ضمانت (اسکرو) می‌توانند به‌راحتی برای محافظت از خریداران پیاده‌سازی شوند. در این مقاله، ما راه‌حلی برای مسئله‌ی دوبار خرج کردن با استفاده از یک سرور مُهر زمانی توزیع‌شده‌ی همتا به همتا برای تولید اثبات محاسباتی از ترتیب زمانی تراکنش‌ها پیشنهاد می‌کنیم. این سیستم تا زمانی امن است که گره‌های صادق مجموعاً قدرت پردازش بیشتری نسبت به هر گروه مهاجم همکار کنترل کنند.

۲. تراکنش‌ها

ما یک سکه‌ی الکترونیکی را به‌عنوان زنجیره‌ای از امضاهای دیجیتال تعریف می‌کنیم. هر مالک، سکه را با امضای دیجیتال درهمی از تراکنش قبلی و کلید عمومی مالک بعدی و افزودن این‌ها به انتهای سکه، به مالک بعدی منتقل می‌کند. گیرنده‌ی پرداخت می‌تواند امضاها را برای تأیید زنجیره‌ی مالکیت بررسی کند.



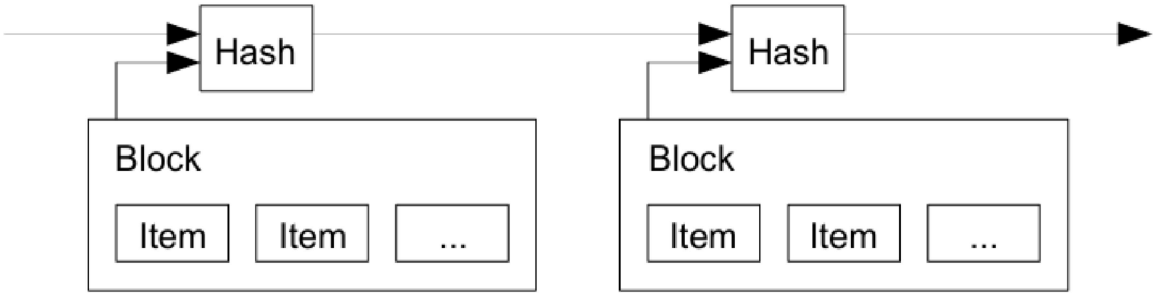
نمودار ۱: زنجیره‌ی تراکنش شامل امضاهای دیجیتال و کلیدهای عمومی.

البته مشکل این است که گیرنده نمی‌تواند تأیید کند که یکی از مالکان قبلی سکه را دوبار خرج نکرده باشد. یک راه‌حل رایج، معرفی یک نهاد مرکزی قابل اعتماد یا «ضرابخانه» است که هر تراکنش را برای دوبار خرج کردن بررسی می‌کند. پس از هر تراکنش، سکه باید به ضرابخانه بازگردانده شود تا سکه‌ای جدید صادر شود و تنها سکه‌هایی که مستقیماً از ضرابخانه صادر می‌شوند، قابل اعتماد برای دوبار خرج نشدن هستند. مشکل این راه‌حل این است که سرنوشت کل سیستم پولی به شرکتی که ضرابخانه را اداره می‌کند بستگی دارد و هر تراکنشی باید از طریق آن‌ها انجام شود، درست مانند یک بانک.

ما به راهی نیاز داریم تا گیرنده بداند که مالکان قبلی هیچ تراکنش زودتری را امضا نکرده‌اند. برای اهداف ما، قدیمی‌ترین تراکنش همان است که به حساب می‌آید، بنابراین به تلاش‌های بعدی برای دوبار خرج کردن اهمیت نمی‌دهیم. تنها راه برای تأیید عدم وجود یک تراکنش، آگاهی از تمام تراکنش‌ها است. در مدل مبتنی بر ضرابخانه، ضرابخانه از تمام تراکنش‌ها آگاه بود و تصمیم می‌گرفت کدام یک زودتر دریافت شده است. برای انجام این کار بدون یک طرف قابل اعتماد، تراکنش‌ها باید به‌طور عمومی اعلام شوند و ما به سیستمی نیاز داریم تا شرکت‌کنندگان بر سر یک تاریخچه‌ی واحد از ترتیب دریافت آن‌ها به توافق برسند. گیرنده‌ی پرداخت نیاز به اثبات دارد که در زمان هر تراکنش، اکثریت گره‌ها توافق داشتند که آن تراکنش اولین تراکنش دریافت‌شده بوده است.

