Bitcoin: Digitální peníze typu peer-to-peer

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

Přeložil: Lukáš Smrž

Shrnutí. Verze elektronických peer-to-peer (rovný s rovným) peněz, která umožňuje posílat online platby mezi dvěma účastníky, aniž by prošly finanční institucí. Digitální podpisy jsou částečné řešení problému, ale hlavní výhody jsou ztraceny, když je zapotřebí třetí strana k zabránění dvojí útratě (double-spending). Navrhujeme řešení problému dvojí útraty pomocí sítě rovný s rovným (peer-to-peer). Síť označí každou transakci „časovým razítkem“ (timestamp) a to tak, že je zařadí do probíhajícího řetězce (chain), pomocí důkazu práce (proof-of-work), který je zahashován do kódu (hash-based). Vytvoří se tak záznam, který je neměnný dokud se nezopakuje vykonaná práce (Proof-of-work). Nejdelší řetězec není jen důkaz o posloupnosti událostí, kterých byl řetězec svědkem, ale i jako důkaz, že dochází z největšího množství výpočetní síly v síti (CPU power). Pokud je kontrolována uzly, které nespolupracují, aby úmyslně zaútočili na síť, vytvoří nejdelší řetězec (chain) a předběhnou tak útočníky. Samotná síť vyžaduje minimální strukturu. Zprávy jsou vysílány na základě nejlepšího úsilí uzlů, které se mohou kdykoli odpojit a připojit. Při připojení (prvním i opětovném) přijmou nejdelší řetězec (chain) důkazů o práci (Proof-of-work), jako důkaz toho, co se stalo v jejich nepřítomnosti.

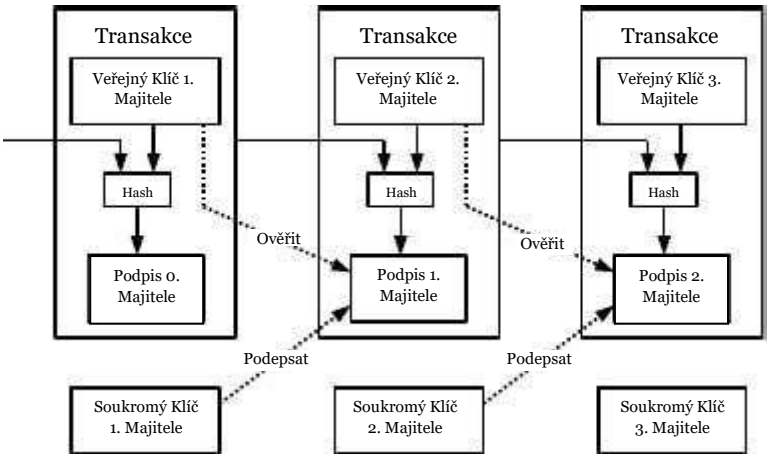
1. Úvod

Obchod na internetu se začal téměř vždy spoléhat na finanční instituce, které slouží při zpracování elektronických plateb jako důvěryhodné třetí strany. Zatím co systém funguje výborně pro většinu transakcí, stále jsou zde nedílné slabosti, kvůli modelu založeném na důvěře. Zcela nevratné transakce nejsou ve skutečnosti možné, protože se finanční instituce nemohou vyhnout vyjednávání sporu. Náklady na zprostředkování zvyšují náklady a limitují minimální praktickou velikost transakcí a omezují možnost malých nepravidelných transakcí a také zde je zjevný náklad spojený se ztrátou schopnosti provádět nevratné platby, za nevratné služby. S možností vrácení, se potřeba důvěry zvyšuje. Obchodníci si musí dávat pozor na své zákazníky a požadovat po nich více informací než by bylo nutné. Určité procento podvodů se dokonce považuje za nevyhnutelné. Těmto nákladům a nejistotám u plateb se lze vyhnout použitím fyzické měny, ale neexistuje žádný platební mechanismus přes komunikační kanál bez nutnosti důvěry v třetí stranu.

