Oslava 'Mistrovství v bitcoinu'

> "Když mluvím o bitcoinu před běžnými posluchači, občas si kladu otázku "ale jak skutečně funguje?" Nyní mám výbornou odpověď na tuto otázku, protože kdokoliv si přečte *Mistrovství v bitcoinu*, tak do hloubky pochopí, jak to funguje, a bude dobře vybaven pro psaní nové generace úžasných kryptoměnových aplikací."
>
> — Gavin Andresen, vedoucí výzkumu Bitcoin Foundation

> "Bitcoin a technologie blockchainu se stávají základním stavebním kamenem pro novou generaci internetu. Nejlepší a nejbystřejší ze Silicon Vallay na tom pracují. Andreasova kniha vám pomůže se zúčastnit softwarové revoluce ve světě financí.
>
> — Naval Ravikant, spoluzakladatel AngelList

> "*Mistrovství v bitcoinu* je nejlepší dnes dostupnou technickou příručkou o bitcoinu. Na bitcoin bude v budoucnosti nahlíženo jako na nejdůležitější technologii tohoto desetiletí. Z tohoto důvodu je tato kniha absolutní nutností pro všechny vývojáře, speciálně pro ty, kteří se zajímají o tvorbu aplikací pro bitcoinový protokol. Důrazně doporučuji."
>
> — Balaji S. Srinivasan (@balajis), hlavní parner Andreessen Horowitz

> "Objev bitcoinového blockchainu přinesl zcela novou platformu, na které lze dále stavět, která nabídne ekosystém tak rozsáhlý a různorodý jako je samotný internet. Jako jeden z výtečných myslitelů je Andreas Antonopoulos perfektním člověkem pro napsání této knihy."
>
> — Roger Ver, bitcoinový podnikatel a investor

Index

Předmluva

Proč kniha o bitcoinu

Poprvé jsem narazil na bitcoin v polovině roku 2011. Moje bezprostřední reakce byla více méně "Fuj! Peníze pro matfyzáky!" a dalších šest měsíců jsem ho ignoroval aniž bych si uvědomil jeho důležitost. Tyto reakce jsem viděl u mnoha z nejchytřejších lidí, které znám, což mě dává alespoň nějaký pocit útěchy. Podruhé jsem narazil na bitcoin v diskuzi emailové konference. Rozhodl jsem se přečíst základní článek (white paper) sepsaný Satoshi Nakamotem. Prostudoval jsem odpovídající zdrojové kódy a prozkoumal jsem, o čem to celé je. Stále si pamatuji okamžik, když jsem pochopil, že bitcoin není jen jednoduchá digitální měna, ale je to důvěryhodná síť, která poskytuje základnu pro mnohem více něž jsou měny. Zjištění, že "toto nejsou peníze", že je to decentralizovaná důvěryhodná síť, odstartovalo mojí čtyřměsíční cestu, na které jsem sebral každý střípek informace o bitcoinu, který jsem objevil. Byl jsem posedlý a fascinovaný, trávil jsem 12 a více hodin každý den přilepený na monitoru, četl jsem, psal jsem, programoval jsem a učil se jak jsem jen mohl. Když jsem se dostal z tohoto stavu absolutního pohlcení, vážil jsem o deset kilo méně v důsledku nedostatku pravidelné stravy. Byl jsem rozhodnut plně se oddat práci pro bitcoin.

Během následujících dvou let jsem vytvořil mnoho malých projektů za účelem prozkoumání služeb a produktů spojených s bitcoinem. Rozhodl jsem, že je čas napsat moji první knihu. Bitcoin bylo téma, které mě přivádělo do zuřivé kreativity a plně mě pohlcovalo. Byla to nejvíce vzrušující technologie, kterou jsem potkal od vzniku internetu. Nastal čas se podělit o moje nadšení z této úžasné technologie se širším publikem.

Komu je kniha určena

Tato kniha je primárně určena programátorům. Pokud ovládáte programovací jazyk, tato kniha vás naučí jak kryptografické měny fungují, jak je používat a jak vyvíjet programy, které s nimi pracují. Prvních několik kapitol je rovněž vhodných jako hloubkový úvod do bitcoinu pro neprogramátory, které chtějí pochopit vnitřní mechanizmy fungování bitcoinu a kryptoměn.

Konvence použité v této knize

V této knize jsou použity následující typografické konvence.

Kurzíva

Označuje nové pojmy, internetové odkazy, emailové adresy, jména a přípony souborů.

Pevná šířka písma

Je použita pro výpisy zrojových kódů a pro další programové konstrukce, jakou jsou jména proměnných a funkcí, databáze, datové typy, proměnné prostředí, výrazy a klíčová slova

Pevná šířka tučná

Označuje příkazy nebo jiné texty, které uživatel musí opsat doslovně.

Pevná šířka kurzíva

Označuje texty, které musí být nahrazeny uživatelsky definovanými hodnotami nebo hodnotami

vyplývajícími z kontextu.

TIP	Tato ikona označuje tip, návrh nebo obecnou poznámku.
WARNING	Tato ikona označuje varování nebo nebezpečí

Příklady zdrojových kódů

Příklady jsou naprogramovány v Python, C++ a za použití příkazové řádky operačních systémů odvozených z Unixu jako jsou Linux nebo Mac OS X. Všechny ukázky zdrojových kódů jsou dostupné na adrese GitHub repository v podadresáři *code* hlavního úložiště. Pomocí GitHub můžete vytvořit vlastní klon knihy, vyzkoušet zdrojové kódy příkladů nebo nahrát vlastní opravy.

Všechny ukázky zdrojových kódů mohou být zprovozněny na většině operačních systému obsahujících minimální instalace překladačů a interpretů odpovídajících jazyků. V nezbytných situacích uvádíme příkazy nutné pro jednotlivé kroky instalace a příklady výstupu těchto kroků.

Některé z ukázek zdrojových kódů a jejich výstupů byly naformátovány pro tisk. V těchto případech byly některé řádky rozděleny znakem zpětného lomítka (\) následovaným znakem nové řádky. Při překladu příkladu odstraňte tyto dva znaky a znovu spojte řádky, měli byste získat shodné výsledky jako jsou zobrazeny v příkladu.

Všechny ukázky zdrojových kódů používají skutečné hodnoty a výpočty všude, kde je to možné. Můžete překladačem přeložit jednotlivé příklady a porovnávat, zda vámi naprogramované zdrojové kódy vracejí stejné hodnoty. Například soukromé a odpovídající veřejné klíče a adresy jsou skutečné. Všechny příklady transakcí, bloků a odkazů do blockchainu jsou zaznamenány ve skutečném bitcoinovém blockchainu a jsou součástí veřejné účetní knihy. Můžete si je vyhledat v bitcoinovém systému.

Poděkování

Tato kniha odráží úsilí a příspěvky mnoha lidí. Jsem vděčný za všechnu pomoc, kterou jsem obdržel od svých přátel, kolegů a dokonce úplně neznámých lidí, kteří se připojili k mé snaze napsat kompletní technickou knihu o kryptoměnách a bitcoinu.

Je nemožné udělat dělící čáru mezi bitcoinovou technologií a bitcoinovou komunitou. Tato kniha je stejně tak produktem této komunity, jako je to kniha technologie. Moje práce na této knize byla povzbuzována, podporována, oslavována a odměňována celou bitcoinovou komunitou od úplného počátku až do úplného konce. Navíc mne tato kniha umožnila se na dva roky stát částí úžasné komunity. Nedokážu vyjádřit svojí ohromnou vděčnost, za to, že jsem byl přijat do této komunity. Je příliš mnoho lidí na to, abych je všechny jmenoval - lidi, které jsem potkal na konferencích, akcích, seminářích, setkáních, schůzkách u pizzy, malých soukromých schůzkách, těch, kteří se mnou komunikovali přes Twitter, reddit, bitcointalk.org, a GitHub. Všichni tito lidé měli vliv na tuto knihu. Každá myšlenka, analogie, otázka, odpověď a vysvětlení, které najdete v této knize byla v nějakém

aspektu inspirována, testována nebo vylepšena díky mojí spoluprací s komunitou. Děkuji za všechnu podporu, bez které by tato kniha nevznikla. Budu navždy vděčný.

Moje cesta jak se stát autorem započala samozřejmě dlouho před touto první knihou. Můj rodný (a školní) jazyk je řečtina, proto jsem absolvoval doučování z psaní v angličtině během mého prvního roku studia na univerzitě. Rád bych poděkoval Dianě Kordas, mojí první učitelce psaní v angličtině, která mě během toho roku pomohla si vybudovat sebejistotu v této dovednosti. Později, jako profesionál, jsem objevil svojí dovednost psaní technických textů při psaní článků na téma datových center do časopisu *Network World*. Rád bych poděkoval Johnu Dixovi a Johnu Gallantovi, kteří mě dali mojí první novinářskou práci jako sloupkaři v *Network World*, mému editorovi Michaelu Cooneyovi a mým kolegovi Johnovi Tillu Johnsonovi, který editoval mé první sloupky a připravoval je pro publikaci. Napsat 500 slov za týden během čtyř let mě dalo dostatečné zkoušenosti na to, abych uvažoval stát se autorem. Děkuji Jeaně de Vera za její povzbuzování, abych se stal autorem, a a její důvěru a naléhání, abych napsal knihu.

Děkuji také těm, kteří mě podporovali, když jsem zaslal můj návrh knihy nakladatelství O'Reilly. Poskytli mě zpětnou vazbu a oponenturu mého návrhu. Speciálně děkuji Johnu Gallantovi, Gregory Nessovi, Richardu Stiennonovi, Joeli Snyderovi, Adamu B. Levinovi, Sandre Gittlen, Johnovi Dixovi, Johně Till Johnson, Rogerovi Verovi, a Jonu Matonisovi. Speciálně děkuji Richardu Kaganovi a Tymonovi Mattoszkovi, kteří oponovali brzké verze mého návrhu a Matthewovi Owainovi Taylorovi, redigoval můj návrh.

Děkuji Cricketovi Liuovi, autorovi knihy *DNS and BIND* vydané v O'Reilly., který mě představil v O'Reilly. Děkuji také Michaelu Loukidesovi a Allyson MacDonald v O'Reilly, kteří pracovali několik měsíců, aby mě pomohli tuto knihu uskutečnit. Allyson byla neobyčejně trpělivá, když nebyly splněny termíny a výstupy byly opožděny, když život ovlivňoval naplánovaný program.

První náčrt prvních několika kapitol byl nejtěžší, protože bitcoin je složité klubko na rozuzlení. Pokaždé, když jsem se ponořil do jednoho vlákna bitcoinové technologie, musel jsem se ponořit do celé věci. Opakovaně jsem se zasekl a byl trochu sklíčený, když jsem zápasil, abych udělal téma snadno pochopitelné a zároveň abych vytvořil příběh o tomto těžkém technickém tématu. Nakonec jsem se rozhodl vyprávět příběh bitcoinu pomocí příběhů lidí používajících bitcoin a celá kniha se mě začala psát snadněji. Rád bych poděkoval mému příteli a učiteli Richardu Kaganovi, který mě pomohl rozuzlit příběh a pomohl mě se dostat přes chvíle, kdy jsem měl autorský blok. Rád bych poděkoval Pamele Morgan, která oponovala první koncepty každé kapitoly a pokládala těžké otázky, které mě pomohly je vylepšit. Také děkuji vývojáři ze San Francisco Bitcoin Developers Meetup group a Taariqovi Lewisovi, spoluzakladateli skupiny, za pomoc s testováním prvních materiálů.

Během vývoje této knihy jsem vytvořil předběžné koncepty dostupné na GitHubu a nechal jsem je okomentovat veřejností. Jako odpověď jsem obdržel více než stovku komentářů, návrhů, oprav a příspěvků. Těmto přispívatelům jsem jmenovitě poděkoval v Brzký zveřejněný koncept (GitHub přispívatelé). Rád bych poděkoval Minhovi T. Nguyenovi, který dobrovolně organizoval příspěvky na GitHubu a osobně přispěl mnoha významnými příspěvky. Děkuji také Andrewovi Nauglerovi za ikonografický design.

Po vytvoření prvního konceptu následovalo několik kol technické oponentury. Děkuji Cricket Liu and

Lorne Lantz za jejich recenze, komentáře a podporu.

Několik bitcoinových vývojářů přispělo vzorky zdrojových kódů, recenzemi, komentáři a povzbuzováním. Děkuji Amirovi Taakovi a Ericovi Voskuilovi za vzorové úryvky zdrojových kódů a výborné komentáře; Vitaliku Buterinovi and Richardu Kissovi za pomoc s matematikou eliptických křivek a příspěvky zdrojových kódů; Gavinovi Andresenovi za opravy, komentáře a povzbuzování; Michalisovi Kargakisovi za komentáře, příspěvky a zálohování btcd a Robinu Ingemu za zaslání seznamu chyb prvního tisku, díky kterému jsem vylepšil druhý tisk.

Rád by poděkoval své matce za lásku ke slovům a knihám, Theresa, která jsi mě vychovala v domě, který měl knihami vyloženou každou stěnu. Moje matka mě také koupila můj první počítač v roce 1982, přestože se označovala za technický antitalent. Můj tec, Menelaos, stavební inženýr, který právě publikoval svojí první knihu ve věku 80 let, mě naučil logickému a analytickému myšlení a lásce k vědě a inženýrství.

Děkuji za všechnu podporu na této mé cestě.

Brzký zveřejněný koncept (GitHub přispívatelé)

Mnoho přispěvatelů nabídlo komentáře, opravy a doplňky k brzkému zveřejněnému konceptu na GitHubu. Děkuji vám všem přispívatelům této knihy. Následuje seznam významných přispívatelů. včetně jejich identifikátoru na GitHubu uvedeného v závorkách.

- Minh T. Nguyen, organizátor příspěvků GitHubu (enderminh)
- Ed Eykholt (edeykholt)
- Michalis Kargakis (kargakis)
- Erik Wahlström (erikwam)
- Richard Kiss (richardkiss)
- Eric Winchell (winchell)
- Sergej Kotliar (ziggamon)
- Nagaraj Hubli (nagarajhubli)
- ethers
- Alex Waters (alexwaters)
- Mihail Russu (MihailRussu)
- Ish Ot Jr. (ishotjr)
- James Addison (jayaddison)
- Nekomata (nekomata-3)
- Simon de la Rouviere (simondlr)
- Chapman Shoop (belovachap)
- Holger Schinzel (schinzelh)

- effectsToCause (vericoin)
- Stephan Oeste (Emzy)
- Joe Bauers (joebauers)
- Jason Bisterfeldt (jbisterfeldt)
- Ed Leafe (EdLeafe)

Otevřené vydání

Přisouzení

Překlad

Pokud čtete tuto kniho v jiném jazyce než angličtině, byla přeložena dobrovolníky. Následující lidé se podíleli na české verzi tohoto překladu.

- RNDr. Jan Lánský, Ph.D. (zizelevak@gmail.com) - od roku 2015 vyučuje předmět Technologie kryptoměn na oboru Aplikovaná informatika na Vysoké škole finanční a správní
- Poděkování: Tento překlad byl podporován Grantovou agenturou České republiky jako součást projektu: New Sources of Systemic Risk on Financial Markets (GA ČR 16-21506S).
- Korektury: Tomáš Valenta, zde dopište své jméno, pokud jste se podíleli na opravě chyb

Rychlý slovníček

Tento rychlý slovníček obsahuje mnoho termínů použitých ve spojení s bitcoinem. Tyto termíny jsou používány v průběhu celé knihy, proto jsou zde uvedené pro rychlý přehled.

adresa

Bitcoinová adresa vypadá jako 1DSrfJdB2AnWaFNgSbv3MZC2m74996JafV. Skládá se z řetězce písmen a čísel uvozeného číslem jedna "1". Stejně jako požádáte druhé, aby vám zaslali e-mail na vaší e-mailovou adresu, požádáte druhé, aby vám zaslali bitcoin na vaší bitcoinovou adresu.

bip

Návrhy na vylepšení bitcoinu (Bitcoin Improvement Proposals). Množina návrhů, které členové bitcoinové komunity podaly, za účelem vylepšení bitcoinu. Například BIP0021 je návrh na vylepšení bitcoinového jednotného identifikátoru zdroje (URI) scheme.

bitcoin

Jméno jednotky měny (mince), sítě a software.

blok

Skupina transakcí označená časovou značkou a otiskem předchozího bloku. Hlavička bloku je hašovaná pomocí důkazu prací, čímž dojde k ověření transakcí. Platné bloky jsou po odsouhlasení sítí přidány do blockchainu.

blockchain

V překladu řetěz bloků. Seznam platných bloků, každý blok je spojen se svým předchůdcem, tranzitivně se dostaneme do nejstaršího bloku zvaného genesis.

potvrzení

Jakmile je transakce zahrnuta do bloku, získala jedno potvrzení. Jakmile je vytěžen *další* blok, který je přidán do totožného blockchainu, získá transakce druhé potvrzení, atd. Šest a více potvrzení je považováno za dostatečný důkaz, že transakce nemůže být změněna.

obtížnost

Jednotné nastavení v rámci celé sítě, které určuje, kolik výpočtů bude nutno provést na vytvoření důkazu prací.

obtížnostní cíl

Obtížnost, při které všechny výpočty v síti povedou k nalezení bloku přibližně každých 10 minut.

změna obtížnostního cíle

V rámci celé sítě jednotná změna obtížnostního cíle, která nastává jednou za 2106 bloků. Nový obtížnostní cíl je spočten na základě hašovací síly použité v předchozích 2106 blocích.

poplatky

Odesilatel transakce často vkládá poplatek síti za zpracování požadované transakce. Většina transakcí vyžaduje minimální poplatek ve výši 0,5 mBTC

haš

Digitální otisk nějakého binárního vstupu.

základní blok

První blok v blockchainu užívaný k incializaci kryptoměny (v originále genesis)

těžař

Uzel sítě, který hledá platný důkaz prací pro nové bloky pomocí opakovaného hašování.

síť

Peer-to-peer síť, která šíří transakce a bloky mezi všechny bitcoinové uzly na síti.

Důkaz prací V originále Proof-Of-Work. Hondota k jejichž nalezení je třeba značného množství výpočtů. V bitcoinu musejí těžaři hledat numerické řešení SHA256 algoritmu, které splňuje obtížnostní cíl.

odměna

Množství, které je vloženo do každého nového bloku, je odměnou sítě pro těžaře, který našel řešení důkazu prací. V současné době činí 12,5 BTC za blok.

soukromý klíč (privátní klíč)

Tajné číslo, které odemyká bitcoiny zaslané na odpovídající adresu. Soukromý klíč vypadá jako 5J76sF8L5jTtzE96r66Sf8cka9y44wdpJjMwCxR3tzLh3ibVPxh.

transakce

Zjednodušeně řečeno, přenost bitcoinů z jedné adresy na jinou. Přesněji, transakce je podepsaná datová struktura vyjadřující přenášenou hodnotu. Transakce jsou přenášeny přes bitcoinovou síť, sbírány těžaři a vkládány do bloků, které jsou archivovány na blockchainu.

peněženka

Software, který uchovává všechny vaše bitcoinové adresy a soukromé klíče. Používá se k posílání, přijímání a skladování bitcoinů.

Úvod

Co je bitcoin?

Bitcoin je kolekce konceptů a technologií, která vytváří základy ekosystému digitálních peněz. Jednotky měny zvané bitcoiny se využívají pro uchování a převod hodnoty mezi účastníky bitcoinové sítě. Bitcoinoví uživatelé komunikují mezi sebou za použití bitcoinového protokolu nejčastěji po internetu, přestože mohou být využity i jiné přenosové sítě. Implementace bitcoinového protokolu je dostupná jako volně šiřitelný software, může být spouštěna na široké škále výpočetních zařízení, včetně laptopů a chytrých telefonů, což dělá tuto technologii snadno dostupnou.

Uživatelé mohou převádět bitcoiny v síti stejně tak, jako mohu dělat cokoliv, co mohou dělat s klasickými měnami, včetně nákupu a prodeje zboží, zasílání peněz lidem nebo organizacím nebo poskytovat úvěry. Bitcoiny mohou být nakupovány, prodávány a směňovány za jiné měny ve specializovaných směnárnách měn. Bitcoin je v jistém smyslu perfektní formou peněz pro internet, protože je rychlý, bezpečný a neomezený hranicemi.

Narozdíl od tradičních měn, bitcoiny jsou zcela virtuální. Neexistují žádné fyzické mince, dokonce ani žádné digitální mince. Mince jsou odvozeny z transakcí, které přenáší hodnotu mezi odesilatelem a příjemcem. Uživatelé bitcoinu vlastní klíče, které jim umožňují prokázat vlastnictví transakcí v bitcoinové síti a odemknout jim hodnotu, která může být utracena a převedena na nového příjemce. Tyto klíče jsou často uloženy v digitální peněžence na počítači daného uživatele. Vlastnictví klíčů, které odemykají transakci, je jediným nutným předpokladem pro utracení bitcoinů, po kterém je kontrola nad nimi zcela předána do rukou jiného uživatele.

Bitcoin je distribuovaný peer-to-peer systém. Jako takový nemá žádný "centrální" server nebo kontrolní bod. Bitcoiny jsou vytvářeny během procesu zvaného "těžba", který zahrnuje soupeření o najití řešení matematického problému vedoucího ke zpracování bitcoinových transakcí. Každý účastník bitcoinové sítě může fungovat jako těžař, využívat výpočetní výkon svého počítače na ověřování a zaznamenávání transakcí. V průměru každých 10 minut je někdo schopen potvrdit transakce, které proběhly za předchozích 10 minut a je odměněn nově vzniklými bitcoiny. V podstatě těžba bitcoinů decentralizuje distribuci měny a plní funkci centrální banky a nahrazuje potřebu centrální banky celosvětovou hospodářskou soutěží.

Bitcoinový protokol zahrnuje vestavěné algoritmy, které regulují proces těžby v celé síti. Obtížnost zpracování úkolu, který musejí těžaři řešit, aby úspěšně zaznamenali blok transakcí do bitcoinové účetní knihy, je dynamicky upravována tak, aby v průměru každých 10 minut někdo tohoto úspěchu dosáhl bez ohledu na to, kolik těžařů (a počítačů) pracuje na tomto úkolu v daný čas. Protokol také každé čtyři roky snižuje na polovinu rychlost, kterou jsou vytvářeny nové bitcoiny a omezuje tak celkový počet bitcoinů, které budou vytvořeny na pevnou hodnotu 21 milionu mincí. Výsledkem je, že počet bitcoinů v oběhu těsně sleduje snadno predikovatelnou křivku, která dosáhne hodnoty 21 milionu do roku 2140. Vzhledem ke snižující se rychlosti emise nových mincí je z dlouhodobého hlediska bitcoin deflační měnou. Mimoto, nelze vyvolat inflaci bitcoinu pomocí "tisku" nových peněz nad rámec očekávané rychlosti emise.

V zákulisí je bitcoin také jméno protokolu, sítě a inovací v distribuovaných výpočtech. Měna bitcoin je pouze první skutečnou aplikací tohoto objevu. Jako vývojář na bitcoin nahlížím jako na internet peněz, síť pro šíření hodnoty a zabezpečení vlastnictví digitálních aktiv prostřednictvím distribuovaných výpočtů. Bitcoin je více, než se na první pohled zdá.

V této kapitole začneme vysvětlením některých z hlavních konceptů a pojmů, získáme nezbytný software a použijeme bitcoin pro jednoduché transakce. V následujících kapitolách začneme rozbalovat jednotlivé vrstvy technologie, která umožňuje existenci bitcoinu a prozkoumáme vnitřní fungování bitcoinové sítě a protokolu.

Digitální měny před bitcoinem

Vznik životaschopných digitálních peněz je úzce spojen s rozvojem kryptografie. Toto není překvapující, když si uvědomíme, že základní výzvou je, jak použít bity pro reprezentaci hodnoty, která může být vyměněna za zboží a služby. Kdokoliv přijímá digitální peníze si musí položit následující dvě otázky.

1. Mohu věřit, že měna je autentická a není padělána?
2. Mohu si být jistý, že nikdo jiný nemůže prohlásit, že tyto peníze náleží jemu a nikoliv mě?(viz problém “dvojitého utracení”)

Vydavatelé papírových peněz neustále bojují s problémem padělatelství pomocí stále propracovanějších způsobů výroby papíru a metod tisku. Fyzické peníze nemusejí řešit problematiku dvojitého utrácení, protože ten samý papír nemůže být v jednu chvíli na dvou místech. Samozřejmě, klasické peníze bývají často uloženy a přenášeny digitálně. V těchto případech problematika padělání a dvojitého utrácení je vyřešena tím, že všechny elektronické transakce vypořádají centrální autority, které mají globální přehled o měně v oběhu. Digitální měny nemohou využívat výhod neviditelného inkoustu nebo holografických skriptů. Kryptografie poskytuje základy pro důvěru v legitimitu uživatelem vlastněné hodnoty. Speciálně, kryptografické digitální podpisy umožňují uživateli podepsat digitální aktiva nebo transakce prokazující vlastnictví těchto aktiv. Pomocí vhodné architektury, digitální podpisy mohou být použity k vyřešení problému dvojité útraty.

Když kryptografie začala být šířeji dostupná a srozumitelná koncem 80. let minulého století, mnoho výzkumníků se začalo pokoušet využít kryptografii pro vytvoření digitálních měn. Tyto první projekty digitálních měn vydávaly digitální peníze obvykle kryté státní měnou nebo drahým kovem jako zlato.

Přestože tyto první digitální peníze fungovaly, byly centralizované a ve výsledku byly snadno zranitelné útoky ze strany vlád nebo hackerů. Brzké digitální měny používaly centrální zúčtovací středisko, které provádělo všechny transakce v pravidelných časových intervalech, obdobně jako pracuje klasický bankovní systém. Bohužel v mnoha případech, byly tyto rodící se digitální měny napadeny vládami, které z nic měly obavy, a které nakonec vynutily jejich zánik. Některé z nich zanikly při okázalých pádech, když jejich mateřská firma byla náhle zrušena. Pro odolání proti zásahům protivníku, ať již legálních vlád nebo kriminálních živlů, je třeba

decentralizované digitální měny, tím se vyhneme útoku zaměřenému na centrální kontrolní bod. Bitcoin je takovýto systém, navržený kompletně decentralizovaně, nezávislý na jakékoliv centrální autoritě nebo centrálním kontrolním bodu, který může být napaden nebo poškozen.

Bitcoin reprezentuje vrchol desítky let trvajícího výzkumu kryptografie a distribuovaných systémů a zahrnuje čtyři klíčové inovace spojené dohromady v jedinečnou a mocnou kombinaci. Bitcoin se skládá z:

- Decentralizovaná peer-to-peer síť (bitcoin protokol)
- Veřejná účetní kniha transakcí (blockchain)
- Decentralizovaná matematická a deterministická emise měny (distribuovaná těžba)
- Decentralizovaný systém ověřování transakcí (transakční skript)

Historie bitcoinu

Bitcoin byl představen v roce 2008 v článku pojmenovaném "Bitcoin: peer-to-peer elektronický pokladní systém," podepsaném přezdívkou Satoshi Nakamoto. Nakamoto zkombinoval několik předchozích vynálezů jako b-money a HashCash, aby vytvořil kompletně decentralizovaný platební systém, který se nespoléhá na centrální autoritu pro emitování peněžní zásoby nebo vypořádání a ověřování transakcí. Hlavní inovací bylo vytvoření distribuovaného výpočetního systému (nazvaného algoritmus důkazu prací) řídícího globální "volbu" probíhající každých 10 minut, která umožňuje decentralizované síti dosáhnout *koncenzu* o stavu transakcí. Tím je elegantně vyřešen problém dvojité útraty, při kterém může být jedna jednotka měny utracena dvakrát. Přesněji problém dvojité útraty byl slabinou digitálních měn a do té doby řešen vypořádáním transakcí pomocí centrálního zúčtovacího střediska.

Bitcoinová síť začala fungovat v roce 2009 za pomoci referenční implementace publikované Nakamotou a následně korigované množstvím jiných programátorů. Složitost distribuovaných výpočtů zajišťujících bezpečnost a odolnost bitcoinu narůstá exponenciální řadou a nyní přesahuje sečtenou výpočetní kapacitu nejlepších světových superpočítačů. Celková tržní kapitalizace bitcoinu se pohybuje okolo 10 miliard amerických dolarů (cca 250 miliard českých korun), v závislosti na směnném kurzu bitcoinu a dolaru (a české koruny). Nejvyšší hodnota jedné provedené transakce byla 150 milionu amerických dolarů (téměř 4 miliardy českých korun). Tato transakce proběhla okamžitě a bez jakýchkoliv poplatků.

Satoshi Nakamoto se přestal veřejně projevovat v dubnu roku 2011, zanechal zodpovědnost za vývoj zdrojových kódů a sítě prosperující skupině dobrovolníků. Identita osoby nebo skupiny osob stojících za bitcoinem je stále neznámá. Nicméně Satoshi Nakamoto ani nikdo jiný nevykonává kontrolu nad bitcoinovým systémem, který funguje zcela transparentně na matematických základech. Samotný objev je průkopnický a vytvořil nové vědní oblasti v oborech distribuovaných výpočtů, ekonomie a ekonometrie.

Řešení problému distribuovaného výpočtu

Objev Satoshi Nakamota je rovněž praktickým řešením do té doby nevyřešeného problému distribuovaného výpočtu, známého jako "Problém byzantských generálů." Stručně, problém se zabývá pokusem najít shodu na dalším postupu prostřednictvím výměny informací přes nespolehlivou a potenciálně narušenou síť. Řešení Satoshi Nakamota využívá koncept důkazu prací, aby dosáhlo shody bez použití důvěryhodné centrální autority, Toto řešení přineslo průlom ve výzkumu distribuovaných výpočtů a má široké použití mimo kryptoměny. Může být použito k dosažení shody v decentralizované síti, která dokazuje poctivost voleb, loterií, evidence aktiv, digitálních notářů atd.

Použití bitcoinu, uživatelé a jejich příběhy

Bitcoin je technologie, ale vyjadřuje peníze, které jsou základním jazykem pro výměnu hodnoty mezi lidmi. Podívejme se na lidi, kteří používají bitcoin a na některé nejobvyklejší způsoby použití měny a protokolu demonstrované na příbězích těchto lidí. Tyto příběhy budeme znovu využívat v průběhu celé knihy za účelem vykreslení použití digitálních peněz v reálném světě a jak toto použití zpřístupňují různé technologie, které jsou součástí bitcoinu.

Levné nákupy v severní Americe

Alice žije v severní Kalifornii v San Francisco Bay Area. Slyšela o bitcoinu od svých technicky zdatných přátel a chce ho začít používat. Budeme sledovat její příběh, jak se učila o bitcoinu, získala nějaké bitcoiny a utratila je za šálek kafe v Bobově kavárně v Palo Altu. Tento příběh nám představí software, směnárnu a základy transakcí z pohledu maloobchodního zákazníka.

Drahé nákupy v severní Americe

Carol vlastní uměleckou galerii v San Francisku. Prodává drahé obrazy za bitcoiny. Tento příběh nám představí riziko 51 % organizovaného útoku pro obchodníky s drahými předměty.

Zahraniční poskytoval služeb

Bok, vlastník kavárny v Palo Altu chce vytvořit nové webové stránky. Uzavřel smlouvu s Gopeshem webovým vývojářem, který žije v Bengalúru v Indii. Gopesh souhlasil, že mu bude zaplaceno v bitcoinech. Tento příběh prozkoumá využití bitcoinu pro outsourcing, smluvní dodavatele a mezinárodní bankovní převody.

Příspěvky na dobročinnost

Eugenia je ředitelkou dětské nadace na Filipínách. Nedávno objevila bitcoin a chtěla by ho použít při získávání příspěvků pro její nadaci od zcela nové skupiny zahraničních a domácích dárců. Zároveň prozkoumává možnosti, jak použít bitcoin pro rychlou distribuci prostředků do oblastí kde jsou potřeba. Tento příběh ukáže použití bitcoinu pro celosvětové získávání finančních prostředků bez omezení měnami a hranicemi a použití otevřené účetní knihy pro kontrolu transparentnosti hospodaření charitativních organizací.

Dovoz a vývoz

Mohammed je dovozce elektroniky v Dubaji. Pokouší se použít bitcoin pro nákup elektroniky v USA a Číně určené na import do Spojených arabských emirátů. Jeho motivací je zrychlení plateb za dovoz. Tento příběh ukáže jak může být bitcoin využitý obchodními společnostmi pro mezinárodní platby za fyzické zboží.

Těžba bitcoinu

Jing je student počítačového inženýrství v Šanghaji. Vytvořil "těžební" soupravu pro těžbu bitcoinů, využívá svých inženýrských dovedností pro zvýšení svého příjmu. Tento příběh vysvětlí "průmyslové" základy bitcoinu: specializované vybavení použité pro zabezpečení bitcoinové sítě a emitování nových jednotek měny.

Každý z těchto příběhů je založen na skutečných lidech a skutečném průmyslu, který v současné době využívá bitcoin pro vytváření nových trhů, nového průmyslu a inovativních řešeních problémů celosvětové ekonomiky.

Začínáme

Pro připojení do bitcoinové sítě a začátek používání měny si musí každý uživatel stáhnout aplikaci nebo použít webovou aplikaci. Protože bitcoin je standard, existuje mnoho implementací softwarových bitcoinových klientů. Existuje také referenční implementace, známá pod názvem Satoshiho klient, která je spravována jako open source projekt týmem vývojářů a je odvozena z původní implementace vytvořené Satoshi Nakamotem.

Tři hlavní druhy bitcoinových klientů jsou:

Úplný klient

Úplný klient nebo "úplný uzel" je klient, který uchovává celou historii bitcoinových transakcí (úplně všechny transakce všech uživatelů), spravuje uživatelské peněženky a může zahájit proces vytvoření transakce v bitcoinové síti. Jedná se o obdobu samostatného e-mailového serveru, který se umí vypořádat se všemi aspekty protokolu bez spoléhání se na jiný server nebo službu třetí strany.

Odlehčený klient

Odlehčený klient uchovává peněženky uživatele, ale spoléhá se na servery vlastněné třetí stranou, které využívá pro přístup k bitcoinovým transakcím a síti. Odlehčený klient neuchovává úplnou kopii všech transakcí a proto musí věřit serverům vlastněným třetí stranou při ověřování transakcí. Je to obdoba samostatného e-mailového klienta, který se připojuje na poštovní server pro přístup do e-mailové schránky, a který spoléhá na třetí stranu při interakci se sítí.

Webový klient

Webový klient je dostupný přes webový prohlížeč a uchovává uživatelské peněženky na serveru vlastněném třetí stranou. Je to obdoba webového e-mailového klienta, který kompletně spoléhá na server třetí strany.

Mobilní bitcoin

Mobilní klienti pro chytré telefony, které jsou založeny na operačním systému Android, mohou snadno působit jako úplný klient, odlehčený klient nebo webový klient. Někteří mobilní klienti jsou synchronizováni s webovým nebo stolním klientem, poskytující multiplatformní peněženku pro mnoho zařízení pracujících s jednotným zdrojem finančních prostředků.

Volba bitcoinového klienta závisí na tom, jak velkou kontrolu chce mít uživatel nad svými finančními prostředky. Úplný klient nabídne nejvyšší míru kontroly a nezávislosti uživatele, ale zatíží uživatelem břemenem provádění záloh a zabezpečení. Na druhém konci spektra možností je webový klient, který je nejsnazší pro použití, což je vykoupeno bezpečnostním rizikem. Uživatel je závislý na zabezpečení majitele a provozovatele webové služby. Pokud je prolomeno zabezpečení služby webové peněženky, jak se již mnohokrát stalo, uživatelé mohou ztratit všechny své finanční prostředky. Naopak pokud má uživatel úplného klienta bez použití vhodného zálohování, může ztratit své finanční prostředky při selhání počítače.

Pro účely této knihy představíme použití různých stáhnutelných bitcoinových klientů, od referenční implementace (Satoshiho klient) po webové peněženky. Některé příklady požadují použití referenčního klienta, který navíc oproti úplnému klientovi také zpřístupňuje programové rozhraní pro peněženky, síť služby provádějící transakce. Pokud plánujete prozkoumat programové rozhraní bitcoinového systému, budete potřebovat referenčního klienta.

Rychlý start

Alice, kterou jsme si představili v Použití bitcoinu, uživatelé a jejich příběhy, není technicky zdatný uživatel a pouze nedávno slyšela o bitcoinu od kamaráda. Započala svojí cestu návštěvou oficiální webové stránky bitcoin.org, kde nalezla širokou nabídku bitcoinových klientů. Postupovala dle rad webové stránky bitcoin.org a vybrala si odlehčeného bitcoinového klienta Multibit.

Alice následuje odkaz ze stránky bitcoin.org na stažení a instaluje Multibit na svůj stolní počítač. Multibit je dostupný pro stolní počítače s operačními systémy Windows, Mac OS, a Linux.

Bitcoinová peněženka musí byt zabezpečena heslem nebo přístupovou frází. Mnoho ošklivých lidí se pokouší prolomit slabá hesla jiných uživatelů, proto pečlivě vybírejte taková hesla, která nemohou být snadno prolomena. Používejte kombinaci velkých a malých písmen, čísel a symbolů. Vyhněte se použití osobních informací jakou jsou data narození nebo jména sportovních týmů. Vyhněte se použití slov běžně se vyskytujících ve slovnících v jakémkoliv jazyce. Pokud je to možné, použijte generátor hesel na vytvoření zcela náhodného hesla majícího délku alespoň 12 znaků. Pamatuje, bitcoin jsou peníze a mohou být okamžitě převedeny kamkoliv na světě. Pokud je dobře nechráníte, můžou být snadno ukradeny.

Alice si stáhla a nainstalovala Multibit aplikaci, spustila ji a byla pozdravena úvodní obrazovkou, jak je znázorněno na Úvodní obrazovka bitcoinového klienta Multibit.

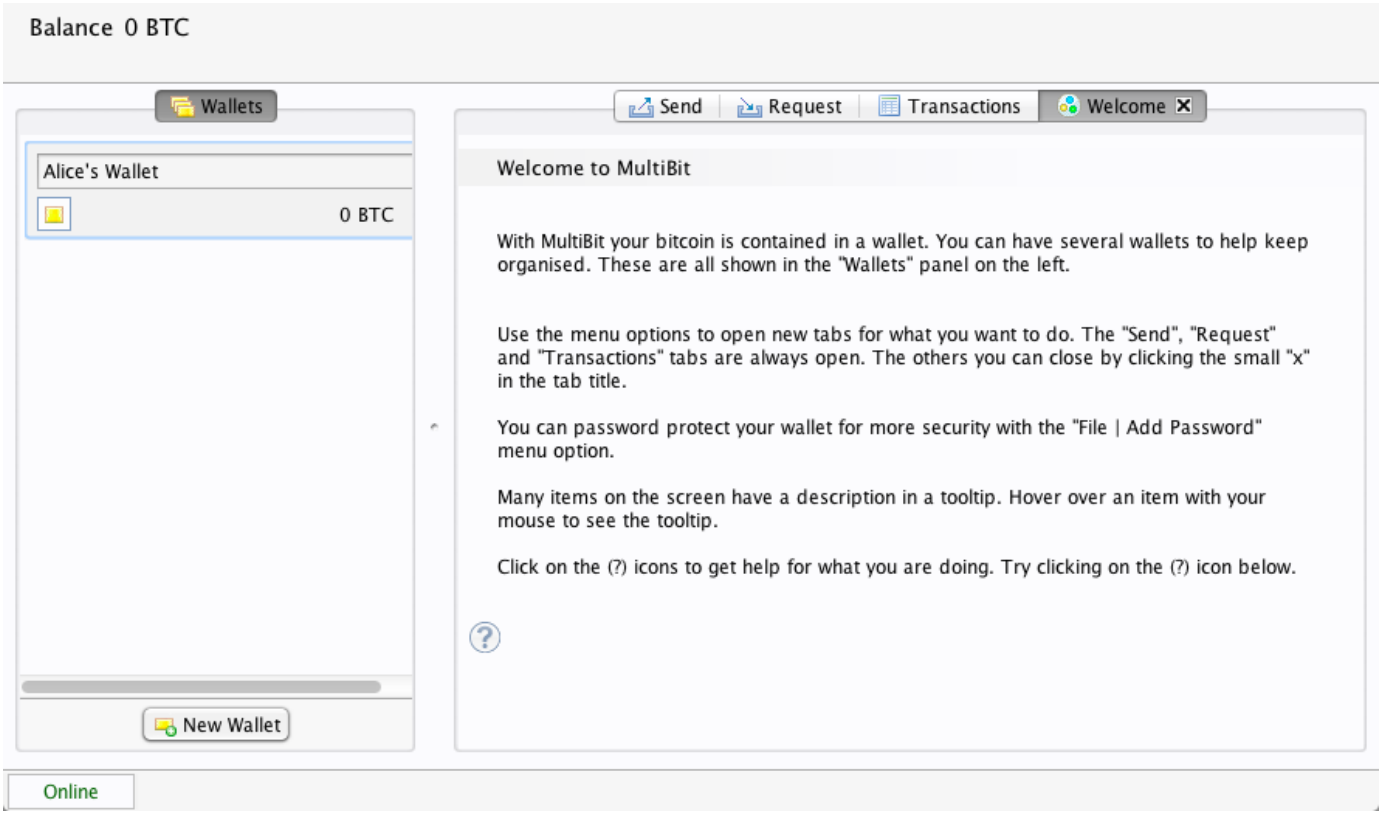


Figure 1. Úvodní obrazovka bitcoinového klienta Multibit

Multibit automaticky vytvoří peněženku a novou bitcoinovou adresu pro Alici, kterou může Alice zobrazit kliknutím na záložku Request, jak je znázorněno na Request Nová bitcoinová adresa pro Alici na záložce Request v klientu Multibit.

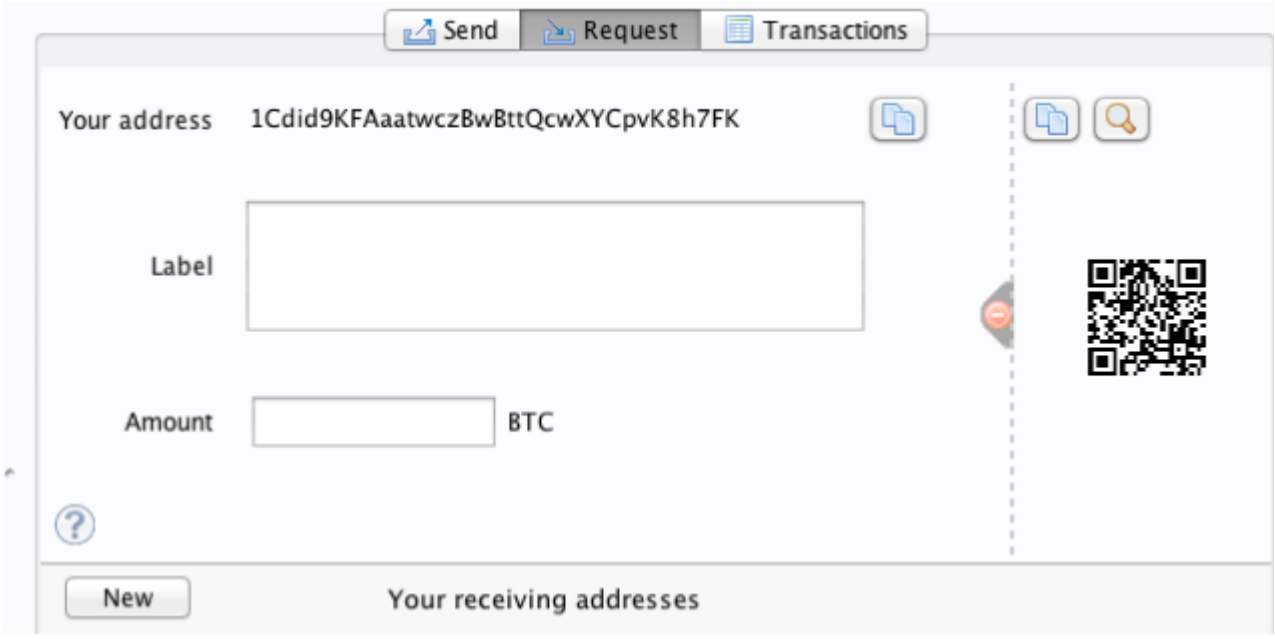


Figure 2. Nová bitcoinová adresa pro Alici na záložce Request v klientu Multibit

Nejdůležitější částí této obrazovky je *bitcoinová adresa* Alice. Obdobně jako e-mailovou adresu, může Alice zveřejnit tuto adresu a kdokoliv ji může použít pro zaslání peněz do její nové peněženky. Na

obrazovce se zobrazuje dlouhý řetězec písem a číslic: 1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK. Vedle bitcoinové adresy peněženky je QR kód, druh čárového kódu, který obsahuje stejné informace ve formátu, který může být nasnímán kamerou chytrého telefonu. QR kód je černobílý čtverec na pravé straně obrazovky. Alice může okopírovat bitcoinovou adresu nebo QR kód do schránky kliknutím na tlačítko copy nacházející se vedle nich. Po kliknutí na QR kód dojde k jeho zvětšení, takže může být snadno naskenován kamerou chytrého telefonu.

Alice může také QR kód vytisknut a takto snadno předávat svojí adresu druhým lidem bez nutnosti přepisování dlouhého řetězce písmen a číslic.

TIP

Bitcoinová adresa začíná číslicí 1 nebo 3. Obdobně jako e-mailová adresa, může být zveřejněna ostatním uživatelům bitcoinu, kteří ji mohou použít pro zaslání bitcoinů přímo do peněženky. Na rozdíl od e-mailové adresy, můžete vytvořit novou bitcoinovou adresu tak často, jak chcete. Všechny tyto adresy budou směřovat finanční prostředky do vaší peněženky. Peněženka je jednoduše řečeno kolekce adres a k ním příslušných klíčů, které odemykají finanční prostředky na těchto adresách. Můžete zvýšit svoje soukromí používáním nové adresy pro každou transakci. Neexistuje prakticky žádný limit počtu adres, které může uživatel vytvořit.

Alice je nyní připravena začít používat svojí novou bitcoinovou peněženku

Získání prvních bitcoinů

V současné době není možné bitcoiny koupit v bance nebo v kamenné směnárně. V mnoha zemích je nyní v roce 2014 stále těžké získat bitcoiny. Existuje mnoho specializovaných směnáren kde lze koupit a prodat bitcoiny výměnou za místní měnu. Tyto směnárny fungují jako webové aplikace, patří mezi ně:

Bitstamp

Evropská směnárna, která podporuje několik měn včetně eura (EUR) a amerického dolaru (USD), podporuje bankovní převody.

Coinbase

Americká (USA) bitcoinová peněženka a platforma, kde můžou obchodníci a zákazníci provádět transakce s bitcoiny. Coinbase umožňuje snadno nakupovat a prodávat bitcoiny, umožňuje napojit americké účty prostřednictvím automatického zúčtovacího systému (AHC).

Kryptoměnové směnárny ze své podstaty fungují na průniku státních měn a kryptoměn. Z tohoto důvodu jsou předmětem národních a mezinárodních regulací. Často jsou specifické pro jednu zemi nebo ekonomickou oblast a specializují se na národní státní měny této oblasti. Váš výběr směnárny bude specifický pro státní měnu, kterou používáte a bude omezen směnárnami, které fungují v právní jurisdikci vaší země. Podobně jako otevření bankovního účtu, zpřístupnění služeb pro váš účet trvá několik dní nebo týdnů, protože vyžadují různé formy identifikace, aby splnili bankovní regulace KYC (poznej svého zákazníka) a AML (proti praní špinavých peněz). Jakmile máte účet na bitcoinové burze, můžete nakupovat a prodávat bitcoiny tak rychle, jako byste nakupoval zahraniční státní měnu na

makléřském účtu.

Obsáhlejší seznam můžete nalézt na bitcoin charts, tato stránka nabízí aktuální ceny a další data z mnoha desítek směnáren.

Noví uživatelé mohou získat bitcoiny následujícími čtyřmi způsoby:

- Najít přítele, který má bitcoiny a nějaké od něj přímo koupit. Mnoho uživatelů bitcoinů začalo tímto způsobem.
- Použít utajenou službu jako localbitcoins.com k najití prodejce ve vašem okolí a koupit bitcoiny za hotovost při osobním nákupu.
- Prodat zboží nebo službu za bitcoiny. Pokud jste programátor, prodejte své programovací schopnosti.
- Použít bitcoinový bankomat ve vašem městě. Najděte bankomat, který máte blízko na online mapě CoinDesk.

Alici představil bitcoin její kamarád, tak měla snadnou cestu jak získat své první bitcoiny, zatímco čekala na ověření a aktivaci svého účtu ve směnárně v Kalifornii.

Posílání a přijímání bitcoinů

Alice si vytvořila vlastní bitcoinovou peněženku a je připravena přijímat finanční prostředky. Její peněženková aplikace náhodně vygenerovala soukromý klíč (podrobněji popsáno v [private_keys]) společně s odpovídající bitcoinovou adresou. V tomto okamžiku její bitcoinová peněženka není známa bitcoinové síti ani není "registrovaná" v žádné části bitcoinového systému. Její bitcoinová adresa je pouze číslo, které odpovídá klíči, který může použít pro přístup k finančním prostředkům. Neexistuje žádný účet nebo spojení mezi adresou a nějakým účtem. Do okamžiku, kdy je tato adresa uvedena jako příjemce hodnoty v transakci zaslané do bitcoinového účetního systému (blockchainu), je to pouze část z téměř nekonečného množství možných adres, které jsou "platné" v bitcoinové síti. Jakmile je jednou tato adresa spojená s transakcí, stává se součástí známých adres v síti a Alice si může zkontrolovat stav finančních prostředků ve veřejném účetním systému.

Alice se setkává se svým kamarádem Joem, který ji seznámil s bitcoinem, v místní restauraci. Vymění si nějaké dolary za nějaké bitcoiny, které jsou převedeny na její účet. Přinesla si vytištěnou svoji bitcoinovou adresu a QR kód, jak byl zobrazen v její bitcoinové peněžence. Informace o bitcoinové adrese nejsou z bezpečnostního hlediska vůbec důvěrné. Mohou být uveřejněny kdekoliv bez bezpečnostního rizika pro její účet.

Alice chce směnit 10 dolarů za bitcoiny, nechce riskovat příliš mnoho peněz v této nové technologii. Dala Joeovi desetidolarovou bankovku a svoji vytištěnou adresu. Joe jí nyní může zaslat této částce odpovídající množství bitcoinů.

Dále, Joe si musel zjistit směnný kurz, aby mohl dát Alici správné množství bitcoinů. Existují stovky aplikací a webových stránek, které poskytují aktuální tržní kurz. Uvedeme si několik nejpopulárnějších:

Bitcoin Charts

Služba poskytuje tržní data, zobrazuje kurz bitcoinu na mnoha směnárnách na zeměkouli vyjádřený v různých místních měnách.

Bitcoin Average

Stránka nabízí jednoduchý přehled průměrných (objemem obchodů vážených) kurzů pro jednotlivé měny

ZeroBlock

Bezplatné aplikace pro Android a iOS, která zobrazuje cenu bitcoinu z různých směnáren (viz ZeroBlock, aplikace pro Android a iOS zobrazující tržní kurz)

Bitcoin Wisdom

Další služba poskytující tržní data

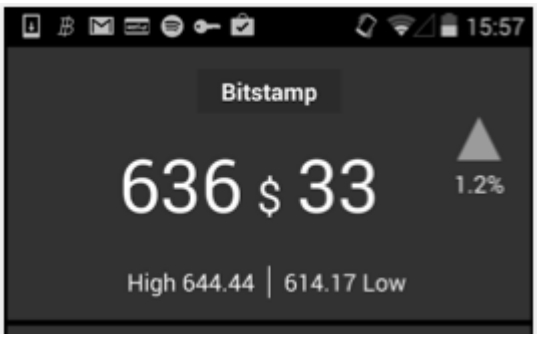


Figure 3. ZeroBlock, aplikace pro Android a iOS zobrazující tržní kurz

Za použití jedné z aplikací nebo webových stánek, které byly právě uvedeny, Joe určil cenu bitcoinu na přibližně, 100 amerických dolarů za bitcoin. V tomto kurzu by Alice měla dostat 0,10 bitcoinu neboli 100 milibitů výměnou za 10 dolarů, které mu dala.

Jakmile Joe určil spravedlivou výměnnou cenu, otevřel svojí mobilní peněženkovou aplikaci a vybral "zaslat" bitcoin. Například pokud by použil Blockchain mobilní peněženku na Androidu, viděl by obrazovku požadující dva vstupy, jak je zobrazeno na Blockchain mobilní bitcoinová peněženka - odesílání.

- Cílová bitcoinová adresa transakce
- Množství zasílaných bitcoinů

Ve vstupním políčku pro bitcoinovou adresu je malá ikona, která vypadá jako QR kód. Toto umožní Joeovi naskenovat čárový kód kamerou svého chytrého telefonu, takže nemusel opisovat Aličinu bitcoinovou adresu (1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK), která je docela dlouhá a těžko se opisuje. Joe klepá na ikonu QR kódu a aktivuje kameru chytrého telefonu, scanuje QR kód Aličiny vytisknuté peněženky, kterou mu přinesla. Mobilní peněženková aplikace vyplní bitcoinovou adresu a Joe může zkontrolovat, že byla oskenována správně porovnáním několika číslic z této adresy s adresou vytištěnou Alicí.

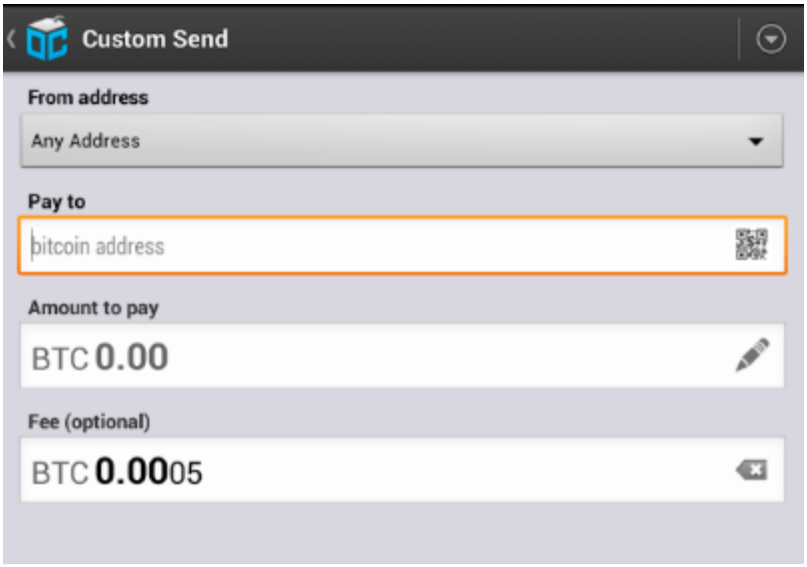


Figure 4. Blockchain mobilní bitcoinová peněženka - odesílání

Joe poté vložil bitcoinovou hodnotu transakce, 0,10 bitcoinu. Pečlivě zkontroloval, zda vložil správné množství, protože se připravuje převést peníze a jakákoliv chyba může být drahá. Nakonec stiskl "Send" k přenosu transakce. Joeova mobilní bitcoinová peněženka vytvořila transakci, která přiřazuje 0,10 bitcoinu adrese poskytnuté Alicí, zdrojem těchto finančních prostředků je Joeova peněženka podepisuje transakci Joeovými soukromými klíči. Toto říká bitcoinové síti, že Joe potvrdil přenos hodnoty z jedné z jeho adres na novou adresu Alice. Jakmile je transakce přenesena pomocí peer-to-peer protokolu, začíná se rychle šířit v celé bitcoinové síti. Za méně než sekundu, většina z uzlů sítě s dobrým připojením obdrží transakci a poprvé vidí adresu Alice.

Pokud má Alice s sebou chytrý telefon nebo laptop, může si transakci prohlédnout. Bitcoinový účetní systém je neustále rostoucí soubor, který zaznamenává každou transakci, která se kdy objevila. Tento systém je veřejný, Alici proto stačí si vyhledat svojí vlastní adresu a podívat se, zda finanční prostředky byly odeslány. Může se snadno podívat na webovou stránku blockchain.info napsáním této adresy ve vyhledávacím řádku svého internetového prohlížeče. Webová stránka ji ukáže přehled všech transakcí z a na její adresu. Pokud se Alice dívá na tuto stránku, aktualizuje se a ukáže novou transakci, která zvyšuje stav jejího účtu o 0,10 bitcoinu, brzy poté, co Joe stiskl "Send".

Potvrzení

Nejprve, adresa Alice ukazuje transakci od Joa jako "Nepotvrzenou". To znamená, že transakce byla rozšířena po síti, ale ještě jí nikdo nezahrnul do bitcoinového účetního systému, známého jako blockchain. Pro vložení je nutné, aby transakce byla "vybrána" těžařem a vložena do bloku transakcí. Jakmile je blok vytvořen, přibližně za 10 minut, transakce v tomto bloku budou sítí přijaty jako "potvrzená" a mohou být utraceny. Transakce vidí všichni okamžitě, ale jsou "důvěryhodné" pro ostatní pouze tehdy, pokud jsou vloženy do nově vytvořeného bloku.

Alice je nyní hrdou majitelkou 0,10 bitcoinů, které může utratit. V další kapitole se podíváme na její první nákup za bitcoiny a detailněji prozkoumáme podkladovou transakci a technologii její šíření.

Jak bitcoin funguje

Transakce, bloky, těžba a blockchain

Bitcoinový systém, na rozdíl od klasických bankovních a platebních systémů, je založen na decentralizované důvěře. Místo centrální důvěryhodné autority, se v bitcoinu dosahuje důvěry jako vznikající vlastnosti při interakci různých účastníků bitcoinového systémů. V této kapitole prozkoumáme bitcoin vysokoúrovňově stopováním jedné transakce skrz bitcoinový systém a budeme sledovat, jak se stává "důvěryhodnou" a je akceptována bitcoinovým mechanizmem distribuované shody a jak je nakonec zaznamenána v blockchainu, distribuovaném účetním systému všech transakcí.

Každý příklad je založen na skutečné transakci proběhlé v bitcoinové síti, simulující interakci mezi uživateli (Joe, Alice a Bob) zasíláním finančních prostředků z jedné peněženky do druhé. Při stopování transakce skrz bitcoinovou síť a blockchain použijeme webovou stránku *blockchain průzkumník*, která vizualizuje jednotlivé kroky. Blockchain průzkumník je webová aplikace, která pracuje jako bitcoinový vyhledávač, umožňuje hledat adresy, transakce a bloky a zobrazuje vztahy a toky mezi nimi.

Populární blockchainoví průzkumníci jsou:

- Blockchain info
- Bitcoin Block Explorer
- insight
- blockr Block Reader

Každý z nich má funkci hledání, která jako vstup přijímá adresu, haš transakce nebo číslo bloku, a najde příslušná data na bitcoinové síti a v blockchainu. U každého příkladu uvedeme i webovou adresu, která nám přímo zobrazí odpovídající záznam, který budeme studovat detailněji.

Přehled bitcoinového systému

V přehledovém diagramu Přehled bitcoinového systému vidíme bitcoinový systém skládající se z uživatelů s jejich peněženkami obsahujícími klíče, transakcí propagovaných skrz síť a těžařů vytvářejících (skrz výpočetní soutěžení) shodu na stavu blockchainu, který je spolehlivým účetním systémem obsahujícím všechny transakce. V této kapitole budeme stopovat jednu transakci na její cestě sítí a prozkoumáme vysokoúrovňovou spolupráci mezi jednotlivými částmi bitcoinového systému. Následující kapitoly se ponoří do technologií, stojícími za peněženkami, těžbou a obchodními systémy.

Figure 1. Přehled bitcoinového systému

Kupování šálku kávy

Alice, představená v předchozí kapitole, je novým uživatelem, který právě získal své první bitcoiny. Transakce, kterou vytvořil Joe, naplnila Alicinu peněženku finančními prostředky ve výši 0,10 BTC. Nyní Alice udělá svojí první spotřebitelskou transakci, koupí si šálek kávy v Bobově kavárně v Palo Alto, v Kalifornii. Bobova kavárna nedávno začala přijímat bitcoinové platby přidáním možnosti platit v bitcoinech ve svém pokladním systému. Ceny v Bobově kavárně jsou uvedeny v místní měně (americký dolar), ale u pokladny mají zákazníci možnost platit buďto dolary nebo bitcoiny. Alice si objednala šálek kávy a Bob zaznamenal transakci v pokladně. Pokladní systém převede celkovou cenu z dolarů na bitcoiny za použití běžného tržního kurzu a zobrazí ceny v obou měnách, stejně tak zobrazí QR kód obsahující *platební požadavek* pro tuto transakci (viz Platební požadavek QR kódem (Nápověda: zkuste ho naskenovat!)):

```
Celkem:
$1,50 USD
0,015 BTC
```

Figure 2. Platební požadavek QR kódem (Nápověda: zkuste ho naskenovat!)

Platební požadavek QR kódem kóduje následující webovou adresu, definovanou v BIP0021:

```
bitcoin:1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA?
amount=0.015&
label=Bob%27s%20Cafe&
message=Purchase%20at%20Bob%27s%20Cafe

Části webové adresy

Bitcoinová adresa: "1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA"
Výše platby: "0,015"
Označení adresy příjemce: "Bob's Cafe"
Popis platby: "Purchase at Bob's Cafe"
```

TIP Narozdíl od QR kódu, který obsahuje cílovou bitcoinovou adresu, platební požadavek je do QR zakódovaná webová adresa obsahující cílovou adresu, výši platby a druhový popis jako "Bob’s Cafe." To umožní bitcoinové peněženkové aplikaci předvyplnit informace nutné pro zaslání platby, zatímco je zobrazí v lidsky čitelné formě uživateli. Můžete si naskenovat QR kód pomocí bitcoinové peněženkové aplikace, abyste viděli, co viděla Alice.

Bob říká. "Je to dolar padesát, nebo patnáct milibitů"

Alice použila svůj chytrý telefon k naskenování čárového kódu na displej. Její chytrý telefon ukazuje platbu 0.0150 BTC ve prospěch Bob's Cafe a Alice zvolila Send pro schválení platby. Do několika sekund (podobně jako je čas autorizace kreditní karty), Bob uvidí transakci v pokladně, čímž je transakce úspěšně zakončena.

V následujících sekcích prozkoumáme transakce do větších podrobností, podíváme se jak Alicina peněženka sestavila transakci, jak ji propagovala sítí, jak byla ověřena a nakonec jak Bob může utratit částku v následujících transakcích.

NOTE

Bitcoinová síť může přenášet zlomkové částky, například od milibitcoinů (1/1000 bitcoinů) až po 1/100 000 000 bitcoinu, která je známá jako satoshi. V této knize používáme pojem "bitcoiny" pro odkaz na libovolné množství bitcoinové měny, od nejmenší jednotky (1 satoshi) po celkový počet všech bitcoinů (21 000 000), které budou kdy vytěženy.

Bitcoinové transakce

Jednoduše řečeno, transakce říká síti, že majitel nějakého množství bitcoinů schválil převod jejich části ve prospěch jiného uživatele. Nový majitel nyní může utratit tyto bitcoiny vytvořením nové transakce, kterou schválí převod na jiného vlastníka, a tak dále, v řetězu vlastnictví.

Transakce jsou jako řádky v knize podvojného účetnictví. Jednoduše řečeno, každá transakce obsahuje jeden nebo více "vstupů", které jsou vydáními z bitcoinového účtu. Na druhé straně transakce je jeden nebo více výstupů, které jsou příjmy bitcoinového účtu. Součty vstupů a součty výstupů (výdajů a příjmů) se nemusejí nutně rovnat. Místo toho, součty výstupů jsou mírně menší než součty vstupů. Rozdíl reprezentuje implicitní "transakční poplatek", který je malou částí platby ve prospěch těžaře, který zařadí transakci do účetního systému. Bitcoinové transakce se zobrazují jako záznam účetní knihy Transakce jako záznam v knize podvojného účetnictví.

Transakce také obsahují důkaz vlastnictví pro každé množství bitcoinů (vstupy), které jsou převáděny, ve formě digitálního podpisu majitele, která může být nezávisle ověřen kýmkoliv. V bitcoinové terminologii "utracení" je podepsání transakce, která převádí hodnotu z předchozí transakce na nového majitele identifikovaného bitcoinovou adresou.

TIP

Transakce přesouvá hodnotu ze *vstupů transakce* do *výstupů transakce*. Vstup je zdroj, odkud hodnota pochází, obvykle výstup předchozí transakce. Transakční výstup přiřazuje nového vlastníka této hodnotě pomocí přiřazení klíče, k této hodnotě. Cílový klíč se nazývá *břemeno*, které ukládá požadavek podpisu pro budoucí uvolnění těchto finančních prostředků v nové transakci. Výstupy z jedné transakce mohou být použity jako vstupy nové transakce, což vytváří řetěz vlastnictví, jak se hodnota přesouvá z adresy na adresu (viz Řetěz transakcí, kde výstup jedné transakce je vstupem následující transakce).

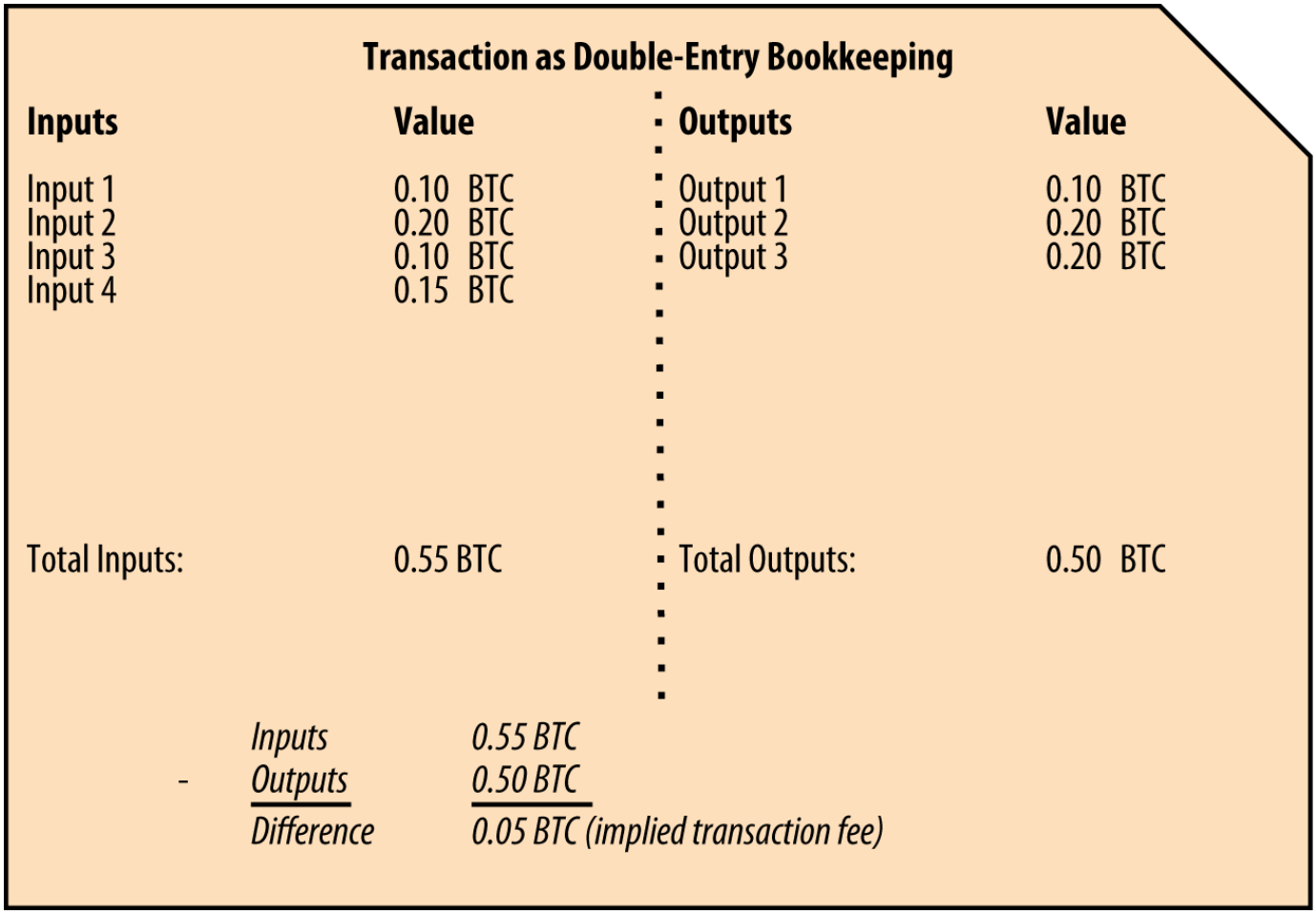


Figure 3. Transakce jako záznam v knize podvojného účetnictví

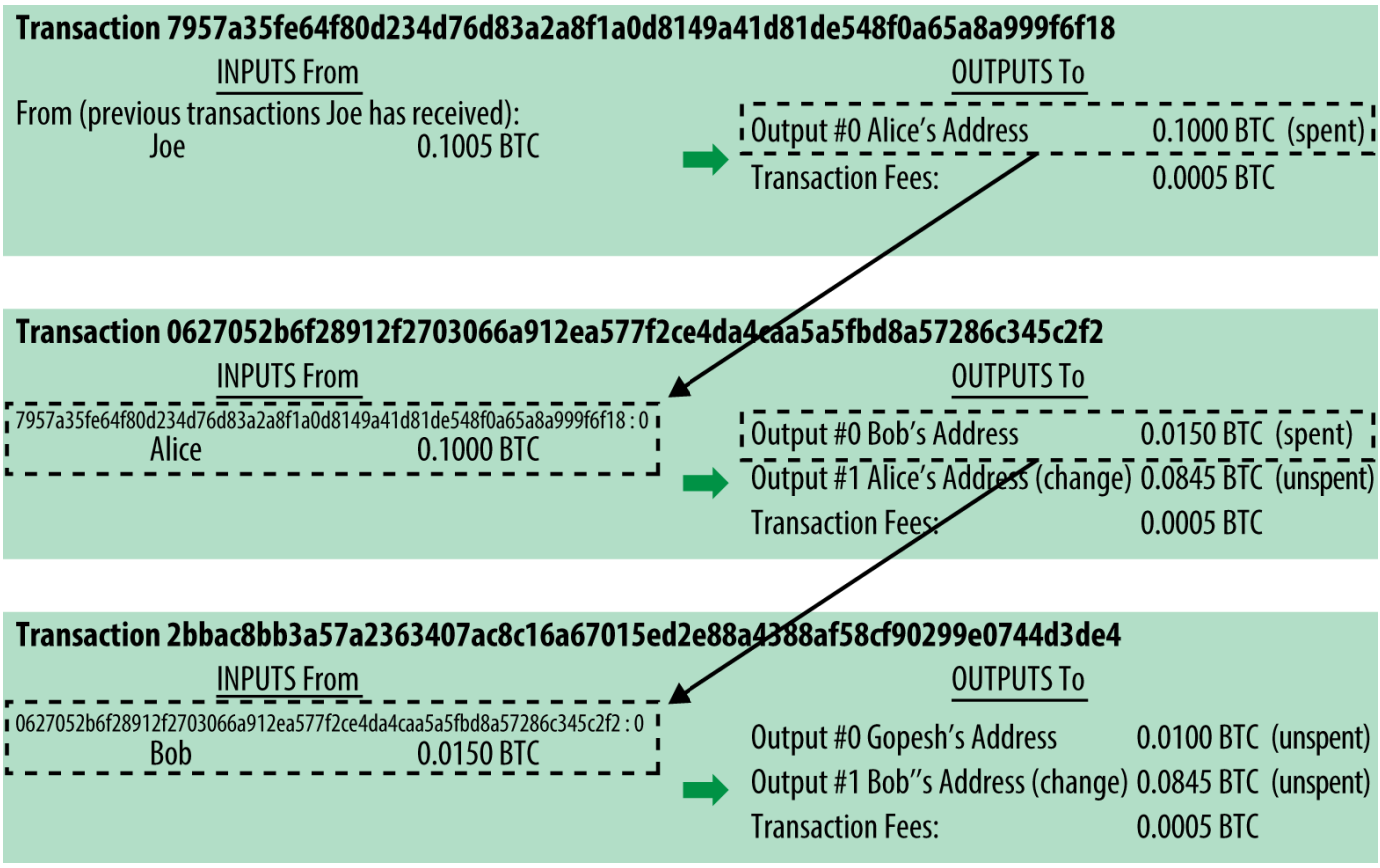


Figure 4. Řetěz transakcí, kde výstup jedné transakce je vstupem následující transakce

Alicina platba Bobově kavárně používá předchozí transakci jako svůj vstup. V předchozí kapitole Alice nakoupila bitcoiny za hotovost od svého kamaráda Joa. Transakce má nějaké množství bitcoinů zamčených (zatížených břemenem) Aliciným klíčem. Její nová transakce Bobově kavárně se odkazuje na předchozí transakci jako na vstup a vytváří nové výstupy na zaplacení šálku kávy a obdržení zbytku neutracené vstupní částky. Transakce vytvářejí řetěz, ve kterém vstupy z novějších transakcí odpovídají výstupům z předchozích transakcí. Alicin klíč poskytuje podpis, který odemkne tyto výstupy předchozí transakce, čímž dokazuje bitcoinové síti, že Alice tyto finanční prostředky vlastní. Alice přidala platbu za kávu na Bobovu adresu, čímž výstup "zatížila břemenem" požadavku, aby Bob podepsal budoucí požadavek na utracení této částky. Toto reprezentuje převod hodnoty mezi Alicí a Bobem. Tento řetěz transakcí, od Joa přes Alici k Bobovi, je znázorněn na Řetěz transakcí, kde výstup jedné transakce je vstupem následující transakce.

Obvyklé formy transakcí

Nejobvyklejší formy transakcí jsou jednoduchá platba z jedné adresy na druhou, která často obsahuje nějakou "vratku" - vracející část částky původnímu vlastníkovi. Tento typ transakcí má jeden vstup a dva výstupy a je zobrazen na Nejobvyklejší transakce.

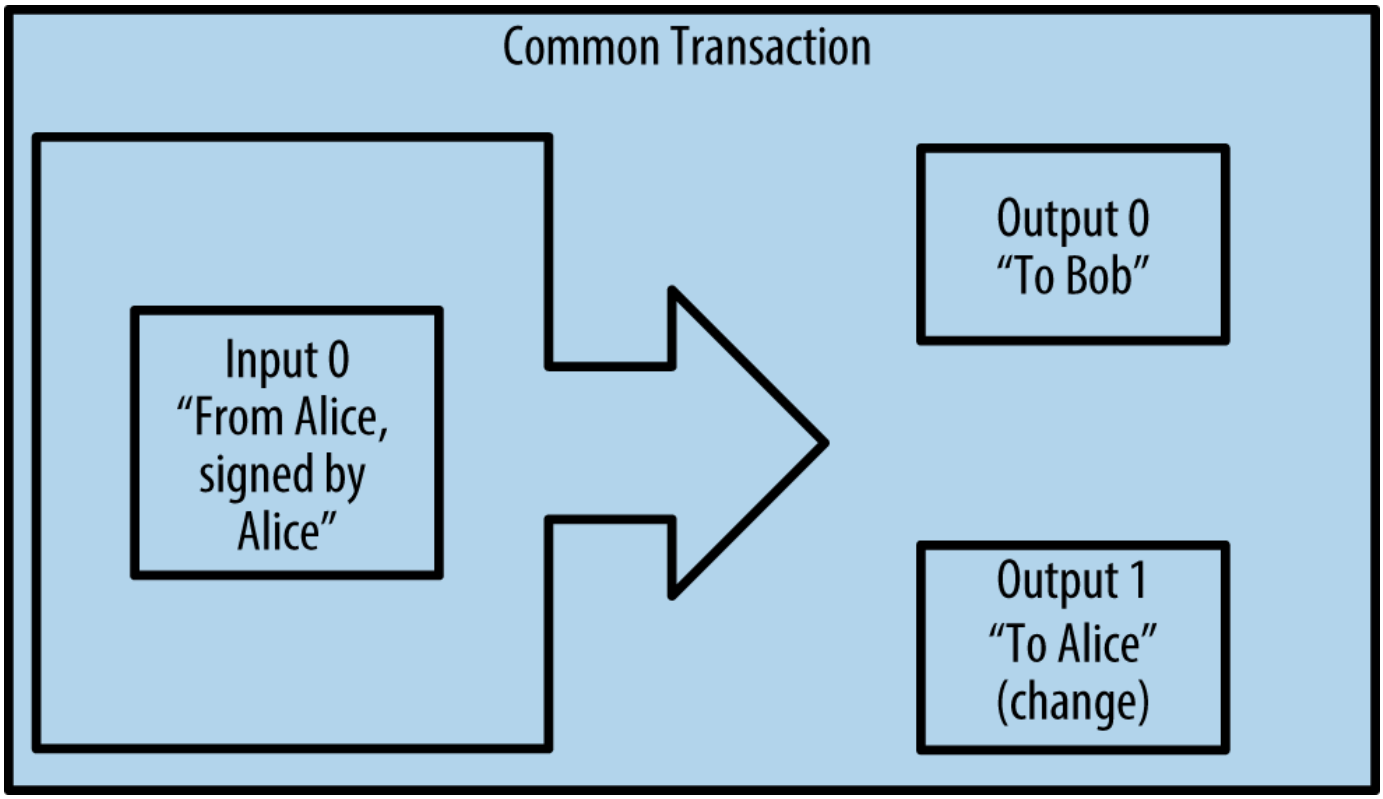


Figure 5. Nejobvyklejší transakce

Další obvyklá forma transakce je sloučení několika vstupů do jednoho výstupu (viz Transakce slučující finanční prostředky). Toto reprezentuje v reálném životě výměnu hrsti mincí za jednu bankovku. Transakce jako tyto jsou občas vytvářeny peněžníkovými aplikacemi za účelem vyčištění mnoha malých částek, které byly obdrženy jako vratky plateb.

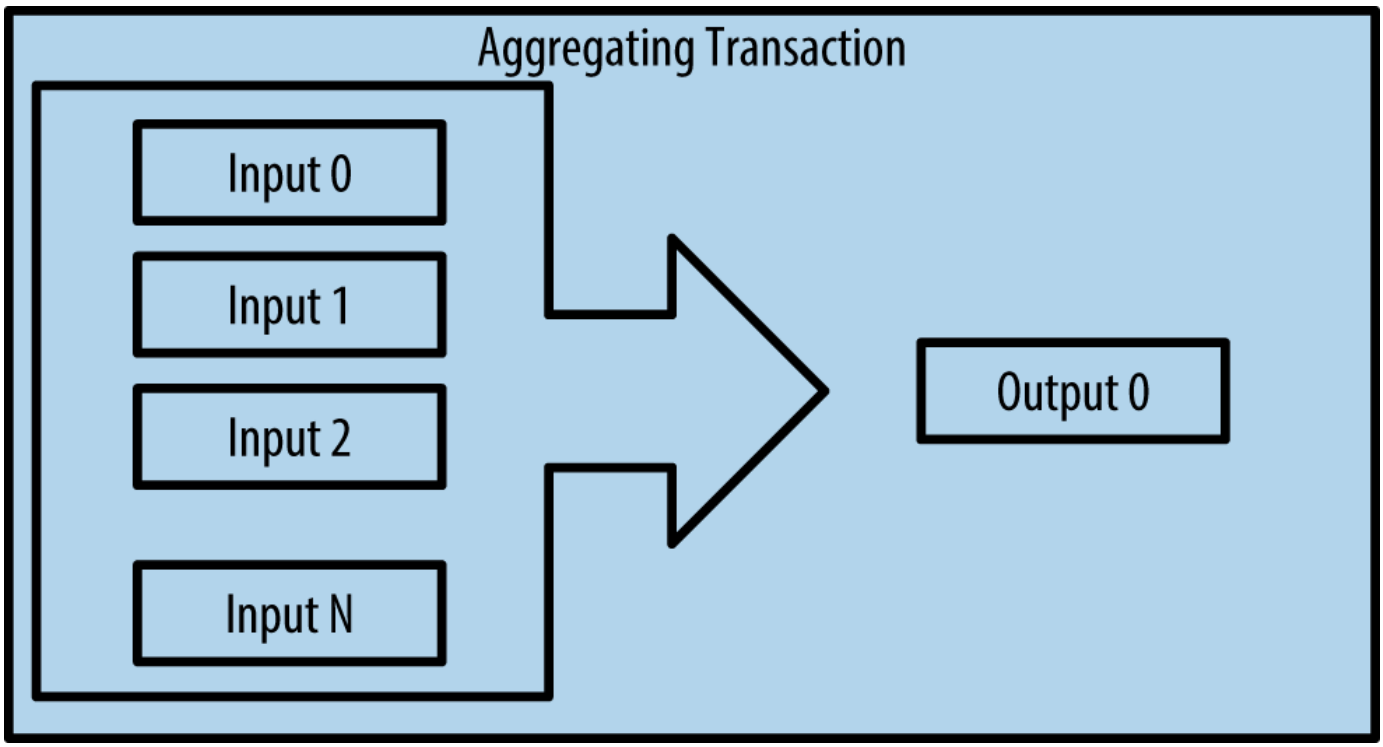


Figure 6. Transakce slučující finanční prostředky

Nakonec, další forma transakce, která je často vidět v bitcoinovém účetním systému, rozděluje jeden vstup na více výstupů, určených různým příjemcům (viz Transakce rozdělující finanční prostředky). Tento typ transakcí je občas používán komerčními subjekty pro rozdělení finančních prostředků, jako když se zasílají platby mzdy více zaměstnancům.

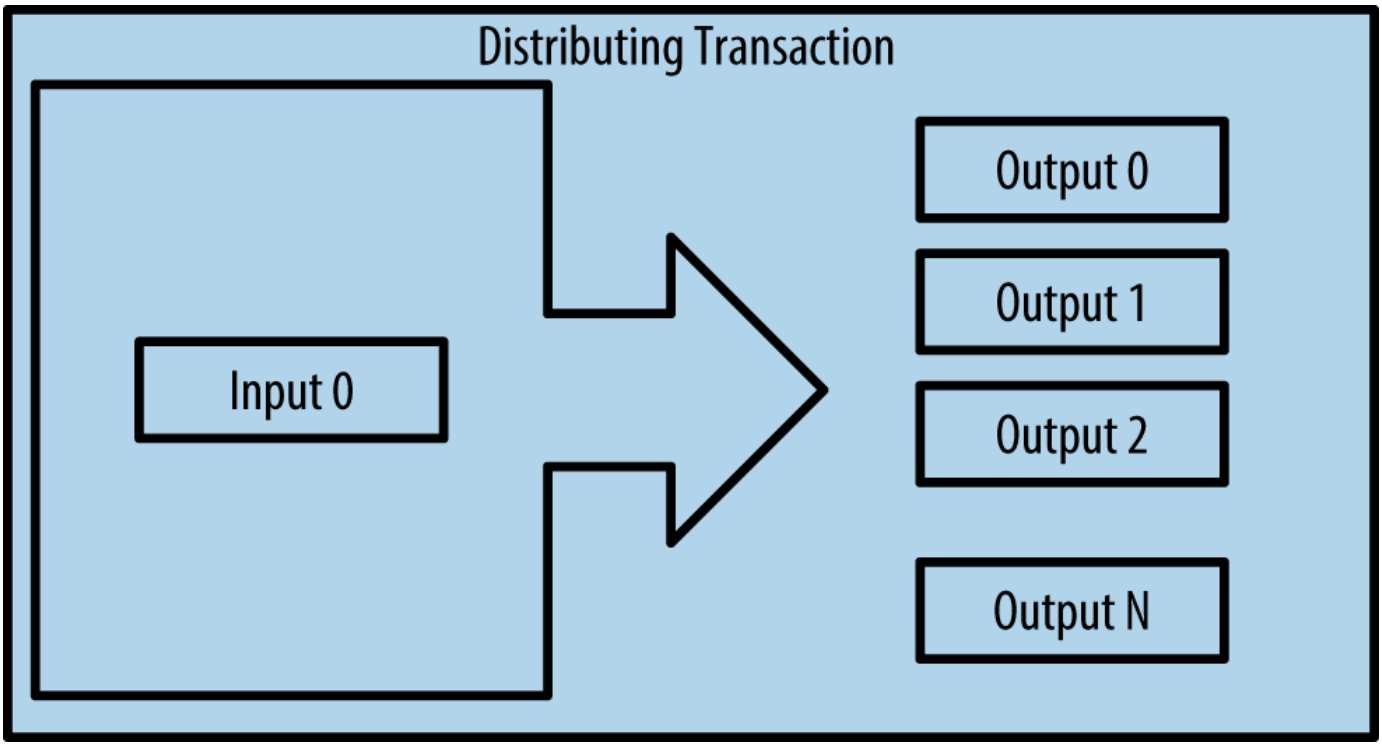


Figure 7. Transakce rozdělující finanční prostředky

Sestavení transakce

Alicina peněženková aplikace obsahuje všechny postupy potřebné pro výběr vhodných vstupů a výstupů pro sestavení transakce, dle požadavků Alice. Alice pouze potřebuje určit cíl a množství a zbytek se uskuteční v peněženkové aplikaci, aniž by Alice viděla podrobnosti. . Důležité je, že peněženková aplikace může sestavit transakci dokonce i když je úplně offline. Podobně jako sepsání šeku doma a jeho pozdější odeslání do banky v obálce, nemusíte být připojeni na bitcoinnovou síť, abyste mohli sestavit a podepsat transakci. Stačí, když bude do sítě odeslána po sestavení.

Získání správných vstupů

Alicina peněženková aplikace musí nejdříve najít vstupy, ze kterých může zaplatit částku, kterou Alice chce zaslat Bobovi. Většina peněženkových aplikací udržuje malou databázi "neutracených výstupů transakcí", které jsou zamčeny (zatíženy břemenem) klíči majitele peněženky. Proto bude Alicina peněženka obsahovat kopii transakčního výstupu z transakce od Joa, která byla vytvořena při směně za hotovost (viz [getting_first_bitcoin]). Aplikace bitcoinové peněženky, která běží jako plně indexovaný klient, obsahuje aktuální kopii každého neutraceného výstupu z každé transakce v blockchainu. To umožňuje peněžence sestavit transakční vstupy stejně tak rychle jako ověřit, že příchozí transakce mají správné vstupy. Nicméně, protože plně indexovaní klienti zabírají mnoho místa na disku, většina uživatelů používá "odlehčené" klienty, které sledují pouze vlastní neutracené

výstupy uživatele.

Pokud peněženková aplikace neobsahuje kopie neutracených výstupů transakcí, může se zeptat bitcoinové sítě a získat tyto informace použitím různých API poskytovaných různými poskytovali. Také se může zeptat plně indexovaného uzlu za použití bitcoinového JSON RPC API. Vyhledání všech neutracených výstupů Aliciny bitcoinové adresy ukazuje požadavek REST API, sestavený jako HTTP GET příkaz na konkrétní URL. Toto URL vrátí všechny neutracené výstupy transakcí, což dává aplikaci informace nutné pro vytvoření transakčních vstupů pro utracení. Použijeme jednoduchého HTTP klienta *cURL* běžícího z příkazové řádky, který nám vrátí odpověď.

Example 1. Vyhledání všech neutracených výstupů Aliciny bitcoinové adresy

```
$ curl https://blockchain.info/unspent?active=1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK
```

Example 2. Odpověď na vyhledání

```
{

    "unspent_outputs":[

        {
            "tx_hash":"186f9f998a5...2836dd734d2804fe65fa35779",
            "tx_index":104810202,
            "tx_output_n": 0,
            "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
            "value": 10000000,
            "value_hex": "00989680",
            "confirmations":0
        }

    ]
}
```

Odpověď uvedená v Odpověď na vyhledání ukazuje jeden neutracený výstup (ten, který nebyl ještě uvolněn) ve vlastnictví Aliciny adresy 1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK. Odpověď obsahuje odkaz na transakci, ve které se tento neutracený výstup nachází (platba od Joa), jeho hodnotu v satoshi, která činí 10 miliónů (= 0,10 BTC). S touto informací může Alicina peněženková aplikace sestavit transakci pro převod hodnoty na adresy nových vlastníku.

TIP Podívejte se na transaction from Joe to Alice.

Jak můžete vidět, Alicina peněženka obsahuje dostatek bitcoinů v jednotlivém neutraceném výstupu,

aby mohla zaplatit za šálek kávy. Nenastala situace, aby se Alicina peněženková aplikace musela "prohrabávat" v hromadě menších neutracených výstupů, jako vybírání mincí z měšce, dokud nenajdeme dostatek hodnoty na zaplacení kávy. V obou případech může nastat potřeba získat nějakou částku zpátky (vratka), což uvidíme v další části, jak peněženková aplikace vytváří transakční výstupy (platby).

Vytváření výstupů

Transakční výstup je vytvořen formou skriptu, který vytváří břemeno na hodnotě, která může být uvolněna představením řešení skriptu. Jednoduše řečeno, Alicin výstup transakce bude obsahovat skript, který říká "Výstup patří tomu, kdo je schopen vytvořit podpis odpovídající Bobově veřejnému klíči." Protože pouze Bob má peněženku s klíči odpovídajícími této adrese, pouze Bobova peněženka může vytvořit takovýto podpis, který je nutný k uvolnění hodnoty tohoto výstupu. Alice proto zatíží hodnotu výstupu břemenem, k jehož odemknutí je třeba podpis Boba.

Tato transakce bude také obsahovat druhý výstup, protože Aliciny finanční prostředky ve formě 0,10 BTC výstupu jsou příliš vysoké na zaplacení 0,015 BTC za kávu. Alice bude chtít nazpět vratku 0,085 BTC. Alicina vratková platba je vytvořena *Alicinou peněženkou* ve stejné transakci jako se nachází platba Bobovi. V podstatě, Alicina peněženka rozdělí její finanční prostředky na dvě platby: jedna Bobovi, druhá zpátky Alici. Alice může použít vratkový výstup v následující transakci, může ji utratit později.

Nakonec, aby transakce byla sítí zpracována včas, Alicina peněženková aplikace přidá malý poplatek. Tento poplatek není explicitně v transakci, je odvozen z rozdílů součtu hodnot vstupů a výstupů. Pokud Alice místo vratky 0,085 vytvoří druhý výstup pouze s hodnotou 0,0845, zůstane tam 0,0005 BTC (půlka milibitcoinu). Hodnota vstupu 0,10 BTC není plně utracena dvěma výstupy, protože součet jejich hodnot je trochu nižší než 0,10 BTC. Výsledný rozdíl je *transakční poplatek*, který je vyzvednut těžařem jako poplatek za vložení transakce do bloku, který je vložen do blockchainového účetního systému.

Výsledná transakce může být zobrazena za pomocí webové aplikace blockchainového průzkumníka, jak ukazuje Alicina transakce pro Bobovu kavárnu.

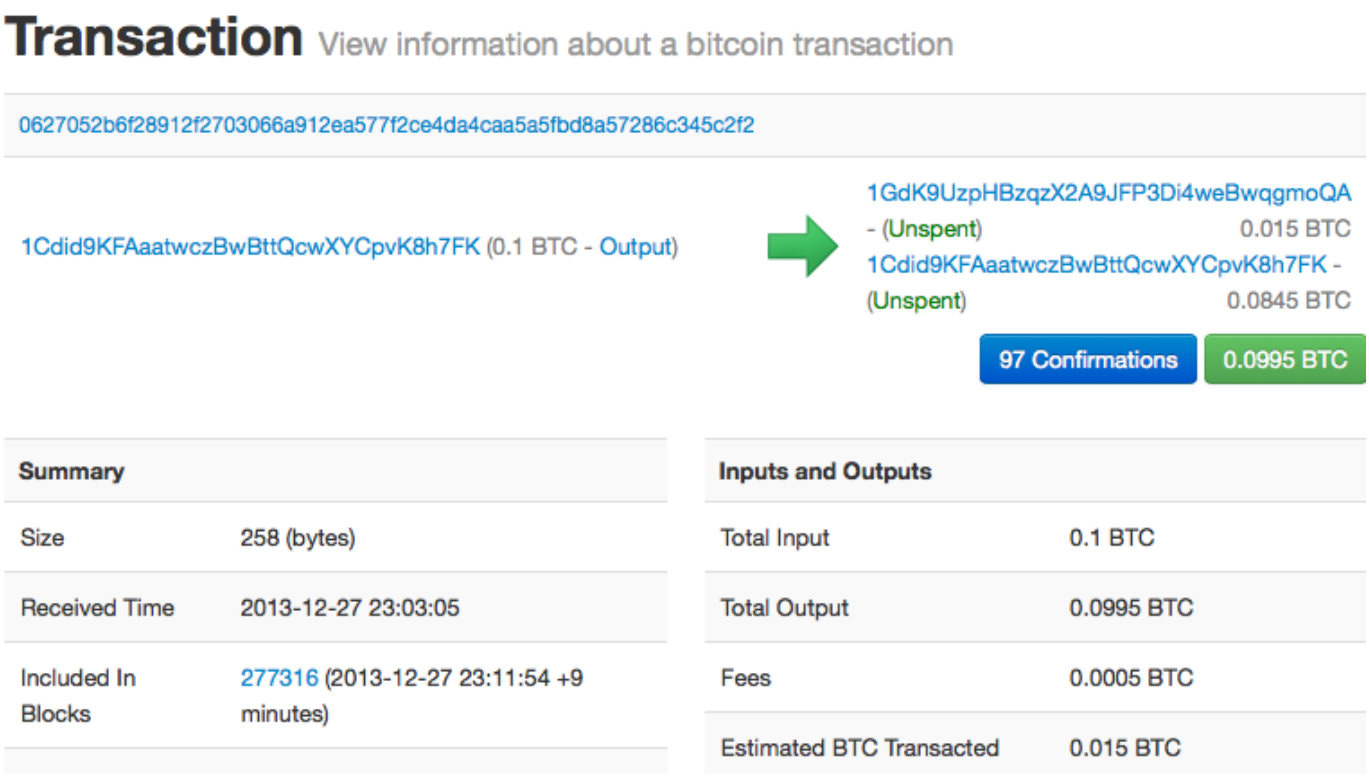


Figure 8. Alicina transakce pro Bobovu kavárnu

TIP prohlédněte si transaction from Alice to Bob's Cafe.