۳. سرور مُهر زمانی

راه‌حلی که پیشنهاد می‌کنیم با یک سرور مُهر زمانی آغاز می‌شود. یک سرور مُهر زمانی با گرفتن درهمی از مجموعه‌ای از اقلامی که باید مُهر زمانی بخورند و انتشار گسترده‌ی آن درهم، مثلاً در یک روزنامه یا پست یوزنت، کار می‌کند. مُهر زمانی ثابت می‌کند که داده‌ها در آن زمان وجود داشته‌اند تا بتوانند در درهم قرار گیرند. هر مُهر زمانی، مُهر زمانی قبلی را در درهم خود شامل می‌شود و زنجیره‌ای را تشکیل می‌دهد که با هر مُهر زمانی اضافی، مُهرهای پیشین تقویت می‌شوند.

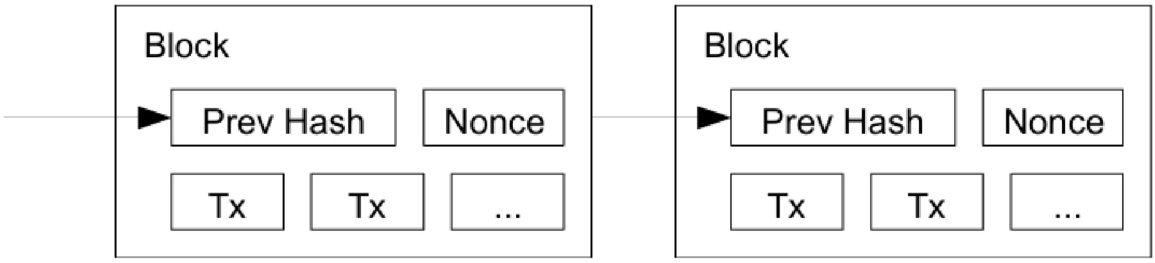


نمودار ۲: سرور مُهر زمانی که زنجیره‌ای از بلوک‌ها را با درهم‌های متوالی ایجاد می‌کند.

۴. اثبات کار

برای پیاده‌سازی یک سرور مُهر زمانی توزیع‌شده به‌صورت همتا به همتا، نیاز داریم از یک سیستم اثبات کار مشابه Hashcash آدام بک استفاده کنیم، نه پست‌های روزنامه یا یوزنت. اثبات کار شامل اسکن برای یافتن مقداری است که وقتی درهم‌سازی می‌شود (مثلاً با SHA-256)، درهم با تعداد مشخصی بیت صفر شروع شود. متوسط کار مورد نیاز با تعداد بیت‌های صفر مورد نیاز به‌صورت نمایی افزایش می‌یابد و با اجرای یک درهم‌سازی واحد قابل تأیید است.

برای شبکه‌ی مُهر زمانی خود، ما اثبات کار را با افزایش یک nonce در بلوک پیاده‌سازی می‌کنیم تا زمانی که مقداری یافت شود که به درهم بلوک، بیت‌های صفر مورد نیاز را بدهد. هنگامی که تلاش CPU برای ارضای اثبات کار صرف شد، بلوک بدون انجام دوباره‌ی کار قابل تغییر نیست. با زنجیره شدن بلوک‌های بعدی پس از آن، کار برای تغییر بلوک شامل انجام دوباره‌ی تمام بلوک‌های بعد از آن نیز می‌شود.



نمودار ۳: زنجیره‌ی اثبات کار که بلوک‌ها با Nonce و اشاره به درهم قبلی به هم متصل می‌شوند.

اثبات کار همچنین مشکل تعیین نمایندگی در تصمیم‌گیری اکثریت را حل می‌کند. اگر اکثریت بر اساس «یک آدرس IP یک رأی» می‌بود، هر کسی که می‌توانست IPهای زیادی تخصیص دهد، آن را براندازی می‌کرد. اثبات کار اساساً «یک CPU یک رأی» است. تصمیم اکثریت توسط طولانی‌ترین زنجیره نشان داده می‌شود که بیشترین تلاش اثبات کار روی آن سرمایه‌گذاری شده است. اگر اکثریت قدرت CPU توسط گره‌های صادق کنترل شود، زنجیره‌ی صادق سریع‌ترین رشد را خواهد داشت و از هر زنجیره‌ی رقیب پیشی می‌گیرد. برای تغییر یک بلوک گذشته، مهاجم باید اثبات کار آن بلوک و تمام بلوک‌های پس از آن را دوباره انجام دهد و سپس به کار گره‌های صادق برسد و از آن پیشی بگیرد. ما بعداً نشان خواهیم داد که احتمال رسیدن یک مهاجم کندتر با اضافه شدن بلوک‌های بعدی به‌صورت نمایی کاهش می‌یابد.