Co je potřebné, je elektronický platební systém založený na kryptografickém důkazu, místo důvěry, umožňující jakýmkoliv dvěma stranám obchodovat přímo mezi sebou, bez potřeby důvěryhodné třetí strany. Transakce, které jsou výpočetně nepraktické na vratnost platby, by chránili prodejce před podvody a běžné mechanizmy podmínečných smluv (escrow) by se mohly lehko implementovat, kvůli ochraně kupujícího. V této práci navrhujeme řešení problému dvojí útraty (double-spending) pomocí využití distribuovaného serveru s „časovými razítkami“ (timestamp), fungujícího na bázi peer-to-peer (rovný s rovným) s cílem vytvoření výpočetního důkazu o chronologickém pořadí transakcí. Systém je bezpečný pokud čestné uzly (honest nodes) mají společně větší výpočetní sílu než, spolupracující skupinka uzlů útočníka.

2. Transakce

Elektronickou minci definujeme jako řetězec digitálních podpisů. Každý majitel převede mince jinému, tím že digitálně podepíše hash z předchozí transakce spolu s veřejným klíčem budoucího vlastníka a přidá je na konec mince. Příjemce platby může ověřit podpisy a tím ověřit řetězec vlastnictví.

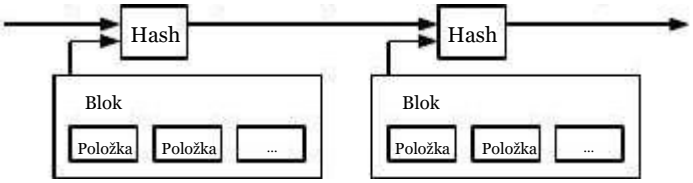


Problém je, že si příjemce nemůže ověřit, jestli jeden z majitelů nezaplatil dvakrát (double-spend) jednou mincí. Běžným řešením je představit důvěryhodnou centrální autoritu nebo mincovnu, která kontroluje každou transakci, kvůli dvojí útratě. Po každé transakci se musí mince vrátit mincovně, aby vydala novou minci, a jenom nové mince vydané přímo z mincovny mají důvěru, že nebudou dvakrát utracené. Problém s tímto řešením spočívá v tom, že osud celého tohoto peněžního systému závisí na společnosti, která provozuje mincovnu, přičemž každá transakce musí touto společností projít, stejně jako v případě bank.

Potřebujeme, aby příjemce věděl, že předchozí vlastníci, již dříve nepodepsali jiné transakce. Pro naše účely je nejranější transakce ta, která se počítá, takže nás nezajímají pozdější pokusy o dvojí útratu (double-spend). Jediným způsobem, jak potvrdit absenci transakce, je znát všechny transakce. V modelu založeném na mincovně si mincovna byla vědoma všech transakcí a rozhodovala o jejich chronologickém pořadí. Abychom dosáhli tohoto bez důvěryhodné třetí strany, musí být transakce veřejně oznámené [1], a taky je potřeba systém pro účastníky, který jim umožní souhlasit s historií a chronologickým pořadím transakcí. Příjemce potřebuje důkaz, že v době každé transakce se většina uzlů (Nodes) shodla na tom, že byla přijata jako první.

3. Server s časovým razítkem (Timestamp server)

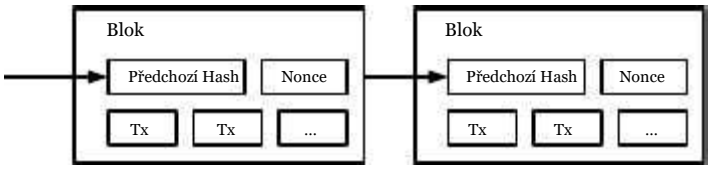
Řešení, které navrhujeme, začíná serverem s časovým razítkem (timestamp). Tento server funguje tak, že bere hash bloku položek, který má být časově označen a široce publikovaný, například v novinách nebo v Usenet oznámení [2-5]. Časové razítko dokazuje, že data v tom čase musela existovat, aby se dostala do hashe. Každé časové razítko obsahuje předchozí časové razítko ve svém hashi, přičemž každé další časové razítko posiluje ty před ním.