Přidání transakce do účetního systému

Transakce vytvořená Alicinou peněženkovou aplikací je dlouhá 258 bytů a obsahuje všechny nezbytnosti pro potvrzení vlastnictví finančních prostředků a jejich přiřazení novým majitelům. Nyní, musí být transakce přeneseny do bitcoinové sítě, kde se stanou součástí distribuovaného účetního systému (blockchain). V další části uvidíme, jak se transakce stává částí nového bloku a jak je nový blok "vytěžen". Nakonec uvidíme, jak je nový blok přidán do blockchainu a jak je postupně zvyšována jeho důvěryhodnost v síti s tím, jak jsou za něj přidávány další bloky.

Přenos transakcí

Protože transakce obsahuje všechny informace nezbytné pro zpracování, není důležité jak nebo kde je přenesena do bitcoinové sítě. Bitcoinová síť je peer-to-peer síť, ve které je každý zúčastněný bitcoinový klient spojen s několika dalšími bitcoinovými klienty. Účelem bitcoinové sítě je šířit transakce a bloky mezi všechny účastníky.

Jak probíhá šíření

Alicina peněženková aplikace může poslat novou transakci jakémukoliv jinému bitcoinovému klientovi, který je připojen přes jakékoliv internetové spojení: drátové, WiFi nebo mobilní. Její bitcoinová peněženka nemá přímé spojení s Bobovou bitcoinovou peněženkou a nemusí použít internetové přípojení poskytované kavárnou, přestože tyto dvě možnosti jsou také možné. Každý uzel

bitcoinové sítě (jiný klient), který obdrží platnou transakci, kterou doposud neviděl, ji okamžitě přepošle ostatním uzlům, ke kterým je připojen. Z tohoto důvodu se transakce rychle rozšiřuje po celé peer-to-peer síti a dosáhne vysokého procentního rozšíření mezi uzly do pár sekund.

Bobův pohled

Pokud je Bobova bitcoinová peněženková aplikace přímo spojena s Alicinou peněženkovou aplikací, Bobova peněženková aplikace může být prvním uzlem, který obdrží tuto transakci. Nicméně, dokonce pokud Alicina peněženka pošle transakci jiným uzlům, obdrží ji Bobova peněženka do pár sekund. Bobova peněženka okamžitě identifikuje Alicinu transakci jako příchozí platbu, protože obsahuje výstup uvolnitelný Bobovými klíči. Bobova peněženková aplikace může také nezávisle ověřit, že je transakce dobře formovaná, používá dříve neutracené vstupy a obsahuje dostatečný transakční poplatek, aby byla zařazena do dalšího bloku. V tuto chvíli může Bob předpokládat (s malým rizikem), že transakce bude brzo zařazena do bloku a potvrzena.

TIP

Často se o bitcoinových transakcích mylně uvádí, že musejí být "potvrzeny" 10-ti minutovým čekáním na nový blok, nebo 60-ti minutovým čekáním na plných šest potvrzení. Přestože potvrzení zajišťují, že transakce byla přijata sítí jako celkem, takovéto zpoždění není nezbytné pro položky s nízkou hodnotou jako je šálek kávy. Obchodník může přijmout platnou transakci s nízkou hodnotou bez potvrzení, aniž by riskoval více než riskuje při přijetí kreditní karty bez dokladu totožnosti nebo podpisu, což je dnes běžnou praxí.

Těžba bitcoinů

Transakce je nyní rozšířena v bitcoinové síti. Nestane se částí sdíleného účetního systému (*blockchainu*) dokud není ověřena a vložena do bloku pomocí procesu zvaného *těžba*. Viz [ch8] pro bližší vysvětlení,

Bitcoinový systém důvěry je založen na výpočtech. Transakce jsou shlukovány do *bloků*, které požadují vysoké množství výpočtů pro jeho správné vytvoření, ale pouze malé množství výpočtů pro ověření jeho správnosti. Proces těžby bitcoinu slouží ke dvěma účelům:

Těžba vytváří nové bitcoiny v každém bloku, hodně podobně jako centrální banka tiskne nové peníze. Množství bitcoinu vytvořených v jednom bloku je pevně dané a snižuje se v čase. * Těžba vytváří důvěru ujištěním, že transakce byly potvrzeny pouze tehdy, pokud byl věnován dostatek výpočetní síly k vytvoření bloku, který je obsahuje. Více bloků znamená více výpočtů, které znamenají více důvěry.

Dobrým způsobem popsání těžby je obří soutěž v luštění sudoku, které se začíná znova od začátku pokaždé, když je nalezeno řešení, a jehož obtížnost se automaticky upravuje, aby najití řešení trvalo přibližně 10 minut. Představte si obří hlavolam sudoku, o velikosti několika tisíc řádek a sloupců. Pokud vám ukážu vyplněný hlavolam, můžete rychle ověřit, zda je správně vyplněný. Nicméně pokud hlavolam má několik čtverečků vyplněných a zbytek je prázdných, je třeba hodně práce k jeho vyřešení! Obtížnost sudoku může být upravena změnou jeho velikosti (více nebo méně řádků a sloupců), ale ověření správnosti řešení je docela snadné, dokonce i když je hlavolam velmi velký.

Hlavolam použitý v bitcoinu je založen na kryptografickém hašování a splňuje podobné charakteristiky: je asymetricky těžký k vyřešení, ale snadný k ověření a obtížnost může být upravována.

V [user-stories] jsme představili Jinga, studenta počítačového inženýrství ze Šanghaje. Jing se účastní bitcoinové sítě jako těžař. Přibližně každých 10 minut se Jing připojuje k tisícům jiných těžařů v celosvětové soutěži v hledání řešení tvořícího blok transakcí. Hledání takového řešení, takzvaný důkaz prací, požaduje biliardy hašovacích operací za sekundu v celé bitcoinové síti. Algoritmus důkazu prací zahrnuje opakovaný výpočet haše hlavičky bloku a náhodného čísla za použití kryptografického algoritmu SHA 256. Výpočet probíhá tak dlouho, dokud není nalezeno řešení které odpovídá předem stanovenému vzoru. První těžař, který najde takovéto řešení vyhrává kolo soutěže a publikuje blok na blockchainu.

Jing začal těžit v roce 2010 za použití velmi rychlého stolního počítače, hledal vhodné důkazy prací pro nové bloky. Jak se zvyšoval počet těžařů v bitcoinové síti, rychle rostla obtížnost řešeného problému. Brzy Jing a ostatní těžaři přešli na specializovaný hardware jako jsou high-end grafické karty (GPU) jaké jsou používány herními počítačovými sestavami nebo konzolemi. V čase psaní této knihy je obtížnost tak vysoká, že je výdělečné těžit pouze s aplikačně specifickými integrovanými obvody (ASIC), v podstatě stovky těžebních algoritmů vytištěných v hardware, běžících paralelně na každém křemíkovém čipu. Jing se také připojil k "těžební skupině", která je podobná sázkařské skupině dovolující několika účastníkům sdílet jejich úsilí a odměny. Jing nyní provozuje dva USB propojené ASIC zařízení těžící bitcoiny 24 hodin denně. Bitcoiny, které vytvoří během těžby prodává. Z utržených peněz platí své účty za elektřinu, zbylé peníze jsou jeho příjmem. Jeho počítač provozuje referenčního bitcoinového klienta bitcoind, který je koncovou částí jeho specializovaného těžebního software.

Těžba transakcí v blocích

Transakce přenášená po síti není ověřena dokud se nestane částí celosvětového distribuovaného účetního systému, blockchainu. V průměru každých 10 minut těžaři vytvoří nový blok, který obsahuje všechny transakce od posledního bloku. Nové transakce stále přicházejí do sítě od peněženek uživatelů nebo jiných aplikací. Jakmile jsou spatřeny uzlem bitcoinové sítě, jsou přidány do dočasného úložiště nepotvrzených transakcí spravovaných každým uzlem. Když težař tvoří nový blok, přidá nepotvrzené transakce z tohoto úložiště do nového bloku a pokusí se poté vyřešit velmi těžký problém (zvaný důkaz prací), aby dokázal platnost tohoto nového bloku. Proces těžby je detailně vysvětlen v [mining].

Při vkládání transakcí do nového bloku jsou upřednostňovány transakce s nejvyššími poplatky a splňující další kritéria. Každý těžař začíná proces těžby nového bloku transakcí jakmile obdrží předchozí blok ze sítě, čímž se dozví, že prohrál předchozí kolo těžby. Okamžitě vytvoří nový blok, vyplní ho transakcemi a otiskem předchozího bloku a začne počítat důkaz prací pro tento nový blok. Každý těžař vkládá speciální transakci do svého bloku. Tato transakce platí na jeho vlastní bitcoinovou adresu odměnu nově vytvořených bitcoinů (aktuálně 12,5 BTC za blok). Pokud najde řešení, které dělá blok platným, "vyhrál" tuto odměnu, protože jeho úspěšný blok je přidán do celosvětového blockchainu a transakce s odměnou, kterou vložil, je utratitelná. Jing, který se účastní těžební skupiny, nastavuje ve svém software, aby odměna za vytvoření nového bloku byla zaplacena na adresu těžební skupiny. Tato těžební skupina zašle Jingovi a ostatním těžařům podíly ze získané odměny rozdělené

proporčně podle množství práce, kterou přispěli v posledním kole těžby.

Alicina transakce byla sebrána sítí a vložena do úložiště nepotvrzených transakcí. Protože má dostatečný poplatek, bude vložena do nového bloku vytvořeného Jingovou těžební skupinou. Přibližně za pět minut poté, co byla transakce poprvé přenesena Alicinou peněženkou, Jingův těžební ASIC našel řešení bloku a publikoval jej jako blok číslo 277316, obsahující dalších 419 transakcí. Jingův těžební ASIC publikoval nový blok na bitcoinové síti, kde jej ostatní těžaři ověřili a začali soutěžit ve vytváření dalšího boku.

Můžete si prohlédnout blok, který obsahuje Alice's transaction.

Za pár minut je nový blok číslo 277317 vytěžen jiným těžařem. Protože je tento nový blok založen na předchozím bloku (číslo 277316), který obsahuje Alicinu transakci, je na vrchol bloku přidáno ještě více výpočtů, což posiluje důvěru v tyto transakce. Transakce nacházející se v bloku obsahujícím Alicinu transakci mají jedno "potvrzení". Každý blok vytěžený na vrcholu jiného bloku činí exponenciálně těžším zvrátit transakce, což je dělá více a více důvěryhodnými pro síť.

V diagramu Alicina Transakce zahrnutá v bloku číslo 277316 můžete vidět blok číslo 277316, který obsahuje Alicinu transakci. Pod tím je 277136 bloků (včetně bloku číslo 0), navzájem spojených v řetěz bloků (blockchain) vedoucích cestou zpět k bloku číslo 0 nazývaného *základní blok* (v originále genesis blok). Časem, jak se výška bloků zvyšuje, roste i výpočetní složitost pro každý blok a řetěz jako celek. Blok vytěžený po bloku, který obsahuje Alicinu transakci, působí jako další ujištění, jak se hromadí více výpočtů v delším a delším řetězu. Podle konvence je blok s více než šesti potvrzeními označován za neodvolatelný, protože by to vyžadovalo obrovské množství výpočetního výkonu k zneplatnění a přepočítání šesti bloků. Proces těžby a způsob budování důvěry prozkoumáme detailněji v [ch8].

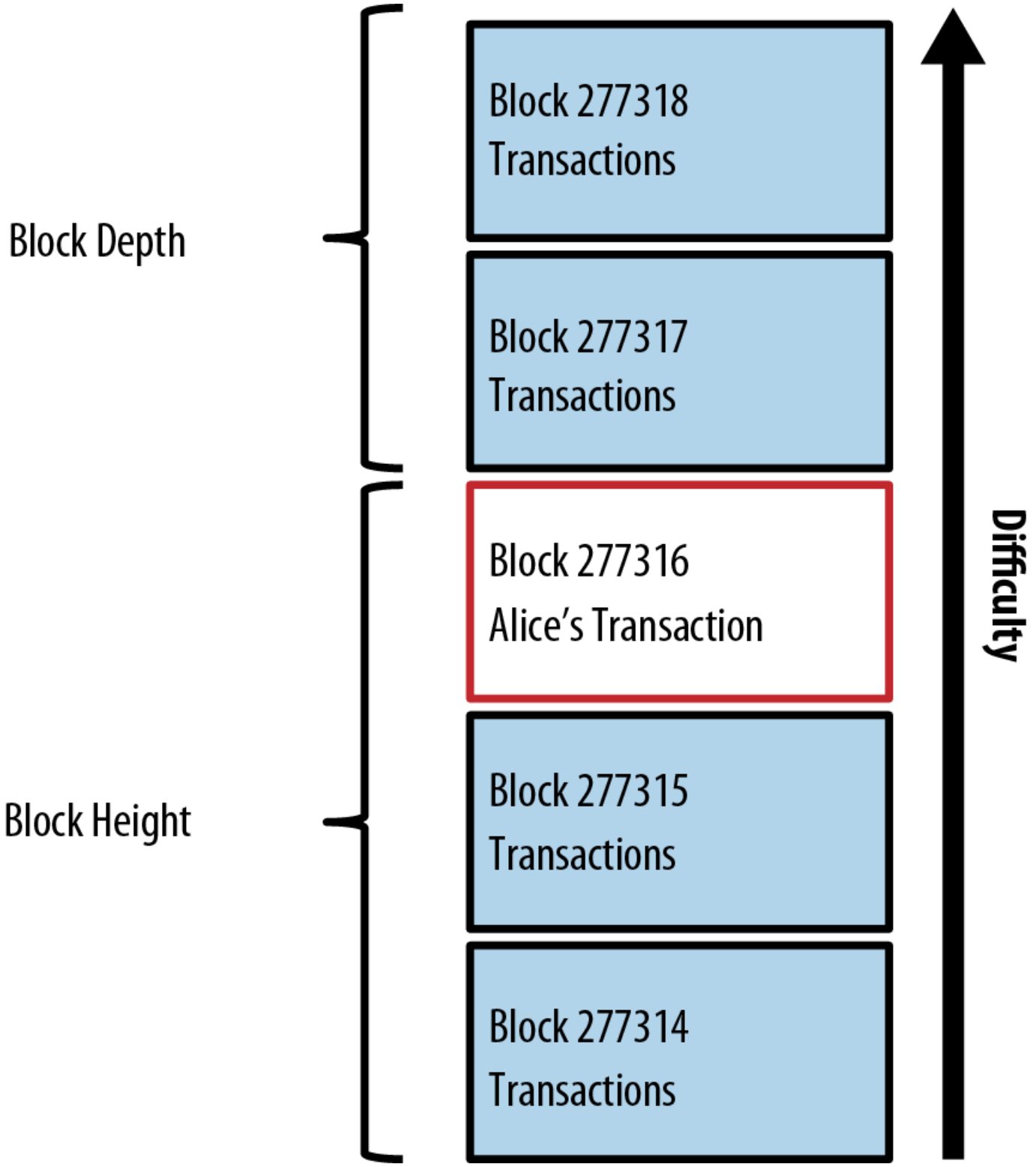


Figure 9. Alicina Transakce zahrnutá v bloku číslo 277316

Utrácení transakce

Nyní byla Alicina transakce zahrnuta do blockchainu jako část bloku, který je částí distribuovaného účetního systému bitcoinu a je viditelný pro všechny bitcoinové aplikace. Každý bitcoinový klient může nezávisle ověřit, že transakce je platná a utratitelná. Plně indexovaný klient může vystopovat

zdroj finančních prostředků od okamžiku, kdy byly prvně vytvořeny v bloku, postupně z transakce do transakce, dokud nedosáhnou Bobovy adresy. Odlehčený klient může zavolat zjednodušené ověření platby (viz [spv_nodes]) potvrzením transakce v blockchainu a existencí několika bloků vytěžených nad blokem, ve kterém je transakce zahrnuta. Toto poskytuje ujištění, že síť transakci přijala, protože je platná.

Bob může nyní utratit výstup z této a jiných transakcí vytvořením jeho vlastní transakce, která odkazuje na tyto výstupy jako na své vstupy a přiřazením nových majitelů. Například Bob může zaplatit dodavatelům, přenosem hodnoty z platby za Alicin šálek kávy na tyto nové majitele. Pravděpodobně, Bobův bitcoinový software bude shlukovat mnoho malých plateb do větší platby, možná shromažďovat všechny denní příjmy z bitcoinových transakcí do jedné transakce. Přesune různé platby na jednu adresu, používanou jako obecný kontrolní účet obchodu. Diagram shlukování transakcí viz Transakce slučující finanční prostředky.

Jakmile Bob utratí platby získané od Alice a ostatních zákazníku, rozšíří řetěz transakcí, které jsou přidány do celosvětového účetního systému blockchain, aby je všichni viděli a věřili jim. Předpokládejme, že Bob zaplatí svému webovému návrháři Gopreshovi z Bengalúru za novou webovou stránku. Nyní řetěz transakcí bude vypadat jako Alicina transakce jako část řetězu transakcí od Joa po Gopesha..

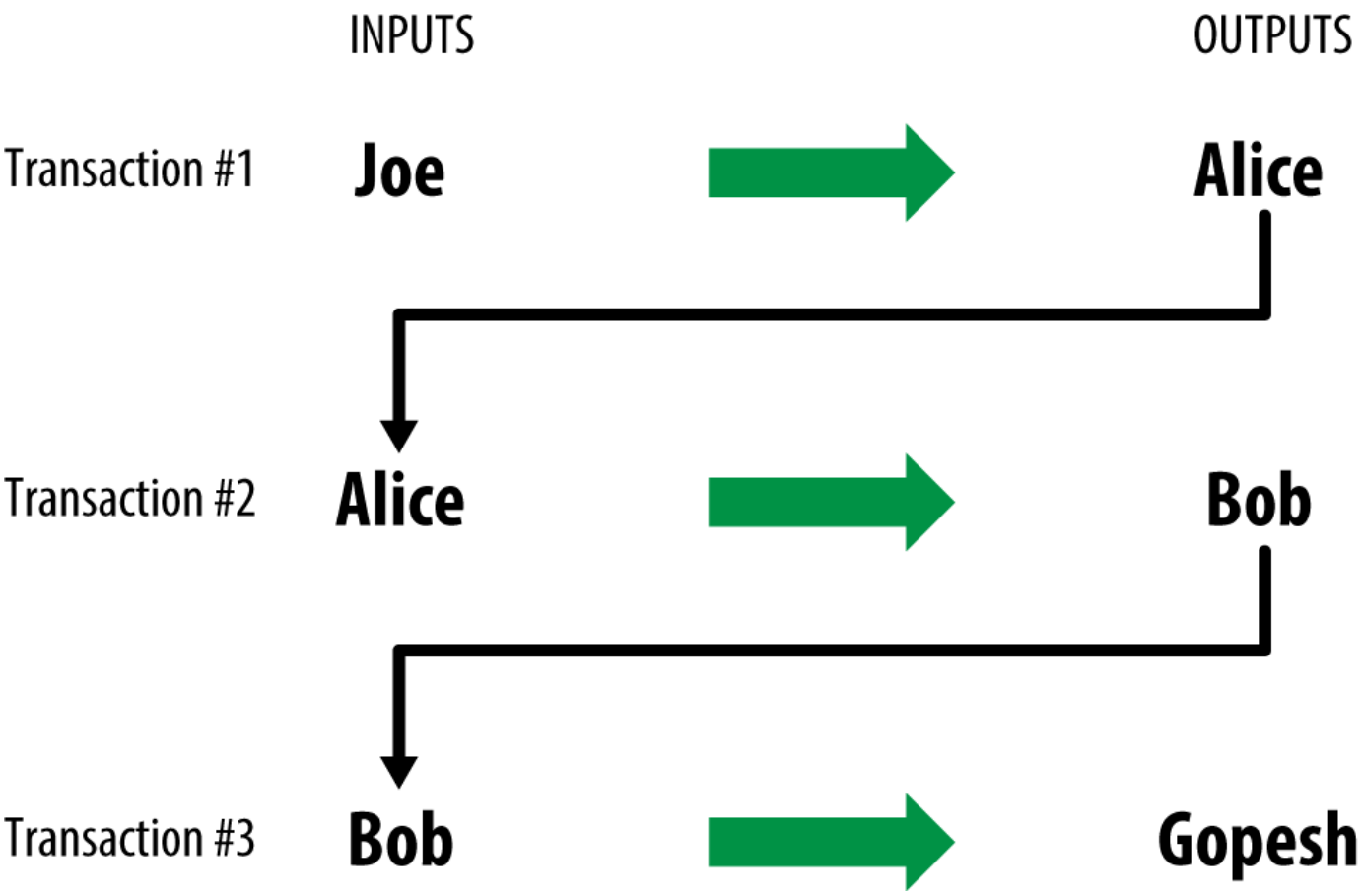


Figure 10. Alicina transakce jako část řetězu transakcí od Joa po Gopesha.

Bitcoinový klient

Bitcoin Core: referenční implementace

Můžete si stáhnout referenčního klienta *Bitcoin Core*, také známého jako "Satoshiho klient" z bitcoin.org. Referenční klient realizuje všechna hlediska bitcoinové sítě, včetně peněženek, ověřování transakcí pomocí plné kopie celého transakčního účetního systému (blockchainu) a úplný uzel bitcoinové peer-to-peer sítě.

Na Bitcoin's Choose Your Wallet page vyberte Bitcoin Core pro stažení referenčního klienta. V závislosti na vašem operačním systému si stáhnete konkrétní instalátor. Pro Windows je k dispozici buďto ZIP archív nebo spustitelný .exe soubor. Pro MacOS je k dispozici .dmg obraz disku. Linux verze obsahuje PPA balíček pro Ubuntu nebo tar.gz archív. Webová stránka bitcoin.org uvádí doporučené bitcoinové klienty jak vidíte v Výběr bitcoinového klienta na bitcoin.org.

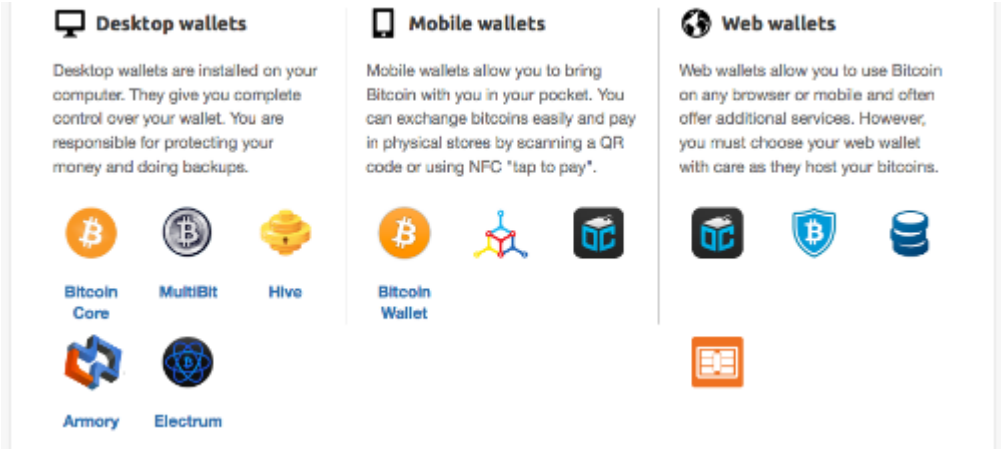


Figure 1. Výběr bitcoinového klienta na bitcoin.org

První spuštění Bitcoin Core

Pokud stáhnete instalovatelný balíček jako .exe, .dmg nebo PPA, můžete jej nainstalovat stejným způsobem jako jakoukoliv jinou aplikaci na vašem operačním systému. Ve Windows spusťte .exe a sledujte pokyny krok za krokem. V Mac OS, spusťte .dmg a přetáhněte ikonu Bitcoin-QT do vašeho *aplikačního* adresáře. V Ubuntu, dvojitě klikněte PPA ve vašem prohlížeči souborů a otevře se organizátor balíčků, který nainstaluje tento balíček. Po dokončení instalace byste měli mít novou aplikaci nazvanou Bitcoin-Qt v seznamu vašich aplikací. Dvojitě klikněte na ikonku a spustíte bitcoinového klienta.

Při prvním spuštění Bitcoin Core se začne stahovat blockchain, tento proces může trvat několik dní (viz Obrazovka Bitcoin Core během inicializace blockchainu). Nechte to běžet v pozadí dokud se neobjeví "Synchronized" a dále již nezobrazuje "out of sync" vedle stavu účtu.

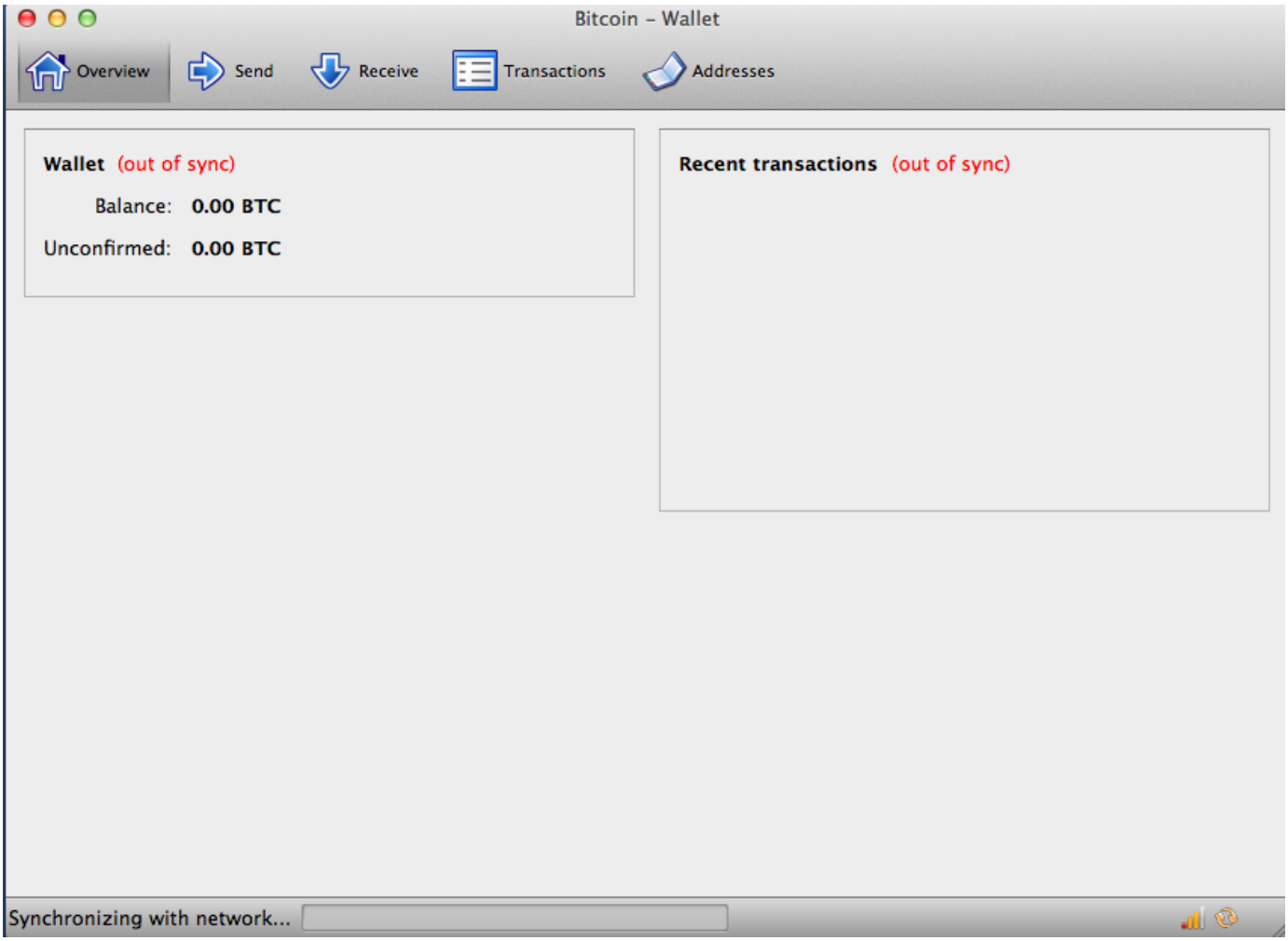


Figure 2. Obrazovka Bitcoin Core během inicializace blockchainu

TIP

Bitcoin Core udržuje plnou kopii transakčního účetního systému (blockchain) obsahující každou transakci, která se kdy objevila v bitcoinové síti od jejího spuštění v roce 2009. Tato datová množina je několik gigabytů velká (přibližně 16 GB koncem roku 2013) a je stahována postupně během několika dní. Klient nebude schopný zpracovat transakce nebo aktualizovat stavy účtu, dokud nebude celá datová množina stažena. Během této doby klient bude ukazovat "out of sync" vedle stavu účtu a ukazuje "Synchronizing" v patičce. Ujistěte se, že máte dostatek místa na disku, dostatečné síťové připojení a čas na dokončení úvodní synchronizace.

Kompilace Bitcoin Core ze zdrojových kódů

Pro vývojáře je zde možnost stáhnout si plné zdrojové kódy v ZIP archívu nebo udělat kopii zdrojových kódů ze spolehlivého úložiště GitHub. Na GitHub bitcoin page, vyberte "Download Zip" z postranní nabídky. Alternativně použijte git příkazovou řádku na vytvoření místní kopie zdrojových kódů ve vašem systému. V následujícím příkladě kopírujeme zdrojové kódy pomocí Unixové příkazové řádky v Linuxu nebo Mac OS:

```
$ git clone https://github.com/bitcoin/bitcoin.git
Cloning into 'bitcoin'...
remote: Counting objects: 31864, done.
remote: Compressing objects: 100% (12007/12007), done.
remote: Total 31864 (delta 24480), reused 26530 (delta 19621)
Receiving objects: 100% (31864/31864), 18.47 MiB | 119 KiB/s, done.
Resolving deltas: 100% (24480/24480), done.
$
```

TIP Příkazy a výsledný výstup se může lišit verzi od verze. Postupujte dle dokumentace, která byla přiložena ke zdrojovým kódům, dokonce i když se liší od příkazů zde uvedených a nebuďte překvapeni, pokud zobrazený výstup na vaší obrazovce se mírně liší od příkladu zde uvedeného.

Když byla kopírovací operace git dokončena, budete mít kompletní místní kopii zdrojových kódů z úložiště v adresáři *bitcoin*. Přesuňte se do tohoto adresáře napsáním cd bitcoin do příkazové stránky.

```
$ cd bitcoin
```

Ve výchozím stavu bude místní kopie synchronizována s nejnovějšími zdrojovými kódy, které mohou být nestabilní nebo beta verzí bitcoinu. Před kompilací zdrojových kódů vyberte konkrétní verzi kontrolou *značky* vydání. Toto synchronizuje místní kopii se specifickou verzí zdrojových kódů, označených danou značkou klíčového slova. Vývojáři používají značky k označení specifických vydání zdrojových kódů číslem verze. Nejprve k nalezení dostupných značek použijeme příkaz git tag:

```
$ git tag
v0.1.5
v0.1.6test1
v0.2.0
v0.2.10
v0.2.11
v0.2.12

[... mnoho dalších značek ...]

v0.8.4rc2
v0.8.5
v0.8.6
v0.8.6rc1
v0.9.0rc1
```

Seznam značek zobrazuje všechny vydané verze bitcoinu. Podle konvence *kandidáti vydání*, kteří jsou určeni k testování, mají příponu "rc". Stabilní vydání, která mohou běžet na produkčních systémech

nemají příponu. Z předchozího seznamu vybereme nejvyšší verzi vydání, která je v době psaní této knihy v0.9.0rc1. Pro synchronizaci místních zdrojových kódů s touto verzí použijeme příkaz git checkout

```
$ git checkout v0.9.0rc1
Note: checking out 'v0.9.0rc1'.

HEAD is now at 15ec451... Merge pull request #3605
$
```

Zdrojové kódy obsahují dokumentaci, která se nachází v mnoha souborech. Prohlédněte si hlavní dokumentaci umístěnou v *README.md* v bitcoinovém adresáři napsáním příkazu more README.md do příkazové řádky a použijte mezerník pro pokračování na další stranu. V této kapitole zkompilujeme bitcoinového klienta ovládaného z příkazové řádky známého jako bitcoind na Linuxu. Prohlédněte si pokyny pro zkompilování bitcoind klienta ovládaného z příkazové řádky na vaší platformě napsáním příkazu more doc/build-unix.md. Alternativní pokyny pro Mac OS X a Windows lze nalézt v adresáři *doc* v souborech *build-osx.md* respektive *build-msw.md*,

Pozorně si prohlédněte požadavky pro kompilaci, které jsou uvedeny v první části kompilační dokumentace. Před zahájením kompilace bitcoinu musíte mít nainstalovány některé knihovny ve vašem systému. Pokud některý z požadavků není splněn, proces kompilace skončí s chybou. Pokud k tomu dojde, nainstalujte požadovanou chybějící knihovnu a pokračujte v kompilačním procesu tam, kde jste ho předtím opustili. Předpokládejme, že požadované knihovny jsou nainstalovány, můžete začít kompilační proces vytvořením množiny kompilačních skriptů za použití skriptu *autogen.sh*.

TIP Kompilační proces Bitcoin Core byl změněn na použití systému autogen/configure/make od verze 0.9. Starší verze používají jednoduchý Makefile a pracují trochu odlišně než následující příklad. Řiďte se pokyny pro verzi, kterou chcete zkompilovat. Ve verzi 0.9 představený autogen/configure/make introduced bude pravděpodobně používán pro všechny budoucí verze zdrojových kódů a je představen v následujících příkladech

```
$ ./autogen.sh
configure.ac:12: installing `src/build-aux/config.guess'
configure.ac:12: installing `src/build-aux/config.sub'
configure.ac:37: installing `src/build-aux/install-sh'
configure.ac:37: installing `src/build-aux/missing'
src/Makefile.am: installing `src/build-aux/depcomp'
$
```

Skript *autogen.sh* vytváří množinu automaticky konfigurovaných skriptů, které budou zkoumat váš systém, aby objevily správné nastavení a ujistily vás, že máte nainstalované všechny knihovny nutné pro kompilaci zdrojových kódů. Nejdůležitější z nich je skript configure, který nabízí mnoho různých možností, jak upravit kompilační proces. Napište ./configure --help a zobrazí se přehled různých

možností:

```
$ ./configure --help

`configure' configures Bitcoin Core 0.9.0 to adapt to many kinds of systems.

Usage: ./configure [OPTION]... [VAR=VALUE]...

To assign environment variables (e.g., CC, CFLAGS...), specify them as
VAR=VALUE. See below for descriptions of some of the useful variables.

Defaults for the options are specified in brackets.

Configuration:
  -h, --help display this help and exit
      --help=short display options specific to this package
      --help=recursive display the short help of all the included packages
  -V, --version display version information and exit

[... následuje přehled mnoha dalších nastavení a proměnných ...]

Optional Features:
  --disable-option-checking ignore unrecognized --enable/--with options
  --disable-FEATURE do not include FEATURE (same as --enable-FEATURE=no)
  --enable-FEATURE[=ARG] include FEATURE [ARG=yes]

[... další nastavení ...]

Use these variables to override the choices made by `configure' or to help
it to find libraries and programs with nonstandard names/locations.

Report bugs to <info@bitcoin.org>.

$
```

Skript configure vám umožňuje zapnout nebo vypnout jisté funkce bitcoind pomocí přepínačů --enable-FEATURE a --disable-FEATURE, kde FEATURE je nahrazeno jménem funkce, jejichž seznam je uveden ve výstupu nápovědy. V této kapitole budeme kompilovat bitcoind klienta se standardním nastavením funkcí. Nebudeme používat konfigurační přepínače, ale vy byste si měli jejich seznam prohlédnout, abyste pochopili možné funkce, které jsou součástí klienta. Dále spustíte skript configure, který automaticky objeví všechny nezbytné knihovny a vytvoří kompilační skript na míru vašemu systému.

```
$ ./configure
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes

[... následuje testování mnoha dalších systémových funkcí ...]

configure: creating ./config.status
config.status: creating Makefile
config.status: creating src/Makefile
config.status: creating src/test/Makefile
config.status: creating src/qt/Makefile
config.status: creating src/qt/test/Makefile
config.status: creating share/setup.nsi
config.status: creating share/qt/Info.plist
config.status: creating qa/pull-tester/run-bitcoind-for-test.sh
config.status: creating qa/pull-tester/build-tests.sh
config.status: creating src/bitcoin-config.h
config.status: executing depfiles commands
$
```

Pokud půjde vše dobře, příkaz configure skončí vytvořením pro váš systém upraveného kompilačního skriptu, který umožní kompilaci bitcoined. Pokud chybí některá z knihoven nebo nastanou chyby, příkaz configure je ukončen s chybou místo vytvoření kompilačního skriptu. Pokud nastanou chyby, je to nejčastěji kvůli chybějícím nebo nekompatibilním knihovnám. Znovu se podívejte do kompilační dokumentace a ujistěte se, že máte nainstalovány chybějící knihovny. Poté spusťte znovu configure a podívejte se, zda jsou chyby opraveny. Dále zkompilujete zdrojové kódy, tento proces může trvat až hodinu než se dokončí. Během kompilačního procesu sledujte výstup každých pár sekund nebo každých pár minut. Pokud se něco nezdaří, zobrazí se chyby. Kompilační proces může být kdykoliv obnoven, pokud byl přerušen. Napište make pro zahájení kompilace

```
$ make
Making all in src
make[1]: Entering directory `/home/ubuntu/bitcoin/src'
make all-recursive
make[2]: Entering directory `/home/ubuntu/bitcoin/src'
Making all in .
make[3]: Entering directory `/home/ubuntu/bitcoin/src'
  CXX addrman.o
  CXX alert.o
  CXX rpcserver.o
  CXX bloom.o
  CXX chainparams.o

[... následuje mnoho dalších zpráv překladače ...]

  CXX test_bitcoin-wallet_tests.o
  CXX test_bitcoin-rpc_wallet_tests.o
  CXXLD test_bitcoin
make[4]: Leaving directory `/home/ubuntu/bitcoin/src/test'
make[3]: Leaving directory `/home/ubuntu/bitcoin/src/test'
make[2]: Leaving directory `/home/ubuntu/bitcoin/src'
make[1]: Leaving directory `/home/ubuntu/bitcoin/src'
make[1]: Entering directory `/home/ubuntu/bitcoin'
make[1]: Nothing to be done for `all-am'.
make[1]: Leaving directory `/home/ubuntu/bitcoin'
$
```

Pokud jde všechno dobře, bitcoind je nyní zkompilován. Závěrečným krokem instalace bitcoind je přidání spustitelného souboru do systémové cesty za použití příkazu make

```
$ sudo make install
Making install in src
Making install in .
 /bin/mkdir -p '/usr/local/bin'
  /usr/bin/install -c bitcoind bitcoin-cli '/usr/local/bin'
Making install in test
make install-am
 /bin/mkdir -p '/usr/local/bin'
  /usr/bin/install -c test_bitcoin '/usr/local/bin'
$
```

Můžete se ujistit, že bitcoin je správně nainstalován dotazem systému na cestu dvou spustitelných souborů, jak je uvedeno dále.

```
$ which bitcoind
/usr/local/bin/bitcoind

$ which bitcoin-cli
/usr/local/bin/bitcoin-cli
```

Obvyklá instalace bitcoind jí umísťuje do */usr/local/bin*. Při prvním spuštění bitcoind, vám bude připomenuto, abyste vytvořili konfigurační soubor se silným heslem pro JSON-RPC rozhraní. Spustíte bitcoind napsáním bitcoind do příkazového řádku.

```
$ bitcoind
Error: To use the "-server" option, you must set a rpcpassword in the configuration file:
/home/ubuntu/.bitcoin/bitcoin.conf
It is recommended you use the following random password:
rpcuser=bitcoinrpc
rpcpassword=2XA4DuKNCbtZXsBQRRNDEwEY2nM6M4H9Tx5dFjoAVVbK
(you do not need to remember this password)
The username and password MUST NOT be the same.
If the file does not exist, create it with owner-readable-only file permissions.
It is also recommended to set alertnotify so you are notified of problems;
for example: alertnotify=echo %s | mail -s "Bitcoin Alert" admin@foo.com
```

Upravte konfigurační soubor ve vámi oblíbeném editoru a nastavte parametry, změňte heslo za silné heslo, jak bylo doporučeno bitcoind. *Nepoužívejte* heslo, které zde vidíte. V adresáři *.bitcoin* vytvořte soubor pojmenovaný *.bitcoin/bitcoin.conf* a zadejte uživatelské jméno a heslo:

```
rpcuser=bitcoinrpc
rpcpassword=2XA4DuKNCbtZXsBQRRNDEwEY2nM6M4H9Tx5dFjoAVVbK
```

Při úpravách konfiguračního souboru, můžete nastavit i několik dalších nastavení jako txindex (viz Index transakční databáze a txindex nastavení). Příkaz bitcoind --help. zobrazí úplný seznam dostupných možností nastavení.

Nyní spustíme Bitcoin Core klienta. Při prvním spuštění bude postaven bitcoin blockchain stažením všech bloků. To je soubor o velikosti několika gigabytů a jeho stažení obvykle trvá dva dny. Tento proces můžete zrychlit "blockchains","downloading with bittorrent clients") stažením částečné kopie blockchainu pomocí BitTorrent klienta z adresy SourceForge.

Spusťte bitcoind na pozadí s nastavením -daemon:

```
$ bitcoind -daemon

Bitcoin version v0.9.0rc1-beta (2014-01-31 09:30:15 +0100)
Using OpenSSL version OpenSSL 1.0.1c 10 May 2012
Default data directory /home/bitcoin/.bitcoin
Using data directory /bitcoin/
Using at most 4 connections (1024 file descriptors available)
init message: Verifying wallet...
dbenv.open LogDir=/bitcoin/database ErrorFile=/bitcoin/db.log
Bound to [::]:8333
Bound to 0.0.0.0:8333
init message: Loading block index...
Opening LevelDB in /bitcoin/blocks/index
Opened LevelDB successfully
Opening LevelDB in /bitcoin/chainstate
Opened LevelDB successfully

[... následují další startovní zprávy ...]
```

Použití Bitcoin Core JSON-RPC API z příkazové řádky

Bitcoin Core klient implementuje rozhraní JSON-RPC , které může být používáno pomocníkem bitcoin-cli spouštěným z příkazové řádky. Příkazová řádka nám umožňuje interaktivně experimentovat s možnostmi, které jsou také programově přístupné přes API. Na začátku vyvolejte příkaz help a prohlédněte si seznam dostupných RPC příkazů.

```
$ bitcoin-cli help
addmultisigaddress nrequired ["key",...] ( "account" )
addnode "node" "add|remove|onetry"
backupwallet "destination"
createmultisig nrequired ["key",...]
createrawtransaction [{"txid":"id","vout":n},...] {"address":amount,...}
decoderawtransaction "hexstring"
decodescript "hex"
dumpprivkey "bitcoinaddress"
dumpwallet "filename"
getaccount "bitcoinaddress"
getaccountaddress "account"
getaddednodeinfo dns ( "node" )
getaddressesbyaccount "account"
getbalance ( "account" minconf )
getbestblockhash
getblock "hash" ( verbose )
getblockchaininfo
getblockcount
```

```
getblockhash index
getblocktemplate ( "jsonrequestobject" )
getconnectioncount
getdifficulty
getgenerate
gethashespersec
getinfo
getmininginfo
getnettotals
getnetworkhashps ( blocks height )
getnetworkinfo
getnewaddress ( "account" )
getpeerinfo
getrawchangeaddress
getrawmempool ( verbose )
getrawtransaction "txid" ( verbose )
getreceivedbyaccount "account" ( minconf )
getreceivedbyaddress "bitcoinaddress" ( minconf )
gettransaction "txid"
gettxout "txid" n ( includemempool )
gettxoutsetinfo
getunconfirmedbalance
getwalletinfo
getwork ( "data" )
help ( "command" )
importprivkey "bitcoinprivkey" ( "label" rescan )
importwallet "filename"
keypoolrefill ( newsize )
listaccounts ( minconf )
listaddressgroupings
listlockunspent
listreceivedbyaccount ( minconf includeempty )
listreceivedbyaddress ( minconf includeempty )
listsinceblock ( "blockhash" target-confirmations )
listtransactions ( "account" count from )
listunspent ( minconf maxconf ["address",...] )
lockunspent unlock [{"txid":"txid","vout":n},...]
move "fromaccount" "toaccount" amount ( minconf "comment" )
ping
sendfrom "fromaccount" "tobitcoinaddress" amount ( minconf "comment" "comment-to" )
sendmany "fromaccount" {"address":amount,...} ( minconf "comment" )
sendrawtransaction "hexstring" ( allowhighfees )
sendtoaddress "bitcoinaddress" amount ( "comment" "comment-to" )
setaccount "bitcoinaddress" "account"
setgenerate generate ( genproclimit )
settxfee amount
signmessage "bitcoinaddress" "message"
signrawtransaction "hexstring" (
```

```
[{"txid":"id","vout":n,"scriptPubKey":"hex","redeemScript":"hex"},...]
["privatekey1",...] sighashtype )
stop
submitblock "hexdata" ( "jsonparametersobject" )
validateaddress "bitcoinaddress"
verifychain ( checklevel numblocks )
verifymessage "bitcoinaddress" "signature" "message"
walletlock
walletpassphrase "passphrase" timeout
walletpassphrasechange "oldpassphrase" "newpassphrase"
```

Získání informací o stavu Bitcoin Core klienta

Příkaz: getinfo

Bitcoinový RPC příkaz getinfo zobrazuje základní informace o stavu uzlu bitcoinové sítě, peněženky a blockchainové databáze. Použijte bitcoin-cli pro jeho spuštění:

```
$ bitcoin-cli getinfo
```

```
{
    "version" : 90000,
    "protocolversion" : 70002,
    "walletversion" : 60000,
    "balance" : 0.00000000,
    "blocks" : 286216,
    "timeoffset" : -72,
    "connections" : 4,
    "proxy" : "",
    "difficulty" : 2621404453.06461525,
    "testnet" : false,
    "keypoololdest" : 1374553827,
    "keypoolsize" : 101,
    "paytxfee" : 0.00000000,
    "errors" : ""
}
```

Data jsou vrácená ve formátu objektů JavaSkriptu (JSON). Tento formát může být snadno zpracován všemi programovacími jazyky, ale je zároveň i relativně dobře lidsky čitelný. Mezi těmito daty vidíme čísla verzí bitcoinových softwarových klientů (90000), protokolů (70002) a peněženek (60000). Vidíme současný stav peněženky, který je nula. Vidíme aktuální výšku bloku, ukazující kolik bloků zná náš klient (286216). Vidíme různé statistiky o bitcoinové síti a nastavení týkající se tohoto klienta. Toto nastavení prozkoumáme detailněji ve zbytku této kapitoly.

TIP	Bude trvat nějaký čas, možná déle než den, než bitcoind klient "dostihne" aktuální výšku blockchainu stahováním bloků od jiných bitcoinových klientů. Tento proces můžete kontrolovat za použití getinfo, které zobrazí počet známých bloků.

Nastavení peněženky a šifrování

Příkazy: encryptwallet, walletpassphrase

Předtím než přistoupíte k tvorbě klíčů a dalším příkazům, měli byste nejprve zašifrovat peněženku heslem. V tomto příkladě budeme používat příkaz encryptwallet s heslem "foo". Samozřejmě, nahraďte "foo" silným a složitým heslem!

```
$ bitcoin-cli encryptwallet foo
wallet encrypted; Bitcoin server stopping, restart to run with encrypted wallet. The
keypool has been flushed, you need to make a new backup.
$
```

Můžete ověřit, že peněženka byla zašifrována opětovným spuštěním getinfo. Tentokrát uvidíte nový záznam nazvaný unlocked_until. Tento čítač ukazuje, jak dlouho bude dešifrovací heslo k peněžence uloženo v paměti, čímž bude peněženka udržována odemčená. Jakmile se čítač vynuluje, peněženka je zamčena.

```
$ bitcoin-cli getinfo
```

```
{
    "version" : 90000,

#[... další informace...]

    "unlocked_until" : 0,
    "errors" : ""
}
$
```

Pro odemčení peněženky vydejte příkaz walletpassphrase , který má dva parametry - heslo a počet sekund do opětovného automatického uzamčení peněženky (časový čítač)

```
$ bitcoin-cli walletpassphrase foo 360
$
```

Můžete potvrdit odemčení peněženky a sledovat odpočet opětovným spuštěním příkazu getinfo.

```
$ bitcoin-cli getinfo
```

```
{
    "version" : 90000,

#[... other information...]

    "unlocked_until" : 1392580909,
    "errors" : ""
}
```

Zálohování peněženky, textový výpis a obnova

Příkazy: backupwallet, importwallet, dumpwallet

Dále budeme procvičovat vytváření záložního souboru peněženky a následnou obnovu peněženky z tohoto souboru. Použijeme příkaz backupwallet pro zálohování. Parametrem příkazu je jméno souboru. V našem příkladě zálohujeme peněženku do souboru *wallet.backup*:

```
$ bitcoin-cli backupwallet wallet.backup
$
```

Nyní obnovíme záložní soubor, použijeme příkaz importwallet. Pokud je peněženka zamčena, musíte ji nejprve odemknout (viz walletpassphrase v předchozí části), abyste mohli importovat záložní soubor.

```
$ bitcoin-cli importwallet wallet.backup
$
```

Příkaz dumpwallet může být použit pro uložení peněženky v textové, lidsky čitelné podobě.

```
$ bitcoin-cli dumpwallet wallet.txt
$ more wallet.txt
# Wallet dump created by Bitcoin v0.9.0rc1-beta (2014-01-31 09:30:15 +0100)
# * Created on 2014-02- 8dT20:34:55Z
# * Best block at time of backup was 286234
(000000000000000f74f0bc9d3c186267bc45c7b91c49a0386538ac24c0d3a44),
# mined on 2014-02- 8dT20:24:01Z

KzTg2wn6Z8s7ai5NA9MVX4vstHRsqP26QKJCzLg4JvFrp6mMaGB9 2013-07- 4dT04:30:27Z change=1 #
addr=16pJ6XkwSQv5ma5FSXMRPaXEYrENCEg47F
Kz3dVz7R6mUpXzdZy4gJEVZxXJwA15f198eVui4CUivXotzLBDKY 2013-07- 4dT04:30:27Z change=1 #
addr=17oJds8kaN8LP8kuAkWTco6ZM7BGXFC3gk
[... mnoho dalších klíčů ...]

$
```

Adresy peněženek a příchozí transakce

Příkazy: getnewaddress, getreceivedbyaddress, listtransactions, getaddressesbyaccount, getbalance

Bitcoin reference klient udržuje seznam adres, jehož velikost je zobrazena keypoolsize když použijete příkaz getinfo. Tyto adresy jsou vytvářeny automaticky a mohou být použity jako veřejné přijímací adresy nebo adresy pro vratky. Pro získání jedné z těchto adres, použijte příkaz getnewaddress:

```
$ bitcoin-cli getnewaddress
1hvzSofGwT8cjb8JU7nBsCSfEVQX5u9CL
```

Nyní můžeme tuto adresu použít k zaslání malého množství bitcoinů do naší bitcoind peněženky z externí peněženky (předpokládáme, že máme nějaké bitcoiny ve směnárně, webové peněžence nebo jiné bitcoind peněžence držené jinde). V tomto příkladě, pošleme 50 milibitů (0,050 BTC) na uvedenou adresu.

Můžeme se dotázat bitcoind klienta na částku obdrženou touto adresou a určit kolik potvrzení požadujeme před tím, než je částka započítána do jejího stavu. V tomto příkladě jsme určili nula potvrzení. Za pár vteřin po odeslání bitcoinů z jiné peněženky je uvidíme v naší peněžence. Použijeme příkaz getreceivedbyaddress s parametry adresou a počtem potvrzení nastaveným na nula (0):

```
$ bitcoin-cli getreceivedbyaddress 1hvzSofGwT8cjb8JU7nBsCSfEVQX5u9CL 0
0.05000000
```

Pokud odstraníme nulu z konce tohoto příkazu, uvidíme pouze částky, které mají alespoň minconf potvrzení, kde minconf je nastavení minimálního počtu potvrzení nutných k započítání transakce do stavu peněženky. Nastavení minconf je určeno v bitcoind konfiguračním souboru. Protože transakce

posílající tyto bitcoiny byla odeslána teprve před pár sekundami, má stále nula potvrzení a proto uvidíme v peněžence nulový stav.

```
$ bitcoin-cli getreceivedbyaddress 1hvzSofGwT8cjb8JU7nBsCSfEVQX5u9CL
0.00000000
```

Transakce obdržené celou peněženkou mohou být také zobrazeny za použití příkazu listtransactions:

```
$ bitcoin-cli listtransactions
```

```
[
    {
        "account" : "",
        "address" : "1hvzSofGwT8cjb8JU7nBsCSfEVQX5u9CL",
        "category" : "receive",
        "amount" : 0.05000000,
        "confirmations" : 0,
        "txid" : "9ca8f969bd3ef5ec2a8685660fdbf7a8bd365524c2e1fc66c309acbae2c14ae3",
        "time" : 1392660908,
        "timereceived" : 1392660908
    }
]
```

Můžeme vypsat všechny adresy celé peněženky použitím příkazu getaddressesbyaccount

```
$ bitcoin-cli getaddressesbyaccount ""
```

```
[
    "1LQoTPYy1TyERbNV4zZbhEmgyfAipC6eqL",
    "17vrg8uwMQUibkvS2ECRX4zpcVJ78iFaZS",
    "1FvRHWhHBBZA8cGRRsGiAeqEzUmjJkJQWR",
    "1NVJK3JsL41BF1KyxrUyJW5XHjunjfp2jz",
    "14MZqqzCxjc99M5ipsQSRfieT7qPZcM7Df",
    "1BhrGvtKFjTAhGdPGbrEwP3xvFjkJBuFCa",
    "15nem8CX91XtQE8B1Hdv97jE8X44H3DQMT",
    "1Q3q6taTsUiv3mMemEuQQJ9sGLEGaSjo81",
    "1HoSiTg8sb16oE6SrmazQEwcGEv8obv9ns",
    "13fE8BGhBvnoy68yZKuWJ2hheYKovSDjqM",
    "1hvzSofGwT8cjb8JU7nBsCSfEVQX5u9CL",
    "1KHUmVfCJteJ21LmRXHSpPoe23rXKifAb2",
    "1LqJZz1D9yHxG4cLkdujnqG5jNNGmPeAMD"
]
```

Nakonec, příkaz getbalance ukáže celkový stav peněženky, přidáním všech transakcí majících nejméně minconf potvrzení:

```
$ bitcoin-cli getbalance
0.05000000
```

TIP	Pokud transakce nebyla ještě potvrzena, stav vrácený getbalance bude nula. Nastavení konfigurace "minconf" udává minimální počet potvrzení nutných k tomu, aby transakce byla započítána do stavu peněženky.

Prozkoumání a dekódování transakcí

Příkazy: gettransaction, getrawtransaction, decoderawtransaction

Prozkoumáme příchozí transakci, která byla předtím vypsána příkazem gettransaction. Můžeme získat transakci podle jejího transakčního haše ukázaného dříve v txid s příkazem gettransaction:

```
{
    "amount" : 0.05000000,
    "confirmations" : 0,
    "txid" : "9ca8f969bd3ef5ec2a8685660fdbf7a8bd365524c2e1fc66c309acbae2c14ae3",
    "time" : 1392660908,
    "timereceived" : 1392660908,
    "details" : [
        {
            "account" : "",
            "address" : "1hvzSofGwT8cjb8JU7nBsCSfEVQX5u9CL",
            "category" : "receive",
            "amount" : 0.05000000
        }
    ]
}
```

TIP Transakční ID nejsou spolehlivá dokud není transakce potvrzena. Chybějící transakční haš v blockchainu neznamená, že transakce nebude zpracována. Toto je známo jako "transakční ohebnost", protože transakční haš může být změněn před potvrzením transakce v bloku. Po potvrzení, transakční ID je neměnné a spolehlivé.

Transakční formát zobrazený příkazem gettransaction je zjednodušeným formátem. Pro získání plného zdrojového kódu transakce slouží dva příkazy: getrawtransaction a decoderawtransaction. První z nich getrawtransaction má jako parametr *haš transakce (txid)* a vrací celou transakci jako "syrový" hexadecimální řetězec, přesně takový jaký existuje na bitcoinové síti.

Pro dekódování tohoto hexadecimálního řetězce, použijeme příkaz decoderawtransaction. Okopírujeme a vložíme hexadecimální řetězec jako jeho první parametr. Získáme úplný obsah zpracovaný do JSON datové struktury (z formátovacích důvodů je hexadecimální řetězec v předchozím příkladu zkrácen):

Dekódovaná transakce ukazuje všechny součásti této transakce, včetně transakčních vstupů a výstupů. V tomto případě vidíme transakci, která zvýšila balanci na nové adrese o 50 milibitů. Transakce využívá jeden vstup a vytvářející dva výstupy. Vstup této transakce byl výstupem předtím potvrzené transakce (zobrazeno jako vin txid začínající na d3c7). Dva výstupy odpovídají 50 milibitovému vkladu a vratce zbytku hodnoty odesilateli.

Můžeme dále prozkoumat blockchain prozkoumáním předchozích transakcí na které je odkazováno pomocí jejich txid v této transakci za použití stejných příkazů (např. gettransaction). Skákáním z transakce do transakce můžeme sledovat tok řetězce transakcí zpátky tak jak hodnota byla přenášena z vlastníka adresy na vlastníka adresy.

Jakmile transakce, kterou jsme obdrželi, byla potvrzena vložením do bloku, příkaz gettransaction vrátí dodatečné informace, ukazuje *haš bloku* do kterého byla transakce vložena.

Zde vidíme nové informace v záznamech blockhash (haš bloku, do kterého byla transakce vložena) a blockindex s hodnotou 18 (znamenající, že naše transakce byla 18-tou transakcí v tomto bloku).

Index transakční databáze a txindex nastavení

Standardně, Bitcoin Core vytváří databázi obsahující *pouze* transakce vztahující se k uživatelově peněžence. Pokud chcete být schopní přistupovat k *libovolné* transakci pomocí příkazů jako gettransaction, musíte nastavit Bitcoin Core, aby vytvořil úplný transakční index, čehož dosáhneme pomocí volby txindex. Nastavte txindex=1 v the Bitcoin Core konfiguračním souboru (obvykle ho najdete v domovském adresáři v souboru *.bitcoin/bitcoin.conf*). Jakmile je parametr změněn, musíte restartovat bitcoind a počkat na přestavění indexu.

Průzkum bloků

Příkazy: getblock, getblockhash

Nyní, když víme, ve kterém bloku je naše transakce zahrnuta, můžeme se dotázat na tento blok. Použijeme příkaz getblock s hašem bloku jako parametrem:

Blok obsahuje 367 transakcí a jak můžete vidět, 18-tá zobrazená transakce (9ca8f9...) má txid transakce, která přidává 50 milibitů na naší adresu. Záznam height nám udává, že se jedná o 286384-tý blok v blockchainu

Můžeme také získat blok pomocí jeho hloubky, za použití příkazu getblockhash, který jako svůj parametr má hloubku bloku a vrací haš tohoto bloku.

Nyní získáme haš "základního bloku", prvního bloku vytěženého Satoshi Nakamotem, s hloubkou nula. Získaný blok vypadá:

Příkazy getblock, getblockhash, a gettransaction mohou být použity k programovému prozkoumání blockchainové databáze.

Vytváření, podepisování a odesílání transakcí založených na hesle :[<phrase role="keep-together">Unspent Outputs</phrase>]

Příkazy: listunspent, gettxout, createrawtransaction, decoderawtransaction, signrawtransaction, sendrawtransaction

Bitcoinové transakce jsou založeny na konceptu utrácení "výstupů", které jsou výsledky předchozích transakcí, což vytváří řetěz transakcí který převádí vlastnictví z adresy na adresu. Naše peněženka nyní získala transakci, která přiřazuje jeden takovýto výstup naší adrese. Jakmile je potvrzena, můžeme utratit její výstup.

Nejprve použijeme příkaz listunspent, který zobrazuje všechny neutracené *potvrzené* výstupy v naší peněžence:

```
$ bitcoin-cli listunspent
```

Vidíme, že transakce 9ca8f9... vytvořila výstup (s indexem 0) přiřazující adrese 1hvzSo... částku 50 milibitů, která v tuto chvíli získala sedm potvrzení. Transakce používají dříve vytvořené výstupy jako své vstupy pomocí odkazování na předchozí txid a vout index. Vytvoříme transakci, která utratí 0tý vout transakce txid 9ca8f9... jako svůj vstup a přiřadí jej novému výstupu, který odesílá hodnotu na novou adresu.

Nejprve se podíváme na konkrétní výstup pro více podrobností. Použijeme gettxout k získání podrobností o tomto neutraceném výstupu. Transakční výstupy jsou vždy odkazovány pomocí txid a vout a tyto dva parametry předáme gettxout:

Vidíme zde výstup, který přiřazuje 50 milibitů naší adrese 1hvz.... K utracení tohoto výstupu vytvoříme novou transakci. Nejprve vytvoříme adresu, na kterou zašleme peníze.

```
$ bitcoin-cli getnewaddress
1LnfTndy3qzXGN19Jwscj1T8LR3MVe3JDb
```

Pošleme 25 milibitů na novou adresu 1LnfTn..., kterou jsme právě vytvořili v naší peněžence. V naší nové transakci utratíme 50 milibitový výstup a zašleme 25 milibitů na tuto novou adresu. Protože musíme utratit celý výstup předchozí transakce, musíme vytvořit i vratku. Vytvoříme vratku na adresu 1hvz... posláním zbývající hodnoty na tuto původní adresu. Nakonec musíme zaplatit poplatek za transakci. Pro zaplacení poplatku snížíme výstup vratky o 0,5 milibitů, hodnota vratky bude 24,5 milibitů. Rozdíl mezi součtem nových výstupů (25 mBTC + 24.5 mBTC = 49.5 mBTC) a součtem vstupů (50 mBTC) bude vyzvednut jako poplatek za transakci těžařem.

Použijeme createrawtransaction pro vytvoření této transakce. Jako parametry poskytneme transakční vstup (50 milibitů neutraceného výstupu z předchozí potvrzené transakce) a dvou transakčních výstupů (hodnota zaslaná na novou adresu a vratka poslaná zpátky na původní adresu)

Příkaz createrawtransaction vytváří nový syrový hexadecimální řetězec, který kóduje transakci, jejíž podrobnosti jsme poskytli. Podíváme se, zda je vše v pořádku pomocí příkazu decoderawtransaction, který dekóduje tento syrový řetězec.

Vypadá správně! Naše nová transakce "spotřebuje" neutracený výstup předchozí potvrzené transakce a utratí ho ve prospěch dvou výstupů, jednoho o velikosti 25 milibitů na naší novou adresu a druhého (vratky) o velikosti 24,5 milibitů na původní adresu. Rozdíl 0,5 milibitů reprezentuje transakční poplatek, který si vyzvedne těžař, který najde blok obsahující naší transakci.

Jak můžete pozorovat, transakce obsahuje prázdný scriptSig, protože jsme jí ještě nepodepsali. Bez podpisu je tato transakce bezvýznamná. Ještě jsme neprokázali, že _vlastníme_adresu, která neutracený výstup poskytuje. Podpisem odstraníme zámek na výstupu a prokážeme, že vlastníme výstup a můžeme ho utratit. Použijeme příkaz signrawtransaction k podepsání transakce. Parametrem je syrová transakce ve formě hexadecimálního řetězce.

TIP	Zašifrovaná peněženka musí být odemčena před tím, než je transakce podepsána, protože podepsání transakce vyžaduje přístup k soukromému klíči peněženky.

Příkaz signrawtransaction vrací jinou hexadecimálně zakódovanou syrovou transakci. Dekódujeme ji příkazem decoderawtransaction abychom viděli, co se změnilo:

Nyní, vstupy použité v transakci obsahují scriptSig, což je digitální podpis prokazující vlastnictví adresy 1hvz... a odstraňující uzamčení výstupu, takže může být utracen. Tento podpis dělá transakci ověřitelnou jakýmkoliv uzlem bitcoinové sítě.

Nyní je čas odeslat nově vytvořenou transakci do sítě pomocí příkazu sendrawtransaction, kterému předáme jako parametr hexadecimální řetězec vytvořený příkazem signrawtransaction, jedná se o stejný řetězec, který jsme právě dekódovali.

Příkaz sendrawtransaction vrací *transakční haš (txid)* jakmile odešle transakci do sítě. Můžeme se pomocí příkazu gettransaction dotázat na transakci s tímto ID:

```
{
    "amount" : 0.00000000,
    "fee" : -0.00050000,
    "confirmations" : 0,
    "txid" : "ae74538baa914f3799081ba78429d5d84f36a0127438e9f721dff584ac17b346",
    "time" : 1392666702,
    "timereceived" : 1392666702,
    "details" : [
        {
            "account" : "",
            "address" : "1LnfTndy3qzXGN19Jwscj1T8LR3MVe3JDb",
            "category" : "send",
            "amount" : -0.02500000,
            "fee" : -0.00050000
        },
        {
            "account" : "",
            "address" : "1hvzSofGwT8cjb8JU7nBsCSfEVQX5u9CL",
            "category" : "send",
            "amount" : -0.02450000,
            "fee" : -0.00050000
        },
        {
            "account" : "",
            "address" : "1LnfTndy3qzXGN19Jwscj1T8LR3MVe3JDb",
            "category" : "receive",
            "amount" : 0.02500000
        },
        {
            "account" : "",
            "address" : "1hvzSofGwT8cjb8JU7nBsCSfEVQX5u9CL",
            "category" : "receive",
            "amount" : 0.02450000
        }
    ]
}
```

Jako předtím, můžeme prozkoumat transakci detailněji příkazy getrawtransaction a decodetransaction. Tyto příkazy vrátí přesně ten samý hexadecimální řetězec, který jsme vyrobili a dekódovali předtím, než jsme ho odeslali do sítě.

Alternativní klienti, knihovny a nástroje

Kromě referenčního klienta (bitcoind) existují další klienti a knihovny, které mohou být použity pro komunikaci s bitcoinovou sítí a datovými strukturami. Jsou implementováni v různých

programovacích jazycích, nabízejí programátorům přirozená rozhraní v jejich vlastních jazycích.

Mezi alternativní implementace patří:

libbitcoin
: Bitcoinová mezi-platformní vývojářská sada nástrojů v C++.

bitcoin explorer
: Bitcoinový nástroj z příkazové řádky

bitcoin server
: Bitcoinový úplný uzel a dotazovací server

bitcoinj
: Java úplný klient a knihovna

btcd
: Úplný bitcoinový klient v jazyce Go

Bits of Proof (BOP)
: Java podniková implementace bitcoinu

picocoin
: Odlehčený klient a knihovna pro bitcoin v jazyce C

pybitcointools
: Python bitcoin knihovna

pycoin
: Další python bitcoin knihovna

Existuje mnoho dalších knihoven v mnoha jiných programovacích jazycích a stále vznikají nové.

Libbitcoin a Bitcoinový Průzkumník

Knihovna libbitcoin je mezi-platformní vývojářská sada nástrojů v C++, která podporuje libbitcoin server s úplným uzlem a Bitcoinového Průzkumníka (v originále Bitcoin Explorer, zkratka bx), který je nástrojem ovládaným z příkazové řádky.

Příkazy bx nabízí mnoho stejných možností jako příkazy bitcoind klienta, které jsme si ukázali v této kapitole. Příkazy bx také nabízí nástroje na správu klíčů, které nejsou nabízeny bitcoind, včetně typu-2 deterministických klíčů a mnemotechnické kódování klíčů, stejně tak jako skryté adresy, platba a dotazovací podporu.

Instalace bitcoinového průzkumníka

Pro použití bitcoinového průzkumníka, jednoduše download the signed executable for your operating system. Jsou dostupné verze pro hlavní i testovací síť pro Linux, OS X a Windows.

Napište bx bez parametrů a zobrazí se seznam dostupných příkazů (viz [appdx_bx]).

Bitcoinový průzkumník nabízí instalátor pro building from sources on Linux and OS X, as well as Visual Studio projects for Windows. Zdrojové kódy mohou být zkompilovány ručně pomocí Autotools. Toto nainstaluje i závislostní knihovnu libbitcoin.

TIP	Bitcoinový průzkumník nabízí mnoho užitečných příkazů pro kódování a dekódování adres a konverzi mezi různými formáty a reprezentacemi. Použijte ho pro prozkoumání různých formátů jako Base16 (hexadecimální), Base58, Base58Check, Base64, atd.

Instalace Libbitcoin

Knihovna libbitcoin nabízí instalátor building from sources on Linux and OS X, as well as Visual Studio projects for Windows. Zdrojové kódy mohou být kompilovány ručně za použití Autotools.

TIP	Instalátor Bitcoinového průzkumníka nainstaluje jak bx, tak i knihovnu libbitcoin. Pokud jste zkompilovali bx ze zdrojových kódů, můžete přeskočit tento krok.

pycoin

Python knihovna *pycoin*, originálně napsaná a spravovaná Richardem Kissem, slouží ke správě bitcoinových klíčů a zpracování transakcí, dokonce podporuje skriptovací jazyk, který je dostatečně silný na správné zpracování nestandardních transakcí.

Knihovna pycoin podporuje jak Python 2 (2.7.x), tak i Python 3 (od 3.3) a přináší nástroje ovládané z příkazové řádky ku a tx. Pro instalaci pycoin 0.42 pod Python 3 ve virtuálním prostředí (venv) použijeme následující příkazy:

```
$ python3 -m venv /tmp/pycoin
$ . /tmp/pycoin/bin/activate
$ pip install pycoin==0.42
Downloading/unpacking pycoin==0.42
  Downloading pycoin-0.42.tar.gz (66kB): 66kB downloaded
  Running setup.py (path:/tmp/pycoin/build/pycoin/setup.py) egg_info for package
pycoin

Installing collected packages: pycoin
  Running setup.py install for pycoin

    Installing tx script to /tmp/pycoin/bin
    Installing cache_tx script to /tmp/pycoin/bin
    Installing bu script to /tmp/pycoin/bin
    Installing fetch_unspent script to /tmp/pycoin/bin
    Installing block script to /tmp/pycoin/bin
    Installing spend script to /tmp/pycoin/bin
    Installing ku script to /tmp/pycoin/bin
    Installing genwallet script to /tmp/pycoin/bin
Successfully installed pycoin
Cleaning up...
$
```

Zde je příklad Python skriptu, který vyzvedne a utratí nějaké bitcoiny pomocí knihovny pycoin.

```
#!/usr/bin/env python

from pycoin.key import Key

from pycoin.key.validate import is_address_valid, is_wif_valid
from pycoin.services import spendables_for_address
from pycoin.tx.tx_utils import create_signed_tx

def get_address(which):
    while 1:
        print("enter the %s address=> " % which, end='')
        address = input()
        is_valid = is_address_valid(address)
        if is_valid:
            return address
        print("invalid address, please try again")

src_address = get_address("source")
spendables = spendables_for_address(src_address)
print(spendables)

while 1:
    print("enter the WIF for %s=> " % src_address, end='')
    wif = input()
    is_valid = is_wif_valid(wif)
    if is_valid:
        break
    print("invalid wif, please try again")

key = Key.from_text(wif)
if src_address not in (key.address(use_uncompressed=False),
key.address(use_uncompressed=True)):
    print("** WIF doesn't correspond to %s" % src_address)
print("The secret exponent is %d" % key.secret_exponent())

dst_address = get_address("destination")

tx = create_signed_tx(spendables, payables=[dst_address], wifs=[wif])

print("here is the signed output transaction")
print(tx.as_hex())
```

Příklad použití příkazů příkazové řádky ku a tx nalezneme [appdxbitcoinimpproposals].

btcd

Implementace úplného bitcoinového uzlu napsaná v jazyce Go se nazývá btcd. V současné době stahuje, ověřuje a slouží blockchainu pomocí stejných pravidel (včetně chyb) pro přijetí bloku jako referenční klient bitcoind. Také správně propaguje nově vytěžené bloky, udržuje dočasné úložiště transakcí a propaguje transakce, které dosud nebyly vloženy do bloku. Zajišťuje, že transakce přijaté do dočasného úložiště dodržují požadovaná pravidla a také obsahuje velkou většinu ostatních přísných kontrol, které rozdělují transakce dle požadavků těžařů ("standardní" transakce).

Mezi btcd a bitcoind je jeden významný rozdíl, btcd neobsahuje funkčnost peněženky, což bylo záměrem jeho tvůrců. To znamená, že nemůžete přijímat nebo vytvářet platby přímo pomocí btcd. Tato funkčnost je poskytována projekty btcwalled a btcgui, které jsou ve stádiu vývoje. Dalším významným rozdílem mezi btcd a bitcoind je, že btcd podporuje jak HTTP POST požadavek (stejně jako bitcoind), ale preferuje Websockets a skutečnost, že btcd RPC spojení mají standardně povolené TLS.

Instalace btcd

K nainstalovaní btcd pro Windows, stáhněte a spusťte msi dostupné na GitHub, nebo spusťte následující příkazy na Linuxu. Předpokládáme, že jste již nainstalovali jazyk Go.

```
$ go get github.com/conformal/btcd/...
```

Pro update btcd na nejnovější verzi spusťte:

```
$ go get -u -v github.com/conformal/btcd/...
```

Správa btcd

Možnosti nastavení btcd si můžete prohlédnout spuštěním:

```
$ btcd --help
```

Spolu s btcd je v balíčku rovněž nástroj ovládaný z příkazové řádky btcctl, který se používá na správu a dotazování v btcd pomocí RPC. Protože btcd nemá povolené RPC server v základním nastavení, musíme nastavit alespoň RPC uživatelské jméno a heslo v následujících konfiguračních souborech.

- *btcd.conf*:

```
[Application Options]
rpcuser=myuser
rpcpass=SomeDecentp4ssw0rd
```

- *btcctl.conf*:

```
[Application Options]
rpcuser=myuser
rpcpass=SomeDecentp4ssw0rd
```

Pokud chcete přepsat konfigurační soubory z příkazové řádky:

```
$ btcd -u myuser -P SomeDecentp4ssw0rd
$ btcctl -u myuser -P SomeDecentp4ssw0rd
```

Následující příkaz zobrazí seznam dostupných nastavení:

```
$ btcctl --help
```

Klíče, adresy, peněženky

Úvod

Vlastnictví bitcoinu je stanoveno pomocí *digitálních klíčů*. *bitcoinových adres* a *digitálních podpisů*. Digitální klíče nejsou uloženy v síti, ale jsou místo toho vytvořeny a uloženy uživateli v souboru, nebo jednoduché databázi zvané *peněženka*. Digitální klíče v uživatelské peněžence jsou zcela nezávislé na bitcoinovém protokolu a mohou být vytvářeny a spravovány uživatelskou peněženkovou aplikací bez odkazu na blockchain nebo bez přístupu na internet. Klíče umožňují mnoho zajímavých vlastností bitcoinu, včetně decentralizované důvěry a kontroly, osvědčení vlastnictví a bezpečností protokol založený na kryptografii.

Každá bitcoinová transakce požaduje platný podpis, aby mohla být vložena do blockchainu. Tento podpis může být vytvořen pouze s platnými digitálními klíči, proto kdokoliv s kopií těchto klíčů má kontrolu nad bitcoiny na tomto účtu. Klíče jsou tvořeny párem: soukromý a veřejný klíč. Veřejný klíč si můžete představovat jako číslo bankovního účtu a soukromý klíč jako tajný PIN nebo podpis na šeku, který poskytuje kontrolu nad účtem. Tyto digitální klíče vidí uživatelé bitcoinu jen málokdy. Z větší části jsou uloženy uvnitř peněženkového souboru a spravovány bitcoinovou peněženkovou aplikací.

V platební části bitcoinové transakce je příjemcův veřejný klíč reprezentován jeho digitálním otiskem, zvaným *bitcoinová adresa*, která je používána stejným způsobem jako jméno příjemce na šeku (např. "zaplatit k rukám"). Ve většině případů je bitcoinová adresa vytvořena z odpovídajícího veřejného klíče. Nicméně ne všechny bitcoinové adresy reprezentují veřejné klíče. Mohou reprezentovat jiné příjemce jako jsou skripty, na což se podíváme později v této kapitole. Tímto způsobem bitcoinové adresy rozšiřují pojem příjemce finančních prostředků, což dělá transakce více flexibilní, podobně jako papírové šeky. Jeden platební nástroj může být použit k platbě na účty fyzických osob, firem, platby účtů nebo platby v hotovosti. Bitcoinová adresa je pouze reprezentace klíčů, kterou uživatelé běžně vidí, protože je potřebují sdílet se světem.

V této kapitole si představíme peněženky obsahující kryptografické klíče. Podíváme se na to, jak jsou klíče vytvářeny, uloženy a spravovány. Prozkoumáme různé formáty kódování použité pro reprezentaci soukromých a veřejných klíčů, adres a adres skriptu. Nakonec se podíváme na speciální použití klíčů: podpis zpráv, prokázání vlastnictví a vytváření ozdobných adres a papírových peněženek.

Kryptografie veřejného klíče a kryptoměny

Kryptografie veřejného klíče byla objevena v 70. letech minulého století a je matematickým základem pro počítačovou a informační bezpečnost.

Od objevu kryptografie veřejného klíče bylo objeveno několik vhodných matematických funkcí, jako jako umocňování prvočísel a násobení eliptických křivek. Tyto matematické funkce jsou prakticky nevratné, což znamená, že jsou snadno spočitatelné v jednom směru, ale nemožné spočítat v druhém směru. Na základě těchto matematických funkcí, kryptografie umožňuje vytvoření digitálního

tajemství a nepadělatelných digitálních podpisů. Bitcoin používá násobení eliptických křivek jako základ pro jeho kryptografii veřejného klíče.

V bitcoinu, používáme kryptografii pro vytvoření páru klíčů, které kontrolují přístup k bitcoinům. Tento pár klíčů se skládá ze soukromého klíče a (z něho odvozeného) jedinečného veřejného klíče. Veřejný klíč je použit pro příjem bitcoinů a soukromý klíč je použit pro podpis transakce, která tyto bitcoiny utrácí.

Existuje matematický vztah mezi veřejným a soukromým klíčem, který umožňuje použití soukromého klíče pro vytvoření podpisu zprávy. Tento podpis může být ověřen proti veřejnému klíči, aniž by soukromý klíč byl prozrazen.

Při utrácení bitcoinů, současný vlastník bitcoinů poskytuje jeho veřejný klíč a podpis (pokaždé různý, ale vytvořený ze stejného soukromého klíče) transakce, aby mohl utratit tyto bitcoiny. Po poskytnutí veřejného klíče a podpisu, může každý v bitcoinové síti ověřit a přijmout transakci jako platnou a potvrdit, že osoba převádějící bitcoiny je v okamžiku převodu vlastní.

TIP V mnoha implementacích peněženek jsou z pohodlnosti soukromý a veřejný klíč uloženy dohromady. Nicméně veřejný klíč může být spočítán ze soukromého klíče a je tedy postačující ukládat pouze soukromý klíč.

Soukromý a veřejný klíč

Bitcoinová peněženka obsahuje sbírku dvojic klíčů, skládajících se ze soukromého a veřejného klíče. Soukromý klíč (k) je číslo, obvykle náhodně vybrané. Ze soukromého klíče vytvoříme veřejný klíč (K) za použití jednosměrné kryptografické funkce násobení eliptických křivek. Z veřejného klíče (K) vytvoříme adresu (A) za pomoci jednosměrné kryptografické hašovací funkce. V této sekci začneme s vytvářením soukromého klíče, podíváme se na použitou matematiku eliptických křivek a nakonec vytvoříme bitcoinovou adresu z veřejného klíče. Vztah mezi soukromým klíčem, veřejným klíčem a bitcoinovou adresou je znázorněn v Soukromý klíč, veřejný klíč, bitcoinová adresa.

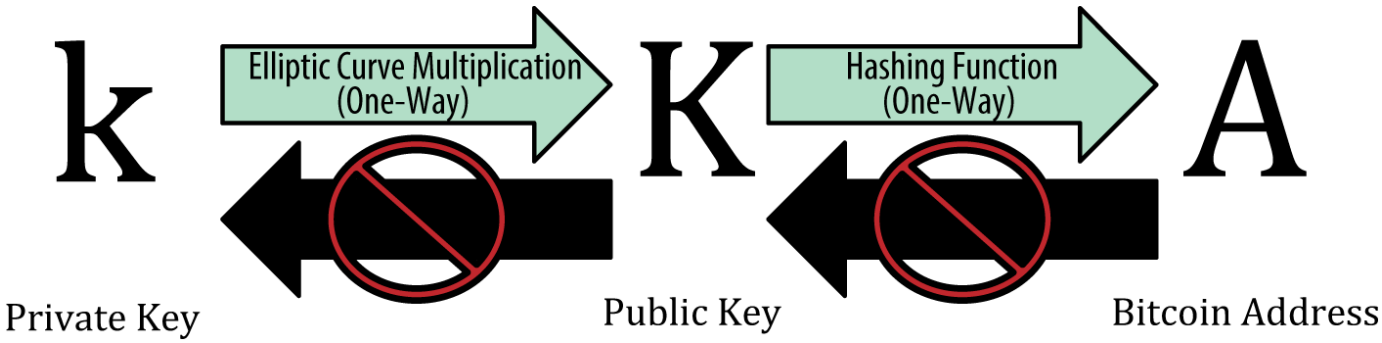


Figure 1. Soukromý klíč, veřejný klíč, bitcoinová adresa

Soukromé klíče

Soukromý klíč je jednoduše číslo, které je vybráno náhodně. Vlastnictví a kontrola soukromého klíče je základem uživatelovy kontroly všech finančních prostředků spojených s odpovídající bitcoinovou

adresou. Soukromý klíč je použit k vytvoření podpisů, které jsou vyžadovány k utracení bitcoinů prokázáním vlastnictví finančních prostředků použitých v transakci. Soukromý klíč musí zůstat utajen celou dobu, protože jeho odhalení třetí straně je stejné jako sdílení kontroly s touto třetí stranou nad bitcoiny zabezpečenými tímto klíčem. Soukromý klíč musí být zálohován a chráněn před náhodnou ztrátou, protože pokud je ztracen, nemůže být navrácen a finanční prostředky zabezpečené tímto klíčem jsou také navždy ztraceny.

TIP

Bitcoinový soukromý klíč je pouhé číslo. Můžete si vytvořit vlastní soukromý klíč náhodně pouze za pomoci mince, tužky a papíru. Hoďte si mincí 256-krát a získáte binární číslice náhodného soukromého klíče, který můžete použit v bitcoinové peněžence. Veřejný klíč může být vytvořen ze soukromého klíče.

Vytváření soukromého klíče z náhodného čísla

První a nejdůležitější krok ve vytváření klíčů je najití bezpečného zdroje entropie nebo náhodnosti. Vytváření bitcoinového klíče je v zásadě stejné jako "vybrání čísla mezi 1 a 2^{256}". Přesná metoda, kterou vyberete číslo není důležitá, pokud není předpověditelná nebo opakovatelná. Bitcoinový software používá jako podklad generátor náhodných čísel poskytovaných operačním systémem, aby vytvořil 256 bitů entropie (náhodnosti). Obvykle generátor náhodných čísel OS je inicializován lidským zdrojem náhodnosti, jako je třeba pohyb myši po obrazovce po několik sekund. Pokud jste opravdu paranoidní, nic nepřekoná kostku, tužku a papír.

Přesněji, soukromý klíč může být číslo mezi 1 a n - 1, kde n je konstanta (n = 1,158 * 10^{77}, mírně menší než 2^{256}) definovaná kvůli použití eliptických křivek v bitcoinu (viz Vysvětlení kryptografie eliptických křivek). Pro vytvoření takovéhoto klíče náhodně zvolíme 256-bitové číslo a zkontrolujeme, zda je menší než n - 1. Při programování se obvykle nejprve vytvoří větší řetězec tvořený náhodnými bity získanými z kryptograficky bezpečného zdroje náhodnosti. Tento řetězec je dán na vstup hašovací funkci SHA256, která vytvoří 256-bitové náhodné číslo. Pokud je výsledek menší než n - 1, máme vhodný soukromý klíč. V opačném případě jednoduše zopakujeme tento postup pro jiné náhodné číslo.

TIP

Neprogramujte si vlastní zdrojový kód k vytváření náhodných čísel nebo nepoužívejte "jednoduchý" generátor náhodných čísel poskytovaný programovacím jazykem. Použijte Use a kryptograficky bezpečný pseudonáhodný generátor čísel (CSPRNG) se semínkem z dostatečného zdroje entropie. Prostudujte dokumentaci knihovny pro vytváření náhodných čísel, abyste jste se ujistili, že je kryptograficky bezpečná. Správná implementace CSPRNG je kritickým bodem bezpečnosti klíčů.

Následuje náhodně vytvořený soukromý klíč (k) zobrazený v hexadecimálním formátu (256 binárních číslic zobrazeno jako 64 hexadecimálních číslic, každé 4-bitové)

```
1E99423A4ED27608A15A2616A2B0E9E52CED330AC530EDCC32C8FFC6A526AEDD
```

TIP Velikost množiny možných bitcoinových soukromých klíčů je 2^{256}, což je bezmezně velké číslo, je to přibližně 10^{77} desítkově. Viditelný vesmír obsahuje přibližně 10^{80} atomů.

Pro vytvoření nového klíče pomocí Bitcoin Core klienta (viz[ch03_bitcoin_client]) použijte příkaz getnewaddress. Z bezpečnostních důvodů se zobrazí pouze veřejný klíč, nikoliv soukromý klíč. Použijte příkaz dumpprivkey pro zobrazení soukromého klíče. Tento příkaz zobrazí klíč ve formátu *Wallet Import Format* (WIF), což je formát Base58 zakončený kontrolním součtem. Tento formát podrobněji prozkoumáme v Formáty soukromého klíče. Uvedeme si příklad vytvoření a zobrazení soukromého klíče pomocí těchto dvou příkazů.

```
$ bitcoind getnewaddress
1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy
$ bitcoind dumpprivkey 1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy
KxFC1jmwwCoACiCAWZ3eXa96mBM6tb3TYzGmf6YwgdGWZgawvrtJ
```

Příkaz dumpprivkey otevře peněženku a vyzvedne z ní soukromý klíč, který byl vytvořen příkazem getnewaddress. Není možné, aby bitcoind odvodil znalost soukromého klíče ze znalosti veřejného klíče, pokud nejsou oba dva uloženy v peněžence.

TIP Příkaz dumpprivkey nevytváří nový soukromý klíč z veřejného klíče, protože je to nemožné. Příkaz pouze zobrazí soukromý klíč, který je již znám peněžence a který byl vytvořen příkazem getnewaddress.

Můžete také z příkazové řádky použít Bitcoinového Průzkumníka (viz[libbitcoin]) pro vytvoření a zobrazení soukromých klíčů s pomocí příkazů seed, ec-new a ec-to-wif:

```
$ bx seed | bx ec-new | bx ec-to-wif
5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn
```

Veřejný klíč

Veřejný klíč je spočítán ze soukromého klíče za použití násobení eliptických křivek, které je nezvratné: \(K = k * G\) kde *k* je soukromý klíč, *G* je konstantní bod zvaný *vytvářecí bod* a *K* je výsledný veřejný klič. Reverzní operace známá jako "najití diskrétního logaritmu" - výpočet *k* ze znalosti *K* je stejně těžká, jako vyzkoušet všechny možné hodnoty k hrubou silou. Předtím, než si ukážeme jak vytvořit veřejný klič ze soukromého klíče, podívejme se na kryptografii eliptických křivek detailněji.

Vysvětlení kryptografie eliptických křivek

Kryptografie eliptických křivek je typ asymetrické kryptografie veřejného klíče, která je založena na problému diskrétního logaritmu vyjádřeného sčítáním a násobením bodů na eliptické křivce.

Eliptická křivka je příklad eliptické křivky, podobné jaká je použita bitcoinem.

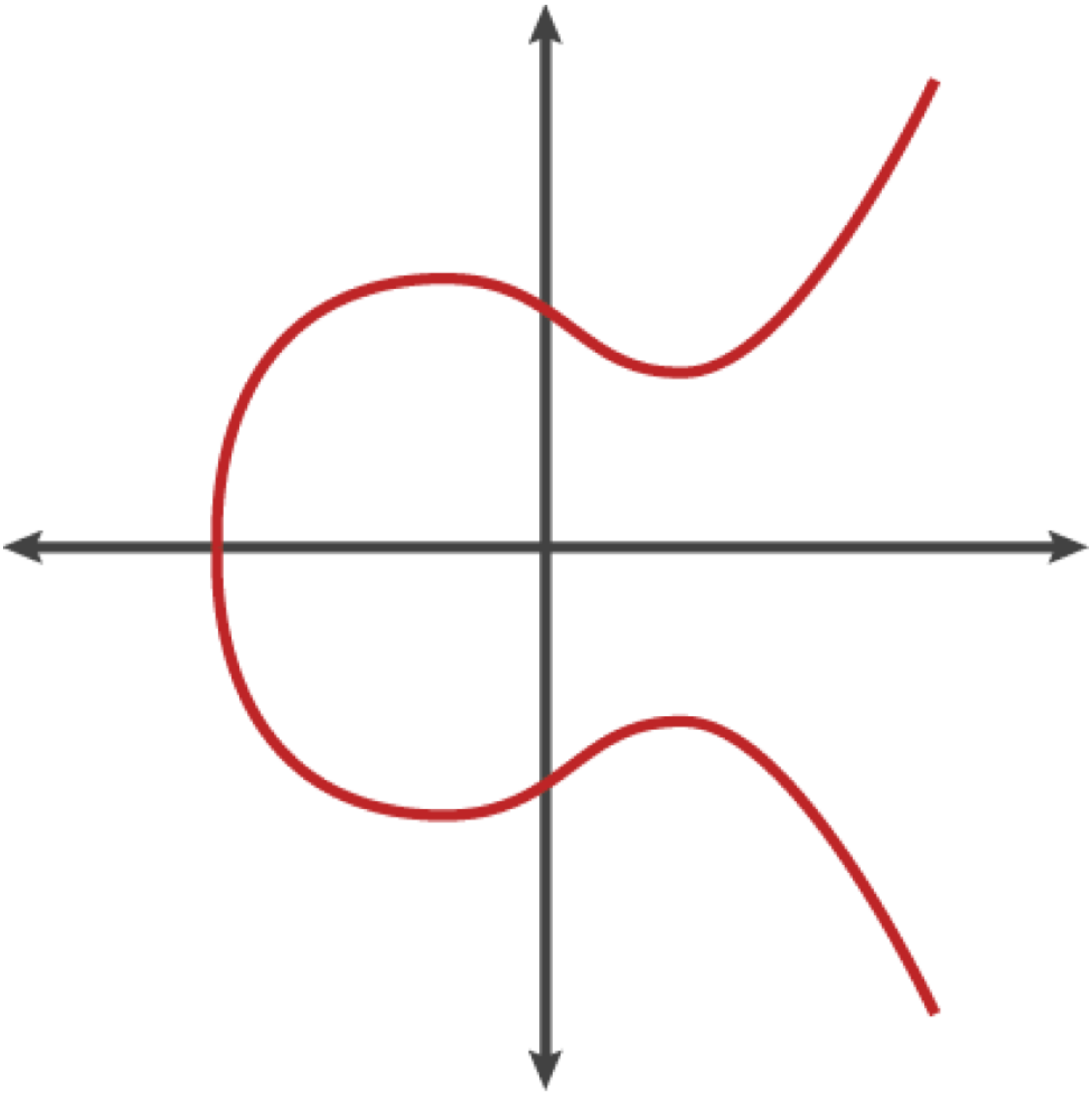


Figure 2. Eliptická křivka

Bitcoin používá specifickou eliptickou křivku a množinu matematických konstant, definovaných standardem nazvaným secp256k1 vydaným Národním institutem pro standardy a technologie (NIST). Křivka secp256k1 je definována následující funkcí, která vytváří eliptickou křivku:

$y^2 = (x^3 + 7)$ nad tělesem Z_p NEBO $y^2 \bmod p = (x^3 + 7) \bmod p$

Značení *mod p* (zbytek po celočíselném dělení prvočíslem p) značí, že tato křivka je definována nad konečným tělesem řádu *p*, také zapsáno Z_p, kde $p=2^{256} – 2^{32} – 2^9 – 2^8 – 2^7 – 2^6 – 2^4 – 1$ je velmi vysoké prvočíslo.