برای جبران سرعت فزاینده‌ی سخت‌افزار و علاقه‌ی متغیر در اجرای گره‌ها در طول زمان، سختی اثبات کار توسط یک میانگین متحرک تعیین می‌شود که تعداد متوسط بلوک در ساعت را هدف قرار می‌دهد. اگر بلوک‌ها خیلی سریع تولید شوند، سختی افزایش می‌یابد.

۵. شبکه

مراحل اجرای شبکه به شرح زیر است:

1. تراکنش‌های جدید به تمام گره‌ها پخش می‌شوند.
2. هر گره تراکنش‌های جدید را در یک بلوک جمع‌آوری می‌کند.
3. هر گره روی یافتن یک اثبات کار دشوار برای بلوک خود کار می‌کند.
4. هنگامی که گره‌ای اثبات کار را یافت، بلوک را به تمام گره‌ها پخش می‌کند.
5. گره‌ها بلوک را تنها در صورتی می‌پذیرند که تمام تراکنش‌های موجود در آن معتبر بوده و قبلاً خرج نشده باشند.

6. گره‌ها پذیرش خود از بلوک را با شروع به کار برای ایجاد بلوک بعدی در زنجیره، با استفاده از درهم بلوک پذیرفته‌شده به‌عنوان درهم قبلی، ابراز می‌کنند.

گره‌ها همواره طولانی‌ترین زنجیره را به‌عنوان زنجیره‌ی صحیح در نظر می‌گیرند و به کار برای گسترش آن ادامه می‌دهند. اگر دو گره نسخه‌های متفاوتی از بلوک بعدی را به‌طور همزمان پخش کنند، برخی گره‌ها ممکن است یکی را زودتر از دیگری دریافت کنند. در این صورت، آن‌ها روی اولین موردی که دریافت کرده‌اند کار می‌کنند، اما شاخه‌ی دیگر را در صورتی که طولانی‌تر شود ذخیره می‌کنند. با یافتن اثبات کار بعدی و طولانی‌تر شدن یکی از شاخه‌ها، تساوی شکسته خواهد شد؛ گره‌هایی که روی شاخه‌ی دیگر کار می‌کردند سپس به شاخه‌ی طولانی‌تر تغییر مسیر می‌دهند.

پخش تراکنش‌های جدید لزوماً نیازی ندارد به تمام گره‌ها برسد. تا زمانی که به گره‌های زیادی برسند، به‌زودی در یک بلوک قرار خواهند گرفت. پخش بلوک‌ها نیز نسبت به پیام‌های گم‌شده تحمل‌پذیر است. اگر گره‌ای بلوکی را دریافت نکند، هنگامی که بلوک بعدی را دریافت کند و متوجه شود که یکی را از دست داده، آن را درخواست خواهد کرد.