4. Důkaz prací (Proof-of-Work)

Na implementaci distribuovaného serveru s časovým razítkem na bázi rovný s rovným (peer-to-peer) budeme muset místo systému novin nebo Usenet oznámení použít systém důkazu práce podobný Hashcashu [6] od Adama Backa. Důkaz o práci zahrnuje hledání hodnoty, která při zadání do hashovací funkce, jako například u SHA-256, začíná s počtem nulových bitů. Průměrná požadovaná práce je exponenciální v počtu požadovaných nulových bitů a může být ověřena vykonáním jediného hashe.

Implementaci důkazu práce v naší síti s časovým razítkem (timestamp) vykonáme připočítáním tzv. nonce v bloku, dokud není nalezena hodnota, která přidává hashu bloku požadované nulové bity. Jakmile se vynaloží výpočetní síla, která vyhoví důkazu o práci, blok není možné změnit bez opětovného vykonání této práce. Vzhledem k tomu, že pozdější bloky jsou řetězově navázané na předešlé, práce na změnu bloku by zahrnovala opětovné přepracování všech navazujících bloků.



Důkaz prací (Proof-of-work) taky řeší problém spojený s určováním zastoupení, v systému rozhodování většiny. Pokud by většina byla na jedné IP-adrese-jednom-hlasu, každý kdo by věděl, jak se rozdělit na víc IP adres by mohl tento systém rozvrátit. Důkaz prací je v podstatě jedna-výpočetní-síla-jeden-hlas (one-CPU-one-vote). Rozhodnutí většiny představuje nejdelší řetězec, protože na něj bylo vynaložené největší úsilí spojené s důkazem práce. Pokud je většina výkonu procesní síly (CPU) řízena poctivými uzly, poctivý řetězec bude růst nejrychleji a předběhne všechny konkurenční řetězce. K úpravě minulého bloku by musel útočník opět vykonat důkaz o práci toho bloku a následně všech následujících bloků, dokud se nevyrovná a nepředežene řetězec, na kterém pracují čestné uzly. Později si ukážeme, že pravděpodobnost útočníka doběhnout čestné uzly exponenciálně klesá s každým přidaným blokem.

Aby se kompenzovala časem zvyšující se rychlost hardwaru a měnící se zájem o provozování uzlů, náročnost důkazu práce je určena klouzavým průměrem zaměřeným na průměrný počet bloků za hodinu. Jestli jsou bloky generovány příliš rychle, dochází ke zvýšení náročnosti.

5. Síť

Kroky na spuštění sítě jsou následující:

1) Nové transakce se vysílají do všech uzlů.
2) Každý uzel shromažďuje jednotlivé transakce do bloku.
3) Každý uzel pracuje na nalezení náročného důkazu o práci pro jeho blok.
4) Když uzel najde důkaz o práci, vysílá blok do všech uzlů.
5) Uzly přijmou blok jen tehdy, pokud jsou všechny transakce v něm platné a nejsou už utracené.
6) Uzly vyjádří přijetí bloku tak, že při práci na vytvoření dalšího bloku v řetězci použijí hash přijatého bloku jako předchozí hash / referenci.

Uzly vždy považují nejdelší řetězec za správný a budou pokračovat v práci na jeho prodlužování. Pokud dva uzly vysílají současně různé verze dalšího bloku, mohou je některé uzly přijmout v různém pořadí. V takovém případě pracují na prvním, který obdrželi, ale uloží si i druhou větev (s druhým blokem), pro případ, kdyby byla nakonec delší. Nerozhodnost bude přerušena, když se najde další důkaz o práci a jedna větev se stane delší; uzly, které pracovaly na druhé větvi, se potom přepnou na delší větev.