Protože tato křivka je definována nad konečným tělesem místo nad reálnými čísly, vypadá jako tečkovaný vzorek rozptýlený ve dvou dimenzích, což ztěžuje jí zobrazení. Nicméně, matematika je stejná jako u eliptických křivek nad reálnými čísly. Příklad Kryptografie eliptických křivek: zobrazení eliptické křivky nad tělesem Z_{17} zobrazuje tu samou eliptickou křivku nad mnohem menším konečným tělesem Z_{17}, ukazuje vzorek teček na mřížce. Bitcoinovou eliptickou křivku secp256k1 si lze představit jako výrazně složitější vzorek teček na nepředstavitelně velké mřížce.

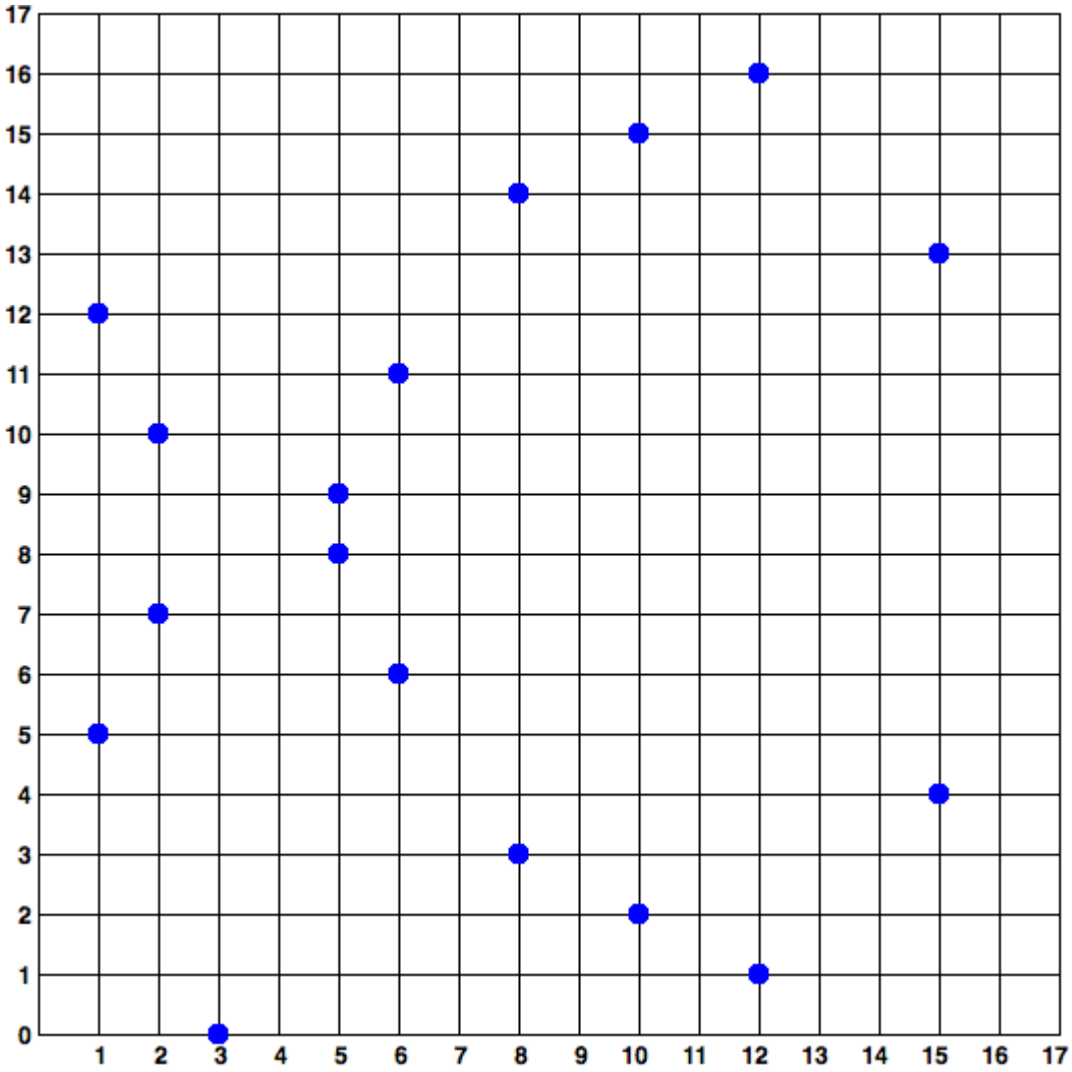


Figure 3. Kryptografie eliptických křivek: zobrazení eliptické křivky nad tělesem Z_{17}

Například se podíváme na bod P se souřadnicemi (x,y), který je bodem křivky secp256k1. Můžete si sami ověřit pomocí Pythonu:

```
P = (55066263022277343669578718895168534326250603453777594175500187360389116729240,
32670510020758816978083085130507043184471273380659243275938904335757337482424)
```

```
Python 3.4.0 (default, Mar 30 2014, 19:23:13)
[GCC 4.2.1 Compatible Apple LLVM 5.1 (clang-503.0.38)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> p =
115792089237316195423570985008687907853269984665640564039457584007908834671663
>>> x = 55066263022277343669578718895168534326250603453777594175500187360389116729240
>>> y = 32670510020758816978083085130507043184471273380659243275938904335757337482424
>>> (x ** 3 + 7 - y**2) % p
0
```

V matematice eliptických křivek, existuje bod zvaný "bod v nekonečnu", který zastává úlohu 0 při sčítání. V počítačích je občas reprezentován výrazem x = y = 0 (který nesplňuje rovnici eliptické křivky, ale je to jednoduchý speciální případ, který může být zkontrolován).

Používá se také operátor + zvaný "sčítání", který má ty samé vlastnosti jako tradiční sčítání reálných čísel, které se učí děti na základní škole. Pokud jsou dva body P_1 a P_2 na eliptické křivce, pak na eliptické křivce také i třetí bod $P_3 = P_1 + P_2$.

Geometricky je tento třetí bod P_3 spočítán nakreslením přímky mezi P_1 a P_2. Tato přímka protne eliptickou křivku v právě jednom novém bodě. Nazvěme tento bod $P_3' = (x, y)$. Poté použijeme osovou symetrii dle osy x, získáme $P_3 = (x, –y)$.

Nastává řada speciálních případů, které vysvětlují potřebu "bodu v nekonečnu."

Pokud jsou P_1 a P_2 identickým bodem, přímka "mezi" P_1 a P_2 by měla být rozšířena, aby byla tečnou křivky v bodě P_1. Tato tečna protne křivku v právě jednom novém bodě. Můžete použít techniku z goniometrie pro určení sklonu této tečny. Tato technika překvapivě funguje, dokonce pokud omezíme náš zájem na body křivky s oběma celočíselnými souřadnicemi.

V některých případech (např. P_1 a P_2 mají stejnou hodnotu x ale rozdílnou hodnotu y) bude tečna přesně svislá, v tomto případě P3 = "bod v nekonečnu,"

Pokud P_1 je "bod v nekonečnu", poté součet $P_1 + P_2 = P_2$. Obdobně pokud P_2 je "bod v nekonečnu", poté součet $P_1 + P_2 = P_1$. Toto demonstruje, že bod v nekonečnu zastává roli 0.

Z toho vyplývá, že +je asociativní, což znamená že (A + B) + C = A + (B + C). Proto můžeme psát A + B + C bez závorek, aniž by došlo k nejednoznačnosti.

Nyní máme definované sčítání, můžeme definovat násobení standardním způsobem jako rozšířené sčítání. Pro bod P eliptické křivky, pokud k je celé číslo, pak kP = P + P + P + … + P (k-krát). Poznámka, k je v tomto případě občas zmatečně nazýváno "exponentem".

Vytvoření veřejného klíče

Začneme soukromým klíčem ve tvaru náhodně vytvořeného čísla *k*. Vynásobíme ho předdefinovaným bodem na křivce nazvaným *vytvářecí bod G*. Tím získáme jiný bod na křivce, který odpovídá veřejnému klíči *K*. Vytvářecí bod je určen ve standardu secp256k1 a je stejný pro všechny klíče v bitcoinu.

kde k je soukromý klíč, G je vytvářecí bod a K je výsledný veřejný klíč, bod na křivce. Protože vytvářecí bod je shodný pro všechny uživatele bitcoinu, soukromý klíč k vynásobený G vede vždy ke stejnému veřejnému klíči K. Vztah mezi k a K je pevný, ale může být spočítán pouze v jednom směru, od k ke K. To je důvod proč bitcoinová adresa (odvozená z K) může být sdílena s kýmkoliv a neprozrazuje uživatelův soukromý klíč (k).

TIP	Soukromý klíč může být přeměněn na veřejný klíč, ale veřejný klič nemůže být přeměněn zpátky na soukromý klíč, protože matematicky funguje pouze jedním směrem.

Implementujeme násobení eliptických křivek, vezmeme předtím vytvořený soukromý klíč k a vynásobíme ho vytvářecím bodem G, abychom nalezli veřejný klíč K:

```
K = 1E99423A4ED27608A15A2616A2B0E9E52CED330AC530EDCC32C8FFC6A526AEDD * G
```

Veřejný klíč K je definován jako bod K = (x,y):

```
K = (x, y)

kde

x = F028892BAD7ED57D2FB57BF33081D5CFCF6F9ED3D3D7F159C2E2FFF579DC341A
y = 07CF33DA18BD734C600B96A72BBC4749D5141C90EC8AC328AE52DDFE2E505BDB
```

Pro znázornění násobení bodu celým číslem použijeme jednodušší eliptickou křivku nad reálnými čísly - pamatujte, je to ta samá matematika. Naším cílem je nalézt součin kG vytvářecího bodu G. je to stejné jako přičítat G sám k sobě, k-krát za sebou. V eliptických křivkách je sčítání bodu sama se sebou ekvivalentní k nakreslení tečny v tomto bodě a hledání, kde znova protne křivku a následnému provedení osové symetrii dle osy x.

Kryptografie eliptických křivek: Zobrazení násobení bodu G celými čísly k na eliptické křivce ukazuje proces odvození G, 2G, 4G jako geometrické operace na křivce.

TIP	Většina bitcoinových implementací používá OpenSSL cryptographic library na provádění výpočtů eliptických křivek. Například pro odvození veřejného klíče se používá funkce EC_POINT_mul().

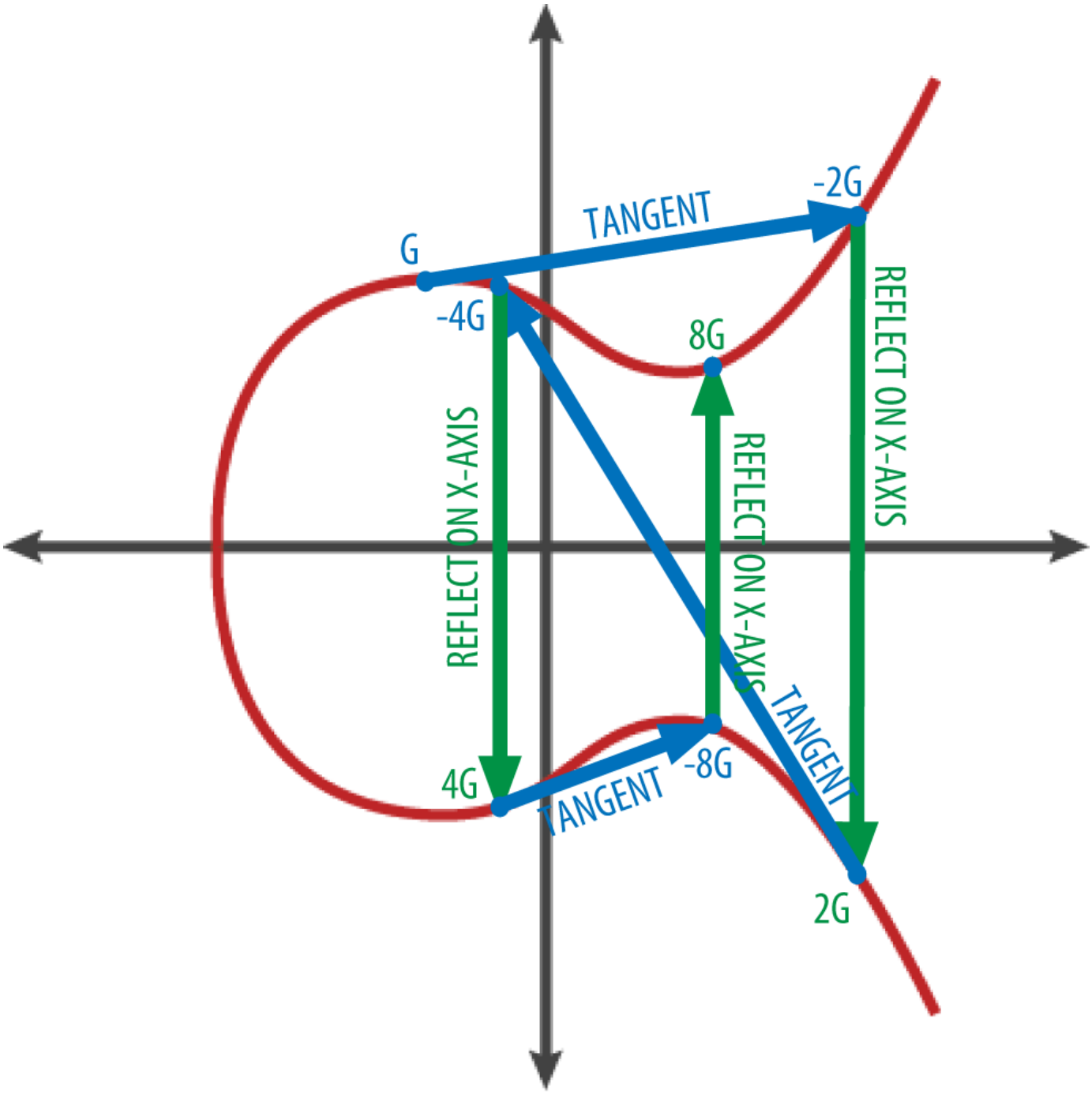


Figure 4. Kryptografie eliptických křivek: Zobrazení násobení bodu G celými čísly k na eliptické křivce

Bitcoinová adresa

Bitcoinová adresa je řetězec čísel a znaků, který může být sdílen s kýmkoliv, kdo vám chce zaslat peníze. Adresa odvozená z veřejného klíče se skládá z řetězce číslic a písmen, začínajícím číslici "1". Zde je příklad bitcoinové adresy:

```
1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy
```

Bitcoinová adresa je to, co se nejčastěji objevuje v transakci jako "příjemce" finančních prostředků. Pokud srovnáte bitcoinovou transakci s papírovým šekem, bitcoinová adresa je příjemce, kterého napíšeme na řádku za "Zaplatit k rukám." Na papírovém šeku může být příjemce občas jméno majitele bankovního účtu, ale rovněž to může být firma, instituce nebo dokonce hotovost. Protože papírové šeky nemusejí specifikovat účet, ale spíše abstraktní jméno příjemce finančních prostředků, což dělá z šeku velmi pružný platební nástroj. Bitcoinové transakce používají podobnou abstrakci. Bitcoinová adresa je také velmi pružná. Bitcoinová adresa může reprezentovat majitele dvojice soukromý/veřejný klič, nebo něco jiného, jako třeba platební skript, jak uvidíme v [p2sh]. Nyní prozkoumáme jednodušší případ, který reprezentuje bitcoinová adresa odvozená z veřejného klíče.

Bitcoinová adresa je odvozena z veřejného klíče pomocí jednosměrného kryptografického hašování. "Hašovací algoritmus" je jednosměrná funkce, která vyrobí otisk nebo "haš" libovolně velkého vstupu. Kryptografické hašovací funkce jsou používány v bitcoinu v mnoha situacích: v bitcoinové adrese, v adrese skriptu a těžebním algoritmu důkazu prací. Algoritmy použité k vytvoření bitcoinové adresy z veřejného klíče jsou Secure Hash Algorithm (SHA) a RACE Integrity Primitives Evaluation Message Digest (RIPEMD), konkrétně verze SHA256 a RIPEMD160.

Začneme s veřejným klíčem K, spočítáme SHA256 a poté spočítáme RIPEMD160 haš tohoto výsledku, ze kterého vznikne 160-bitové (20-bytové) číslo:

kde K je veřejný klíč a A je výsledná bitcoinová adresa.

TIP	Bitcoinová adresa _není_stejná jako veřejný klíč. Bitcoinová adresa je odvozena z veřejného klíče za použití jednocestné funkce.

Bitcoinové adresy jsou téměř vždy uživatelům prezentovány ve formátu zvaném "Base58Check" (viz Base58 and Base58Check kódování), který používá 58 znaků (číselná soustava o základu 58) a kontrolní součet. Tento formát je lidsky čitelný, zabraňuje nejednoznačnosti, chrání před chybami při opisování a zadávaní adresy. Base58Check se také používá v mnoha jiných situacích v bitcoinu, kde potřebujeme používat lidsky čitelná a správně zapsaná čísla jako u bitcoinových adres, soukromých klíčů, zašifrovaných klíčů nebo hašů skriptů. V další části prozkoumáme algoritmus Base58Check kódování a dekódování a výsledné reprezentace. Konverze veřejného klíče na bitcoinovou adresu zobrazuje konverzi veřejného klíče na bitcoinovou adresu.

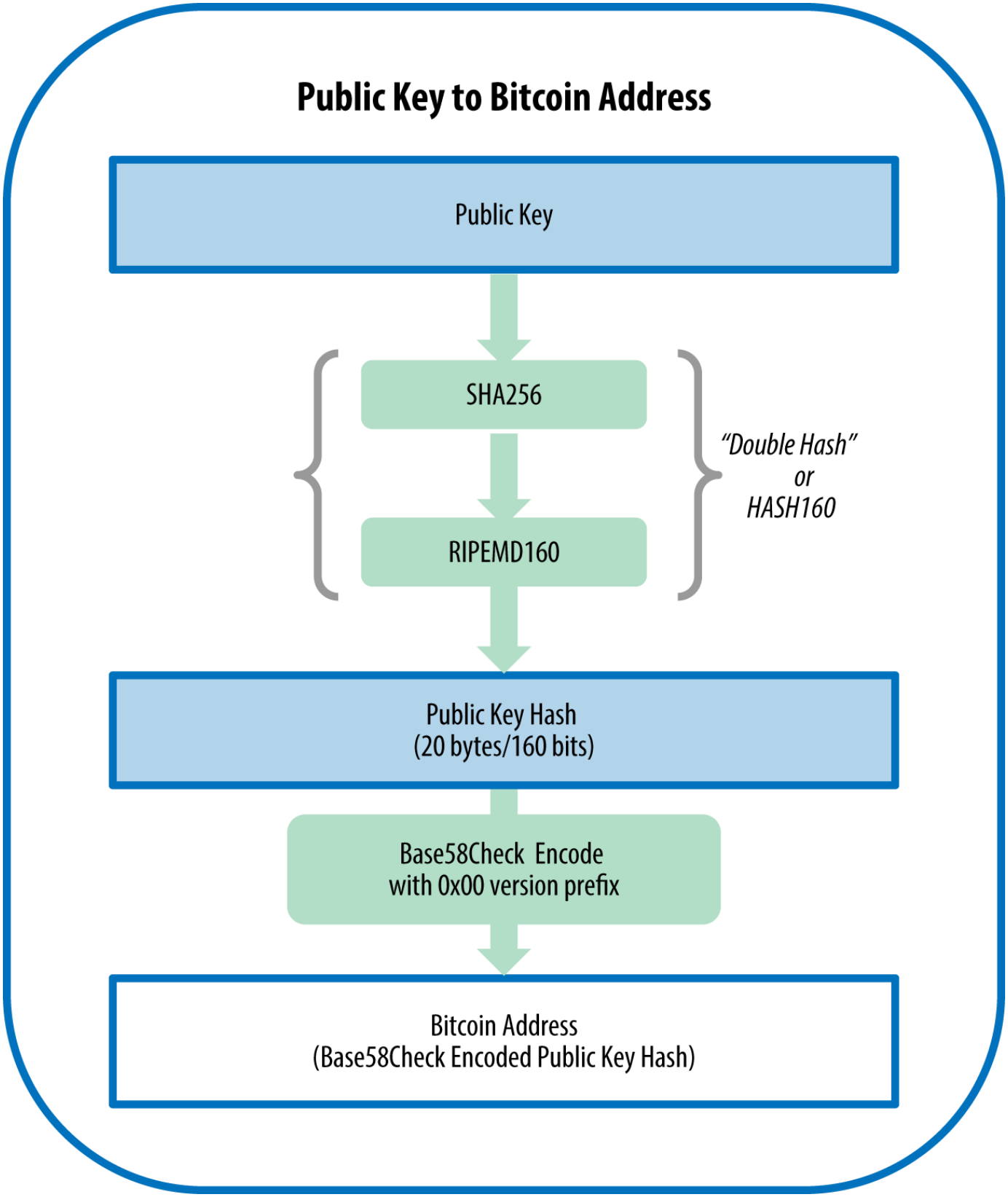


Figure 5. Konverze veřejného klíče na bitcoinovou adresu

Base58 and Base58Check kódování

Za účelem reprezentace dlouhých čísel celistvým způsobem, za použití méně znaků, používá mnoho počítačových systémů smíšené alfanumerické reprezentace se základem větším než 10. Například

tradiční desítkový systém používá 10 číslic od 0 do 9, hexadecimální systém používá 16, navíc písmena od A do F, které reprezentují 6 dodatečných symbolů. Zápis čísla v hexadecimální soustavě je kratší než jeho zápis v desítkové soustavě, dokonce hustější. Reprezentace Base64 používá 26 malých písmen, 26 velkých písmen, 10 číslic a dva další symboly "+" and "/" pro přenos binárních dat prostřednictvím textově orientovaného média jako je email. Base64 se nejčastěji používá při přidávání binárních příloh do emailu. Base58 je textově orientovaný formát kódování binárních čísel vyvinutý pro použití v bitcoinu a používaný mnoha jinými kryptoměnami. Nabízí rovnováhu mezi hustotou reprezentace, čitelností, prevencí a detekcí chyb. Base58 je podmnožinou Base64, používá velká a malá písmena, čísla ale vynechává některé znaky, které jsou často zaměňovány jeden za druhý a mohou vypadat shodně při zobrazení v některých fontech. Speciálně Base 58 je Base 64 bez 0 (číslo 0), O (velké o), l (malé L), I (velké i) symbolů "\+" a "/". Zjednodušeně, je to množina malých a velkých písmen a čísel bez čtyř (0, O, l, I) právě zmíněných.

Example 1. Bitcoinová abeceda Base58

```
123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz
```

Pro zvýšení bezpečnosti proti překlepům a chybám přepisů, Base58Check je Base58 kódovací formát, často používaný v bitcoinu, který má zabudovaný kontrolní kód. Kontrolní součet je odvozen z haše a je zakódován do dat, proto může být použitý pro detekci a prevenci překlepů a chyb přepisu. Při předložení Base58Check kódu dekódovací software spočítá kontrolní součet dat porovná s kontrolním součtem vloženým v kódu. Pokud se tyto dva součty neshodují, je to příznakem výskytu chyba a Base58Check data jsou neplatná. Například toto zabraňuje bitcoinové adrese s překlepem, aby byla peněženkovou aplikací přijata jako platný příjemce, tato chyba by jinak vedla v nevratnou ztrátu finančních prostředků.

Pro konverzi dat (čísla) do Base58Check formátu, musíme nejprve přidat prefix datům, nazývaný "byte verze", což slouží ke snadné identifikaci typu dat, která jsou zakódována. Například bitcoinová adresa má prefix nula (0x00 hex), zatímco soukromý klíč má prefix 128 (0x80 hex). Seznam obvyklých prefixů je zobrazen v Příklady prefixů verzí v Base58Check.

Dále, spočítáme "dvojitý SHA" kontrolní součet, což znamená, že aplikujeme SHA256 hašovací algoritmus dvakrát na předchozí výsledek (prefix a data):

```
checksum = SHA256(SHA256(prefix+data))
```

Z výsledného 32-bytového haše (haše haše), vezeme pouze první 4 byty. Tyto 4 byty slouží jako kontrolní součet . Kontrolní součet je připojen na konec.

Výsledek vznikne složením tří složek: prefix, data a kontrolní součet. Tento výsledek je zakódován pomocí Base58 abecedy popsané dříve. Base58Check kódování: Base58, verzovaný a kontrolním součtem opatřený formát pro jednoznačné kódování bitcoinových dat. zachycuje postup kódování Base58Check.

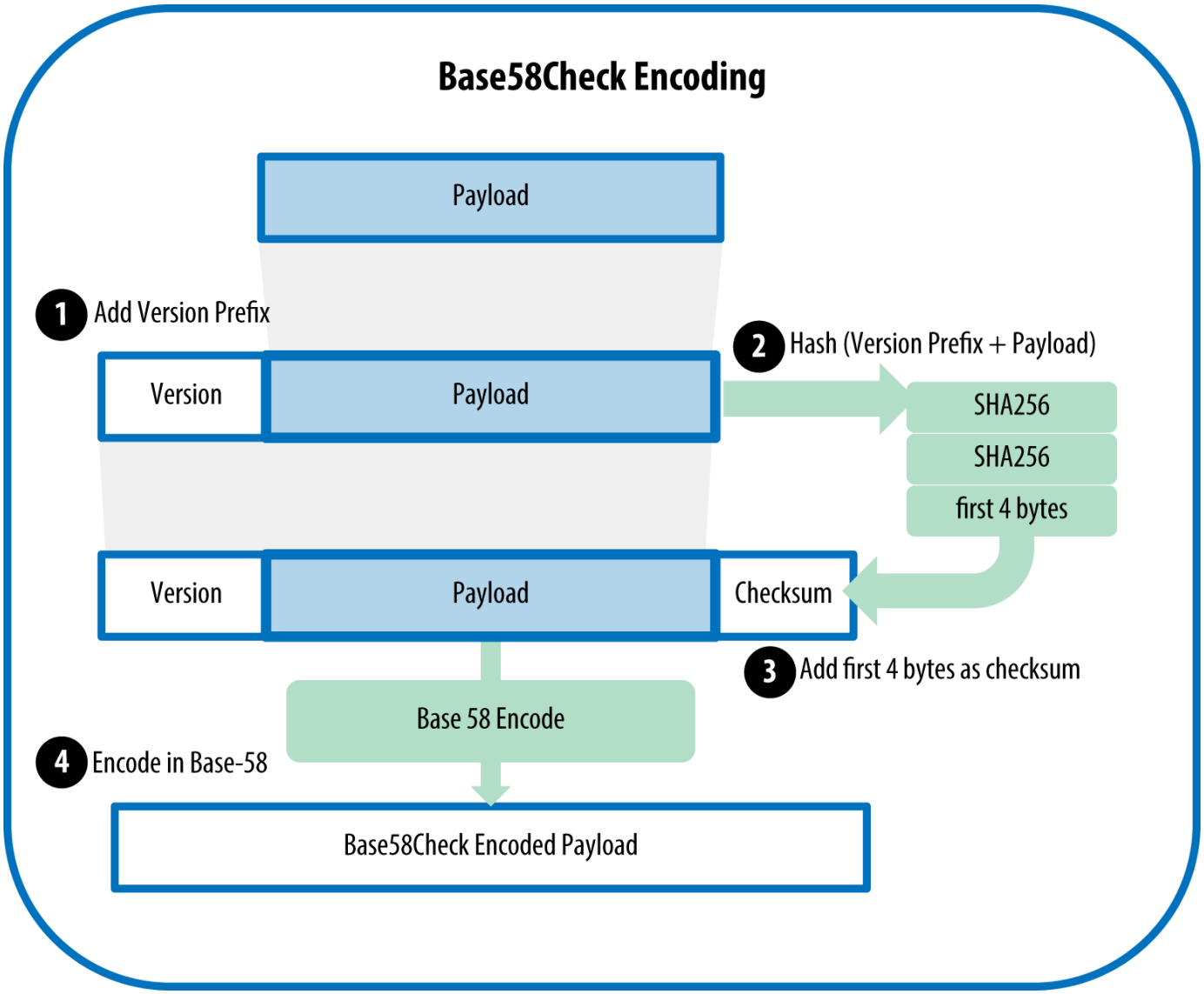


Figure 6. Base58Check kódování: Base58, verzovaný a kontrolním součtem opatřený formát pro jednoznačné kódování bitcoinových dat.

V bitcoinu je většina dat předkládaných uživateli kódována pomocí Base58Check, aby byla hustá, snadno čitelná a schopná detekovat chyby. Prefix verze v Base58Check kódování je použit pro vytvoření snadno rozlišitelných formátů, které poté, co jsou kódovány v Base58, obsahují specifické znaky na začátku obsahu kódovaného Base58Check. Tyto znaky usnadňují lidem identifikaci typu kódovaných dat a jejich použití. Je to odlišení, například, bitcoinová adresa kódovaná Base58Check začíná na 1 a soukromý klíč ve formátu WIF kódovaný Base58Check začíná na 5. Některé další prefixy verzí a výsledné Base58 znaky jsou uvedeny v Příklady prefixů verzí v Base58Check.

Table 1. Příklady prefixů verzí v Base58Check

Typ	Prefix verze (hex)	Base58 výsledný prefix
Bitcoinová adresa	0x00	1
Adresa platby haši skriptu	0x05	3
Adresa v bitcoinové testovací síti	0x6F	m nebo n

Typ	Prefix verze (hex)	Base58 výsledný prefix
Soukromý klíč WIF	0x80	5, K nebo L
BIP38 Zašifrovaný soukromý klič	0x0142	6P
BIP32 rozšířený veřejný klíč	0x0488B21E	xpub

Podíváme se na celý postup tvorby bitcoinové adresy ze soukromého klíče přes veřejný klíč (bod na eliptické křivce), přes dvojitě hašovanou adresu až nakonec k Base58Check kódování. Zdrojový kód v C++ v Tvorba bitcoinové adresy kódované Base58Check ze soukromého klíče ukazuje celý postup krok za krokem, ze soukromého klíče do bitcoinové adresy kódované Base58Check. Zdrojový kód příkladu používá knihovnu libbitcoin představenou v [alt_libraries] pro některé pomocné funkce.

Example 2. Tvorba bitcoinové adresy kódované Base58Check ze soukromého klíče

```
#include <bitcoin/bitcoin.hpp>

int main()
{
    // Private secret key.
    bc::ec_secret secret;
    bool success = bc::decode_base16(secret,
        "038109007313a5807b2eccc082c8c3fbb988a973cacf1a7df9ce725c31b14776");
    assert(success);
    // Get public key.
    bc::ec_point public_key = bc::secret_to_public_key(secret);
    std::cout << "Public key: " << bc::encode_hex(public_key) << std::endl;

    // Create Bitcoin address.
    // Normally you can use:
    //   bc::payment_address payaddr;
    //   bc::set_public_key(payaddr, public_key);
    //   const std::string address = payaddr.encoded();

    // Compute hash of public key for P2PKH address.
    const bc::short_hash hash = bc::bitcoin_short_hash(public_key);

    bc::data_chunk unencoded_address;
    // Reserve 25 bytes
    //   [ version:1  ]
    //   [ hash:20    ]
    //   [ checksum:4 ]
    unencoded_address.reserve(25);
    // Version byte, 0 is normal BTC address (P2PKH).
    unencoded_address.push_back(0);
    // Hash data
    bc::extend_data(unencoded_address, hash);
    // Checksum is computed by hashing data, and adding 4 bytes from hash.
    bc::append_checksum(unencoded_address);
    // Finally we must encode the result in Bitcoin's base58 encoding
    assert(unencoded_address.size() == 25);
    const std::string address = bc::encode_base58(unencoded_address);

    std::cout << "Address: " << address << std::endl;
    return 0;
}
```

Zdrojový kód používá předdefinovaný soukromý klíč, takže vyrobí stejnou bitcoinovou adresu při

každém jeho běhu, jak ukazuje Kompilace a spuštění zdrojového kódu addr

Example 3. Kompilace a spuštění zdrojového kódu addr

```
# Překlad souboru zdrojových kódů addr.cpp
$ g++ -o addr addr.cpp $(pkg-config --cflags --libs libbitcoin)
# Spuštění souboru addr, který vznikl při překladu
$ ./addr
Public key: 0202a406624211f2abbdc68da3df929f938c3399dd79fac1b51b0e4ad1d26a47aa
Address: 1PRTTaJesdNovgne6Ehcdu1fpEdX7913CK
```

Formáty klíčů

Jak soukromý tak i veřejný klíč můžou být reprezentované jako číslo v různých formátech. Všechny tyto reprezentace kódují stejná čísla, dokonce i když vypadají různě. Tyto formáty jsou primárně určeny pro snadnou čitelnost lidmi a vyhnutí se chybám při přepisování klíčů.

Formáty soukromého klíče

Soukromý klíč může být reprezentován v několika různých formátech, všechny odpovídají tomu samému 256-bitovému číslu. Reprezentace soukromého klíče (formáty kódování) zobrazuje tři časté formáty používané k reprezentaci soukromých klíčů.

Table 2. Reprezentace soukromého klíče (formáty kódování)

Typ	Prefix	Popis
Hexadecimální	Žádný	64 hexadecimálních číslic
WIF	5	Base58Check kódování: Base58 s prefixem 128 a 32-bitový kontrolní součet
Komprimovaný WIF	K or L	jako nahoře, ale přidána přípona 0x01 před kódováním.

Příklad: stejný klíč, různé formáty ukazuje soukromý klíč vytvořený v těchto třech formátech.

Table 3. Příklad: stejný klíč, různé formáty

Formát	Soukromý klič
Hexadecimální	1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526aedd
WIF	5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn

Formát	Soukromý klič
Komprimovaný WIF	KxFC1jmwwCoACiCAWZ3eXa96mBM6tb3TYzGmf6YwgdGWZgawvrtJ

Všechny tyto reprezentace se liší způsobem zobrazení stejného čísla, stejného soukromého klíče. Vypadají různě, ale každý formát může být snadno převeden na jakýkoliv jiný formát.

Použijeme příkaz wif-to-ec z Bitcoinového Průzkumníka (viz [libbitcoin]), abychom ukázali, že oba WIF klíče reprezentují stejný soukromý klíč.

```
$ bx wif-to-ec 5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn
1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526aedd

$ bx wif-to-ec KxFC1jmwwCoACiCAWZ3eXa96mBM6tb3TYzGmf6YwgdGWZgawvrtJ
1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526aedd
```

Převod z Base58Check

Příkazy Bitcoinového Průzkumníka (viz [libbitcoin]) usnadňují psaní shell skriptů a "rour" na příkazové řádce, které spravují bitcoinové klíče, adresy a transakce. Můžete použít Bitcoinového Průzkumníka na Převod z Base58Check formátu na jiné formáty.

Použijeme příkaz base58check-decode pro převod nekomprimovaného klíče.

```
$ bx base58check-decode 5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn
wrapper
{
    checksum 4286807748
    payload 1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526aedd
    version 128
}
```

Výsledek obsahuje klíč jako náklad, prefix 128 pro Wallet Import Format (WIF) a kontrolní součet.

Všimněte si, že k "nákladu" komprimovaného klíče je přidána přípona 01, oznamující, že odvozený veřejný klíč bude komprimovaný.

```
$ bx base58check-decode KxFC1jmwwCoACiCAWZ3eXa96mBM6tb3TYzGmf6YwgdGWZgawvrtJ
wrapper
{
    checksum 2339607926
    payload 1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526aedd01
    version 128
}
```

Převod z hexadecimálního do Base58Check

Pro převod do Base58Check (opak předchozího příkazu) použijeme příkaz base58check-encode z Bitcoinového průzkumníka (viz [libbitcoin]) a poskytneme hexadecimální soukromý klíč, následovaný prefixem 128 pro Wallet Import Format (WIF).

```
bx base58check-encode 1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526aedd
--version 128
5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn
```

Převod z hexadecimálního (komprimovaný klíč) do Base58Check

Pro převod do Base58Check jako "komprimovaného" soukromého klíče (viz Komprimované soukromé klíče), přidáme příponu 01 k hexadecimálnímu klíči a poté převedeme jak je uvedeno výše.

```
$ bx base58check-encode
1e99423a4ed27608a15a2616a2b0e9e52ced330ac530edcc32c8ffc6a526aedd01 --version 128
KxFC1jmwwCoACiCAWZ3eXa96mBM6tb3TYzGmf6YwgdGWZgawvrtJ
```

Výsledný komprimovaný WIF formát začíná s "K". To značí soukromý klíč s příponou "01" a bude použit pro vytvoření komprimovaného veřejného klíče (viz Komprimovaný veřejný klíč).

Formáty veřejného klíče

Veřejný klíč je také uváděn různými způsoby, nejdůležitější jsou *komprimovaný* a *nekomprimovaný* veřejný klíč.

Jak jsme viděli dříve, veřejný klíč je bod na eliptické křivce skládající se z dvojici souřadnic (x,y). Je obvykle označen prefixem 04 následovaným dvěma 256-bitovými čísly, jedno pro souřadnici *x* tohoto bodu a druhé pro *y* souřadnici. Prefix 04 je použit pro rozlišení nekomprimovaného veřejného klíče od komprimovaného veřejného klíče, který začíná 02 nebo 03.

Zde je veřejný klíč, který jsme vytvořili ze soukromého klíče již dříve, zobrazen jako souřadnice x a y:

```
x = F028892BAD7ED57D2FB57BF33081D5CFCF6F9ED3D3D7F159C2E2FFF579DC341A
y = 07CF33DA18BD734C600B96A72BBC4749D5141C90EC8AC328AE52DDFE2E505BDB
```

Zde je ten samý veřejný klíč zobrazený jako 520-bitové číslo (130 hexadecimálních číslic) s prefixem 04 následovaný souřadnicemi x a y, tedy 04 x y:

```
K = 04F028892BAD7ED57D2FB57BF33081D5CFCF6F9ED3D3D7F159C2E2FFF579DC341A<?pdf-
cr?>07CF33DA18BD734C600B96A72BBC4749D5141C90EC8AC328AE52DDFE2E505BDB
```

Komprimovaný veřejný klíč

Komprimované veřejné klíče byly zavedeny do Bitcoinu za účelem snížení velikosti transakcí a úspoře diskového místa u uzlů, které ukládají databázi bitcoinového blockchainu. Mnoho transakcí obsahuje veřejný klíč, požadovaný k ověření oprávnění uživatele k utracení bitcoinů. Každý veřejný klíč obsahuje 520 bitů (prefix \+ x \+ y), který po vynásobení několika stovkami transakcí v bloku nebo desítkami tisíc transakcí denně, značně zvyšuje množství dat v blockchainu.

Jak jsme viděli v sekci Veřejný klíč, veřejný klíč je bod (x,y) na eliptické křivce. Protože křivka vyjadřuje matematickou funkci, bod na křivce reprezentuje řešení rovnice a proto, pokud známe souřadnici *x*, můžeme dopočítat souřadnici *y* vyřešením rovnice $y^2 \bmod p = (x^3 + 7) \bmod p$. To nám umožňuje uložit pouze souřadnici *x* z bodu veřejného klíče, vynechat souřadnici *y* a snížit tak velikost klíče a místa potřebného k jeho uložení na 256 bitů. Téměř 50 % snížení velikosti v každé transakci přispívá k velké úspoře uložených dat v průběhu času!

Zatímco nekomprimovaný veřejný klíč má prefix 04, komprimovaný veřejný klíč začíná prefixem 02 nebo 03. Podíváme se na to, proč jsou možné dva prefixy. Protože levá strana rovnice je y^2, což znamená, že řešení pro y je základem mocniny, který může mít buď kladné nebo záporné znaménko. To znamená, že výsledná *y* souřadnice může být nad nebo pod osou x. Jak můžeme vidět z grafu eliptické křivky v Eliptická křivka, křivka je symetrická, osově souměrná dle osy x. Když vynecháme souřadnici y, musíme uložit *znaménko* y (kladné nebo záporné). Jinými slovy, musíme si pamatovat, zda se bod nachází nad nebo pod osou x, protože každá z těchto možností reprezentuje jiný bod a jiný veřejný klíč. Při výpočtu eliptické křivky v binární aritmetice nad konečným tělesem řádu p, souřadnice *y* je buďto lichá nebo sudá, což odpovídá kladnému/zápornému znaménku jak bylo vysvětleno dříve. Proto pro rozlišení mezi dvěma možnými hodnotami y uloženými v komprimovaném veřejném klíči používáme prefix 02 pokud y je sudé a 03 pokud je liché. To umožňuje software správně odvodit souřadnici *y* ze souřadnice *x* a dekomprimovat veřejný klíč do plných souřadnic bodu. Komprese veřejného klíče je znázorněna na Komprese veřejného klíče.

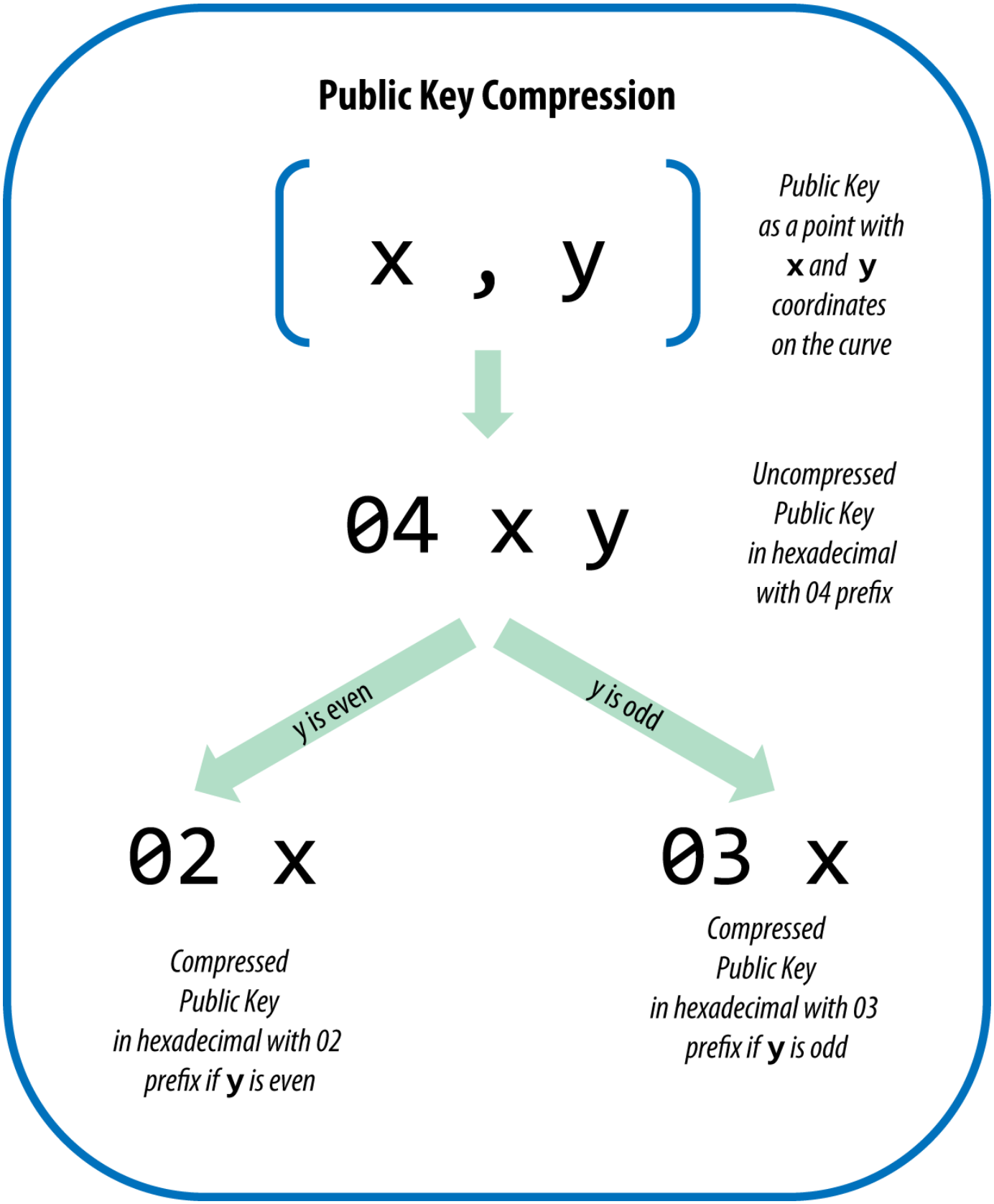


Figure 7. Komprese veřejného klíče

Zde je ten samý, dříve vytvořený, veřejný klíč ukázán v komprimované podobě, uložený v 264 bitech (66 hexadecimálních číslic) s prefixem 03 označujícím, že souřadnice *y* je lichá.

```
K = 03F028892BAD7ED57D2FB57BF33081D5CFCF6F9ED3D3D7F159C2E2FFF579DC341A
```

Komprimovaný veřejný klíč odpovídá tomu samému soukromému klíči, je vytvořen ze stejného soukromého klíče. Nicméně vypadá odlišně od nekomprimovaného veřejného klíče. Co je důležitější, pokud převedeme komprimovaný veřejný klíč na bitcoinovou adresu za použití dvojité hašovací funkce (RIPEMD160(SHA256(K))), vytvoří odlišnou *bitcoinovou* adresu. To může být matoucí, protože to znamená, že jeden soukromý klíč může vyrobit veřejný klíč vyjádřený ve dvou rozdílných formátech (komprimovaný a nekomprimovaný), které vyrobí dvě rozdílné bitcoinové adresy. Nicméně soukromý klíč je shodný pro obě bitcoinové adresy.

Komprimované veřejné klíče se postupně stávají normou mezi bitcoinovými klienty, což má značný dopad na redukci velikosti transakcí a blockchainu. Nicméně ne všichni klienti už podporují komprimované veřejné klíče. Novější klienti, kteří podporují komprimované veřejné klíče musejí počítat s transakcemi od starších klientů, kteří komprimované veřejné klíče nepodporují. To je speciálně důležité když peněženková aplikace importuje soukromé klíče z jiné bitcoinové peněženkové aplikace, protože nová peněženka potřebuje prohledat blockchain, aby našla transakce odpovídající těmto importovaným klíčům. Které bitcoinové adresy by bitcoinová peněženka měla hledat? Bitcoinové adresy vyrobené z nekomprimovaných veřejných klíčů, nebo bitcoinové adresy vyrobené z komprimovaných veřejných klíčů? Oboje jsou platné bitcoinové adresy, mohou být podepsány soukromým klíčem, ale jsou to různé adresy!

K vyřešení tohoto problému, když jsou soukromé klíče exportovány z peněženky, Wallet Import Format (WIF) použitý pro jejich reprezentaci je rozdílně implementován v novějších bitcoinových peněženkách, což indikuje, že tyto soukromé klíče byly použit pro výrobu *komprimovaných* veřejných klíčů a proto *komprimované* bitcoinové adresy. To umožňuje importovat peněženky s rozlišením mezi soukromými klíči pocházejících ze starších nebo novějších peněženek a na blockchainu hledat transakce s odpovídající bitcoinovou adresou odpovídající nekomprimovanému nebo komprimovanému veřejnému klíči. V další sekci se podíváme podrobněji na to, jak to funguje.

Komprimované soukromé klíče

Ironicky, pojem "komprimovaný soukromý klíč" je zavádějící, protože když je soukromý klíč exportován jako komprimovaný WIF, je vlastně o jeden byte *delší* než "nekomprimovaný" soukromý klíč. Je to tím, že je k němu přidána přípona 01, která značí, že byl vytvořen novější peněženkou, a že může být použit pouze k vytváření komprimovaných veřejných klíčů. Soukromé klíče nejsou komprimovány a nemohou být komprimovány. Pojem "komprimovaný soukromý klíč" ve skutečnosti znamená "soukromý klíč, ze kterého má být odvozen komprimovaný veřejný klíč, zatímco "nekomprimovaný soukromý klíč" znamená "soukromý klíč, ze kterého má být odvozen nekomprimovaný veřejný klíč." Měli byste odkazovat pouze na formát exportu "komprimovaný WIF" nebo "WIF" a neodkazovat na soukromý klíč jako na komprimovaný, tím nebude docházet k dalším nejasnostem.

Pamatujte, tyto formáty se *nedají* používat zaměnitelně. V novějších peněženkách jsou implementovány komprimované veřejné klíče, soukromé klíče budou exportovány jako

komprimované WIF (s předponou K nebo L). Pokud má peněženka starší implementaci a nepoužívá komprimované veřejné klíče, soukromé klíče budou exportovány jako WIF (s předponou 5). Cílem je oznámit peněžence importující tyto soukromé klíče, zda má v blockchainu hledat komprimované nebo nekomprimované veřejné klíče a adresy.

Pokud je bitcoinová peněženka schopná implementovat komprimované veřejné klíče, bude je používat ve všech transakcích. Soukromé klíče v peněžence budou použity pro odvození veřejných klíčů (bodů na křivce), které budou komprimovány. Komprimované veřejné klíče budou použity k vytvoření bitcoinových adres a ty budou použity v transakcích. Při exportu soukromých klíčů z nové peněženky, která implementuje komprimované veřejné klíče, Wallet Inport Format (WIF) je změněn přidáním jednobytové přípony 01 k soukromému klíči. Výsledný soukromý klíč ve formátu Base58Check je nazýván "komprimovaný WIF" a začíná písmeny "K" nebo "L", místo "5", kterým začínají (nekomprimované) klíče ve formátu WIF ze starších peněženek.

Příklad: stejný klíč, různé formáty ukazuje stejný klíč kódovaný ve formátech WIF a komprimovaném WIF.

Table 4. Příklad: stejný klíč, různé formáty

Formát	**Soukromý klič**
Hexadecimální	1E99423A4ED27608A15A2616A2B0E9E52CED330AC530EDCC32C8FFC6A526AEDD
WIF	5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn
komprimovaný hexadecimální	1E99423A4ED27608A15A2616A2B0E9E52CED330AC530EDCC32C8FFC6A526AEDD_01_
Komprimovaný WIF	KxFC1jmwwCoACiCAWZ3eXa96mBM6tb3TYzGmf6YwgdGWZgawvrtJ

TIP "Komprimované soukromé klíče" je nevhodné označením. Nejsou komprimované, spíše, komprimovaný WIF formát označuje, že by měly být použity k odvození komprimovaného veřejného klíče a odpovídající bitcoinové adresy. Ironicky "komprimovaný WIF" kóduje soukromý klíč s použitím jednoho bytu navíc. Přidáním přípony 01 ho odlišíme od "nekomprimovaného."

Implementace klíčů a adres v Pythonu

Nejrozsáhlejší bitcoinová knihovna v Pythonu je pybitcointools od Vitalika Buterina. V Vytváření a formátování klíčů a adres pomocí knihovny pybitcointools použijeme knihovnu pybitcointools (imporotvanou jako "bitcoin") k vytvoření a zobrazení klíčů a adres v různých formátech.

Example 4. Vytváření a formátování klíčů a adres pomocí knihovny pybitcointools

```
import bitcoin

# Generate a random private key
valid_private_key = False
while not valid_private_key:
    private_key = bitcoin.random_key()
    decoded_private_key = bitcoin.decode_privkey(private_key, 'hex')
    valid_private_key =  0 < decoded_private_key < bitcoin.N

print "Private Key (hex) is: ", private_key
print "Private Key (decimal) is: ", decoded_private_key

# Convert private key to WIF format
wif_encoded_private_key = bitcoin.encode_privkey(decoded_private_key, 'wif')
print "Private Key (WIF) is: ", wif_encoded_private_key

# Add suffix "01" to indicate a compressed private key
compressed_private_key = private_key + '01'
print "Private Key Compressed (hex) is: ", compressed_private_key

# Generate a WIF format from the compressed private key (WIF-compressed)
wif_compressed_private_key = bitcoin.encode_privkey(
    bitcoin.decode_privkey(compressed_private_key, 'hex'), 'wif')
print "Private Key (WIF-Compressed) is: ", wif_compressed_private_key

# Multiply the EC generator point G with the private key to get a public key point
public_key = bitcoin.fast_multiply(bitcoin.G, decoded_private_key)
print "Public Key (x,y) coordinates is:", public_key

# Encode as hex, prefix 04
hex_encoded_public_key = bitcoin.encode_pubkey(public_key,'hex')
print "Public Key (hex) is:", hex_encoded_public_key

# Compress public key, adjust prefix depending on whether y is even or odd
(public_key_x, public_key_y) = public_key
if (public_key_y % 2) == 0:
    compressed_prefix = '02'
else:
    compressed_prefix = '03'
hex_compressed_public_key = compressed_prefix + bitcoin.encode(public_key_x, 16)
print "Compressed Public Key (hex) is:", hex_compressed_public_key

# Generate bitcoin address from public key
print "Bitcoin Address (b58check) is:", bitcoin.pubkey_to_address(public_key)

# Generate compressed bitcoin address from compressed public key
print "Compressed Bitcoin Address (b58check) is:", \
    bitcoin.pubkey_to_address(hex_compressed_public_key)
```

Běh key-to-address-ecc-example.py zobrazuje výstup proběhlého zdrojového kódu.

Example 5. Běh key-to-address-ecc-example.py

Skript demonstrující matematiku eliptických křivek použitou pro bitcoinové klíče je další příklad, používající knihovnu Python ECDSA pro matematiku eliptických křivek bez použití jakékoliv specializované bitcoinové knihovny.

Example 6. Skript demonstrující matematiku eliptických křivek použitou pro bitcoinové klíče

```
import ecdsa
import os
from ecdsa.util import string_to_number, number_to_string

# secp256k1, http://www.oid-info.com/get/1.3.132.0.10
_p = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEFFFFFC2FL
_r = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141L
_b = 0x0000000000000000000000000000000000000000000000000000000000000007L
_a = 0x0000000000000000000000000000000000000000000000000000000000000000L
_Gx = 0x79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798L
_Gy = 0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8L
curve_secp256k1 = ecdsa.ellipticcurve.CurveFp(_p, _a, _b)
generator_secp256k1 = ecdsa.ellipticcurve.Point(curve_secp256k1, _Gx, _Gy, _r)
oid_secp256k1 = (1, 3, 132, 0, 10)
SECP256k1 = ecdsa.curves.Curve("SECP256k1", curve_secp256k1, generator_secp256k1,
oid_secp256k1)
ec_order = _r

curve = curve_secp256k1
generator = generator_secp256k1

def random_secret():
    convert_to_int = lambda array: int("".join(array).encode("hex"), 16)

    # Collect 256 bits of random data from the OS's cryptographically secure random
generator
    byte_array = os.urandom(32)

    return convert_to_int(byte_array)

def get_point_pubkey(point):
    if point.y() & 1:
        key = '03' + '%064x' % point.x()
    else:
        key = '02' + '%064x' % point.x()
    return key.decode('hex')
```

```
def get_point_pubkey_uncompressed(point):
    key = '04' + \
          '%064x' % point.x() + \
          '%064x' % point.y()
    return key.decode('hex')


# Generate a new private key.
secret = random_secret()
print "Secret: ", secret

# Get the public key point.
point = secret * generator
print "EC point:", point

print "BTC public key:", get_point_pubkey(point).encode("hex")

# Given the point (x, y) we can create the object using:
point1 = ecdsa.ellipticcurve.Point(curve, point.x(), point.y(), ec_order)
assert point1 == point
```

Instalace knihovny Python ECDSA a spuštění skriptu ec_math.py ukazuje výstup vytvořený běžícím skriptem.

NOTE

Předchozí příklad používá os.urandom, který je založen na kryptograficky bezpečném generátoru náhodných čísel (CSRNG) poskytnutém operačním systémem. V případě UNIXových operačních systémů jako Linux, čerpá z /dev/urandom; a v případě Windows, volá CryptGenRandom(). Pokud vhodný zdroj náhodnosti není nalezen, vyvolá chybu NotImplementedError. Zde použitý generátor náhodných čísel je vhodný pouze pro demonstrační účely, *není* vhodný pro vytvoření bitcoinových klíčů pro skutečné použití, protože tyto klíče nejsou implementovány s dostatečným zabezpečením.

Example 7. Instalace knihovny Python ECDSA a spuštění skriptu ec_math.py

```
$ # Instalace správce balíčků PIP pro Python
$ sudo apt-get install python-pip
$ # Instalace knihovny  ECDSA pro Python
$ sudo pip install ecdsa
$ # Spuštění skriptu
$ python ec-math.py
Secret: 380908350159543588624811326288874439059062049959123782780601687035806602940000
EC point:
(700488535318671794898577504976069662723825834713229354546245955400072693126 27,
1052622064786867431910608002634795893299202095272858039357360216860455423533 80)
BTC public key: 029ade3effb0a67d5c8609850d797366af428f4a0d5194cb221d807770a1522873
```

Peněženky

Peněženky jsou schránky pro soukromé klíče, obvykle implementovány jako strukturované soubory nebo jednoduché databáze. Další metodou pro vytváření klíčů je *deterministická tvorba klíčů*. Při níž vytvoříte pokaždé nový soukromý klíč za použití jednosměrné hašovací funkce z předchozího soukromého klíče, čímž je spojíte v posloupnost. Tak dlouho dokud budete prodlužovat tuto posloupnost, budete potřebovat pouze první klíč (zvaný jako *semínko* nebo *hlavní klíč*) k vygenerování všech. V této části prozkoumáme různé metody tvorby klíče a peněženkových struktur, které jsou kolem něj postaveny.

TIP

Bitcoinové peněženky obsahují klíče, ne mince. Každý uživatel má peněženku obsahující klíče. Peněženky jsou ve skutečnosti řetězy klíčů obsahujících dvojice soukromého a veřejného klíče viz Soukromý a veřejný klíč). Uživatelé podepisují transakce s klíči, čímž prokazují, že vlastní transakční výstupy (jejich mince). Mince jsou uloženy v blockchainu ve formě transakčních výstupů (často označených jako vout nebo txout).

Nedeterministické (náhodné) peněženky

V prvních bitcoinových klientech byly peněženky jednoduché sbírky náhodně vytvořených soukromých klíčů. Tento typ peněženky je nazván *Typ-0 nedeterministická peněženka*. Například, Bitcoin Core klient předvytvoří 100 náhodných soukromých klíčů při svém prvním spuštění a vytváří další klíče jak je potřeba, každý klíč použije jen jednou. Tento typ peněženek je přezdíván "jen svazek klíčů" (v originále "Just a Bunch Of Keys", zkratka JBOK). Tyto peněženky jsou nahrazovány deterministickými peněženkami z důvodu těžkopádnosti jejich správy, zálohování a importování. Nevýhodou náhodných klíčů je to, že když jich vytváříte mnoho, musíte udržovat kopie všech z nich, což znamená, že peněženka musí být často zálohována. Každý klíč musí být zálohován, nebo peníze, které kontroluje jsou nevratně ztraceny, pokud se peněženka stane nedostupnou. To je v konfliktu s principem vyhýbání se znovupoužívání adres, používání adresy pouze pro jednu transakci. Znovupoužívání adres snižuje soukromí, protože vytváří vazby mezi více transakcemi a adresami. Typ-

0 nedeterministická peněženka je špatnou volbou peněženky, zvláště pokud se chcete vyhnout znovupoužívání adres, protože to znamená správu mnoha klíčů, což vytváří potřebu častého zálohování. Přestože Bitcoin Core klient obsahuje peněženku Typu-0, její používání je nedoporučeno od vývojářů Bitcoin Core. Type-0 nedeterministická (náhodná) peněženka: sbírka náhodně vytvořených klíčů ukazuje nedeterministickou peněženku obsahující širokou sbírku náhodných klíčů.

Deterministické (semínkové) peněženky

Deterministické nebo semínkové peněženky jsou peněženky, které obsahují soukromý klíč, který je odvozen ze společného semínka za použití jednosměrné hašovací funkce. Semínko je náhodně vytvořené číslo, které je složeno s dalšími daty, jako indexové číslo nebo "číslo řetězce" (viz Hierarchické Deterministické Peněženky (BIP0032/BIP0044)) za účelem odvození soukromého klíče. V deterministické peněžence je semínko dostatečné k obnovení všech odvozených klíčů a proto postačí jeho jednorázová záloha v době vytvoření. Semínko je také dostatečné pro export nebo import peněženky, umožňuje snadný přesun uživatelských klíčů mezi různými implementacemi peněženek.

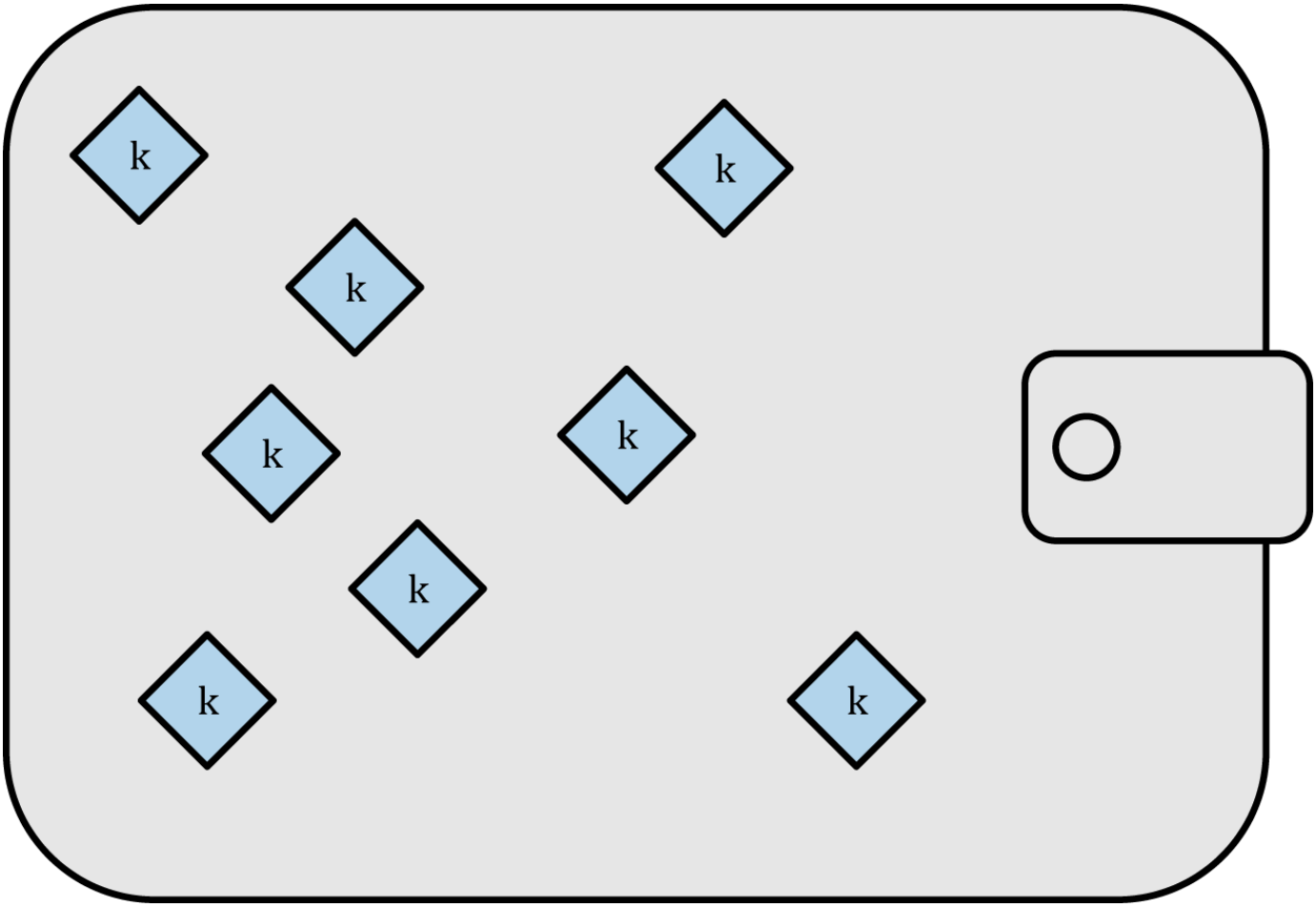


Figure 8. Type-0 nedeterministická (náhodná) peněženka: sbírka náhodně vytvořených klíčů

Mnemotechnická kódová slova

Mnemotechnická kódová slova jsou posloupnosti anglických slov, které reprezentují (kódují) náhodné číslo použité jako semínko pro odvození deterministické peněženky. Tato posloupnost slov je

dostatečná ke znovuvytvoření semínka a z něho ke znovuvytvoření peněženky se všemi odvozenými klíči. Peněženková aplikace, která implementuje deterministické peněženky s mnemotechnickými klíči ukáže uživateli posloupnost 12 až 24 slov při prvním vytvoření peněženky. Tato posloupnost slov je zálohou peněženky a může být použita pro obnovení a znovuvytvoření všech klíčů ve stejné nebo kompatibilní peněženkové aplikaci. Mnemotechnická kódová slova umožňují uživateli snazší zálohování peněženek, protože mohou být snadno přečtena a správně přepsána ve srovnání s náhodnou posloupností čísel.

Mnemotechnické kódy jsou definovány v BIP0039 Návrhu na zlepšení bitcoinu (viz[bip0039]), aktuálně ve stavu přijatého standardu. Konkrétně, existuje jiný standard, s jinou množinou slov, používaný peněženkou Electrum a konkurující BIP0039. BIP0039 je použit v peněžence Trezor a několika dalších peněženkách, ale je nekompatibilní s Electrum implementací.

BIP0039 definuje tvorbu mnemotechnického kódu a semínka následovně:

1. Vytvoření náhodné posloupnosti (entropie) od 128 do 256 bitů. 2.Vytvoření kontrolního součtu náhodné posloupnosti vzetím prvních několika bitů jejího SHA256 haše.
2. Přidání kontrolního součtu na konec náhodné posloupnosti
3. Rozdělení posloupnosti na části po 11 bitech, které slouží jako index do slovníku 2048 předdefinovaných slov.
4. Vyrobení 12 až 24 slov reprezentujících mnemotechnické kódy.

Mnemotechnické kódy: entropie a délka slov ukazuje vztah mezi velikostí entropie dat a délkou mnemotechnických kódových slov.

Table 5. Mnemotechnické kódy: entropie a délka slov

Entropie (bity)	Kontrolní součet (bity)	Entropie+kontrolní součet	Počet slov
128	4	132	12
160	5	165	15
192	6	198	18
224	7	231	21
256	8	264	24

Mnemotechnické kódy reprezentují od 128 do 256 bitů, které jsou použity pro odvození delšího (512-bitového) semínka pomocí funkce na natahování klíče PBKDF2. Výsledné semínko je použito pro vytvoření deterministické peněženky a všech jejích odvozených klíčů.

Tabulky 128-bitová entropie mnemotechnického kódu a výsledné semínko a 256-bitová entropie mnemotechnického kódu a výsledné semínko ukazují některé příklady mnemotechnických kódu a semínek, které vyrobily.

Table 6. 128-bitová entropie mnemotechnického kódu a výsledné semínko

Entropický vstup (128 bitů)	0c1e24e5917779d297e14d45f14e1a1a
Mnemotechnická slova (12)	army van defense carry jealous true garbage claim echo media make crunch
Semínko (512 bitů)	3338a6d2ee71c7f28eb5b882159634cd46a898463e9d2d0980f8e80dfbba5b0fa0291e5fb888a599b44b93187be6ee3ab5fd3ead7dd646341b2cdb8d08d13bf7

Table 7. 256-bitová entropie mnemotechnického kódu a výsledné semínko

Entropy input (256 bitů)	2041546864449caff939d32d574753fe684d3c947c3346713dd8423e74abcf8c
Mnemotechnická slova (24)	cake apple borrow silk endorse fitness top denial coil riot stay wolf luggage oxygen faint major edit measure invite love trap field dilemma oblige
Semínko (512 bitů)	3972e432e99040f75ebe13a660110c3e29d131a2c808c7ee5f1631d0a977fcf473bee22fce540af281bf7cdeade0dd2c1c795bd02f1e4049e205a0158906c343

Hierarchické Deterministické Peněženky (BIP0032/BIP0044)

Deterministické peněženky byly vyvinuty, aby snadno odvodily mnoho klíčů z jednoho "semínka". Nejpokročilejší formou deterministických peněženek je *hierarchická deterministická peněženka* nebo *HD peněženka* definována ve standardu BIP0032. Hierarchické deterministické peněženky obsahují klíče odvozené ze stromové struktury takové, že rodičovský klíč může odvodit posloupnost klíčů dětí, každý z nich může odvodit posloupnost klíčů vnoučat, každý z nich může odvodit posloupnost klíčů pravnoučat, atd. do nekonečné hloubky. Tato struktura je nakreslena v Typ-2 hierarchická deterministická peněženka: strom klíčů vytvořený z jednoho semínka.

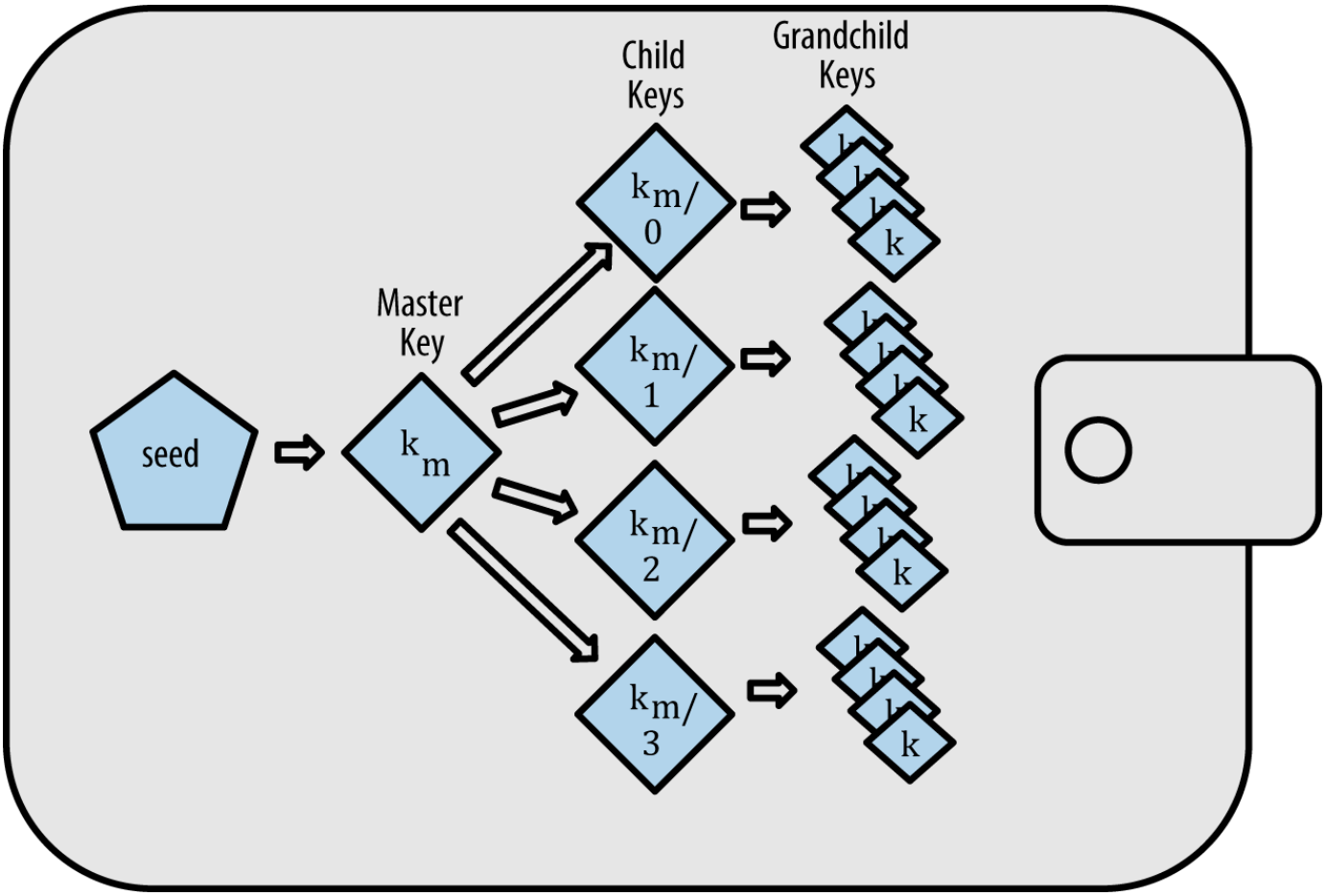


Figure 9. Typ-2 hierarchická deterministická peněženka: strom klíčů vytvořený z jednoho semínka

TIP	Pokud implementujete bitcoinovou peněženku, měli byste jí vytvořit jako HD peněženku dle standardů BIP0032 a BIP0044.

HD peněženky nabízejí dvě hlavní výhody oproti náhodným (nedeterministickým klíčům). Za prvé, stromová struktura může být využita pro vyjádření dodatečných organizačních významů, jako když konkrétní větev podklíčů je použita pro příjem příchozích plateb a jiná větev je použita pro příjem vratek z odchozích transakcí. Větve klíčů mohou být použity ve firemním nastavení, které přiřazuje různé větve oddělením, dceřiným společnostem, konkrétním funkcím nebo účetním kategoriím.

Druhou výhodou HD peněženky je, že uživatelé mohou vytvářet posloupnosti veřejných klíčů aniž by měli přístup k odpovídajícím soukromým klíčům. To umožňuje HD peněžence, aby byla použita na nezabezpečeném serveru, který pouze přijímá finanční prostředky a vydává pro každou transakci nový veřejný klíč, který nemusí být načten ze zálohy nebo odvozen do zásoby, zatímco server nemá soukromý klíč nutný k utracení finančních prostředků.

Tvorba HD peněženky ze semínka

HD peněženky jsou vytvářeny z jednoho *kořenového semínka*, které je 128-, 256--, nebo 512-bitové číslo. Všechno ostatní je v HD peněžence deterministicky odvozeno z tohoto kořenového semínka, což umožňuje vytvořit znovu celou HD peněženku z tohoto semínka v jakékoliv kompatibilní HD peněžence. Díky tomu je snadné zálohovat, obnovovat, exportovat a importovat HD peněženky

obsahující tisíce nebo dokonce miliony klíčů pouhým přenosem jediného kořenového semínka. Kořenové semínko je nejčastěji reprezentováno *mnemotechnickými kódovými slovy*, které byly popsány v předchozí části Mnemotechnická kódová slova, a které usnadňují lidem přepis a uložení.

Postup tvorby hlavního klíče a kódu hlavního řetězu pro HD peněženku je znázorněn Tvorba hlavních klíčů a kódu řetězu z kořenového semínka..

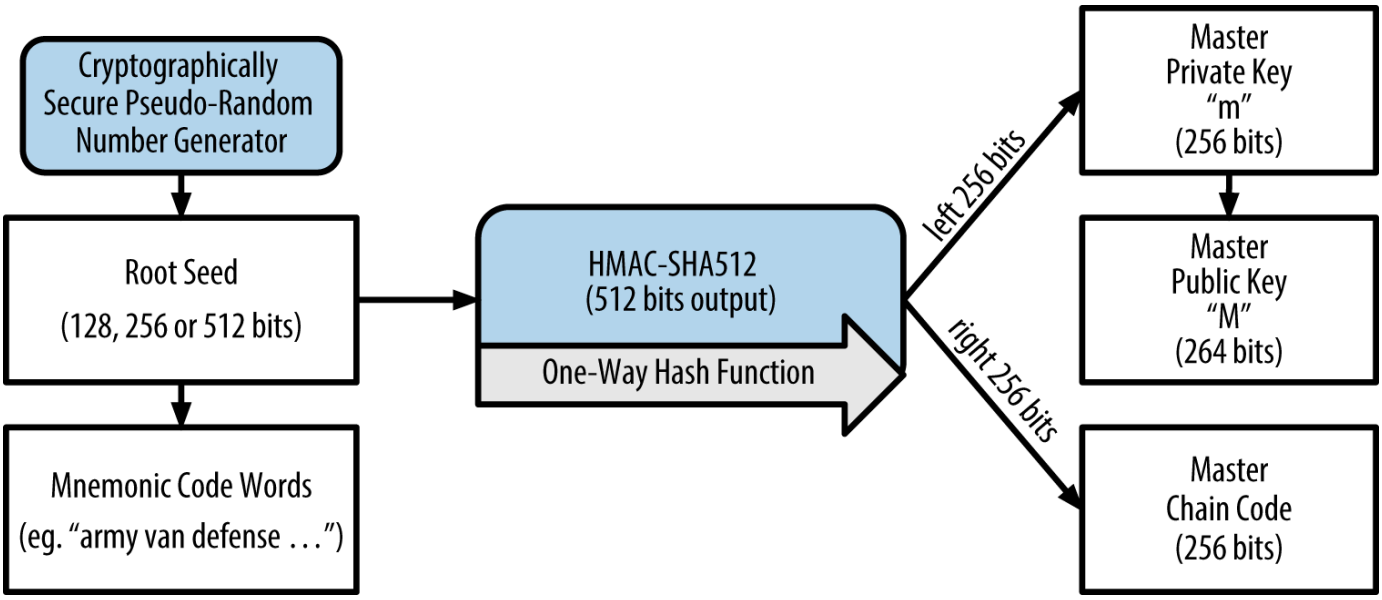


Figure 10. Tvorba hlavních klíčů a kódu řetězu z kořenového semínka.

Kořenové semínko je vstupem algoritmu HMAC-SHA512 a výsledný haš je použit pro vytvoření *hlavního soukromého klíče* (m) a *kódu hlavního řetězu*. Hlavní soukromý klíč vytváří odpovídající hlavní veřejný klíč (M), během běžného postupu násobení eliptické křivky m * G, jak bylo popsáno dříve v této kapitole. Kód řetězu je použit pro zavedení entropie do funkce, které tvoří klíče dětí z rodičovského klíče, jak jsme viděli v předchozí části.

Odvození soukromého klíče dítěte

Hierarchické deterministické peněženky používají funkci *tvorba klíče dítěte* (v originále *child key derivation*, zkratka CKD) pro vytvoření klíče dítěte z rodičovského klíče.

Odvození klíče dítěte je založeno na jednosměrné hašovací funkci, která skládá:

- Rodičovský soukromý nebo veřejný klíč (ECDSA nekomprimovaný klíč)
- semínko zvané kód řetězu (256 bitů)
- indexové číslo (32 bitů)

Kód řetězu je použit k zavedení zdánlivě náhodných dat do tohoto postupu, takže index sám o sobě nestačí k odvození dalších klíčů dětí. Proto, ze znalosti klíče dítěte není možné najít jeho sourozence, pokud nemáme také kód řetězu. Počáteční semínko kódu řetězu (v kořenu stromu) je vytvořeno z náhodných dat, zatímco následné kódy řetězu jsou odvozeny z každého kódu rodičovského řetězu.

Tyto tři položky jsou spojeny a je z nich spočítán haš pro vytváření klíčů dětí, a to následovně.

Rodičovský veřejný klíč, kód řetězu a indexové číslo jsou spojeny a je vytvořen 512-bitový haš za použití algoritmu HMAC-SHA512. Výsledný haš je rozdělen do dvou polovin. Pravých 256 bitů výsledného haše se stane kódem řetězu pro dítě. Levých 256 výsledného haše a indexové číslo jsou přidány k rodičovskému soukromému klíči, aby vytvořili soukromý klíč dítěte. Toto je nakresleno v Rozšíření rodičovského soukromého klíče, pro vytvoření soukromého klíče dítěte.pro index nastavený na 0 při vzniku 0-tého (prvního od indexu) dítěte rodiče.

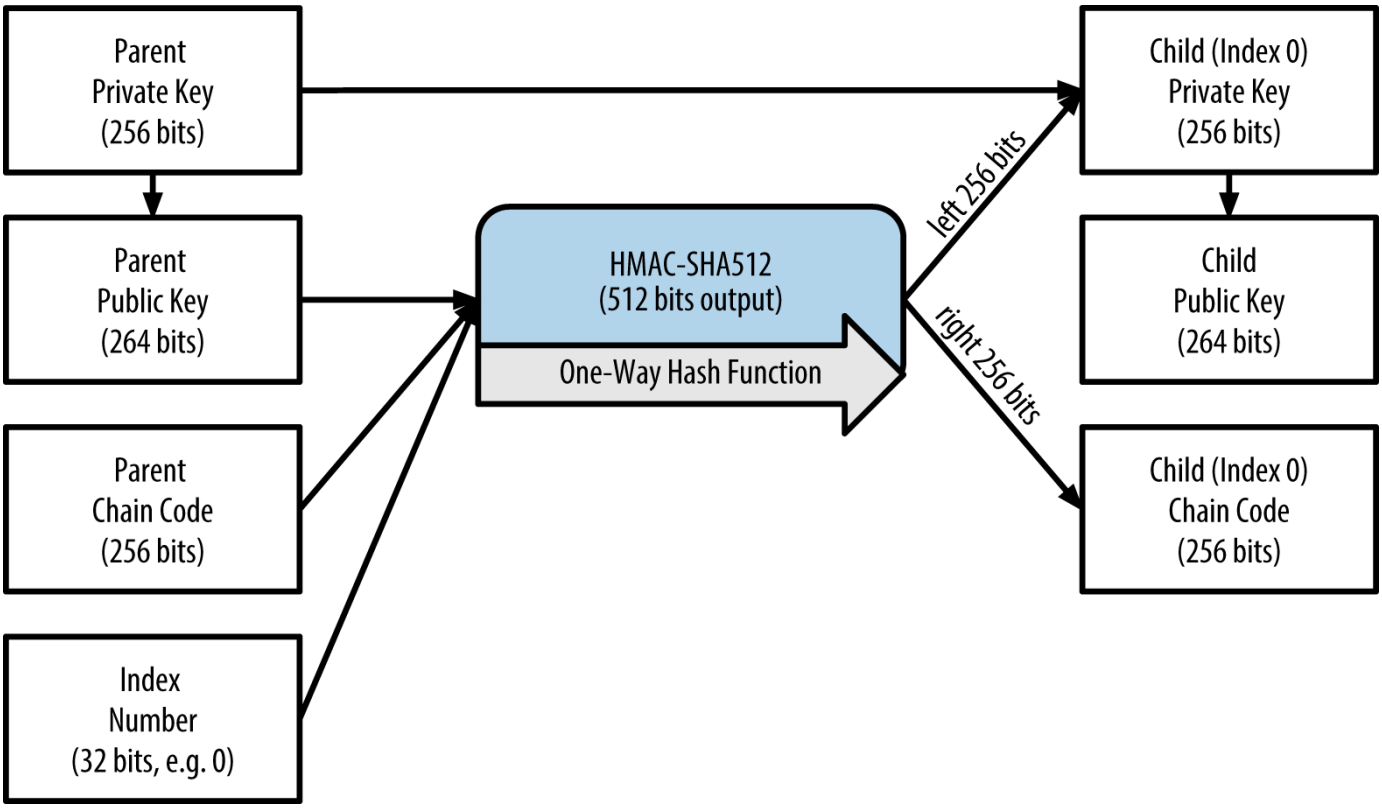


Figure 11. Rozšíření rodičovského soukromého klíče, pro vytvoření soukromého klíče dítěte.

Změna indexu nám umožňuje rozšířit rodiče a vytvořit další děti v této posloupnosti, např. Dítě 0, Dítě 1, Dítě 2, atd. každý rodičovský klíč může mít 2 miliardy klíčů dětí.

Opakováním postupu o jednu úroveň níže ve stromu, každé dítě se může stát rodičem a vytvořit své vlastní děti, nekonečného počtu generací.

Použití odvozených klíčů dětí

Soukromé klíče dětí jsou nerozeznatelné od nedeterministických (náhodných klíčů). Protože odvozovací funkce je jednosměrná, klíč dítěte nemůže být použit k nalezení klíče rodiče. Klíč dítěte nemůže být použit k nalezení jakýchkoliv sourozenců. Pokud máme n-té dítě, nemůžeme najít jeho sourozence, jako n-1-ní dítě, n+1-ní dítě, nebo jiné děti, které jsou částí posloupnosti. Pouze rodičovský klíč a kód řetězu mohou odvodit všechny děti. Bez kódu řetězu dítěte, nemohou být děti ani použity k odvození vnoučat.

K čemu můžeme soukromý klíč dítěte samostatně použít? Může být použit k vytvoření veřejného klíče a bitcoinové adresy. Tedy, může být použit pro podepisování transakcí a utrácení finančních prostředků zaslaných na tuto adresu.

TIP	Soukromý klíč dítěte, odpovídající veřejný klíč a bitcoinová adresa jsou k nerozeznání od klíčů a adres vytvořených náhodně. Skutečnost, že jsou součástí posloupnosti není viditelná mimo funkčnost HD peněženky, která je vytvořila. Jakmile jsou vytvořeny, mohou pracovat stejně jako "normální" klíče.

Rozšířené klíče

Jak jsme viděli dříve, funkce pro odvození klíče může být použita k vytvoření dětí na jakékoliv úrovni ve stromě na základě tří vstupů: klíč, kód řetězu a index požadovaného dítěte. Dvě nezbytné složky jsou klíč a kód řetězu a jejich kombinace se nazývá *rozšířený klíč*. Pojem "rozšířený" může být chápán jako "rozšiřitelný klíč", protože takovýto klíč může být použit k odvození dětí.

Rozšířené klíče jsou uloženy a reprezentovány jednoduše jako zřetězení 256-bitového klíče a 256-bitového kódu do 512-bitové posloupnosti. Jsou dva typy rozšířených klíčů. Rozšířený soukromý klíč je kombinací soukromého klíče a kódu řetězu a může být použit pro odvození soukromých klíčů dětí (a z nich poté veřejných klíčů). Rozšířený veřejný klíč je kombinace veřejného klíče a kódu řetězu, který může být použit k vytvoření veřejných klíčů dětí, jak je popsáno v Vytvoření veřejného klíče.

Představte si rozšířený klíč jako kořen větve ve stromové struktuře HD peněženky. Z kořene větve můžete odvodit zbytek větve. Rozšířený veřejný klíč může vytvořit jen větev veřejných klíčů.

TIP	Rozšířený klíč se skládá ze soukromého klíče nebo veřejného klíče a kódu řetězu. Rozšířený klíč může vytvořit dítě, vytvořit svojí vlastní větev ve stromové struktuře. Sdílení rozšířeného klíče dává přístup k celé větvi.

Rozšířené klíče jsou kódovány pomocí Base58Check, takže mohou být snadno importovány a exportovány mezi různými BIP0032-kompatibilními peněženkami. Kódování Base58Check pro rozšířené klíče používá zvláštní číslo verze, které vede k předponě "xprv" a "xpub" při zakódování do znaků Base58, což je činí snáze rozeznatelné. Protože rozšířený klíč má 512 nebo 513 bitů, je také o mnoho delší než ostatní řetězce kódované Base58Check, které jsme viděli dříve.

Zde je příklad rozšířeného soukromého klíče, zakódovaný v Base58Check:

```
xprv9tyUQV64JT5qs3RSTJkXCWKMyUgoQp7F3hA1xzG6ZGu6u6Q9VMNjGr67Lctvy5P8oyaYAL9CAWrUE9i6GoNMK
Uga5biW6Hx4tws2six3b9c
```

Zde je příklad rozšířeného soukromého klíče, kódovaný Base58Check:

```
xpub67xpozcx8pe95XVuZLHXZeG6XWXHpGq6Qv5cmNfi7cS5mtjJ2tgypeQbBs2UAR6KECeeMVKZBPLrtJunSDMst
weyLXhRgPxdp14sk9tJPW9
```

Odvození veřejného klíče dítěte

Jak bylo dříve zmíněno, velmi užitečnou vlastností hierarchických deterministických peněženek je

schopnost odvozovat veřejné klíče dětí z veřejných klíčů rodičů *bez* znalosti soukromých klíčů. To nám dává dva způsoby odvození veřejného klíče dítěte: buďto ze soukromého klíče dítěte nebo přímo z veřejného klíče rodiče.

Rozšířený veřejný klíč může být tedy použit k odvození všech veřejných klíčů (ale pouze veřejných klíčů) v této větvi struktury HD peněženky.

Tato zkratka může být použita pro vytvoření velmi bezpečných nasazení výhradně veřejných klíčů, když server nebo aplikace má kopii rozšířeného veřejného klíče a vůbec žádné soukromé klíče. Tento způsob nasazení může vyrobit nekonečné množství veřejných klíčů a bitcoinových adres, ale nemůže utratit žádné peníze zaslané na tyto adresy. Mezitím na dalším více bezpečném serveru může rozšířený soukromý klíč odvodit odpovídající soukromé klíče a podepisovat transakce a utrácet peníze.

Jednou běžnou aplikací tohoto řešení je nainstalovat rozšířený veřejný klíč na webový server, který provozuje elektronickou obchodní aplikaci. Webový server může použít funkci odvození veřejného klíče k vytvoření nové bitcoinové adresy pro každou transakci (např. nákupní vozík zákazníka). Webový server nebude mít žádné soukromé klíče, které by mohly být zranitelné při krádeži. Bez HD peněženek, by toto šlo dělat pouze vygenerováním tisíců bitcoinových adres na odděleném bezpečném serveru a jejich nahrání v předstihu na server elektronického obchodu. Tento postup je těžkopádný a požaduje neustálou údržbu, abychom se ujistili, že server elektronické obchodní aplikace "nevyčerpal" zásobu klíčů.

Další běžnou aplikací tohoto řešení je studené úložiště nebo hardwarová peněženka. V tomto scénáři může být rozšířený soukromý klíč uložen na papírové peněžence nebo hardwarovém zařízení (jako hardwarová peněženka Trezor), zatímco rozšířený veřejný klíč je udržován online. Uživatel může vytvářet "přijímací" adresy dle libosti, zatímco soukromé klíče jsou bezpečně uloženy offline. Pro utracení finančních prostředků, uživatel může použít rozšířený soukromý klíč na offline bitcoinovém klientovi podpisujícím transakce nebo podepsat transakce v hardwarové peněžence (např. Trezor). Rozšíření veřejného klíče rodiče pro vytvoření veřejného klíče dítěte. vykresluje mechanizmus pro rozšíření veřejného klíče rodiče k odvození veřejného klíče dítěte.