۶. انگیزه

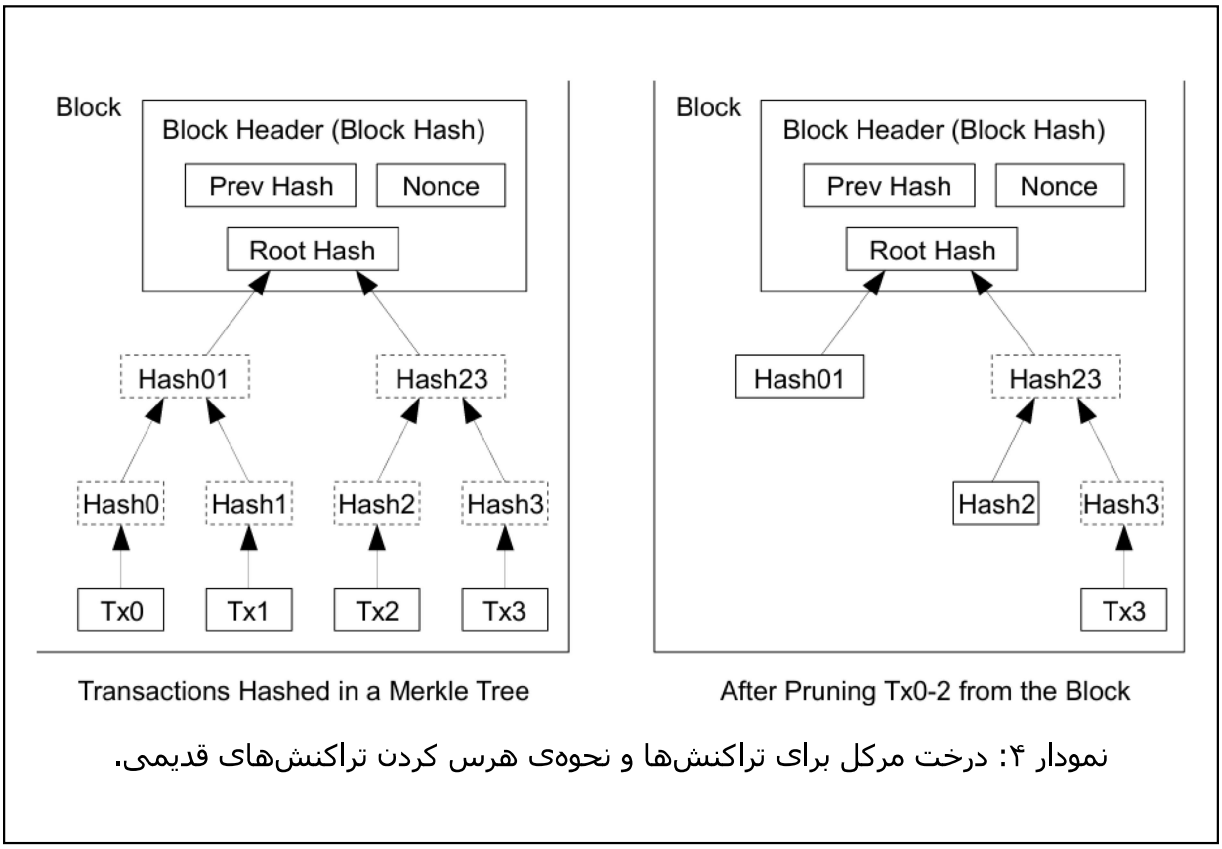
طبق قرارداد، اولین تراکنش در یک بلوک، یک تراکنش ویژه است که سکه‌ای جدید متعلق به ایجادکننده‌ی بلوک را آغاز می‌کند. این امر انگیزه‌ای برای گره‌ها جهت پشتیبانی از شبکه ایجاد می‌کند و راهی برای توزیع اولیه‌ی سکه‌ها در گردش فراهم می‌آورد، زیرا هیچ نهاد مرکزی برای انتشار آن‌ها وجود ندارد. افزودن پیوسته‌ی مقدار ثابتی از سکه‌های جدید مشابه استخراج‌کنندگان طلا است که برای افزودن طلا به گردش، منابع مصرف می‌کنند. در مورد ما، این زمان CPU و الکتریسیته است که مصرف می‌شود.

انگیزه همچنین می‌تواند با کارمزد تراکنش‌ها تأمین شود. اگر مقدار خروجی یک تراکنش کمتر از مقدار ورودی آن باشد، این تفاوت یک کارمزد تراکنش است که به ارزش انگیزشی بلوک حاوی آن تراکنش اضافه می‌شود. هنگامی که تعداد از پیش تعیین‌شده‌ای از سکه‌ها وارد گردش شد، انگیزه می‌تواند به‌طور کامل به کارمزد تراکنش‌ها منتقل شده و کاملاً عاری از تورم باشد.

انگیزه ممکن است به تشویق گره‌ها برای صادق ماندن کمک کند. اگر یک مهاجم حریص بتواند قدرت CPU بیشتری نسبت به تمام گره‌های صادق جمع‌آوری کند، باید بین استفاده از آن برای کلاهبرداری از مردم با پس گرفتن پرداخت‌هایش، یا استفاده از آن برای تولید سکه‌های جدید، یکی را انتخاب کند. او باید دریابد که بازی بر اساس قوانین، قوانینی که به او سکه‌های جدید بیشتری نسبت به مجموع دیگران می‌دهد، سودآورتر از تضعیف سیستم و اعتبار ثروت خودش است.