Nové vysílání transakcí nemusí nutně zasáhnout všechny uzly. Pokud dosáhne na mnoho uzlů, budou brzy přidány do bloku. Vysílání bloku je též tolerantní k nedoručeným zprávám. Když uzel blok neobdrží, uvědomí si tuto mezeru při obdržení dalšího bloku a vyžádá si chybějící blok.

6. Motivace

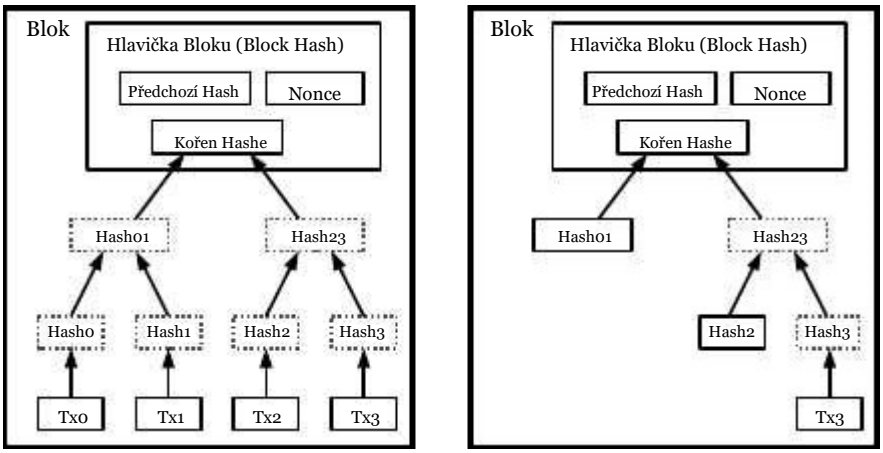
První transakce v bloku je zpravidla speciální, protože obsahují novou minci, kterou vlastní tvůrce bloku. To motivuje uzly podporovat síť a poskytuje způsob jak nově vytěžené mince distribuovat do oběhu, protože neexistuje žádný centrální orgán, který by je vydával. Stálý přírůstek konstantního množství nových mincí je analogický k těžbě zlata, kde horníci vynakládají zdroje na vytěžení a vpuštění zlata do oběhu. V našem případě jsou těmito vynaloženými věcmi čas a elektřina věnovaná výpočtům (hashum).

Stimuly lze také financovat transakčními poplatky. Pokud je výstupní hodnota transakce menší než vstupní hodnota, rozdíl je transakční poplatek. Ten se připočítá k stimulační hodnotě bloku, ve kterém se transakce nachází. Jakmile se do oběhu dostane předem určený počet mincí, transakční poplatky se stanou jediným stimulem a inflace zcela vymizí.

Tento podnět motivuje uzly nepodvádět. Pokud je chamtivý útočník schopen shromáždit více výpočetní síly než všechny čestné uzly, musel by si vybrat, jestli podvede lidi ukradením zpět jeho plateb nebo generováním nových mincí. Potencionální útočník by měl pochopit, že je pro něho výdělečnější hrát podle pravidel, které mu upřednostní víc mincí než všem ostatním dohromady, než aby si podkopal systém a tím i hodnotu jeho majetku.

7. Přivlastňování místa na disku

Jakmile je nejnovější transakce pochována pod dostatečným množstvím bloků, dříve utracené transakce mohou být vymazané, aby se ušetřilo místo na disku. Abychom mohli tuto metodu využívat bez porušení hashu bloku, je transakce hashovaná v tzv. Merkle stromě [7][2][5], přičemž jenom kořen je obsažen v hashi bloku. Staré bloky mohou být odstraněním větví tohoto stromu kompaktnější a vnitřní hashe nemusí být uloženy.