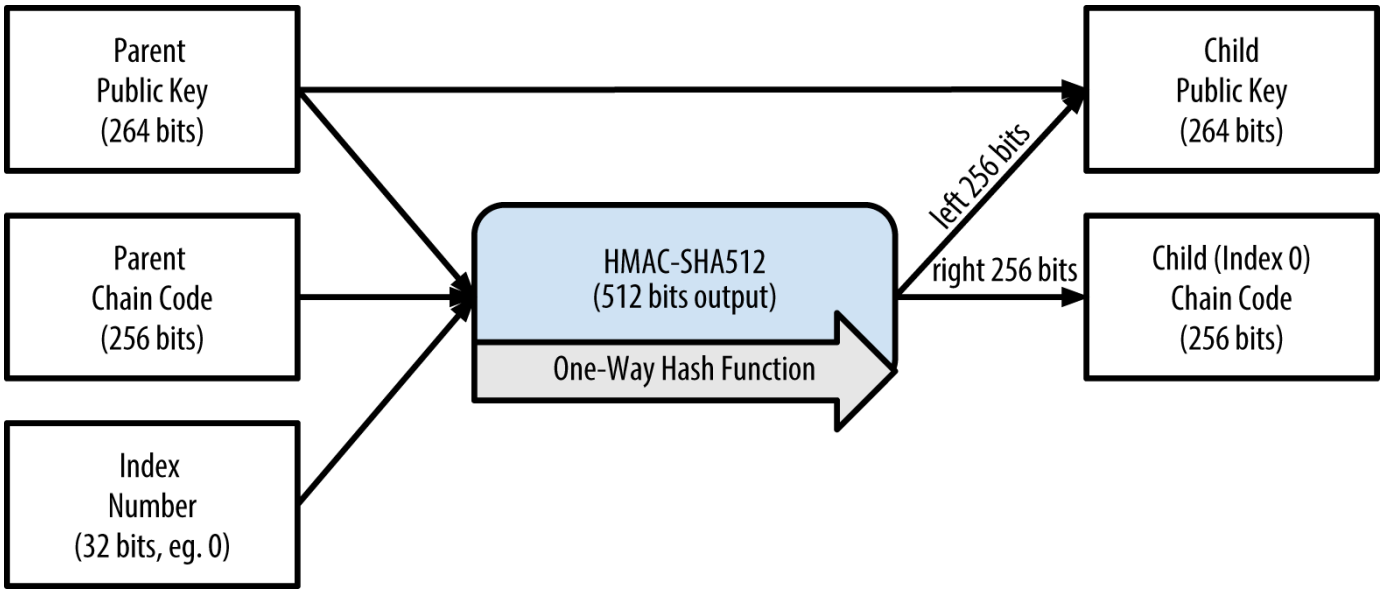


Figure 12. Rozšíření veřejného klíče rodiče pro vytvoření veřejného klíče dítěte.

Odvození tvrzeného klíče dítěte

Schopnost odvodit větev veřejných klíčů z rozšířeného veřejného klíče je užitečná, ale přináší potenciální riziko. Přístup k rozšířenému veřejnému klíči nedává přístup k soukromým klíčům dětí. Nicméně, protože rozšířený veřejný klíč obsahuje kód řetězu, pokud je znám nebo prozrazen soukromý klíč dítěte, může být použit s kódem řetězu k odvození všech ostatních soukromých klíčů dětí. Jeden prozrazený soukromý klíč dítěte společně s rodičovským kódem řetězu odhaluje všechny soukromé klíče všech dětí. Hůře, soukromý klíč dítěte spolu s rodičovským kódem řetězu může být použit k odvození soukromého klíče rodiče.

Aby se zabránilo tomuto riziku, HD peněženky používají alternativní odvozovací funkci zvanou *tvrzené odvození*, které přerušuje vztah mezi rodičovským veřejným klíčem a kódem řetězu dítěte. Funkce tvrzeného odvození používá soukromý klíč rodiče k odvození kódu řetězu dítěte, místo veřejného klíče rodiče. To tvoří "firewall" v posloupnosti rodič/dítě, kód řetězu nemůže být použit k ohrožení soukromých klíčů rodiče nebo sourozenců. Tvrzená odvozovací funkce vypadá téměř shodně jako normální odvození soukromého klíče dítěte, pouze na vstupu je použit soukromý klíč rodiče místo veřejného klíče rodiče, jak ukazuje diagram Tvrzené odvození klíče dítěte, vynechán veřejný klíč rodiče.

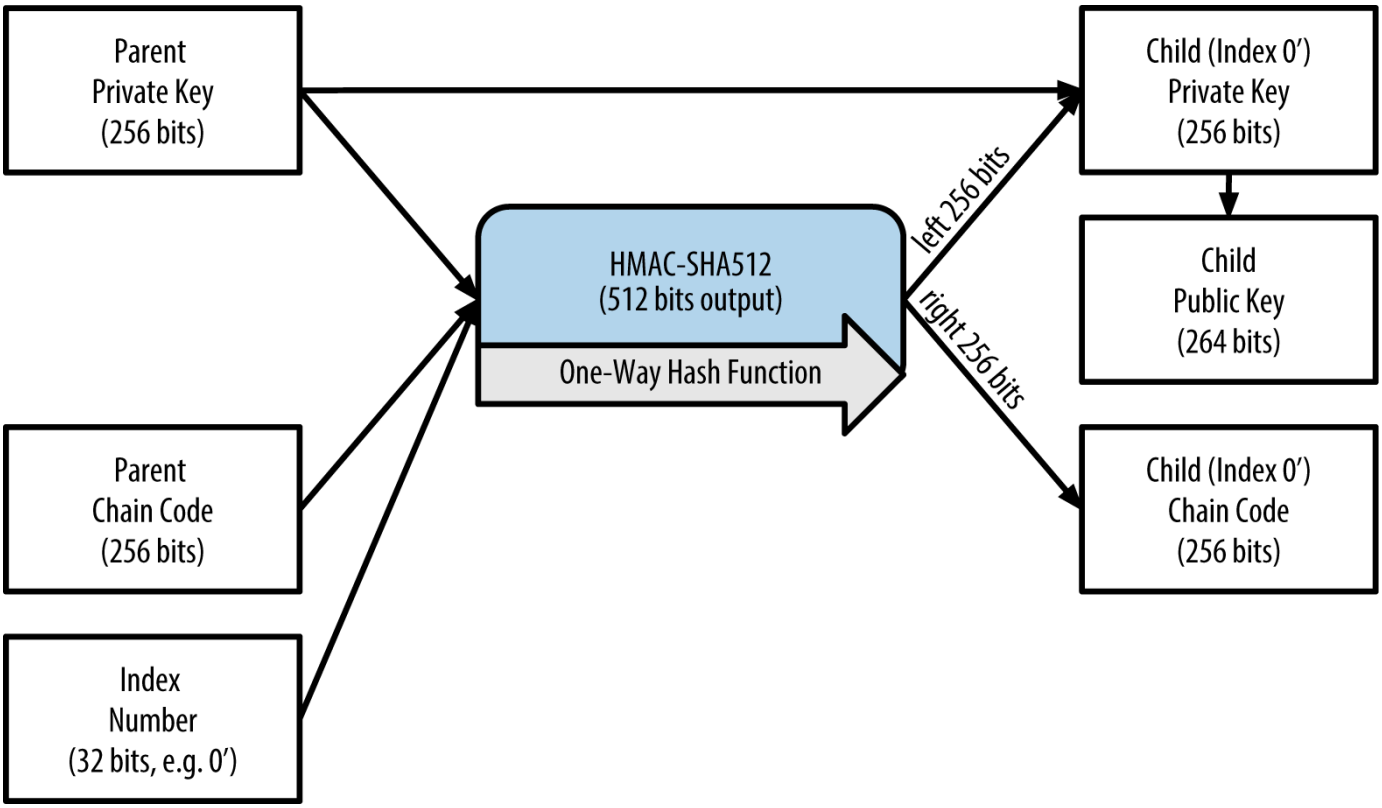


Figure 13. Tvrzené odvození klíče dítěte, vynechán veřejný klíč rodiče

Když je použita funkce odvození tvrzeného soukromého klíče, výsledný soukromý klíč dítěte a kód řetězce jsou zcela různé od těch, které bychom získali normální odvozovací funkcí. Výsledná "větev" klíčů může být použita pro výrobu veřejných klíčů, které nejsou zranitelné, protože kód řetězu nemůže být zneužit k odhalení řádného soukromého klíče. Tvrzené odvození proto vytváří "mezeru" ve stromu nad úrovní, kde rozšířené veřejné klíče byly použity.

Jednoduše řečeno, pokud chcete používat pohodlí rozšířených veřejných klíčů k odvození větví veřejných klíčů, bez vystavení se riziku prozrazeného kódu řetězu, měli byste ho odvozovat z tvrzeného rodiče, místo normálního rodiče. Nejlepší praxí je, že děti 1. úrovně hlavních klíčů jsou vždy odvozeny pomocí tvrzeného odvození, což zabraňuje zneužití hlavních klíčů.

Indexová čísla pro normální a tvrzené odvození

Indexové číslo použité v odvozovací funkci je 32-bitové celé číslo. Pro snadné rozlišení mezi klíči odvozenými normálním odvozením a tvrzeným odvozením, jsou indexová čísla rozdělena do dvou intervalů. Čísla indexů mezi 0 až 2^{31}-1 (0x0 až 0x7FFFFFFF) jsou použita *pouze* pro normální odvození. Čísla indexů mezi 2^{31} až 2^{32}–1 (0x80000000 to 0xFFFFFFFF) jsou použita *pouze* pro tvrzené odvození. Proto, čísla indexů nižší než 2^{31} znamenají, že dítě je normální, zatímco čísla indexů rovna nebo vyšší 2^{31} znamenají, že dítě je tvrzené.

Aby indexová čísla bylo snazší číst a zobrazovat, indexová čísla pro tvrzené děti jsou zobrazena počínaje nulou, ale s čárkou. První normální klíč dítěte je proto zobrazen jako 0, zatímco první tvrzené dítě (index 0x80000000) je zobrazeno jako 0'. V jejich posloupnosti, druhý tvrzený klíč by měl mít index 0x80000001 a měl by být zobrazen jako 1', atd. Když uvidíte v HD peněžence index i', znamená 2^{31}+i.

HD peněženky: identifikátory klíčů (cesty)

Klíče v HD peněžence jsou identifikovány za použití pojmenovávací konvence "cesta", ve které je každá úroveň stromu oddělena znakem (/) (viz HD peněženka: příklady cest). Soukromé klíče odvozené z hlavního klíče začínají písmenem "m". Veřejné klíče odvozené z hlavního klíče začínají písmenem "M". Proto, první soukromý klíč dítěte odvozený z hlavního soukromého klíče je m/0. První veřejný klíč dítěte je M/0. Druhé vnouče, jehož otec je první dítě, je m/0/1, atd.

Rodokmen klíčů se čte zprava doleva, dokud nenarazíme na hlavní klíč, ze kterého byl odvozen. Například, označení m/x/y/z popisuje klíč, který je z-tým dítětem klíče m/x/y, který je y-tým dítětem klíče m/x, který je x-tým dítětem klíče m.

Table 8. HD peněženka: příklady cest

HD cesta	Popis klíče
m/0	Soukromý klíč prvního (0) dítěte odvozený z hlavního soukromého klíče (m)
m/0/0	Soukromý klíč prvního vnoučete od prvního syna (m/0)
m/0'/0	První normální vnouče od prvního *tvrzeného* dítěte (m/0')
m/1/0	Soukromý klíč prvního vnoučete od druhého syna (m/1)
M/23/17/0/0	Veřejný klíč Prvního prapravnoučete od prvního pravnoučete od 18-tého vnoučete od 24-tého syna

Navigace ve stromové struktuře HD peněženky

Struktura HD peněženky nabízí obrovskou pružnost. Každý rodičovský rozšířený klíč může mít 4 miliardy dětí, 2 miliardy normálních dětí a 2 miliardy tvrzených dětí. Každý z těchto dětí může mít další 4 miliardy dětí, atd. Strom může být hluboký jak chcete, s nekonečným počtem generací. Se vší touto pružností, nicméně, se navigace v tomto nekonečném stromě stává docela obtížnou. Obzvláště je těžké přenášet HD peněženky mez implementacemi, protože možnosti vnitřní organizace do větví a podvětví jsou nekonečné.

Dva návrhy na vylepšení bitcoinu (BIP) nabízejí řešení této složitosti vytvořením standardů pro strukturu stromů v HD peněženkách. BIP0043 navrhuje užití indexu prvního tvrzeného syna jako speciálního identifikátoru, který označuje účel této stromové struktury. Na základě BIP0043 by HD peněženka měla používat pouze větve stromu úrovně 1, s indexovými čísly identifikujícími strukturu a jmenný prostor zbytku stromu definovaného jeho účelem. Například HD peněženka užívající pouze větev m/i'/ je určena pro specifický důvod, který je identifikován indexovým číslem "i".

Rozšíření specifikace BIP0044 navrhuje strukturu pro více účtů pod číslem 44' v BIP0043. Všechny HD peněženky řídíce se BIP0044 se rozpoznají tak, že používají pouze jedinou větev ve stromu m/44'/.

BIP0044 specifikuje strukturu skládající se z pěti předdefinovaných úrovní stromů:

m / důvod' / typ_mince' / účet' / vratka / index_adresy

První úroveň "důvod" je vždy nastavena na 44'. Druhá úroveň "typ_mince" určuje typ kryptoměny, umožňuje HD peněženky pro více různých měn, ve kterých každá měna má svůj vlastní podstrom druhé úrovně. Zatím jsou definovány tři kryptoměny Bitcoin je m/44'/0', Testovací síť Bitcoinu m/44'/1' a Litecoin je m/44'/2'.

Třetí úrovní stromu je "účet", který umožňuje uživateli rozdělit jejich peněženky na oddělené logické účty pro účetní a organizační důvody. Například HD peněženka může obsahovat dva bitcoinové "účty": m/44'/0'/0' a m/44'/0'/1'. každý účet je strom jeho vlastního podstromu.

Na čtvrté úrovni "vratka" má HD peněženka dva podstromy, jeden pro vytváření přijímacích adres a jeden pro vytváření adres vratek. Zatímco předchozí úrovně používaly tvrzené odvození, tato úroveň používá normální odvození. To umožňuje této úrovni exportovat rozšířený veřejný klíč pro použití v nezabezpečeném prostředí. Použitelné adresy jsou odvozeny z HD peněženky jako děti čtvrté úrovně, čímž tvoří pátou úroveň "index adresy". Například, třetí přijímací adresa pro bitcoinové platby v prvním účtu bude M/44'/0'/0'/0/2. Příklady struktur HD peněženek dle BIP0044 ukazuje několik dalších příkladů.

Table 9. Příklady struktur HD peněženek dle BIP0044

HD cesta	Popis klíče
M/44'/0'/0'/0/2	Třetí veřejný klíč pro příjem na prvním bitcoinovém účtě

HD cesta	Popis klíče
M/44'/0'/3'/1/14	Patnáctý veřejný klíč pro vratky na čtvrtém bitcoinovém účtu
m/44'/2'/0'/0/1	Druhý soukromý klíč pro podpisy transakcí pro první účet pro Litecoin

Pokusy s HD peněženkami za použití Bitcoinového Průzkumníka

Použití příkazů Bitcoinového Průzkumníka je představeno v [ch03_bitcoin_client], můžete experimentovat s tvorbou a rozšiřováním BIP0032 deterministických klíčů, jako s jejich zobrazováním v různých formátech.:

```
$ bx seed | bx hd-new > m # Vytvoří nový hlavní soukromý klíč ze semínka a uloží ho do
souboru
$ cat m # ukáže hlavní rozšířený soukromý klíč
xprv9s21ZrQH143K38iQ9Y5p6qoB8C75TE71NfpyQPdfGvzghDt39DHPFpovvtWZaRgY5uPwV7RpEgHs7cvdg
fiSjLjjbuGKGcjRyU7RGGSS8Xa
$ cat m | bx hd-public # vytvoří rozšířený veřejný klíč M/0
xpub67xpozcx8pe95XVuZLHXZeG6XWXHpGq6Qv5cmNfi7cS5mtjJ2tgypeQbBs2UAR6KECeeMVKZBPLrtJunS
DMstweyLXhRgPxdp14sk9tJPW9
$ cat m | bx hd-private # vytvoří rozšířený soukromý klíč m/0
xprv9tyUQV64JT5qs3RSTJkXCWKMyUgoQp7F3hA1xzG6ZGu6u6Q9VMNjGr67Lctvy5P8oyaYAL9CAWrUE9i6G
oNMKUga5biW6Hx4tws2six3b9c
$ cat m | bx hd-private | bx hd-to-wif # ukáže soukromý klíč m/0 ve formátu WIF
L1pbvV86crAGoDzqmgY85xURkz3c435Z9nirMt52UbnGjYMzKBUN
$ cat m | bx hd-public | bx hd-to-address # zobrazí bitcoinovou adresu příslušnou k
M/0
1CHCnCjgMNb6digimckNQ6TBVcTWBAmPHK
$ cat m | bx hd-private | bx hd-private --index 12 --hard | bx hd-private --index 4 #
vytvoří m/0/12'/4
xprv9yL8ndfdPVeDWJenF18oiHguRUj8jHmVrqqD97YQHeTcR3LCeh53q5PXPkLsy2kRaqgwoS6YZBLatRZRy
UeAkRPe1kLR1P6Mn7jUrXFquUt
```

Pokročilé klíče a adresy

V následujících sekcích se podíváme na pokročilé formy klíčů a adres, jako šifrované soukromé klíče, skripty, vícepodpisové adresy, okrasné adresy a papírové peněženky.

Šifrované soukromé klíče (BIP0038)

Soukromé klíče musejí zůstat utajené. Potřeba *důvěrnosti* soukromých klíčů je pravdou, ale je docela těžké jí dosáhnout v praxi, protože je v konfliktu se stejně důležitým bezpečnostním cílem *dostupností*.

Udržování soukromých klíčů v tajnosti je mnohem těžší, když potřebujete uložit zálohu soukromého klíče, abyste se vyhnuly jeho ztrátě. Soukromý klíč uložený v peněžence, která je zašifrovaná heslem, může být bezpečný, ale peněženka potřebuje být zálohovaná. Čas od času uživatelé potřebují přestěhovat klíče z jedné peněženky do jiné - například aktualizovat nebo nahradit peněženkový software. Zálohy soukromých klíčů mohou být uloženy na papíru viz Papírové peněženky) nebo na externím úložném médiu, jako je USB flash disk. Co ale, pokud je sama záloha ztracena nebo ukradena? Tento konflikt bezpečnostních cílů vede k zavedení přenosného a pohodlnějšího standardu pro šifrování soukromých klíčů, který může být pochopen mnohými různými peněženkami a bitcoinovými klienty, standardizovaný v návrhu vylepšení bitcoinu 38, nebo BIP0038 (viz [bip0038]).

BIP0038 navrhuje obecný standard pro šifrování soukromých klíčů pomocí heslové fráze a jejich kódování s Base58Check, takže mohou být bezpečně uloženy na záložním médiu, přenášeny bezpečně mezi peněženkami, nebo uchovávány v jiných podmínkách, kde klíče mohou být vystaveny. Standard pro šifrování používá Pokročilý šifrovací standard (AES), standard ustanovený Národním úřadem pro standardy a technologie (NIST), a je široce používán při implementaci šifrování dat pro komerční a vojenské aplikace.

A BIP0038 šifrovací schéma bere na vstupu bitcoinový soukromý klíč, obvykle kódovaný ve WIF jako Base58Check řetězec s předponou "5". Navíc šifrovací schéma BIP0038 bere frázi - dlouhé heslo - obvykle složené z několika slov nebo složitý řetězec alfanumerických znaků. Výsledek šifrovacího schématu BIP0038 je Base58Check kódovaný zašifrovaný soukromý klíč, který začíná předponou 6P. Pokud uvidíte klíč začínající 6P, znamená to, že je šifrovaný a potřebuje heslovou frázi pro dešifrování zpět do WIF formátovaného soukromého klíče (předpona 5), který může být použit v jakékoliv peněžence. Mnoho peněženkových aplikací nyní rozpoznává BIP0038 šifrované soukromé klíče a požádá uživatele o heslovou frázi pro dešifrování a import klíče. Aplikace třetích stran, jako neskutečně užitečné prohlížečové Bit Address (Wallet Details tab), můžou být použity pro dešifrování BIP0038 klíčů.

Nejvíce používaný případ BIP0038 šifrovaných klíčů je papírová peněženka. která se používá pro zálohování soukromých klíčů na kus papíru. Jakmile uživatel zvolí silnou frázi, papírová peněženka s BIP0038 zakódovanými klíči je neskutečně bezpečná a je skvělým způsobem jak vytvořit offline bitcoin úložiště (také zvané "studené úložiště".

Test šifrovaných klíčů v Příklad BIP0038 zakódovaného soukromého klíče používá bitaddress.org k ukázce jak lze získat zašifrovaný klíč vložením heslové fráze.

Table 10. Příklad BIP0038 zakódovaného soukromého klíče

Soukromý klíč (WIF)	5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn
*Heslová fráze	MyTestPassphrase
Zašifrovaný klíč (BIP0038)	6PRTHL6mWa48xSopbU1cKrVjpKbBZxcLRRCdctLJ3z5yxE87MobKoXdTsJ

Platba haši skriptu (P2SH) a vícepodpisové adresy

Jak víme, tradiční bitcoinové adresy začínají číslem "1" a jsou odvozeny z veřejného klíče, který je odvozen ze soukromého klíče. Přestože kdokoliv může zaslat bitcoiny na "1" adresu, tyto bitcoiny mohou být utraceny jen prokázáním se odpovídajícím podpisem soukromého klíče odpovídajícího haši veřejného klíče.

Bitcoinové adresy, které začínají číslem "3" jsou adresy platby haši skriptu (v originále pay-to-script hash, P2SH), občas chybně nazývány vícepodpisové adresy. Jsou navrženy tak, že příjemce bitcoinové transakce je haš skriptu místo majitele veřejného klíče. Tato funkce byla představena v lednu 2012 v návrhu vylepšení bitcoinu 16 nebo BIP0016 (viz [bip0016]), a je široce přijímána, protože poskytuje příležitost přidat funkcionalitu do adresy samotné. Na rozdíl od transakcí, které posílají finanční prostředky na tradiční "1" adresy, také známé platba haši veřejného klíče (v originále pay-to-public-key-hash, zkratka P2PKH), finanční prostředky zaslané na "3" adresy požadují něco dalšího než jen prokázání jednoho haše veřejného klíče a jednoho podpisu soukromého klíče jako důkazu vlastnictví. Požadavky jsou navrženy v čase, kdy je adresa vytvořena, skriptem. Všechny vstupy na tuto adresu budou zatíženy stejnými požadavky.

Adresa platby haši skriptu je vytvořena z transakčního skriptu, který definuje, kdo může utratit transakční výstupy (více podrobností viz [p2sh]). Kódování platby adrese haši skriptu zahrnuje použití stejné dvojité hašovací funkce, která je použita při vytváření bitcoinové adresy, pouze je aplikována na skript místo na veřejný klíč.

```
script hash = RIPEMD160(SHA256(script))
```

Výsledný "haš skriptu" je zakódován Base58Check s předponou verze 5, která je výsledkem zakódování adresy začínající s 3. Příkladem P2SH adresy je 3F6i6kwkevjR7AsAd4te2YB2zZyASEm1HM, která může být odvozena za použití Bitcoinového Průzkumníka příkazy script-encode, sha256, ripemd160, and base58check-encode (viz [libbitcoin]) následovně:

```
$ echo dup hash160 [ 89abcdefabbaabbaabbaabbaabbaabbaabbaabba ] equalverify checksig >
script
$ bx script-encode < script | bx sha256 | bx ripemd160 | bx base58check-encode --version
5
3F6i6kwkevjR7AsAd4te2YB2zZyASEm1HM
```

TIP	P2SH není nezbytně to samé jako vícepodpisová standardní transakce. P2SH adresa *nejčastěji* reprezentuje vícepodpisový skript, ale může reprezentovat skript kódující jiný typ transakcí

Vícepodpisové adresy a P2SH

Nyní, nejrozšířenější implementace P2SH funkce je skript vícepodpisové adresy. Jak říká název,

podkladový skript požaduje více než jeden popis dokazující vlastnictví a pro utracení finančních prostředků. Bitcoinová vícepodpisová funkce je navržena, že požaduje M podpisů (známe také jako prahová hodnota") z celkového počtu N klíčů, známá jako M-z-N vícepodpisovost, kde M je rovno nebo menší N. Například vlastník kavárny Bob z [ch01_intro_what_is_bitcoin] bude používat vícepodpisovou adresu požadující 1-ze-2 podpisů z klíčů náležejících jemu a jeho manželce, zajišťujících, že kdokoliv z nich může utratit transakční výstupy zamčené na této adrese. Je to podobné jako "vícemajitelový účet" implementovaný v tradičním bankovnictví, kde každý z manželů mohou utrácet peníze jediným podpisem. Nebo Gopesh, webový návrhář, kterému Bob zaplatil, může mít 2-ze-3 vícepodpisovou adresu pro jeho podnikání, která zajistí, že žádné finanční prostředky nebudou utraceny aniž by alespoň dva ze tří obchodních partnerů podepsali transakci.

Prozkoumáme jak vytvořit transakce, které utrácí finanční prostředky z P2SH (a vícepodpisové) adresy v [transactions].

Okrasné adresy

Okrasné adresy jsou platné bitcoinové adresy, které obsahují lidsky čitelné zprávy. Například, 1LoveBPzzD72PUXLzCkYAtGFYmK5vYNR33 je platná adresa, která obsahuje písmena tvořící slovo "Love" jako první čtyři Base-58 písmena. Okrasné adresy požadují tvoření a testování miliard kandidátských soukromých klíčů, dokud jeden z nich neodvodí bitcoinovou adresu s požadovaným vzorem. Přestože existují některé optimalizace v algoritmu tvorby okrasných adres, postup v podstatě zahrnuje výběr náhodného soukromého klíče, odvození veřejného klíče, odvození bitcoinové adresy a kontrola, zda odpovídá požadovanému okrasnému vzoru, opakováno miliardkrát dokud není nalezena shoda.

Jakmile je okrasná adresa odpovídající požadovanému vzoru nalezena, soukromý klíč, ze kterého byla odvozena může být použít majitelem pro utraceni bitcoinů přesně tím samým způsobem jako u jiných adres. Jsou závislé a stejné kryptografii eliptických křivek (ECC) a Secure Hash Algorithm (SHA) jako mnoho jiných adres. Nemůžete snadněji najít soukromé klíče adres začínajících okrasným vzorem než můžete najít soukromé klíče ostatních adres.

V [ch01_intro_what_is_bitcoin], jsme představili Eugenii, ředitelku dětské charity působící na Filipínách. Řekněme, že Eugenia organizuje bitcoinovou finanční sbírku a chce použít okrasnou bitcoinovou adresu pro zvýšení publicity. Eugenia vytvoří okrasnou adresu, která začíná na "1Kids" pro propagaci dětská charitativní sbírky. Podíváme se jak okrasná adresa bude vytvořena a co to znamená pro bezpečnost Eugeniny charity.

Tvorba okrasných adres

Je důležité si uvědomit, že bitcoinová adresa je jednoduše číslo reprezentované znaky v abecedě Base58. Na hledání vzoru "1Kids" může být nahlíženo jako na hledání adresy z intervalu od 1Kids11111111111111111111111111111 do 1Kidszzzzzzzzzzzzzzzzzzzzzzzzzzzzz. V tomto intervalu je přibližně 58^{29} adres (přibližně $1{,}4*10^{51}$), všechny začínající s "1Kids". Rozsah okrasných adres začínajících na "1Kids" ukazuje interval adres, které mají předponu 1Kids.

Table 11. Rozsah okrasných adres začínajících na "1Kids"

Od	1Kids11111111111111111111111111111
	1Kids11111111111111111111111111112
	1Kids11111111111111111111111111113
	...
do	1Kidszzzzzzzzzzzzzzzzzzzzzzzzzzzzz

Podívejme se na vzor "1Kids" jako na číslo a jak často můžeme najít tento vzor v bitcoinových adresách (viz Frekvence okrasných vzorů (1KidsCharity) a průměrný čas nalezení na stolním počítači). Průměrný stolní počítač bez specializovaného hardware může vyhledat přibližně 100,000 klíčů za vteřinu.

Table 12. Frekvence okrasných vzorů (1KidsCharity) a průměrný čas nalezení na stolním počítači

Délka	Vzor	Frekvence	Průměrný čas nalezení
1	1K	1 z 58 klíčů	< 1 millisekunda
2	1Ki	1 z 3 364	50 milisekund
3	1Kid	1 z 195 000	< 2 sekundy
4	1Kids	1 z 11 millionů	1 minuta
5	1KidsC	1 z 656 millionů	1 hodina
6	1KidsCh	1 z 38 miliard	2 dny
7	1KidsCha	1 z 2,2 bilionů	3–4 měsíce
8	1KidsChar	1 z 128 bilionů	13–18 let
9	1KidsChari	1 z 7 biliard	800 let
10	1KidsCharit	1 z 400 biliard	46 000 let
11	1KidsCharity	1 z 23 trilionů	2,5 millionů let

Jak můžete vidět, Eugenia nemůže vytvořit okrasnou adresu "1KidsCharity" v prakticky realizovatelném čase, ani kdyby měla přístup k několika tisícům počítačů. Každý další znak zvyšuje faktor obtížnosti 58-krát. Vzory s více než sedmi znaky jsou obvykle hledány specializovaným hardware, jako uživatelsky sestavené stolní počítače s více multiple grafickými kartami (GPU). jedná se často o změnu účelu bývalých těžebních sestav, které nejsou dále výdělečné při těžbě bitcoinu, ale můžou být použity pro hledání okrasných adres. Hledání okrasných adres na GPU systémech je o mnoho řádů rychlejší než na nespecializovaných CPU.

Další cesta k najití okrasných adres je najmout si těžební skupinu hledačů okrasných adres na Vanity Pool. Tato těžební skupina je služba, která umožňuje vlastníkům GPU hardware si vydělat bitcoiny hledáním okrasných adres pro ostatní. Za malý poplatek (0,01 bitcoinu nebo přibližně 5 amerických dolarů v čase tohoto psaní), Eugenia může zadat skupině hledání sedmiznakového vzoru okrasné

adresy a získat výsledky v řádu pár hodin místo nutnosti hledat na počítači několik měsíců.

Tvorba okrasných adres se provádí hrubou silou: vyzkouší se náhodný klíč, ověří se výsledná adresa, zda odpovídá požadovanému vzoru, opakuje se dokud není úspěch. Těžař okrasných adres ukazuje příklad "okrasného těžaře", program vytvořený k hledání okrasných adres, napsaný v C++. Příklad používá knihovnu libbitcoin, kterou jsme představili v [alt_libraries].

Example 8. Těžař okrasných adres

```
#include <bitcoin/bitcoin.hpp>

// The string we are searching for
const std::string search = "1kid";

// Generate a random secret key. A random 32 bytes.
bc::ec_secret random_secret(std::default_random_engine& engine);
// Extract the Bitcoin address from an EC secret.
std::string bitcoin_address(const bc::ec_secret& secret);
// Case insensitive comparison with the search string.
bool match_found(const std::string& address);

int main()
{
    // random_device on Linux uses "/dev/urandom"
    // CAUTION: Depending on implementation this RNG may not be secure enough!
    // Do not use vanity keys generated by this example in production
    std::random_device random;
    std::default_random_engine engine(random());

    // Loop continuously...
    while (true)
    {
        // Generate a random secret.
        bc::ec_secret secret = random_secret(engine);
        // Get the address.
        std::string address = bitcoin_address(secret);
        // Does it match our search string? (1kid)
        if (match_found(address))
        {
            // Success!
            std::cout << "Found vanity address! " << address << std::endl;
            std::cout << "Secret: " << bc::encode_hex(secret) << std::endl;
            return 0;
        }
    }
    // Should never reach here!
    return 0;
```

```
}

bc::ec_secret random_secret(std::default_random_engine& engine)
{
    // Create new secret...
    bc::ec_secret secret;
    // Iterate through every byte setting a random value...
    for (uint8_t& byte: secret)
        byte = engine() % std::numeric_limits<uint8_t>::max();
    // Return result.
    return secret;
}

std::string bitcoin_address(const bc::ec_secret& secret)
{
    // Convert secret to pubkey...
    bc::ec_point pubkey = bc::secret_to_public_key(secret);
    // Finally create address.
    bc::payment_address payaddr;
    bc::set_public_key(payaddr, pubkey);
    // Return encoded form.
    return payaddr.encoded();
}

bool match_found(const std::string& address)
{
    auto addr_it = address.begin();
    // Loop through the search string comparing it to the lower case
    // character of the supplied address.
    for (auto it = search.begin(); it != search.end(); ++it, ++addr_it)
        if (*it != std::tolower(*addr_it))
            return false;
    // Reached end of search string, so address matches.
    return true;
}
```

NOTE Příklad nahoře používá std::random_device. V závislosti na implementaci by měl odrážet kryptograficky bezpečný generátor náhodných čísel (CSRNG) poskytnutý operačním systémem. V případě UNIX-ových operačních systémů jako Linux, čerpá z /dev/urandom. Zde použitý generátor náhodných čísel je vhodný pouze pro demonstrační účely, *není* vhodný pro vytvoření bitcoinových klíčů pro skutečné použití, protože tyto klíče nejsou implementovány s dostatečným zabezpečením.

Zdrojové kódy v příkladu musejí být zkompilovány C kompilátorem a spojeny s knihovnou libbitcoin (která musí být instalována v systému). Pro spuštění příkladu, spustíme přeložený spustitelný soubor

vanity-miner++ bez parametrů (viz kompilace a spuštění příkladu Těžař okrasných adres) a pokusíme se najít okrasnou adresu začínající "1kid".

Example 9. kompilace a spuštění příkladu Těžař okrasných adres

```
$ # Kompilace zdrojových kódů s g++
$ g++ -o vanity-miner vanity-miner.cpp $(pkg-config --cflags --libs libbitcoin)
$ # Spustíme příklad
$ ./vanity-miner
Found vanity address! 1KiDzkG4MxmovZryZRj8tK81oQRhbZ46YT
Secret: 57cc268a05f83a23ac9d930bc8565bac4e277055f4794cbd1a39e5e71c038f3f
$ # Spustíme příklad znovu a získáme jiný výsledek
$ ./vanity-miner
Found vanity address! 1Kidxr3wsmMzzouwXibKfwTYs5Pau8TUFn
Secret: 7f65bbbbe6d8caae74a0c6a0d2d7b5c6663d71b60337299a1a2cf34c04b2a623
# Použijeme příkaz "time", abychom zjistili, jak dlouho trvalo nalézt výsledek
$ time ./vanity-miner
Found vanity address! 1KidPWhKgGRQWD5PP5TAnGfDyfWp5yceXM
Secret: 2a802e7a53d8aa237cd059377b616d2bfcfa4b0140bc85fa008f2d3d4b225349

real  0m8.868s
user  0m8.828s
sys  0m0.035s
```

Příkladu zdrojových kódů potrvá několik sekund k nalezení tříznakového vzoru "kid", jak můžeme vidět, když použijeme Unixový příkaz time k změření doby běhu. Změňte search vzor ve zdrojovém kódu, a sledujte jak moc déle to potrvá pro čtyř- nebo pěti- znakové vzory!

Bezpečnost okrasných adres

Okrasné adresy mohou být použity ke zvýšení *ale i* zdolání bezpečnostních opatření; jsou skutečným dvouhranným mečem. Při použití pro zvýšení bezpečnosti výrazná adresa znesnadňuje útočníkovi její nahrazení jeho vlastní adresou a napálení zákazníku platících jemu místo vám. Naneštěstí, okrasná adresa také umožňuje komukoliv vytvořit adresu, která *se podobá* jakékoliv náhodné adrese, nebo dokonce jiné okrasné adrese, čímž napálí vaše zákazníky.

Eugenia může inzerovat náhodně vytvořenou adresu (např. 1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy), na kterou mohou lidé posílat jejich příspěvky. Nebo může vytvořit okrasnou adresu, která začíná s 1Kids, aby jí zvýraznila.

V obou případech, jedno z rizik použití jedné pevné adresy (místo oddělených dynamických adres různých pro každého dárce) je, že zloděj může být schopný infiltrovat vaší webovou stránku a nahradit vaší adresu jeho vlastní adresou a tím přesměrovat příspěvky na něj. Pokud máte reklamu na vaší dárcovskou adresu na různých místech, vaši uživatelé by měli visuálně zkontrolovat adresu před provedením platby, že je stejná jakou vidí na vašem webu nebo v emailu od vás a na vašem letáku. V

případě náhodné adresy jako 1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy, průměrný uživatel bude pravděpodobně zkoumat prvních pár znaků "1J7mdg" a bude spokojený, pokud se adresy shodují. Použitím generátoru okrasných adres, někdo s úmyslem krást náhradou podobně vypadající adresy může rychle vytvořit adresy, které se shodují v prvních několika znacích, jak ukazuje [table_4-13].

1. Tvorba okrasné adresy shodné s náhodnou adresou

Původní náhodná adresa	1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy
Okrasná (4 znaky shody)	1J7md1QqU4LpctBetHS2ZoyLV5d6dShhEy
Okrasná (5 znaků shody)	1J7mdgYqyNd4ya3UEcq31Q7sqRMXw2XZ6n
Okrasná (6 znaků shody)	1J7mdg5WxGENmwyJP9xuGhG5KRzu99BBCX

Zvyšuje okrasná adresa bezpečnost? Pokud Eugenia vytvořila okrasnou adresu 1Kids33q44erFfpeXrmDSz7zEqG2FesZEN, uživatelé se budou spíše dívat na okrasný vzorek a na pár znaků za, například všimnou si části adresy "1Kids33". To může donutit útočníka vytvořit okrasnou peněženku odpovídající alespoň šesti znakům (dvěma navíc), prodražující jeho úsilí 3 364 krát (58 × 58) více než úsilí Eugenie vynaložené za její čtyřznakovou okrasu. V podstatě úsilí Eugenie vynaložené (nebo zaplacené těžební skupině okras) "nutí" útočníka vyrobit delší okrasný vzorek. Pokud Eugenia zaplatí těžební skupině za 8-znakovou okrasnou adresu, útočník by měl být tlačen do světa 10 znakových, které jsou nedosažitelné na osobním počítači a jsou drahé dokonce s uživatelským okrasnou těžební sestavou nebo okrasnou těžební skupinou. Co je cenově dostupné pro Eugenii se stává cenově nedostupným pro útočníka, zvláště pokud potenciální odměna zpronevěry není dostatečně vysoká na pokrytí vytvoření okrasné adresy..

Papírové peněženky

Papírové peněženky jsou bitcoinové soukromé klíče vytištěné na papíru. Často papírové peněženky také obsahují odpovídající bitcoinovou adresu pro usnadnění, ale to není nutné, protože může být odvozena ze soukromého klíče. Papírové peněženky jsou velmi efektivním způsobem vytvoření záloh nebo offline bitcoinových úložišť, také známých jako "studené úložiště." Jako zálohovací mechanizmus, papírová peněženka může nabízet zabezpečení proti ztrátě klíče v důsledku nehody počítače, jako je třeba selhání pevného disku, krádež nebo náhodné smazání. Jako mechanizmus "studeného úložiště", pokud klíče papírové peněženky byly vytvořeny offline a nikdy nebyly uloženy v počítačovém systému, jsou výrazně bezpečnější před hackery, key-logery a dalšímu online počítačovými hrozbami.

Papírové peněženky se nabízí v mnoha tvarech, velikostech a vzhledech, ale na nejzákladnější úrovni je to jen klíč a adresa vytištěná na papíře. Nejjednodušší podoba papírové peněženky - vytištěná bitcoinová adresa a soukromý klíč ukazuje nejednoduší podobu papírové peněženky.

Table 13. Nejjednodušší podoba papírové peněženky - vytištěná bitcoinová adresa a soukromý klíč

Veřejná adresa	**Soukromý klíč (WIF)**
1424C2F4bC9JidNjjTUZCbUxv6Sa1Mt62x	5J3mBbAH58CpQ3Y5RNJpUKPE62SQ5tfcvU2JpbnkeyhfsYB1Jcn

papírové peněženky mohou být snadno vytvořeny nástroji jako klientský JavaScript generátor na *bitaddress.org*. Tato stránka obsahuje zdrojový kód nezbytný pro vytvoření klíčů a papírových peněženek, dokonce při úplném odpojení od internetu. Pro jejich použití uložte HTML stránku na vašem místním disku nebo na externím USB flash disku. Odpojte počítač od internetu a otevřete soubor v prohlížeči. Ještě lépe, spusťte počítač za pomocí čerstvé instalace operačního systému, jako Linux OS spustitelný z CD-ROMu. Jakýkoliv klíč vytvořený tímto nástrojem offline může být vytisknut na místní tiskárně přes USB kabel (ne bezdrátové), tím je vytvořena papírová peněženka, jejíž klíče existují pouze na papíru a nikdy nebyly uložena na žádném online systému. Uložte tyto papírové peněženky do ohnivzdorného trezoru a odešlete bitcoiny na jejich bitcoinové adresy, čímž vytvoříte jednoduché a přesto velmi efektivní řešení "studeného úložiště". Příklad jednoduché papírové peněženky z bitaddress.org ukazuje papírovou peněženku vytvořenou z webu bitaddress.org.

Figure 14. Příklad jednoduché papírové peněženky z bitaddress.org

Nevýhoda jednoduchého systému papírové peněženky je, že klíče jsou zranitelné fyzickou krádeží. Zloděj, který je schopen získat přístup k papíru, jej může snadno ukrást nebo vyfotografovat klíče a převzít kontrolu nad bitcoiny zamčené těmito klíči. Sofistikovanější úložní systém papírové peněženky je chráněn heslovou frází, kterou se vlastník naučí zpaměti. Bez heslové fráze, jsou zašifrované klíče nepoužitelné. Nejlepší jsou heslovou frází chráněné peněženky, jejíchž klíče nebyly nikdy online a musejí být fyzicky vyzvednuty z trezoru nebo jiného fyzicky zabezpečeného úložiště. Příklad zašifrované papírové peněženky z bitaddress.org. Heslová fráze je "test" ukazují papírovou peněženku se zašifrovaným soukromým klíčem (BIP0038) vytvořeným na webu bitaddress.org

Figure 15. Příklad zašifrované papírové peněženky z bitaddress.org. Heslová fráze je "test"

WARNING

Přestože můžete vkládat finanční prostředky do papírové peněženky několikrát, měli byste vyzvednout všechny prostředky najednou, utratit všechno. To je důvod proč při procesu odemčení a utracení finančních prostředků některé peněženky můžou vytvářet adresy vratek, pokud je útrata menší než celá částka. Navíc, pokud počítač, který používáte k podpisu transakce je napaden, riskujete prozrazení soukromého klíče. Utracením celé částky z papírové peněženky pouze jednou, snižujete riziko prozrazení klíče. Pokud potřebujete pouze malé množství, zašlete zbývající prostředky na novou papírovou peněženku v této samé transakci.

Papírové peněženky jsou dostupné v mnoha verzích a velikostech s mnoha různými funkcemi. Některé jsou zamýšlené, aby byly dány jako dárky a mají sezónní tématiku jako Vánoce a Nový Rok. Jiné jsou navrženy pro uskladnění v bankovním trezoru se soukromým klíčem skrytém nějakým způsobem, buď neprůhlednou seškrabatelnou samolepkou nebo přeložená a zapečetěné s nepadělatelnou lepící fólií. Obrázky <xref linkend="paper_wallet_bpw" xrefstyle="select: labelnumber"/> přes <xref linkend="paper_wallet_spw" xrefstyle="select: labelnumber"/> ukazují různé příklady papírových peněženek s bezpečnostními a zálohovacími funkcemi.

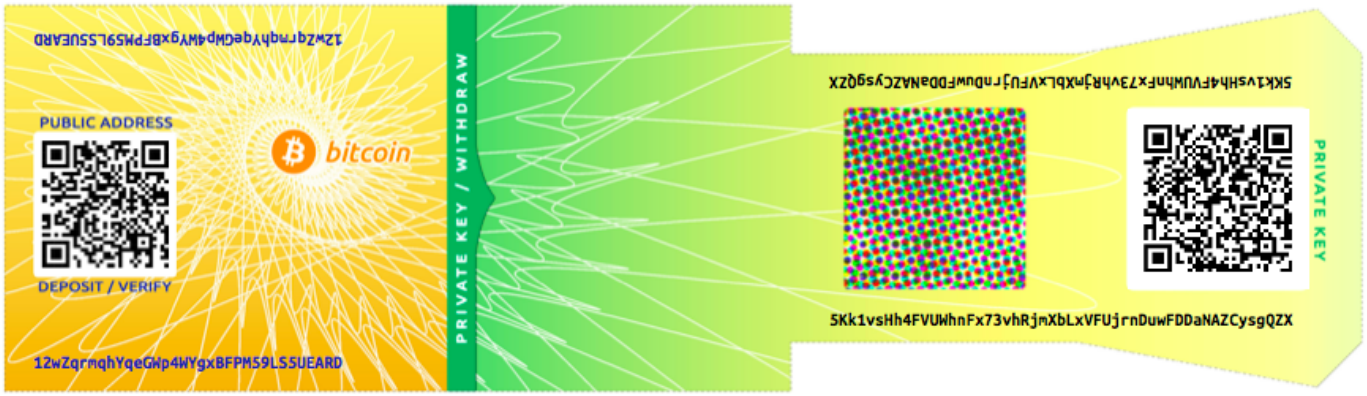


Figure 16. Příklad papírové peněženky z bitcoinpaperwallet.com se soukromým klíčem na skládací klapce.

Figure 17. Papírová peněženka bitcoinpaperwallet.com se skrytým soukromým klíčem

Jiné konstrukce nabízejí dodatečné kopie klíče nebo adresy ve formě oddělitelných ústřižků, podobných ústřižkům lístků, umožňující uchovávat více kopií chráněných proti ohni, potopě nebo jiným přírodním pohromám.

Figure 18. Příklad papírové peněženky s dodatečnými kopiemi klíčů na záložním "ústřižku"

Transakce

Úvod

Transakce jsou nejdůležitější částí bitcoinového systému. Cokoliv jiného v bitcoinu je navrženo k zajištění, že transakce mohou být vytvářeny, rozšířeny po síti, ověřeny a nakonec přidány do celosvětového účetního systému transakcí (blockchain). Transakce jsou datové struktury, které kódují přenos hodnoty mezi účastníky bitcoinového systému. Každá transakce je veřejným záznamem v bitcoinovém blockchainu, celosvětové knize podvojného účetnictví.

V této kapitole prozkoumáme všechny možné typy transakcí, co mohou obsahovat, jak je vytvořit, jak jsou ověřeny a jak se stávají částí trvalého záznamu všech transakcí.

Životní cyklus transakce

Životní cyklus transakce začíná jejím vytvořením, známým jako *zrození*. Transakce je podepsána jedním nebo více podpisy udávajícími oprávnění utratit finanční prostředky, na které transakce odkazuje. Transakce je poté vysílána po bitcoinové síti, kde každý uzel (účastník) ověřuje a šíří transakci dokud nedosáhne (téměř) všech uzlů sítě. Nakonec, transakce je ověřena těžebním uzlem a vložena do bloku transakcí, který je zaznamenán v blockchainu.

Jakmile je zaznamenán na blockchainu a potvrzen dostatečným počtem následujících bloků (potvrzení), transakce se stává trvalou částí bitcoinového účetního systému a je přijímána jako platná všemi účastníky. Finanční prostředky přidělené novému vlastníkovi transakce mohou být utraceny v nové transakci, rozšiřující řetěz vlastnictví a začínající znova životní cyklus transakce.

Tvorba transakcí

V některých směrech pomáhá přemýšlet o transakci jako o papírovém šeku. Jako šek, transakce je nástroj, který vyjadřuje záměr převést peníze a není viditelný finančnímu systému dokud není podán k provedení. Jako šek, tvůrce transakce nemusí tím, kdo transakci podepisuje.

Transakce mohou být vytvořeny online nebo offline kýmkoliv, dokonce i když osoba tvořící transakci nemá podpisové oprávnění k účtu. Například, úředník starající se o splatné faktury může vytvořit šek, který podepíše výkonný ředitel. Podobně tento úředník může vytvořit bitcoinovou transakci, kterou opatří výkonný ředitel svým elektronickým podpisem, aby ji učinil platnou. Zatímco šek odkazuje na konkrétní částku ve zdrojové měně, bitcoinová transakce odkazuje na konkrétní předchozí transakci a její zdroje, spíše než na účet.

Jakmile byla transakce vytvořena, je podepsána vlastníkem (nebo vlastníky) finančních prostředků. Pokud má správný tvar a je podepsána, podepsaná transakce je nyní platnou a obsahuje všechny informace nutné k provedení převodu finančních prostředků. Nakonec platná transakce se musí dostat do bitcoinové sítě, tedy musí být šířena dokud se nedostane k těžaři, který ji zahrne do veřejné účetní knihy (blockchainu).

Šíření transakcí v bitcoinové síti

Nejprve, transakce musí být doručena do bitcoinové sítě, aby mohla být šířena a vložena do blockchainu. V zásadě platí, bitcoinová transakce je jen 300 až 400 bytů dat, která se musí dostat k jednomu z desítek tisíc bitcoinových uzlů. Odesilatelé nemusí důvěřovat uzlům, které šíří transakci, pokud použijí více jak jeden uzel, aby se ujistili, že dojde k jejímu rozšíření. Uzly nepotřebují věřit odesilateli nebo prokazovat "identitu" odesilatele. Protože transakce je podepsána a neobsahuje žádné důvěrné informace, soukromé klíče nebo pověření, může být veřejně šířena za použití podkladové sítě, která ji pohodlně přenese. Na rozdíl od transakcí kreditních karet, například, které obsahují citlivé informace a mohou být přenášeny jen po zašifrované síti, bitcoinová transakce může být přenášena po jakékoliv síti. Jakmile je transakce schopná dosáhnout prvního bitcoinového uzlu, který ji rozšíří do bitcoinové sítě, nezáleží jak byla přenesena k tomuto prvnímu uzlu.

Proto mohou být bitcoinové transakce přenášeny do bitcoinové sítě přes nezabezpečené sítě jako WiFi, Bluetooth, NFC, Chirp,čárové kódy nebo kopírováním a vložením do webového formuláře. V extrémních příkladě paketovým rádiem, satelitním přenosem, krátkovlnně pomocí přerušovaného přenosu, rozšířeného spektra, přepínání frekvencí, aby se vyhnuly odhalení a zablokování. Bitcoinové transakce mohou být kódovány dokonce smajlíky a poslány na veřejné fórum nebo poslány jako textové zpráva přes Skype. Bitcoin proměnil peníze na datovou strukturu, takže je takřka nemožné zastavit kohokoliv ve vytváření a vykonávání bitcoinových transakcí.

Propagace transakcí v bitcoinové síti

Jakmile je bitcoinová transakce poslána nějakému uzlu připojenému do bitcoinové sítě, transakce bude ověřena uzlem. Pokud je platná, uzel ji rozšíří do ostatních uzlů, se kterými je spojen a zpráva o úspěchu bude synchronně vrácena zasilateli transakce. Pokud je transakce neplatná, uzel ji odmítne a synchronně vrátí zasilateli zprávu o odmítnutí.

Bitcoinová síť je peer-to-peer síť, každý bitcoinový uzel je spojen s několika dalšími uzly (sousedy), které objevil při svém spuštění pomocí peer-to-peer protokolu. Celá síť tvoří volně spojenou síťovinu bez pevné topologie nebo nějaké struktury, všechny uzly jsou stejného typu, jsou si rovny. Zprávy, obsahující transakce a bloky, jsou šířeny z každého uzlu do všech jeho sousedů, se kterými je propojen pomocí postupu zvaného "záplava". Nově zkontrolovaná transakce vložená do uzlu na síti bude zaslána všem jeho sousedům, každý z nich ji zašle všem svým sousedům atd. Tímto způsobem během pár sekund je platná transakce šířena v exponenciálně se zvětšující vlně skrz síť dokud ji neobdrží všechny uzly v síti.

Bitcoinová síť je navržena na rozšiřování transakcí a bloků všem uzlům účinným a pružným způsobem, který je odolný vůči útokům. Jako prevence spamu, útoku odepření služby nebo jiného zátěžového útoku proti bitcoinové síti, každý uzel nezávisle ověřuje každou transakci před jejím dalším šířením. Pravidla ověřování transakcí jsou podrobněji popsána v [tx_verification].

Struktura transakce

Transakce je *datová struktura*, která kóduje přenos hodnoty ze zdroje finančních prostředků zvaného

vstup do cíle zvaného *výstup*. Transakční vstupy a výstupy se nevztahují k účtům nebo identitám. Místo toho, byste měli o nich přemýšlet jako množství bitcoinů - kouscích bitcoinu - zamčeném určitým tajemstvím známým pouze vlastníkovi. Osoba, která zná tajemství, může odemykat. Transakce obsahuje několik položek, jak ukazuje Struktura transakce.

Table 1. Struktura transakce

Velikost	Pole	Popis
4 byty	Version	Určuje jakými pravidly se transakce řídí
1–9 bytů (VarInt)	Input Counter	Kolik vstupů bylo vloženo
Proměnlivá	Inputs	Jeden nebo více vstupů transakce
1–9 bytů (VarInt)	Output Counter	kolik vstupů je vloženo
Proměnlivá	Outputs	Jeden nebo více výstupů transakce
4 byty	Locktime	Unixová časová známka nebo číslo bloku

Časový zámek transakce

Časové zámek transakce, známý jako nLockTime podle jména proměnné použité v referenčním klientovi, definuje okamžik, od kterého je transakce platná a může být předána síti nebo přidána do blockchainu. Ve většině transakcí je nastaven na nula, což znamená, že transakce může být šířena a vykonána okamžitě. Pokud je časový zámek nenulový ale nižší než 500 milionů je interpretován jako výška bloku, což znamená, že transakce je neplatná a nemůže být šířena nebo vožena do bloku před blokem určité výšky. Pokud je vyšší než 500 milionů, je interpretován jako Unixová časová značka (počet sekund od 1. 1. 1970) a transakce není platná před tímto určeným časem. Transakce s časovým zámkem specifikujícím budoucí blok nebo čas musí být drženy v systémech jejich stvořitelů a do bitcoinové sítě smějí být šířeny až po nastání jejich platnosti. Použití časového zámku odpovídá papírovým šekům s odloženým datem platby.

Transakční výstupy a vstupy

Základní stavební blok bitcoinové transakce je *neutracený výstup transakce* neboli UTXO. UTXO jsou samostatné bloky bitcoinové měny zamčené ve prospěch určitého vlastníka, zaznamenány na blockchainu a rozpoznávány jako jednotky měny celou sítí. Bitcoinová síť sleduje všechny dostupné (neutracené) UTXO, v současné době jich jsou miliony. Kdykoliv uživatel obdrží bitcoiny, jejich množství je zaznamenáno do blockchainu jako UTXO. Tedy, bitcoiny uživatele mohou být roztříštěny jako UTXO mezi stovky transakcí a stovky bloků. Ve skutečnosti neexistuje nic jako stav účtu nebo bitcoinové adresy, existují pouze roztříštěné UTXO zamčené ve prospěch určitého majitele. Pojem stav

bitcoinového účtu je vytvořen peněženkou aplikaci jako odvozená statistika. Peněženka počítá stav uživatelova účtu prohledáváním blockchainu a sečteném hodnot všech UTXO patřících uživateli.

TIP	V bitcoinu neexistují stavy účtů, existují pouze *neutracené zůstatky transakcí* (UTXO) roztříštěné po blockchainu.

UTXO může mít libovolnou hodnotu vyjádřenou v jednotkách f satosi. Podobně jako dolary můžou být rozděleny na dvě desetinná místa jako centy, bitcoiny mohou být rozděleny na osm desetinných míst jako satoshi. Přestože UTXO může mít jakoukoliv libovolnou hodnotu, jakmile je vytvořen, je nedělitelný podobně jako mince, která nemůže být rozdělena na poloviny. Pokud UTXO je větší než požadovaná hodnota transakce, musí být spotřebován celý a v transakci musí být vytvořena vratka. Jinými slovy, pokud máte 20 bitcoinový UTXO a chcete zaplatit 1 bitcoin, vaše transakce musí spotřebovat všech 20 bitcoinů v UTXO a vyrobit dva výstupy: jeden platí 1 bitcoin požadovanému příjemci a druhá platí 19 bitcoinu vratky zpátky vaší peněžence. Ve výsledku, většina bitcoinových transakcí tvoří vratku.

Představte si zákazníka kupujícího nápoj za 1,50 dolarů, sahajícího do své peněženky a snažícího se najít správnou kombinací mincí a bankovek, aby zaplatil částku 1,50 dolarů. Zákazník vybere přesnou částku, pokud je to možné (dolar a dva čtvrťáky), nebo kombinaci menších mincí (šest čtvrťáků), nebo pokud je nezbytné větší jednotku jako je pětidolarová bankovka. pokud předá prodavači příliš mnoho peněz, řekněme 5 dolarů, prodavač vrátí nazpět 3,50 dolarů, které se vrátí do peněženky zákazníka a jsou dostupné pro budoucí transakce.

Podobně v bitcoinové transakci musí být utracen celý uživatelův UTXO bez ohledu na částku, kterou má uživatel k dispozici. Uživatelé nemůžou rozdělit UTXO na poloviny, stejně jako nemohou rozpůlit dolarovou bankovku a použít ji jako platidlo. Peněženková aplikace uživatele obvykle vybere z uživatelových dostupných UTXO různé jednotky, aby složila částku větší nebo rovnou požadovanému částce transakce.

Jako ve skutečném životě, bitcoinová aplikace může používat několik strategií k zajištění částky platby: kombinování několika menších jednotek, hledání přesné částky nebo použít větší jednotku než je hodnota transakce a vytvoření vratky. Všechna tato složitá skládání utratitelných UTXO dělá peněženka uživatele automaticky a neviditelně pro uživatele. Musíte se tímto zabývat pouze, pokud pomocí programu tvoříte syrové transakce z UTXO.

UTXO spotřebované transakcemi jsou nazývány transakční vstupy UTXO vytvořené transakcemi jsou nazývány výstupy. Tímto způsobem kousky bitcoinové hodnoty se přesouvají od majitele k majiteli v řetězu transakcí konzumujícím a vytvářejícím UTXO. Transakce spotřebovávají UTXO jejich odemčením pomocí podpisu současného majitele a vytvářejí UTXO jejich zamčením ve prospěch bitcoinové adresy nového vlastníka.

Výjimkou z tohoto řetězu výstupu a vstupů tvoří speciální typ transakce nazvaná *mincetvorná* transakce, která je první v každém bloku. Tato transakce je zde umístěna "vítězným" těžařem a vytváří zbrusu nové bitcoiny splatné těžaři, jako odměna za těžbu. Takto je tvořena bitcoinová peněžní zásoba během těžebního procesu, jak uvidíme v [ch8].

TIP	Co bylo dřív? Vstupy nebo výstupy, slepice nebo vejce? Striktně řečeno, výstupy byly první, protože mincetvorné transakce, které vytvářejí nové bitcoiny, nemají vstupy a vytváří výstupy z ničeho.

Výstupy transakcí

"bitcoin ledger, outputs in", id="ix_ch05-asciidoc2", range="startofrange")Každá bitcoinová transakce tvoří výstupy, které jsou zaznamenány v bitcoinovém účetním systému. Téměř všechny z těchto výstupů s jednou výjimkou (viz Datový výstup (OP_RETURN)) vytváří utratitelné kousky bitcoinu zvané *neutracené výstupy transakcí* nebo UTXO, které jsou rozpoznávány celou sítí a jsou dostupné vlastníkovi pro utracení v budoucích transakcích. Zaslání bitcoinů někomu je vytvoření neutraceného transakčního výstupu (UTXO) a zaregistrování ho ve prospěch jeho adresy, čímž jsou pro něj dostupné k utracení.

UTXO jsou sledovány každým úplným bitcoinovým klientem jako datová množina zvaná *UTXO množina* nebo *UTXO skupina*, držena v databázi. Nové transakce spotřebovávají (utrácejí) jeden nebo více výstupů z této množiny UTXO.

Transakční výstupy se skládají ze dvou částí:

- Množství bitcoinů, vyjádřené v *satoshi*, nejmenší bitcoinové jednotce
- *zamykací skript*, také známý jako "břemeno", které "zamyká" toto množství specifikací podmínek, které musejí být splněny pro utracení tohoto výstupu

Transakční skriptovací jazyk, použitý v zamykacím skriptu, právě zmíněném, je podrobněji popsán v Transakční skripty a skriptovací jazyk. Struktura transakčního výstupu zobrazuje strukturu transakčního výstupu.

Table 2. Struktura transakčního výstupu

Velikost	Pole	Popis
8 bytů	Amount	Bitcoinová hodnota v satoshi (10^{-8} bitcoin)
1-9 bytů (VarInt)	Locking-Script Size	Délka (v byte) zamykacího skriptu, který následuje
Variable	Locking-Script	Skript definující podmínky nutné k utracení výstupu

V Skript volající blockchain.info API k nalezení UTXO vztahujících se k adrese použijeme blockchain.info API pro najití neutracených výstupů (UTXO) zadané adresy.

Example 1. Skript volající blockchain.info API k nalezení UTXO vztahujících se k adrese

```
# get unspent outputs from blockchain API

import json
import requests

# example address
address = '1Dorian4RoXcnBv9hnQ4Y2C1an6NJ4UrjX'

# The API URL is https://blockchain.info/unspent?active=<address>
# It returns a JSON object with a list "unspent_outputs", containing UTXO, like this:
#{  "unspent_outputs":[
#   {
#     "tx_hash":"ebadfaa92f1fd29e2fe296eda702c48bd11ffd52313e986e99ddad9084062167",
#     "tx_index":51919767,
#     "tx_output_n": 1,
#     "script":"76a9148c7e252f8d64b0b6e313985915110fcfefcf4a2d88ac",
#     "value": 8000000,
#     "value_hex": "7a1200",
#     "confirmations":28691
#   },
# ...
#]}

resp = requests.get('https://blockchain.info/unspent?active=%s' % address)
utxo_set = json.loads(resp.text)["unspent_outputs"]

for utxo in utxo_set:
    print "%s:%d - %ld Satoshis" % (utxo['tx_hash'], utxo['tx_output_n'],
utxo['value'])
```

Běžící skript, vidíme seznam transakčních ID, dvojtečka indexové číslo daného neutraceného transakčního výstupu(UTXO), a hodnota UTXO v satoshi. Zamykací skript není zobrazen ve výstupu v Běžící skript get-utxo.py.

Example 2. Běžící skript get-utxo.py

```
$ python get-utxo.py
ebadfaa92f1fd29e2fe296eda702c48bd11ffd52313e986e99ddad9084062167:1 - 8000000 Satoshis
6596fd070679de96e405d52b51b8e1d644029108ec4cbfe451454486796a1ecf:0 - 16050000
Satoshis
74d788804e2aae10891d72753d1520da1206e6f4f20481cc1555b7f2cb44aca0:0 - 5000000 Satoshis
b2affea89ff82557c60d635a2a3137b8f88f12ecec85082f7d0a1f82ee203ac4:0 - 10000000
Satoshis
...
```

Podmínky utracení (břemena)

Výstupy transakcí spojují určitou částku (v satoshi) s určitým *břemenem* nebo zamykacím skriptem, který definuje podmínky nutné k utracení této částky. Ve většině případů, zamykací skript zamkne výstup ve prospěch určité bitcoinové adresy, tím přenáší vlastnictví tohoto množství na nového majitele. Když Alice zaplatila Bobově kavárně za šálek kávy, její transakce vytvořila 0,015 bitcoinový výstup *zatížený břemenem* nebo zamčený ve prospěch bitcoinové adresy kavárny. Tento 0,015 bitcoinový výstup bude zaznamenán v blockchainu a stane se částí množiny *neutracených výstupů transakcí*. Bobova peněženka tento výstup zobrazí jako část stavu peněženky - dostupných finančních prostředků. Pokud se Bob rozhodne utratit tuto částku, jeho transakce uvolní břemeno, odemkne výstup poskytnutím odemykacího skriptu obsahujícího podpis vytvořený Bobovým soukromým klíčem.

Transakční vstupy

Jednoduše řečeno, transakční vstupy jsou ukazatele na UTXO. Ukazují na konkrétní UTXO pomocí odkazu na haš transakce a posloupnost čísel, kde je UTXO zaznamenáno v blockchainu. Pro utracení UTXO, transakční vstup také obsahuje odemykací skript, který splňuje podmínky pro utracení nastavené tomuto UTXO. Odemykací skript je obvykle podpis prokazující vlastnictví bitcoinové adresy, která je v zamykacím skriptu.

Když uživatelé udělají platbu, jejich peněženka vytvoří transakci vybráním dostupných UTXO. Například, pro vytvoření 0,015 bitcoinové platby, peněženková aplikace může vybrat 0,01 UTXO a 0,005 UTXO, použije je oba, aby dosáhla částky požadované platbou.

V Skript počítající kolik bitcoinů bude použito pro platbu, vidíme použití "hladového algoritmu" pro výběr dostupného UTXO, tak aby byl vytvořeno konkrétní částka pro platbu. V příkladě, dostupné UTXO jsou poskytnuty jako konstantní pole, ve skutečnosti by se dostupné UTXo získávaly pomocí RPC volání Bitcoin Core, nebo API třetí strany, jak je zobrazeno v Skript volající blockchain.info API k nalezení UTXO vztahujících se k adrese.

Example 3. Skript počítající kolik bitcoinů bude použito pro platbu

```
# Selects outputs from a UTXO list using a greedy algorithm.

from sys import argv

class OutputInfo:

    def __init__(self, tx_hash, tx_index, value):
        self.tx_hash = tx_hash
        self.tx_index = tx_index
        self.value = value

    def __repr__(self):
        return "<%s:%s with %s Satoshis>" % (self.tx_hash, self.tx_index,
                                             self.value)

# Select optimal outputs for a send from unspent outputs list.
# Returns output list and remaining change to be sent to
# a change address.
def select_outputs_greedy(unspent, min_value):
    # Fail if empty.
    if not unspent:
        return None
    # Partition into 2 lists.
    lessers = [utxo for utxo in unspent if utxo.value < min_value]
    greaters = [utxo for utxo in unspent if utxo.value >= min_value]
    key_func = lambda utxo: utxo.value
    if greaters:
        # Not-empty. Find the smallest greater.
        min_greater = min(greaters)
        change = min_greater.value - min_value
        return [min_greater], change
    # Not found in greaters. Try several lessers instead.
    # Rearrange them from biggest to smallest. We want to use the least
    # amount of inputs as possible.
    lessers.sort(key=key_func, reverse=True)
    result = []
    accum = 0
    for utxo in lessers:
        result.append(utxo)
        accum += utxo.value
        if accum >= min_value:
            change = accum - min_value
            return result, "Change: %d Satoshis" % change
    # No results found.
    return None, 0

def main():
    unspent = [
```

```
OutputInfo("ebadfaa92f1fd29e2fe296eda702c48bd11ffd52313e986e99ddad9084062167", 1,
8000000),

OutputInfo("6596fd070679de96e405d52b51b8e1d644029108ec4cbfe451454486796a1ecf", 0,
16050000),

OutputInfo("b2affea89ff82557c60d635a2a3137b8f88f12ecec85082f7d0a1f82ee203ac4", 0,
10000000),

OutputInfo("7dbc497969c7475e45d952c4a872e213fb15d45e5cd3473c386a71a1b0c136a1", 0,
25000000),

OutputInfo("55ea01bd7e9afd3d3ab9790199e777d62a0709cf0725e80a7350fdb22d7b8ec6", 17,
5470541),

OutputInfo("12b6a7934c1df821945ee9ee3b3326d07ca7a65fd6416ea44ce8c3db0c078c64", 0,
10000000),

OutputInfo("7f42eda67921ee92eae5f79bd37c68c9cb859b899ce70dba68c48338857b7818", 0,
16100000),
    ]

    if len(argv) > 1:
        target = long(argv[1])
    else:
        target = 55000000

    print "For transaction amount %d Satoshis (%f bitcoin) use: " % (target,
target/10.0**8)
    print select_outputs_greedy(unspent, target)

if __name__ == "__main__":
    main()
```

Pokud spustíme skript *select-utxo.py* bez parametrů, pokusí se sestavit množinu UTXO (a vratku) pro platbu 55 000 000 satoshi (0,55 bitcoinu). Pokud v parametru uvedeme cílovou částku platby, skript vybere UTXO pro vytvoření cílové částky platby. V Běh skriptu select-utxo.py spustíme skript snažící se vytvořit platbu 0,5 bitcoinu nebo 50 000 000 satoshi.

Example 4. Běh skriptu select-utxo.py

```
$ python select-utxo.py 50000000
For transaction amount 50000000 Satoshis (0.500000 bitcoin) use:
([<7dbc497969c7475e45d952c4a872e213fb15d45e5cd3473c386a71a1b0c136a1:0 with 25000000
Satoshis>, <7f42eda67921ee92eae5f79bd37c68c9cb859b899ce70dba68c48338857b7818:0 with
16100000 Satoshis>,
<6596fd070679de96e405d52b51b8e1d644029108ec4cbfe451454486796a1ecf:0 with 16050000
Satoshis>], 'Change: 7150000 Satoshis')
```

Jakmile jsou UTXO vybrány, peněženka vytvoří odemykací skripty obsahující podpisy pro každý z UTXO, tím je činí utratitelnými splněním podmínek jejich zamykacího skriptu.

Table 3. Struktura transakčních vstupů

Velikost	Pole	Popis
32 bytů	Haš transakce	Ukazatel na transakci obsahující UTXO pro utracení
4 byty	Index výstupu	Indexové číslo UTXO pro utracení, první je 0
1-9 bytů (VarInt)	Velikost odemykacího skriptu	Velikost odemykacího skriptu (v bytech), který následuje
Proměnlivá	Odemykací skript	Skript, který splňuje podmínky UTXO zamykacího skriptu.
4 byty	Číslo posloupnosti	V současnosti neaktivní, funkce náhrady transakcí, nastavena na 0xFFFFFFFF

NOTE Číslo posloupnosti je použito pro přepsání transakce před vypršením časového zámku transakce, tato funkce je v současné době v bitcoinu zablokována. Většina transakcí nastavuje tuto hodnotu na maximální hodnotu celého čísla (0xFFFFFFFF) a je ignorovaná bitcoinovou sítí. Pokud má transakce nenulový časový zámek, alespoň jeden z jejich vstupů musí mít číslo posloupnosti pod 0xFFFFFFFF, aby byl povolen časový zámek.

Transakční poplatky

Mnoho transakcí vkládá transakční poplatky, které jsou odměnou bitcoinovým těžařům za zabezpečování sítě. Těžba a poplatky a odměny sbírané těžaři jsou podrobněji probrány v [ch8]. Tato část zkoumá jak transakční poplatky jsou vkládány do typické transakce. Většina peněženek počítá a vkládá transakční poplatky automaticky. Nicméně, pokud sestavujete transakci programem, nebo za použití rozhraní příkazové řádky, musíte ručně zahrnout tyto poplatky.

Transakční poplatky slouží jako pobídka k zahrnutí (vytěžení) transakce v dalším bloku. Zavedením malé ceny za každou transakci odrazujeme "spamové" transakce nebo jiné druhy obtěžování systému. Transakční poplatky jsou sebrány těžařem, který vytěží blok, který zaznamená transakci na blockchainu.

Transakční poplatky jsou počítány na základě velikosti transakce v kilobytech, nikoliv na základě hodnoty transakce v bitcoinech. Celkově, transakční poplatky jsou založeny na základě tržních sil v bitcoinové síti. Těžaři určují prioritu transakcí na základě mnoha různých kritérií včetně poplatků a mohou dokonce zpracovat transakce zdarma za jistých okolností. Transakční poplatky ovlivňují prioritu zpracování, což znamená, že transakce s dostatečným poplatkem jsou spíše vloženy do dalšího vytěženého bloku, zatímco transakce s nedostatečným nebo žádným poplatkem mohou být zpožděny, zpracovány teprve za několik bloků nebo nezpracovány vůbec. Transakční poplatky nejsou povinné a transakce bez poplatků mohou být nakonec zpracovány, nicméně vložení transakčních poplatků podněcuje k jejich přednostnímu zpracování.

V průběhu času, způsob výpočtu transakčních poplatků a jejich efektu na určení priority transakcí se vyvíjel. Nejdříve byly transakční poplatky v pevné výši v celé síti. Postupně se struktura poplatků uvolňovala, takže mohla být ovlivňována tržními silami, v závislosti ne kapacitě sítě a objemu transakcí. V současné době je minimální výše poplatku nastavena na 0,0001 bitcoinu nebo jedné desetině milibitcoinu za kilobyte, nedávno došlo ke snížení z jednoho milibitcoinu. Většina transakcí je menších než jeden kilobyte; nicméně transakce s mnoha vstupy nebo výstupy mohou být větší. V budoucích verzích bitcoinového protokolu se očekává, že peněženkové aplikace pro výpočet nejvhodnějšího poplatku, který má být k transakci připojen, budou používat statistické analýzy průměrných poplatků předchozích transakcí.

Současný algoritmy používaný těžaři určující prioritu transakcí pro jejich vložení do bloku v závislosti na jejich poplatcích je detailně prozkoumán v [ch8].

Přidání poplatků do transakcí

Datová struktura transakce nemá položku pro poplatek. Místo toho, poplatky jsou implicitní jako rozdíl mezi součtem vstupů a součtem výstupů. Jakákoliv přebývající částka zůstávající poté, co jsou všechny výstupy odečteny od všech vstupů, je poplatkem, který je sebrán těžařem.

Transakční poplatky jsou implicitní, jako přebytek vstupů minus výstupů:

```
Poplatky = Suma(Vstupů) – Suma(Výstupů)
```

Toto je občas matoucím prvkem transakcí, ale je to důležitým bodem pro pochopení, protože pokud sestavujete vaše vlastní transakce, musíte se ujistit, že jste nezahrnuli neadekvátně vysoký poplatek tím, že jste nevyčerpali vstupy. To znamená, že musíte počítat se všemi vstupy, pokud je to nezbytné, tak vytvořit vratku, nebo nakonec těžaře odměníte velmi vysokým spropitným.

Například, pokud spotřebujete 20-ti bitcoinový UTXO pro provedení 1-bitcoinové platby, musíte vložit 19-bitcoinovou vratku zpátky do vaší peněženky. Jinak "přebytečných" 19 bitcoinů bude započteno jako

transakční poplatek a bude sebráno těžařem, který vytěží vaší transakci v bloku. Přestože se vám dostane prioritního zpracování a uděláte těžaře velmi šťastným, pravděpodobně to není to, co jste zamýšleli.

WARNING	Pokud zapomenete přidat výstup pro vratku v ručně sestavované transakci, zaplatíte vratku jako transakční poplatek. "Nechte si drobné!" možná není to, co jste zamýšleli.

Podívejme se jak to funguje v praxi, podíváme se na znova na nákup kávy Alicí. Alice chce utratit 0,015 bitcoinu, aby zaplatila za kávu. Aby se ujistila, že transakce bude ihned zpracována, chce zahrnout transakční poplatek, řekněme 0,001. To znamená, že celková cena transakce bude 0,016. Její peněženka proto musí poskytnout množinu UTXO, která obsahuje hodnotu 0,016 bitcoinu nebo více, a pokud je to nezbytné, vytvořit vratku. Řekněme, že peněženka má dostupný 0,2-bitcoinový UTXO. Bude proto třeba spotřebovat tento UTXO a vytvořit jeden výstup pro Bobovu kavárnu s hodnotou 0,015 a druhý výstup s vratkou 0,184 bitcoinu do její vlastní peněženky. Zůstává nepřiděleno 0,001 bitcoinu, jako implicitní poplatek za transakci.

Nyní se podívejme na rozdílný scénář. Eugenia, naše ředitelka dětské charity na Filipínách dokončila sbírku na nákup učebnic pro děti. Obdržela několik tisíc menších darů od lidí z celého světa v celkové výši 50 bitcoinů. Její peněženka je plná velmi malých UTXO, nyní chce koupit stovky učebnic od místního vydavatele a zaplatit v bitcoinech.

Peněženková aplikace Eugenie se pokusí vytvořit jednu větší platební transakci, musí poskytnout dostupnou množinu UTXO, která je složena z mnoha menších částek. To znamená, že výsledná transakce bude obsahovat více než stovku nízkohodnotových UTXO vstupů a pouze jeden výstup, platbu vydavateli knih. Transakce s mnoha vstupy bude větší než jeden kilobyte, pravděpodobně bude velká 2 až 3 kilobyty. Ve výsledku bude požadovat vyšší transakční poplatek než minimální poplatek sítě 0,0001 bitcoinu.

Peněženková aplikace Eugenie spočítá vhodný poplatek změřením velikosti transakce a vynásobení této velikosti cenou za kilobyte. Mnoho peněženek platí ještě vyšší poplatky pro vetší transakce, aby se ujistily, že transakce bude zpracována ihned. Vyšší poplatek není kvůli tomu, že Eugenia utrácí více peněz, ale protože její transakce je složitější a větší velikosti - poplatek nezávisí na hodnotě převáděných bitcoinů.

Zřetězování transakcí a osiřelé transakce

Jak jsme viděli, transakce tvoří řetěz, přičemž jedna transakce utrácí výstupy předchozích transakce (známé jako rodič) a tvoří výstupy pro následující transakci (známou jako dítě). Někdy celý řetěz transakcí záleží na každé z nich. Řekněme, že rodič, dítě a vnouče transakce, jsou vytvořeny ve stejný čas, aby splnili složitý pracovní postup, který požaduje, aby platný potomek byl podepsán dříve než je podepsán rodič. Například, tato technika se používá v transakci CoinJoin, při které mnoho stran spojuje své transakce dohromady, aby ochránili své soukromí.

Když je řetěz transakcí přenášen po síti, nemusí vždy dorazit ve stejném pořadí. Někdy dítě může

dorazit před rodičem. V tomto případě, uzly, které uvidí dítě dříve, poznají, že dítě odkazuje na rodičovskou transakci, která není ještě známa. Místo odmítnutí dítěte ho uloží do dočasného úložiště a čekají na příchod jeho rodiče a rozšiřují ho dalším uzlům. Úložiště transakcí bez rodičů je známé jako *úložiště osiřelých transakcí*. Jakmile rodič dorazí, všichni sirotci odkazující na UTXO vytvořené rodičem jsou propuštěni z úložiště, rekurzivně znovuověřeny, a celý řetěz transakcí může být vložen do úložiště transakcí, připravený k vytěžení v bloku. Transakční řetězy mohou být libovolně dlouhé, s libovolným počtem generací přenášených současně. Mechanizmus udržování sirotků v úložišti osiřelých transakcí zajišťuje, že jinak platné transakce nebudou odmítnuty pouze protože jejich rodič má zdržení, a že nakonec řetěz ke kterému patří je sestaven ve správném pořadí, bez ohledu na pořadí příchodu.

Existuje omezení na počet osiřelých transakcí uložených v paměti, aby se předešlo útoku odepřením služby proti uzlům bitcoinové sítě. Omezení je definováno jako MAX_ORPHAN_TRANSACTIONS ve zdrojových kódech bitcoinového referenčního klienta. Pokud počet osiřelých transakcí v úložišti přesáhne MAX_ORPHAN_TRANSACTIONS, jedna nebo více náhodně vybraných osiřelých transakcí jsou odstraněny z úložiště, dokud velikost úložiště opět nepoklesne pod toto omezení.

Transakční skripty a skriptovací jazyk

Bitcoinoví klienti ověřují transakce vykonáním skriptu, napsaného ve skriptovacím jazyku odvozeném od jazyka Forth. Jak zamykací skript (břemeno) uložený v UTXO a odemykací skript obvykle obsahují podpisy napsané v tomto skriptovacím jazyku. Když je transakce ověřována, odemykací skript pro každý vstup je vykonán spolu s odpovídajícím zamykacím skriptem, aby se vidělo, zda splňuje podmínku pro utracení.

Dnes, většina transakcí je zpracována skrz bitcoinovou síť má tvar "Alice platí Bobovi" a jsou založeny na stejném skriptu zvaném skript platby haši veřejného klíče (v originále Pay-to-Public-Key-Hash script). Nicméně, použití skriptů pro zamčení výstupů a odemčení vstupů za použití programovacího jazyka znamená, že transakce mohou mít nekonečné množství podmínek. Bitcoinové transakce nejsou omezeny na formu vzoru "Alice platí Bobovi".

Toto je pouze vrcholek ledovce možností, které mohou být vyjádřeny tímto skriptovacím jazykem. V této části si představíme součásti skriptovacího jazyka bitcoinových transakcí a ukážeme si, jak mohou být použity pro vyjádření složitých podmínek pro utrácení a jak tyto podmínky mohou být splněny odemykacími skripty.

TIP	Ověřování bitcoinových transakcí není založeno na pevném vzoru, ale je dosaženo pomocí výkonu skriptovacího jazyka. Tento jazyk umožňuje téměř nekonečno různých podmínek, které mohou být vyjádřeny. Tímto bitcoin získává sílu "programovatelných peněz."

Konstrukce skriptu (zamčení a odemčení)

Mechanizmus ověřování bitcoinových transakcí se opírá o dva typy skriptů pro ověřování transakcí: zamykací skript a odemykací skript.

Zamykací skript je závazkem, kterým je zatížen výstup, a určuje podmínky, které musejí být splněny pro utracení výstupu v budoucnu. Historicky, zamykací skript byl nazýván *skript veřejného klíče* protože obvykle obsahoval veřejný klíč nebo bitcoinovou adresu. V této knize používáme označení "zamykací skript", abychom ocenili mnohem širší škálu možností této skriptovací technologie. V mnoha bitcoinových aplikacích, co my nazýváme zamykacím skriptem se objevuje ve zdrojových kódech jako scriptPubKey.

Odemykací skript je skript, který "řeší" nebo splňuje podmínky, kterými zamykací skript zatížil výstup, a které umožňují utracení výstupu. Odemykací skript je částí každého transakčního vstupu a většinou obsahuje digitální podpis vyrobený uživatelovou peněženkou z jeho soukromého klíče. Historicky, odemykací skript byl nazýván *skriptový podpis*, protože obvykle obsahoval digitální podpis. V mnoha bitcoinových aplikacích, zdrojové kódy označují odemykací skript jako scriptSig. V této knize odkazujeme na něj jako na "odemykací skript", abychom ocenili mnohem širší škálu požadavků zamykacího skriptu, protože ne všechny odemykací skripty musejí obsahovat podpisy.

Každý bitcoinový klient ověřuje transakce vykonáním zamykacího a odemykacího skriptu dohromady. Pro každý vstup transakce, ověřovací software nejprve vyzvedne UTXO, na které odkazuje vstup. Toto UTXO obsahuje zamykací skript definující podmínky potřebné pro jeho utracení. Ověřovací software poté vezme odemykací skript obsažený ve vstupu, který se pokouší utratit tento UTXO, a vykonat tyto dva skripty.

V původním bitcoinovém klientovi, odemykací a zamykací skripty byly zřetězeny a vykonány v posloupnosti. Z bezpečnostních důvodů došlo v roce 2010 ke změně, kvůli zranitelnosti, která umožňuje chybnému odemykacímu skriptu vložit data na zásobník a poškodit zamykací skript. V současné implementaci, jsou oba skripty vykonány odděleně se zásobníkem přenesením mezi těmito dvěma vykonáními, jak je popsáno dále.

Nejprve, odemykací skript je vykonán, za použití mechanizmu zpracování zásobníku. Pokud odemykací skript byl vykonán bez chyb (například nezbývají žádné "visící" operátory), hlavní zásobník (ne alternativní zásobník) je okopírován a zamykací skript je spuštěn. Pokud výsledek vykonání zamykacího skriptu proběhlého na datech okopírovaných z odemykacího skriptu je "TRUE", odemykací skript uspěl ve vyřešení podmínek uložených zamykacím skriptem a proto vstup má platné oprávnění utratit UTXO. Pokud jakýkoliv jiný výsledek než "TRUE" zůstane po vykonání složeného skriptu, vstup je neplatný, protože neuspěl při splňování podmínek utracení umístěných v UTXO. Všimněte si, UTXO je stále uložen v blockchainu, proto je neměnný a neovlivněny neúspěšným pokusem o jeho utracení v nové transakci. Pouze platná transakce, která správně splní podmínky UTXO vede k označení UTXO jako "utraceného" a jeho odstranění z množiny dostupných (neutracených) UTXO.

Složený scriptSig a scriptPubKey pro vyhodnocení transakčního skriptu je příkladem odemykacího a zamykacího skriptu pro nejčastější typ bitcoinových transakcí (platba haši veřejného klíče). Ukazuje složený skript vzniklý zřetězením odemykacího a zamykacího skriptu před ověřením skriptu.

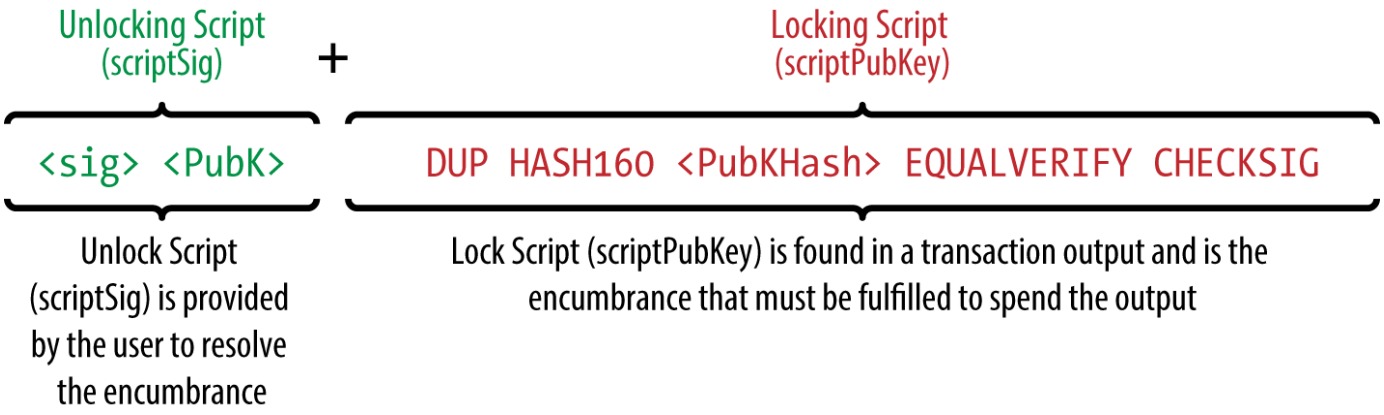


Figure 1. Složený scriptSig a scriptPubKey pro vyhodnocení transakčního skriptu

Skriptovací jazyk

Skriptovací jazyk bitcoinových transakcí zvaný *Script*, je zásobníkově orientovaný jazyk založený na jazyce Forth, používá posfixovou notaci. Pokud vám to zní jako hatmatilka, tak jste pravděpodobně nestudovali programovací jazyky 60. let minulého století. Script je velmi jednoduchý jazyk, který byl navržen s omezenou výpočetní silou a je spustitelný na řadě hardware, třeba tak jednoduchém jako jsou vestavěná zařízení, například kapesní kalkulačka. Vyžaduje minimální zpracování a nemůže provádět mnoho úžasných věcí, které mohou dělat moderní programovací jazyky. V případě programovatelných peněz je to záměrná bezpečnostní funkce.

Bitcoinový skriptovací jazyk je nazýván zásobníkový jazyk, protože používá datovou strukturou zvanou *zásobník*. Zásobník je velmi jednoduchá datová struktura, která může být zobrazena jako balíček karet. Zásobník umožňuje dvě operace: vložení a odebrání. *Vložení* přidá položku na vrchol zásobníku, *odebrání* odebere vrchní položku ze zásobníku.

Skriptovací jazyk vykonává skript zpracováním jednotlivých položek zleva doprava. Čísla (datové konstanty) jsou vkládány na zásobník. Operátory vkládají nebo odebírají jeden nebo více parametrů ze zásobníku, pracují s nimi a mohou vložit výsledek na zásobník. Například OP_ADD odebere dvě položky ze zásobníku, sečte je a výsledný součet vloží na zásobník.

Podmíněné operátory vyhodnotí podmínku, vytvoří boolovský výsledek TRUE nebo FALSE. Například OP_EQUAL odebere dvě položky ze zásobníku a vloží TRUE (reprezentované číslem 1), pokud tyto položky byly shodné nebo FALSE (reprezentované nulou), pokud tyto položky nebyly shodné. Bitcoinové transakční skripty obvykle obsahují podmíněný operátor, tak mohou vytvořit výsledek TRUE značící platnou transakci.

V Ověření bitcoinového skriptu dělajícího jednoduchou matematiku, skirpt 2 3 OP_ADD 5 OP_EQUAL představuje aritmetickou operaci OP_ADD sčítající dvě čísla a vkládající výsledek na zásobník následovanou podmíněným operátorem OP_EQUAL, který kontroluje, zda výsledný součet je roven hodnotě 5. Pro zestručnění, předpona OP_ je vynechána v příkladě krok za krokem.

Následuje mírně složitější skript, který počítá 2 + 7 – 3 + 1. Všimněte si, že když skript obsahuje několik operátorů v řadě, zásobník umožňuje výsledek jednoho operátoru použít dalším operátorem.

```
2 7 OP_ADD 3 OP_SUB 1 OP_ADD 7 OP_EQUAL
```

Zkuste zkontrolovat předchozí skript sami za použití tužky a papíru. Když vykonání skriptu skončí, měla by vám na zásobníku zůstat hodnota TRUE.

Přestože většina zamykacích skriptů odkazuje na bitcoinovou adresu nebo veřejný klíč, čímž požadují důkaz vlastnictví utrácených prostředků. skripty nemusejí být takto složité. Jakákoliv kombinace zamykacího a odemykacího skriptu, která vede k výsledků TRUE je platná. Jednoduchá aritmetika, kterou jsme použili v příkladu skriptovacího jazyka, je také platným zamykacím skriptem a může být použita k zamčení transakčního výstupu.

Použití části skriptu z aritmetického příkladu jako zamykacího skriptu:

```
3 OP_ADD 5 OP_EQUAL
```

která může být splněna transakcí obsahující na vstupu odemykacího skriptu:

```
2
```

Ověřovací software spojí zamykací a odemykací skripty a výsledný skript je:

```
2 3 OP_ADD 5 OP_EQUAL
```

Jak vidíme v příkladu krok za krokem v Ověření bitcoinového skriptu dělajícího jednoduchou matematiku, když je skript vykonán, výsledek OP_TRUE učiní transakci platnou. Není to jen platný transakční výstup zamykacího skriptu, ale výsledný UTXO může být utracen kýmkoliv s aritmetickou dovedností, kdo ví, že číslo 2 splňuje skript.

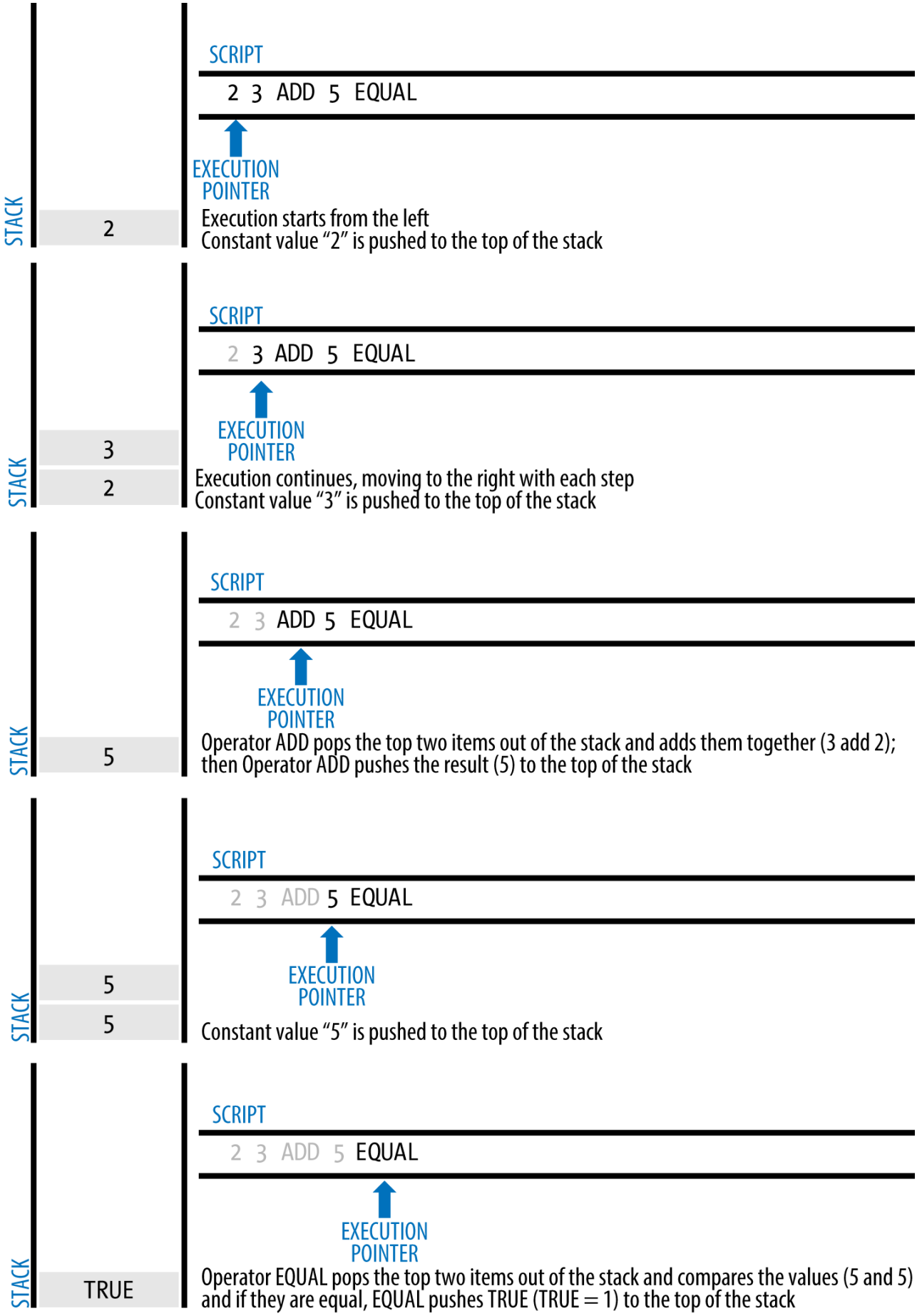

SCRIPT
2 3 ADD 5 EQUAL
EXECUTION POINTER
STACK
2
Execution starts from the left
Constant value "2" is pushed to the top of the stack
SCRIPT
2 3 ADD 5 EQUAL
EXECUTION POINTER
STACK
3
2
Execution continues, moving to the right with each step
Constant value "3" is pushed to the top of the stack
SCRIPT
2 3 ADD 5 EQUAL
EXECUTION POINTER
STACK
5
Operator ADD pops the top two items out of the stack and adds them together (3 add 2);
then Operator ADD pushes the result (5) to the top of the stack
SCRIPT
2 3 ADD 5 EQUAL
EXECUTION POINTER
STACK
5
5
Constant value "5" is pushed to the top of the stack
SCRIPT
2 3 ADD 5 EQUAL
EXECUTION POINTER
STACK
TRUE
Operator EQUAL pops the top two items out of the stack and compares the values (5 and 5)
and if they are equal, EQUAL pushes TRUE (TRUE = 1) to the top of the stack

Figure 2. Ověření bitcoinového skriptu dělajícího jednoduchou matematiku

TIP

Transakce jsou platné pokud výsledek na vrcholu zásobníku je TRUE (značeno jako {0x01}), jakákoliv jiná nenulové hodnota nebo pokud je vrchol zásobníku prázdný po vykonání skriptu. Transakce jsou neplatné pokud na vrcholu zásobníku je FALSE (prázdná hodnota nulové délky, značeno jako { }), nebo pokud běh skriptu byl přerušen explicitně operátorem jako OP_VERIFY, OP_RETURN nebo podmíněným ukončením jako OP_ENDIF. Viz [tx_script_ops] pro podrobnosti.