۷. بازپس‌گیری فضای دیسک

هنگامی که آخرین تراکنش در یک سکه زیر تعداد کافی بلوک مدفون شد، تراکنش‌های خرج‌شده‌ی قبل از آن می‌توانند برای صرفه‌جویی در فضای دیسک دور ریخته شوند. برای تسهیل این کار بدون شکستن درهم بلوک، تراکنش‌ها در یک درخت مرکل درهم‌سازی می‌شوند و تنها ریشه در درهم بلوک گنجانده می‌شود. بلوک‌های قدیمی سپس می‌توانند با هرس کردن شاخه‌های درخت فشرده شوند. درهم‌های داخلی نیازی به ذخیره‌سازی ندارند.



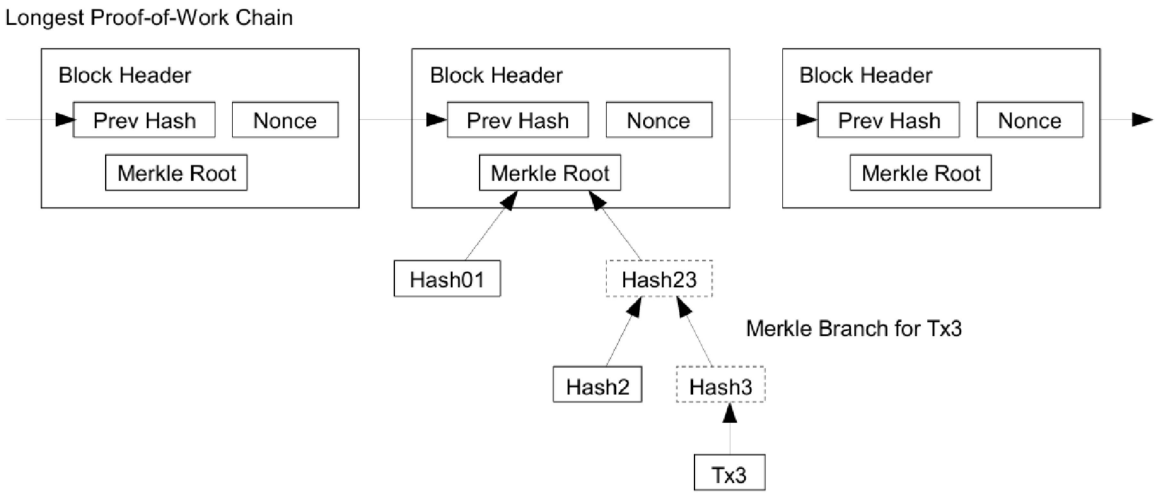
نمودار ۴: درخت مرکل برای تراکنش‌ها و نحوه‌ی هرس کردن تراکنش‌های قدیمی.

یک سرآیند بلوک بدون تراکنش حدود ۸۰ بایت خواهد بود. اگر فرض کنیم بلوک‌ها هر ۱۰ دقیقه تولید می‌شوند، ۸۰ بایت * ۶ * ۲۴ * ۳۶۵ = ۴.۲ مگابایت در سال. با توجه به اینکه سیستم‌های کامپیوتری معمولاً در سال ۲۰۰۸ با ۲ گیگابایت رم فروخته می‌شدند و قانون مور رشد فعلی ۱.۲ گیگابایت در سال را پیش‌بینی می‌کند، ذخیره‌سازی حتی اگر سرآیندهای بلوک باید در حافظه نگهداری شوند، نباید مشکلی ایجاد کند.

۸. تأیید ساده‌ی پرداخت

تأیید پرداخت‌ها بدون اجرای یک گره‌ی کامل شبکه امکان‌پذیر است. یک کاربر تنها نیاز دارد یک کپی از سرآیندهای بلوک طولانی‌ترین زنجیره‌ی اثبات کار را نگه دارد که می‌تواند با پرس‌وجو از گره‌های شبکه تا زمانی که متقاعد شود طولانی‌ترین زنجیره را دارد، به دست آورد و شاخه‌ی مرکلی که تراکنش را به بلوکی که در آن مُهر زمانی خورده متصل می‌کند، دریافت کند. او نمی‌تواند تراکنش را خودش بررسی کند، اما با

اتصال آن به مکانی در زنجیره، می‌تواند ببیند که یک گره‌ی شبکه آن را پذیرفته است و بلوک‌های اضافه‌شده پس از آن، پذیرش شبکه را بیشتر تأیید می‌کنند.