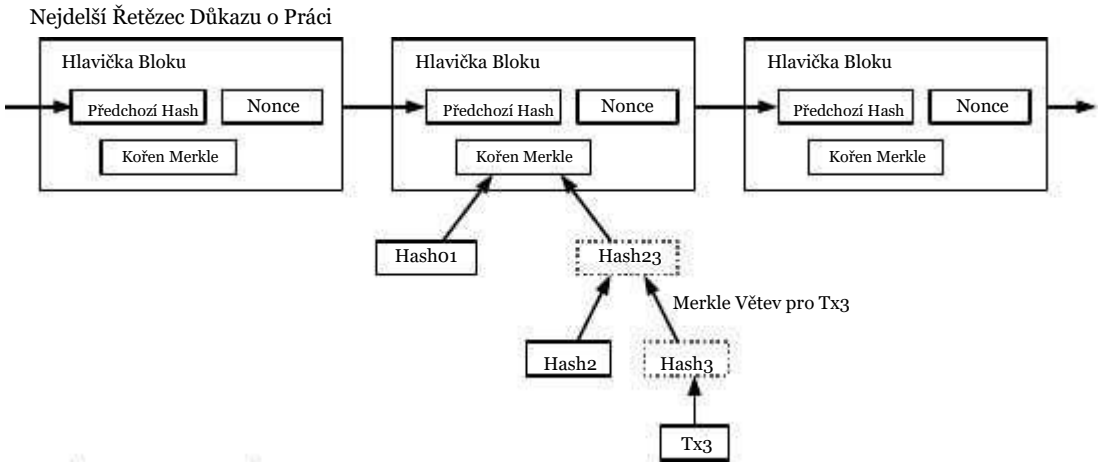
Transakce hashované ve stromu Merkle

Po oříznutí Tx0-2 z bloku

Hlavička bloku bez transakcí by byla asi 80 bajtů. Pokud předpokládáme, že bloky se generují každých 10 minut, 80 bajtů * 6 * 24 * 365 = 4,2MB ročně. S počítačovými systémy, které se od roku 2008 obvykle prodávají s 2GB paměti RAM a podle Moorova zákona předpokládáme, že současný růst bude zhruba 1,2GB ročně. Skladování by tedy nemělo být problém, i kdyby se hlavičky bloků měli uchovávat v paměti.

8. Zjednodušená verifikace plateb

Platby je možné ověřit i bez spuštění uzlu s kompletní historií sítě (full network node). Uživatel potřebuje pouze uschovat kopii hlavičky bloku nejdelšího řetězce důkazů o práci, který může získat dotazováním síťových uzlů, dokud není přesvědčený, že má nejdelší řetězec a získat větev Merkle stromu, která spojuje danou transakci s blokem, ve kterém je časově označená. Nemůže sice sám zkontrolovat danou transakci, ale může vidět její přijetí uzly a umístění v řetězci. Bloky přidané po této transakci dále potvrzují její akceptaci sítí.

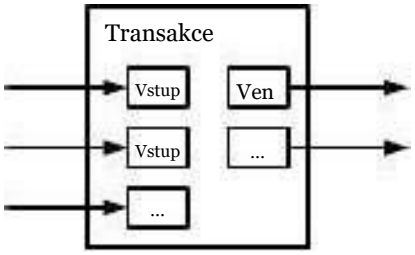


Dokud čestné uzly ovládají síť, je verifikace důvěryhodná, ale je zranitelná pokud síť přemůže útočník. Zatím co síťové uzly ověřují transakce pro sebe, zjednodušená metoda může být oklamána vymyšlenými transakcemi útočníka, do doby, dokud ovládá síť. Jednou ze strategií, jak tomuto bránit je akceptace upozornění od síťových

uzlů, když narazí na neplatný blok, což vyzve software uživatele, aby si stáhl kompletní blok, a upozorní transakce, aby potvrdili nekonzistenci. Firmy, které dostávají časté platby, budou pravděpodobně chtít provozovat své vlastní uzly, kvůli nezávislejšímu bezpeční a rychlejšímu ověřování.