Turingovská neúplnnost

Bitcoinový transakční skriptovací jazyk obsahuje mnoho operátorů, ale je úmyslně omezen v jednom důležitém směru - neobsahuje cykly ani schopnosti složitého řízení toku programu jiného než podmíněné řízení toku programu. Toto zajišťuje, že tento jazyk není *turingovsky úplný*, což znamená, že skripty mají omezenou složitost a předpověditelný čas běhu. Script není jazyk pro obecné použití. Tato omezení zajišťují, že jazyk nemůže být použit pro vytvoření nekonečného cyklu nebo jiných "logických bomb", které by mohly být obsaženy v transakci takovým způsobem, že by způsobily útok odepřením přístupu proti bitcoinové síti. Pamatujte, každá transakce je ověřována každým úplným uzlem bitcoinové sítě. Omezený jazyk zabraňuje, aby mechanizmus ověřování transakcí byl použit jako zranitelnost.

Bezestavové ověřování

Bitcoinový transakční skript je bezstavový jazyk, není zde žádný stav před vykonáním skriptu nebo stav uložený po vykonání skriptu. Proto, všechny informace nutné pro vykonání skriptu jsou obsaženy ve skriptu samotném. Skript bude předpověditelným způsobem vykonán stejně na každém systému. Pokud systém ověří skript, můžete si být jisti, že každý jiný systém v bitcoinové síti také ověří skript, což znamená, že platná transakce je platná pro každého a každý to ví. Tato předpověditelnost výstupů je základní přínosem bitcoinového systému.

Standardní transakce

V prvních několika letech vývoje bitcoinu, vývojáři představili některá omezení, v typech skriptu, které mohou být zpracovány referenčním klientem. Tato omezení jsou zakódována ve funkci zvané isStandard(), která definuje pět typů "standardních" transakcí. Tato omezení jsou dočasná a mohou být odstraněna v čase čtení tohoto textu. Do té doby, pět standardních typů transakčních skriptů jsou jedinými, které budou přijímány referenčním klientem a většinou těžařů, kteří provozují referenčního klienta. Přestože je možné vytvořit nestandardní transakci obsahující skript, který není jedním ze standardních typů musíte nalézt těžaře, který nebude následovat tato omezení a vytěží tuto transakci do bloku.

Zkontrolujte zdrojové kódy Bitcoin Core klienta (referenční implementace), abyste viděli co je v současnosti dovoleno za povolené transakční skripty.

Pět standardních typů transakčních skriptů jsou platba haši veřejného klíče (P2PKH), veřejný klíč,

vícepodpisové (omezeno na 15 klíčů), platba haši skriptu (P2SH) a datový výstup (OP_RETURN), kterou jsou detailněji popsány v následujících sekcích.

Platba haši veřejného klíče (P2PKH)

Drtivá většina transakcí zpracovaných bitcoinové sítí jsou transakce P2PKH. Ty obsahují zamykací skript, který výstup zatíží břemenem ve prospěch haše veřejného klíče, známějšího jako bitcoinová adresa. Transakce, které platí bitcoinové adrese obsahují P2PKH skripty. Výstup zamčený P2PKH skriptem může být odemčen (utracen) prokázání se veřejným klíčem a digitálním podpisem vytvořeným odpovídajícím soukromým klíčem.

Například, podívejme se znova na platbu Alice Bobově kavárně. Alice udělala platbu 0,015 bitcoinu na bitcoinovou adresu kavárny. Transakční výstup bude mít zamykací skript ve tvaru:

```
OP_DUP OP_HASH160 <Cafe Public Key Hash> OP_EQUAL OP_CHECKSIG
```

Cafe Public Key Hash je shodný s bitcoinovou adresou kavárny, bez kódování Base58Check. Většina aplikací ukáže *haš veřejného klíče* v hexadecimálním kódování a ne známém Base58Check formátu bitcoinové adresy, který začíná na "1".

Předchozí zamykací skript může být splněn odemykacím skriptem ve tvaru:

```
<Cafe Signature> <Cafe Public Key>
```

Tyto dva skripty dohromady vytvoří spojený ověřovací skript:

```
<Cafe Signature> <Cafe Public Key> OP_DUP OP_HASH160
<Cafe Public Key Hash> OP_EQUAL OP_CHECKSIG
```

Když je vykonán, spojený skript bude vyhodnocen na TRUE, pokud a jen pokud, odemykací skript splnil podmínky zadané zamykacím skriptem. Jinými slovy, výsledek bude TRUE, pokud odemykací skript má platný podpis soukromého klíče kavárny, který odpovídá haši veřejného klíče v jehož prospěch je nastaveno břemeno.

Schémata Vyhodnocení skriptu pro P2PKH transakci (část 1 z 2) a Vyhodnocení skriptu pro P2PKH transakci (část 2 z 2) ukazují (ve dvou částech) krok za krokem vykonání spojeného skriptu, který dokáže, že toto je platná transakce.

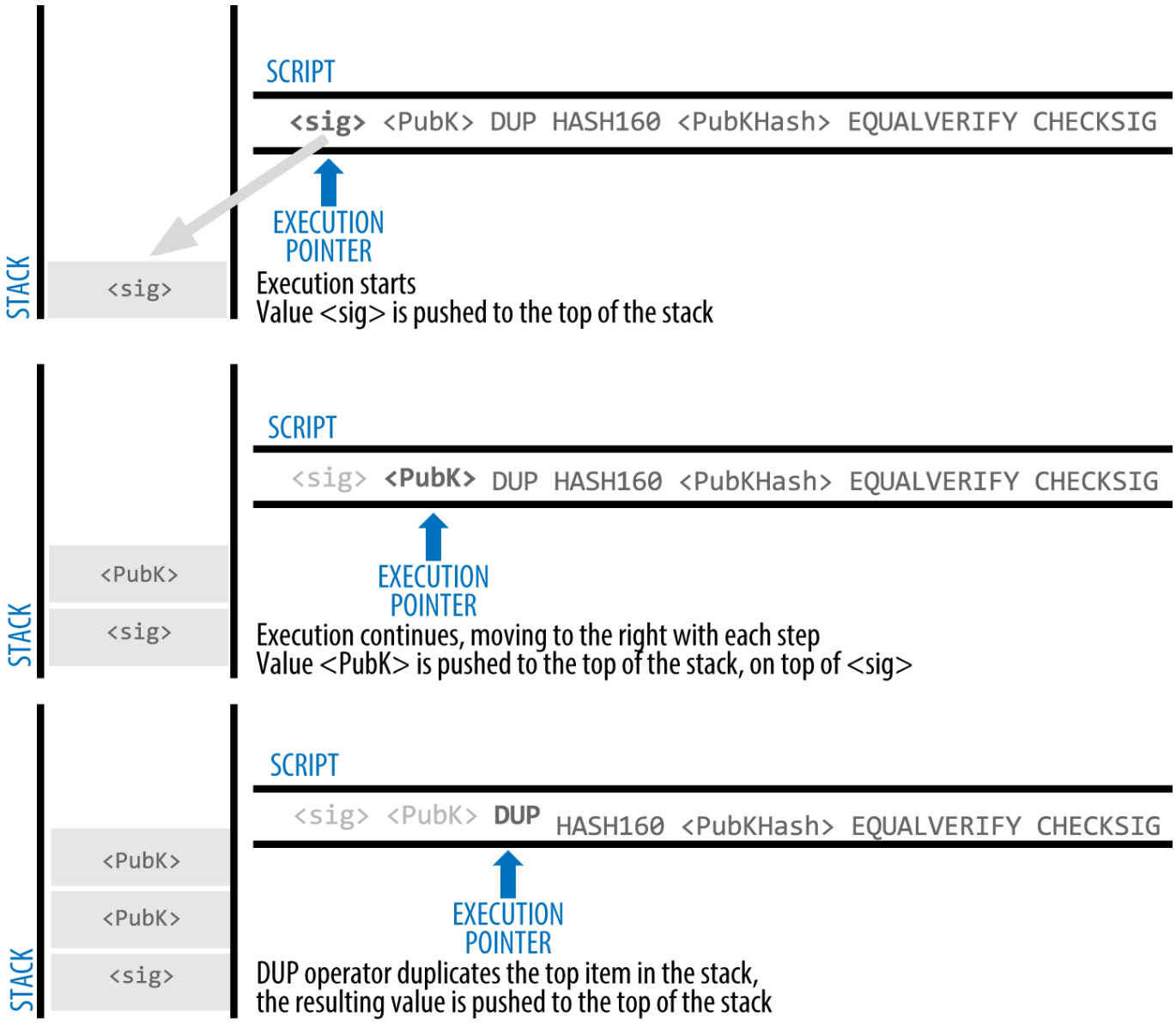


Figure 3. Vyhodnocení skriptu pro P2PKH transakci (část 1 z 2)

Platba veřejnému klíči

Platba veřejnému klíči je jednodušší tvar bitcoinové platby než platba haši veřejného klíče. S tímto tvarem skriptu, samotný veřejný klíč je uložen v zamykacím skriptu místo haše veřejného klíče, jak tomu bylo u P2PKH, který je o mnoho kratší. Platba haši veřejného klíče byla objevena Satoshim, aby udělal bitcoinové adresy kratší pro snazší použití. Platba veřejnému klíči je nyní často k vidění v mincetvorných transakcích, tvořených starším těžebním software, který nebyl aktualizován na použití P2PKH.

Platba veřejnému klíči zamykací skript vypadá jako tento:

```
<Public Key A> OP_CHECKSIG
```

Odpovídající odemykací skript musí být předložen pro odemknutí tohoto typu výstupu je jednoduše podpis, jako tento:

```
<Signature from Private Key A>
```

Složený skript, který je ověřován softwarem ověřování transakcí je:

```
<Signature from Private Key A> <Public Key A> OP_CHECKSIG
```

Tento skript je jednoduchým vyvoláním operátoru CHECKSIG, který ověřuje podpis, zda patří ke správnému klíči a vrací TRUE na zásobník.

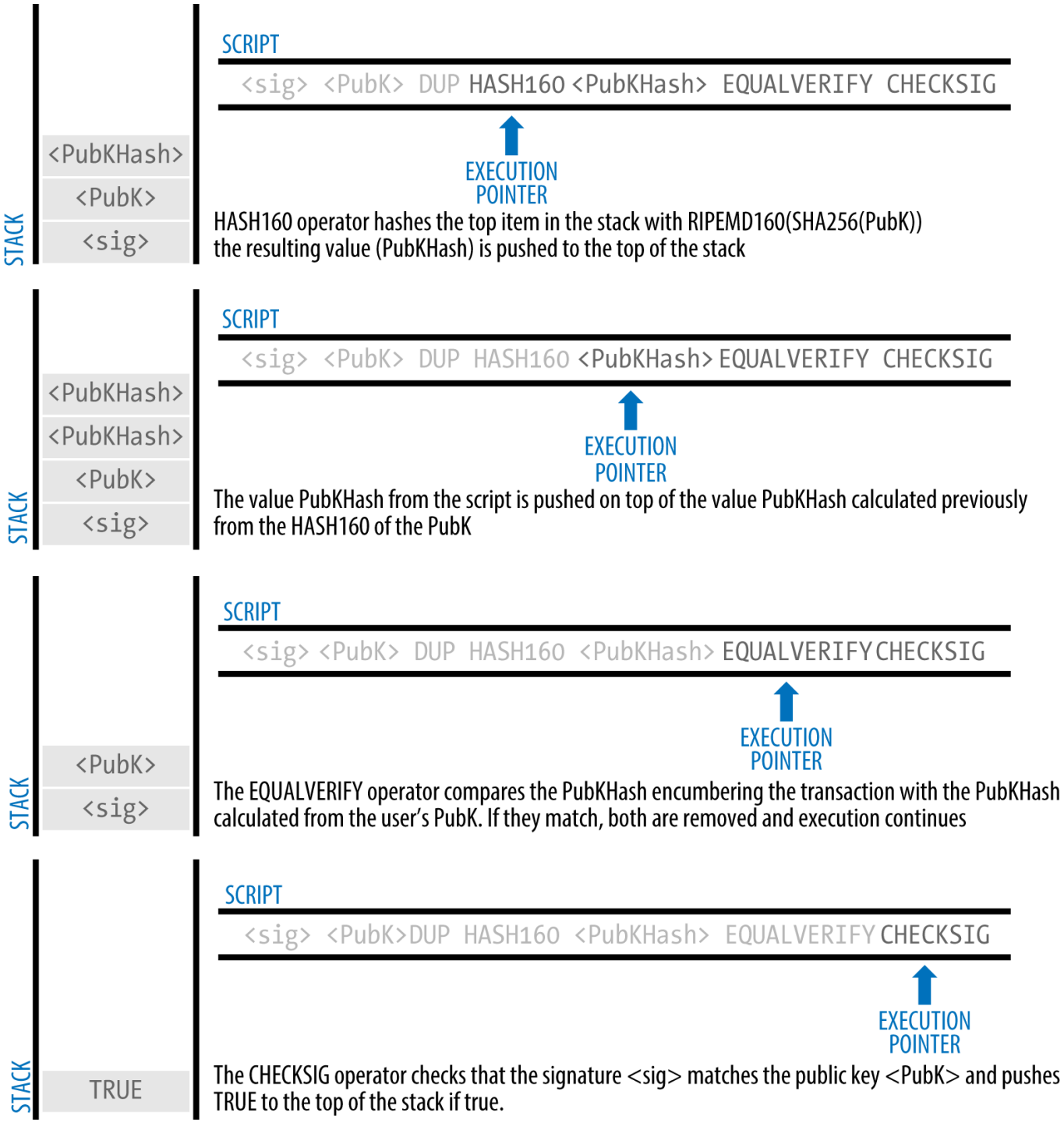


Figure 4. Vyhodnocení skriptu pro P2PKH transakci (část 2 z 2)

Vícepodpisové

Vícepodpisové skripty nastavují podmínku, kde N veřejných klíčů je zaznamenáno ve skriptu a minimálně M z nich musí poskytnout podpis pro uvolnění břemene. Toto je také známo jako M-z-N schéma, kde N je celkový počet klíčů a M je práh podpisů nutných pro ověření. Například 2-ze-3 vícepodpisovost znamená, že tři veřejné klíče jsou v seznamu potenciálních podpisovatelů a alespoň dva z nich musejí být použity pro vytvoření podpisu pro platnou transakci utrácející tyto prostředky. V tomto čase standardní vícepodpisové skripty jsou omezeny na seznam nejvýše 15 veřejných klíčů, což

znamená, že můžete vytvořit cokoliv od 1-z-1 až do 15-z-15 vícepodpisovosti nebo jakoukoliv kombinaci v tomto intervalu. Omezení 15 klíčů v seznamu může být zvýšeno v čase publikování této knihy, proto zkontrolujte funkci isStandard(), abyste viděli, co je aktuálně akceptováno sítí.

Obecný tvar zamykacího skriptu nastavujícího M-z-N vícepodpisovou podmínku je:

```
M <Public Key 1> <Public Key 2> ... <Public Key N> N OP_CHECKMULTISIG
```

kde N je celkový počet veřejných klíčů v seznamu a M je práh potřebných podpisů pro utracení výstupu.

Zamykací skript nastavující 2-ze-3 vícepodpisovou podmínku vypadá takto:

```
2 <Public Key A> <Public Key B> <Public Key C> 3 OP_CHECKMULTISIG
```

Předchozí zamykací skript může být splněn odemykacím skriptem obsahujícím dvojici podpisů a veřejných klíčů:

```
OP_0 <Signature B> <Signature C>
```

nebo jinou kombinaci dvou podpisů od soukromých klíčů odpovídajícím třem veřejných klíčů ze seznamu.

NOTE Předpona OP_0 je požadována protože v originální implementaci CHECKMULTISIG je chyba, která jednu položku mnohokrát odebírá ze zásobníku. Je to ignorováno CHECKMULTISIG a je to pouze zástupný symbol.

Tyto dva skripty dohromady vytvoří spojený ověřovací skript:

```
OP_0 <Signature B> <Signature C> 2 <Public Key A> <Public Key B> <Public Key C> 3
OP_CHECKMULTISIG
```

Když je vykonán, spojený skript bude vyhodnocen TRUE, pokud a jen pokud, odemykací skript splnil podmínky zadané zamykacím skriptem. V tomto případě, podmínkou je, zda odemykací skript má platný podpis ze dvou soukromých klíčů, které odpovídají dvěma ze tří veřejných klíčů nastavených jako břemeno.

Datový výstup (OP_RETURN)

Bitcoinový distribuovaný a časovými značkami opatřený účetní systém, blockchain, má potenciální využití daleko za hranicí plateb. Mnoho vývojářů zkoušelo použít transakční skriptovací jazyk, aby využili bezpečnosti a odolnosti systému pro aplikace jako digitální služby notářů, burzovní certifikáty

a chytré kontrakty. Brzké pokusy použití bitcoinového skriptovacího jazyka pro tyto účely zahrnovaly vytváření transakčních výstupů a zaznamenávání dat na blockchainu; například, pro zaznamenání digitálního otisku souboru, takovým způsobem, že kdykoliv může potvrdit důkaz existence tohoto souboru v daném čase odkazem na tuto transakci.

Použití bitcoinového blockchainu pro uložení dat nesouvisejících s bitcoinovými platbami je rozporuplné téma. Mnoho vývojářů považuje takové použití za obtěžující a chce od něho odrazovat. Ostatní v tom vidí ukázku mocných schopností blockchainové technologie a chtějí podporovat takovéto experimentování. Ty, kteří nesouhlasí s vkládáním neplatebních dat, tvrdí, že to způsobuje nafukování blockchainu, zatěžování běžících úplných bitcoinových uzlů a přináší to náklady na disková úložiště pro data, která blockchain neměl původně v úmyslu nést. Navíc takovéto transakce vytvářejí UTXO, které nemohou být utraceny, používají cílovou bitcoinovou adresu jako volné 20-bytové pole. Protože adresa je použita pro data, neodpovídá soukromému klíči a výsledné UTXO nemůže být *nikdy* utraceno; je to falešná platba. Tyto transakce, které nemůžou být utraceny a proto nebudou nikdy odstraněny z množiny UTXO a způsobují, že velikost UTXO databáze je navždy zvýšena nebo "nafouknutá".

Ve verzi 0.9 Bitcoin Core klienta bylo dosaženo kompromisu. Byl představen operátor OP_RETURN, který umožňuje vývojářům přidat 80 bytů neplatebních dat do transakčního výstupu. Nicméně, na rozdíl od použití "falešného" UTXO, operátor OP_RETURN vytváří explicitně _dokazatelně neutratitelný_ výstup, který není třeba ukládat v množině UTXO. Výstupy +OP_RETURN jsou zaznamenány v blockchainu, mohou spotřebovávat diskový prostor a přispívat ke zvýšení velikosti blockchainu, ale nejsou uloženy v množině UTXO a proto nenafukují paměťově úložiště UTXO a nezatěžují úplné uzly s náklady na dražší RAM.

Skript OP_RETURN vypadá jako tento:

```
OP_RETURN <data>
```

Datová část je omezena na 80 bytů a nejčastěji reprezentuje haš, jako výstup algoritmu SHA256 (32 bytů). Mnoho aplikací vkládá předponu na začátek dat, aby pomohli identifikovat aplikaci. Například Proof of Existence digitální notářská služba používá 8-bytový prefix "DOCPROOF, který je v ASCII kódován jako 44f4350524f4f46 hexadecimálně.

Mějme na paměti, že tyto "neodemykací skripty" odpovídají OP_RETURN, který by případně mohl být použit pro "utracení" výstupu OP_RETURN. Smyslem OP_RETURN je, že nemůže utratit peníze, zamčené v tomto výstupu, a proto nepotřebuje být držen v množině UTXO jako možně utratitelný. OP_RETURN *dokazatelně neutratitelný*. OP_RETURN je obvykle výstup s nulovou částkou bitcoinů, protože jakékoliv bitcoiny přiřazené tomuto výstupu jsou fakticky navždy ztracené. Pokud na OP_RETURN narazí software ověřující skript, dojde k okamžitému zastavení běhu validačního skriptu a transakce je označena za neplatnou. Proto, pokud nešťastnou náhodou odkážete na výstup OP_RETURN jako na vstup transakce, tato transakce je neplatná.

Standardní transakce (jedna z odpovídajících kontrole isStandard()) může mít pouze jeden výstup OP_RETURN . Nicméně jeden výstup OP_RETURN může být kombinován v transakci s výstupy

jakýchkoliv jiných typů.

Nová nastavení příkazové řádky byla přidána do Bitcoin Core ve verzi 0.10. Nastavení datacarrier kontroluje přenos a těžbu OP_RETURN transakcí, standardně je nastavena na "1", čímž je umožňuje. Nastavení datacarriersize ve svém číselném parametru určuje maximální velikost dat v bytech pro OP_RETURN data, standardně 40 bytů.

NOTE	OP_RETURN byl původně navržen na omezení 80 bytů, ale omezení bylo sníženo na 40 bytů, když byla tato funkce uvolněna. V únoru 2015, ve verzi 0.10 Bitcoin Core, byl limit vrácen nazpět na 80 bytů. Uzly si mohou vybrat přenášet nebo těžit OP_RETURN, nebo přenášet a těžit OP_RETURN obsahující méně než 80 bytů dat.

Platba haši skriptu (P2SH)

Platba haši skriptu (v originále pay-to-script-hash, zkratka P2SH) byla představena v roce 2012 jako nový účinný typ transakcí, které značně zjednodušili použití složitých transakčních skriptů. Pro vysvětlení potřeby P2SH se podívejme na praktický příklad.

V [ch01_intro_what_is_bitcoin] jsme představili Mohammeda, dovozce elektroniky do Dubaje. Mohammedova firma používá bitcoinovou vícepodpisovou funkci rozsáhle pro podnikové účty. Vícepodpisové skripty jsou jedním z nejčastějších použití výhod bitcoinových pokročilých skriptovacích schopností a jsou velmi účinnou funkcí. Mohamedova firma používá vícepodpisové skripty pro platby všech zákazníků, známé v účetnictví pod pojmem "pohledávky". Ve vícepodpisovém schématu, každá platba učiněná zákazníky je zamčena takovým způsobem, že jsou potřeba alespoň dva podpisy pro její odemčení, od Mohammeda a jednoho z jeho partnerů nebo jeho právníka, který má záložní klíč. Vícepodpisové schéma jako toto nabízí vedení podniku kontrolu a ochranu proti krádeži, zpronevěře nebo ztrátě.

Výsledný skript je docela dlouhý a vypadá jako tento

```
2 <Mohammed's Public Key> <Partner1 Public Key> <Partner2 Public Key> <Partner3 Public
Key> <Attorney Public Key> 5 OP_CHECKMULTISIG
```

Přestože vícepodpisové skripty jsou mocnou funkcí, jsou těžkopádné k použití. Vezměme předchozí skript, Mohammed bude muset sdělit tento skript každému zákazníkovi před platbou. Každý zákazník bude muset použít zvláštní software bitcoinové peněženky se schopností vytvářet uživatelské transakční skripty a každý zákazník bude muset rozumět tomu, jak vytvořit transakci používající uživatelské skripty. Kromě toho výsledný transakční skript bude pětkrát větší než jednoduchá platební transakce, protože tento skript obsahuje velmi dlouhé veřejné klíče. Břemeno příliš velké transakce zatíží zákazníka ve formě poplatků. Nakonec, velký transakční skript jako tento bude udržován v množině UTXO v RAM na každém plném uzlu, dokud nebude utracen. Všechny tyto problémy dělají použití složitých výstupních skriptů obtížným v praxi.

Platba haši skriptu (P2SH) byla vyvinuta, aby vyřešila tyto praktické obtíže a udělala použití složitých skriptů tak snadným, jako platby bitcoinové adrese. S platbami P2SH, složitý zamykací skript je

nahrazen digitálním otiskem, kryptografickým hašem. Když se transakce pokusí utratit tento UTXO, je uvedena později, musí obsahovat skript, odpovídající haši navíc k odemykajícímu skriptu. Jednoduše řečeno, P2SH znamená platba skriptu odpovídajícímu tomuto haši, skript bude uveden později, až bude tento výstup utracen.

V P2SH transakcích, zamykací skript je nahrazen hašem, který je nazýván jako *vyplácející skript* protože je představen systému v čase vyplácení na rozdíl od zamykacího skriptu. Složitý skript bez P2SH ukazuje skript bez P2SH a Složitý skript jako P2SH ukazuje ten samý skript kódovaný s P2SH.

Table 4. Složitý skript bez P2SH

Zamykací skript	2 PubKey1 PubKey2 PubKey3 PubKey4 PubKey5 5 OP_CHECKMULTISIG
Odemykací skript	Sig1 Sig2

Table 5. Složitý skript jako P2SH

Vyplácející skript	2 PubKey1 PubKey2 PubKey3 PubKey4 PubKey5 5 OP_CHECKMULTISIG
Zamykací skript	OP_HASH160 <20-bytový haš vyplácejícího skriptu> OP_EQUAL
Odemykací skript	Sig1 Sig2 vyplácející skript

Jak můžete vidět z tabulek, s P2SH složitý skript, který popisuje podmínky pro utracení výstupu (vyplácející skript) není součástí zamykacího skriptu. Pouze jeho haš je v zamykacím skriptu a samotný vyplácející skript je uveden později, jako část odemykacího skriptu, když je výstup utracen. Toto přesouvá břemeno poplatků a složitosti z odesilatele na příjemce (utratitele) transakce.

Podívejme se na Mohammedovu firmu, složitý vícepodpisový skript a odpovídající P2SH skript.

Nejprve, vícepodpisový skript, který Mohammedova firma používá pro všechny příchozí platby od zákazníku:

```
2 <Mohammed's Public Key> <Partner1 Public Key> <Partner2 Public Key> <Partner3 Public
Key> <Attorney Public Key> 5 OP_CHECKMULTISIG
```

Pokud jsou značky nahrazeny aktuálními veřejnými klíči (zde zobrazeny jako 520-bitová čísla začínající na 04), můžete vidět, že skript se stává velmi dlouhým:

```
2
04C16B8698A9ABF84250A7C3EA7EEDEF9897D1C8C6ADF47F06CF73370D74DCCA01CDCA79DCC5C395D7EEC6984
D83F1F50C900A24DD47F569FD4193AF5DE762C58704A2192968D8655D6A935BEAF2CA23E3FB87A3495E7AF308
EDF08DAC3C1FCBFC2C75B4B0F4D0B1B70CD2423657738C0C2B1D5CE65C97D78D0E34224858008E8B49047E632
48B75DB7379BE9CDA8CE5751D16485F431E46117B9D0C1837C9D5737812F393DA7D4420D7E1A9162F0279CFC1
0F1E8E8F3020DECDBC3C0DD389D99779650421D65CBD7149B255382ED7F78E946580657EE6FDA162A187543A9
D85BAAA93A4AB3A8F044DADA618D087227440645ABE8A35DA8C5B73997AD343BE5C2AFD94A5043752580AFA1E
CED3C68D446BCAB69AC0BA7DF50D56231BE0AABF1FDEEC78A6A45E394BA29A1EDF518C022DD618DA774D207D1
37AAB59E0B000EB7ED238F4D800 5 OP_CHECKMULTISIG
```

Celý skript může být místo toho reprezentován 20-bytovým kryptografickým hašem, nejprve aplikací hašovacího algoritmu SHA256 a následně aplikací algoritmu RIPEMD160 na výsledek. Výsledný 20-bytový haš předchozího skriptu je:

```
54c557e07dde5bb6cb791c7a540e0a4796f5e97e
```

P2SH transakce zamyká výstup tohoto ve prospěch tohoto haše místo delšího skriptu, za použití zamykacího skriptu:

```
OP_HASH160 54c557e07dde5bb6cb791c7a540e0a4796f5e97e OP_EQUAL
```

který, jak můžete vidět, je mnohem kratší. Místo "zaplať tomuto 5-ti klíčovému vícepodpisovému skriptu", odpovídající P2SH transakce je "zaplať skriptu s tímto hašem." Zákazník dělající platbu Mohammedově firmě potřebuje pouze vložit mnohem kratší zamykací skript do jeho platby. Když Mohammed chce utratit tento UTXO, musí uvést původní vyplácející skript (ten samý, jehož haš zamyká UTXO) a podpisy nutné k jeho odemčení, jako tento:

```
<Sig1> <Sig2> <2 PK1 PK2 PK3 PK4 PK5 5 OP_CHECKMULTISIG>
```

Dva skripty jsou složeny ve dvou fázích. Nejprve, vyplácející skript je zkontrolován proti zamykacímu skriptu, abychom se ujistili, že haše se shodují:

```
<2 PK1 PK2 PK3 PK4 PK5 5 OP_CHECKMULTISIG> OP_HASH160 <redeem scriptHash> OP_EQUAL
```

Pokud souhlasí haš vyplácejícího skriptu, odemykací skript je vykonán sám o sobě, pro odemčení vyplácejícího skriptu:

```
<Sig1> <Sig2> 2 PK1 PK2 PK3 PK4 PK5 5 OP_CHECKMULTISIG
```

Adresy plateb haši skriptu

Další důležitou částí funkce P2SH je schopnost kódovat haše skriptů jako adresy, jak je definováno v BIP0013. P2SH adresy jsou Base58Check kódy 20-bytového haše skriptu, stejně jako bitcoinové adresy jsou Base58Check kódy 20-bytového haše veřejného klíče. P2SH adresy používají předponu "5", která vede k tomu, že Base58Check kódované adresy začínají na "3". Například, Mohamedův složitý skript, hašovaný a kódovaný Base58Check jako P2SH adresa se stává 39RF6JqABiHdYHkfChV6USGMe6Nsr66Gzw. Nyní, Mohammed může dát tuto "adresu" jeho zákazníkům a ty mohou použít téměř jakoukoliv bitcoinovou peněženku k provedení jednoduché platby, jako by to byla bitcoinová adresa. Předpona "3" jim prozrazuje, že se jedná o zvláštní typ adresy, odpovídající skriptu místo veřejnému klíči, ale jinak pracuje přesně stejným způsobem jako platba bitcoinové adrese.

P2SH adresy skrývají všechnu svojí složitost, takže osoba provádějící platbu nevidí skript.

Výhody platby haši skriptu

Funkce platba haši skriptu nabízí následující výhody v porovnání s přímým použitím složitého skriptu zamykajícího výstupy:

- Složité skripty jsou nahrazeny kratšími otisky v transakčních výstupech, což zmenšuje velikost transakce.
- Skripty mohou být kódovány jako adresy, takže odesilatel a jeho peněženka nepotřebují složité konstrukce pro implementaci P2SH.
- P2SH přesouvá břemeno konstrukce skriptu na příjemce, nikoliv na odesilatele.
- P2SH přesouvá břemeno datového úložiště pro dlouhý skript z výstupu (který je v množině UTXO) na vstup (uložený v blockchainu).
- P2SH přesouvá břemeno datového úložiště pro dlouhý skript z ze současnosti (platba) do budoucnosti (když je utracena).
- P2SH přesouvá zvýšený transakční poplatek za dlouhý skript z odesilatele na příjemce, který vloží dlouhý vyplácející skript pro utracení.

Vyplácející skript a isStandard ověřování

Před verzí 0.9.2 Bitcoin Core klienta, platba haši skriptu byla omezena na standardní typy bitcoinových transakčních skriptů uvedených ve funkci isStandard(). To znamená, že vyplácející skript představený v utrácející transakci mohl být pouze jedním ze standardních typů: P2PK, P2PKH. přirozený vícepodpisový, vynechávající OP_RETURN a samotný P2SH.

Ve verzi 0.9.2 Bitcoin Core klienta, P2SH transakce mohou obsahovat jakýkoliv platný skript, což činí standard P2SH mnohem více pružným a umožňující experimentovat se spoustou nových a složitých typů transakcí.

Všimněte si, že nejste schopni vložit P2SH dovnitř P2SH vyplácejícího skriptu, protože P2SH specifikace není rekurzivní. Nejste stále schopni použít OP_RETURN ve vyplácejícím skriptu, protože

OP_RETURN nemůže vyplácet z definice.

Všimněte si, protože vyplácející skript není představený síti, dokud se nepokusíte utratit P2SH výstup, pokud zamknete výstup hašem neplatného skriptu, bude transakce zpracována bez ohledu na to. Nicméně, nebudete schopni tento výstup utratit protože utrácející transakce, která obsahuje vyplácející skript, nebude přijata, protože její skript je neplatný. To vytváří riziko, protože můžete zamknout bitcoiny v P2SH, který nemůže být utracen později. Síť příjme P2SH břemeno, dokonce pokud odpovídá neplatnému vyplácejícímu skriptu, protože haš skriptu nedává žádný náznak toho, jaký skript reprezentuje.

WARNING

P2SH zamykací skripty obsahují haš vyplácejícího skriptu, který nedává žádnou nápovědu, pokud jde o obsah samotného vyplácejícího skriptu. P2SH transakce bude považována za platnou a bude akceptována dokonce i když je skript neplatný. Můžete nedopatřením zamknout bitcoiny takovým způsobem, že nepůjdou utratit v budoucnu.

Bitcoinová síť

Peer-to-Peer architektura sítě

Bitcoin má strukturu peer-to-peer síťové architektury na vrcholu internetu. Pojem peer-to-peer nebo P2P (česky klient-klient) znamená, že počítače účastnící se sítě jsou si vrstevníky každý každému, jsou si všechny rovny, neexistují žádné "zvláštní" uzly, a že všechny uzly sdílejí břemeno poskytování síťových služeb. Uzly sítě jsou propojeny v obecný graf s plochou topologií. Není zde server, nejsou zde centrální služby a není zde hierarchie v síti. Uzly v peer-to-peer síti zároveň poskytují a konzumují služby ve stejnou dobu s reciprocitou, která působí jako pobídka k účasti. Peer-to-peer sítě jsou neodmyslitelně odolné, decentralizované a otevřené. Jedním z největších příkladu P2P síťové architektury byl brzký internet samotný, kde uzly sítě internetového protokolu si byly rovny. Dnešní internetová architektura je hierarchická, ale internetový protokol stále zachovává ve své podstatě plochou topologií. Za bitcoinem, největší a nejúspěšnější aplikace P2P technologie je sdílení souborů Napsterem jako průkopníkem a BitTorrentem jako nejnovější evolucí v architektuře.

Bitcoinová P2P síťová architektura je mnohem více než volbou topologie. Bitcoin je peer-to-peer digitální hotovostní systém návrhem, a sítová architektura je zároveň odrazem a základem této klíčové charakteristiky. Decentralizace kontroly je základním principem návrhu a může ji být dosaženou a udrženo pouze plochou, decentralizovanou P2P shodou sítě.

pojem "bitcoinová síť" odkazuje na kolekci uzlů provozujících bitcoinový P2P protokol. Navíc k bitcoinovému P2P protokolu, existují další protokoly jako Stratum, který je používán pro těžbu a odlehčené nebo mobilní peněženky. Tyto dodatečné protokoly jsou poskytovány branami směrovacích serverů, které přistupují k bitcoinové síti pomocí bitcoinového P2P protokolu a poté rozšiřují síť o uzly provozující jiné protokoly. Například Stratum servery spojují Stratum těžební uzly pomocí Stratum protokolu s hlavní bitcoinovou sítí a přemosťují Stratum protokol do bitcoin P2P protokolu. Používáme pojem "rozšířená bitcoinová síť" pro označení celé bitcoinové sítě obsahující bitcoin P2P protokol, protokoly sdílené těžby, Stratum protokol a další protokoly vztahující se ke spojení komponent bitcoinového systému.

Typy uzlů a role

Přestože uzly v bitcoinové P2P síti jsou si rovny, mohou přejímat různé role v závislosti na funkcionalitě, kterou podporují. Bitcoinový uzel je kolekce funkcí: směrování, blockchainová databáze, těžba a služby peněženek. Úplný uzel se všemi čtyřmi z těchto funkcí je ukázán v Uzel bitcoinové sítě se všemi čtyřmi funkcemi: peněženka, těžař, úplná blockchainová databáze a směrování sítě.

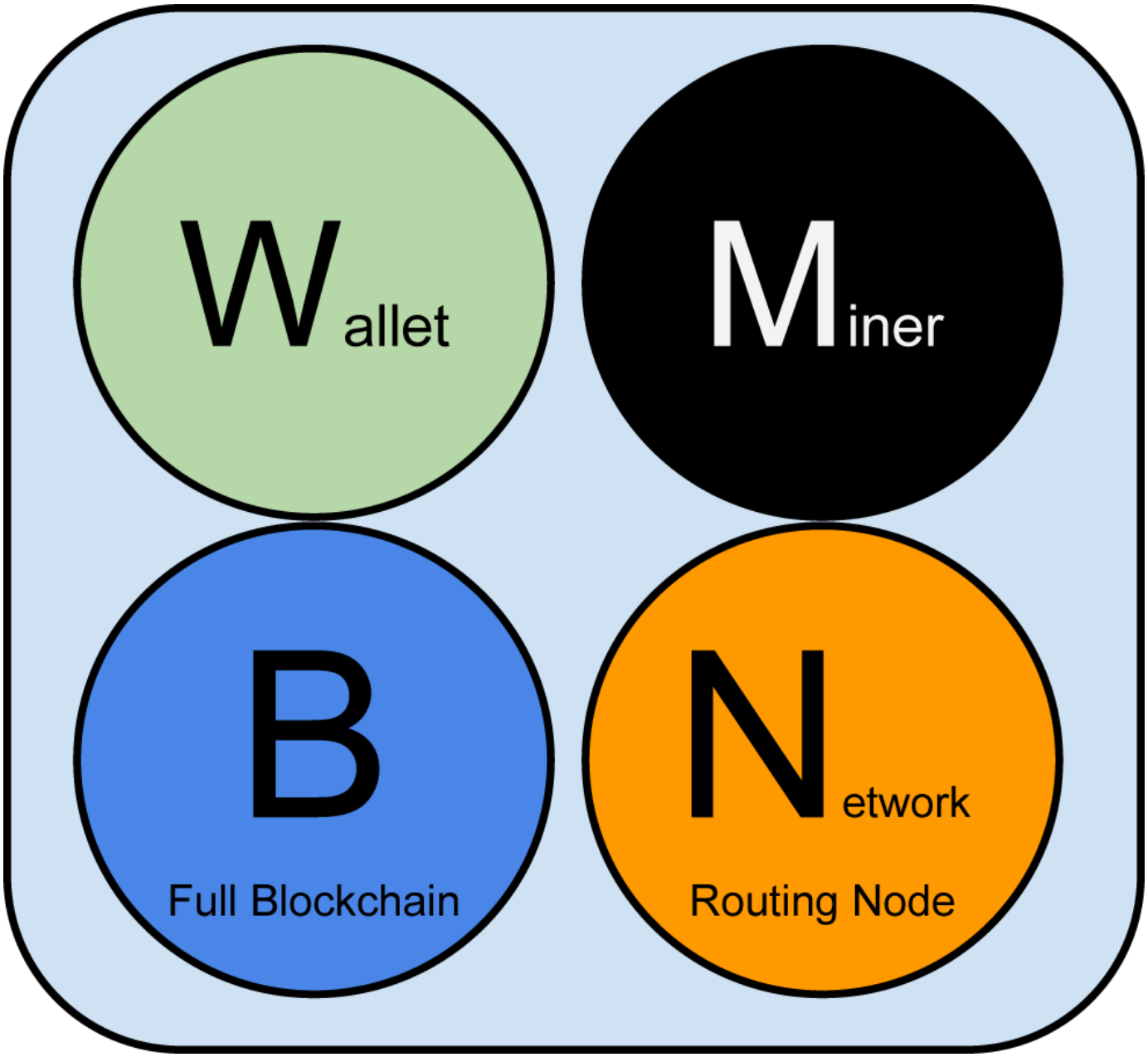


Figure 1. Uzel bitcoinové sítě se všemi čtyřmi funkcemi: peněženka, těžař, úplná blockchainová databáze a směrování sítě

Všechny uzly obsahují směrovací funkci, aby se účastnili sítě a mohou obsahovat další funkčnost. Všechny uzly ověřují a rozšiřují transakce a bloky a objevují a udržují spojení se s jinými uzly. V úplné uzlu na příkladu Uzel bitcoinové sítě se všemi čtyřmi funkcemi: peněženka, těžař, úplná blockchainová databáze a směrování sítě je směrovací funkce označena oranžovým kruhem pojmenovaným "Network Routing Node."

Některé uzly, zvané úplné uzly, také udržují a doplňují aktuální kopii blockchainu. Úplné uzly mohou anonymně a panovačně ověřovat transakce bez vnějších odkazů. Některé uzly udržují pouze podmnožinu blockchainu a ověřují transakce za použití metody zvané "simplified payment verification (SPV) nodes","defined") *zjednodušené ověřování plateb* (v originále simplified payment verification, zkratka SPV). Tyto uzly jsou známé jako SPV nebo odlehčené uzly. V příkladě úplného uzlu na obrázku, funkce blockchainové databáze je označena modrým kruhem pojmenovaným "Full

Blockchain." V Rozšířená bitcoinová síť ukazující různé typy uzlů, bran a protokolů SPV uzly jsou kresleny bez modrého kruhu, což ukazuje, že neudržují plnou kopii blockchainu.

Těžební uzly soutěží ve vytváření nových bloků. Provozují specializovaný hardware řešící algoritmus důkazu prací. Některé těžební uzly jsou úplné uzly, udržující plnou kopii blockchainu, zatímco ostatní jsou odlehčené uzly účastnící se těžební skupiny a závisející na serveru těžební skupiny, který udržuje úplný uzel. Těžební funkce je zakreslena v úplném uzlu jako černý kruh pojmenovaný "Miner."

Uživatelské peněženky mohou být částí úplného uzlu, což je obvykle případ stolních bitcoinových klientů. Zvyšující se počet uživatelů peněženek, speciálně těch běžících ne zařízeních s omezenými zdroji, jako jsou chytré telefony, jsou SPV uzly. Funkce peněženky je zobrazena v Uzel bitcoinové sítě se všemi čtyřmi funkcemi: peněženka, těžař, úplná blockchainová databáze a směrování sítě jako zelený kruh pojmenovaný "Wallet".

Navíc k hlavním typům uzlům v bitcoin P2P protokolu existují servery a uzly provozující jiné protokoly, jako specializované protokoly těžebních skupin a odlehčení klienti mající pouze přístupové protokoly.

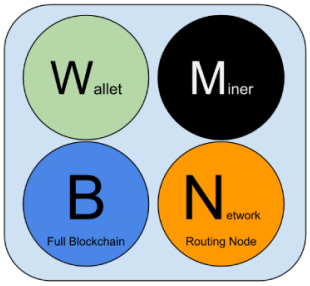
Různé typy uzlů v rozšířené bitcoinové síti ukazuje nejčastější typy uzlů v rozšířené bitcoinové síti.

Rozšířená bitcoinová síť

Hlavní bitcoinová síť, běžící na P2P protokolu, se skládá ze 7 000 až 10 000 poslouchajících uzlů provozujících různé verze bitcoinového referenčního klienta (Bitcoin Core) a několika stovek uzlů provozujících různé jiné implementace bitcoinového P2P protokolu, jako "BitcoinJ library") BitcoinJ, Libbitcoin, a btcd. Malé procento uzlů v bitcoinové P2P síti jsou také těžební uzly, soupeřící v těžebním procesu, ověřování transakcí a tvorbě nových bloků. Různé velké společnosti se účastní bitcoinové sítě provozováním úplných klientů založených na Bitcoin Core klientovi, s plnými kopiemi blockchainu a uzlem sítě, ale bez těžebních nebo peněženkových funkcí. Tyto uzly slouží jako vstupní směrovače, umožňující existenci mnoha jiných služeb (směnárny, peněženky, průzkumníci bloku, zpracování obchodních plateb) na nich postavených.

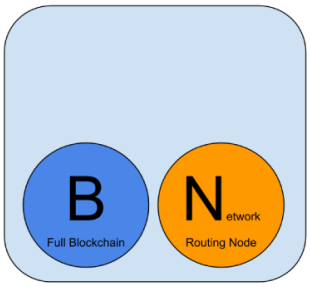
Rozšířená bitcoinová síť obsahuje síť provozující bitcoinový P2P protokol, popsaný dříve, stejně jako uzly provozující specializované protokoly. K hlavními bitcoinové P2P síti je připojeno mnoho skupinových serverů a protokolových bran, které spojují uzly provozující jiné protokoly. Tyto ostatní protokolové uzly jsou většinou uzly těžební skupiny (viz [ch8]) a odlehčení klienti peněženek, které neudržují plnou kopii blockchainu.

Rozšířená bitcoinová síť ukazující různé typy uzlů, bran a protokolů zobrazuje rozšířenou bitcoinovou síť s různými typy uzlů, serverových bran, vstupních routerů a peněženkových klientů a mnoho protokolů, které jsou použity pro vzájemné spojení.



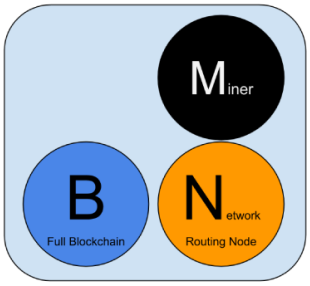
Reference Client (Bitcoin Core)

Contains a Wallet, Miner, full Blockchain database, and Network routing node on the bitcoin P2P network.



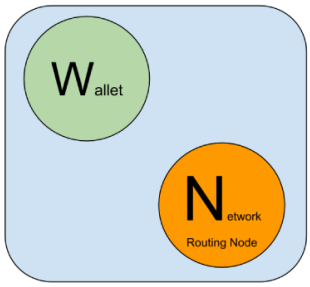
Full Block Chain Node

Contains a full Blockchain database, and Network routing node on the bitcoin P2P network.



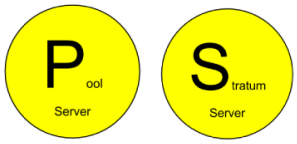
Solo Miner

Contains a mining function with a full copy of the blockchain and a bitcoin P2P network routing node.



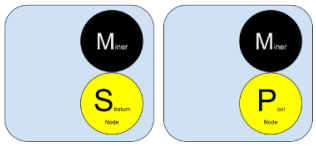
Lightweight (SPV) wallet

Contains a Wallet and a Network node on the bitcoin P2P protocol, without a blockchain.



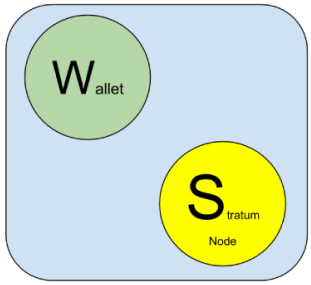
Pool Protocol Servers

Gateway routers connecting the bitcoin P2P network to nodes running other protocols such as pool mining nodes or Stratum nodes.



Mining Nodes

Contain a mining function, without a blockchain, with the Stratum protocol node (S) or other pool (P) mining protocol node.



Lightweight (SPV) Stratum wallet

Contains a Wallet and a Network node on the Stratum protocol, without a blockchain.

Figure 2. Různé typy uzlů v rozšířené bitcoinové síti

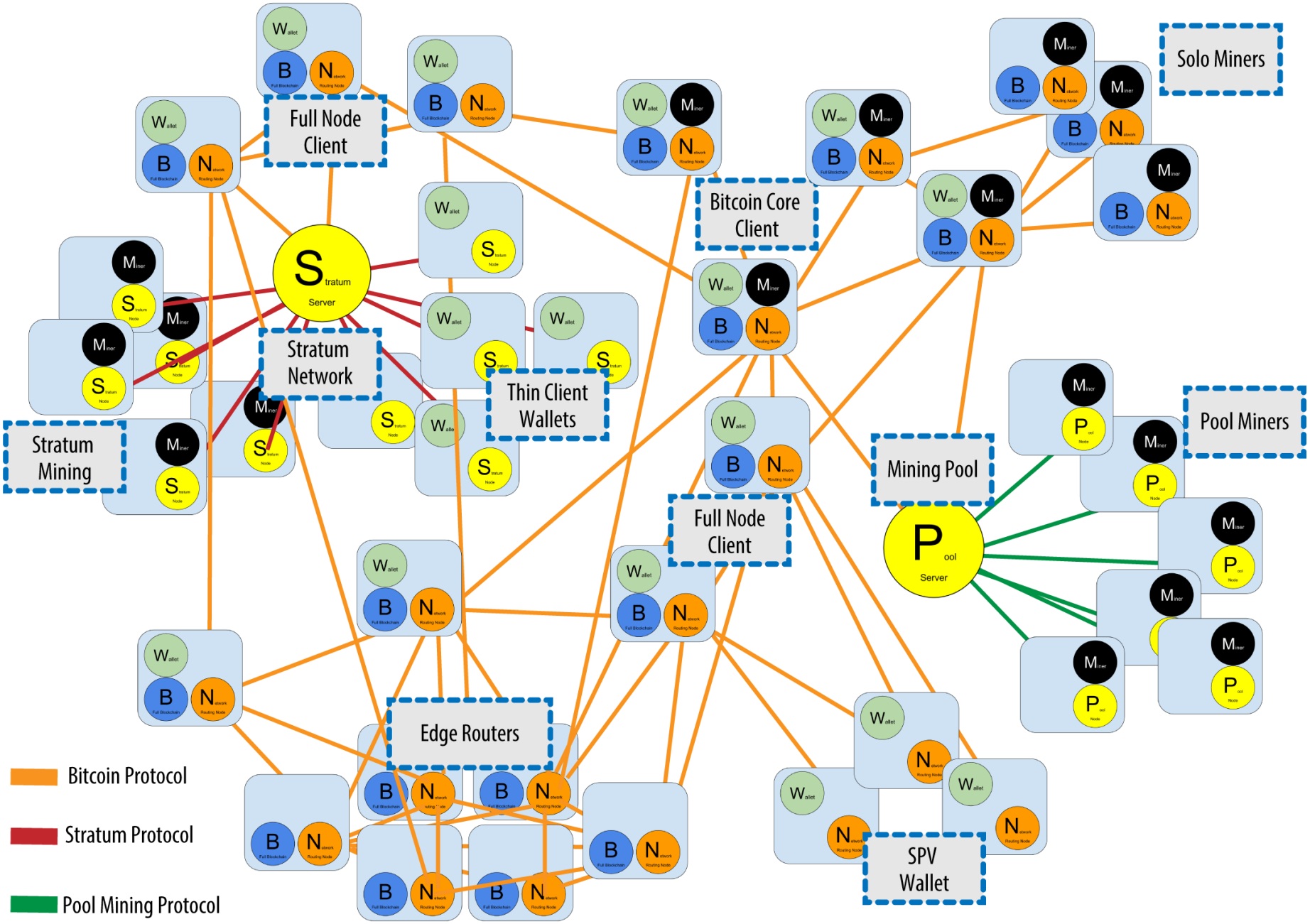

Solo Miners
Full Node Client
Bitcoin Core Client
Stratum Server
Stratum Network
Thin Client Wallets
Stratum Mining
Mining Pool
Pool Server
Pool Miners
Full Node Client
Edge Routers
SPV Wallet
Bitcoin Protocol
Stratum Protocol
Pool Mining Protocol

Figure 3. Rozšířená bitcoinová síť ukazující různé typy uzlů, bran a protokolů

Objevení sítě

Když nový uzel nastartuje, musí objevit ostatní uzly sítě, aby se mohl účastnit. Pro zahájení tohoto postupu, musí nový uzel objevit alespoň jeden existující uzel sítě a připojit se na něj. Zeměpisná poloha ostatních uzlů není důležitá, topologie bitcoinové sítě není zeměpisně určená. Proto může být náhodně vybrán jakýkoliv existující bitcoinový uzel.

Pro připojení ke známému klientskému uzlu, uzel zahájí TCP spojení, obvykle na port 8333 (tento port je všeobecně známý, že je používaný bitcoinem), nebo na alternativní port, pokud je poskytnut. Po navázání spojení, uzel začne "potřesení rukou" (viz Počáteční potřesení rukou mezi klientskými uzly) přenesením "version message") zprávy version, která obsahuje základní identifikační informace, včetně:

PROTOCOL_VERSION

Konstanta, která definuje verzi bitcoinového P2P protokolu, kterou klient "mluví" (např. 70002)

nLocalServices

Seznam místních služeb podporovaných uzlem, nyní pouze NODE_NETWORK

nTime

aktuální čas

addrYou

IP adresa vzdáleného uzlu, jak je viděna tímto uzlem

addrMe

IP adresa místního uzlu, jak je viděna místním uzlem

subver

Vedlejší verze ukazující typ software běžícího na tomto uzlu (např. "/Satoshi:0.9.2.1/")+

BestHeight

Výška (v blocích) blockchainu tohoto uzlu

(Viz GitHub pro příklad síťové zprávy version.)

Oslovený klientský uzel odpoví verack, čímž potvrzuje vytvoření spojení a volitelně zašle vlastní zprávu version, pokud si přeje, aby spojení bylo obousměrné a připojí se zpátky jako klientský uzel.

Jak nový uzel najde klientský uzel? První způsob je použít DNS dotaz za použití několika "DNS semínek", které jsou DNS servery poskytující seznam IP adres bitcoinových uzlů. Některý tyto DNS semínka poskytují pevný seznam IP adres stabilních naslouchajících bitcoinových uzlů. Některé DNS semínka jsou uživatelské implementace BIND (Berkeley Internet Name Daemon), který vrací

náhodnou podmnožinu ze seznamu adres bitcoinových uzlů sebraných vyhledávačem nebo dlouho běžícím bitcoinových uzlem. Bitcoin Core klient obsahuje jména pěti různých DNS semínek. Rozdílnost vlastnictví a rozdílnost implementací těchto různých DNS semínek nabízí vysoký stupeň spolehlivosti pro zahájení postupného načítání dat sítě. V Bitcoin Core klientovi, je možnost použití DNS semínek ovládána nastavením -dnsseed (standardně nastaveno na 1 pro použití DNS semínek).

Alternativně, startující uzel, který nezná nic ze sítě, musí mít k dispozici IP adresu alespoň jednoho bitcoinového uzlu, pomocí kterého může navázat další spojení přes další představování. Parametr -seednode může být použit k připojení k jednomu uzlu pouze pro představení, použití ho jako semínka. Poté, co je počáteční semínkový uzel použit pro představení, klient se odpojí od něj a použije nově objevené klientské uzly.

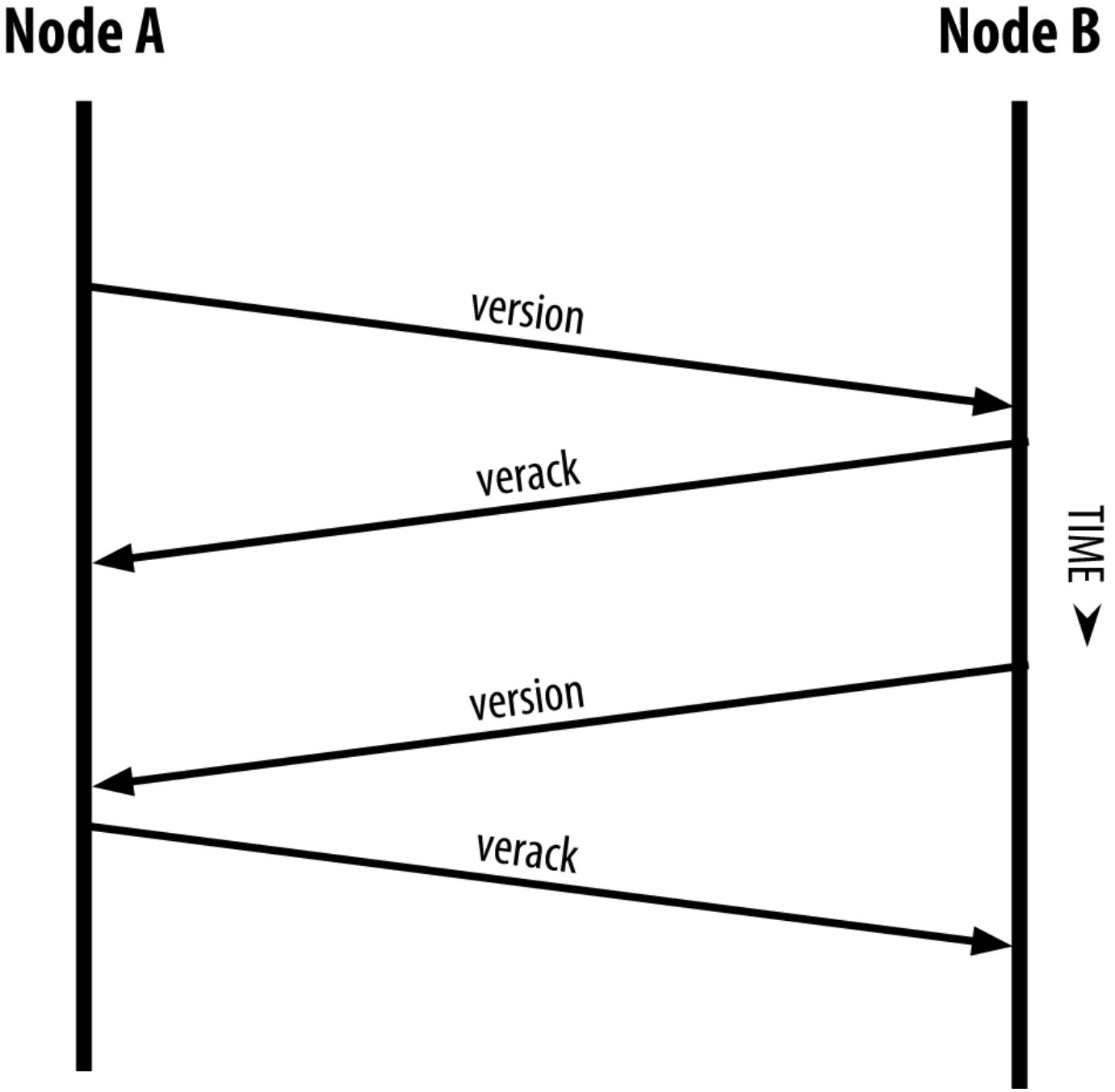


Figure 4. Počáteční potřesení rukou mezi klientskými uzly

Jakmile je jedno nebo více spojení navázáno, nový uzel bude zasílat "addr message") zprávu addr obsahující jeho vlastní IP adresu jeho sousedům. Sousedé budou postupně přeposílat zprávu addr svým sousedům, což zajišťuje, že nově připojený uzel se stává lépe známý a lépe připojený. Navíc nově připojený uzel může zaslat sousedům zprávu getaddr, žádající je o vrácení seznamu IP adres jejich klientských uzlů. Tímto způsobem uzel může najít klientské uzly a připojit se oznamovat svojí existenci v síti dalším uzlům, aby je našel. Oznamování a objevování adresy ukazuje protokol objevování adresy.

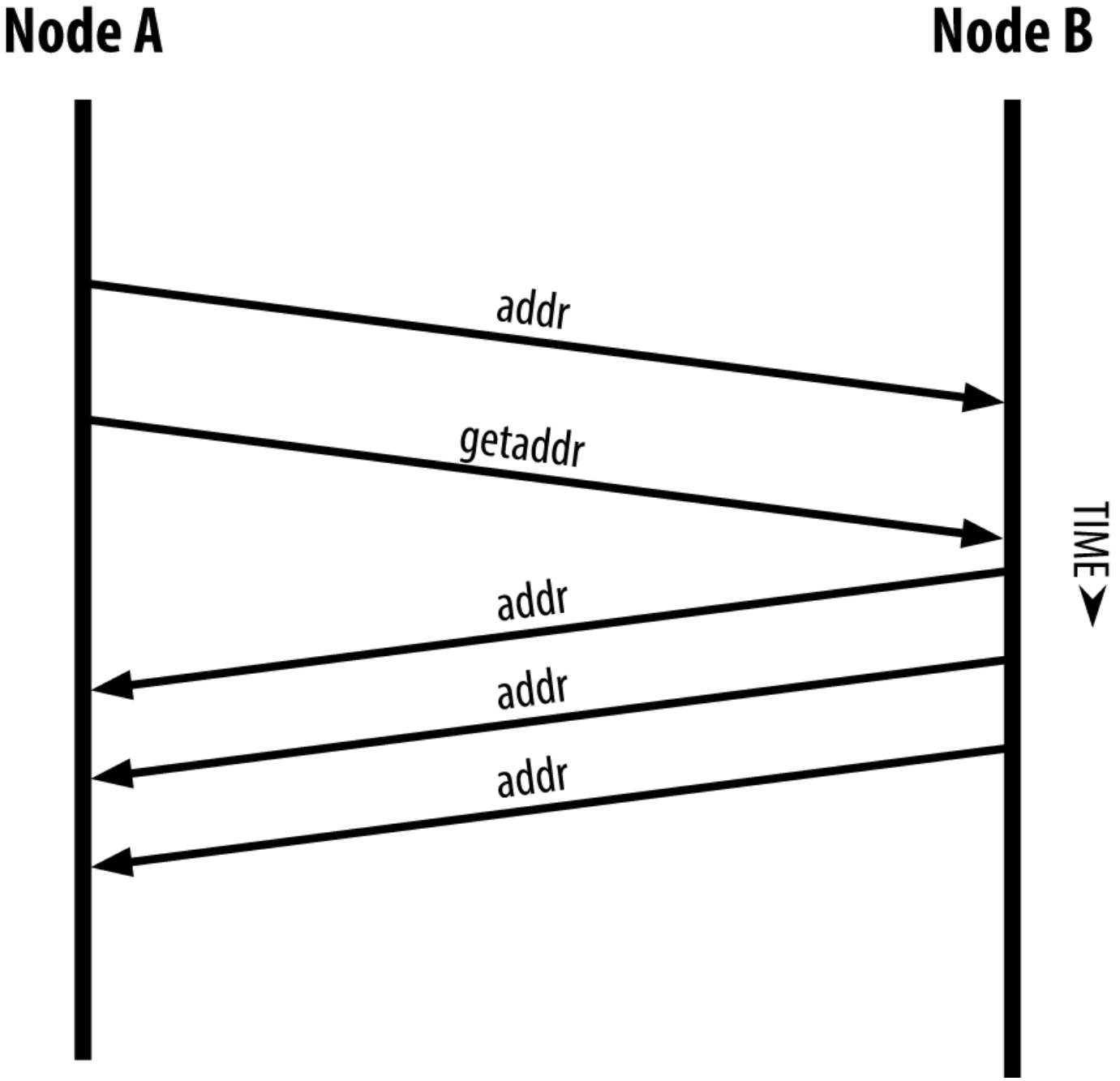


Figure 5. Oznamování a objevování adresy

Uzel musí být připojen k několika různým klientským uzlům, aby založil rozdílné cesty do bitcoinové sítě. Cesty jsou nespolehlivé, uzly mohou přicházet a odcházet, takže uzel musí pokračovat v objevování nových uzlů, když ztratí stará spojení, stejně tak pomáhá jiným uzlům při jejich inicializaci. Při inicializaci je potřeba pouze jedno připojení, protože první uzel může nabídnout představení svým

klientským uzlům a oni mohou nabídnout další představení jejich klientským uzlům. Být připojen k více než deseti uzlům je zbytečné a jedná se o plýtvání síťovými prostředky. Po inicializaci, uzel si bude pamatovat svá poslední úspěšná klientská spojení, pokud je restartován může rychle obnovit spojení se svými bývalými klientskými uzly. Pokud žádný z bývalých klientských uzlů neodpovídá na připojovací žádost, uzel může použít semínkové uzly pro zahájení opětovné inicializace.

Na uzlu provozujícím Bitcoin Core klienta, můžete vypsat seznam klientských spojení pomocí příkazu getpeerinfo:

```
$ bitcoin-cli getpeerinfo
```


```
[
    {
        "addr" : "85.213.199.39:8333",
        "services" : "00000001",
        "lastsend" : 1405634126,
        "lastrecv" : 1405634127,
        "bytessent" : 23487651,
        "bytesrecv" : 138679099,
        "conntime" : 1405021768,
        "pingtime" : 0.00000000,
        "version" : 70002,
        "subver" : "/Satoshi:0.9.2.1/",
        "inbound" : false,
        "startingheight" : 310131,
        "banscore" : 0,
        "syncnode" : true
    },
    {
        "addr" : "58.23.244.20:8333",
        "services" : "00000001",
        "lastsend" : 1405634127,
        "lastrecv" : 1405634124,
        "bytessent" : 4460918,
        "bytesrecv" : 8903575,
        "conntime" : 1405559628,
        "pingtime" : 0.00000000,
        "version" : 70001,
        "subver" : "/Satoshi:0.8.6/",
        "inbound" : false,
        "startingheight" : 311074,
        "banscore" : 0,
        "syncnode" : false
    }
]
```

Pro potlačení automatické správy klientů a určení seznamu IP adres, uživatel může poskytnout nastavení -connect=<IPAddress>+ a určit jednu nebo více IP adres. Pokud je toto nastavení použito, uzel se bude připojovat pouze na vybrané IP adresy, místo automatického objevování a udržování připojení ke klientům.

Pokud není žádný provoz se spojením, uzel periodicky zasílá zprávu, aby udržel spojení. Pokud uzel nekomunikoval se spojením více než 90 minut, je to považováno za odpojení a nový klient bude vyhledán. Proto, síť se dynamicky upravuje s dočasnými uzly a síťovými problémy a může přírodně růst a zmenšovat se jak je potřeba bez centrální kontroly.

Úplné uzly

Úplné uzly jsou uzly, které udržují úplný blockchain se všemi transakcemi. Přesněji, měly by být nazývány "úplné blockchainové uzly." V prvních letech bitcoinu, všechny uzly byly úplné uzly a v současnosti Bitcoin Core klient je úplným blockchainovým uzlem. V předchozích dvou letech nicméně nové druhy bitcoinových klientů byly představeny, které neudržují úplný blockchain, ale běží jako odlehčení klienti. Prozkoumáme je podrobněji v následující části.

Úplné blockchainové uzly spravují úplnou a aktuální kopii bitcoinového blockchainu se všemi transakcemi, které jsou nezávisle sestaveny a ověřeny, začínají prvním základním blokem (genesis blok) a staví na něm až do posledního známého bloku v síti. Úplný blockchainový uzel může nezávisle a panovačně ověřovat jakoukoliv transakci bez postihu nebo spoléhání se na nějaký jiný uzel nebo zdroj informací. Úplný blockchainový uzel se spoléhá na síť, že dostává aktualizace o nových blocích nebo transakcích, které poté ověří a zabuduje je do své místní kopie blockchainu.

Provozování úplného blockchainového uzlu přináší čistou bitcoinovou zkušenost: nezávislé ověřování všech transakcí bez nutnosti spoléhat se nebo věřit jinému systému. Je to lehké říci, ale pokud provozuje úplný uzel potřebujete 20+ gigabytů diskového úložiště pro uložení úplného blockchainu a dva až tři dny na synchronizaci se sítí. Je to cena za úplnou nezávislost a svobodu od centrální autority.

Existuje několik alternativních implementací úplných blockchainových bitcoinových klientů, postavených za použití různých programovacích jazyků a softwarových architektur. Nicméně, nejrozšířenější implementace je referenční klient Bitcoin Core, také známý jako Satoshi klient. Více než 90 % uzlů v bitcoinové síti provozují různé verze Bitcoinového klienta. Jsou identifikovány jako "Satoshi" a řetězcem vedlejší verze zaslaným zprávou version a zobrazeným příkazem getpeerinfo; jak jsme viděli dříve; například /Satoshi:0.8.6/.

Výměna "zásob dat"

Úplný uzel po svém připojení ke klientskému uzlu se nejprve pokusí sestavit kompletní blockchain. Pokud se jedná o zcela nový uzel a nemá žádný blockchain, zná pouze první základní blok, který je napevno zabudován v software klienta. Začínáme s blokem #0 (základní blok), nový uzel musí stáhnout stovky tisíc bloků, aby se synchronizoval se sítí a obnovil úplný blockchain.

Postup synchronizace blockchainu začíná zprávou version, protože ta obsahuje BestHeight udávající

aktuální výšku blockchainu uzlu (počet bloků). Uzel uvidí zprávu version od svých klientských uzlů, čímž se dozví, kolik bloků oni mají a je schopen porovnat kolik bloků má ve vlastním blockchainu. Klientské uzly budou vyměňovat zpráva getblocks která obsahuje haš (otisk) vrchního bloku v jejich místním blockchainu. Jeden z klientských uzlů bude schopen identifikovat obdržený haš, který patří bloku, který není na vrcholu, ale patří staršímu bloku, čímž odvodí, že jeho vlastní místní blockchain je delší než blockchain jeho klientského uzlu.

Klientský uzel, který má delší blockchain, má více bloků než ostatní uzly a může určit, které bloky ostatní uzly potřebují, aby ho "dostihly." Identifikuje prvních 500 bloků, aby je sdílel a přenášel jejich haše za použití zprávy inv (inventory, česky zásoba dat). Uzel, kterému tyto bloky chybí, je získá vysláním posloupnosti zpráv getdata požadujících úplná data bloku a identifikující požadované bloky za použití hašů ze zprávy inv.

Předpokládejme, například, že uzel má pouze základní blok. Obdrží zprávu inv od svého klientského uzlu obsahující haše dalších 500 bloků v řetězu. Začne požadovat bloky od všech svých klientských uzlů. rozděluje tím zátěž, aby nezahltil jeden klientský uzel svými požadavky. Uzly uchovávají kolik bloků "se přenáší" pro jednotlivá klientská spojení, čímž jsou myšleny bloky, o které bylo požádáno, ale nebyly obdrženy, kontrolují, aby nedošlo k překročení omezení (MAX_BLOCKS_IN_TRANSIT_PER_PEER). Tímto způsobem, pokud potřebujete mnoho bloků, požádáte o nové až po splnění předchozích požadavků, umožníte klientským uzlům kontrolovat rychlost aktualizace a nezahlcujete síť. Jakmile je blok obdržen, je přidán na blockchain, jak vidíme v [blockchain]. Jak se místní blockchain postupně rozrůstá, více bloků je požadováno a obdrženo, postup pokračuje dokud uzel nedostihne zbytek sítě.

Postup porovnání místního blockchainu s klientskými uzly a obdržení chybějících bloků nastává kdykoliv uzel jde offline na nějaký čas. Jakmile uzel byl offline několik minut a chybí několik bloků nebo měsíc a chybí pár tisíc bloků, začíná se zasláním getblocks a obdržením odpovědi inv a započetím stahování chybějících bloků. Synchronizace uzlů blockchainu získáním bloků od klientského uzlu ukazuje zásobu dat a protokol šíření bloku.

Zjednodušené ověřování plateb SPV uzly

Ne všechny uzly mají schopnost uložit úplný blockchain. Mnoho bitcoinových klientů je navrženo pro provozování na zařízeních s omezeným úložištěm a výkonem, jako jsou chytré telefony, tablety nebo jednoduchá vestavěná zařízení. Pro tato zařízení se používá metoda *zjednodušené ověřování plateb* (v originále simplified payment verification, zkratka SPV), které umožňuje pracovat bez uloženého úplného blockchainu. Tyto typy klientů jsou zvané SPV klienti nebo odlehčení klienti. Se vzrůstajícím rozšířením bitcoinu, SPV uzly se stávají nejčastějším druhem bitcoinových uzlů, obzvláště pro bitcoinové peněženky.

SPV uzly stahují pouze hlavičky bloků a nestahují transakce obsažené v každém bloku. Výsledný řetěz bloků, bez transakcí je 1 000 krát menší než úplný blockchain. SPV uzly nemohou sestavit úplný obrázek všech UTXO, které jsou dostupné k utracení, protože neznají všechny transakce na síti. SPV uzly ověřují transakce za použití trochu odlišné metodiky, která spoléhá na klientské uzly poskytující částečný pohled na příslušnou část blockchainu na požádání.

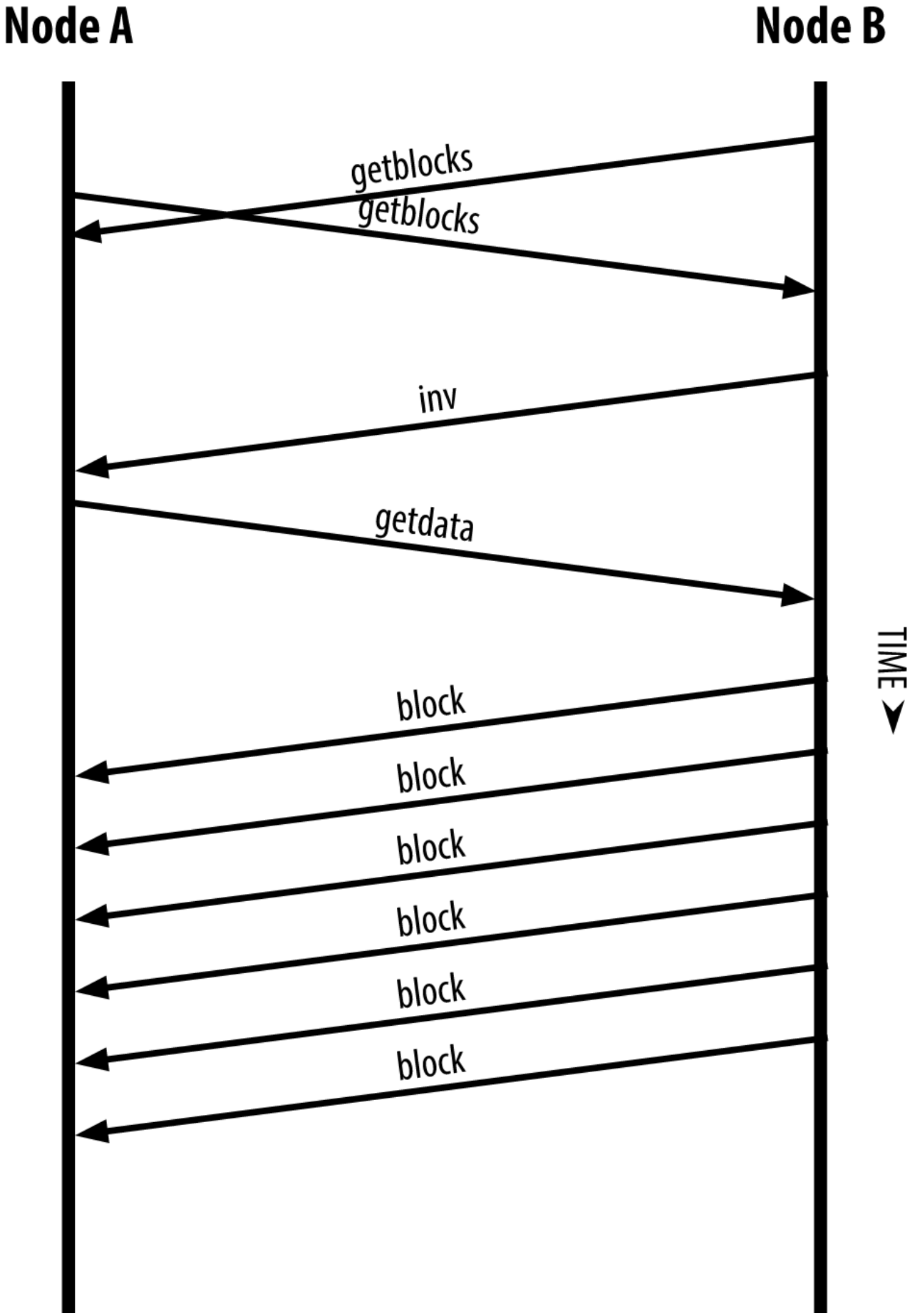

Node A
Node B
getblocks
getblocks
inv
getdata
block
block
block
block
block
block
TIME

Figure 6. Synchronizace uzlů blockchainu získáním bloků od klientského uzlu

Jako podobenství, úplný uzel je jako turista v cizím městě, vybaven podrobnou mapou každé ulice a a každé adresy. Pro srovnání, SPV uzel je jako turista v cizím městě ptající se náhodných kolemjdoucích na cestu krok za krokem, zatímco zná pouze jednu hlavní třídu. Přestože oba turisté mohou ověřit existenci ulici její návštěvou, turista bez mapy neví, co leží v bočních ulicích a jaké jiné ulice existují. Stojící před Kostelní 23, turista bez mapy neví, zda zde není dalších deset jiných "Kostelních 23" adres ve městě a jestli je na správné z nich. Nejlepší možností turisty bez mapy je zeptat se dostatečného množství lidí a doufat, že někteří z nich se ho nebudou nesnažit přepadnout.

Zjednodušené ověřování plateb ověřuje transakce odkazem a jejich *hloubku* v blockchainu místo jejich *výšky*. Zatímco úplný blockchainový uzel sestaví celý ověřený řetěz tisíců bloků a transakcí, který dosáhne dolů skrz blockchain (zpátky v čase) celou cestu k základnímu bloku, SPV uzel ověří řetěz všech bloků (ale ne všech transakcí) a spojí řetěz transakcí, které ho zajímají.

Například, když prozkoumáme transakce v bloku 300 000, úplný uzel spojí všech 300 000 bloků dolů k základnímu bloku a vytvoří úplnou databázi UTXO, zajišťující platnost transakce potvrzením, že UTXO zůstalo neutraceno. SPV uzel nemůže ověřit, zda UTXO je utraceno, místo toho, SPV uzel stanoví odkaz mezi transakcí a blokem, který ji obsahuje, za použití *merkle cesty* (viz [merkle_trees]). Poté SPV uzel čeká dokud neuvidí šest bloků od 300 001 do 300 006 navršených na vrchol bloku obsahující transakci a ověří ji stanovením její hloubky pod bloky 300 006 do 300 001. Skutečnost, že ostatní uzly sítě akceptovali blok 300 000 a poté provedli nezbytnou práci pro vytvoření šesti dalších bloků na jejím vrcholu, je důkazem, zprostředkovaným, že transakce není dvojitě utracena.

SPV nemůže být přesvědčen, že transakce existuje v bloku, když transakce ve skutečnosti neexistuje. SPV uzel stanoví existenci transakce v bloku požadováním důkazu merkle cesty a ověřením důkazem prací v řetězu bloků. Nicméně existence transakce může být skrytá SPV uzlu. SPV uzel může jednoznačně dokázat, že transakce existuje, ale nemůže ověřit, že transakce, jako dvojité utracení toho samého UTXO neexistuje, protože nemá záznam všech transakcí. Tato zranitelnost může být použita při útoku odepření služby nebo při útoku dvojitého utracení proti SPV uzlu. Jako obrana proti tomuto, SPV uzel potřebuje být připojen náhodně k několika uzlům, aby zvýšil pravděpodobnost, že je v kontaktu s alespoň jedním čestným uzlem. Potřeba náhodného připojení znamená, že SPV uzly jsou také zranitelné útoky oddělení sítě nebo Sybil útoky, při kterých jsou spojeny s falešnými uzly nebo falešnou sítí a nemají přístup k čestným uzlům nebo skutečné bitcoinové síti.

Pro většinu praktických účelů, dobře propojené SPV uzly jsou dostatečně zabezpečené, nacházejí správnou rovnováhu mezi potřebou zdrojů, praktičností a bezpečností. Pro neomylnou bezpečnost, nicméně, nic nemůže porazit provozování úplného blockchainového uzlu.

TIP Úplný blockchainový uzel ověřuje transakce kontrolou celého řetězu tisíců bloků pod za účelem garance, že UTXO není utraceno, zatímco SPV uzel kontroluje jak hluboko je blok pohřben několika bloky nad ním.

Pro získání hlaviček bloků, SPV uzly používají zprávu getheaders místo getblocks. Dotázaný klientský uzel zašle až 2000 hlaviček bloků za použití jedné zprávy headers. Postup je jinak stejný, jako byl použit úplnými uzly pro získání úplných bloků. SPV uzly mohou nastavit filter na spojení s klientskými uzly,

aby odstranili proud budoucích bloků a transakcí zasílaných klientskými uzly. Jakákoliv transakce jejich zájmu je získána pomocí žádosti getdata. Klientské uzly vytvářejí jako odpověď zprávu tx obsahující transakce.SPV uzel synchronizující hlavičky bloků ukazuje synchronizaci hlaviček bloků.

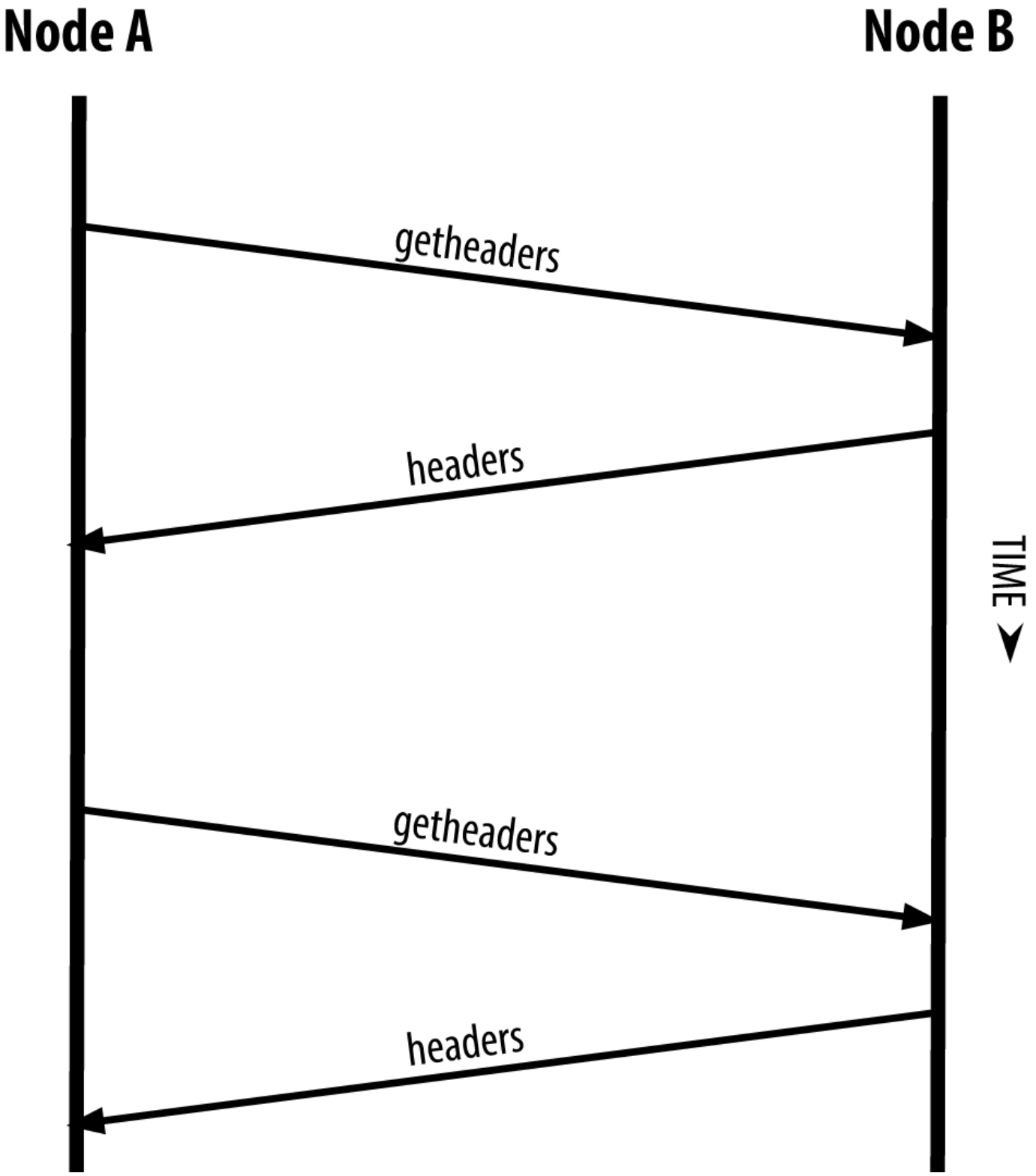


Figure 7. SPV uzel synchronizující hlavičky bloků

Protože SPV uzly potřebují získat konkrétní transakce, aby je jednotlivě ověřili, vytváří také riziko ohrožení soukromí. Na rozdíl od úplných blockchainových uzlů, které sbírají všechny transakce všech

bloků, SPV uzly požadují konkrétní data, čímž mohou neúmyslně odhalit adresy jejich peněženky. Například, třetí strana monitorující síť může uchovávat stopu všech transakcí požadovaných peněženkou na SPV uzlu a použít je pro spojení bitcoinových adres s uživatelem peněženky, zničit uživatelovo soukromí.

Krátce po představení SPV/odlehčených uzlů, bitcoinoví vývojáři přidali funkci zvanou *Bloomův filtr* pro vypořádání se s rizikem ohrožení soukromí SPV uzlů. Bloomovy filtry umožňují SPV uzlům získat podmnožinu transakcí bez odhalení o jakou přesnou adresu se zajímají, pomocí mechanizmu používajícího pravděpodobnosti místo pevných vzorů.

Bloomovy filtry

Bloomův filtr je pravděpodobnostní vyhledávací filtr, způsob popsání požadovaného vzoru bez určení jaký je přesně. Bloomovy filtry nabízejí účinný způsob popsání hledaného vzoru při ochránění soukromí. Jsou použity SPV uzly, aby se ptali svých klientských uzlů na transakce splňující daný vzor, bez prozrazení jaké adresy hledají.

V našem předchozím podobenství, turista bez mapy se ptal na směr ke konkrétní adrese "Kostelní 23". Pokud se zeptal kolemjdoucího na směr k této ulici, neúmyslně odhalil svůj cíl cesty. Bloomův filtr je jako se ptát "Je zde nějaká ulice v sousedství, jejíž jméno končí na L-N-Í?" Tato otázka prozrazuje trochu méně o požadovaném cíli cesty než zeptání se na "Kostelní 23." Použitím této techniky, může turista určit požadovanou adresu přesněji "konči na E-L-N-Í" nebo méně detailněji jako "končící na Í." Změnou přesnosti hledání, turista odhaluje více nebo méně informací za cenu získání více nebo méně konkrétního výsledku. Pokud požádá o méně konkrétní vzor, dostane více možných adres a lepší soukromí, ale mnoho výsledků je neužitečných. Pokud se zeptá na velmi konkrétní vzorek, dostane méně odpovědí, ale ztrácí soukromí.

Bloomovy filtry slouží této funkci, umožňují SPV uzlům určit hledaný vzor transakcí, tak aby se mohly rozhodovat mezi potřebou větší přesnosti nebo většího soukromí. Konkrétnější Bloomův filtr vyrobí přesnější výsledky, ale za cenu odhalení jaké adresy se používají v uživatelově peněžence. Méně konkrétní Bloomův filtr vyrobí více dat o více transakcích, mnoho nesouvisejících s uzlem, ale umožňuje uzlu zachovat lepší soukromí.

Uzel SPV nastaví Bloomův filtr jako "prázdný" a v tomto stavu Bloomův filtr nebude odpovídat žádnému vzoru. SPV uzel poté udělá seznam všech adres v jeho peněžence a vytvoří takový vzorek hledání, který odpovídá transakčních výstupům odpovídajícím jednotlivým adresám. Obvykle je hledací vzorek skript platby haši veřejného klíče, což je očekávaný zamykací skript, který bude přítomný v jakékoliv transakci platící haši veřejného klíče (adrese). Pokud SPV uzel sleduje stavy P2SH adres, hledaný vzor bude místo toho skript platby haši skriptu. SPV uzel přidá každý z hledaných vzorů do Bloomova filtru, takže Bloomův filtr může rozpoznat hledané vzory pokud jsou přítomny v transakcích. Nakonec je Bloomův filtr zaslán klientskému uzlu a klientský uzel jej použije pro najití transakcí pro přenos do SPV uzlu.

Bloomovy filtry jsou implementovány jako pole proměnlivé délky mající N binárních číslic (bitové pole) a proměnlivý počet M hašovacích funkcí. Hašovací funkce jsou navrženy, aby vytvářeli vždy

výstup mezi 1 až N, odpovídající poli binárních číslic. Hašovací funkce jsou vytvářeny deterministicky, takže každý uzel implementující Bloomův filtr bude vždy používat stejné hašovací funkce, aby získal stejné výsledky pro konkrétní vstup. Výběrem různé délky (N) Bloomových filtrů a různého počtu (M) hašovacích funkcí může být Bloomův filtr laděn, lišící se stupně přesnosti a tedy i soukromí.

V Příklad zjednodušeného Bloomova filtru se 16-bitovým polem a třemi hašovacími funkcemi použijeme velmi malé pole 16 bitů a množinu tří hašovacích funkcí pro ukázku jak Bloomovy filtry pracují.

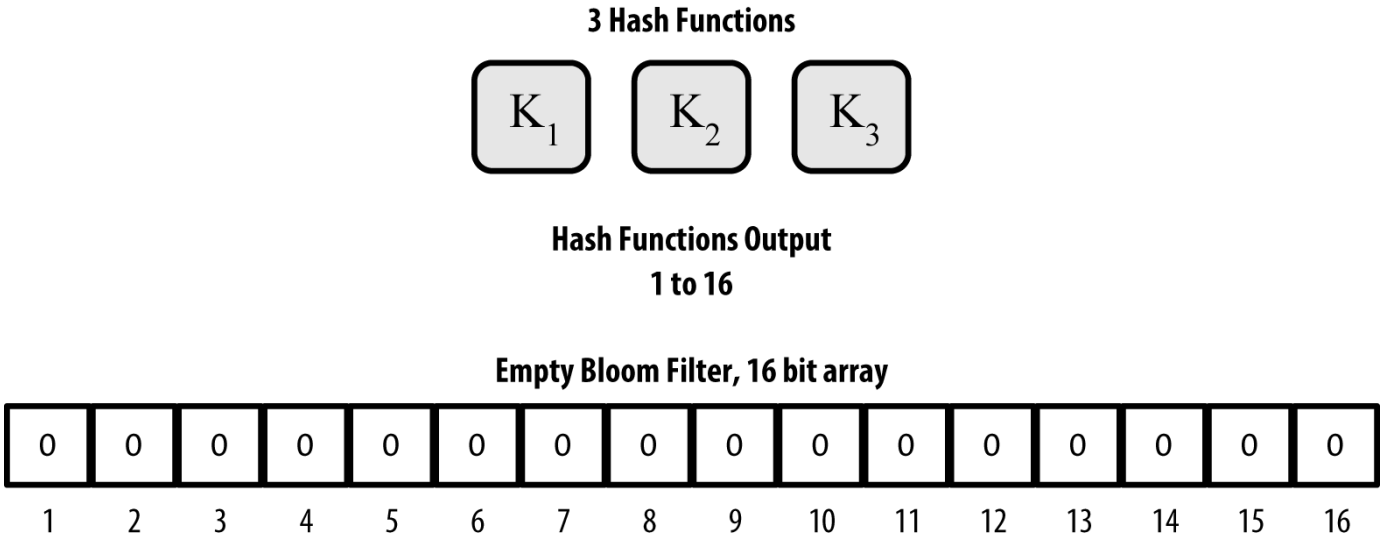


Figure 8. Příklad zjednodušeného Bloomova filtru se 16-bitovým polem a třemi hašovacími funkcemi

Bloomův filtr je inicializován, takže pole bitů jsou samé nuly. Pro přidání vzoru do Bloomova filtru jsou postupně brány jednotlivé hašovací funkce, každou je spočítán haš vzoru. Aplikací první hašovací funkce na vstup získáme číslo mezi 1 až N. Odpovídající bit v poli (indexovaném od 1 do N) je nalezen a nastaven na 1, tímto se zaznamená výstup hašovací funkce. Následně další hašovací funkce je použita pro nastavení dalšího bit, atd. Jakmile je všech M hašovacích funkcí aplikováno, hledaný vzorek je zaznamenán v Bloomově filtru jako M bitů, které se změnily z 0 na 1.