نمودار ۵: فرآیند تأیید ساده‌ی پرداخت با استفاده از طولانی‌ترین زنجیره و شاخه‌ی مرکل.

به این ترتیب، تأیید تا زمانی که گره‌های صادق شبکه را کنترل می‌کنند قابل اعتماد است، اما اگر شبکه توسط یک مهاجم تحت‌الشعاع قرار گیرد، آسیب‌پذیرتر است. در حالی که گره‌های شبکه می‌توانند تراکنش‌ها را برای خود تأیید کنند، روش ساده‌شده می‌تواند توسط تراکنش‌های جعلی مهاجم فریب داده شود، تا زمانی که مهاجم بتواند به تسلط بر شبکه ادامه دهد. یک استراتژی برای محافظت در برابر این امر، پذیرش هشدارها از گره‌های شبکه هنگامی که یک بلوک نامعتبر را شناسایی می‌کنند است که نرم‌افزار کاربر را وادار به دانلود بلوک کامل و تراکنش‌های هشدار داده‌شده برای تأیید ناسازگاری می‌کند. کسب‌وکارهایی که پرداخت‌های مکرر دریافت می‌کنند احتمالاً همچنان مایل خواهند بود گره‌های خود را برای امنیت مستقل بیشتر و تأیید سریع‌تر اجرا کنند.

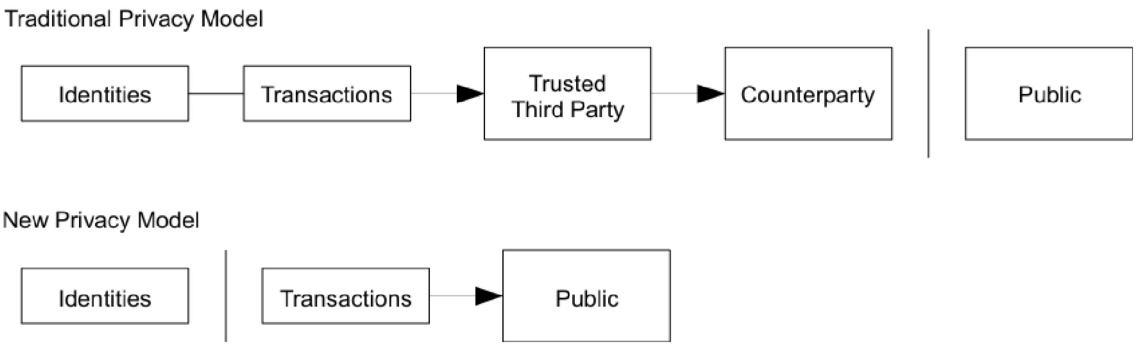
۹. ترکیب و تقسیم ارزش

اگرچه امکان مدیریت تکی سکه‌ها وجود دارد، اما ایجاد یک تراکنش جداگانه برای هر سنت در یک انتقال، دشوار خواهد بود. برای امکان‌پذیر شدن تقسیم و ترکیب ارزش، تراکنش‌ها شامل چندین ورودی و خروجی هستند. به‌طور معمول، یا یک ورودی واحد از یک تراکنش بزرگ قبلی وجود خواهد داشت یا چندین ورودی که مقادیر کوچک‌تر را ترکیب می‌کنند، و حداکثر دو خروجی: یکی برای پرداخت و دیگری برای بازگرداندن باقی‌مانده، در صورت وجود، به فرستنده.

لازم به ذکر است که پدیده‌ی Fan-out، جایی که یک تراکنش به چندین تراکنش وابسته است و آن تراکنش‌ها به تعداد بیشتری وابسته هستند، در اینجا مشکلی ایجاد نمی‌کند. هرگز نیازی به استخراج یک کپی مستقل کامل از تاریخچه‌ی یک تراکنش نیست.

۱۰. حریم خصوصی

مدل بانکداری سنتی با محدود کردن دسترسی به اطلاعات به طرف‌های درگیر و شخص ثالث قابل اعتماد، به سطحی از حریم خصوصی دست می‌یابد. ضرورت اعلام عمومی تمام تراکنش‌ها این روش را منتفی می‌کند، اما همچنان می‌توان با شکستن جریان اطلاعات در جایی دیگر، حریم خصوصی را حفظ کرد: با ناشناس نگه‌داشتن کلیدهای عمومی. عموم می‌توانند ببینند که شخصی مبلغی را برای شخص دیگری ارسال می‌کند، اما بدون اطلاعاتی که تراکنش را به فرد خاصی مرتبط کند. این شبیه به سطح اطلاعاتی است که توسط بورس‌های اوراق بهادار منتشر می‌شود، جایی که زمان و اندازه‌ی معاملات فردی، «نوار»، عمومی می‌شود، اما بدون اینکه مشخص شود طرف‌ها چه کسانی بوده‌اند.