9. Kombinování a rozdělování hodnoty

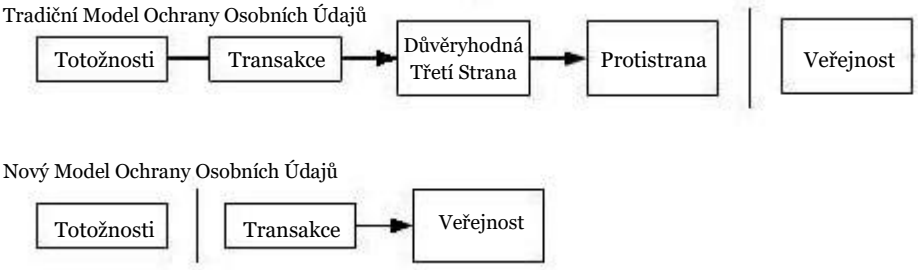
Ačkoli by bylo možné zacházet s mincemi jednotlivě, bylo by nešikovné dělat jednotlivé transakce pro každý jeden cent v převodu. Aby bylo možné rozdělit a kombinovat hodnotu, transakce obsahují vícero vstupů a výstupů. Za normálních okolností bude buď jeden vstup z větší předcházející transakce, nebo vícero vstupů kombinující menší sumy a nejvíc dva výstup: jeden pro platbu a jeden pro případné vrácení zpět odesílateli.



Je třeba poznamenat, že fan-out (počet vstupů, které je možné připojit ke specifickému výstupu), kde jedna transakce závisí na několika jiných transakcích a ty závisí na mnoha dalších, tu není problém. Nikdy tedy není potřebné získat kompletní samostatnou kopii historie transakce.

10. Soukromí

Tradiční bankovní model dosahuje úroveň ochrany soukromí tím, že omezí přístup k informacím zúčastněným stranám a důvěryhodné třetí straně. Nutnost veřejně oznamovat všechny transakce tuto metodu vylučuje, ale soukromí lze stále udržovat uchováním veřejných klíčů v anonymitě. Transakce jsou veřejnosti přístupné, ale bez informací, které by transakci s někým spojovali. Toto je podobné úrovni informací uvolňovaných z burz, kde se zveřejňuje čas a velikost jednotlivých obchodů (the tape/páska), ale bez zveřejnění informací o obchodních stranách.



Na každou transakci by měl být použit nový pár klíčů, jako další ochranná vrstva zabraňující identifikaci původce transakce. Nějaké propojení je stále nevyhnutelné u transakcí s více vstupy, které nutně odhalují, že jejich vstupy byly vlastněny stejným majitelem. Riziko spočívá v tom, že pokud je odhalen vlastník klíče, propojení by mohlo odhalit další transakce, které patří stejné osobě.

11. Výpočty

Zvažujeme scénář, kdy se útočník snaží generovat alternativní řetězec rychleji než čestný řetězec. I kdyby se mu to povedlo, nevytvoří to mezeru v systému, kterou by se dali dělat svévolné změny, jako například vytvořit peníze ze vzduchu, nebo dostat peníze, které útočníkovi nikdy nepatřily. Uzly nepřijmou neplatnou transakci jako platbu a

čestné uzly nikdy neschválí blok, který je obsahuje. Útočník se může pouze pokusit změnit jednu ze svých vlastních transakcí a peníze, které nedávno utratil vzít zpět.

Závod mezi čestným řetězcem a řetězcem útočníka můžeme charakterizovat jako bionomickou náhodnou procházku. Úspěšnou událostí je čestný řetězec, který se prodlužuje o jeden blok, čímž se zvyšuje jeho náskok o +1 a neúspěšnou událostí je řetězec útočníka prodlužující se o jeden blok, čímž se snižuje mezera o -1.

Pravděpodobnost, že se útočník vyrovná s daným deficitem, je obdobná tzv. Gambler's Ruin problému. Předpokládejme, že gambler s neomezeným kreditem začíná v deficitu a hraje potencionálně nekonečný počet her, dokud nedosáhne bodu, kdy není v zisku ani ve ztrátě. Pravděpodobnost, že někdy dosáhne tohoto bodu, nebo že útočník někdy doběhne čestný řetězec, můžeme vypočítat takto [8]:

p = pravděpodobnost, že čestný uzel najde další blok
q = pravděpodobnost, že útočník najde další blok
q_z = pravděpodobnost, že útočník někdy dožene mezeru z bloků

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

Vzhledem na náš předpoklad, že p>q, pravděpodobnost klesá exponenciálně s počtem narůstajících bloků, které musí útočník dohnat. Pokud neučiní velký náhodný skok vpřed hned na začátku, jeho šance pomalu upadá a náskok se zvětšuje. Pravděpodobnost je proto v jeho neprospěch.