Přidání vzoru "A" do našeho jednoduchého Bloomova filtru je příklad přidání vzoru "A" do jednoduchého Bloomova filtru zobrazeného v Příklad zjednodušeného Bloomova filtru se 16-bitovým polem a třemi hašovacími funkcemi.

Přidání druhého vzoru je tak jednoduché jako zopakovat tento postup. Vzor je hašován každou hašovací funkcí v radě a výsledek je zaznamenán nastavení bitů na 1. Všimněte si, když je Bloomův filtr naplněn více vzorky, výsledky hašovací funkce se mohou shodovat s bity, které jsou skutečně nastavená na 1, v tomto případě ke změně bitu nedojde. V podstatě, čím více vzorů je zaznamenáno do shodných bitů, tím se Bloomův filtr začíná stávat nasycený s více bity nastavenými na 1 a přesnost filtrování klesá. To je důvod, proč filtr je pravděpodobnostní datová struktura, dává menší přesnost, čím více vzorů je přidáno. Přesnost závisí na počtu přidaných vzorů oproti velikosti bitového pole (N) a počtu hašovacích funkcí (M). Větší bitové pole a více hašovacích funkcí může zaznamenat více vzorů s vyšší přesností. Menší bitové pole nebo méně hašovacích funkcí zaznamená méně vzorů a vyrobí menší přesnost.

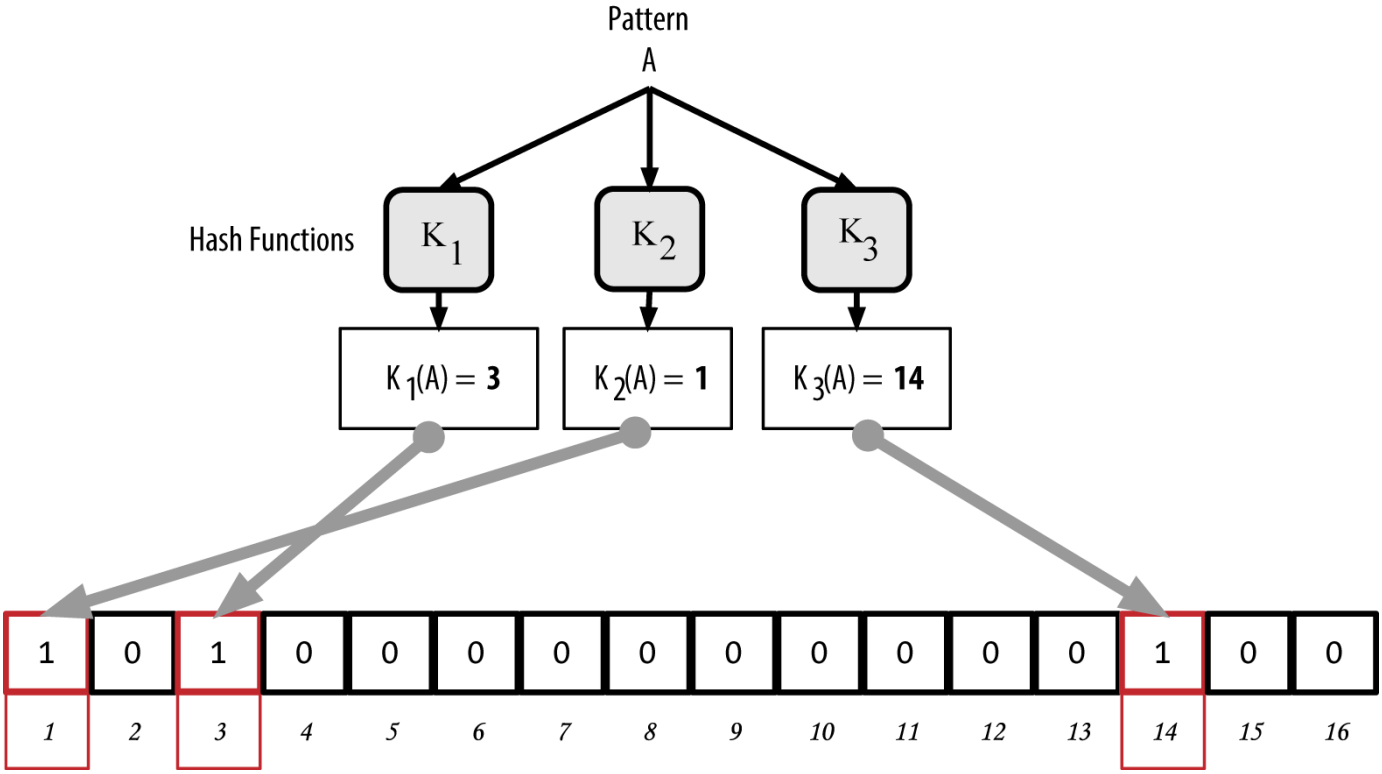


Figure 9. Přidání vzoru "A" do našeho jednoduchého Bloomova filtru

Přidání druhého vzoru "B" do našeho jednoduchého Bloomova filtru je příklad přidání druhého vzoru "B" do jednoduchého Bloomova filtru.

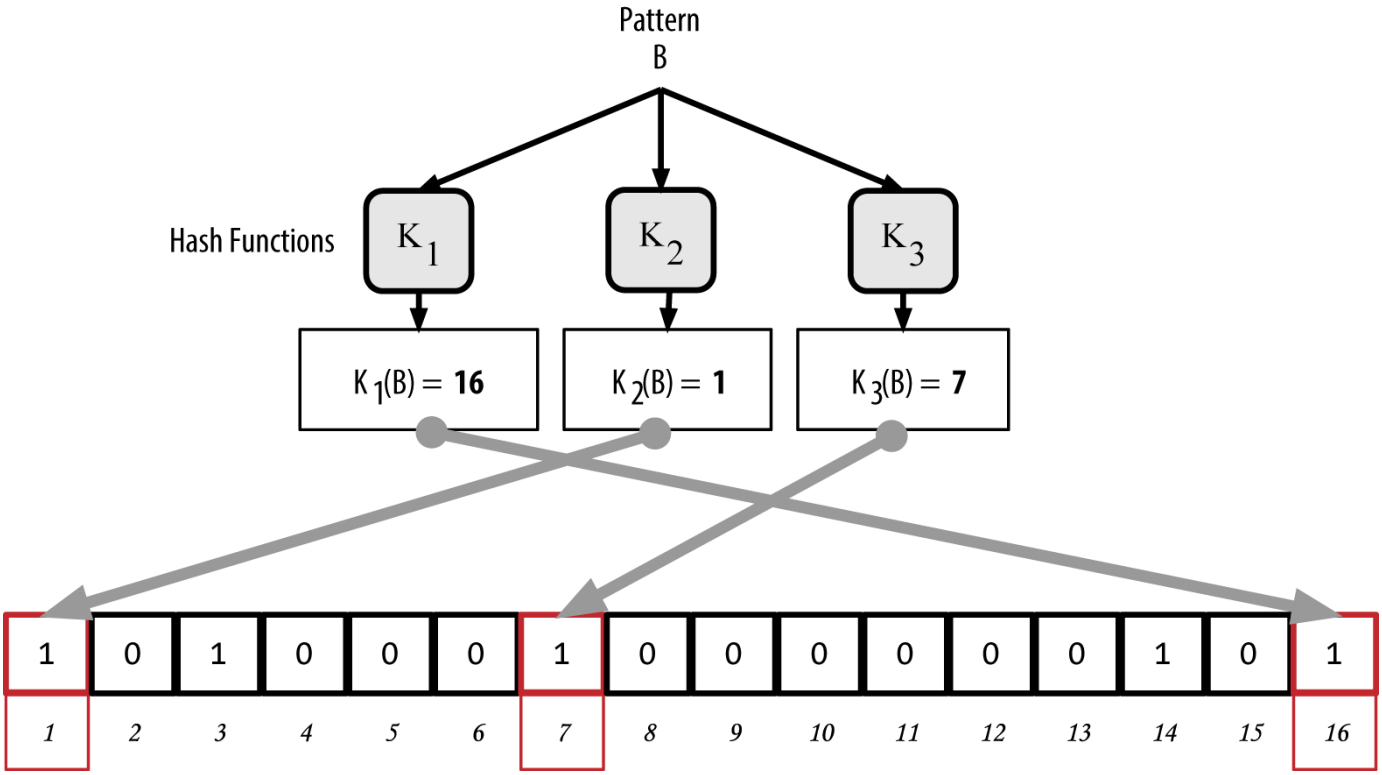


Figure 10. Přidání druhého vzoru "B" do našeho jednoduchého Bloomova filtru

Pro otestování, zda vzor je částí Bloomova filtru, vzor je hašován každou hašovací funkcí a výsledný

bitový vzor je testován oproti bitovému poli. Pokud všechny bity indexované výsledky hašovacích funkcí jsou nastaveny na 1, poté je vzor *pravděpodobně* zaznamenán v Bloomově filtru. Protože bity mohou být nastaveny kvůli překryvu z více vzorů, odpověď není jistá, ale je spíše pravděpodobnostní. Jednoduše řečeno, Bloomův filtr, vracející kladnou shodu znamená "Možná, ano."

Testování existence vzoru "X" v Bloomově filtru. Výsledek je pravděpodobnostně kladná shoda, znamenající "Možná." je příkladem testování existence vzoru "X" v jednoduchém Bloomově filtru. Odpovídající bity jsou nastaveny na 1, takže vzor je pravděpodobnou shodou.

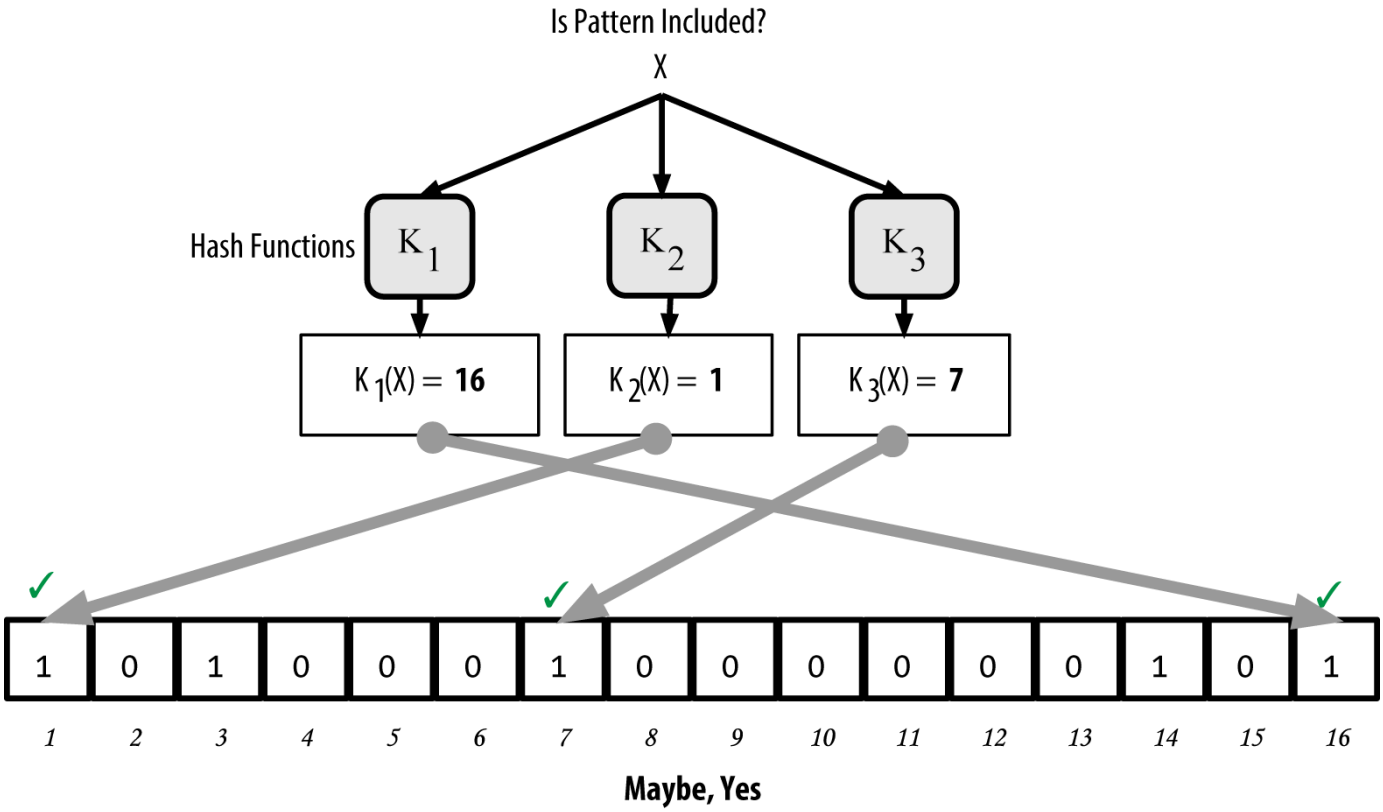


Figure 11. Testování existence vzoru "X" v Bloomově filtru. Výsledek je pravděpodobnostně kladná shoda, znamenající "Možná."

Naopak v případě, že vzor je testován oproti Bloomově filtru a jeden z bitů je nastaven na 0, dokazuje to, že vzor není zaznamenán Bloomovým filtrem. Záporný výsledek není pravděpodobný, je jistý. Jednoduše řečeno, záporná shoda v Bloomově filtru je "Jednoznačné ne!"

Testování existence vzoru "Y" v Bloomově filtru. Výsledek je jednoznačnou zápornou shodou, znamenající "Jednoznačné ne!" je příkladem testování existence vzoru "Y" v jednoduchém Bloomově filtru. Jeden z odpovídajících bitů je nastaven na 0, takže vzor se definitivně neshoduje.

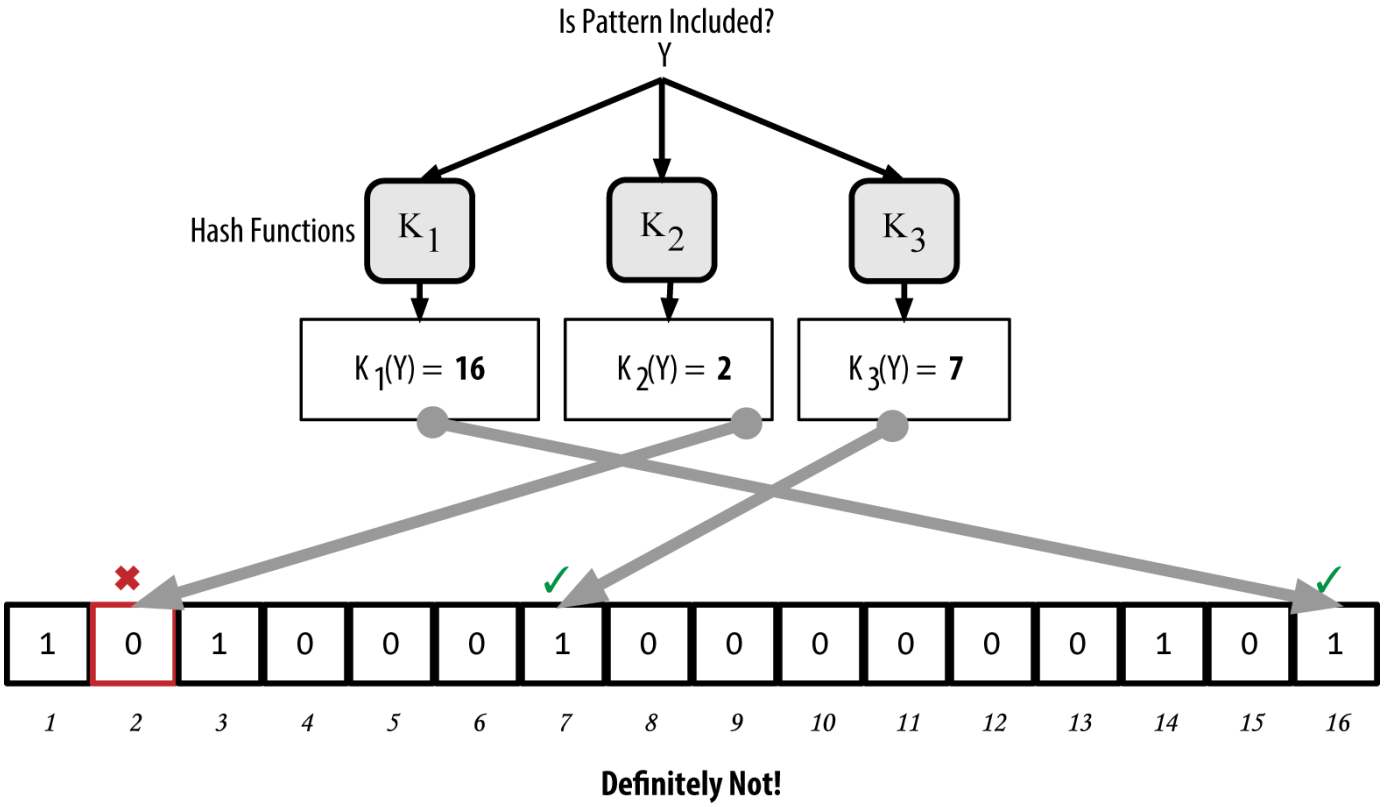


Figure 12. Testování existence vzoru "Y" v Bloomově filtru. Výsledek je jednoznačnou zápornou shodou, znamenající "Jednoznačné ne!"

Bitcoinová implementace Bloomových filtrů je popsána v návrhu na vylepšení bitcoinu 37 (BIP0037). Viz [appdxbitcoinimpproposals] nebo navštivte GitHub.

Bloomovy filtry a aktualizace zásob dat

Bloomovy filtry jsou použity pro filtrování transakcí (a bloků, které je obsahují), které SPV uzel přijme od svých klientských uzlů. SPV uzly vytvoří filtr, který splňují pouze adresy držené v peněžence SPV uzlu. SPV uzel poté zašle zprávu filterload klientskému uzlu, obsahující Bloomův filtr k použití na připojení. Poté, co je filtr stanovený, klientský uzel jím otestuje všechny výstupy transakcí. Pouze transakce splňující filtr jsou vráceny uzly.

V odpovědi na zprávu getdata z uzlu, klientské uzly zašlou zprávu merkleblock obsahující pouze hlavičky bloků splňujících filtr a merkle cestu (viz [merkle_trees]) pro každou odpovídající transakci. Klientský uzel také zašle zprávu tx obsahující transakce odpovídající filtru.

Uzlové nastavení Bloomova filtru může průběžně přidávat vzory do filtru jejich zasláním zprávou filteradd. Pro vyčištění Bloomova filtru uzel může zaslat zprávu filterclear. Protože není možné odstranit vzor z Bloomova filtru, uzel musí vyčistit a znovu naplnit Bloomův filtr, pokud vzor již není požadován.

Úložiště transakcí

Téměř všechny uzly v bitcoinové síti udržují seznam nepotvrzených transakcí zvaný *paměťové úložiště* (v originále memory pool nebo mempool) nebo _ úložiště transakcí_ (v originále transaction pool). Uzly používají toto úložiště pro sledování transakcí, které jsou známy síti, ale ještě nebyly zahrnuty do blockchainu. Například, uzel, který drží uživatelovu peněženku bude používat úložiště transakcí pro sledování příchozích plateb do uživatelovy peněženky, které byly obdrženy sítí, ale ještě nebyly potvrzeny.

Jak jsou transakce přijímány a ověřovány, jsou přidávány do úložiště transakcí a jsou zasílány sousedním uzlům, aby byly šířeny po síti.

Některé implementace také udržují oddělené úložiště osiřelých transakcí. Pokud transakční vstup odkazuje na transakci, která dosud není známa, takové chybí rodič, osiřelá transakce bude dočasně uložena v úložišti osiřelých transakcí dokud rodičovská transakce nepřijde.

Když je transakce přidána do úložiště transakcí, je zkontrolováno úložiště osiřelých transakcí, zda neobsahuje sirotka odkazujícího se na transakční výstupu (její dítě). Jakýkoliv odpovídající sirotek je poté ověřen. Pokud je platný, je odstraněn ze úložiště osiřelých transakcí a je přidán do úložiště transakcí, čímž je dokončen řetěz začínající u rodičovské transakce. Ve světle nově přidané transakce, která jíž není sirotkem, je postup rekurzivně zopakován. hledáme další potomky, dokud žádní potomci nejsou nalezeni. Tímto postupem příchod rodičovské transakce může spustit vodopád rekonstrukcí celého řetězu vzájemně závislých transakcí slučování sirotků s jejich rodiči celou cestou řetězem.

Jak úložiště transakcí tak úložiště osiřelých transakcí (kde je implementováno) jsou uloženy v místní paměti a nejsou uloženy na trvalém úložišti; spíše; jsou dynamicky naplňovány z příchozích zpráv sítě. Při startu uzlu jsou obě úložiště prázdná a jsou postupně naplňována s obdržením nových transakcí na síti.

Některé implementace bitcoinových klientů také udržují databázi nebo úložiště UTXO, což je množina všech neutracených výstupů na blockchainu. Přestože název "úložiště UTXO" zní podobně jako "úložiště transakcí", reprezentuje odlišnou množinu dat. Na rozdíl od úložiště transakcí nebo sirotků, úložiště UTXO není inicializováno prázdné, ale obsahuje miliony záznamů neutracených transakčních výstupů, včetně dat až z roku 2009. úložiště UTXO může být umístěno v místní paměti nebo být indexováno v databázové tabulce trvalého úložiště.

Zatímco úložiště transakcí a sirotků zobrazují pohled konkrétního uzlu a mohou se lišit značně mezi jednotlivými uzly, v závislosti na tom, kdy uzel byl spuštěn nebo restartován, úložiště UTXO reprezentuje vznikající shodu sítě a proto se bude lišit velmi málo mezi uzly. Dokonce, úložiště transakcí a sirotků pouze obsahují nepotvrzené transakce, zatímco úložiště UTXO obsahuje pouze potvrzené výstupy.

Výstražné zprávy

Výstražné zprávy jsou zřídkakdy používané funkce, ale jsou přesto implementovány ve většině uzlů.

Výstražné zprávy jsou bitcoinovým "systémem nouzového vysílání", způsobem, kterým vývojáři jádra bitcoinu mohou posílat nouzové zprávy všem bitcoinových uzlům. Tato funkce byla implementována, aby umožnila vývojářskému týmu informovat všechny bitcoinové uživatele o vážných problémech sítě, jako kritických chybách, které vyžadují uživatelovu akci. Výstražný systém byl použít pouze několikrát, zejména na začátku roku 2013, když kritická chyba způsobila víceblokové rozvětvení bitcoinového blockchainu.

Výstražné zprávy jsou šířeny zprávou alert obsahující několik položek, včetně:

ID

Identifikaci výstrahy, aby mohly být odhaleny zdvojené výstrahy.

Expiration

Čas po kterém výstraha přestane platit

RelayUntil

Čas, po kterém by výstraha neměla být přenášena

MinVer, MaxVer

Rozpětí verzí bicoinového protokolu, na které se výstraha vztahuje

subVer

Verze software klienta, na kterou se tato výstraha vstahuje

Priority

Stupeň závažnosti výstrahy, aktuálně se nepoužívá

Výstrahy jsou kryptograficky podepsány veřejným klíčem. Odpovídající soukromý klíč je držen několika vybranými členy vývojářského týmu jádra. Digitální podpis zajišťuje, že falešné výstrahy nebudou šířeny sítí.

Každý uzel, který obdrží tuto výstražnou zprávu, ji ověří, zkontroluje dobu platnost a rozšíří ji všem svým klientským uzlům, což zajistí rychlé rozšíření do celé sítě. Navíc k propagací výstrahy mohou uzly implementovat funkci uživatelského rozhraní předávající výstrahu uživateli.

V Bitcoin Core klientovi je výstraha konfigurována nastavením příkazové řádky -alertnotify, které určuje příkaz, který se má spustit. když je výstraha obdržena. Výstražná zpráva je předána jako parametr příkazu alertnotify, Nejčastěji, příkaz alertnotify je nastaven, aby vytvořil emailovou zprávu administrátorovi uzlu, obsahující výstražnou zprávu. Výstraha je rovněž zobrazena ve vyskakovacím okně v grafickém uživatelském rozhraní (bitcoin-Qt), pokud běží.

Jiné implementace bitcoinového protokolu mohou zpracovávat výstrahy jinými způsoby. Mnoho vestavěných hardwarových zařízení bitcoinového těžebního systému neimplementují funkci výstražné zprávy protože nemají uživatelské rozhraní, Je důrazně doporučováno, aby tito těžaři provozující tento těžební systém se přihlásili k odběru výstrah prostřednictvím provozovatele těžební skupiny nebo provozováním odlehčeného uzlu pouze pro účely výstrah.

Blockchain

Úvod

Datová struktura blockchain je uspořádány spojový seznam bloků transakcí. Blockchain může být uložen v jednom plochém souboru nebo v jednoduché databázi. Bitcoin Core klient ukládá metadata blockchainu za použití databáze Google's LevelDB. Bloky jsou spojeny "zpět", každý odkazuje na předchozí blok v řetězu. Blockchain je často zobrazován jako svislý zásobník, s bloky vrstvenými na vrcholu každého z nich a první základní blok slouží jako základna zásobníku. Zobrazení bloků naskládaných na sobě vede k běžně používaným výrazům jako "výška" odkazující na vzdálenost od prvního bloku a "vrchol" pro odkaz na nejaktuálněji přidaný blok.

Každý blok v blockchainu je identifikován hašem, vytvořeným kryptografickým hašovacím algoritmem SHA256 aplikovaným na hlavičku bloku. Každý blok také odkazuje na předchozí blok, známý jako *rodičovský* blok, pomocí pole "haš předchozího bloku" v hlavičce bloku. Jinými slovy, každý blok obsahuje haš svého rodiče uvnitř své hlavičky. Posloupnost hašů spojuje každý blok se svým rodičem vytváří řetěz jdoucí zpátky k prvnímu bloku, který byl kdy vytvořen, známému jako *základní blok* (v originále genesis).

Přestože blok má právě jednoho rodiče, může mít dočasně více dětí. Každý z dětí odkazuje na ten samý blok jako na svého rodiče a obsahuje stejný (rodičovský) haš v položce "haš předchozího bloku." Více dětí vznikne během "rozvětvení" blockchainu, dočasné situace, která nastane když různé bloky byly objeveny v téměř shodný čas různými těžaři (viz [forks]). Nakonec, pouze jeden dětský blok se stane částí blockchainu a "rozvětvení" je vyřešeno. I když blok může mít více dětí, každý blok může mít pouze jednoho rodiče. To je protože, blok má pouze jedno pole "haš předchozího bloku" odkazující na jeho jediného rodiče.

Položka "haš předchozího bloku" je uvnitř hlavičky bloku a proto ovlivňuje haš *aktuálního* bloku. Vlastní identita dítěte se změní, pokud se změní identita rodiče. Pokud je rodič změněn v jakémkoliv směru, haš rodiče se změní. Změna haše rodiče vyžaduje změnu v odkazu "haš předchozího bloku" u dítěte. Tento krok způsobí změnu haše dítěte, což vyžaduje změnu v odkazu vnoučete, což způsobí změnu u pravnoučete, atd. Tento kaskádovitý efekt zajišťuje, že jakmile blok má mnoho generací následovníků, nemůže být změněn bez vynucení přepočítání všech následujících bloků. Protože takovéto přepočítání vyžaduje mnoho výpočtů, existence dlouhého řetězu bloků dělá blockchain v hluboké minulosti nezměnitelným, což je klíčovou vlastností bitcoinové bezpečnosti.

V jednom směru můžeme o blockchainu uvažovat jako o geologickém souvrství, nebo vrstvách ledovce. Povrchová vrstvy se mohou změnit s ročními obdobími, nebo dokonce mohou být odfouknuty pryč, předtím než mají čas ses se usadit. Ale jakmile jdete několik centimetrů hluboko, geologické vrstvy se stávají více a více stálé. V době kdy se podíváte několik stovek metrů dolů, díváte se na snímek minulosti, která zůstala neporučena miliony let. V blockchainu, několik nejnovějších bloků může být změněno, pokud je zde přepočítání řetězu z důvodu rozvětvení. Vrchních šest bloků jsou jako několik centimetrů ornice. Ale jakmile jdete hlouběji do blockchainu, za šest bloků, je méně a méně pravděpodobně, že se bloky změní. Po 100 blocích zpátky je už taková stabilita, že mincetvorná

transakce, transakce obsahující nově vytěžené bitcoiny, může být utracena. Několik tisíc bloků zpět (měsíc) a blockchain je ustálenou historií pro všechny praktické účely. Přestože protokol vždy dovoluje vrátit řetěz delším řetězem a protože možnost jakéhokoliv bloku, aby byl vrácen vždy existuje, pravděpodobnost takovéto události se snižuje s uplynulým časem a stává se nekonečně malou.

Struktura bloku

Blok je kontejnerová datová struktura, která shromažďuje transakce pro jejich vložení do veřejného účetního systému, blockchainu. Blok je tvořen hlavičkou obsahující metadata, následovanou dlouhým seznamem transakcí, které tvoří převážnou část jeho velikosti. Hlavička bloku je 80-bytová, zatímco průměrná transakce má alespoň 250 bytů a průměrný blok obsahuje více než 500 transakcí. Celý blok, se všemi transakcemi je proto 1000 krát větší než hlavička bloku. Struktura bloku popisuje strukturu bloku.

Table 1. Struktura bloku

Velikost	Pole	Popis
4 byty	Block Size	Velikost bloku, v kilobytech, následujícího tuto položku
80 bytů	Block Header	Několik položek tvořících hlavičku bloku
1-9 bytů (VarInt)	Transaction Counter	Kolik transakcí následuje
Proměnlivé	Transactions	Transakce zaznamenané v tomto bloku

Hlavička bloku

Hlavička bloku se skládá ze tří množin metadat bloku. Nejprve, je zde odkaz na haš předchozího bloku, který spojuje tento blok s předchozím blokem v blockchainu. Druhou množinou jsou metadata, pojmenované *obtížnost*, *časová značka*, *nonce* vztahující se k těžební soutěži a popsané v [ch8]. Třetí částí metadat je kořenem merkle stromu, datová struktura použitá pro účinné shrnutí všech transakcí v bloku. Struktura hlavičky bloku popisuje strukturu hlavičky bloku.

Table 2. Struktura hlavičky bloku

Velikost	Pole	Popis
4 byty	Version	Číslo verze sledující aktualizaci software / protokolu
32 bytů	Previous Block Hash	Odkaz na haš předchozího (rodičovského) bloku v řetězu
32 bytů	Merkle Root	Haš kořene merkle stromu tohoto bloku transakcí

Velikost	Pole	Popis
4 byty	Timestamp	Přibližný čas vytvoření tohoto bloku (sekundy dle Unix konvence)
4 byty	Difficulty Target	Obtížnostní cíl tohoto bloku pro algoritmus důkazu prací
4 byty	Nonce	Čítač použity pro algoritmus důkazu prací

Nonce, obtížnostní cíl a časová značka použité při těžebním procesu budou probrány podrobněji v [ch8].

Identifikátory bloku: haš hlavičky bloku, výška bloku

Hlavním identifikátorem bloku je kryptografický haš, digitální otisk, vyrobený hašováním hlavičky bloky dvakrát pomocí algoritmu SHA256. Výsledný 32-bytový haš je nazván *haš bloku*, ale přesnější je *haš hlavičky bloky*, protože pouze hlavička bloku je použita pro jeho výpočet. Například, 000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f je haš bloku prvního bitcoinového bloku, který byl kdy vytvořen. Haš bloku identifikuje blok jedinečně a jednoznačně a může být nezávisle odvozeny jakýmkoliv uzlem jednoduše hašováním hlavičky bloku.

Všimněte si, že haš bloku není aktuálně vložen do datové struktury bloku, ani když je blok přenášen po síti, ani když je uložen do trvalého úložiště bloků jako součást blockchainu. Místo toho, haš bloku je počítán každým uzlem, který obdrží blok ze sítě. Haš bloku musí být uložen v oddělené databázové tabulce od zbytku metadat blok, pro snadnější vyhledávání a vyzvedávání bloků z disků.

Druhý způsob identifikace bloku je pomocí jeho pozice v blockchainu, zvané *výška bloku*. První kdy vytvořený blok má výšku 0 (nula) a jedná se o ten samý blok, na který jsme před chvíli odkazovali následujícím hašem bloku 000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f. Blok může být tedy identifikován dvěma způsoby: odkazem na haš bloku nebo odkazem na výšku bloku. Každý následující blok přidaný "na vrchol" tohoto prvního bloku je o jednu pozici výše v blockchainu, jako jsou krabice naskládány jedna na druhé. Výška bloku 1. ledna 2014 byla přibližně 278 000, což znamená, že 278 000 bloků bylo naskládáno na vrcholu prvního bloku vytvořeného v lednu 2009.

Na rozdíl od haše bloku, výška bloku není jedinečný identifikátor. Přestože každý blok bude mít vždy konkrétní a neměnnou výšku bloku, obráceně to neplatí, výška bloku nemusí pokaždé identifikovat jeden blok. Dva a více bloků mohou mít stejnou výšku bloku, při soupeření o stejnou pozici v blockchainu. Tato situace je podrobněji probrána v sekci [forks]. Výška bloku také není součástí datové struktury bloku, není uložena v bloku. Každý uzel dynamicky určuje pozici bloku (výšku) v blockchainu, když ho obdrží z bitcoinové sítě. Výška bloku může být také uložena v matadatech v indexované databázové tabulce pro rychlejší vyhledávání

TIP

Haš bloku vždy identifikuje konkrétní blok jednoznačně. Blok má rovněž konkrétní *výšku bloku*. Nicméně, ne vždy může výška bloku identifikovat konkrétní blok. Občas dva nebo více bloků může soutěžit o jednu pozici v blockchainu.

Základní blok

"blockchains","genesis block") První blok v blockchainu je nazýván základní blok (v originále genesis) a byl vytvořen v roce 2009. Je přímým předkem všech bloků v blockchainu, což znamená, že pokud začnete v jakémkoliv bloku a bude sledovat řetěz nazpátek v čase, nakonec dorazíte do základního bloku.

Každý uzel vždy začíná s blockchainem obsahujícím alespoň jeden blok, protože základní blok je napevno uložen v software bitcoinového klienta, takže nemůže být změněn. Každý uzel vždy "zná" haš základního bolu a strukturu, pevně stanovenou v čase jeho vytvoření, dokonce s jednou transakcí v něm. Proto, každý uzel má tento startovní bod blockchainu, bezpečný "kořen", ze kterého lze postavit důvěryhodný blockchain.

Podívejte se na pevně zakódovaný základní blok v Bitcoin Core klientovi na chainparams.cpp.

Následující identifikační haš patří základnímu bloku:

```
000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f
```

Můžete vyhledat haš bloku v jakémkoliv prohlížeči bloků na webu, jako blockchain.info, a najdete stránku popisující obsah tohoto bloku s URL obsahujícím haš:

https://blockchain.info/block/000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

https://blockexplorer.com/block/000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

Použijte Bitcoin Core referenčního klienta z příkazové řádky:

```
$ bitcoind getblock 000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f
```

```
{
    "hash" : "000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f",
    "confirmations" : 308321,
    "size" : 285,
    "height" : 0,
    "version" : 1,
    "merkleroot" : "4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b",
    "tx" : [
        "4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b"
    ],
    "time" : 1231006505,
    "nonce" : 2083236893,
    "bits" : "1d00ffff",
    "difficulty" : 1.00000000,
    "nextblockhash" : "00000000839a8e6886ab5951d76f411475428afc90947ee320161bbf18eb6048"
}
```


Základní blok obsahuje skrytou zprávu uvnitř. Vstup mincetvorné transakce obsahuje text "The Times 3. ledna 2009 Kancléř na pokraji druhého záchranného balíčku pro banky." Zpráva byla zamýšlena jako důkaz nejranějšího data, kdy mohl být blok vytvořen, odkazem na titulek britských novin *The Times*. Zároveň slouží jako vtipné připomenutí důležitosti nezávislého měnového systému, který se spuštěním bitcoinu objevil ve stejný čas jako bezprecedentní světová finanční krize. Zpráva byla zakódována v prvním bloku od Satoshi Nakamota, tvůrce bitcoinu.

Spojení bloků v blockchainu

Bitcoinové úplné uzly obsahují místní kopii blockchainu, začínající v základním bloku. Místní kopie blockchainu je stále aktualizována novými bloky, které jsou nalezeny a použity pro prodloužení blockchainu. Když uzel obdrží příchozí bloky ze sítě, ověří tyto bloky a spojí je s existujícím blockchainem. Pro vytvoření spojení, uzel musí prozkoumat příchozí hlavičku bloku a podívat se na "haš předchozího bloku."

Předpokládejme, například, že uzel má 277 314 bloků v místní kopii blockchainu. Haš hlavičky bloku 00000000000000027e7ba6fe7bad39faf3b5a83daed765f05f7d1b71a1632249 patří poslednímu známému bloku 277 314.

Bitcoinový uzel poté obdrží nový blok ze sítě, který analyzuje následovně:

```
{
    "size" : 43560,
    "version" : 2,
    "previousblockhash" :
        "0000000000000027e7ba6fe7bad39faf3b5a83daed765f05f7d1b71a1632249",
    "merkleroot" :
        "5e049f4030e0ab2debb92378f53c0a6e09548aea083f3ab25e1d94ea1155e29d",
    "time" : 1388185038,
    "difficulty" : 1180923195.25802612,
    "nonce" : 4215469401,
    "tx" : [
        "257e7497fb8bc68421eb2c7b699dbab234831600e7352f0d9e6522c7cf3f6c77",

 #[... mnoho dalších transakcí vynecháno ...]

        "05cfd38f6ae6aa83674cc99e4d75a1458c165b7ab84725eda41d018a09176634"
    ]
}
```

Podíváme se na tento nový blok, uzel našel pole previousblockhash, které obsahuje haš rodičovského bloku. Je to haš známý uzlu, je to haš posledního bloku řetězu s výškou 277 314. Proto nový blok je dítětem posledního bloku řetězu a prodlužuje existující blockchain. Uzel přidá nový blok na konec řetězu, čímž prodlouží blockchain na novou výšku 277 315. Bloky spojené v řetěz odkazy na haš hlavičky předchozího bloku ukazuje řetěz tří bloků spojených odkazem v položce previousblockhash.

Merkle stromy

Každý blok v bitcoinovém blockchainu obsahuje souhrn všech transakcí v bloku za použití *merkle stromu*.

Merkle strom také známý jako *binární hašovaný strom* je datová struktura použita pro účinné shrnutí a ověření celistvosti velké množiny dat. Merkle stromy jsou binární stromy obsahující kryptografické haše. Pojem "strom" je použit v počítačové vědě pro popis rozvětvené datové struktury, ale tyto stromy jsou obvykle zobrazovány vzhůru nohama, s kořenem nahoře a listy dole, jak uvidíme v následujících příkladech.

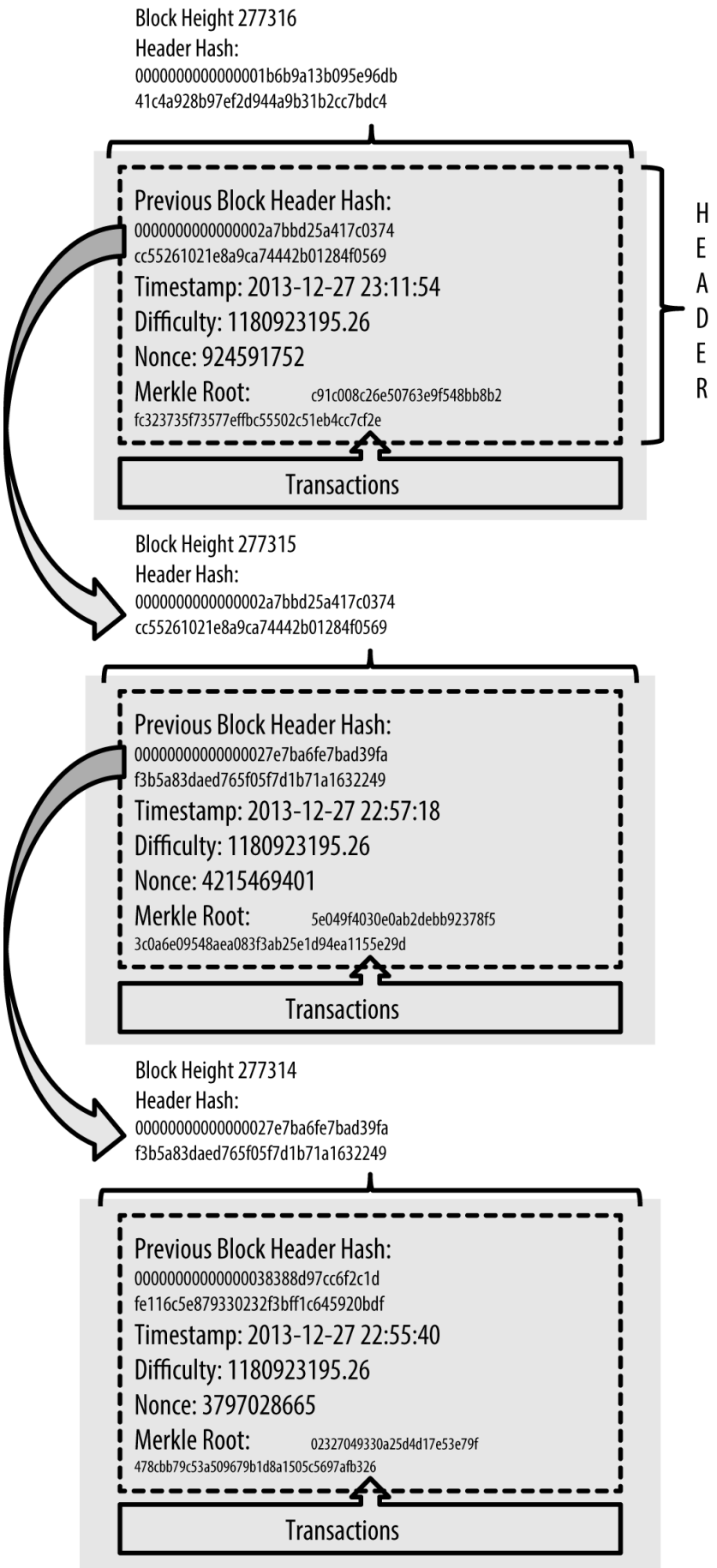

Block Height 277316
Header Hash:
0000000000000001b6b9a13b095e96db
41c4a928b97ef2d944a9b31b2cc7bdc4
Previous Block Header Hash:
0000000000000002a7bbd25a417c0374
cc55261021e8a9ca74442b01284f0569
Timestamp: 2013-12-27 23:11:54
Difficulty: 1180923195.26
Nonce: 924591752
Merkle Root: c91c008c26e50763e9f548bb8b2
fc323735f73577effbc55502c51eb4cc7cf2e
Transactions
HEADER
Block Height 277315
Header Hash:
0000000000000002a7bbd25a417c0374
cc55261021e8a9ca74442b01284f0569
Previous Block Header Hash:
00000000000000027e7ba6fe7bad39fa
f3b5a83daed765f05f7d1b71a1632249
Timestamp: 2013-12-27 22:57:18
Difficulty: 1180923195.26
Nonce: 4215469401
Merkle Root: 5e049f4030e0ab2debb92378f5
3c0a6e09548aea083f3ab25e1d94ea1155e29d
Transactions
Block Height 277314
Header Hash:
00000000000000027e7ba6fe7bad39fa
f3b5a83daed765f05f7d1b71a1632249
Previous Block Header Hash:
00000000000000038388d97cc6f2c1d
fe116c5e879330232f3bff1c645920bdf
Timestamp: 2013-12-27 22:55:40
Difficulty: 1180923195.26
Nonce: 3797028665
Merkle Root: 02327049330a25d4d17e53e79f
478cbb79c53a509679b1d8a1505c5697afb326
Transactions

Figure 1. Bloky spojené v řetěz odkazy na haš hlavičky předchozího bloku

Merkle stromy jsou použity v bitcoinu pro shrnutí všech transakcí v bloku, vytvářejí celkový digitální otisk celé množiny transakcí, poskytují velmi účinný postup ověření zda se transakce nachází v bloku. Merkle stromy jsou vytvořeny rekurzivním hašováním dvojic uzlů dokud nezbývá jediný haš zvaný *kořen* nebo *kořen merkle stromu*. Kryptografický hašovací algoritmus použitý v bitcoinovém merkle stromu je SHA256 aplikovaný dvakrát, také známý dvojitý SHA256.

Když N datových prvků je hašováno a shrnuto v merkle stromu, můžete zkontrolovat, zda nějaká datová položka je vložena do stromu s nejvýše 2*log~2~(N) výpočty, což činí tuto datovou strukturu velmi účinnou.

Merkle strom je postaven zdola nahoru. V následujícím příkladě začneme se čtyřmi transakcemi A, B, C a D, která tvoří *listy* merkle stromu, jak ukazuje Výpočet uzlů v merkle stromu. Transakce nejsou uloženy v merkle stromu; spíše jejích data jsou hašována a výsledné haše jsou uloženy v jednotlivých listových uzlech jako H_A, H_B, H_C, a H_D:

```
+H~A~ = SHA256(SHA256(Transakce A))+
```

Po sobě jdoucí dvojice listových uzlů jsou shrnuty v rodičovském uzlu, zřetězením těchto dvou hašů a zahašováním jich dohromady. Například při tvorbě rodičovského uzlu H_{AB} dva 32-bytové haše dětí jsou spojeny, aby vytvořili 64-bytový řetězec. Tento řetězec je dvakrát hašován, aby vytvořil haš rodičovského uzlu.

```
H_AB = SHA256(SHA256(H_A + H_B))
```

Celý postup pokračuje dokud nezůstane pouze jeden uzel na vrcholu, známý jak merkle kořen. Tento 32-bytový haš je uložen v hlavičce bloku a shrnuje všechna data ve všech čtyřech transakcích.

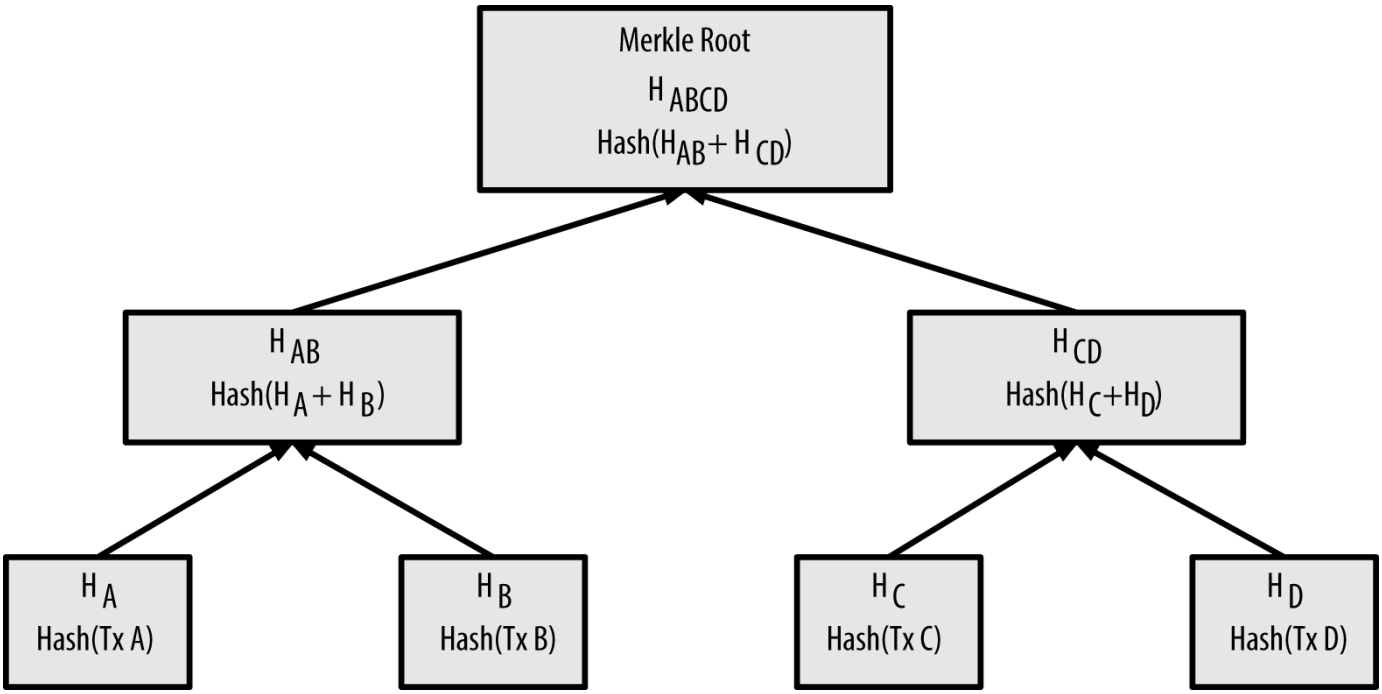


Figure 2. Výpočet uzlů v merkle stromu

Protože merkle strom je binární strom, potřebuje sudý počet listových uzlů. Pokud je lichý počet transakcí pro shrnutí, haš poslední transakce bude zdvojen, aby byl vytvořen sudý počet listových uzlů, také známý jako *vyvážený strom*. Toto je znázorněno v Zdvojením datového prvku dosáhneme sudého počtu datových prvků, kde transakce C je zdvojena.

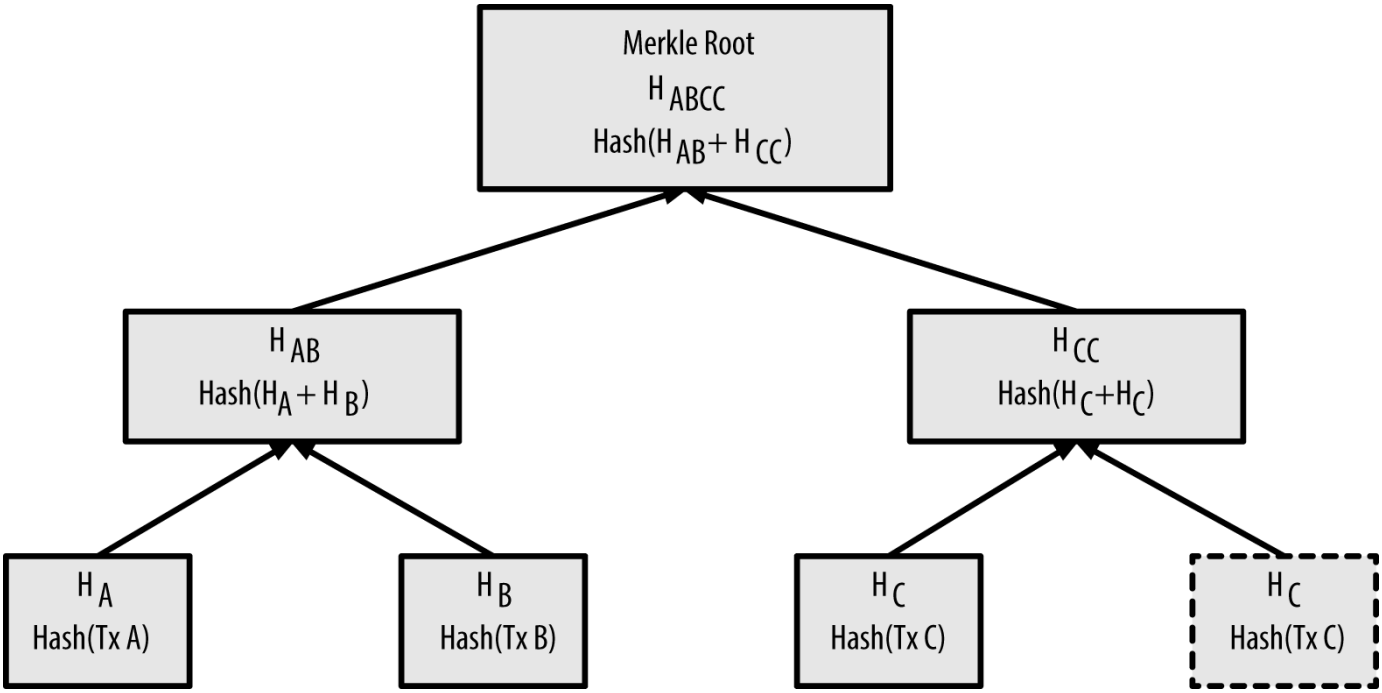


Figure 3. Zdvojením datového prvku dosáhneme sudého počtu datových prvků

Ta samá metoda pro konstrukci stromu ze čtyř transakcí může být zobecněna na konstrukci stromu libovolné velikosti. V bitcoinu je obvyklé mít několik stovek až více než tisíc transakcí v jediném bloku, kterou jsou shrnuty přesně stejným způsobem, vytvářející 32 bytů dat jako jediný kořen merkle

stromu. V Merkle strom shrnující mnoho datových prvků vidíte stavbu ze 16 transakcí. Všimněte si, přestože kořen vypadá větší než listové uzly v diagramu, mají přesně stejnou velikost, právě 32 bytů. Je jedno zda v bloku jedna transakce, nebo jich jsou stovky tisíc, kořen merkle stromu je vždy shrne do 32 bytů.

Pro dokázání, že konkrétní transakce je zahrnuta v bloku, uzel potřebuje pouze vyrobit log~2~(N) 32-bytových hašů, tvořících *ověřovací cestu* nebo *merkle cestu* spojující konkrétní transakci s kořenem stromu. To je zvláště důležité jak se počet transakcí zvyšuje, protože logaritmus o základu 2 z počtu transakcí se zvyšuje mnohem pomaleji. To umožňuje bitcoinovým uzlům účinně vyrobit cestu 12 až 12 hašů (320 - 384 bytů). která může poskytnout důkaz jednotlivé transakce z více než tisíce transakcí v megabytovém bloku.

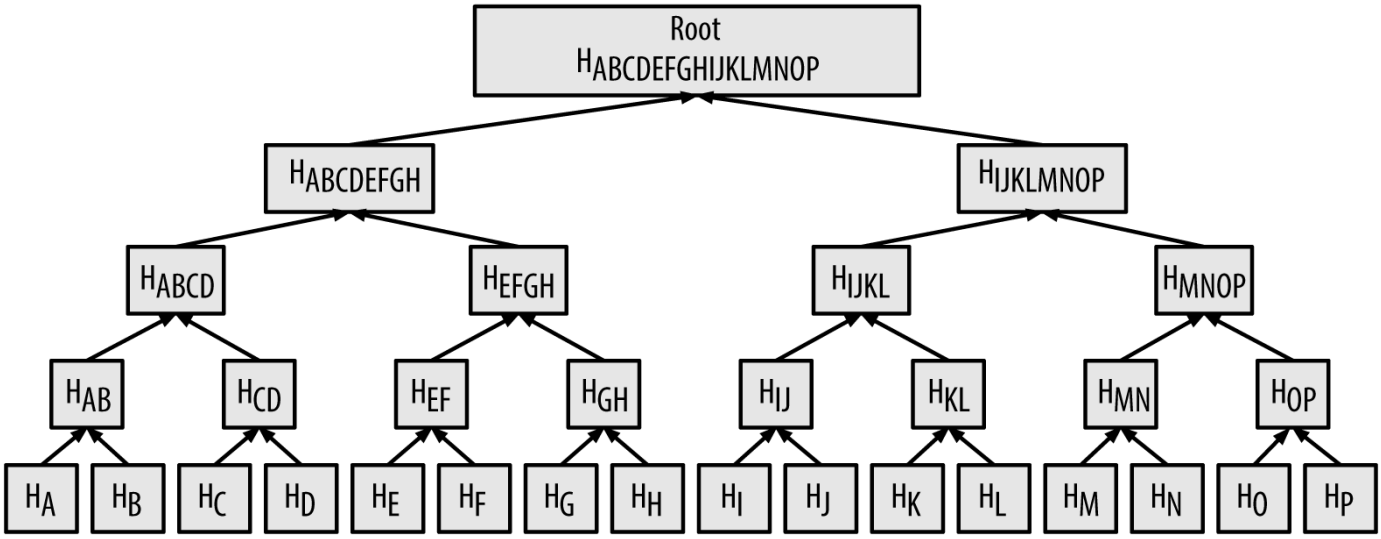


Figure 4. Merkle strom shrnující mnoho datových prvků

V Merkle cesta použita pro důkaz vložení datového prvku, uzel může dokázat, že transakce K je vložena v bloku vyrobením merkle cesty, která je pouze dlouhá čtyři 32-bytové haše (128 bytů celkem). Cesta se skládá ze čtyř hašů (označeny modře v Merkle cesta použita pro důkaz vložení datového prvku) H_L, H_{IJ}, H_{MNOP} a $H_{ABCDEFGH}$. S těmito čtyřmi haši poskytnutými jako ověřovací cesta, jakýkoliv uzel může dokázat, že H_K (označen zeleně v diagramu) je vložen v kořeni merkle stromu pomocí výpočtu čtyř dodatečných párových hašů H_{KL}, H_{IJKL}, $H_{IJKLMNOP}$a kořene merkle stromu (ohraničené čárkovanou čárou v diagramu).

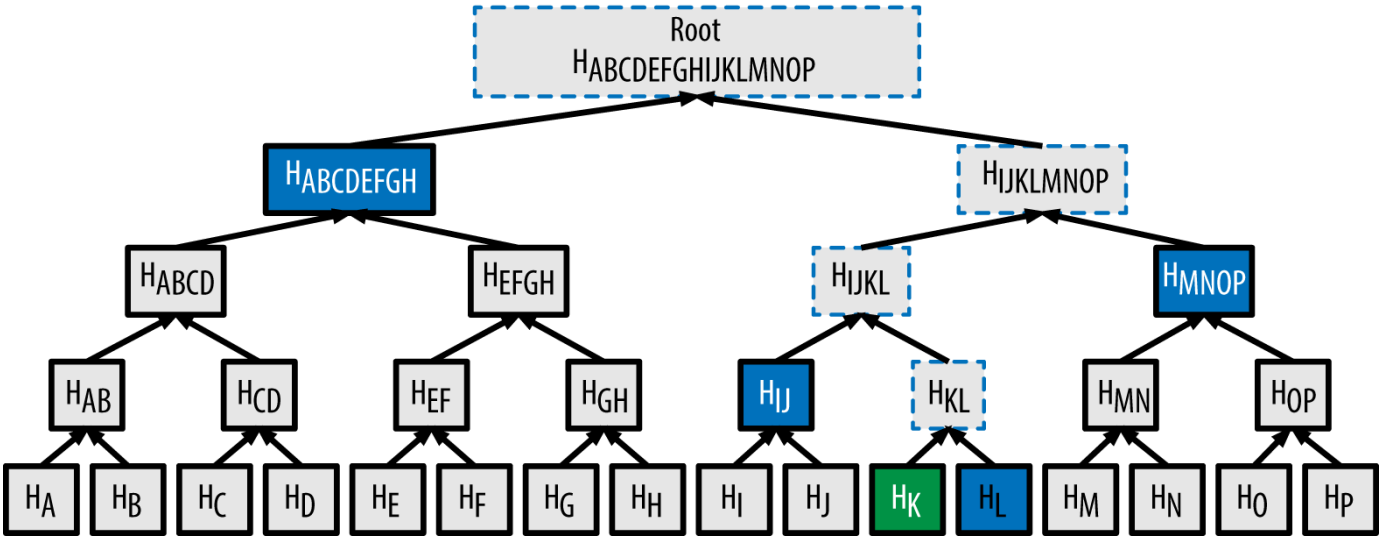


Figure 5. Merkle cesta použita pro důkaz vložení datového prvku

Zdrojový kód představuje postup tvorby merkle stromu od listových hašů nahoru ke kořenu, za použití knihovny libbitcoin a některých pomocných funkcí.

Example 1. Vytváření merkle stromu

```
#include <bitcoin/bitcoin.hpp>

bc::hash_digest create_merkle(bc::hash_list& merkle)
{
    // Stop if hash list is empty.
    if (merkle.empty())
        return bc::null_hash;
    else if (merkle.size() == 1)
        return merkle[0];

    // While there is more than 1 hash in the list, keep looping...
    while (merkle.size() > 1)
    {
        // If number of hashes is odd, duplicate last hash in the list.
        if (merkle.size() % 2 != 0)
            merkle.push_back(merkle.back());
        // List size is now even.
        assert(merkle.size() % 2 == 0);

        // New hash list.
        bc::hash_list new_merkle;
        // Loop through hashes 2 at a time.
        for (auto it = merkle.begin(); it != merkle.end(); it += 2)
        {
            // Join both current hashes together (concatenate).
            bc::data_chunk concat_data(bc::hash_size * 2);
```

```
            auto concat = bc::make_serializer(concat_data.begin());
            concat.write_hash(*it);
            concat.write_hash(*(it + 1));
            assert(concat.iterator() == concat_data.end());
            // Hash both of the hashes.
            bc::hash_digest new_root = bc::bitcoin_hash(concat_data);
            // Add this to the new list.
            new_merkle.push_back(new_root);
        }
        // This is the new list.
        merkle = new_merkle;

        // DEBUG output -------------------------------------
        std::cout << "Current merkle hash list:" << std::endl;
        for (const auto& hash: merkle)
            std::cout << "  " << bc::encode_hex(hash) << std::endl;
        std::cout << std::endl;
        // --------------------------------------------------
    }
    // Finally we end up with a single item.
    return merkle[0];
}

int main()
{
    // Replace these hashes with ones from a block to reproduce the same merkle root.
    bc::hash_list tx_hashes{{

bc::hash_literal("0000000000000000000000000000000000000000000000000000000000000000"),

bc::hash_literal("0000000000000000000000000000000000000000000000000000000000000011"),

bc::hash_literal("0000000000000000000000000000000000000000000000000000000000000022"),
    }};
    const bc::hash_digest merkle_root = create_merkle(tx_hashes);
    std::cout << "Result: " << bc::encode_hex(merkle_root) << std::endl;
    return 0;
}
```

Kompilace a spuštění zdrojového kódu merkle ukazuje výsledek kompilace a spuštění zdrojového kódu merkle.

Example 2. Kompilace a spuštění zdrojového kódu merkle

```
$ # Kompilace zdrojového kódu merkle.cpp
$ g++ -o merkle merkle.cpp $(pkg-config --cflags --libs libbitcoin)
$ # Spuštění spustitelného merkle
$ ./merkle
Current merkle hash list:
  32650049a0418e4380db0af81788635d8b65424d397170b8499cdc28c4d27006
  30861db96905c8dc8b99398ca1cd5bd5b84ac3264a4e1b3e65afa1bcee7540c4

Current merkle hash list:
  d47780c084bad3830bcdaf6eace035e4c6cbf646d103795d22104fb105014ba3

Result: d47780c084bad3830bcdaf6eace035e4c6cbf646d103795d22104fb105014ba3
```

Účinnost merkle stromů se stává zřejmou se zvyšujícím se rozsahem. Účinnost merkle stromu ukazuje množství dat potřebných pro výměnu jako merkle cesty dokazující, že transakce je částí bloku.

Table 3. Účinnost merkle stromu

Počet transakcí	Přibližná velikost bloku	Velikost cesty (haše)	Velikost cesty (byty)
16 transakcí	4 kilobyty	4 haše	128 bytů
512 transakcí	128 kilobytů	9 hašů	288 bytů
2048 transakcí	512 kilobytů	11 hašů	352 bytů
65 535 transakcí	16 megabytů	16 hašů	512 bytů

Jak můžete vidět z tabulky, zatímco velikost bloku roste rychle, z 4 KB se 16 transakcemi na blok velikosti 16 MB pro 65 535 transakcí, merkle cesta požadovaná pro dokázání zahrnutí transakce se zvyšuje mnohem pomaleji ze 128 bytů na pouze 512 bytů. S merkle stromy, uzly mohou stahovat jen hlavičky bloků (80 bytů na blok) a stále jsou schopny identifikovat vložení transakce do bloku získáním malé merkle cesty od úplného uzlu, bez ukládání nebo přenášení rozsáhlé části blockchainu, který může být několik gigabytu velký. Uzly, které neudržují úplný blockchain, zvané zjednodušené ověřování plateb (SPV uzly), používají merkle cesty pro ověření transakcí bez stahování úplných bloků.

Merkle stromy a zjednodušené ověřování transakcí (SPV)

Merkle stromy jsou používány značně SPV uzly. SPV uzly nemají všechny transakce a nestahují úplné bloky, pouze hlavičky bloků. Za účelem ověření, že transakce je zahrnuta v bloku, aniž by stahovali všechny transakce v bloku, používají ověřovací cestu nebo merkle cestu.

Zvažte, například, SPV uzel, který se zajímá o příchozí platby na adresu obsaženou v jeho peněžence. SPV uzel založí Bloomův filtr na své spojení ke klientskému uzlu, aby omezil obdržené transakce pouze na ty obsahující adresy zájmu. Když klientský uzel vyhledává transakce, které splňují Bloomův filtr, zašle jejich blok za použití zprávy merkleblock. Zpráva merkleblock obsahuje hlavičky bloků stejně jako merkle cesty, které spojují tyto transakce zájmu s kořenem merkle stromu bloku. SPV uzel může použít tyto merkle cesty pro spojení transakce s blokem a ověření, že transakce je vložena do bloku. SPV uzel také používá hlavičky bloků pro spojení bloků se zbytkem blockchainu. Kombinace těchto dvou spojení, mezi transakcí a blokem a mezi blokem a blockchainem dokazuje, že transakce je zaznamenána v blockchainu. Vše v jednom, SPV uzel potřebuje obdržet méně než kilobyte dat (hlavička bloku a merkle cesta), což je množství dat více než tisíckrát menší než plný blok (aktuálně okolo 1 megabyte).

Těžba a shoda

Úvod

Těžba je proces, při kterém jsou nové bitcoiny přidány do peněžní zásoby. Těžba také slouží k zabezpečení bitcoinového systému proti podvodným transakcím nebo transakcím utrácejícím stejné bitcoiny více než jednou, známé jako dvojité utracení. Těžaři poskytují zpracovatelskou sílu bitcoinové síti výměnou za možnost být odměněni bitcoiny.

Těžaři ověřují nové transakce a zaznamenávají je do celosvětového účetního systému. Nový blok obsahující transakce, které se objevili od posledního bloku, je "vytěžen" v průměru každých 10 minut, čímž přidává tyto transakce do blockchainu. Transakce, které se staly částí bloku a byly přidány do blockchainu jsou považovány za potvrzené, což umožňuje novým majitelům bitcoinů utratit tyto bitcoiny získané v těchto transakcích.

Těžaři získávají dva typy odměn za těžbu: nové mince vytvořené s každým novým blokem a transakční poplatky ze všech transakcí zahrnutých do bloku. Pro získání této odměny, těžaři soutěží v řešení obtížného matematického problému založeného na kryptografickém hašovacím algoritmu. Řešení tohoto problému nazvané důkaz prací je zahrnuto do nového bloku a slouží jako důkaz, že těžař vynaložil značné výpočetní úsilí. Soutěž v řešení algoritmu důkazu prací pro získání odměny a právo zaznamenat transakce do blockchainu je základem bitcoinového bezpečnostního modelu.

Postup tvorby nových mincí je nazýván těžba, protože odměna je navržena, aby napodobovala zmenšující se výnosy, jako při těžbě drahých kovů. Bitcoinová peněžní zásoba je tvořena těžbou, podobně jako centrální banky vydávají nové peníze tiskem bankovek.Počet nově vytvořených bitcoinů, které těžař smí přidat do bloku se snižuje přibližně každé čtyři roky (přesněji každých 210 000 bloků). Začínalo na 50 bitcoinech za blok v lednu 2009 a bylo rozpůleno na 25 bitcoinů za blok v listopadu 2012 a znovu rozpůleno na 12,5 bitcoinu za blok v červenci 2016. Založeno na tomto vzorci, odměna za těžbu bitcoinu bude klesat exponenciálně přibližně do roku 2140, kdy všechny bitcoiny (20 999 999,98) budou vydány, Po roce 2140 nové bitcoiny nebudou vydávány.

Bitcoinoví těžaři také získávají poplatky z transakcí. Každá transakce může obsahovat transakční poplatek ve formě přebytku bitcoinů mezi transakčními vstupy a výstupy. Vítězný těžař obdrží "nechte si drobné" z transakcí vložených do vítězného bloku. Dnes, poplatky reprezentují 5% nebo méně příjmů bitcoinových těžařů, převážná většina pochází z nově ražených mincí. Nicméně, jak se odměna snižuje postupem času a počet transakcí v bloku roste, vetší část příjmů z těžby bude pocházet z poplatků. Po roce 2014 všechny příjmy těžařů budou ve formě transakčních poplatků.

Slovo "těžba" je občas zavádějící. Odvolání se na těžbu drahých kovů, zaměřujeme naší pozornost na odměnu za těžbu, nové bitcoiny v každém bloku. Přestože těžba je motivována touto odměnou, hlavním účelem těžby není odměna nebo tvorba nových mincí. Pokud se na těžbu díváte pouze jako na proces, při kterém jsou vytvářeny nové mince, pletete si prostředky (pobídky) s cílem celého postupu. Těžba je hlavní proces decentralizovaného zúčtovacího systému, kterým jsou transakce ověřovány a vypořádávány. Těžba zabezpečuje bitcoinový systém a umožňuje vznik shody v celé síti bez centrální

autority.

Těžba je vynález, který dělá bitcoin zvláštní, decentralizovaný bezpečnostní mechanizmus je základem peer-to-peer digitální hotovosti. Odměna nově vyraženými mincemi a transakčními poplatky je pobídkou, která slaďuje akce těžařů s bezpečností sítě, zatímco současně poskytuje měnovou zásobu.

V této kapitole, nejprve prozkoumáme těžbu jako mechanizmus peněžní zásoby a poté se podíváme na nejdůležitější funkci těžby: decentralizovaný mechanizmus vzniku shody, který je základem bitcoinové bezpečnosti

Bitcoinová ekonomika a tvorba měny

Bitcoiny jsou "raženy" během tvorby každého bloku s pevnou a zmenšující se rychlostí. Každý blok vytvořený v průměru každých 10 minut, obsahuje zcela nové bitcoiny vytvořené z ničeho. Každých 210 000 bloků, přibližně každé 4 roky, se rychlost vydávání měny snižuje o 50%. První čtyři roky fungování sítě, každý blok obsahoval 50 nových bitcoinů.

V listopadu 2012 rychlost vydávání nových bitcoinů byla snížena na 25 bitcoinů za blok a následně poklesla na 12,5 bitcoinu od bloku 420 000, který byl vytěžen v červenci 2016. Rychlost vydávání nových mincí se snižuje exponenciálně, dojde k 64 "půlení" do bloku 13 230 000 (vytěženému přibližně v roce 2137), když dosáhne minimální jednotky měny 1 satoshi. Nakonec po bloku 13,44 milionu, přibližně v roce 2140 téměř 21 milionu bitcoinu (přesněji 2 099 999 997 690 000 satoshi) bude vydáno. Poté bloky nebudou obsahovat žádné nové bitcoiny a těžaři budou odměňování výhradně transakčními poplatky. Nabídka bitcoinové měny v průběhu času založena na geometricky klesající rychlosti vydávání ukazuje celkový počet bitcoinů v oběhu v průběhu času, jak vydávání měny bude klesat.

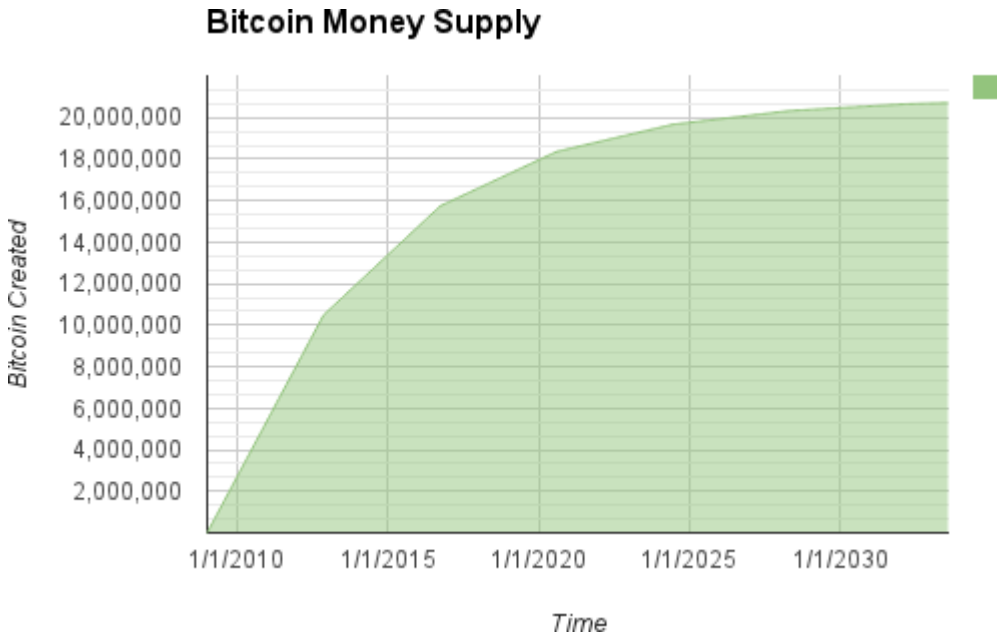


Figure 1. Nabídka bitcoinové měny v průběhu času založena na geometricky klesající rychlosti vydávání

NOTE	Maximální počet vytěžených mincí je *horní omezení* možných odměn těžby pro bitcoin. Ve skutečnosti, těžaři mohou záměrně vytěžit blok a vzít menší než plnou odměnu. Takové bloky byly skutečně vytěženy a další mohou být vytěženy v budoucnu, což vede k celkově nižší peněžní zásobě.

V příkladu zdrojového kódu v Skript počítající kolik bitcoinů bude vydáno počítáme celkový počet bitcoinů, které budou vydány

Example 1. Skript počítající kolik bitcoinů bude vydáno

```
# Original block reward for miners was 50 BTC
start_block_reward = 50
# 210000 is around every 4 years with a 10 minute block interval
reward_interval = 210000

def max_money():
    # 50 BTC = 50 0000 0000 Satoshis
    current_reward = 50 * 10**8
    total = 0
    while current_reward > 0:
        total += reward_interval * current_reward
        current_reward /= 2
    return total

print "Total BTC to ever be created:", max_money(), "Satoshis"
```

Spuštění skriptu max_money.py ukazuje výstup vyrobený tímto běžícím skriptem

Example 2. Spuštění skriptu max_money.py

```
$ python max_money.py
Total BTC to ever be created: 2099999997690000 Satoshis
```

Konečná a zmenšující se emise měny vytváří pevnou nabídku měny, která odolává inflaci. Na rozdíl od běžných měn, které jsou tištěné v nekonečných počtech centrálním bankami, bitcoin nemůže být nikdy inflačně tištěn.

Deflační peníze

Nejdůležitějším a nejvíce diskutovaným důsledkem pevné a zmenšující se emise měny je, že měna bude ze své podstaty *deflační*. Deflace je jev posilování hodnoty v důsledku nesouladu v nabídce a poptávce, která zvyšuje hodnotu (a směnný kurz) měny. Na rozdíl od inflace, deflační ceny znamenají, že peníze zvyšují svojí kupní sílu v průběhu času.

Mnoho ekonomů argumentuje, že deflační ekonomika je neštěstím, kterém se je třeba vyhnout za každou cenu. Protože v období prudké deflace, lidé hromadí peníze, místo jejich utrácení, doufají, že ceny zboží ještě více spadnou. Tento jev se objevil během Japonského "Ztraceného desetiletí", když úplné selhání poptávky tlačilo měnu do inflační spirály.

Bitcoinoví experti argumentují, že deflace není špatná sama o sobě. Spíše deflace je spojována s poklesem poptávky, protože je to jediný příklad deflace, který byl studován. U klasických měn s možností neomezeného tisku, je velmi těžké vstoupit do deflační spirály dokud nedojde k úplnému selhání poptávky a neochoty tisknout peníze. Deflace v bitcoinu není způsobena selháním poptávky, ale ke předpověditelná omezenou nabídkou.

V praxi se stalo zřejmým, že instinkt hromadění způsobený deflační měnou lze překonat slevami obchodníků, až slevy překonají instinkt hromadění zákazníků. Protože prodávající jsou také motivování hromadit, slevy se stanou vyváženou cenou při které se oba instinkty hromadění setkají. Se slevou 30% v bitcoinových cenách většinou držitelů bitcoinů nebude pociťovat obtíže způsobení instinktem hromaděni a tvorby zisku. Je třeba vyčkat zda deflační vlastnost měny je skutečným problém, když není řízena rychlým poklesem ekonomiky.

Decentralizovaná shoda

V předchozí kapitole jsme se podívali na blockchain, celosvětový účetní systém, seznam všech transakcí, který každý v bitcoinové síti přijímá jako směrodatný záznam vlastnictví.

Ale jak se můžou všichni v síti shodnout na jedné univerzální "pravdě" o tom, kdo co vlastní, bez nutnosti věřit někomu? Všechny tradiční platební systémy závisí na modelu důvěry, který má centrální autoritu poskytující službu zúčtovacího střediska, obvykle ověřující a vypořádávající všechny transakce. Bitcoin nemá centrální autoritu, ale přesto každý úplný uzel má kompletní kopii veřejného účetního systému, kterému může být věřeno jako spolehlivému záznamu. Blockchain není vytvořen centrální autoritou, ale je složen nezávisle každým uzlem v síti. Nějakým způsobem, každý uzel v sítí, působí na informace přenášené přes nezabezpečená síťová spojení, může dojít ke stejným závěrům a složit kopii stejného veřejného účetního systému jako kdokoliv jiný. Tato kapitola prozkoumá postup, kterým bitcoinová síť dosahuje celosvětové shody bez centrální autority.

Hlavní vynález Satoshi Nakamota je decentralizovaný mechanizmus pro *vznikání shody*. Vznikání, protože shoda není dosažena explicitně, nejsou zde žádné volby nebo pevné okamžiky, kdy shoda nastává. Místo toho, shoda je vznikajícím pozůstatkem asynchronního působení tisíců nezávislých uzlů řídících se jednoduchými pravidly. Všechny vlastnosti bitcoinu, včetně měny, transakcí, plateb a

bezpečnostního modelu, který nezávisí na centrální autoritě nebo důvěře, jsou odvozeny z tohoto vynálezu.

Bitcoinová decentralizovaná vznikající shoda ze souhry čtyř procesů, které nastávají nezávisle na uzlech v síti:

- Nezávislé ověřování všech transakcí, každým pevným uzlem, založené na obsáhlém seznamu kritérií.
- Nezávislé shromažďování těchto transakcí do nových bloků těžebními uzly, spolu s prokázanými výpočty pomocí algoritmu důkazu prací
- Nezávislé ověřování nových bloků každým uzlem a jejich skládání do řetězu
- Nezávislá volba, každým uzlem, řetězu s nejvyšším součtem výpočtů prokázaných pomocí důkazu prací

V následujících několika sekcích prozkoumáme tyto procesy jak vzájemně působí na vznikající vlastnost shody v celé síti, která umožňuje každému bitcoinovému uzlu složit si vlastní kopii platného, důvěryhodného, veřejného, celosvětového účetního systému.

Nezávislé ověřování transakcí

V [transactions] jsme viděli, jak peněženkový software vytváří transakce shromažďováním UTXO, poskytnutím odpovídajících odemykacích skriptů a konstrukcí nových výstupů přiřazených novému majiteli. Výsledná transakce je poté zaslána sousedním uzlům bitcoinové sítě, takže může být rozšířena po celé bitcoinové síti.

Nicméně, před přeposláním transakce svým sousedům, každý bitcoinový uzel, který obdržel transakci nejprve tuto transakci ověří. To zajišťuje, že pouze platné transakce jsou šířený sítí, zatímco neplatné transakce jsou zahozeny prvním uzlem, který se s nimi setká.

Každý uzel ověřuje každou transakci proti dlouhému seznamu kontrolních kritérií.

- Syntax transakce a datová struktura jsou v pořádku
- Ani seznam vstupů ani seznam výstupů není prázdný
- Velikost transakce v bytech je menši než MAX_BLOCK_SIZE.
- Každá výstupní hodnota, stejně jako jejich součet, musí být v povoleném intervalu hodnot (méně než 21 milionu mincí, více než 0).
- Žádný ze vstupu nemá haš=0, N=-1 (mincetvorná transakce by neměla být přenášena).
- nLockTime je menší nebo roven INT_MAX.
- Velikost transakce v bytech je vyšší nebo rovna 100.
- Počet podpisových operací obsažených v transakci je nižší nebo roven omezení počtu podpisových operací.

- Odemykací skript (scriptSig) může pouze vkládat čísla na zásobník a zamykací skript (scriptPubkey) musí splňovat isStandard tvar (toto odmítá "nestandardní" transakce).
- Odpovídající transakce v úložišti nebo bloku v hlavní větvi musí existovat.
- Pro každý vstup, pokud odkazuje na existující výstup jiné transakce v úložišti,, transakce musí být odmítnuta.
- Pro každý vstup, se podívá v hlavní větvi a transakčním úložišti, aby nalezl odkazovanou výstupní transakci. Pokud výstupní transakce chybí pro nějaký vstup, toto bude sirotčí transakce. Je přidána do úložiště sirotčích transakcí, pokud odpovídající transakce není již v úložišti.
- Pro každý vstup, pokud odkazovaný výstup transakce je mincetvorný výstup, musí dosáhnout alespoň COINBASE_MATURITY (100) potvrzení.
- Pro každý vstup, odkazovaný výstup musí existovat a nesmí být již utracen.
- Použitím odkazovaného výstupu transakcí pro získání vstupné hodnoty, zkontroluje hodnotu každého vstupu, stejně jako jejich součet, zda jsou z intervalu hodnotu (nižší než 21 milionu mincí, vyšší než 0).
- Odmítne, pokud součet vstupních hodnot je nižší než součet výstupních hodnot.
- Odmítne pokud transakční poplatek by byl příliš nízký pro dostání se do prázdného bloku.
- Odemykací skripty pro každý vstup musejí být ověřeny proti odpovídajícím zamykacím skriptům výstupům.

Tyto podmínky mohou být prohlédnuty detailněji ve funkcích AcceptToMemoryPool, CheckTransaction, a CheckInputs v bitcoinovém referenčním klientovi. Tyto podmínky se mění v průběhu času v reakci na nové typy útoků odepřením služby nebo občasným uvolňováním pravidel, aby mohlo být vloženo více typů transakcí.

Nezávislým ověřováním každé transakce po jejím přijetí a před jejím rozšířením, každý uzel vytváří úložiště platných (ale nepotvrzených) transakcí známé jako *dočasné úložiště transakcí* nebo _paměťové úložiště.

Těžební uzly

Některé z uzlů bitcoinové sítě jsou specializované uzly zvané *těžaři*. V [ch01_intro_what_is_bitcoin] jsme si představili Jinga, studenta počítačového inženýrství ze Šanghaje v Číně, který je bitcoinovým těžařem. Jing vydělává bitcoiny provozováním "těžební soupravy", která je specializovaným těžebním hardware připojené na server provozující úplný bitcoinový uzel. Na rozdíl od Jinga, někteří těžaři těží bez úplného uzlu, jak uvidíme v Těžební skupiny. Jako každý jiný úplný uzel, Jingův uzel získává a rozšiřuje nepotvrzené transakce v bitcoinové síti. Jingův uzel, nicméně, také skládá tyto transakce do nových bloků.

Jingův uzel je schopen přijímat nové bloky, rozšiřovat tyto bloky po bitcoinové síti, jako to dělají všechny uzly. Nicméně, příchod nového bloku má zvláštní význam pro těžební uzel. Soutěž mezi těžaři fakticky končí s rozšířením nového bloku, který působí jako oznámení vítěze. Pro těžaře, kteří obdrží

nový blok to znamená, že někdo jiný vyhrál soutěž a oni prohráli. Nicméně konec jednoho kola soutěže je zároveň začátkem dalšího kola soutěže. Nový blok není jen cílovou vlajkou, označující konec závodu ale zároveň je startovní v závodu o další blok.

Seskupování transakcí do bloků

Po ověření transakcí, bitcoinový uzel je přidá do *paměťového úložiště* nebo *úložiště transakcí*, kde transakce čekají dokud nejsou zahrnuty (vytěženy) do bloku. Jingův uzel sbírá, ověřuje a přenáší nové transakce jako jakýkoliv jiný uzel. Na rozdíl od jiných uzlů. nicméně, Jingův uzel bude seskupovat tyto transakce do *kandidátského bloku*.

Podívejme se na bloky, které byly vytvořeny během doby, kdy Alice koupila šálek kávy v Bobově kavárně (viz [cup_of_coffee]). Transakce Alice byla vložena do bloku 277316. Pro ukázkové účely pojmů v této kapitole, předpokládejme, že blok byl vytěžen Jingovým těžebním systémem a sledujeme transakci Alice, jak se stává součástí tohoto nového bloku.

Jingův těžební uzel udržuje místní kopii blockchainu, seznam všech bloků vytvořených od začátku bitcoinového systému v roce 2009. V čase nákupu šálku kávy Alicí, Jingův uzel složil řetěz do bloku 277 314. Jingův uzel naslouchá transakcím, pokouší se vytěžit blok a také naslouchá blokům objevenými jinými uzly. Když Jingův uzel těžil, obdržel blok 277 315 od bitcoinové sítě. Příchod tohoto bloku ukončil soutěž o blok 277 315 a začal soutěž o vytvoření bloku 277 316.

Během předchozích 10 minut, zatímco Jingův uzel hledal řešení bloku 277 315, rovněž sbíral transakce pro přípravu dalšího bloku. Do současnosti nasbíral několik stovek transakcí v paměťovém úložišti. Po obdržení bloku 277 315 a jeho ověření, Jingův uzel také zkontroloval všechny transakce v paměťovém úložišti a odstranil ty, které byly vloženy do bloku 277 315. Jakékoliv transakce zůstaly v paměťovém úložišti jsou nepotvrzená a čekají pro zařazení do nového bloku.

Jingův uzel okamžitě sestavil nový prázdný blok, kandidátský blok na 277 316. Tento blok je zvaný kandidátský, protože ještě není platným blokem, protože ještě neobsahuje platný důkaz prací. Blok je platným pouze, pokud těžař uspěje v najití řešení algoritmu důkazu prací.

Stáří transakcí, poplatky a priorita

Pro sestavení kandidátského bloku, Jingův bitcoinový uzel vybere transakce z paměťového úložiště použitím prioritní metriky pro každou transakci a přidáním transakcí s nejvyšší prioritou jako prvních. Priorita transakcí je založena na "stáří" UTXO, který jsou utráceny na jejich vstupech, umožňující vstupům, které jsou velmi staré a mají vysokou hodnotu, aby byly upřednostněny před novějšími a menšími vstupy. Prioritní transakce jsou přenášeny bez poplatků, pokud je dostatek místa v bloku.

Priorita transakce je počítána jako součet součinů hodnot transakcí a stáří transakcí vydělený celkovou velikostí transakce:

```
Priorita = Suma (Hodnota vstup * Stáří transakce) / Velikost transakce
```

V této rovnici, hodnota vstupu je měřena v základních jednotkách satoshi (1/100 000 000 bitcoinu). Stáří UTXO je počet bloků, které uplynuly od jejího zaznamenání v blockchainu, měřící kolik bloků hluboko v blockchainu je. Velikost transakce je měřena v bytech.

Transakce je považována, že má "vysokou prioritu", pokud je její priorita vyšší než 57 600 000, což odpovídá jednomu bitcoinu (100 milionu satoshi) starému jeden den (144 bloků) pro transakci velikosti 250 bytů.

```
Vysoká priorita > 100 000 000 satoshi * 144 bloků / 250 bytů = 57 600 000
```

Prvních 50 kilobytů transakčního prostoru v bloku je vyčleněno pro transakce s vysokou prioritou. Jingův uzel zaplní prvních 50 kilobytů, upřednostňuje transakce s nejvyšší prioritou jako první, bez ohledu na poplatek. To umožňuje transakcím s vysokou prioritou zpracování i když nesou nulové poplatky.

Jingův těžební uzel poté zaplní zbytek bloku až po maximální velikost bloku (MAX_BLOCK_SIZE ve zdrojovém kódu), s transakcemi, které nesou alespoň minimální poplatky, upřednostňuje je podle nejvyššího poplatku za kilobyte transakce.

Pokud zůstane v bloku nějaké místo, Jingův těžební uzel se může rozhodnout ho zaplnit transakcemi bez poplatků. Někteří těžaři volí těžbu transakcí bez poplatků na základě nejlepšího úsilí. Jiní těžaři se mohou rozhodnout ignorovat transakce bez poplatků.

Jakákoliv transakce zbylá v paměťovém úložišti, poté co je blok naplněn, zůstane v úložišti pro vložení do dalšího bloku. Jak transakce zůstávají v paměťovém úložišti, jejich vstupy "stárnou" jak se UTXO, které utrácejí, dostávají hlouběji v blockchainu s přidáváním nových bloků na vrchol. Protože priorita transakcí závisí na stáří jejich vstupů, transakce zůstávající v úložišti stárnou a proto zvyšují svoji prioritu. Dokonce transakce bez poplatků může dosáhnout dostatečné priority, aby byla vložena do bloku zadarmo.

Bitcoinové transakce nemají čas vypršení. Transakce, která je platná nyní bude platná neomezenou dobu. Nicméně transakce je šířena po sítí jen jednou, přetrvává pouze tak dlouho, jak je držena v paměťovém úložišti těžebního uzlu. Když se těžební uzel restartuje, jeho paměťové úložiště je vyčištěno, protože se jedná o dočasnou, netrvalou formu uložení. Přestože platné transakce mohou být šířeny v sítí, pokud nejsou vykonány nemusejí nakonec zůstat v paměťovém úložišti žádného těžaře. Peněženkový software by měl takovéto transakce znovu zaslat nebo sestavit je s vyšším poplatkem, pokud nebyly úspěšně vykonány v rozumném čase.