نمودار ۶: مقایسه‌ی مدل حریم خصوصی سنتی (بالا) و مدل جدید با کلیدهای عمومی ناشناس (پایین).

به‌عنوان یک دیوار آتش اضافی، باید برای هر تراکنش از یک جفت کلید جدید استفاده شود تا از ارتباط آن‌ها به یک مالک مشترک جلوگیری شود. برخی ارتباطها همچنان با تراکنش‌های چندورودی اجتناب‌ناپذیر است، که ضرورتاً نشان می‌دهند ورودی‌های آن‌ها متعلق به یک مالک بوده است. این خطر وجود دارد که اگر مالک یک کلید فاش شود، ارتباطدهی بتواند تراکنش‌های دیگری را که متعلق به همان مالک بوده‌اند، آشکار کند.

۱۱. محاسبات

ما سناریوی مهاجمی را در نظر می‌گیریم که تلاش می‌کند زنجیره‌ای جایگزین را سریع‌تر از زنجیره‌ی صادق تولید کند. حتی اگر این کار انجام شود، سیستم را به روی تغییرات دلخواه، مانند ایجاد ارزش از هیچ یا گرفتن پولی که هرگز به مهاجم تعلق نداشته، باز نمی‌کند. گره‌ها یک تراکنش نامعتبر را به‌عنوان پرداخت نخواهند پذیرفت و گره‌های صادق هرگز بلوکی حاوی آن‌ها را نمی‌پذیرند. یک مهاجم تنها می‌تواند تلاش کند یکی از تراکنش‌های خود را تغییر دهد تا پولی را که اخیراً خرج کرده است، پس بگیرد.

رقابت بین زنجیره‌ی صادق و زنجیره‌ی مهاجم می‌تواند به‌عنوان یک گشت تصادفی دوجمله‌ای (Binomial Random Walk) توصیف شود. رویداد موفقیت، گسترش زنجیره‌ی صادق به میزان یک بلوک است که برتری آن را ۱+ افزایش می‌دهد و رویداد شکست، گسترش زنجیره‌ی مهاجم به میزان یک بلوک است که شکاف را ۱- کاهش می‌دهد.

احتمال رسیدن مهاجم از یک عقب‌ماندگی مشخص، مشابه مسئله‌ی ورشکستگی قمارباز (Gambler's Ruin) است. فرض کنید یک قمارباز با اعتبار نامحدود از یک کسری شروع می‌کند و به‌طور بالقوه تعداد بی‌نهایت تلاش برای رسیدن به نقطه‌ی سر به سر انجام می‌دهد. ما می‌توانیم احتمال اینکه او هرگز به نقطه‌ی سر به سر برسد، یا اینکه مهاجم هرگز به زنجیره‌ی صادق برسد را به صورت زیر محاسبه کنیم:

p = احتمال اینکه یک گره‌ی صادق بلوک بعدی را پیدا کند

q = احتمال اینکه مهاجم بلوک بعدی را پیدا کند

q_z = احتمال اینکه مهاجم هرگز از z بلوک عقب‌ماندگی به زنجیره برسد

$q_z = 1$ اگر $p \leq q$

$q_z = (q/p)^z$ اگر $p > q$

با فرض ما مبنی بر اینکه $p > q$، احتمال با افزایش تعداد بلوک‌هایی که مهاجم باید جبران کند، به‌صورت نمایی کاهش می‌یابد. با توجه به اینکه شانس علیه اوست، اگر در ابتدا یک جهش خوش‌شانس نداشته باشد، با عقب‌افتادن بیشتر، شانس‌هایش به شدت ناچیز می‌شود.