Nyní uvažujeme, jak dlouho musí příjemce nové transakce čekat, dokud si nebude dostatečně jistý, že odesílatel nemůže transakci změnit. Předpokládejme, že útočník je odesílatel, který chce, aby příjemce uvěřil, že mu zaplatil, ale aby pak následně vrátil tuto platbu. Příjemce bude upozorněn, když k tomu dojde, ale odesílatel doufá, že bude příliš pozdě.

Příjemce vygeneruje nový pár veřejných klíčů a krátce před podepsáním dá veřejný klíč odesílateli. To zabrání odesílateli v přípravě řetězce bloků, na kterém by nepřetržitě pracoval, dokud by neměl dost štěstí a dostal se dostatečně daleko dopředu, potom by vykonal transakci v ten potřebný moment. Po odeslání transakce začne nečestný odesílatel tajně pracovat na paralelním řetězci, který obsahuje alternativní verzi jeho transakce.

Příjemce čeká, dokud nebude transakce přidána do bloku a *z* bloků se na něj potom napojí. Neví přesné množství pokroku, kterého útočník dosáhl, ale za předpokladu, že čestným uzlům trvá jeden blok průměrný předpokládaný čas, útočníkův pokrok bude Poissonovo rozdělení s předpokládanými hodnotami:

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

Na získání pravděpodobnosti útočníka stále dohnat náskok, vynásobíme hustotu Poissona pro každou možnou úroveň pokroku, kterou mohl dosáhnout, pravděpodobností, kterou má na dostihnutí v tom daném momentu.

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

Přeuspořádání, aby se zabránilo sečtení nekonečného ocasu distribuce.

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

Převod do kódu C ...

```
#include <math.>
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;
}
```

Při pár výpočtech můžeme vidět, že pravděpodobnost klesá exponenciálně se *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
```

Řešení pro P méně než 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. Závěr

Navrhli jsme systém elektronických transakcí bez spoléhání se na důvěru. Začali jsme s obvyklým rámcem mincí vyrobených z digitálních podpisů, který poskytuje silnou kontrolu nad vlastnictvím, ale bez způsobu jak zamezit dvojí útratě (double-spend). Abychom to vyřešili, navrhli jsme síť na bázi rovný s rovným (peer-to-peer), která využívá důkaz o práci (proof-of-work) k záznamu veřejné historie transakcí, která se rychle stane výpočetně nepraktická pro útočníka, pokud čestné uzly disponují větší výpočetní sílou v síti. Síť je robustní ve své nestrukturované jednoduchosti. Všechny uzly fungují najednou s malou koordinací. Nemusí být identifikovány, protože zprávy nejsou směřovány na konkrétní místo a je třeba je pouze dodávat na základě maximálního úsilí (best effort basis). Uzly se mohou libovolně odpojit a znovu se připojit k síti, přičemž řetězec s důkazem o práci (proof-of-work) přijmou jako důkaz toho, co se stalo, v jejich nepřítomnosti. Hlasují svojí výpočetní sílou, vyjadřují svůj souhlas s platnými bloky tím, že pracují na jejich rozšíření a odmítají neplatné bloky tím, že na nich odmítají pracovat. Pomocí tohoto mechanismu konsensu lze vymáhat veškerá potřebná pravidla a pobídky.

Reference

[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. Black, "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.

Poznámka k překladu

Tento překlad byl vytvořen s maximální snahou, ale v žádném případě není dokonalou náhradou původního dokumentu (https://bitcoin.org/bitcoin.pdf). Proto by měl být tento dokument použit s náležitou opatrností a k vytvoření odvozených prací by měl být použit originální dokument.