Když Jingův uzel sestavil všechny transakce z paměťového úložiště, nový kandidátský blok měl 418 transakcí s celkovými transakčními poplatky 0.09094928 bitcoinu. Můžete vidět tento blok za použití Bitcoin core klientského rozhraní příkazová řádky jak vidíme v Blok 277 316.

```
$ bitcoin-cli getblockhash 277316
000000000000001b6b9a13b095e96db41c4a928b97ef2d944a9b31b2cc7bdc4

$ bitcoin-cli getblock
000000000000001b6b9a13b095e96db41c4a928b97ef2d944a9b31b2cc7bdc4
```

Example 3. Blok 277 316

```
{
    "hash" : "000000000000001b6b9a13b095e96db41c4a928b97ef2d944a9b31b2cc7bdc4",
    "confirmations" : 35561,
    "size" : 218629,
    "height" : 277316,
    "version" : 2,
    "merkleroot" :
"c91c008c26e50763e9f548bb8b2fc323735f73577effbc55502c51eb4cc7cf2e",
    "tx" : [
        "d5ada064c6417ca25c4308bd158c34b77e1c0eca2a73cda16c737e7424afba2f",
        "b268b45c59b39d759614757718b9918caf0ba9d97c56f3b91956ff877c503fbe",

        ... dalších 417 transakcí ...

       ],
    "time" : 1388185914,
    "nonce" : 924591752,
    "bits" : "1903a30c",
    "difficulty" : 1180923195.25802612,
    "chainwork" : "0000000000000000000000000000000000000000000934695e92aaf53afa1a",
    "previousblockhash" :
"000000000000002a7bbd25a417c0374cc55261021e8a9ca74442b01284f0569",
    "nextblockhash" :
"000000000000000010236c269dd6ed714dd5db39d36b33959079d78dfd431ba7"
}
```

Tvorba transakce

První transakce přidána do bloku je speciální transakcí, zvaná *mincetvorná transakce*. Tato transakce je sestavena Jingovým uzlem a je jeho odměnou za těžební úsilí. Jingův uzel vytvoří mincetvornou transakcí jako platbu své vlastní peněžence: "zaplať Jingově adrese 25,09094928 bitcoinů." Celková výše odměny, kterou Jing získá za vytěžení bloku je součet nově vzniklých mincí (25 nových bitcoinů) a transakčních poplatků (0,09094928) ze všech transakcí zahrnutých do bloku, jak zobrazuje Mincetvorná transakce:

```
$ bitcoin-cli getrawtransaction
d5ada064c6417ca25c4308bd158c34b77e1c0eca2a73cda16c737e7424afba2f 1
```

Example 4. Mincetvorná transakce

```
{
    "hex" :
"010000000100000000000000000000000000000000000000000000000000000000000000000000ffffffff0f
03443b0403858402062f503253482ffffffff0110c08d9500000000232102aa970c592640d19de03ff6f
329d6fd2eecb023263b9ba5d1b81c29b523da8b21ac00000000",
    "txid" : "d5ada064c6417ca25c4308bd158c34b77e1c0eca2a73cda16c737e7424afba2f",
    "version" : 1,
    "locktime" : 0,
    "vin" : [
        {
            "coinbase" : "03443b0403858402062f503253482f",
            "sequence" : 4294967295
        }
    ],
    "vout" : [
        {
            "value" : 25.09094928,
            "n" : 0,
            "scriptPubKey" : {
                "asm" :
"02aa970c592640d19de03ff6f329d6fd2eecb023263b9ba5d1b81c29b523da8b21OP_CHECKSIG",
                "hex" :
"2102aa970c592640d19de03ff6f329d6fd2eecb023263b9ba5d1b81c29b523da8b21ac",
                "reqSigs" : 1,
                "type" : "pubkey",
                "addresses" : [
                    "1MxTkeEP2PmHSMze5tUZ1hAV3YTKu2Gh1N"
                ]
            }
        }
    ],
    "blockhash" : "000000000000001b6b9a13b095e96db41c4a928b97ef2d944a9b31b2cc7bdc4",
    "confirmations" : 35566,
    "time" : 1388185914,
    "blocktime" : 1388185914
}
```

Na rozdíl od běžných transakcí, mincetvorná transakce nespotřebovává (neutrácí) UTXO na vstupech. Místo toho, má pouze jeden vstup zvaný *mincetvorný*, který vytváří bitcoiny z ničeho. Mincetvorná transakce má jeden výstup, platbu na těžařovu vlastní bitcoinovou adresu. Výstup mincetvorné transakce zasílá hodnotu 25,09094928 bitcoinů na bitcoinovou adresu těžaře, v tomto případě 1MxTkeEP2PmHSMze5tUZ1hAV3YTKu2Gh1N.

Mincetvorná odměna a poplatky

Pro sestavení mincetvorné transakce, Jingův uzel nejprve spočte celkové množství transakčních poplatků sečtením všech vstupů a výstupů těchto 418 transakcí, které byly přidány do bloku. Výpočet poplatku je následující:

```
Celkové poplatky = Suma(Vstupů) - Suma(Výstupů)
```

V bloku 277 316 celkové transakční poplatky jsou 0,09094928 bitcoinů.

Dále, Jingův uzel spočítá správnou odměnu za nový blok. Odměna je počítána v závislosti na výšce boku, začínající na 50 bitcoinech za blok a snižující se na polovinu každých 210 000 bloků. Protože výška tohoto bloku je 277 316, správná odměna je 25 bitcoinů.

Výpočet si lze prohlédnout ve funkci GetBlockSubsidy v Bitcoin Core klientovi, jak zobrazuje Výpočet odměny za blok - funkce GetBlockValue, Bitcoin Core klient, main.cpp.

Example 5. Výpočet odměny za blok - funkce GetBlockValue, Bitcoin Core klient, main.cpp

```
int64_t GetBlockValue(int nHeight, int64_t nFees)
{
    int halvings = nHeight / Params().SubsidyHalvingInterval();
    // Pokud není pravý bitový posun definován, nastaví mincetvornou odměnu na nulu
(vrátí jen poplatky)
    if (halvings >= 64)
        return nFees;

    int64_t nSubsidy = 50 * COIN;
    // Dotace je rozpůlena každých 210 000 bloků, k čemuž dochází přibližně každé 4
roky.
    nSubsidy >>= halvings;
    return nSubsidy + nFees;
}
```

Počáteční dotace je počítána v satoshi vynásobením 50 s konstantou COIN (100 000 000 satoshi). Toto počáteční nastavení odměny (nSubsidy) je 5 miliard satoshi.

Dále, funkce počítá počet půlení (halvings), které nastaly vydělením výšky aktuálního bloku půlícím

intervalem (SubsidyHalvingInterval). V případě bloku 277 316 a půlícím intervalem 210 000 bloků je výsledek 1 půlení.

Maximální dovolený počet půlení je 64, proto zdrojový kód uděluje nulovou mincetvornou odměnu (odměnou budou jen poplatky), pokud bylo dosaženo 64 nebo více půlení.

Dále, funkce používá operátor bitového posunu doprava, pro vydělení odměny (nSubsidy) dvěma v každém kole půlení. V případě bloku 277 316 sníží tento bitový posun doprava odměnu 5 miliard satoshi jednou (jedno půlení) a výsledek je 2,5 miliardy satoshi nebo 25 bitcoinů. Operátor bitového posunu doprava je použit, protože jeho použití pro dělení je výkonnější než použití dělení dvou celých nebo reálných čísel.

Nakonec, mincetvorná odměna (nSubsidy) je přidána k transakčním poplatkům (nFees), a výsledný součet je vrácen.

Struktura mincetvorné transakce

S tímto výpočtem, Jingův uzel sestaví mincetvornou transakci, která mu zaplatí 25,09094928 bitcoinů

Jak můžete vidět v Mincetvorná transakce, mincetvorná transakce má speciální tvar. Místo transakčního vstupu udávajícího předchozí UTXO k utracení, má "mincetvorný" vstup. Prozkoumali jsme transakční vstupy v [tx_in_structure]. Porovnejme vstup obvyklé transakce se vstupem mincetvorné transakce. Struktura "obvyklého" transakčního vstupu ukazuje strukturu vstupu obvyklé transakce, zatímco Struktura vstupu mincetvorné transakce ukazuje strukturu mincetvorného transakčního vstupu.

Table 1. Struktura "obvyklého" transakčního vstupu

Velikost	Pole	Popis
32 bytů	Transaction Hash	Ukazatel na transakci obsahující UTXO pro utracení
4 byty	Output Index	Indexové číslo UTXO k utracení, první je 0
1-9 bytů (VarInt)	Unlocking-Script Size	Velikost odemykacího skriptu (v bytech), který následuje
Proměnlivá	Unlocking-Script	Skript, který splňuje podmínky UTXO zamykacího skriptu.
4 byty	Sequence Number	V současnosti neaktivní, funkce náhrady transakcí, nastavena na 0xFFFFFFFF

Table 2. Struktura vstupu mincetvorné transakce

Velikost	Pole	Popis
32 bytů	Transaction Hash	Všechny bity jsou nula: žádný odkaz na haš transakce
4 byty	Output Index	Všechny bity jsou jedničky: 0xFFFFFFFF
1-9 bytů (VarInt)	Coinbase Data Size	Délka mincetvorných dat, od 2 do 100 bytů
Variable	Coinbase Data	Libovolná data obvykle používaná pro druhou nonci a těžební značky; ve v2 blocích musí začínat výškou bloku
4 byty	Sequence Number	Nastaveno na 0xFFFFFFFF

V mincetvorné transakci, první dvě pole jsou nastaveny na hodnoty, které nereprezentují odkaz na UTXO. Místo "haše transakce," první pole je vyplněno s 32 byty nastavenými na nula. "Index výstupu" je naplněn 4 byty nastavenými na 0xFF (255 desítkově). "Odemykací skript" je nahrazen mincetvornými daty, libovolnými daty pro použití těžaři.

Mincetvorná data

Mincetvorné transakce nemají položku odemykacího skriptu (scriptSig). Místo toho, je toto pole nahrazeno mincetvornými daty, které musejí být mezi 2 a 100 byty. Kromě prvních několika bytů, je zbytek mincetvorných dat k dispozici těžaři, pro použití jaké zvolí, jsou to libovolná data.

V základním bloku, například, Satoshi Nakamoto přidal text "The Times 3. ledna 2009 Kancléř na pokraji druhého záchranného balíčku pro banky." do mincetvorných dat, použil je jako důkaz data pro zprostředkování zprávy. Nyní, těžaři používají mincetvorná data pro vložení hodnoty druhé nonce a řetězec identifikující těžařskou skupinu, jak uvidíme v následujících částech.

Prvních několik bitů v mincetvorné transakci bylo libovolných, ale už není. Návrh na vylepšení bitcoinu 34 (BIP0034) zavedl bloky verze 2 (bloky s nastavením pole verze na 2), které musejí obsahovat index výšky bloku (jako vkládací operaci skriptu) na začátku mincetvorného pole.

V bloku 277 316 vidíme mincetvorné pole (viz Mincetvorná transakce), které je polem "Odemykacího skriptu" nebo scriptSig v transakčním vstupu, obsahuje hexadecimální hodnotu 03443b0403858402062f503253482f. Pojďme tuto hodnotu dekódovat.

První byte 03 je instrukcí skriptu pro vložení dalších tří bytů na zásobník (viz [tx_script_ops_table_pushdata]). Další tři byty 0x443b04 jsou výška bloku kódovaná v little-endian formátu (pozpátku, nejméně významné bity první). Obraťte pořadí bytů a výsledek je 0x043b44, což je 277 316 desítkově.

Dalších několik hexadecimálních číslic (03858402062) je použito pro zakódování *druhé nonce* (viz Řešení druhé nonce), nebo náhodné hodnoty, použité pro najití vhodného řešení důkazu prací.

Závěrečnou částí mincetvorných dat (2f503253482f) je ASCII kódovaný řetězec /P2SH/, který označuje že těžební uzel, který vytěžil tento blok, podporuje vylepšení platba haši skriptu (P2SH) definované v BIP0016. Zavedení schopností P2SH vyžadovalo "volbu" těžařů, zda podporují buďto BIP0016 nebo BIP0017, Ti, kteří podporovali BIP0016 implementaci, vkládali do svých mincetvorných dat /P2SH/. Ti, kteří podporovali BIP0014 implementaci, vkládali do svých mincetvorných dat p2sh/CHV. BIP0016 byl zvolen jako vítěz, mnoho těžařů pokračuje ve vkládání řetězce /P2SH/ do jejich mincetvorných dat, aby ukázali podporu této funkci.

[satoshi_words] používá knihovnu libbitcoin přestavenou v [alt_libraries] pro vyzvednutí mincetvorných dat ze základního bloku a zobrazení Satoshiho zprávy. Poznámka, knihovna libbitcoin obsahuje pevnou kopii základního bloku, takže zdrojový kód příkladu může vyzvednout základní blok přímo z knihovny.

1. Vyzvednutí mincetvorných dat ze základního bloku

```
/*
  Display the genesis block message by Satoshi.
*/
#include <iostream>
#include <bitcoin/bitcoin.hpp>

int main()
{
    // Create genesis block.
    bc::block_type block = bc::genesis_block();
    // Genesis block contains a single coinbase transaction.
    assert(block.transactions.size() == 1);
    // Get first transaction in block (coinbase).
    const bc::transaction_type& coinbase_tx = block.transactions[0];
    // Coinbase tx has a single input.
    assert(coinbase_tx.inputs.size() == 1);
    const bc::transaction_input_type& coinbase_input = coinbase_tx.inputs[0];
    // Convert the input script to its raw format.
    const bc::data_chunk& raw_message = save_script(coinbase_input.script);
    // Convert this to an std::string.
    std::string message;
    message.resize(raw_message.size());
    std::copy(raw_message.begin(), raw_message.end(), message.begin());
    // Display the genesis block message.
    std::cout << message << std::endl;
    return 0;
}
```

Zkompiluje zdrojový kód s GNU C++ kompilátorem a spustíme výsledný spustitelný soubor, jak ukazuje

[satoshi_words_run].

1. Kompilace a spuštění ukázkového zdrojového kódu satoshi-words

```
$ # Kompilace zdrojových kódů
$ g++ -o satoshi-words satoshi-words.cpp $(pkg-config --cflags --libs libbitcoin)
$ # Spuštění spustitelného souboru
$ ./satoshi-words
^D    <GS>^A^DEThe Times 03/Jan/2009 Chancellor on brink of second bailout for banks
```

Vytvoření hlavičky bloku

Pro vytvoření hlavičky bloku, těžební uzel potřebuje zaplnit šest polí uvedených v Struktura hlavičky bloku.

Table 3. Struktura hlavičky bloku

Velikost	Pole	Popis
4 byty	Verze	Číslo verze sledující aktualizaci software / protokolu
32 bytes	Previous Block Hash	Odkaz na haš předchozího (rodičovského) bloku v řetězu
32 bytů	Merkle Root	Haš kořene merkle stromu tohoto bloku transakcí
4 byty	Timestamp	Přibližný čas vytvoření tohoto bloku (sekundy dle Unix konvence)
4 byty	Difficulty Target	Obtížnostní cíl tohoto bloku pro algoritmus důkazu prací
4 byty	Nonce	Čítač použity pro algoritmus důkazu prací

V čase, kdy blok 277 316 byl vytěžen, číslo verze popisující strukturu bloku je verze 2, které je zakódované v little endian formátu ve 4 bytech jako 0x02000000.

Dále, těžební uzel potřebuje přidat "haš předchozího bloku." Je to haš hlavičky bloku 277 315, předchozí blok získaný ze sítě, který Jingův uzel přijal a označil ho rodičem kandidátského bloku 277 316. Haš hlavičky bloku pro blok 277 315 je:

```
000000000000000002a7bbd25a417c0374cc55261021e8a9ca74442b01284f0569
```

Další krok je shrnout všechny transakce do merkle stromu, za účelem přidání kořene merkle stromu do hlavičky bloku. Mincetvorná transakce je zapsána jako první transakce v bloku. Dále je přidáno dalších 418 transakcí, celkově 419 transakcí v bloku. Jak vidíme v [merkle_trees], musí být sudý počet listových uzlů ve stromu, takže poslední transakce je zdvojena, vytváří 420. uzel, který obsahuje haš této transakce. Haše transakcí jsou spojeny po dvojicích a postupně vytvářejí úrovně stromu, dokud nejsou shrnuty v jeden uzel zvaný "kořen" stromu. Kořen Merkle stromu shrnuje všechny transakce v jednu 32 bytovou hodnotu, kterou můžeme vidět jako "merkle root" v Blok 277 316 a zde:

```
c91c008c26e50763e9f548bb8b2fc323735f73577effbc55502c51eb4cc7cf2e
```

Těžební uzel přidá 4-bytovou časovou značku, zakódovanou jako Unixovou časovou značku, která je založena na počtu sekund uplynulých od 1. ledna 1970, půlnoci UTC/GMT. Čas 1388185914 odpovídá pátku, 27. prosince 2013, 23:11:54 UTC/GMT.

Uzel poté vyplňuje obtížnostní cíl, který určuje požadovanou obtížnost důkazu prací, aby se blok stal platným. Obtížnost je uložena v bloku jako "difficulty bits" metrice, která je mantisa-exponent zakódovaným cílem. Kódování má 1-bytový exponent následovaný 3-byty mantisy (koeficient). V bloku 277 316, například, obtížnostní bitová hodnota je 0x1903a30c. První část 0x19 je hexadecimálně exponent, zatímco další část 0x03a30c je koeficient. Koncept stanovování obtížnostního cíle je vysvětlen v Obtížnostní cíl a jeho změna a reprezentace "obtížnostních bitů" je vysvětlena v Reprezentace obtížnosti.

Poslední pole je nonce, která je inicializována na nula.

Se všemi vyplněními položkami je hlavička bloku nyní úplná a postup těžby může začít. Cílem je najít hodnotu nonce takovou, aby haš hlavičky bloku byl nižší než obtížnostní cíl. Těžební uzel potřebuje testovat miliardy nebo biliony hodnot noncí než je nalezena nonce, která splňuje požadavek.

Těžba bloku

Nyní kandidátský blok byl sestaven Jingovým uzlem, je čas pro Jingovu hardwarovou těžební soupravu, aby "těžila" blok, hledala řešení algoritmu důkazu prací, aby udělala blok platným. V této knize jsme studovali kryptografické hašovací funkce použité v různých aspektech bitcoinové systému. Hašovací funkce SHA256 se používá při průběhu bitcoinové těžby.

Jednoduše řečeno, těžba je postup opakovaného hašování hlaviček bloků, měněním jednoho parametru, dokud výsledný haš neodpovídá požadovanému cíli. Výsledek hašovací funkce nelze stanovit předem, ani nelze vytvořit vzor, který vyrobí konkrétní hodnotu haše. tato vlastnost hašovacích funkcí znamená, že jediným způsobem jak vyrobit haš odpovídající požadovanému cíli, je zkoušet znova znova, náhodně měnit vstup, dokud požadovaný výsledný haš se náhodou neobjeví.

Algoritmus důkazu prací

Hašovací algoritmus bere datový vstup libovolné délky a vytváří výsledek pevná délky

deterministickým způsobem, digitální otisk vstupu. Pro jakýkoliv konkrétní vstup, výsledný haš bude vždy totožný a může být snadno spočítán a ověřen kýmkoliv implementujícím stejný hašovací algoritmus. Klíčovou charakteristikou kryptografického hašovacího algoritmu, je že je prakticky nemožné najít dva různé vstupy, které vytvářejí stejný otisk. Jako důsledek je prakticky nemožné k vybrat vstup takovým způsobem, aby vytvořil požadovaný otisk, jinak, než zkoušením náhodných vstupů.

U SHA256 výstup je vždy 256 bitů dlouhý, bez ohledu na velikost vstupu. V příkladu Příklad SHA256 použijeme Python interpret pro výpočet SHA256 haše fráze "I am Satoshi Nakamoto."

Example 6. Příklad SHA256

```
$ python
```

```
Python 2.7.1
>>> import hashlib
>>> print hashlib.sha256("I am Satoshi Nakamoto").hexdigest()
5d7c7ba21cbbcd75d14800b100252d5b428e5b1213d27c385bc141ca6b47989e
```

Příklad SHA256 ukazuje výsledek výpočtu haše z "I am Satoshi Nakamoto": 5d7c7ba21cbbcd75d14800b100252d5b428e5b1213d27c385bc141ca6b47989e. Toto 256-bitové číslo je *haš* nebo *výtah* z fráze a závisí na každé části fráze. Přidáním jednoho písmena, interpunkčního znaménka nebo jiného znaku vyrobíme rozdílný haš.

Nyní, pokud změníme frázi, můžeme očekávat naprosto odlišné haše. Zkusme přidat číslo na konec fráze, za použití jednoduchého skriptu v Pythonu v [sha256_example_generator].

Skript tvoří mnoho SHA256 hašů zvyšováním nonce

```
# example of iterating a nonce in a hashing algorithm's input

import hashlib

text = "I am Satoshi Nakamoto"

# iterate nonce from 0 to 19
for nonce in xrange(20):

    # add the nonce to the end of the text
    input = text + str(nonce)

    # calculate the SHA-256 hash of the input (text+nonce)
    hash = hashlib.sha256(input).hexdigest()

    # show the input and hash result
    print input, '=>',  hash
```

Spuštěním vyrobíme haše několika frází, odlišných přidáním čísla na konec textu. Zvyšováním čísla můžeme dostat odlišné haše, jak je zobrazeno v Výstup SHA256 skriptu vytvářejícího mnoho hašů zvyšováním nonce..

Example 7. Výstup SHA256 skriptu vytvářejícího mnoho hašů zvyšováním nonce.

```
$ python hash_example.py
```

```
I am Satoshi Nakamoto0 => a80a81401765c8eddee25df36728d732...
I am Satoshi Nakamoto1 => f7bc9a6304a4647bb41241a677b5345f...
I am Satoshi Nakamoto2 => ea758a8134b115298a1583ffb80ae629...
I am Satoshi Nakamoto3 => bfa9779618ff072c903d773de30c99bd...
I am Satoshi Nakamoto4 => bce8564de9a83c18c31944a66bde992f...
I am Satoshi Nakamoto5 => eb362c3cf3479be0a97a20163589038e...
I am Satoshi Nakamoto6 => 4a2fd48e3be420d0d28e202360cfbaba...
I am Satoshi Nakamoto7 => 790b5a1349a5f2b909bf74d0d166b17a...
I am Satoshi Nakamoto8 => 702c45e5b15aa54b625d68dd947f1597...
I am Satoshi Nakamoto9 => 7007cf7dd40f5e933cd89fff5b791ff0...
I am Satoshi Nakamoto10 => c2f38c81992f4614206a21537bd634a...
I am Satoshi Nakamoto11 => 7045da6ed8a914690f087690e1e8d66...
I am Satoshi Nakamoto12 => 60f01db30c1a0d4cbce2b4b22e88b9b...
I am Satoshi Nakamoto13 => 0ebc56d59a34f5082aaef3d66b37a66...
I am Satoshi Nakamoto14 => 27ead1ca85da66981fd9da01a8c6816...
I am Satoshi Nakamoto15 => 394809fb809c5f83ce97ab554a2812c...
I am Satoshi Nakamoto16 => 8fa4992219df33f50834465d3047429...
I am Satoshi Nakamoto17 => dca9b8b4f8d8e1521fa4eaa46f4f0cd...
I am Satoshi Nakamoto18 => 9989a401b2a3a318b01e9ca9a22b0f3...
I am Satoshi Nakamoto19 => cda56022ecb5b67b2bc93a2d764e75f...
```

Každá fráze vytváří úplně odlišný výsledný haš. Vypadají naprosto náhodné, ale můžeme zopakovat stejný výsledek z příkladu na jakémkoliv počítači s Pythonem a uvidíme přesně ty samé haše.

Číslo použité jako proměnná v takovémto scénáři je nazváno *nonce*. Nonce je použita pro změnu výstupu kryptografické funkce, v tomto případě pro odlišený otisk fráze pomocí SHA256.

Abychom vytvořili výzvu pro tento algoritmus, stanovme si nějaký libovolný cíl: najděme frázi, která vytváří hexadecimální haš začínající s nulou. Naštěstí to není těžké! Výstup SHA256 skriptu vytvářejícího mnoho hašů zvyšováním nonce. ukazuje frázi "I am Satoshi Nakamoto13" vytvářející haš+0ebc56d59a34f5082aaef3d66b37a661696c2b618e62432727216ba9531041a5+, který splňuje naši podmínku. Trvalo to 13 pokusů, než byla nalezena. V pojmech pravděpodobnosti, pokud výstup hašovací funkce je rovnoměrně rozložen. můžeme očekávat výsledek 0 jako hexadecimální předponu jednou každých 16 hašů (jedno ze 16 hexadecimálních číslic 0 až F). V číselných pojmech, hledáme haš hodnotu menší než 0x1000000000000000000000000000000000000000000000000000000000000000. Nazýváme ji prahovou hodnotou *cíle* a snažíme se najít haš, který je číselně *nižší než cíl*. Pokud snížíme cíl, úkol nelezení haše, který je menší než cíl se stává více a více obtížným.

Abychom dali jednoduché podobenství, představte si hru, kde hráči házejí párem kostek opakovaně,

snaží se hodit méně než je určený cíl. V prvním kole je cíl 12. Pokud nehodíte dvě šestky, vyhráváte. V dalším kole je cíl 11, Hráči musejí hodit 10 nebo méně pro vítězství, opět lehký úkol. Řekněme, že po pár kolech později je cíl snížen na 5. Nyní více než polovina hodů kostkami dá v součtu více než 5 a proto bude neplatných. Je vyžadováno exponenciálně více hodů kostkami, čím nižší je cíl. Nakonec cíl je 2 (nejmenší možný) a pouze jeden hod z 36, nebo 2 % z nich, vytvoří vítězný výsledek.

V Výstup SHA256 skriptu vytvářejícího mnoho hašů zvyšováním nonce. vítězná "nonce" je 13 a tento výsledek může být potvrzen kýmkoliv nezávisle. Kdokoliv může přidat číslo 13 jako příponu fráze "I am Satoshi Nakamoto" a spočítat haš, ověřit, že je menší než cíl. Úspěšný výsledek je také důkazem prací, protože to dokazuje, že jsme odvedli práci k nalezení nonce. Přestože k ověření je zapotřebí pouze jeden výpočet haše, zabralo nám 13 výpočtů haše nalezení vhodné nonce. Pokud máme nižší cíl (vyšší obtížnost) bude potřeba více výpočtu hašů k nalezení vhodné nonce, ale pouze jeden výpočet haše pro ověření kýmkoliv. Navíc, znalostí cíle můžeme odhadnout obtížnost za použití statistiky a proto víme kolik práce je nutné k najití takovéto nonce.

Bitcoinový důkaz prací je velmi podobný k výzvě zobrazené v Výstup SHA256 skriptu vytvářejícího mnoho hašů zvyšováním nonce.. Těžař sestavuje kandidátský blok naplněný transakcemi. Následně těžař spočítá haš hlavičky bloku a podívá se, zda je menší než současný cíl. Pokud haš není menší než cíl, těžař změní nonci (obvykle ji zvýší o jedna) a zkouší znova. Při současné obtížnosti bitcoinové sítě těžaři musí vyzkoušet biliardy noncí před nalezením nonce, která vede k dostatečně nízkému haši bloku.

Velmi zjednodušený algoritmus důkazu prací je implementován v Pythonu v Zjednodušená implementace důkazu prací.

Example 8. Zjednodušená implementace důkazu prací

```
#!/usr/bin/env python
# example of proof-of-work algorithm

import hashlib
import time

max_nonce = 2 ** 32 # 4 billion

def proof_of_work(header, difficulty_bits):

    # calculate the difficulty target
    target = 2 ** (256-difficulty_bits)

    for nonce in xrange(max_nonce):
        hash_result = hashlib.sha256(str(header)+str(nonce)).hexdigest()

        # check if this is a valid result, below the target
        if long(hash_result, 16) < target:
            print "Success with nonce %d" % nonce
```

```
            print "Hash is %s" % hash_result
            return (hash_result,nonce)

    print "Failed after %d (max_nonce) tries" % nonce
    return nonce


if __name__ == '__main__':

    nonce = 0
    hash_result = ''

    # difficulty from 0 to 31 bits
    for difficulty_bits in xrange(32):

        difficulty = 2 ** difficulty_bits
        print "Difficulty: %ld (%d bits)" % (difficulty, difficulty_bits)

        print "Starting search..."

        # checkpoint the current time
        start_time = time.time()

        # make a new block which includes the hash from the previous block
        # we fake a block of transactions - just a string
        new_block = 'test block with transactions' + hash_result

        # find a valid nonce for the new block
        (hash_result, nonce) = proof_of_work(new_block, difficulty_bits)

        # checkpoint how long it took to find a result
        end_time = time.time()

        elapsed_time = end_time - start_time
        print "Elapsed Time: %.4f seconds" % elapsed_time

        if elapsed_time > 0:

            # estimate the hashes per second
            hash_power = float(long(nonce)/elapsed_time)
            print "Hashing Power: %ld hashes per second" % hash_power
```

Spustíme tento zdrojový kód, nastavíme požadovanou obtížnost (v bitech, kolik vedoucích bitů musí být nulových) a sledujeme jak dlouho trvá počítači nalézt řešení. V Spuštění příkladu důkazu prací s různými obtížnostmi, vidíme jak to pracuje na průměrném laptopu.

Example 9. Spuštění příkladu důkazu prací s různými obtížnostmi

```
$ python proof-of-work-example.py*
```

```
Difficulty: 1 (0 bits)

[...]

Difficulty: 8 (3 bits)
Starting search...
Success with nonce 9
Hash is 1c1c105e65b47142f028a8f93ddf3dabb9260491bc64474738133ce5256cb3c1
Elapsed Time: 0.0004 seconds
Hashing Power: 25065 hashes per second
Difficulty: 16 (4 bits)
Starting search...
Success with nonce 25
Hash is 0f7becfd3bcd1a82e06663c97176add89e7cae0268de46f94e7e11bc3863e148
Elapsed Time: 0.0005 seconds
Hashing Power: 52507 hashes per second
Difficulty: 32 (5 bits)
Starting search...
Success with nonce 36
Hash is 029ae6e5004302a120630adcbb808452346ab1cf0b94c5189ba8bac1d47e7903
Elapsed Time: 0.0006 seconds
Hashing Power: 58164 hashes per second

[...]

Difficulty: 4194304 (22 bits)
Starting search...
Success with nonce 1759164
Hash is 0000008bb8f0e731f0496b8e530da984e85fb3cd2bd81882fe8ba3610b6cefc3
Elapsed Time: 13.3201 seconds
Hashing Power: 132068 hashes per second
Difficulty: 8388608 (23 bits)
Starting search...
Success with nonce 14214729
Hash is 000001408cf12dbd20fcba6372a223e098d58786c6ff93488a9f74f5df4df0a3
Elapsed Time: 110.1507 seconds
Hashing Power: 129048 hashes per second
Difficulty: 16777216 (24 bits)
Starting search...
Success with nonce 24586379
Hash is 0000002c3d6b370fccd699708d1b7cb4a94388595171366b944d68b2acce8b95
```

```
Elapsed Time: 195.2991 seconds
Hashing Power: 125890 hashes per second

[...]

Difficulty: 67108864 (26 bits)
Starting search...
Success with nonce 84561291
Hash is 0000001f0ea21e676b6dde5ad429b9d131a9f2b000802ab2f169cbca22b1e21a
Elapsed Time: 665.0949 seconds
Hashing Power: 127141 hashes per second
```

Jak můžete vidět, zvyšující se obtížnost o 1 bit způsobuje exponenciální nárůst času nutného k najití řešení. Pokud uvažujete celý 256-bitový číselný prostor, pokaždé když omezíme jeden další bit na nulu, snížíme prohledávaný prostor na polovinu. V Spuštění příkladu důkazu prací s různými obtížnostmi, je třeba 84 milionů hašovacích pokusí pro nalezení nonce vytvářející haš s 26 nulovými vedoucími bity. Dokonce při rychlosti větší než 120 000 hašů za sekundu, je třeba 10 minut na běžném laptopu pro najití řešení.

V čase psaní, síť se pokouší najít blok, jehož haš hlavičky je menší než 000000000000004c296e6376db3a241271f43fd3f5de7ba18986e517a243baa7. Jak můžete vidět, je tam mnoho nul na začátku haše, což znamená, že přijatelný interval hašů je mnohem menší, proto je obtížnější najít platný haš. Bude potřeba v průměru 150 biliard výpočtu hašů za sekundu, aby síť objevila další blok. Vypadá to jako nemožný úkol, ale naštěstí síť má výpočetní výkon 100 pentahašů za sekundu (PH/sec), který bude schopen najít blok v průměru za 10 minut.

Reprezentace obtížnosti

V Blok 277 316 vidíme, že blok obsahuje obtížnostní cíl, v notaci zvané "obtížnostní bity" nebo jen "bity", který má v bloku 277 316 hodnotu 0x1903a30c. Tento zápis vyjadřuje obtížnostní cíl v koeficient/exponent formátu, ve kterém jsou první dvě hexadecimální číslice pro exponent a zbylých šest hexadecimálních číslic je koeficient. V tomto bloku, je exponent 0x19 a koeficient je 0x03a30c.

Vzorec pro výpočet obtížnostního cíle z této reprezentace je:

```
cíl = koeficient * 2^(8 * (exponent − 3))
```

Použitím tohoto vzorce a obtížnostní bitové hodnoty 0x1903a30c, získáme:

```
cíl = 0x03a30c * 2^(0x08 * (0x19 - 0x03))^

=> cíl = 0x03a30c * 2^(0x08 * 0x16)^

=> cíl = 0x03a30c * 2^0xB0^
```

což je dekadicky

```
=> cíl = 238,348 * 2^176^

=> cíl = 22,829,202,948,393,929,850,749,706,076,701,368,331,072,452,018,388,575,715,328
```

přepneme zpět do hexadecimálního zápisu:

```
=> cíl = 0x00000000000003A30C0000000000000000000000000000000000000000000000
```

To znamená, že platný blok pro výšku 277 316 je takovým blokem, jehož haš hlavičky je menší než tento cíl. Binárně toto číslo musí mít více než 60 prvních bitů nastavených na nulu. S touto úrovní obtížnosti, jednotlivý těžař zpracovávající 1 bilion hašů za sekundu (1 tera-haš za sekundu nebo 1TH/s) najde řešení jednou za 8 496 bloků nebo každých 59 dní v průměru.

Obtížnostní cíl a jeho změna

Jak vidíme, cíl určuje obtížnost a proto ovlivňuje jak dlouho bude trvat nalezení řešení algoritmu důkazu prací. To vede ke zřejmé otázce: Proč je obtížnost nastavitelná, kdo ji nastavuje a jak?

Bitcoinové bloky jsou vytvářeny v průměru každých 10 minut. To je jeden tip bitcoinového srdce, který podporuje četnost vydávání nových mincí a rychlost vypořádání transakcí. Měl by zůstat konstantní nejen z krátkodobého hlediska, ale i v průběhu desítek let. Během této doby se očekává, že výpočetní výkon počítačů bude se bude zvyšovat rychlým tempem. Navíc, počet účastníku těžby a použitých počítačů se bude stále měnit. Aby bylo zachováno vytváření bloku každých 10 minut, obtížnost těžby musí být nastavitelná, aby se vypořádala s těmito změnami. Ve skutečnosti, obtížnost je dynamický parametr, který se periodicky nastavuje, aby splnil cíl 10-minutových bloků. Jednoduše řečeno, obtížnostní cíl je nastaven tak, aby jakákoliv těžební síla vedla k 10-minutových intervalům mezi bloky.

Jak tedy takovéto přizpůsobení dělat v plně decentralizované síti? Změna obtížnostního cíle nastává automaticky nezávisle na každém úplném uzlu. Každých 2016 bloků, všechny uzly mění obtížnost důkazu prací. Rovnice pro změnu obtížnosti měří čas spotřebovaný pro najití posledních 2016 bloků a porovnává ho s očekávaným časem 20 160 minut (dva týdny měřeno požadovanými 10 minutovými intervaly mezi bloky). Je spočítám poměr mezi aktuálním časovým obdobím a požadovaným časovým obdobím a odpovídající úprava obtížnosti (nahoru nebo dolů) je provedena. Jednoduše řečeno: pokud síť hledá bloky rychleji než každých 10 minut, obtížnost se zvyšuje. Pokud je objevování bloků pomalejší než se očekávalo, obtížnost se zvyšuje.

Může být vyjádřeno následující rovnicí:

```
nová obtížnost = stará obtížnost * (skutečný čas posledních 2016 bloků / 20160 minut)
```

Změna obtížnosti těžby důkazu prací - CalculateNextWorkRequired() v pow.cpp ukazuje zdrojový kód použitý v Bitcoin Core klientovi.

Example 10. Změna obtížnosti těžby důkazu prací - CalculateNextWorkRequired() v pow.cpp

```
// Omezení kroku úpravy
 int64_t nActualTimespan = pindexLast->GetBlockTime() - nFirstBlockTime;
 LogPrintf(" nActualTimespan = %d before bounds\n", nActualTimespan);
 if (nActualTimespan < params.nPowTargetTimespan/4)
     nActualTimespan = params.nPowTargetTimespan/4;
 if (nActualTimespan > params.nPowTargetTimespan*4)
     nActualTimespan = params.nPowTargetTimespan*4;

 // Změna cíle
 const arith_uint256 bnPowLimit = UintToArith256(params.powLimit);
 arith_uint256 bnNew;
 arith_uint256 bnOld;
 bnNew.SetCompact(pindexLast->nBits);
 bnOld = bnNew;
 bnNew *= nActualTimespan;
 bnNew /= params.nPowTargetTimespan;

 if (bnNew > bnPowLimit)
     bnNew = bnPowLimit;
```

NOTE Zatímco k nastavování obtížnosti dochází každých 2016 bloků, kvůli chybné konstantně v originálním Bitcoin Core klientovi, je založená na celkovém času předchozích 2015 bloků (místo 2016 bloků), výsledná změna obtížnosti je o 0,05% vyšší než by měla být.

Parametry Interval (2,016 bloků) a TargetTimespan (dva týdny jako 1 209 600 sekund) jsou definovány v *chainparams.cpp*.

Abychom se vyhnuli značným změním obtížnosti, změna obtížnosti musí být menší než čtyřnásobná (4x) v jednom cyklu. Pokud je požadovaná změna obtížnosti vyšší než činitel čtyři, bude změněna maximálně čtyřnásobně a ne více. Další změna obtížnosti bude provedena v dalším období, proto nerovnováha bude pokračovat dalších 2016 bloků. Proto velké rozdíly mezi hašovací silou a obtížností mohou spotřebovat několik 2016-blokových cyklů k jejich vyrovnání.

TIP	Obtížnost nalezení bitcoinového bloku je přibližně "10 minut zpracování" celou sítí, založeno na čase nalezení předchozích 2016 bloků, měněno každých 2016 bloků.

Poznámka, obtížnostní cíl je nezávislý na počtu transakcí nebo jejich hodnotě. To znamená, že množství hašovací síly a tedy elektrické energie vydané na zabezpečení bitcoinu je zcela nezávislé na počtu transakcí. Bitcoin se může rozšiřovat, dosáhnout širšího přijetí, a zůstat bezpečný bez zvýšení hašovací síly oproti dnešní úrovni. Zvýšení hašovací síly reprezentuje tržní síly, jak noví těžaři vstupují na trh, aby soutěžili o odměnu. Jakmile je dostatek hašovací síly pod kontrolou těžařů, kteří čestně soutěží o odměnu, je to dostatečné k prevenci útoků "pokusů o převzetí" a proto je to dostatečné k zabezpečení bitcoinu.

Cílová obtížnost se blízce vztahuje k ceně elektrické energie a směnnému kurzu bitcoinu na klasické měny, které se používají pro platbu elektřiny. Vysoce výkonné těžební systémy jsou tak účinné, jak je to možné se současnou úrovní zpracování křemíku, přetvářejí elektrickou energii na hašovací výpočty s nejvyšší možnou účinností. Hlavním vlivem na těžební trh je cena jedné kilowatthodiny v bitcoinu, protože ta určuje ziskovost těžby a je výzvou k vstoupení nebo opuštění těžebního trhu.

Úspěšná těžba bloku

Jak jsme viděli dříve, Jingův uzel vytvořil kandidátský blok a připravil ho pro těžbu. Jing má několik hardwarových těžebních souprav s aplikačně specifickými integrovanými obvody (ASIC), kde stovky tisíc integrovaných obvodů počítají paralelně algoritmus SHA256 v neuvěřitelné rychlosti. Tyto specializované stroje jsou spojeny s těžebním uzlem přes USB. Dále, těžební uzel běžící na Jingově stolním počítači přenáší hlavičky bloků do jeho těžebního hardware, který začíná testovat biliony noncí za sekundu.

Téměř 11 minut po začátku těžby bloku 277 316 jeden z těžebních strojů našel řešení a zaslal je zpět těžebnímu uzlu. Po vložení do hlavičky bloku, nonce 4 215 469 401 vytvořila haš bloku:

```
0000000000000002a7bbd25a417c0374cc55261021e8a9ca74442b01284f0569
```

který je menší než cíl:

```
00000000000000003A30C00000000000000000000000000000000000000000000
```

Okamžitě, Jingův těžební uzel přenesl blok všem svým klientským uzlům. Oni ho přijali, ověřili a poté šířili tento nový blok. Jak blok ve vlně proběhl celou sítí, každý uzel si ho přidal do vlastní kopie blockchainu, rozšiřující ho na novou výšku 277 316 bloků. Jakmile těžební uzly obdrželi platný blok, ukončili své úsilí v hledání bloku stejné výšky a začali počítat další blok v řetězu.

V další části se podíváme na postup, jakým každý uzel ověřuje blok a hledá nejdelší řetěz, vytváří shodu na tvaru decentralizovaného blockchainu.

Ověření nového bloku

Třetí krok v mechanizmu bitcoinové shody je nezávislé ověřování každého nového bloku každým uzlem v síti. Jak se nově vyřešený blok posouvá skrz síť, každý uzel provádí posloupnost testů k jeho ověření, předtím, než ho rozšíří svým klientským uzlům. To zajišťuje, že pouze platné bloky jsou šířeny po sítí. Nezávislé ověření také zajišťuje, že těžaři, kteří se chovají čestně dosáhnou zařazení svých bloků do blockchainu a tedy vydělají odměnu. Těžařům, kteří se chovají nečestně, jsou jejich bloky odmítány a nejenže přijdou o odměnu ale také vyplýtvali úsilí vydané k najití řešení důkazu prací, tedy vynaložených nákladů na elektrickou energii bez náhrady.

Když uzel obdrží nový blok, začne ověřovat blok kontrolou dlouhého seznamu kriterií, které musejí být splněny, jinak je blok odmítnut. Tato kritéria můžete vidět v Bitcoin Core klientovi ve funkcích CheckBlock a CheckBlockHeader a obsahují:

- Datová struktura bloku je syntakticky platná
- Hlavička bloku je nižší než obtížnostní cíl (vynucuje důkaz prací)
- Časová značka bloku je nižší než dvě hodiny do budoucna (dovoluje chyby času)
- Velikost bloku je v akceptovatelném rozmezí
- První transakce (a pouze první) je mincetvorná
- Všechny transakce v bloku jsou platné za použití kritérií platnosti transakcí probrané v Nezávislé ověřování transakcí

Nezávislé ověření každého nového bloku každým uzel v sítí zajišťuje, že těžaři nemohou podvádět. V předchozí sekci jsme viděli jak těžaři zapisují transakci, která je odměňuje nově vytvořenými bitcoiny daným blokem a vyzvedávají transakční poplatky. Proč těžař nenapíše transakci ve svůj prospěch na tisíc bitcoinu místo správné odměny? Protože každý uzel ověřuje bloky podle stejných pravidel. Neplatná mincetvorná transakce zneplatní celý blok, což povede k odmítnutí bloku a tato transakce se nikdy nestane součástí účetního systému. Těžaři musí sestavit perfektní blok, založený na sdílených pravidlech, která následují všechny uzly, a vytěžit je správným řešením důkazu prací. Aby to udělali, utratí mnoho elektrické energie při těžbě, a pokud podvádějí, všechna elektrická energie a úsilí je vyplýtváno. To je proč nezávislé ověřování je klíčovým prvkem decentralizované shody.

Sestavování a vybírání řetězů bloků

Poslední krok v mechanizmu bitcoinové decentralizované shody je skládání bloku do řetězů a výběr řetězu s nejvyšším důkazem prací. Jakmile uzel ověří nový blok, pokusí se sestavit řetěz spojující tento blok s existujícím blockchainem.

Uzly spravují tři množiny bloků: uzly spojené s hlavním blockchainem, uzly tvořící větve z hlavního blockchainu (secondary chains) a nakonec uzly, které nemají známého rodiče ve známém řetězu (sirotci). Neplatné bloky jsou odmítnuty jakmile jedno z ověřovacích kritérií selže a proto nejsou vloženy do žádného řetězu.

"Hlavní blok" je v jakoukoliv chvíli ten řetěz bloků, který má nejvyšší součet obtížností s ním spojený. Ve většině případů je to také řetěz s nejvyšším počtem bloků, pokud zde nejsou dva stejně dlouhé řetězy a jeden má vyšší důkaz prací. Hlavní řetěz má také větve s bloky, které jsou sourozenci bloků v hlavním řetězu. Tyto bloky jsou platné, ale nejsou součástí hlavního řetězu. Jsou uchovávány pro budoucí odkazy, pro případ, že jeden řetěz je rozšířen, takže předstihne hlavní řetěz v obtížnosti. V další části (Větvení blockchainu), uvidíme jak vedlejší řetězy vznikají jako výsledek téměř současného vytěžení bloků ve stejné výšce.

Když je nový blok přijat, uzel se pokusí jej umístit do existujícího blockchainu. Uzel se podívá na položku bloku "haš předchozího bloku", která je odkazem na rodiče nového bloku. Poté se uzel pokusí najít rodiče v existujícím blockchainu. Většinou, rodič bude vrcholem hlavního řetězu, nový blok rozšíří tento hlavní řetěz. Například nový blok 277 316 měl odkaz na haš svého rodiče bloku 277 315. Většina uzlů, které přijala 277 316 již měla blok 277 315 na vrcholu jejich hlavního řetězu a proto spojí nový blok a prodlouží řetěz.

Občas, jak uvidíme v Větvení blockchainu, nové bloky rozšiřují řetěz, který není hlavním řetězem. V tomto případě, uzel přidá nový blok na vedlejší řetěz a rozšíří ho, porovná obtížnost vedlejšího a hlavního řetězu. Pokud vedlejší řetěz má vyšší součet obtížností než hlavní řetěz, uzel *zotaví* vedlejší řetěz, což znamená, že vybere vedlejší řetěz jako nový hlavní řetěz a ze starého hlavního řetězu udělá vedlejší řetěz. Pokud je uzel těžař, začne stavět blok prodlužující tento nový, delší, řetěz.

Pokud platný blok je přijat a není nalezen rodič v existujících řetězech, tento blok je považován za "sirotka." Sirotčí bloky jsou uloženy v úložišti sirotčích bloků, kde zůstávají dokud není obdržen jejich otec. Jakmile je otec obdržen a spojen do existujících řetězů, sirotek je vyzvednut z úložiště sirotků a spojen se svým otcem, stává se částí bloku. Sirotčí bloky se obvykle objevují, když dva bloky byly vytěženy v krátkém čase po sobě a jsou přijaty v obráceném pořadí (dítě před rodičem).

Výběrem řetězu s nejvyšší obtížností, všechny uzly nakonec dosáhnout shody s celou sítí. Dočasné nesrovnalosti mezi uzly jsou vyřešeny nakonec jak jsou přidány další důkazy prací, rozšiřující jeden z možných řetězů. Těžební uzly "volí" svojí těžební silou výběrem, který řetěz rozšíří dalším blokem. Když vytěží nový blok a rozšíří řetěz, nový blok sám o sobě reprezentuje jejich volbu.

V další části uvidíme jak nesrovnalosti mezi soupeřícími řetězy (větvemi) jsou vyřešeny nezávislou volbou nejdelšího obtížnostního řetězu.

Větvení blockchainu

Protože blockchain je decentralizovaná datová struktura, různé kopie nejsou vždy stejné. Bloky mohou přijít různým uzlům v různé časy, což způsobuje různý pohled uzlů na blockchain. K vyřešení tohoto, každý uzel se snaží vybírat a pokouší se rozšiřovat řetěz bloků, který reprezentuje největší důkaz prací, také známý jako nejdelší řetěz nebo řetěz s nejvyšší celkovou obtížností. Sečtením obtížností zaznamenaných v každém bloku řetězu, každý uzel může spočítat celkové množství práce, která byla vynaložena na vytvoření tohoto řetězu. Tak dlouho dokud, všechny uzly budou vybírat řetěz nejdelší součtem obtížností, celosvětová bitcoinová síť nakonec dojde do shodného stavu. Větvení přichází jako dočasné nesrovnalosti mezi verzemi blockchainu, která jsou vyřešeny případnou změnou hlavního řetězu jak jsou další bloky přidány na jednu z větví.

V následujících několika diagramech budeme sledovat vývoj události "rozvětvení" na síti. Diagram je zjednodušenou reprezentací bitcoinu jako celosvětové sítě. Ve skutečnosti, topologie bitcoinové sítě není organizována zeměpisně. Spíše tvoří pavučinovou síť vzájemně spojených uzlů, které mohou být od sebe zeměpisně velmi vzdáleny. Zeměpisná reprezentace topologie je zjednodušením použitím pro účely znázornění rozvětvení. Ve skutečné bitcoinové síti, "vzdálenost" mezi uzly je měřena "skoky" z uzlu na uzel, ne jejich fyzickou polohou. Pro lepší přehlednost, různé bloky jsou znázorněny odlišnými barvami, rozšiřujícími se po síti a barvící spojení, které použili.

V prvních diagramu (Zobrazení události větvení: blockchain před větvením) síť má jednotný pohled na blockchain, s modrým blokem na vrcholu jeho hlavního řetězu.

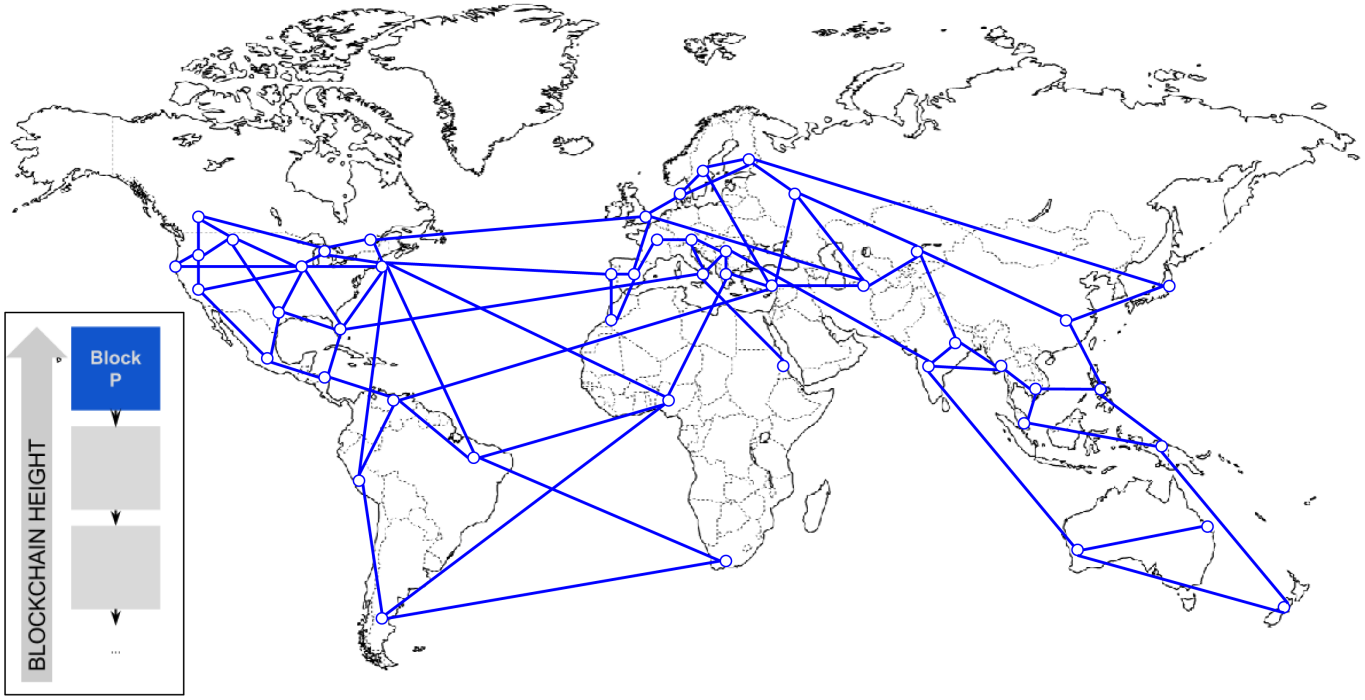


Figure 2. Zobrazení události větvení: blockchain před větvením

"Rozvětvení nastává" když dva kandidátské bloky soupeří ve vytvoření nejdelšího blockchainu. To nastává za normálních okolností, když dva těžaři vyřeší algoritmus důkazu prací v krátkém časovém úseku jeden po druhém. Jakmile těžaři objeví řešení jejich příslušných kandidátských bloků, okamžitě vysílají jejich vlastní "vítězný" blok svým přímým sousedům. kteří začnou šířit tento blok po celé síti. Každý blok, který obdrží platný blok jej zařadí do blockchainu, rozšíří blockchain o jeden blok. Pokud se později objeví jiný kandidátský blok rozšiřující stejného rodiče, spojí druhého kandidáta do vedlejšího řetězu. Ve výsledku, některé uzly "uvidí" jeden kandidátský blok první, zatímco jiné uzly uvidí jiný kandidátský blok a dvě soupeřící verze blockchainu se objeví.

V Zobrazení události větvení: dva bloky nalezeny současně, vidíme dva těžaře, kteří vytěžili dva různé bloky téměř ve stejnou chvíli. Oba tyto bloky jsou dítětem modrého bloku, což znamená, že rozšiřují řetěz stavbou na vrcholu modrého bloku. Pro usnadnění stopování jsme jeden blok zobrazili červenou barvou (pocházející z Kanady) a druhý blok je označen zeleně (pocházející z Austrálie).

Předpokládejme, například, že těžař z Kanady najde řešení důkazu prací pro "červený" blok, který rozšiřuje blockchainu, staví na vrcholu svého rodičovského "modrého" bloku. Téměř ve stejný

okamžik, australský těžař, který také rozšiřuje "modrý" blok našel řešení pro "zelený" blok, jeho kandidátský blok. Nyní jsou dva možné bloky, jeden nazýváme "červený", pocházející z Kanady a druhý nazýváme "zelený" pocházející z Austrálie. Oba bloky jsou platné, oba obsahují platné řešení důkazu prací a oba bloky rozšiřují stejného rodiče. Oba bloky pravděpodobně obsahují převážně ty samé transakce, možná jen pár rozdílu v pořadí transakcí.

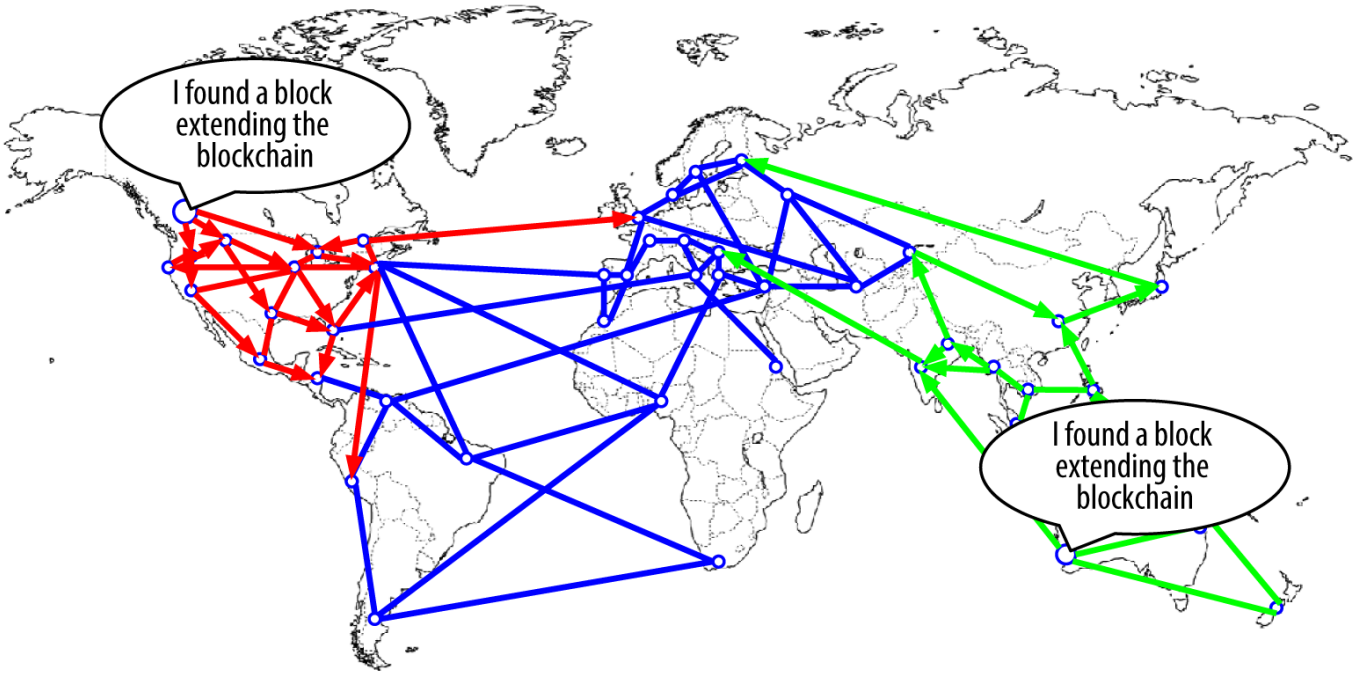


Figure 3. Zobrazení události větvení: dva bloky nalezeny současně

Jak se tyto dva bloky šíří, některé uzly přijmou "červený" blok jako první a jiné přijmou "zelený" blok jako první. Jak ukazuje Zobrazení události větvení blockchainu: šíření dvou bloků, rozdělování sítě síť je rozdělena na dva různé pohledy na blockchain, jeden je na vrcholu s "červeným" blokem, druhý je na vrcholu se "zeleným" blokem.

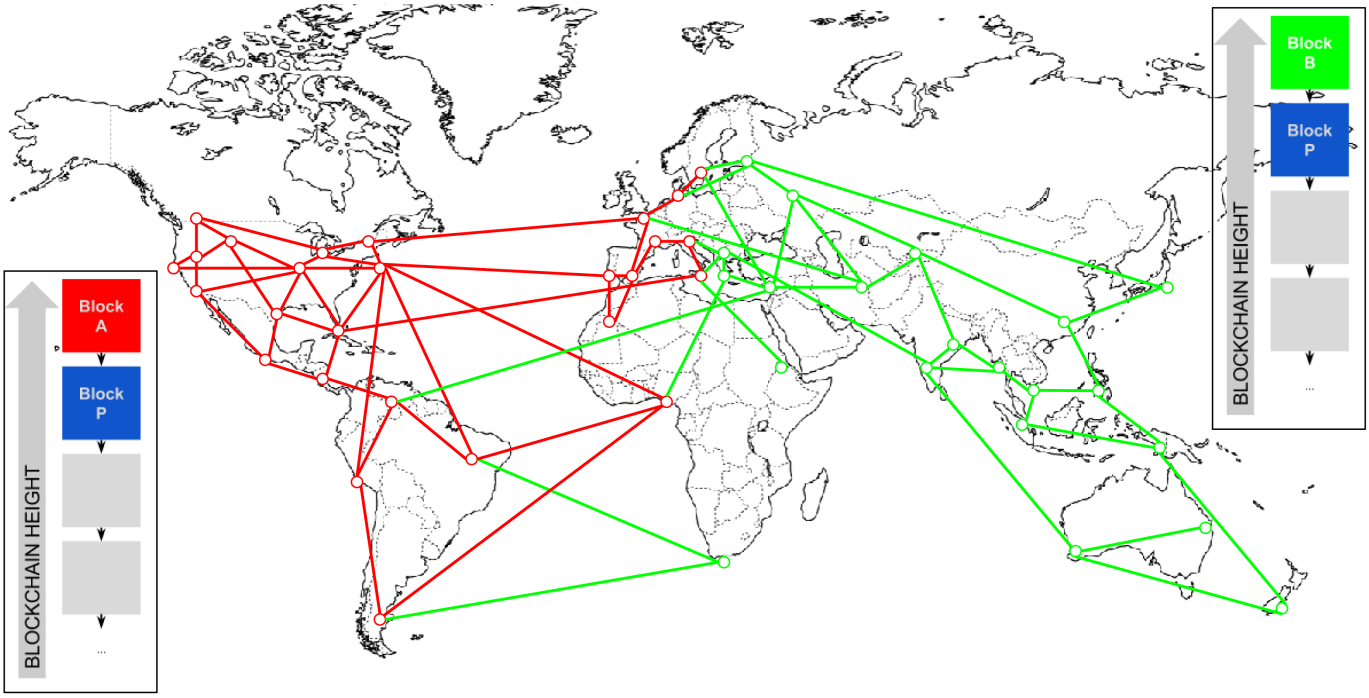


Figure 4. Zobrazení události větvení blockchainu: šíření dvou bloků, rozdělování sítě

Od této chvíle, uzly bitcoinové sítě nejbližší (topologicky, ne zeměpisně) uslyší o "červeném" bloku jako prvním a vytvoří nový blockchain s nejdelším součtem obtížností s "červeným" blokem jako posledním blokem řetězu (např. modro-červený), ignorují "zelený" kandidátský blok, který přišel o trochu později. Mezitím uzly bližší australskému uzlu vezmou "zelený" blok jako vítězný blok a rozšíří s ním blockchain o poslední blok (např. modro-zelený), ignorují "červený" uzel, který dorazil o pár sekund později. Jakýkoliv těžař, který viděl první červený blok bude okamžitě stavět svůj kandidátský blok, který odkazuje na "červený" jako na svého rodiče a začne řešit důkaz prací pro tento kandidátský blok. Těžař, který přijal místo něj "zelený" blok, začne stavět na vrcholu "zeleného bloku" a rozšiřovat tento řetěz.

Větvení je téměř vždy vyřešené do jednoho bloku. Část síťové hašovací síly se zaměřuje na stavbu na vrcholu "červeného" rodiče, druhý část hašovací síly je zaměřena na stavbu na "zeleném" rodičovi. Dokonce i když je hašovací síla téměř rovnoměrně rozdělena, je pravděpodobné, že jedna skupina těžařů najde řešení a rozšíří ho předtím než druhá skupina těžařů najde nějaké řešení. Řekněme, například, že těžaři stavějící na vrcholu "zeleného" bloku najdou nový "růžový" blok, který rozšiřuje řetěz (např. modro-zeleno-růžový). Okamžitě rozšíří tento nový blok celou sítí, která jej vidí jako platné řešení, jak je zobrazeno v Zobrazení události větvení blockchainu: nový blok rozšířil jednu větev.

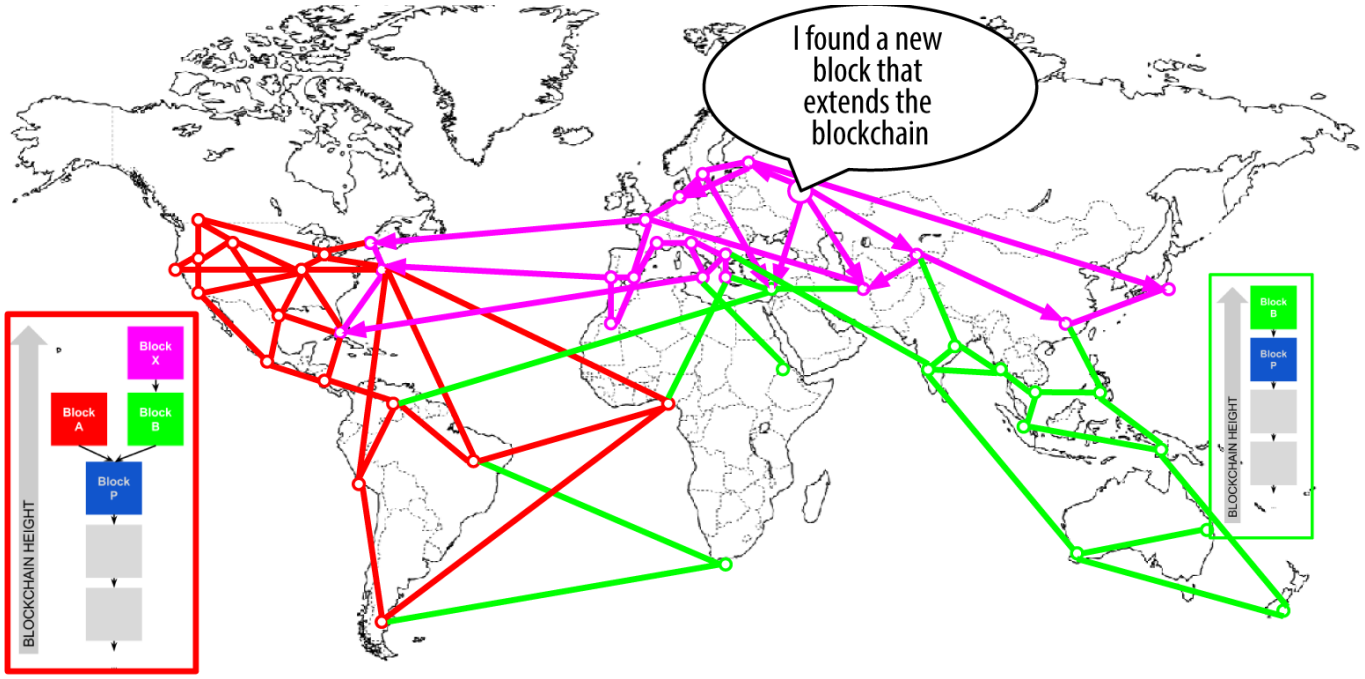


Figure 5. Zobrazení události větvení blockchainu: nový blok rozšířil jednu větev

Všechny uzly, které vybraly "zelený" uzel jako vítěze předchozího kola jednoduše rozšíří řetěz jedním dalším blokem. Uzly, které vybraly "červený" uzel jako vítěze předchozího kola, nicméně nyní vidí dva řetězy; modro-zeleno-růžový a modro-červený. Řetěz modro-zeleno-růžový je nyní delší (větší součet obtížnosti) než řetěz modro-červený. Jako výsledek, tyto uzly nastaví řetěz modro-zeleno-růžový jako hlavní řetěz a změní řetěz modro-červený na vedlejší řetěz, jak vidíme v Zobrazení události větvení blockchainu: síť se nastaví na nový nejdelší řetěz. Toto je změna hlavního řetězu, protože tyto uzly byly donuceny změnit jejich pohled na blockchainu, aby začlenily nové důkazy o delším řetězu. Jakýkoliv těžař pracující na rozšiřování řetězu modro-červeného nyní zastaví prací, protože jejich kandidátský blok je "sirotek", když jeho "červený" rodič není již dále nejdelším řetězem. Transakce z "červeného" uzlu jsou opět zařazeny pro zpracování dalším blokem, protože tento blok není již nadále součástí hlavního bloku. Celá síť se shodne na jednom blockchainu modro-zeleno-růžový s "růžovým" blokem jako posledním blokem řetězu. Všichni těžaři začnou okamžitě pracovat na kandidátském bloku, který odkazuje na "růžový" blok jako na svého rodiče a rozšiřují řetěz modro-zeleno-růžový.

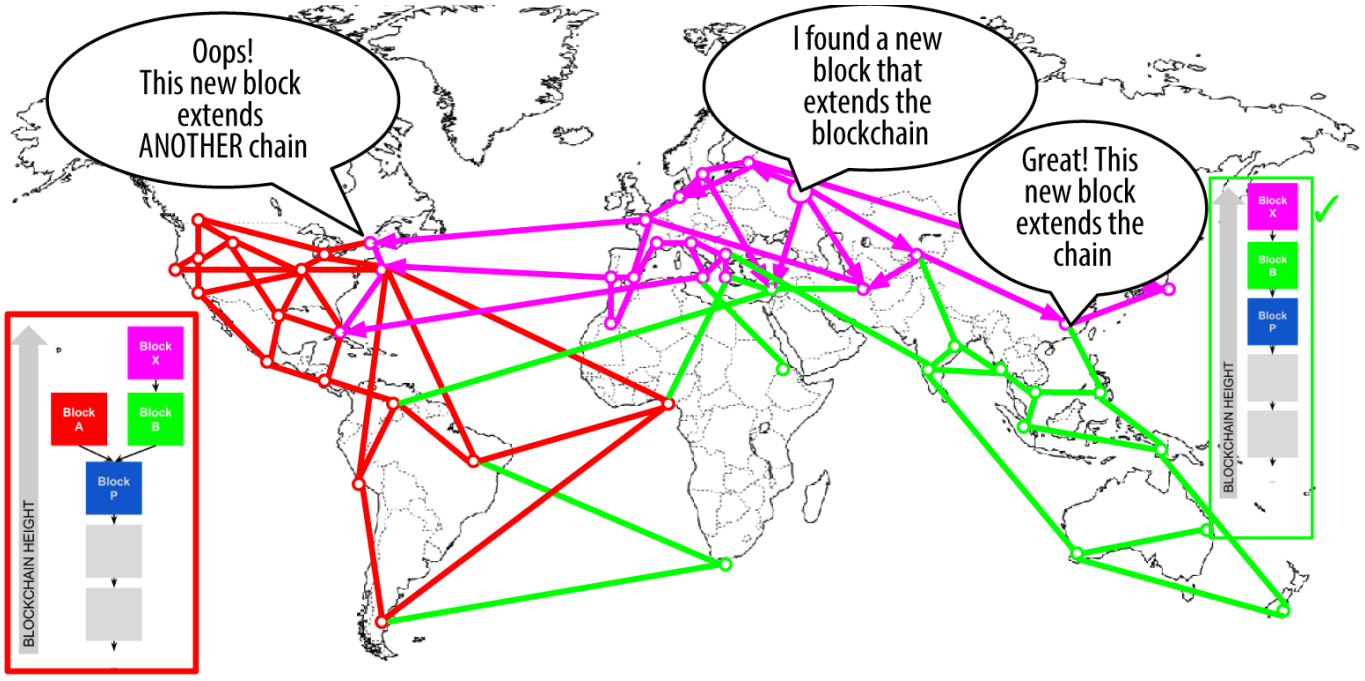


Figure 6. Zobrazení události větvení blockchainu: síť se nastaví na nový nejdelší řetěz

je teoreticky možné, že větvení bude rozšířeno do dvou bloků, pokud dva bloky jsou nalezeny téměř současně těžaři opačných "stran" předchozího větvení. Nicméně, pravděpodobnost, že se tak stane je velmi nízká. Zatímco větvení jednoho bloku nastává každý týden, větvení dvou bloků je mimořádně vzácné.

Bitcoinový 10-minutový interval mezi bloky byl navržen jako kompromis mezi rychlostí potvrzování (vypořádání transakcí) a pravděpodobností větvení. pro kratší intervaly mezi bloky by vypořádání transakcí bylo častější, ale vedlo by častěji k větvení blockchainu, zatímco delší intervaly mezi bloky by snížili počet větvení, ale zpomalily by vypořádání transakcí.

Těžba hašovací závod

Těžba bitcoinu je extrémně konkurenční odvětví. Hašovací síla se zvyšuje exponenciálně každý rok existence bitcoinu. Některé roky tento růst byl odrazem zásadní změny technologie, jako v letech 2010 a 2011, když mnoho těžařů přešlo z používání CPU těžby ("graphical processing units (GPUs)","processing power of"))) na GPU těžbu a těžbu pomocí programovatelných hradlových polí (FPGA). V roce 2013 představení ASIC těžby vedlo k dalšímu obrovskému skoku v těžební síle, umístěním funkce SHA256 přímo na křemíkové čipy specializované pro účely těžby. Jeden takovýto čip může vyvinout více těžební síly v jedné krabičce než celá bitcoinová síť v roce 2010.

Následující seznam zobrazuje celkovou hašovací sílu bitcoinové sítě v prvních pěti letech existence:

2009

0.5 MH/s–8 MH/s (16× nárůst)

2010

8 MH/s–116 GH/s (14 500× nárůst)

2011

16 GH/s–9 TH/s (562× nárůst)

2012

9 TH/s–23 TH/s (2,5× nárůst)

2013

23 TH/s–10 PH/s (450× nárůst)

2014

10 PH/sec–150 PH/s v srpnu (15× nárůst)

V grafu Celková hašovací síla, gigahaše za sekundu, 2012 - 2014 vidíme nárůst hašovací síly bitcoinové sítě v posledních dvou letech. Jak můžeme vidět, soutěživost mezi těžaři a růst bitcoinu má za následek exponenciální nárůst hašovací síly (celkových hašů za sekundu v síti)

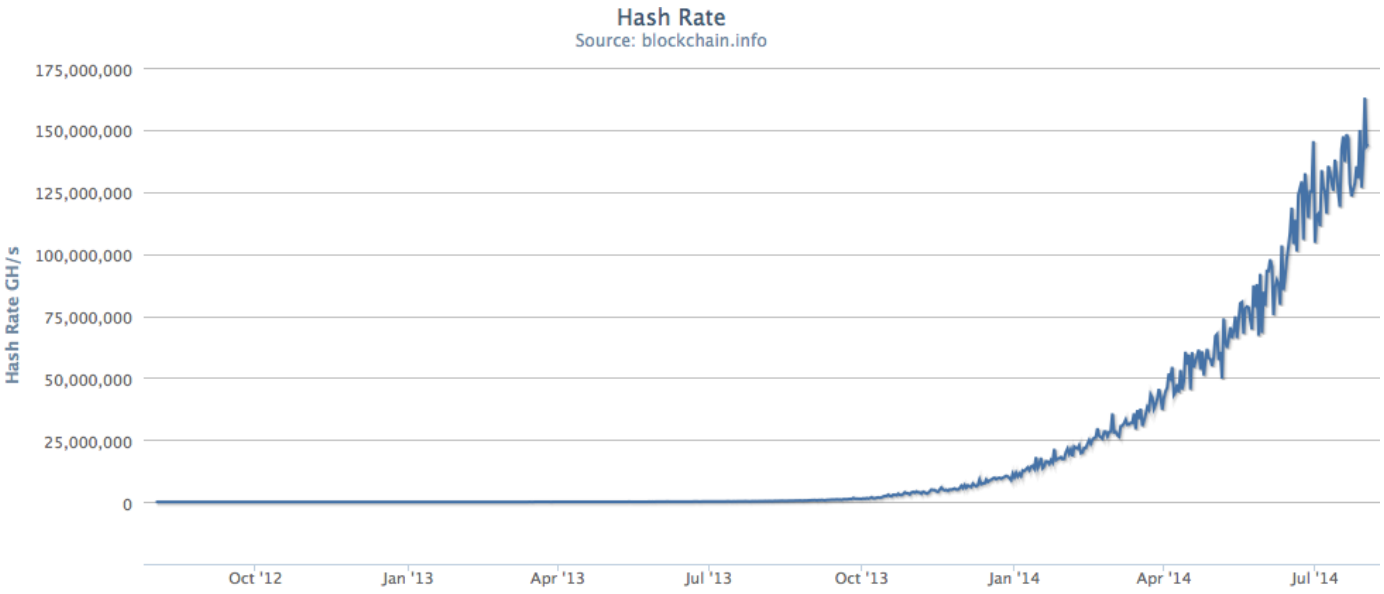


Figure 7. Celková hašovací síla, gigahaše za sekundu, 2012 - 2014

Jak množství hašovací síly použité na těžbu bitcoinu doslova explodovalo, obtížnost těžby se zvýšila, aby došlo k vyrovnání. Obtížnost těžby je zobrazena v Bitcoinová obtížnost těžby, 2012 - 2014 a je měřena poměrem současný obtížnosti ku minimální obtížnosti (obtížnosti pro první blok).

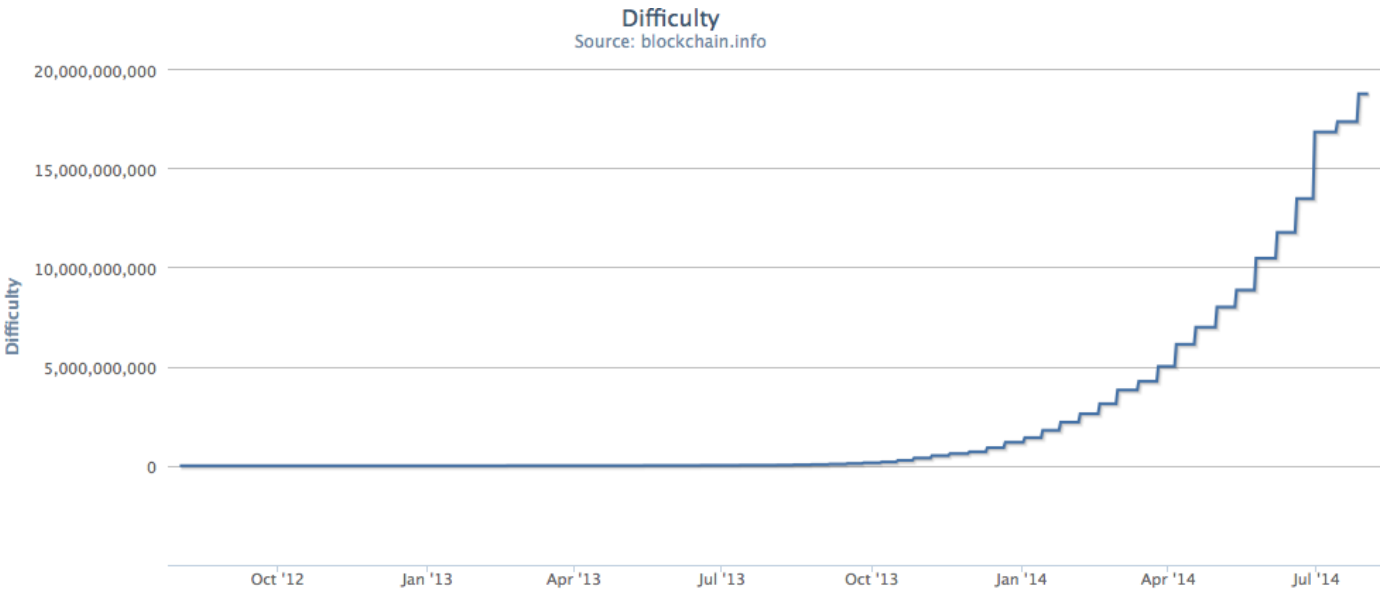


Figure 8. Bitcoinová obtížnost těžby, 2012 - 2014

V letech 2012 až 2014, ASIC těžební čipy se stávaly stále hustšími, blíží technologickým možnostem výroby křemíku s rozlišením 22 nanometrů (nm). Aktuálně, výrobci ASIC se zaměřují na předběhnutí univerzálních výrobců čipů návrhem velikosti 16 nm, protože výnosnost těžby pohání toto odvětví dokonce rychleji něž obecné výpočty. Nejsou žádné další obří skoky v těžbě bitcoinu, protože toto odvětví dosáhlo krajní meze "Moore's Law") Moorova zákona, který stanoví, že hustota čipů se bude zdvojnásobovat přibližně každých 18 měsíců. Stále, těžební síla sítě pokračuje v růstu exponenciálním tempem. Závod o hustotu čipů byl nahrazen závodem o "data centers, mining with") vyšší hustotu výpočetních center, ve kterých mohou být umístěny tisíce těchto čipů. Již to není o tom, kolik těžby vykoná jeden čip, ale kolik čipů lze umístit do budovy, zatímco je dostatečně odváděno teplo a poskytována odpovídající elektrická energie.

Řešení druhé nonce

Od roku 2012 těžba bitcoinu vyvinula řešení zásadního omezení struktury hlavičky bloku. V počátcích bitcoinu, těžař mohl najít blok iterací nonce, dokud výsledný haš nebyl pod cílem. Jak obtížnost rostla, těžaři často prošly všechny 4 miliardy hodnot nonce bez najití bloku. Nicméně bylo snadné aktualizovat časovou značku, zohlednit uplynulý čas. Protože časová značka je součástí hlavičky, tato změno dovolila těžařům znova vyzkoušet všechny hodnoty nonce s rozdílnými výsledky. Jakmile hardware přesáhl 4GH/s, nicméně, tento přístup se stával stále složitějším, protože všechny hodnoty nonce byly vyčerpány za méně než sekundu. Jak se objevila ASIC těžba a toto vybavení přesáhlo TH/s, těžební software potřeboval více místa pro hodnoty nonce, aby našel platný blok. Časová značka může být trochu natažena, ale její posun příliš do budoucnosti může učinit blok neplatným. Nový zdroj "změn" byl potřeba v hlavičce bloku. Řešením bylo použít mincetvornou transakci jako zdroj druhé nonce. Protože mincetvorný skript může uchovat mezi 2 a 100 byty dat, těžaři začali používat prostor jako druhou nonci, což jím umožnilo prozkoumat mnohem větší rozsah hlaviček bloků pro najití platného bloku. Mincetvorná transakce je zahrnuta do merkle stromu, což znamená, že jakákoliv změna v mincetvorného skriptu způsobí změnu kořenu merkle stromu. Osm bytu druhé nonce plus 4 byty "standardní" nonce umožňují těžařům prozkoumat 2^{96} (8 následováno 28 nulami) možností *za sekundu* bez nutnosti měnit časovou značku. V budoucnu mohou těžaři používat všechny tři možnosti,

mohou poté měnit časovou značku. V mincetvorném skriptu je také další místo pro další rozšíření velikosti druhé nonce.

Těžební skupiny

"hashing race","mining pools", id="ix_ch08-asciidoc26", range="startofrange") V tomto vysoce soutěživém prostředí, jednotliví těžaři pracující samostatně (známí také jako samostatní těžaři) nemají šanci. Pravděpodobnost, že najdou blok, který zaplatí jejich výdaje ze elektrickou energii a hardware je tak nízká, že představuje hazard, jako hraní loterie. Dokonce nejrychlejší uživatelský ASIC těžební systém nemůže držet krok s komerčními systémy skládající se z desítek tisící těchto čipů ve velkých skladištích u hydroelektráren. Těžaři nyní spolupracují a vytváří těžební skupiny, skládající jejich hašovací sílu a rozdělující odměnu mezi tisíce účastníků. Účastí ve skupině, těžař dostane menší podíl celkové odměny, ale obvykle dostane odměnu každý den, což snižuje nejistotu.

Podívejme se na konkrétní příklad. Předpokládejme, že těžař si koupil těžební hardware s celkovým hašovací silou 6 000 gigahašu za sekundu (GH/s) nebo 6 TH/s. V srpnu 2014 toto vybavení stálo 10 000 amerických dolarů. Hardware potřebuje 3 kilowatty (kW) elektrické energie když běží, 72 kWh za den, při ceně 7 až 8 dolarů za den. Při současné obtížnosti těžby, těžař bude schopný vytěžit samostatně blok přibližně jednou za 155 dní, nebo každých 5 měsíců. Pokud těžař najde jeden blok v tomto časovém okně, výplata je 25 bitcoinů při 600 dolarech za bitcoin dosáhne jedné výplaty 15 000 dolarů, která pokryje celou cenu hardware a spotřebované elektřiny za toto časové období, a zůstane čistý zisk přibližně 3000 dolarů. Nicméně, šance na najití bloku v tomto pětiměsíčním období závisí na těžařově štěstí. Může vytěžit dva bloky za pět měsíců a mít velmi vysoký zisk. Nebo může neúspěšně hledat blok 10 měsíců a utrpět finanční ztrátu. Dokonce ještě hůře, obtížnost bitcoinového algoritmu důkazu prací může výrazně vzrůst během tohoto období. Při současné rychlosti růstu hašovací síly má těžař nejvýše šest měsíců na průlom, než se je hardware fakticky zastaralý a musí být vyměněn účinnějším těžebním hardware. Pokud těžař se účastní těžební skupiny, místo čekání na jeden pětiměsíční mimořádný příjem 15 000 dolarů, bude vydělávat přibližně 500 až 750 dolarů týdně. Pravidelné platby z těžební skupiny mu pomohou s umořením ceny hardware a elektrické energie bez nutnosti podstupovat obrovské riziko. Hardware stále zastará za šest až devět měsíců a riziko je stále vysoké, ale odměna je alespoň pravidelná a spolehlivá po toto časové období.

Těžební skupiny koordinují mnoho stovek nebo tisíc těžařů pomocí specializovaných protokolů těžebních skupin. Samostatní těžaři nastavují své těžební vybavení, pro připojení do těžební skupiny, po vytvoření účtu v této těžební skupině. Jejich těžební hardware zůstává připojen ke skupinovému serveru během těžby, synchronizujícím jejich úsilí s ostatními těžaři. Proto, těžaři ve skupině sdílejí svoje úsilí pro vytěžení bloku a poté sdílejí své odměny.

Úspěšné bloky platí odměnu bitcoinové adrese skupiny, místo individuálnímu těžaři. Server skupiny provádí periodické platby na bitcoinové adresy těžařů, jakmile jejich podíl na odměně dosáhl jisté prahové hodnoty. Obvykle, server skupiny sráží procentní poplatek jako odměnu za poskytování služby těžení ve skupině.

Těžaři účastnící se skupiny si rozdělují práci hledání řešení kandidátského bloku, získávají "podíly" za jejich těžební přínos. Těžební skupina nastaví nižší obtížnostní cíl pro získání podílu, obvykle 1000 krát snadnější než je obtížnost bitcoinové sítě. Když někdo ve skupině úspěšně vytěží blok, odměna je

získána skupinou a poté je rozdělena všem těžařům proporčně podle počtu podílů, kterými přispěli k tomuto úsilí.

Těžební skupiny jsou otevřeny všem těžařům, velkým nebo malým, profesionálům nebo amatérům. Skupina má tedy některé účastníky s jedním malým těžebním zařízením, jiné s garáží plnou nejkvalitnějšího těžebního hardware. Někteří budou těžit s několika desítkami kilowatt elektrické energie, jiní provozují datová centra spotřebovávající megawatty výkonu. Jak těžební skupina měří jednotlivé přínosy, aby spravedlivě rozdělovala odměny, bez možnosti podvádění? Odpověď je v bitcoinovém algoritmu důkazu prací, který měří přínos každého těžaře, ale je nastaven na nízkou obtížnost, takže i nejmenší těžař ve skupině vyhraje podíl dostatečně často, aby mu stálo za to, přispívat do skupiny. Nastavením nízké obtížnosti pro zisk podílu, skupina měří množství práce vykonané každým těžařem. Pokaždé když těžař skupiny najde haš hlavičky bloku, který je menší než obtížnost skupiny, prokázal hašovací práci na najití tohoto výsledku. Co je důležitější, práce na najití podílu přispívá, statisticky měřitelným způsobem, k celkové snaze na najití haše nižšího než je cíl bitcoinové sítě. Tisíce těžařů se snažících se najít haše nízké hodnoty nakonec najdou jeden dostatečně nízký, že splní cíl bitcoinové sítě.

Vraťme se k podobenství o hře s kostkami. Pokud hráči kostek házejí kostkou s cílem hodit méně než čtyři (celková obtížnost sítě), skupina nastaví snazší cíl, počítající kolikrát hráči skupiny zvládli hodit méně než osm. Když hráči skupiny hodí méně než osm (skupinový cíl podílu), získají podíl, ale nevyhrají hru, protože nedosáhli herního cíle (méně než čtyři). Hráči skupiny budou dosahovat snadnějšího cíle skupiny častěji, budou získávat své podíly velmi pravidelně, dokonce i když nedosáhnou tvrdšího cíle vítězství hry. Občas jeden z hráčů skupiny hodí kombinaci kostek menší než čtyři a skupina vyhraje. Poté, odměny mohou být rozděleny mezi hráče skupiny v závislosti na počtu podílů, které získali. Přestože cíl osm nebo méně nebyl vítězný, byl dostatečně spravedlivou cestou k měření počtu hodů kostkami pro jednotlivé hráče, a náhodně vytvořil hod menší než čtyři.

Podobně, v těžební skupině nastavíme obtížnost skupiny, která zajistí, že jednotliví těžaři skupiny mohou najít haše hlaviček bloku, které jsou nižší než obtížnost skupiny docela často, získají podíly. Občas, jeden z těchto pokusů vyrobí haš hlavičky bloku, který je nižší než cíl bitcoinové sítě, vytvoří platný blok a celá skupina vyhrává.

Spravované těžební skupiny

Většina těžebních skupin je "řízena", což znamená, že společnost nebo jedinec provozuje skupinový server. Vlastník skupinového serveru je nazýván *provozovatel skupiny*, a účtuje těžařům procentní poplatek z příjmů.

Server těžební skupiny provozuje specializovaný software a protokol těžební skupiny, který koordinuje aktivity těžařů skupiny. Server skupiny je také spojen s jedním nebo více úplnými bitcoinovými uzly a má přímý přístup k úplné kopie blockchainové databáze. To umožňuje serveru skupiny ověřovat bloky a transakce v zastoupení za těžaře skupiny, zbavuje je břemene provozování úplného uzlu. Pro těžaře skupiny, je to důležitá úvaha, protože úplný uzel potřebuje vyhrazený počítač s 15 až 20 GB trvalého úložiště (disk) a nejméně 2GB operační paměti (RAM). Navíc, bitcoinový software běžící na úplném uzlu potřebuje být často sledován, udržován a aktualizován. Jakýkoliv výpadek způsobený nedostatkem údržby nebo zdrojů ohrozí výdělečnost těžaře. Pro mnoho těžařů schopnost těžit bez

provozování úplného uzlu je další velkou výhodou vstupu do těžební skupiny.

Těžaři ve skupině se připojují k těžebnímu serveru za použití těžebního protokolu jako "Stratum (STM) mining protocol") Stratum (STM) nebo GetBlockTemplate (GBT). Starší standard zvaný GetWork (GWK) se stal zastaralým od roku 2012, protože snadno nepodporoval těžbu nad rychlost 4 GH/s. Oba SMT a GBT protokoly vytvářejí "block templates") *šablonu* bloku, která obsahuje šablonu hlavičky kandidátského bloku. Server skupiny vytváří kandidátský blok seskupováním transakcí, přidáním mincetvorné transakce (s místem pro druhou nonci), spočítá kořen merkle stromu a připojí haš předchozího bloku. Hlavička kandidátského bloku je zaslána každému těžaři skupiny jako šablona. Každý těžař poté těží za použití šablony bloky na nižší obtížnosti než je obtížnost bitcoinové sítě a zasílá každý úspěšný výsledek zpátky serveru skupiny, aby získal podíly.

P2P těžební skupina (P2Pool)

Spravované těžební skupiny vytvářejí možnost podvodů ze strany provozovatele skupiny, který může nasměrovat úsilí skupiny na dvojité utracení transakcí nebo zneplatnění bloků (viz Útoky na shodu). Kromě toho, centralizované servery skupiny představují kritické místo náchylné k selhání. Pokud server skupiny je nefunkční nebo zpomalený útokem odepření služby, těžaři skupiny nemohou těžit. V roce 2011 byl tento problém centralizace vyřešen. Nová skupinová těžební metoda byla navržena a implementována: P2P těžební skupina (P2Pool) bez centrálního provozovatele.

P2Pool pracuje tak, že decentralizuje funkce serveru skupiny, implementuje paralelní blockchainu podobný systém zvaný *sdílený řetěz*. Sdílený řetěz je blockchainu běžící na nižší obtížnosti než bitcoinový blockchain. Tento sdílený řetěz umožňuje těžařům skupiny spolupracovat v decentralizované skupině, těžbou podílů na sdíleném řetězu rychlostí jeden blok každých 30 sekund. Každý z bloků na sdíleném řetězu zaznamená proporcionální sdílenou odměnu pro těžaře skupiny, kdo přispěl prací, přenáší podíly dopředu od posledního sdíleného bloku. Když jeden ze sdílených bloků také dosáhne obtížnostního cíle bitcoinové sítě, je rozšířen a vložen do bitcoinového blockchainu, odměňuje všechny těžaře skupiny, kteří přispěli do všech podílů, které předcházeli podílu vítězného bloku. V podstatě, místo serveru skupiny udržujícím záznamy o podílech těžařů a odměnách, tento sdílený řetěz umožňuje všem těžařům skupiny udržovat záznam o všech podílech za použití decentralizovaného mechanizmu shody jako je mechanizmus shody bitcoinového blockchainu.

P2Pool těžba je složitější než sdílená těžba, protože vyžaduje, aby každý člen skupiny provozoval oddělený počítač s dostatkem diskového prostoru, pamětí a internetovým připojením pro podporu úplného bitcoinového uzlu a software P2Pool uzlu. P2Pool těžaři připojují jejich těžební hardware do jejich místního P2Pool uzlu, který simuluje funkce serveru skupiny zasíláním šablon bloků těžebnímu hardware. V P2Pool jednotlivé těžaři skupiny sestavují své vlastní kandidátské bloky, shromažďují transakce jako nezávislí těžaři, ale těží společně na sdíleném řetězu. P2Pool je hybridní přístup, který má výhodu větší granularity plateb než nezávislá těžba, ale nedává kontrolu provozovali skupiny jako řízené skupiny.

Nedávno, účast na P2Pool se zvýšila významně, když těžba koncentrovaná v těžebních skupinách dosáhla úrovně, která vytváří obavy z "51% attacks") 51% útoku (viz Útoky na shodu). Další vývoj protokolu P2Pool pokračuje s očekáváním odstranění potřeby provozování úplného uzlu a tedy ještě většího usnadnění decentralizované těžby.

Přestože P2Pool snižuje koncentraci síly operátorem těžební skupiny, je teoreticky zranitelný na 51% útok proti svému sdílenému řetězu. Mnohem širší přijetí P2Pool neřeší problém 51% útoku na bitcoin samotný. Spíše, P2Pool dělá bitcoin celkově odolnějším, jako část diverzifikovaného těžebního ekosystému.

Útoky na shodu

Mechanizmus bitcoinové shody je, alespoň teoreticky, zranitelný útoky těžařů (nebo skupin), které se pokusí použít jejich hašovací sílu k nečestným nebo ničícím koncům. Jak vidíme, mechanizmus shody závisí na tom, aby většina těžařů se chovala čestně bez vlastních zájmů. Nicméně, pokud těžař nebo skupina těžařů je schopná dosáhnout značného podílu těžebního síly, mohou zaútočit na mechanizmus shody, tak aby narušili bezpečnost a dostupnost bitcoinové sítě.