اکنون در نظر می‌گیریم که دریافت‌کننده‌ی یک تراکنش جدید چه مدت باید منتظر بماند تا به‌اندازه‌ی کافی مطمئن شود که فرستنده نمی‌تواند تراکنش را تغییر دهد. فرض می‌کنیم فرستنده یک مهاجم است که می‌خواهد دریافت‌کننده را برای مدتی باور دهد که به او پرداخت کرده است، سپس آن را تغییر دهد تا پس از مدتی به خودش بازپرداخت کند. دریافت‌کننده هنگامی که این اتفاق بیفتد هشدار دریافت می‌کند، اما فرستنده امیدوار است که دیگر خیلی دیر شده باشد.

دریافت‌کننده یک جفت کلید جدید تولید می‌کند و کلید عمومی را کمی قبل از امضا به فرستنده می‌دهد. این کار از آماده‌سازی زنجیره‌ای از بلوک‌ها از پیش توسط فرستنده، با کار مداوم روی آن تا زمانی که به‌اندازه‌ی کافی جلو بیفتد و سپس اجرای تراکنش در آن لحظه، جلوگیری می‌کند. پس از ارسال تراکنش،

فرستنده‌ی متقلب شروع به کار مخفیانه روی یک زنجیره‌ی موازی حاوی نسخه‌ای جایگزین از تراکنش خود می‌کند.

دریافت‌کننده منتظر می‌ماند تا تراکنش به یک بلوک اضافه شود و z بلوک پس از آن به آن متصل شوند. او میزان دقیق پیشرفت مهاجم را نمی‌داند، اما با فرض اینکه بلوک‌های صادق میانگین زمان مورد انتظار برای هر بلوک را گرفته‌اند، پیشرفت بالقوه‌ی مهاجم یک توزیع پواسون با مقدار مورد انتظار خواهد بود:

$$\lambda = z * (q / p)$$

برای به دست آوردن احتمال اینکه مهاجم همچنان بتواند اکنون به زنجیره برسد، چگالی پواسون را برای هر میزان پیشرفتی که می‌توانست داشته باشد در احتمال اینکه بتواند از آن نقطه به زنجیره برسد ضرب می‌کنیم:

$$\left(\text{if } k \leq z \text{ else } 1 \right. \left.^{(z-k)}(q/p) \right) * (!\lambda^k e^{-\lambda} / k) \Sigma$$

با بازآرایی برای جلوگیری از جمع دنباله‌ی بی‌نهایت توزیع...

$$(1 - (q/p)^{(z-k)}) * (!\lambda^k e^{-\lambda} / k) \Sigma - 1$$

تبدیل به کد 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;
}
```

با اجرای برخی نتایج، می‌توانیم کاهش نمایی احتمال را با z مشاهده کنیم.

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 برای P کمتر از 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

۱۲. نتیجه‌گیری

ما سیستمی برای تراکنش‌های الکترونیکی بدون اتکا به اعتماد پیشنهاد کرده‌ایم. ما با چارچوب معمول سکه‌های ساخته‌شده از امضاهای دیجیتال شروع کردیم که کنترل قوی بر مالکیت فراهم می‌کند، اما بدون راهی برای جلوگیری از دوبار خرج کردن ناقص است. برای حل این مشکل، ما یک شبکه‌ی همتا به همتا با استفاده از اثبات کار برای ثبت یک تاریخچه‌ی عمومی از تراکنش‌ها پیشنهاد کردیم که اگر گره‌های صادق اکثریت قدرت CPU را کنترل کنند، تغییر آن برای یک مهاجم به سرعت از نظر محاسباتی غیرعملی می‌شود. این شبکه در سادگی بدون ساختار خود مقاوم است. گره‌ها همگی به‌طور همزمان با هماهنگی اندک کار می‌کنند. آن‌ها نیازی به شناسایی ندارند، زیرا پیام‌ها به مکان خاصی مسیریابی نمی‌شوند و تنها بر اساس بهترین تلاش نیاز به تحویل دارند. گره‌ها می‌توانند به‌دلخواه شبکه را ترک کرده و دوباره به آن ملحق شوند و زنجیره‌ی اثبات کار را به‌عنوان اثبات آنچه در غیابشان رخ داده است بپذیرند. آن‌ها با قدرت CPU خود رأی می‌دهند و پذیرش بلوک‌های معتبر را با کار برای گسترش آن‌ها و رد بلوک‌های نامعتبر را با خودداری از کار روی آن‌ها ابراز می‌کنند. هرگونه قوانین و انگیزه‌های مورد نیاز را می‌توان با این سازوکار اجماع اعمال کرد.

منابع

[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.

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

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