Je důležité poznamenat, že útok na shodu může pouze postihnout budoucí shodu, nebo nejhůře nejnovější minulost (desítky bloků). Bitcoinový účetní systém se stává více a více neměnným jak ubíhá čas. Zatímco teoreticky rozvětvení může nastat v jakékoliv hloubce, v praxi, výpočetní síla nutná k vynucení velmi hlubokých rozvětvení je obrovská, takže staré bloky jsou prakticky neměnné. Útoky na shodu také nepostihují bezpečnost soukromých klíčů a podepisovací algoritmus (ECDSA). Útoky na shodu nemohou krást bitcoiny, utrácet bitcoiny bez podpisů, přesměrovávat bitcoiny, nebo jinak měnit minulé transakce nebo záznamy vlastnictví. Útoky na shodu mohou pouze postihnout nejnovější bloky a způsobit odepření služby narušením vytvářením budoucích bloků.

Jeden scénář útoku proti mechanizmu shody je zván "51% útok." V tomto scénáři, skupina těžařů kontrolující většinu (51 %) celkové hašovací síly sítě, se tajně domluví na útoku na bitcoin. Se schopností vytěžit většinu bloků, útočící těžaři mohou způsobit záměrné "rozvětvení" blockchainu a dvojitě utrácet transakce nebo provádět útok odepření služby proti určitým transakcím nebo adresám. Útok rozvětvením/dvojitým utrácením je ten, kdy útočník způsobí neplatnost předchozích potvrzených bloků pomocí rozvětvení pod nimi a přesměrováním na alternativní řetěz. S dostatečnou silou, útočník může zneplatnit šest nebo více bloků v řadě, způsobit, že transakce považované za neměnné (šest potvrzení) jsou zneplatněny. Poznámka, dvojité utracené může být vykonáno pouze s vlastními transakcemi útočníka, pro které útočník může vytvořit platný podpis. Dvojité utracení vlastní transakce je ziskové, pokud zneplatněním transakce útočník může získat nevratitelnou výměnnou platbu nebo výrobek aniž by zaplatil.

Prozkoumejme praktický příklad 51% útoku. V prvních kapitole jsem se dívali na transakci mezi Alicí a Bobem za šálek kávy. Bob, majitel kavárny je ochotný přijmout platbu za šálek kávy bez čekání na potvrzení (vytěžení bloku), protože riziko dvojitého utracení na šálku kávy je nízké v porovnání s pohodlností rychlé zákaznické služby. Toto je podobné praxi kavárny, která přijímá platby kreditní kartou bez podpisu pro platby pod 25 dolarů, protože riziko zpětné vrácení platby zaplacené kreditní kartou je nízké, zatímco cena zdržení transakce pro získání podpisu je srovnatelně vyšší. Naopak, prodávání dražších věcí za bitcoiny přináší riziko útoku dvojitého utracení, když kupující rozešle soupeřící transakci, která utrácí ten samý vstup) (UTXO) a zruší tak platbu obchodníkovi. Útok dvojitým utracením může nastat dvěma způsoby: buďto před potvrzením transakce, nebo pokud útočník využije výhody rozvětvení blockchainu pro vrácení několika bloků. 51% útok umožňuje dvojitě utrácet jeho vlastní transakce v novém řetězu, tedy vrácení odpovídajících transakcí ze starého

řetězu.

V našem příkladě, zlomyslný útočník Mallory přijde do galerie Carol a koupí si překrásnou trojici obrazů zobrazující Satoshi Nakamota jako Prométha. Carol prodá trojici obrazů "Velký oheň" za 250 000 dolarů v bitcoinech Mallorymu. Místo čekání na šest nebo více potvrzení transakce, Carol zabalí a předá obrazy Mallorymu pouze po jednom potvrzení. Mallory spolupracuje s komplicem, Paulem, který provozuje velkou těžební skupinu. Komplic spustí 51% útok, jakmile Malloryho transakce je vložena do bloku. Paul nařídí těžební skupině znovu vytěžit blok stejné výšky jako je blok obsahující Malloryho transakci, nahrazující Malloryho platbu Carol s transakcí, která dvojitě utratí stejný vstup Malloryho platby. Dvojitě utracená transakce spotřebuje ten samý UTXO a zaplatí zpátky Malloryho peněžence, místo zaplacení Carol, jednoduše dovoluje Mallorymu si ponechat bitcoiny. Paul poté nařídí těžební skupině těžit další dodatečně bloky, aby se řetěz obsahující dvojitě utracenou transakci stal delším než původní řetěz (způsobí rozvětvení pod blokem obsahujícím Malloryho transakci). Když je rozvětvení blockchainu vyřešeno ve prospěch nového (delšího) řetězu, dvojitě utracená transakce nahrazuje původní platbu Carol. Carol nyní chybí tři obrazy a také nemá bitcoinovou platbu. Po celou dobu této aktivity, členové Paulovy těžební skupiny můžou zůstat v blažené nevědomosti o pokusu dvojitého utracení, protože těží s automatizovanými těžaři a nemohou sledovat každou transakci nebo blok.

Pro ochranu před tímto druhem útoku, obchodníci prodávající věci vysoké hodnoty musejí čekat alespoň šest potvrzení před předáním produktu kupci. Alternativně, obchodník může použít úschovu vícepodpisový účet, znovu počkat několik potvrzení poté, co je účet úschovy naplněn. Čím více potvrzení uběhne, tím těžší je zneplatnit transakci při 51% útoku. Pro vysoce hodnotné věci, platba bitcoiny zůstává stále pohodlnější a účinnější dokonce i když kupující čeká 24 hodin na doručením což odpovídá přibližně 144 potvrzením.

Navíc k útoku dvojitému utracení, další scénář pro útok na shodu je odepření služby určitému bitcoinovému účastníkovi (určité bitcoinové adrese). Útočník s většinou výpočetní síly může jednoduše ignorovat určité transakce. Pokud jsou zahrnuty do vytěženého bloku jiným těžařem, útočník může záměrně rozvětvit a znovu vytěžit tento blok, znova bez vložení konkrétních transakcí. Tento typ útoku může vést k trvalému odepření služby vůči konkrétní adrese nebo množině adres, tak dlouho dokud útočník kontroluje většinu těžební síly.

Navzdory svému jménu, scénář 51 % útoku ve skutečnost nevyžaduje 51 % hašovací síly. Ve skutečnosti, pokus o takový útok může být proveden s menším procentem hašovací síly. Práh 51% je jednoduše hranice, při které má útok téměř jistě zaručen úspěch. Útok na shodu je zjednodušeně přetahování se lanem o další blok a "silnější" skupina má větší šanci vyhrát. S menší hašovací silou, pravděpodobnost úspěchu je snížena, protože ostatní těžaři řídí tvorbu nějakých bloků s jejich "čestnou" těžební silou. Jeden úhel pohledu na toto je, že čím více hašovací síly útočník má, tím delší rozvětvení může úmyslně vytvořit, více bloků z nedávné minulosti může zneplatnit, nebo více bloků v budoucnu může ovládat. Bezpečnostní výzkumná skupina použila statistické modelování a tvrdí, že různé typy útoků na shodu jsou možné už od 30 % hašovací síly.

Masivní nárůst celkové hašovací síly dělá bitcoin pravděpodobně odolný vůči útokům jednotlivých těžařů. Neexistuje způsob pro samostatného těžaře, aby kontroloval více než několik málo procent celkové těžební síly. Nicméně, centralizované kontrola způsobená těžebními skupinami přinesla riziko

útoků za účelem zisku ze strany provozovatelů těžebních skupin. provozovatel skupiny v řízených těžebních skupinách kontroluje sestavení kandidátských bloků a také kontroluje transakce, které jsou vkládány. To dává provozovateli skupiny moc odstranit transakce nebo zavést dvojitě utracené transakce. Pokud je takovéto zneužití síly děláno omezeným a decentním způsobem, provozovatel skupiny může teoreticky vydělávat pomocí útoku na shodu, aniž by si toho někdo všiml.

Ne všichni útočníci jsou motivování ziskem, nicméně. Jeden možný scénář útoku je když, útočník se snaží narušit bitcoinovou síť bez možnosti vydělat na tomto narušení. Nebezpečný útok zaměřený na ochromení bitcoinu by vyžadoval obrovské investice a skryté plánování, ale mohl by být případně zahájen dobře financovaným útočníkem (s největší pravděpodobností státem sponzorovaným). Alternativně, dobře financovaný útočník může napadnout bitcoinovou shodu současným hromaděním těžebního hardware a ohrožením provozovatelů těžebních skupin a útokem odepření služby na ostatní těžební skupiny. Všechny tyto scénáře jsou teoreticky možné, ale stále více nepraktické, jak celková hašovací síla bitcoinové sítě pokračuje v exponenciálním růstu.

Nepochybně, vážný útok na shodu může krátkodobě narušit důvěru v bitcoin, možná způsobit značný pokles ceny. Nicméně, bitcoinová síť a software se stále vyvíjejí, takže útok na shodu by se setkal s okamžitým protiopatřením ze strany bitcoinové komunity, což udělá bitcoin odolnější, utajenější a silnější než kdykoliv předtím.

Alternativní řetězy, měny, a aplikace

Bitcoin byl výsledkem 20-letého výzkumu distribuovaných systému a měn a přinesl novou revoluční technologii do vesmíru: mechanizmus decentralizované shody založený na důkazu prací. Tento objev v srdci bitcoinu předznamenal vlnu inovací v měnách, finančních službách, ekonomice, distribuovaných systémech, volebních systémech, správě a řízení a kontraktech.

V této kapitole prozkoumáme mnoho odnoží bitcoinových a blockchainových vynálezů: alternativní řetězy, měny a aplikace vytvořené od představení této technologie v roce 2009. Nejvíce se podíváme na alternativní měny nebo *altcoiny*, které jsou digitálními měnami implementovanými za použití stejného návrhového vzoru jako bitcoin, ale s úplně odděleným blockchainem a sítí.

Pro každý altcoin zmíněny v této kapitole, bude 50 nebo více jiných nezmíněno, což vyvolá vlny hněvu od jejich tvůrců a fanoušků. Cílem této kapitoly není vyhodnocovat nebo kvalifikovat altcoiny, nebo dokonce zmiňovat nejvýznamnější z nich na základě nějakých subjektivních posouzení. Místo toho, zdůrazníme pár příkladů, které ukazují šířku a rozmanitost ekosystémů, zmíníme první svého druhu pro každou inovaci nebo významnou odlišnost. Některé z nejzajímavějších příkladů altcoinů jsou ve skutečnosti úplné propadáky z peněžní perspektivy. To je pravděpodobně dělá tím více zajímavé pro studium a zviditelňuje skutečnost, že tato kapitola nemá být použita jako investiční návod.

S novými mincemi představenými každý den, může být nemožné neminout nějakou důležitou minci, možná tu, která změní historii. Míra inovací je to, co dělá tento prostor tak zajímavým a zaručuje této kapitole, že bude nekompletní a zastaralá jakmile bude vydána.

Taxonomie alternativních měn a řetězů

Bitcoin je projekt s otevřeným zdrojovým kódem a jeho k=od byl použit jako základ mnoha jiných softwarových projektů. Nejčastější druh software zrozeného z bitcoinových zdrojových kódu jsou alternativní decentralizované měny, nebo *altcoiny*, které používají stejné základní stavební bloky pro implementaci digitálních měn.

Existuje množství vrstev protokolu implementovaných na vrcholu bitcoinového blockchainu. Tyto *meta mince*, *meta řetězy* nebo *blockchainové aplikace* používají blockchain jako aplikační platformu nebo rozšíří bitcoinový protokol přidáním protokolové vrstvy. Příklady zahrnují Barvené mince, Mastercoin, NXT a Counterparty.

V další části prozkoumáme několik pozoruhodných altcoinů, jako Dogecoin, Freicoin, Primecoin, Peercoin, Dash, a Zerocoin. Tyto altcoiny jsou pozoruhodné z historických důvodů, protože jsou dobrými příklady konkrétních typů inovací altcoinů, nikoliv protože by měli nejvyšší hodnotu nebo byly "nejlepšími" altcoiny.

Navíc k těmto altcoinům, existuje také mnoho alternativních implementací blockchainu, které nejsou skutečnými "mincemi", které nazývám call *altchain* . Tyto altchainy implementují algoritmus shody a distribuovaný účetní systém na platformě pro kontrakty, registrace názvů nebo jiné aplikace. Altchainy používají stejné stavební bloky a občas také používají měnu nebo žeton jako platební mechanizmus,

ale jejich hlavním důvodem není měna. Podíváme se na Namecoin a Ethereum, jako příklady těchto altchainů.

Nakonec, je zde množství soupeřů bitcoinu, kteří nabízejí digitální měnu nebo digitální platební síť, ale bez použití decentralizovaného účetního systému nebo mechanizmu shody založeném na důkazu prací, jako Ripple a další. Tyto ne-blockchainové technologie jsou mimo zaměření této knihy a nebudou pokryty v této kapitole.

Platformy meta mincí

Meta mince a meta řetězy jsou softwarové vrstvy implementované na vrcholu bitcoinu, buďto implementující měnu uvnitř měny nebo platforma/protokol překryv v bitcoinové systému. Tyto funkční vrstvy rozšiřují základní bitcoinový protokol a přidávají funkce a schopnosti kódovat dodatečná data uvnitř bitcoinových transakcí a bitcoinových adres. První implementace meta mincí používaly různé hacky pro přidání metadat do bitcoinového blockchainu, jako použití bitcoinové adresy pro zakódování dat nebo použití nepoužívaných transakčních polí (např. pole transakční sekvence) pro kódování metadat o přidané protokolové vrstvě. Od představení kódu operátoru transakčního skriptu OP_RETURN, meta mince jsou schopny zaznamenávat data přímočařeji do blockchainu.

Barevné mince

Barevné mince je meta protokol, který připojuje informace k malé množství bicoinu. "Barevná" mince je množství "bicoinu", ke kterému je přidán dodatečný význam, které vyjadřuje jiné aktivum. Představte si, například, že vezmete 1-dolarovou bankovku a označíte ji razítkem a prohlásíte: Toto je 1 podílový list Acme Inc." Nyní 1-dolarová bankovka slouží dvěma cílům: je to bankovka a také podílový list. Protože je hodnotnější jako podílový list, nebudete jí chtít použít k nákupu sladkostí, takže fakticky není už použitelná jako platidlo. Barevné mince pracují stejným způsobem, převádějí velmi malé množství Bitcoinu na obchodovatelná potvrzené vyjadřující jiné aktivum. Pojem "mince" odkazuje k myšlence vyjádření zvláštního významu přidáním vlastnosti jako je barva. Je to metafora, ve skutečnosti není žádné spojení s barvou. Barevné mince nemají žádné barvy.

Barevné mince jsou spravovány specializovanými peněženkami, které zaznamenávají a interpretují metadata připojená k obarveným bitcoinům. Použitém této peněženky, uživatel přemění množství bitcoinů z neobarvené měny na obarvené mince přidáním označení se speciálním významem. Například, označení může vyjadřovat akcie, kupóny, nemovitosti, komodity nebo sběratelské žetony. Je to zcela na uživateli barevných mincí, aby přiřadil a interpretoval význam "barvy" spojené s určitými mincemi. Barva minci, uživatelsky definovaný význam metadata, jako typ emise, zda může být dělena na menší jednotky, symbol a popis a další související informace. Jakmile jsou obarveny, tyto mince mohou být kupovány a prodávány, děleny a seskupovány a přijímat platby dividend. Barevné mince mohou být "odbarveny" odstraněním zvláštního významu a uplatněním svojí nominální hodnoty v bitcoinech.

K představení použití barevných mincí, jsme vytvořili množinu 20 barevných mincí se symbolem "MasterBTC", které reprezentují kupóny na bezplatnou kopii knihy, jak je ukázáno v Metadata

barevných mincích vyjadřujících kupón na bezplatnou kopii knihy.. Každá jednotka MasterBTC reprezentovaná touto barevnou mincí nyní může být dána nebo prodána jakémukoliv uživateli bitcoinů s peněženkou podporující barevné mince, který ji může převádět dalším uživatelům nebo ji u vydavatele vyměnit za bezplatnou kopii knihy. Tento příklad barevných mincí můžeme vidět here.

Example 1. Metadata barevných mincích vyjadřujících kupón na bezplatnou kopii knihy.

```
{
  "source_addresses": [
    "3NpZmvSPLmN2cVFw1pY7gxEAVPCVfnWfVD"
  ],
  "contract_url":
"https://www.coinprism.info/asset/3NpZmvSPLmN2cVFw1pY7gxEAVPCVfnWfVD",
  "name_short": "MasterBTC",
  "name": "Free copy of \"Mastering Bitcoin\"",
  "issuer": "Andreas M. Antonopoulos",
  "description": "This token is redeemable for a free copy of the book \"Mastering
Bitcoin\"",
  "description_mime": "text/x-markdown; charset=UTF-8",
  "type": "Other",
  "divisibility": 0,
  "link_to_website": false,
  "icon_url": null,
  "image_url": null,
  "version": "1.0"
}
```

Mastercoin

Mastercoin je protokolová vrstva nad bitcoinem, která podporuje platformu pro různé aplikace rozšiřující bitcoinový systém. Mastercoin užívá měnu MST jako žeton pro provádění Mastercoin transakcí, ale není to primárně měna. Spíše, je to platforma pro tvorbu jiných věcí, jako jsou uživatelské měny, žetony chytrého vlastnictví, decentralizované burzy aktiv a kontrakty. Přemýšlejme o Mastercoinu jako o aplikační vrstvě protokolu nad transportní vrstvou bitcoinových finančních transakcí, jako HTTP běží nad TCP.

Mastercoin především pracuje pomocí transakcí zaslaných z a na speciální bitcoinovou adresu zvanou "exodus addresses") "výstupní" adresa (1EXoDusjGwvnjZUyKkxZ4UHEf77z6A5S4P), jako HTTP používá konkrétní TCP port (port 80) pro odlišení svého provozu od zbytku provozu TCP. Mastercoin protokol postupně přechází od používání výstupní adresy a vícepodpisovosti na používání bitcoinového operátoru OP_RETURN pro kódování transakčních metadat.

Counterparty

Counterparty je další protokolová vrstva implementována nad bitcoinem. Counterparty umožňuje uživatelské měny, obchodovatelné žetony, finanční nástroje, decentralizovanou výměnu aktiv a další funkce. Counterparty je hlavně implementováno pomocí operátoru OP_RETURN v bitcoinovém skriptovacím jazyku pro zaznamenání metadat a výměnu bitcoinových transakcí s přidaným významem. Counterparty používá měnu XCP jako žeton pro provádění Counterparty transakcí.

Altcoiny

Převážná většina altcoinů je odvozena ze zdrojových kódů bitcoinu, známé jako "klony" (anglicky forks). Některé jsou implementovány "od píky", založené na bitcoinovém modelu ale bez použití bitcoinových zdrojových kódů. Altcoiny a altchainy (v další části) jsou oddělené implementace blockchainové technologie a obě formy používají vlastní blockchainy. Rozdíl v pojmech naznačuje, že altcoiny jsou hlavně používané jako měna, zatímco altchainy jsou používány pro jiné účely, nejsou prvořadě měnou.

Striktně řečeno, první hlavní "alt" klon bitcoinového zdrojového kódu nebyl altcoin, ale altchain _ *Namecoin*, který bude probrán v další části.

Měřeno datem oznámení, první altcoin byl klon bitcoinu představený v srpnu 2011, jmenoval se *IXCoin*. IXCoin a měnil několik bitcoinových parametrů, obzvláště zrychloval tvorbu měny zvýšením odměny na 96 mincí za blok.

V září 2011 byl spuštěn *Tenebrix*. Tenebrix byl první kryptoměnou používající alternativní algoritmus důkazu prací, pojmenovaný "proof-of-work algorithm","alternative") *scrypt*, algoritmus původně navržený pro posilování hesel (jejich odolnosti proti hrubé síle). Počáteční cíl Tenebrixu byl udělat minci, která je odolná těžbě na GPU a ASIC, použitím paměťově náročného algoritmu. Tenebrix neuspěl jako měna, ale stal se základem pro Litecoin, který dosáhl velkého úspěchu a zplodil stovky klonů.

Litecoin, kromě používání scyptu jako algoritmu důkazu prací, také zavedl rychlejší tvorbu bloků, cílových 2,5 minuty místo bitcoinových 10 minut. Výsledná měna je velebena jako "stříbro k bitcoinovému zlatu" a je zamýšlena jako odlehčená alternativní měna. Díky rychlejšímu času potvrzování a omezení na 84 miliónu celkových jednotek měny, mnoho přívržencůLitecoinu věří, že je lepším řešením pro maloobchodní transakce než bitcoin.

Altcoiny se začaly množin v letech 2011 a 2012, buďto založené na bitcoinu nebo na Litecoinu. Do roku 2013 soupeřilo o pozici na trhu 20 altcoinů. Koncem roku 2013 toto číslo narostlo na 200, čímž se rok 2013 stal "rokem altcoinů". Růst altcoinů pokračoval v roce 2014, v době psaní této knihy existovalo přes 500 altcoinů. Více než polovina z dnešních altcoinů jsou klonem Litecoinu.

Tvorba altcoinů je snadná, proto jich máme více jak 500. Většina altcoinů se liší od bitcoinu velmi málo a nenabízejí nic hodného studia. Jedná se o kopie, která po krátkém růstu své počáteční ceny ztratili svoji hodnotu. Existuje však několik výjimek přinášejících důležité inovace. Tyto altcoiny se významně liší svým přístupem nebo přidávají inovace do vzoru bitcoinového návrhu. Jsou tři hlavní oblasti, kterými se altcoiny liší od bitcoinu:

- Odlišná měnová politika
- rozdílný mechanizmus shody nebo důkazu prací
- Zvláštní funkce, jako posílená anonymita

Pro více informací, viz graphical timeline of alt coins and alt chains.

Vyhodnocení altcoinů

Mezi tolika altcoiny, které existují, jak se rozhodnout, které z nich jsou hodny naší pozornost? Některé altcoiny se pokouší dosáhnout širokého rozšíření a použití jako měna. Jiné slouží k pokusům a testování rozdílných funkcí peněžních modelů. Mnoho z nich slouží jen k rychlému zbohatnutí jejich tvůrců. Pro vyhodnocení altcoinů, se dívám na jejich definované vlastnosti a jejich tržní metriky.

Zde jsou některé z otázek k položení, jak moc se altcoin liší od bitcoinu:

- Přináší altcoin významnou inovaci?
- Je rozdíl dostatečně přesvědčivým, aby přilákal bitcoinové uživatele?
- Řeší altcoin zajímavou mezeru na trhu nebo aplikaci?
- Může altcoin přilákat dostatek těžařů, aby ho zabezpečili proti útokům na shodu.

Zde jsou klíčové finanční a tržní metriky pro zvážení:

- Jaká je celková tržní kapitalizace altcoinu?
- Odhad počtu uživatelů/peněženek vlastnících altcoin.
- Kolik obchodníků přijímá altcoin?
- Kolik transakcí denně je vykonáno v tomto altcoinů.
- Jaká hodnota je denně přenášena?

V této kapitole se hlavně zaměříme na technické vlastnosti a inovační potenciál altcoinů vyjádřený první množinou otázek.

Alternativní měnové parametry: Litecoin, Dogecoin, Freicoin

Bitcoin má několik měnových parametrů, které mu dávají zvláštní povahu deflační měny s pevně daným uvolňováním mincí. Je omezen na 21 milionů hlavních jednotek měny (nebo 21 biliard vedlejších jednotek). má geometricky se snižující míru vydávání mincí a má 10-minutové blokové "srdeční tepy", které ovládají rychlost potvrzování transakcí a vytváření nových mincí. Mnoho altcoinů vylepšilo hlavní parametry, aby dosáhli rozdílné měnové politiky. Mezí stovkami těchto altcoinů, mezi nejpozoruhodnější příklady patří následující.

Litecoin

Jeden z prvních altcoinů, uvolněny 2011, Litecoin je druhou nejúspěšnější digitální měnou po bitcoinu.

Jeho hlavní inovací bylo použití *scrypt* jako algoritmu důkazu prací (zděděno z Tenebix) a rychlejší/měkčí měnové parametry.

- Čas vytváření bloku: 2,5 minuty
- Počet mincí: 84 milionů v roce 2140
- Algoritmus shody: důkaz prací Scrypt
- Tržní kapitalizace: 160 milionů dolarů v polovině roku 2014

Dogecoin

Dogecoin byl spuštěn v prosinci 2013, založen jako klon Litecoinu. Dogecoin je významný, protože má měnovou politiku rychlého vydávání mincí a velmi vysokého celkového počtu vydaných mincí, což povzbuzuje k utrácení a dávání spropitného. Dogecoin je významný také proto, že začal jako vtip a stal se docela populárním s velkou a aktivní komunitou před prudkým poklesem v roce 2014.

- Čas vytváření bloku: 60 sekund
- Počet mincí: 100 000 000 000 (100 miliard) v roce 2015
- Algoritmus shody: důkaz prací Scrypt
- Tržní kapitalizace: 12 milionů dolarů v polovině roku 2014

Freicoin

Freicoin byl představen v červnu 2012. Je to *měna se zpoplatněným vlastnictvím*, má negativní úrokovou míru z uložené hodnoty. Hodnota uložena v Freicoinu je zatížena ročním poplatkem 4,5% p.a., aby povzbuzovala spotřebu a odrazovala od hromadění peněz. Freicoin je významný, protože implementuje měnovou politiku, která je přesně opačná k bitcoinové deflační politice. Freicoin neměl úspěch jako měna, ale je to zajímavý příklad jak rozdílné měnové politiky mohou být vyjádřené altcoiny.

- Čas vytváření bloku: 10 minut
- Počet mincí: 100 milionů v roce 2140
- Algoritmus shody: důkaz prací SHA256
- Tržní kapitalizace: 130 tisíc dolarů v polovině roku 2014

Inovace shody: Peercoin, Myriad, Blackcoin, Vericoin, NXT

Bitcoinový mechanizmus shody je založen na důkazu prací použitém algoritmem SHA256. První altcoiny představily scrypt jako alternativní algoritmus důkazu prací, jako způsob udělat těžbu přátelštější pro CPU a méně výhodný k centralizaci s ASIC. Od té doby inovace v mechanizmu shody postupovaly bouřlivým tempem. Několik altcoinů převzalo různé algoritmy jako scrypt, scrypt-N, Skein, Groestl, SHA3, X11, Blake, a další. Některé altcoiny kombinovaly více algoritmů pro důkaz prací. V roce 2013 jsme viděli objev alternativy k důkazu prací, nazvané *důkaz podílem*, který tvoří základy mnoha moderních altcoinů.

Důkaz podílem je systém, ve kterém majitelé měny mohou "vsadit" měnu jako úročené zajištění. Něco jako osvědčení o vkladu, účastníci mohou vyhradit část měny ve svém držení, která vydělává ve formě nově vydané měny (jako platby úroků) a transakčních poplatků.

Peercoin

Peercoin byl představen v srpnu 2012 jako první altcoin používající pro vydávání nových mincí kombinaci algoritmů důkaz prací a důkaz podílem.

- Čas vytváření bloku: 10 minut
- Počet mincí: bez omezení
- Algoritmus shody: Hybridní, důkaz podílem s počátečním důkazem prací
- Tržní kapitalizace: 14 milionů dolarů v polovině roku 2014

Myriad

Myriad byl představen v únoru 2014 a je významný, protože používá zároveň pět různých algoritmů důkazu prací (SHA256d, Scrypt, Qubit, Skein, a Myriad-Groestl), s obtížností se měnící pro každý z algoritmů závisející na jejich využívání těžaři. Záměrem je udělat Myriad imunní vůči specializovaným ASIC a centralizaci a tím odolnější vůči útokům proti shodě, protože několik těžících algoritmů by muselo být napadeno současně.

- Čas vytváření bloku: 30 sekund v průměru (2,5 minutový cíl pro každý algoritmus těžby)
- Počet mincí: 2 miliardy v roce 2024
- Algoritmus shody: více algoritmů důkazu prací
- Tržní kapitalizace: 120 tisíc dolarů v polovině roku 2014

Blackcoin

Blackcoin byl představen v únoru 2014 a používá důkaz podílem jako algoritmus shody. Je významný, protože představil "vícemincové těžební skupiny", automatické přepínání mezi těžbou jednotlivých altcoinů v závislosti na výhodnosti.

- Čas vytváření bloku: 1 minuta
- Počet mincí: bez omezení
- Algoritmus shody: důkaz podílem
- Tržní kapitalizace: 3,7 milionů dolarů v polovině roku 2014

VeriCoin

VeriCoin byl spuštěn v květnu 2014. Používá důkaz podílem jako algoritmus shody s proměnlivou úrokovou sazbou, která se dynamicky mění v závislosti na tržních silách nabídky a poptávky. Je to také první altcoin s funkcí automatické směny na bitcoin pro platby v bitcoinu z peněženky.

- Čas vytváření bloku: 1 minuta
- Počet mincí: bez omezení
- Algoritmus shody: důkaz podílem
- Tržní kapitalizace: 1,1 milionu dolarů v polovině roku 2014

NXT

NXT (výslovnost "Next") je altcoin s "čistým" důkazem podílem, který nepoužívá těžbu důkazem prací. NXT je od píky implementovaná kryptoměna, není klonem bitcoinu nebo nějakého jiného altcoinu. NXT implementuje mnoho pokročilých funkcí, včetně registru názvů (podobné k Namecoin), decentralizovanou burzu aktiv (podobné k barevným mincím), integrovanou decentralizovanou a bezpečnou výměnu zpráv (podobné k Bitmessage) a pověřené podílnictví (svěřit jinému podíl za účelem provedení důkazu podílem). Příznivci NXT ho nazývají "novou generací" nebo kryptoměnou 2.0.

- Čas vytváření bloku: 1 minuta
- Počet mincí: bez omezení
- Algoritmus shody: důkaz podílem
- Tržní kapitalizace: 30 milionů dolarů v polovině roku 2014

Inovace v druhém účelu těžby: Primecoin, Curecoin, Gridcoin

Bitcoinový algoritmus důkazu prací má jen jeden důvod: zabezpečit bitcoinovou síť. V porovnání s bezpečností tradičních platebních systémů, cena těžby není velmi vysoká. Nicméně těžba, bývá mnohými kritizována jako "plýtvání." Další generace altcoinu se snaží vyhnout těmto obavám. Algoritmy důkazu prací s druhým účelem řeší konkrétní "užitečný" problém při hledání důkazu prací pro zabezpečení sítě. Riziko přidání vnějšího využití na bezpečnost měny spočívá v tom, že byl přidán vnější vliv na křivku nabídky a poptávky.

Primecoin

Primecoin byl představen v červenci 2013. Jeho algoritmus důkazu prací hledá prvočísla, počítá Cunninghamovy a podvojné řetězy prvočísel. Prvočísla jsou užitečná v množství vědních oborů. Primecoin blockchain obsahuje objevená prvočísla, proto zveřejňuje veřejný záznam vědeckého objevu vedle veřejného účetního systému transakcí.

- Čas vytváření bloku: 1 minuta
- Počet mincí: bez omezení
- Algoritmus shody: důkaz prací objevující řetězy prvočísel
- Tržní kapitalizace: 1,3 milionů dolarů v polovině roku 2014

Curecoin

Curecoin byl oznámen v květnu 2013. Kombinuje algoritmus důkazu prací SHA256 s výzkumem skládání proteinů v projektu Folding@Home. Skládání proteinů je výpočetně náročnou simulací biochemických vztahů mezi proteiny, používané pro objevování nových léků pro léčbu nemocí.

- Čas vytváření bloku: 10 minut
- Počet mincí: bez omezení
- Algoritmus shody: důkaz prací s výzkumem skládání proteinů
- Tržní kapitalizace: 58 tisíc dolarů v polovině roku 2014

Gridcoin

Gridcoin byl představen v říjnu 2013. Důkaz prací pomocí scrypt doplňuje dotací za účast v BOINC otevřené gridové výpočty. BOINC — Berkeley otevřená infrastruktura pro síťové výpočty — je otevřený protokol pro vědecký výzkum gridových výpočtů, umožňující účastníkům sdílet jejich výpočetní výkon pro širokou škálu akademických výzkumných výpočtů. Gridcoin používá BOINC jako obecnou výpočetní platformu než, aby řešil konkrétní vědecký problém jako hledání prvočísel nebo skládání proteinů.

- Čas vytváření bloku: 150 sekund
- Počet mincí: bez omezení
- Algoritmus shody: Důkaz prací a dotace za BOINC gridové výpočty
- Tržní kapitalizace: 122 tisíc dolarů v polovině roku 2014

Altcoiny s posílenou anonymitou: CryptoNote, Bytecoin, Monero, Zerocash/Zerocoin, Darkcoin

Bitcoin je často nesprávně označován jako "anonymní" měna. Ve skutečnosti, je docela snadné spojit identity bitcoinových adres, použitím analýzy velkých dat, spojit adresy jednu s druhou a vytvořit souhrnný obrázek zvyků utrácení bitcoinů nějaké osoby. Několik altcoinů usiluje o řešení tohoto problému přímo jejich zaměřením na silnou anonymitu. První takový pokus je s největší pravděpodobností *Zerocoin*, metacoin protokol pro zajištění anonymity na vrcholu bitcoinu, představen v článku v roce 2013 na IEEE konferenci Symposium on Security and Privacy. Zerocoin bude implementovaný jako zcela samostatný altcoin zvaný Zerocash, v čase psaní této knihy. Alternativní přístup k anonymitě byl zahájen s *CryptoNote*, článkem publikovaným v říjnu 2013. CryptoNote je podkladová technologie, která je implementována několika altcoinovými klony, popisovanými dále. Navíc k Zerocash a CryptoNotes, existuje několik dalších anonymních altcoinů jako Darkcoin, který používá utajené adresy a míchání transakcí pro zajištění anonymity.

Zerocoin/Zerocash

Zerocoin je teoretický postup anonymity digitální měny představený v roce 2013 výzkumníky z university Johns Hopkins. Zerocash je altcoinová implementace Zerocoinu a je nyní ve vývoji, není

dosud vydána.

CryptoNote

CryptoNote je referenční implementace altcoinu, která poskytuje základy anonymity digitální hotovosti. Byl představen v říjnu 2013. Je navržen, aby mohl být klonován do různých implementací a má vestavěný mechanizmus periodického vymazávání dat, takže je nepoužitelný jako měna samotná. Několik altcoinů se narodilo z CryptoNote, včetně (BCN), Aeon (AEON), Boolberry (BBR), duckNote (DUCK), Fantomcoin (FCN), Monero (XMR), MonetaVerde (MCN), a Quazarcoin (QCN). CryptoNote je také významný také proto, že se jedná o implementaci od píky, není to klon bitcoinu.

Bytecoin

Bytecoin byl první implementací narozenou z CryptoNote, nabízející životaschopnou anonymní měnu založenou na technologii CryptoNote. Bytecoin byl spuštěn v červenci 2012. Poznámka, předtím existoval altcoin pojmenovaný Bytecoin se symbolem BCN, zatímco Bytecoin odvozený z CryptoNote má symbol BCN. Bytecoin používá pro důkaz prací algoritmus Cryptonight, který požaduje alespoň 2 MB RAM pro jednu instanci, což ho dělá nevhodným pro GPU a ASIC těžbu. Bytecoin zdědil z CryptoNote kruhové podpisy, nespojitelné transakce a anonymitu odolnou analýze blockchainu.

- Čas vytváření bloku: 2 minuty
- Počet mincí: 184 miliard
- Algoritmus shody: Důkaz prací Cryptonight
- Tržní kapitalizace: 3 miliony dolarů v polovině roku 2014

Monero

Monero je další implementace CryptoNote má mírně plošší křivku vydávání nových mincí než Bytecoin, vydává 80 % měny v prvních čtyřech letech. Nabízí stejné vlastnosti anonymity zděděné z CryptoNote.

- Čas vytváření bloku: 1 minuta
- Počet mincí: 18,4 milionů
- Algoritmus shody: Důkaz prací Cryptonight
- Tržní kapitalizace: 5 milionů dolarů v polovině roku 2014

Darkcoin

Darkcoin byl spuštěn v lednu 2014. Darkcoin implementuje anonymní měnu používající protokol pro míchání všech transakcí zvaný DarkSend. Darkcoin je také významný používáním 11 kol různých hašovacích funkcí (blake, bmw, groestl, jh, keccak, skein, luffa, cubehash, shavite, simd, echo) pro algoritmus důkazu prací.

- Čas vytváření bloku: 2,5 minuty

- Počet mincí: maximum 22 milionů
- Algoritmus shody: Více algoritmů pro více kol důkazu prací
- Tržní kapitalizace: 19 milionů dolarů v polovině roku 2014

Neměnové altchainy

Altchainy jsou alternativní implementace návrhového vzoru blockchain, které primárně neslouží jako měna. Mnoho obsahuje měnu, protože měna je použita jako žeton pro obsazení něčeho jiného, jako je zdroj nebo kontrakt. Měna, jinými slovy, není hlavní cíl platformy, je to vedlejší funkce.

Namecoin

Namecoin byl první klon bitcoinového zdrojového kódu. Namecoin je decentralizovaný registrační a převodová platforma pro dvojice klíč-hodnota používající blockchainu. podporuje celosvětový registr doménových jmen podobný registračnímu systému doménových jmen na internetu. Namecoin je aktuálně používán jako alternativní služba doménových jmen (DNS) pro doménu prvního řádu .bit. Namecoin, může být také použit pro registraci jmen a dvojic klíč-hodnota v dalších jmenných prostorech; pro ukládání např. emailových adres, šifrovaných klíčů, SSL certifikátů, podpisů souborů, volební systémy, burzovní certifikáty a nesčetné množství dalších aplikací.

Namecoin systém obsahuje Namecoin měnu (symbol NMC), která se používá k placení poplatků za registrace převody jmen. Při současných cenách, poplatek za registraci jména je 0,01 NMC, což je přibližně 1 americký cent. Jako v bitcoinu, poplatky jsou sbírány těžaři Namecoinu.

Namecoin má základní parametry shodné s bitcoinem:

- Čas vytváření bloku: 10 minut
- Počet mincí: 21 milionů v roce 2140
- Algoritmus shody: důkaz prací SHA256
- Tržní kapitalizace: 10 milionů dolarů v polovině roku 2014

jmenné prostory Namecoinu nejsou omezené, a každý může použít jmenný prostor jakýmkoliv způsobem. Nicméně, některé jmenné prostory jsou dohodnuté ve specifikaci, takže při jejích čtení z blockchainu, software na aplikační úrovní ví, jak je číst a zpracovávat. Pokud jsou poškozeny, pak bez ohledu na software, který pro čtení ze specifického jmenného prostoru používáte, vyhodí chybu. Některé oblíbené jmenné prostory jsou:

- d/ jmenný prostor doménových jmen pro doménu .bit id/ je jmenný prostor pro ukládání identifikací osob, jako emailových adres, PGP klíčů, atd. u/ je dodatečná, více strukturovaná specifikace pro uložení identit (založená na openspecs)

Namecoin klient je velmi podobný Bitcoin Core, protože je odvozen z toho samého zdrojového kódu. Po instalaci, klient stáhne úplnou kopii Namecoin blockchainu a poté bude připraven dotazovat se registrovat jména. Obsahuje tří hlavní příkazy:

name_new

Dotaz nebo předběžná registrace jména

name_firstupdate

Registruje jméno a udělá registraci veřejnou

name_update

Změní podrobnosti nebo obnoví registraci jména

Například, pro registraci domény mastering-bitcoin.bit použijeme příkaz name_new následovně:

```
$ namecoind name_new d/mastering-bitcoin
```

```
[
    "21cbab5b1241c6d1a6ad70a2416b3124eb883ac38e423e5ff591d1968eb6664a",
    "a05555e0fc56c023"
]
```

Příkaz name_new registruje zábor jména, vytvoří haš jména s náhodným klíčem. Tyto dva řetězce jsou vráceny funkci name_new a jsou haš a náhodný klíč (a05555e0fc56c023 v předchozím příkladě), který může být použít k vytvoření veřejné registrace. Jakmile je zábor zaznamenán na Namecoin blockchainu, může být převeden na veřejnou registraci příkazem name_firstupdate, poskytnutím náhodného klíče:

```
$ namecoind name_firstupdate d/mastering-bitcoin a05555e0fc56c023 "{"map": {"www":
{"ip":"1.2.3.4"}}}}"
b7a2e59c0a26e5e2664948946ebeca1260985c2f616ba579e6bc7f35ec234b01
```

Tento příklad vytvoří spojení mezi doménovým jménem www.mastering-bitcoin.bit a IP adresou 1.2.3.4. Vrácený haš je transakční ID a může být použito k vyhledání této registrace. Můžete zobrazit seznam vámi registrovaných jmen příkazem name_list:

```
$ namecoind name_list
```

```
[
    {
        "name" : "d/mastering-bitcoin",
        "value" : "{map: {www: {ip:1.2.3.4}}}}",
        "address" : "NCccBXrRUahAGrisBA1BLPWQfSrups8Geh",
        "expires_in" : 35929
    }
]
```

Namecoin registrace potřebují být aktualizovány každých 36 000 bloků (přibližně 200 až 250 dní). Příkaz name_update je bez poplatku a proto obnova domény v Namecoin je zdarma. Poskytovatelé třetích stran mohou řešit registrace, automatické obnovy a aktualizace pomocí webového rozhraní, za malý poplatek. S poskytovatelem třetí strany se můžete vyhnout potřebě provozovat Namecoin klienta, ale ztratíte nezávislou kontrolu decentralizovaného registrů jmen nabízeného Namecoinem.

Ethereum

Ethereum je turingovsky úplná platforma pro zpracování a vykonávání procesů založená na blockchainovém účetním systému. Není to klon Bitcoinu, ale je zcela nezávislý návrhově a implementačně. Ethereum má vestavěnou měnu, zvanou *ether*, která je potřebná pro platby za vykonávání kontraktů. Blockchain Etherea zaznamenává *kontrakty*, které jsou vyjádřeny nízkoúrovňově, jako bajtkód, turingovsky úplného jazyka. V podstatě, kontrakt je program, který běží na každém uzlu systému Etherea. Ethereum kontrakty mohou ukládat data, zasílat a přijímat ether platby, ukládat ether a vykonávat nekonečnou škálu (proto turingovsky úplný) výpočetních akcí, chovající se jako decentralizovaní autonomní softwaroví agenti.

Ethereum může implementovat docela složité systémy, které jsou jinak implementovány jako samostatné altchainy. Například, následuje, Namecoinu podobný kontrakt registrující jména napsaný v Ethereum (přesněji, napsaný ve vysokoúrovňovém jazyku, který může být do kódu Etherea zkompilován):

```
if !contract.storage[msg.data[0]]: # Je klíč obsazen?
    # Zabereme ho
    contract.storage[msg.data[0]] = msg.data[1]
    return(1)
else:

    return(0) // Pokud je klíč obsazen, nebudeme dělat nic
```

Budoucnost měn

Budoucnost kryptografických měn celkově je dokonce světlejší než budoucnost bitcoinu. Bitcoin

představil úplně novou formu decentralizované organizace a shody, která se vyvinula ve stovky neuvěřitelných inovací. Tyto inovace mají vliv na široké sektory ekonomie, od výzkumu distribuovaných systémů, přes finance, ekonomii, měny, centrální bankovnictví, správu a řízení. Mnoho lidských aktivit, které předtím vyžadovali centralizované instituce nebo organizace, aby zastávaly směrodatný nebo důvěryhodný bod kontroly nyní mohou být decentralizované. Objev blockchainu a shody systému značně sníží cenu organizace a koordinace velké škály systémů při odstranění příležitostí pro koncentraci síly, korupci a regulační zásahy.

Bezpečnost bitcoinu

Bezpečnost bitcoinu je výzvou, protože bitcoin není abstraktním odkazem na hodnotu jako je stav bankovního účtu. Bitcoin je mnohem více jako digitální hotovost nebo zlato. Pravděpodobně jste slyšeli výraz "Držení je devět desetin vlastnictví" Dobře, v bitcoinu je držení deset desetin vlastnictví. Držení klíčů k odemčení bitcoinu je ekvivalentem k vlastnictví hotovost nebo kusu drahého kovu. Můžete je ztratit, někam je založit, nechat si je ukrást nebo omylem dát někomu špatné množství bitcoinů. V každém z těchto případů je uživatel bezbranný, jako když ztratí hotovost na veřejném chodníku.

Nicméně bitcoin má schopnosti, které hotovost, zlato a bankovní účty nemají. Bitcoinová peněženka, obsahující vaše klíče, může být zálohována stejně jako jakýkoliv jiný soubor. Může být uchována v mnoha kopiích, dokonce záložní kopie může být vytištěna na papír. Nemůžete zálohovat hotovost, zlato nebo bankovní účty. Bitcoin je značně odlišný od všeho co bylo před ním, proto musíme o bezpečnosti bitcoinu také přemýšlet novým způsobem.

Principy bezpečnosti

Základní princip bitcoinu je decentralizace a ta významných způsobem ovlivňuje bezpečnost. Centralizovaný model, jaký je u tradičních bank nebo platebních systémů, závisí na kontrole přístupu do systému a držení falešných hráčů mimo systém. V protikladu k tomu decentralizovaný systém jako bitcoin přenáší odpovědnost a kontrolu na uživatele. Protože bezpečnost sítě je založena na důkazu prací, není zde kontrola přístupu, síť může být otevřená a nemusí používat šifrování pro přenos bitcoinových transakcí.

V tradičních platebních sítích, jako je systém kreditních karet, není platba omezena, protože obsahuje soukromý identifikátor uživatele (číslo kreditní karty). Po počátečním stržení platby z účtu může kdokoliv s přístupem k tomuto identifikátoru z účtu uživatele strhávat další a další platby. Proto byla platební síť zabezpečena šifrováním mezi koncovými uživateli a musí zajistit, že žádní "eavesdroppers") tajní odposlouchávači nebo prostřednici nemohou ohrozit platební provoz při přenosu nebo při jeho uložení (v klidu). Když falešný hráč získá přístup do systému, může ohrozit aktuální transakce *a* platební žetony může využít pro vytvoření nových transakcí. Ještě hůře, jsou ohrožena data zákazníků. Zákazníci jsou vystaveni krádeži identity a musejí podniknout akce, aby zabránili dalšímu zneužívání ohrožených účtů.

Bitcoin je naprosto odlišný. Bitcoinová transakce opravňuje pouze přesun konkrétní hodnoty konkrétnímu příjemci a nemůže být padělána nebo změněna. Neodkrývá žádné soukromé informace, jako jsou identity jednotlivých stran a nemůže být použita k potvrzení dalších plateb. Proto bitcoinová platební síť nepotřebuje být šifrována nebo chráněna před odposlechem. Ve skutečnosti, můžete šířit bitcoinové transakce přes otevřený veřejný kanál, jako je nezabezpečení WiFi nebo Bluetooth, aniž by došlo ke snížení bezpečnosti.

Bitcoinový decentralizovaný bezpečnostní model dává velké množství moci do rukou uživatelů. S mocí přichází zodpovědnost za uchování utajení klíčů. Pro mnoho uživatelů není lehké toto dělat, speciálně na běžných výpočetních zařízeních jako jsou k internetu připojené chytré telefony nebo laptopy.

Přestože bitcoinový decentralizovaný model zabraňuje masovému ohrožení jako hrozí u kreditních karet, mnoho uživatelů není schopno adekvátně ochránit své klíče a jsou vykradeni, jeden po druhém.

Bezpečný vývoj bitcoinového systému

Nejdůležitějším principem pro bitcoinové vývojáře je decentralizace. Většina vývojářů je zvyklá na centralizované bezpečnostní modely a může být sváděna k aplikaci těchto modelů do jejich bitcoinových aplikací s katastrofálními výsledky.

Bitcoinová bezpečnost spoléhá na decentralizovanou kontrolu klíčů a na nezávislé transakce ověřované těžaři. Pokud chcete využít Bitcoinovou bezpečnost, musíte se ujistit, že zůstáváte uvnitř Bitcoinového bezpečnostního modelu. Jednoduše řečeno, nepřebírejte kontrolu nad klíči od uživatelů a neukládejte transakce mimo blockchain.

Například, mnoho z prvních bitcoinových směnáren koncentrovalo finanční prostředky všech uživatelů v jediné "horké" peněžence s klíči uloženými na jediném serveru. Takovýto návrh zbavuje uživatele možnosti kontroly a centralizuje kontrolu nad klíči v jediném systému. Mnoho takových systému bylo vykradeno s katastrofálními následky pro jejich zákazníky.

Další rozšířenou chybou je dělání transakcí mimo blockchain, Toto nevhodné úsilí se snaží snižovat poplatky za transakce nebo zrychlovat zpracování transakcí. Sytém mimo blockchain zaznamenává transakce v interním centralizovaném účetním systému a pouze občasně je synchronizuje s blockchainem. Tato praktika, znova, nahrazuje decentralizovanou bezpečnosti bitcoinu uzavřeným centralizovaným přístupem. Nedostatečně zabezpečené centralizované účetní systémy mohou být vykrádány a jejich záznamy mohou být falšovány bez povšimnutí.

Dokud nejste připraveni masivně investovat do operační bezpečnosti, mnoho vrstev kontroly přístupu, auditů (jako to dělají tradiční banky), měli byste přemýšlet velmi opatrně o možnosti vzít finanční prostředky mimo kontext bitcoinové decentralizované bezpečnosti. Dokonce pokud máte finanční prostředky a disciplínu na implementaci robustního bezpečnostního systémů, jako návrh pouhé kopie křehkého modelu tradiční finanční sítě, sužované krádežemi identity, korupcí a zpronevěrami. Abyste využili výhod unikátnosti Bitcoinového decentralizovaného bezpečnostního modelu, měli byste se vyhnout pokušení centralizovaných architektur, které na vás mohou působit povědomě, ale v konečném důsledku rozvracejí Bitcoinovou bezpečnost.

Kořen důvěry

Tradiční bezpečností architektura je založena na konceptu zvaném *kořen důvěry*, ve kterém je důvěryhodné jádro použito jako základ bezpečnosti v celém systému nebo aplikaci. Bezpečnostní architektura je vyvíjena okolo tohoto kořenu důvěry v posloupnosti soustředných kruhů, jako vrstev cibule, rozšiřujících důvěru směrem z centra. Každé vrstva staví na důvěryhodnější vnitřní vrstvě užívající kontroly přístupu, digitální podpisy, šifrování a ostatní bezpečnostní primitiva. jak se softwarový systém stává více komplexním, je více náchylný k obsahování chyb, které ho dělají zranitelným k prolomení bezpečnosti. Ve výsledku, čím je softwarový systém více komplexní, tím je těžší ho zabezpečit. Koncept kořenu důvěry zajišťuje, že nejvíce důvěry je umístěno v nejméně komplexní části systému a tedy nejméně zranitelné části systémů, zatímco komplexnější software je

umístěný ve vrstvách okolo. Tato bezpečnostní architektura je opakovaná v různých měřítkách. Nejprve je vytvořen kořen důvěry okolo hardware daného systému, poté je kořen důvěry rozšířeny skrz operační systém do vysokoúrovňových systémů služeb a nakonec do mnoho serverů vrstvených v soustředných kruzích mizející důvěry.

Bitcoinová bezpečnostní architektura je rozdílná. V Bitcoinu shoda systému vytváří důvěryhodný účetní systém, který je plně decentralizovaný. Správně ověřený blockchain používá základní blok jako kořen důvěry, ze kterého je vybudován řetěz důvěry do aktuálního bloku. Bitcoinový systém může a měl by používat blockchain jako jeho kořen důvěry. Při návrhu komplexní bitcoinové aplikace, která se se skládá ze služeb mnoha mnoha různých systémů, byste měli opatrně prozkoumat bezpečnostní architekturu, aby se zjistilo, kde důvěra má byt umístěna. Nakonec, jedinou věcí, které by mělo být explicitně důvěřováno je plně ověřený blockchain. Pokud vaše aplikace explicitně nebo implicitně uděluje důvěru v něco jiného než blockchain, to by mělo být zdrojem obav, protože to přináší zranitelnost. Dobrá metoda na vyhodnocení bezpečnostní architektury vaší aplikace je zvážit každou samostatnou komponentu a vyhodnotit hypotetický scénář, ve kterém je tato komponenta napadena a je zcela pod kontrolou útočníka. Vezměte postupně každou komponentu vaší aplikace, a zhodnoťte dopad na celkovou bezpečnost pokud je tato tato komponenta napadena. Pokud vaše aplikace již není bezpečná, poté co byly některé komponenty napadeny, je to známkou toho, že jste nevhodně umístili důvěru do těchto komponent. Bitcoinová aplikace bez zranitelností by měla být zranitelná pouze pouze při napadení bitcoinového mechanizmu shody., jehož kořen pravdy je založen na nejsilnější části bitcoinové bezpečnostní architektury.

Početné příklady vykradených bitcoinových směnáren pomáhají ke zdůraznění tohoto bodu, protože jejich bezpečnostní architektury a návrhy neuspějí dokonce ani při nejběžnější kontrole. Tyto centralizované implementace investovali důvěru v početné komponenty mimo blockchain, jako jsou horké peněženky, centralizované databáze účetních systémů, zranitelné šifrování klíčů a podobná schémata.

Osvědčené postupy uživatelské bezpečnosti

Lidé používali fyzické bezpečnostní kontroly po tisíce let. Pro srovnání, naše zkušenost s digitální bezpečností je kratší než 50 let. Moderní všeobecné operační systémy jsou stále vystaveny externím hrozbám přes neustálé internetové spojení. Běží na nich tisíce softwarových komponent od stovek autorů, často s neomezeným přístupem k uživatelským souborům. Jediný kus loupežného software, mezi tisíci jiného nainstalovaného na vašem počítači, může ohrozit vaší klávesnici a soubory, ukrást jakýkoliv bitcoin uložený v peněženkové aplikaci. Úroveň údržby počítače potřebná na udržení počítače bez virů a trojských koní je za hranicí schopností téměř všech uživatelů počítačů.

Přes desetiletí výzkumu a vývoje v informační bezpečnosti, digitální aktive jsou stále bolestně zranitelné odhodlaným útočníkem. Dokonce nejvíce chráněné a omezené systémy ve firmách finančních služeb a obraného průmyslu, ve zpravodajských agenturách jsou často prolomeny. Bitcoin vytváří digitální aktiva, která mají vnitřní hodnotu a mohou být ukradena a převedena novému majiteli okamžitě a nevratně. Toto vytváří silnou pobídku pro hackery. Do současnosti, hackeři museli přeměnit informace o identitě nebo účty žetonů (jako jsou kreditní karty a bankovní účty) po jejich prolomení na hodnotu. Přes obtížnost ohraničení a praní finančních informací jsme byli svědky stále

se stupňujících krádeží. Bitcoin stupňuje tyto problémy protože nepotřebuje být ohraničen nebo vyprán, má skutečnou hodnotu digitálního aktiva.

Naštěstí bitcoin také vytváří pobídky pro vylepšení počítačové bezpečnosti. Zatímco dříve riziko napadení počítače bylo obecné a nepřímé, bitcoin udělal tuto hrozbu jasnou a zřejmou. Držet bitcoin na počítači slouží k zacílení myšlenek uživatele na potřebu zvýšení počítačové bezpečnosti. Jako přímý výsledek rychlého růstu a zvýšené adopce bitcoinu a dalších digitálních měn, jsme byli svědky vystupňování jak hackerských technik, tak i bezpečnostních řešení. Jednoduše řečeno, hackeři mají nyní šťavnatý cíl a uživatele mají jasnou výzvu k vlastní obraně.

Poslední tři roky, jako přímý výsledek adopce bitcoinu, jsme svědky obrovských inovací ve světě informační bezpečnosti ve formě hardwarového šifrování, ukládání klíčů a hardwarových peněženek, vícepodpisových technologií a digitálních úschov. V následující části prozkoumáme osvědčené postupy uživatelské bezpečnosti.

Fyzické skladování bitcoinu

Protože pro většinu uživatelů je pohodlnější fyzická bezpečnost než informační bezpečnost, velmi účinná metoda pro ochranu bitcoinů je jejich konverze na fyzickou formu. Bitcoinové klíče nejsou nic víc než dlouhá čísla. To znamená, že mohou být uložena ve fyzické formě, jako vytištěné na papíře nebo vyleptané na železnou minci. Ochrana klíčů se poté stává tak stejně jednoduchá jako fyzická ochrana vytištěné kopie bitcoinových klíčů. Množina klíčů, které jsou vytištěny na papíře, se nazývá *papírová peněženka* a existuje mnoho bezplatných nástrojů, které mohou být použity na jejich vytvoření. Osobně uchovávám převážnou většinu mých bitcoinů (přes 99%) uložených v papírové peněžence, zašifrované pomocí BIP0038, v mnoha kopiích uložených v trezorech. Uložení bitcoinů offline je nazýváno *studené úložiště* a je to jedna z nejvíce účinných bezpečnostních technik. Systém studeného úložiště je takový, kde klíče jsou vytvářeny na offline počítači (nikdy nepřipojeným na internet) a ukládány offline buďto na papír nebo na digitální media jako jsou USB disky.

Hardwarové peněženky

V dlouhodobém hledisku bude bitcoinová bezpečnost mít stále častěji formu hardwarových peněženek odolných proti neoprávněné manipulaci. Na rozdíl od chytrých telefonů nebo stolních počítačů, bitcoinová hardwarová peněženka má pouze jediný účel: udržovat bitcoiny bezpečně. Bez všeobecného software, který by byl hrozbou a s omezeným rozhraním, hardwarové peněženky se stávají převládající metodou ukládání bitcoinu. Příkladem takovéto hardwarové peněženky je Trezor, který vyrábí česká firma SatoshiLabs.

Vyvážení rizika

Přestože většina uživatelů se oprávněně obávají krádeže bitcoinů, je zde ještě větší riziko. Datové soubory se čas od času ztrácejí. Pokud obsahují bitcoiny, ztráta je o to více bolestivá. Ve snaze zabezpečit jejich bitcoinové peněženky, uživatelé musejí být velmi opatrní a nezajít příliš daleko, aby to neskončilo ztrátou bitcoinů. V červenci 2011 dobře známý osvětový a výukový projekt ztratil téměř 7000 bitcoinů. V jejich úsilí předcházet krádeži, majitelé implementovali složitou posloupnost šifrovaných záloh. Na konci nešťastnou náhodou ztratili šifrovací klíče. Zálohy se staly bezcennými a

přišli o jmění. Jako ukrytí peněz jejich pohřbením v poušti, pokud zabezpečíte své bitcoiny příliš dobře, možná je nebudete schopni zpátky nalézt.

Diverzifikace rizika

Chcete uchovávat celé čisté jmění v hotovosti ve vaší peněžence? Většina lidí toto bude považovat za nezodpovědné, zatímco bitcoinoví uživatelé často uchovávají všechny své bitcoiny v jediné peněžence. Místo toho, uživatelé by měli rozprostřít riziko mezi více a různých bitcoinových peněženek. Obezřetní uživatelé budou udržovat pouze malou část (pravděpodobně méně než 5%) jejich bitcoinů v online nebo mobilní peněžence jako "kapesné". Zbytek by měl být rozdělen mezi několik odlišných skladovacích mechanizmů jako peněženka ve stolním počítači a offline (studené) úložiště.

Vícepodpisovost a správa

Kdykoliv společnost nebo jedinec uchovává velké množství bitcoinů, mělo by dojít ke zvážení možnosti použít vícepodpisové adresy. Vícepodpisové adresy zabezpečují finanční prostředky požadavkem více než jednoho podpisu pro provedení platby. Podpisové klíče by měly být uloženy na několika odlišných místech a pod kontrolou různých lidí. Ve firemním prostředí, například, klíče by měly být vytvářeny nezávisle a drženy několika členy vedení, což zajistí, že žádná jednotlivá osoba nemůže zneužít finanční prostředky. Vícepodpisová adresa může také nabízet redundanci, když jedna osoba drží několik klíčů, které jsou uloženy na různých místech.

Pro případ smrti

Jedna důležitá bezpečnostní úvaha je často přehlížena dostupnost, speciálně v kontextu neschopnosti nebo smrti držitele klíče. Bitcoinoví uživatelé jsou poučování, aby používali složitá hesla a udržovali své klíče bezpečně a tajně, nesdíleli je s nikým jiným. Naneštěstí tento postup téměř znemožňuje rodinným příslušníkům uživatelem navrátit jakékoliv finanční prostředky, pokud uživatel není schopný je odemknout. V mnoha případech, ve skutečnosti, rodina uživatele bitcoinu může být zcela nevědoma o existenci bitcoinových finančních prostředků.

Pokud máte mnoho bitcoinu, měli byste zvážit sdělení přístupových podrobností důvěryhodnému příbuznému nebo právníkovi. Složitější scénář pro případ smrti může být nastaven pomocí vícepodpisového přístupu a plánování majetku pomocí právníka specializovaného na pozůstalost digitálních aktiv.

Závěr

Bitcoin je zcela nová, bezprecedentní a komplexní technologie. V průběhu času budou vyvinuté lepší bezpečnostní nástroje a postupy, které budou snazší pro použití neexperty. Nyní bitcoinoví uživatelé mohou využít mnoho rad diskutovaných v této knize a a užívat si bezpečný a bezproblémový bitcoinový zážitek.

Appendix A: Příkazy Bitcoinového Průzkumníka (bx)

```
Použití:  bx COMMAND [--help]

Informace: bx příkazy jsou:

address-decode
address-embed
address-encode
address-validate
base16-decode
base16-encode
base58-decode
base58-encode
base58check-decode
base58check-encode
base64-decode
base64-encode
bitcoin160
bitcoin256
btc-to-satoshi
ec-add
ec-add-secrets
ec-multiply
ec-multiply-secrets
ec-new
ec-to-address
ec-to-public
ec-to-wif
fetch-balance
fetch-header
fetch-height
fetch-history
fetch-stealth
fetch-tx
fetch-tx-index
hd-new
hd-private
hd-public
hd-to-address
hd-to-ec
hd-to-public
hd-to-wif
```

```
help
input-set
input-sign
input-validate
message-sign
message-validate
mnemonic-decode
mnemonic-encode
ripemd160
satoshi-to-btc
script-decode
script-encode
script-to-address
seed
send-tx
send-tx-node
send-tx-p2p
settings
sha160
sha256
sha512
stealth-decode
stealth-encode
stealth-public
stealth-secret
stealth-shared
tx-decode
tx-encode
uri-decode
uri-encode
validate-tx
watch-address
wif-to-ec
wif-to-public
wrap-decode
wrap-encode
```

Více informací lze nalézt na see the Bitcoin Explorer home page a Bitcoin Explorer user documentation.

Příklady použití bx příkazů

Předvedeme si příklady použití příkazů Bitcoinového Průzkumníka (v originále Bitcoin Explorer). Budeme experimentovat s klíči a adresami.

Vytvoříme náhodné semínko příkazem seed, které používá generátor náhodných čísel poskytovaný

operačním systémem. Použijeme semínko v příkazu ec-new k vytvoření nového soukromého klíče. Standardní výstup uložíme do souboru *private_key*:

```
$ bx seed | bx ec-new > private_key
$ cat private_key
73096ed11ab9f1db6135857958ece7d73ea7c30862145bcc4bbc7649075de474
```

Nyní vytvoříme veřejný klíč ze soukromého klíče za použití příkazu ec-to-public. Standardní vstup načteme ze souboru *private_key* file a standardní výstup příkazu uložíme do nového souboru *public_key*:

```
$ bx ec-to-public < private_key > public_key
$ cat public_key
02fca46a6006a62dfdd2dbb2149359d0d97a04f430f12a7626dd409256c12be500
```

Můžeme přeformátovat standardní vstup načtený ze souboru public_key na adresu za použití "Bitcoin Explorer","ec-to-address command") příkazu ec-to-address.

```
$ bx ec-to-address < public_key
17re1S4Q8ZHyCP8Kw7xQad1Lr6XUzWUnkG
```

Klíče vytvořené tímto způsobem vytvoří nedeterministickou peněženku typu 0. To znamená, že každý klíč je vytvořen z nezávislého semínka. Příkazy Bitcoinového Průzkumníka umožňují generovat klíče deterministicky podle BIP0032. V tomto případě je hlavní klíč vytvořen ze semínka a následně je deterministicky rozšiřovat. Vytvořený strom odvozených klíčů tvoří deterministickou peněženku typu 2.

Nejprve použije příkazy seed a hd-new na vytvoření hlavního klíče, který bude použít pro odvození stromu klíčů.

```
$ bx seed > seed
$ cat seed
eb68ee9f3df6bd4441a9feadec179ff1

$ bx hd-new < seed > master
$ cat master
xprv9s21ZrQH143K2BEhMYpNQoUvAgiEjArAVaZaCTgsaGe6LsAnwubeiTcDzd23mAoyizm9cApe51gNfLMkBqkYo
WWMCRwzfuJk8RwF1SVEpAQ
```

Nyní použijeme příkaz k vytvoření hd-private+ naostřeného klíče k "účtu" a posloupnosti dvou soukromých klíčů k tomuto účtu.

```
$ bx hd-private --hard < master > account
$ cat account
xprv9vkDLt81dTKjwHB8fsVB5QK8cGnzveChzSrtCfvu3aMWvQaThp59ueufuyQ8Qi3qpjk4aKsbmbfxwcgS8PYbg
oR2NWHeLyvg4DhoEE68A1n

$ bx hd-private --index 0 < account
xprv9xHfb6w1vX9xgZyPNXVgAhPxSsEkeRcPHEUV5iJcVEsuUEACvR3NRY3fpGhcnBiDbvG4LgndirDsia1e9F3DW
PkX7Tp1V1u97HKG1FJwUpU

$ bx hd-private --index 1 < account
xprv9xHfb6w1vX9xjc8XbN4GN86jzNAZ6xHEqYxzbLB4fzHFd6VqCLPGRZFsdjsuMVERadbgDbziCRJru9n6tzEWr
ASVpEdrZrFidt1RDfn4yA3
```

Dále použijeme příkaz hd-public command k vytvoření odpovídajících dvou veřejných klíčů.

```
$ bx hd-public --index 0 < account
xpub6BH1zcTuktiFu43rUZ2gXqLgzu5F3tLEeTQ5t6iE3aQtM2VMTxMcyLN9fYHiGhGpQe9QQYmqL2eYPFJ3vezHz
5wzaSW4FiGrseNDR4LKqTy

$ bx hd-public --index 1 < account
xpub6BH1zcTuktiFx6CzhPbGjG3UYQ13WR16CmtbPiagEKpEVtpyjshWyMaMV1cn7nUPUkgQHPVXJVqsrA8xWbGQD
hohEcDFTEYMvYzwRD7Juf8
```

Veřejné klíče mohou být vytvořeny z odpovídajících soukromých klíčů příkazem hd-to-public.

```
$ bx hd-private --index 0 < account | bx hd-to-public
xpub6BH1zcTuktiFu43rUZ2gXqLgzu5F3tLEeTQ5t6iE3aQtM2VMTxMcyLN9fYHiGhGpQe9QQYmqL2eYPFJ3vezHz
5wzaSW4FiGrseNDR4LKqTy

$ bx hd-private --index 1 < account | bx hd-to-public
xpub6BH1zcTuktiFx6CzhPbGjG3UYQ13WR16CmtbPiagEKpEVtpyjshWyMaMV1cn7nUPUkgQHPVXJVqsrA8xWbGQD
hohEcDFTEYMvYzwRD7Juf8
```

Můžeme vytvořit prakticky nekonečný počet klíčů tvořících deterministický řetěz odvozený z jednoho semínka. Tutu techniku využívá mnoho peněženkových aplikací pro vytváření klíčů, které mohou být zálohovány a obnoveny z jedné hodnoty semínka. Toto je jednodušší než zálohovat peněženku, která vytváří každý nový klíč zcela náhodně bez možnosti jeho odvození z předchozích klíčů.

Semínko může být zakódováno pomocí příkazu mnemonic-encode.

```
$ bx hd-mnemonic < seed > words
adore repeat vision worst especially veil inch woman cast recall dwell appreciate
```

Semínko může být dekódováno"Bitcoin Explorer","mnemonic-decode command") příkazem+mnemonic-decode+.

```
$ bx mnemonic-decode < words
eb68ee9f3df6bd4441a9feadec179ff1
```

Mnemotechnické zakódování semínka usnadňuje jeho zaznamenání a dokonce zapamatování.

Appendix A: Návrhy na vylepšení bitcoinu

Návrhy na vylepšení bitcoinu (v originále Bitcoin improvement proposals, zkratka BIP) jsou dokumenty, které poskytují informace bitcoinové komunitě nebo popisují návrhy nových vlastností pro bitcoin, jeho procesy nebo prostředí.

V BIP0001 *BIP Účel a obecné zásady* jsou definovány tři druhy BIP:

Standardní BIP

Popisuje změny, která ovlivní většinu nebo všechny bitcoinové implementace, jako je změna síťového protokolu, změna v pravidlech ověřování bloků nebo transakcí, nebo jiná změna nebo rozšíření, která postihuje vzájemnou spolupráci aplikací používajících bitcoin.

Informační BIP

Popisuje problematická místa v návrhu bitcoinu nebo poskytuje obecné pokyny nebo informace bitcoinové komunitě, ale nenavrhuje žádné nové změny. Informační BIP nemusejí nutně vyjadřovat shodu bitcoinové komunity nebo doporučení. Uživatelé a implementátoři mohou informační BIP ignorovat nebo se řídit jejich doporučením.

Procesní BIP

Popisují bitcoinový proces nebo navrhují změnu (nebo událost) v procesu. Procesní BIP jsou jako standardní BIp, ale aplikují se na jiné oblasti než je vlastní bitcoinový protokol. Mohou navrhovat implementaci, ale ne základních zdrojových kódů bitcoinu. Mohou často vyžadovat shodu komunity a narozdíl od informačních BIP jsou častěji více než doporučeními a uživatelé by je typicky neměli ignorovat. Příklady obsahují postupy, návody, změny rozhodovacího procesu, změny nástrojů nebo prostředí použitých při vývoji Bitcoinu. Jakékoliv meta-BIP je také požadováno za procesní BIP.

Návrhy na zlepšení bitcoinu jsou zaznamenávány ve verzovaném uložišti na GitHub. Přehled návrhů na vylepšení bitcoinu (BIP) ukazuje stavi BIP na podzim roku 2014. Podívejte se do zmíněného úložiště na aktuální přehled existujících BIP a jejich obsahů. Při tvorbě českého překladu byla provedena aktualizace jejich stavů k září 2016.

Table 1. Přehled návrhů na vylepšení bitcoinu (BIP)

BIP#	Odkaz	Název	Správce	Typ	Stav
1	https://github.com/bitcoin/bips/blob/master/bip-0001.mediawiki	BIP Účel a obecné zásady	Amir Taaki	Standardní	Aktivní

BIP#	Odkaz	Název	Správce	Typ	Stav
10	https://github.com/bitcoin/bips/blob/master/bip-0010.mediawiki	Distribuce vícepodpisových transakcí	Alan Reiner	informační	Stažený
11	https://github.com/bitcoin/bips/blob/master/bip-0011.mediawiki	Standard vícepodpisových transakcí	Gavin Andresen	Standardní	Dokončený
12	https://github.com/bitcoin/bips/blob/master/bip-0012.mediawiki	OP_EVAL	Gavin Andresen	Standardní	Stažemý
13	https://github.com/bitcoin/bips/blob/master/bip-0013.mediawiki	Formát adres pro pay-to-script-hash	Gavin Andresen	Standardní	Dokončený
14	https://github.com/bitcoin/bips/blob/master/bip-0014.mediawiki	Verze protokolu a uživatelská agenda	Amir Taaki, Patrick Strateman	Standardní	Dokončený
15	https://github.com/bitcoin/bips/blob/master/bip-0015.mediawiki	Přezdívky	Amir Taaki	Standardní	Odložený
16	https://github.com/bitcoin/bips/blob/master/bip-0016.mediawiki	Pay To Script Hash	Gavin Andresen	Standardní	Přijatý

BIP#	Odkaz	Název	Správce	Typ	Stav
17	https://github.com/bitcoin/bips/blob/master/bip-0017.mediawiki	OP_CHECKHASHVERIFY (CHV)	Luke Dashjr	Standardní	Stažený
18	https://github.com/bitcoin/bips/blob/master/bip-0018.mediawikilink:	hashScriptCheck	Luke Dashjr	Standardní	Přijatý
19	https://github.com/bitcoin/bips/blob/master/bip-0019.mediawiki	Standard odlehčených vícepodpisových transakcí	Luke Dashjr	Standardní	Návrh
20	https://github.com/bitcoin/bips/blob/master/bip-0020.mediawiki	URI Scháma	Luke Dashjr	Standardní	Nahrazený
21	https://github.com/bitcoin/bips/blob/master/bip-0021.mediawiki	URI Schéma	Nils Schneider, Matt Corallo	Standardní	Přijatý
22	https://github.com/bitcoin/bips/blob/master/bip-0022.mediawiki	getblocktemplate - základy	Luke Dashjr	Standardní	Dokončený
23	https://github.com/bitcoin/bips/blob/master/bip-0023.mediawiki	getblocktemplate - sdílená těžba	Luke Dashjr	Standardní	Dokončený

BIP#	Odkaz	Název	Správce	Typ	Stav
30	https://github.com/bitcoin/bips/blob/master/bip-0030.mediawiki	Zdvojené transakce	Pieter Wuille	Standardní	Přijatý
31	https://github.com/bitcoin/bips/blob/master/bip-0031.mediawiki	Pong zpráva	Mike Hearn	Standardní	Dokončený
32	https://github.com/bitcoin/bips/blob/master/bip-0032.mediawiki	Hierarchické deterministické peněženky	Pieter Wuille	Informační	Dokončený
33	https://github.com/bitcoin/bips/blob/master/bip-0033.mediawiki	Specializované uzly	Amir Taaki	Standardní	Přijatý
34	https://github.com/bitcoin/bips/blob/master/bip-0034.mediawiki	Block v2, výška v mincetvorné transakci	Gavin Andresen	Standardní	Dokončený
35	https://github.com/bitcoin/bips/blob/master/bip-0035.mediawiki	Úložiště zpráv	Jeff Garzik	Standardní	Dokončený
36	https://github.com/bitcoin/bips/blob/master/bip-0036.mediawiki	Zákaznické služby	Stefan Thomas	Standardní	Přijatý

BIP#	Odkaz	Název	Správce	Typ	Stav
37	https://github.com/bitcoin/bips/blob/master/bip-0037.mediawiki	Bloomovo filtrování	Mike Hearn and Matt Corallo	Standardní	Dokončený
38	https://github.com/bitcoin/bips/blob/master/bip-0038.mediawiki	Ochrana soukromého klíče heslovou frází	Mike Caldwell	Standardní	Návrh
39	https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki	Mnemotechnické kódy pro deterministické generování klíčů	Slush	Standardní	Přijatý
40		Stratum drátový protokoll	Slush	Standard	Přiděleno číslo BIP
41		Stratum těžební protokol	Slush	Standard	Přiděleno číslo BIP
42	https://github.com/bitcoin/bips/blob/master/bip-0042.mediawiki	Konečná peněžní zásoba pro bitcoin	Pieter Wuille	Standardní	Návrh
43	https://github.com/bitcoin/bips/blob/master/bip-0043.mediawiki	Důvodové pole pro deterministické peněženky	Slush	Standardni	Návrh
44	https://github.com/bitcoin/bips/blob/master/bip-0044.mediawiki	Víceúčtová hierarchie pro deterministické peněženky	Slush	Standardní	Přijatý

BIP#	Odkaz	Název	Správce	Typ	Stav
50	https://github.com/bitcoin/bips/blob/master/bip-0050.mediawiki	Rozdvojení řetězu v březnu 2013	Gavin Andresen	Informationační	Dokončený
60	https://github.com/bitcoin/bips/blob/master/bip-0060.mediawiki	Oprava délky zprávy (Relay-Transactions Field)	Amir Taaki	Standardní	Návrh
61	https://github.com/bitcoin/bips/blob/master/bip-0061.mediawiki	Zprává vysvětlující zamítnutí	Gavin Andresen	Standardní	Dokončený
62	https://github.com/bitcoin/bips/blob/master/bip-0062.mediawiki	Vypořadání se s ohebností transakcí	Pieter Wuille	Standardní	Dokončený
63		Ukrývání adres	Peter Todd	Standardní	Přiděleno číslo BIP
64	https://github.com/bitcoin/bips/blob/master/bip-0064.mediawiki	getutxos zpráva	Mike Hearn	Standardní	Přijatý
70	https://github.com/bitcoin/bips/blob/master/bip-0070.mediawiki	Platební protokol	Gavin Andresen	Standardní	Přijatý

BIP#	Odkaz	Název	Správce	Typ	Stav
71	https://github.com/bitcoin/bips/blob/master/bip-0071.mediawiki	Platební protokol typ MIME	Gavin Andresen	Standardní	Dokončený
72	https://github.com/bitcoin/bips/blob/master/bip-0072.mediawiki	Platební protokol URI	Gavin Andresen	Standardní	Dokončený
73	https://github.com/bitcoin/bips/blob/master/bip-0073.mediawiki	Použití "Accept" hlaviček v platebním požadavku URLs	Stephen Pair	Standardní	Dokončený

Appendix A: pycoin, ku, a tx

Knihovna pycoin, která byla původně naprogramována v jazyce Python a udržována Richardem Kissem, slouží ke správě bitcoinových klíčů a zpracování transakcí, dokonce podporuje skriptovací jazyk, který je dostatečně silný na správné zpracování nestandardních transakcí.

Knihovna pycoin podporuje jak Python 2 (2.7.x), tak i Python 3 (od 3.3) a přináší nástroje ovládané z příkazové řádky ku a tx.

Správa klíčů (ku)

Nástroj ovládaný z příkazové řádky ku ("key utility") je švýcarským armádním nožem pro správu klíčů. Podporuje klíče dle normy BIP32 a formátu WIF a adresy bitcoinu a odvozených měn. Následuje několik příkladů.

Vytvoří klíč dle normy BIP32 s použitím zdrojů náhodných čísel GPG a */dev/random*:

```
$ ku create

input : create
network : Bitcoin
wallet key : xprv9s21ZrQH143K3LU5ctPZTBnb9kTjA5Su9DcWHvXJemiJBsY7VqXUG7hipgdWaU
                   m2nhnzdvxJf5KJo9vjP2nABX65c5sFsWsV8oXcbpehtJi
public version : xpub661MyMwAqRbcFpYYiuvZpKjKhnJDZYAkWSY76JvvD7FH4fsG3Nqiov2CfxzxY8
                   DGcpfT56AMFeo8M8KPkFMfLUtvwjwb6WPv8rY65L2q8Hz
tree depth : 0
fingerprint : 9d9c6092
parent f'print : 00000000
child index : 0
chain code : 80574fb260edaa4905bc86c9a47d30c697c50047ed466c0d4a5167f6821e8f3c
private key : yes
secret exponent :
112471538590155650688604752840386134637231974546906847202389294096567806844862
 hex : f8a8a28b28a916e1043cc0aca52033a18a13cab1638d544006469bc171fddfbe
wif : L5Z54xi6qJusQT42JHA44mfPVZGjyb4XBRWfxAzUWwRiGx1kV4sP
 uncompressed : 5KhoEavGNNH4GHKoy2Ptu4KfdNp4r56L5B5un8FP6RZnbsz5Nmb
public pair x :
764606382405464783648433974782784681018771177678734621270215603682901140160 34
public pair y :
598078796574697741020401202982722077309212917366332477370774067536768257777 01
 x as hex : a90b3008792432060fa04365941e09a8e4adf928bdbdb9dad41131274e379322
 y as hex : 843a0f6ed9c0eb1962c74533795406914fe3f1957c5238951f4fe245a4fcd625
y parity : odd
key pair as sec : 03a90b3008792432060fa04365941e09a8e4adf928bdbdb9dad41131274e379322
 uncompressed : 04a90b3008792432060fa04365941e09a8e4adf928bdbdb9dad41131274e379322
                   843a0f6ed9c0eb1962c74533795406914fe3f1957c5238951f4fe245a4fcd625
hash160 : 9d9c609247174ae323acfc96c852753fe3c8819d
 uncompressed : 8870d869800c9b91ce1eb460f4c60540f87c15d7
Bitcoin address : 1FNNRQ5fSv1wBi5gyfVBs2rkNheMGt86sp
 uncompressed : 1DSS5isnH4FsVaLVjeVXewVSpfqktdiQAM
```

Vytvoří BIP32 klíč z heslové fráze.

WARNING Heslová fráze je v tomto příkladě příliš jednoduchá na uhodnutí

```
$ ku P:foo

input : P:foo
network : Bitcoin
wallet key : xprv9s21ZrQH143K31AgNK5pyVvW23gHnkBq2wh5aEk6g1s496M8ZMjxncCKZKgb5j
                    ZoY5eSJMJ2Vbyvi2hbmQnCuHBujZ2WXGTux1X2k9Krdtq
public version : xpub661MyMwAqRbcFVF9ULcqLdsEa5WnCCugQAcgNd9iEMQ31tgH6u4DLQWoQayvtS
                    VYFvXz2vPPpbXE1qpjoUFidhjFj82pVShWu9curWmb2zy
tree depth : 0
fingerprint : 5d353a2e
parent f'print : 00000000
child index : 0
chain code : 5eeb1023fd6dd1ae52a005ce0e73420821e1d90e08be980a85e9111fd7646bbc
private key : yes
secret exponent :
65825730547097305716057160437970790220123864299761908948746835886007793998275
 hex : 91880b0e3017ba586b735fe7d04f1790f3c46b818a2151fb2def5f14dd2fd9c3
wif : L26c3H6jEPVSqAr1usXUp9qtQJw6NHgApq6Ls4ncyqtsvcq2MwKH
 uncompressed : 5JvNzA5vXDoKYJdw8SwwLHxUxaWvn9mDea6k1vRPCX7KLUVWa7W
public pair x :
81821982719381104061777349269130419024493616650993589394553404347774393168191
public pair y :
58994218069605424278320703250689780154785099509277691723126325051200459038290
 x as hex : b4e599dfa44555a4ed38bcfff0071d5af676a86abf123c5b4b4e8e67a0b0b13f
 y as hex : 826d8b4d3010aea16ff4c1c1d3ae68541d9a04df54a2c48cc241c2983544de52
y parity : even
key pair as sec : 02b4e599dfa44555a4ed38bcfff0071d5af676a86abf123c5b4b4e8e67a0b0b13f
 uncompressed : 04b4e599dfa44555a4ed38bcfff0071d5af676a86abf123c5b4b4e8e67a0b0b13f
                    826d8b4d3010aea16ff4c1c1d3ae68541d9a04df54a2c48cc241c2983544de52
hash160 : 5d353a2ecdb262477172852d57a3f11de0c19286
 uncompressed : e5bd3a7e6cb62b4c820e51200fb1c148d79e67da
Bitcoin address : 19Vqc8uLTfUonmxUEZac7fz1M5c5ZZbAii
 uncompressed : 1MwkRkogzBRMehBntgcq2aJhXCXStJTXHT
```

Zobrazení informací ve formátu JSON:

```
$ ku P:foo -P -j
```

```
{
   "y_parity": "even",
   "public_pair_y_hex":
"826d8b4d3010aea16ff4c1c1d3ae68541d9a04df54a2c48cc241c2983544de52",
   "private_key": "no",
   "parent_fingerprint": "00000000",
   "tree_depth": "0",
   "network": "Bitcoin",
   "btc_address_uncompressed": "1MwkRkogzBRMehBntgcq2aJhXCXStJTXHT",
   "key_pair_as_sec_uncompressed":
"04b4e599dfa44555a4ed38bcfff0071d5af676a86abf123c5b4b4e8e67a0b0b13f826d8b4d3010aea16ff4c1
c1d3ae68541d9a04df54a2c48cc241c2983544de52",
   "public_pair_x_hex":
"b4e599dfa44555a4ed38bcfff0071d5af676a86abf123c5b4b4e8e67a0b0b13f",
   "wallet_key":
"xpub661MyMwAqRbcFVF9ULcqLdsEa5WnCCugQAcgNd9iEMQ31tgH6u4DLQWoQayvtSVYFvXz2vPPpbXE1qpjoUFi
dhjFj82pVShWu9curWmb2zy",
   "chain_code": "5eeb1023fd6dd1ae52a005ce0e73420821e1d90e08be980a85e9111fd7646bbc",
   "child_index": "0",
   "hash160_uncompressed": "e5bd3a7e6cb62b4c820e51200fb1c148d79e67da",
   "btc_address": "19Vqc8uLTfUonmxUEZac7fz1M5c5ZZbAii",
   "fingerprint": "5d353a2e",
   "hash160": "5d353a2ecdb262477172852d57a3f11de0c19286",
   "input": "P:foo",
   "public_pair_x":
"818219827193811040617773492691304190244936166509935893945534043477743931681 91",
   "public_pair_y":
"589942180696054242783207032506897801547850995092776917231263250512004590382 90",
   "key_pair_as_sec":
"02b4e599dfa44555a4ed38bcfff0071d5af676a86abf123c5b4b4e8e67a0b0b13f"
}
```


Veřejný klíč ve formátu BIP32

```
$ ku -w -P P:foo
xpub661MyMwAqRbcFVF9ULcqLdsEa5WnCCugQAcgNd9iEMQ31tgH6u4DLQWoQayvtSVYFvXz2vPPpbXE1qpjoUFid
hjFj82pVShWu9curWmb2zy
```

Vytvoření odvozeného klíče

```
$ ku -w -s3/2 P:foo
xprv9wTErTSkjVyJa1v4cUTFMFkWMe5eu8ErbQcs9xajnsUzCBT7ykHAwdrxvG3g3f6BFk7ms5hHBvmbdutNmyg6i
ogWKxx6mefEw4M8EroLgKj
```

Ztížený podklíč

```
$ ku -w -s3/2H P:foo
xprv9wTErTSu5AWGkDeUPmqBcbZWX1xq85ZNX9iQRQW9DXwygFp7iRGJo79dsVctcsCHsnZ3XU3DhsuaGZbDh8iDk
BN45k67UKsJUXM1JfRCdn1
```

WIF:

```
$ ku -W P:foo
L26c3H6jEPVSqAr1usXUp9qtQJw6NHgApq6Ls4ncyqtsvcq2MwKH
```

Adresa:

```
$ ku -a P:foo
19Vqc8uLTfUonmxUEZac7fz1M5c5ZZbAii
```

Vytvoření skupiny odvozených klíčů:

```
$ ku P:foo -s 0/0-5 -w
xprv9xWkBDfyBXmZjBG9EiXBpy67KK72fphUp9utJokEBFtjsjiuKUUDF5V3TU8U8cDzytqYnSekc8bYuJS8G3bhX
xKWB89Ggn2dzLcoJsuEdRK
xprv9xWkBDfyBXmZnzKf3bAGifK593gT7WJZPnYAmvc77gUQVej5QHckc5Adtwxa28ACmANi9XhCrRvtFqQcUxt8r
UgFz3souMiDdWxJDZnQxzx
xprv9xWkBDfyBXmZqdXA8y4SWqfBdy71gSW9sjx9JpCiJEiBwSMQyRxan6srXUPBtj3PTxQFkZJAiwoUpmvtrxKZu
4zfsnr3pqyy2vthpkwuoVq
xprv9xWkBDfyBXmZsA85GyWj9uYPyoQv826YAadKWMaaEosNrFBKgj2TqWuiWY3zuqxYGpHfv9cnGj5P7e8EskpzK
L1Y8Gk9aX6QbryA5raK73p
xprv9xWkBDfyBXmZv2q3N66hhZ8DAcEnQDnXML1J62krJAcf7Xb1HJwuW2VMJQrCofY2jtFXdiEY8UsRNJfqK6DAd
yZXoMvtaLHyWQx3FS4A9zw
xprv9xWkBDfyBXmZw4jEYXUHYc9fT25k9irP87n2RqfJ5bqbjKdT84Mm7Wtc2xmzFuKg7iYf7XFHKkSsaYKWKJbR5
4bnyAD9GzjUYbAYTtN4ruo
```

Vytvoření odpovídajících adres:

```
$ ku P:foo -s 0/0-5 -a
1MrjE78H1R1rqdFrmkjdHnPUdLCJALbv3x
1AnYyVEcuqeoVzH96zj1eYKwoWfwte2pxu
1GXr1kZfxE1FcK6ZRD5sqqqs5YfvuzA1Lb
116AXZc4bDVQrqmcinzu4aaPdrYqvuiBEK
1Cz2rTLjRM6pMnxPNrRKp9ZSvRtj5dDUML
1WstdwPnU6HEUPme1DQayN9nm6j7nDVEM
```

Vytvoření odpovídajících WIF

```
$ ku P:foo -s 0/0-5 -W
L5a4iE5k9gcJKGqX3FWmxzBYQc29PvZ6pgBaePLVqT5YByEnBomx
Kyjgne6GZwPGB6G6kJEhoPbmyjMP7D5d3zRbHVjwcq4iQXD9QqKQ
L4B3ygQxK6zH2NQGxLDee2H9v4Lvwg14cLJW7QwWPzCtKHdWMaQz
L2L2PZdorybUqkPjrmhem4Ax5EJvP7ijmxbNoQKnmTDMrqemY8UF
L2oD6vA4TUyqPF8QG4vhUFSgwCyuuvFZ3v8SKHYFDwkbM765Nrfd
KzChTbc3kZFxUSJ3Kt54cxsogeFAD9CCM4zGB22si8nfKcThQn8C
```

Ověření funkčnosti výběrem BIP32 řetězce odpovídajícího odvozenému klíči 0/3

```
$ ku -W
xprv9xWkBDfyBXmZsA85GyWj9uYPyoQv826YAadKWMaaEosNrFBKgj2TqWuiWY3zuqxYGpHfv9cnGj5P7e8EskpzK
L1Y8Gk9aX6QbryA5raK73p
L2L2PZdorybUqkPjrmhem4Ax5EJvP7ijmxbNoQKnmTDMrqemY8UF
$ ku -a
xprv9xWkBDfyBXmZsA85GyWj9uYPyoQv826YAadKWMaaEosNrFBKgj2TqWuiWY3zuqxYGpHfv9cnGj5P7e8EskpzK
L1Y8Gk9aX6QbryA5raK73p
116AXZc4bDVQrqmcinzu4aaPdrYqvuiBEK
```

Ano, vypadá povědomě.

Ze soukromého exponentu

```
$ ku 1

input : 1
network : Bitcoin
secret exponent : 1
 hex : 1
wif : KwDiBf89QgGbjEhKnhXJuH7LrciVrZi3qYjgd9M7rFU73sVHnoWn
 uncompressed : 5HpHagT65TZzG1PH3CSu63k8DbpvD8s5ip4nEB3kEsreAnchuDf
public pair x :
55066263022277343669578718895168534326250603453777594175500187360389116729240
public pair y :
32670510020758816978083085130507043184471273380659243275938904335757337482424
 x as hex : 79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
 y as hex : 483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8
y parity : even
key pair as sec : 0279be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
 uncompressed : 0479be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
                    483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8
hash160 : 751e76e8199196d454941c45d1b3a323f1433bd6
 uncompressed : 91b24bf9f5288532960ac687abb035127b1d28a5
Bitcoin address : 1BgGZ9tcN4rm9KBzDn7KprQz87SZ26SAMH
 uncompressed : 1EHNa6Q4Jz2uvNExL497mE43ikXhwF6kZm
```

Verze pro Litecoin

```
$ ku -nL 1

input : 1
network : Litecoin
secret exponent : 1
 hex : 1
wif : T33ydQRKp4FCW5LCLLUB7deioUMoveiwekdwUwyfRDeGZm76aUjV
 uncompressed : 6u823ozcyt2rjPH8Z2ErsSXJB5PPQwK7VVTwwN4mxLBFrao69XQ
public pair x :
55066263022277343669578718895168534326250603453777594175500187360389116729240
public pair y :
32670510020758816978083085130507043184471273380659243275938904335757337482424
 x as hex : 79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
 y as hex : 483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8
y parity : even
key pair as sec : 0279be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
 uncompressed : 0479be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
                     483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8
hash160 : 751e76e8199196d454941c45d1b3a323f1433bd6
 uncompressed : 91b24bf9f5288532960ac687abb035127b1d28a5
Litecoin address : LVuDpNCSSj6pQ7t9Pv6d6sUkLKoqDEVUnJ
 uncompressed : LYWKqJhtPeGyBAw7WC8R3F7ovxtzAiubdM
```

Dogecoin WIF:

```
$ ku -nD -W 1
QNcdLVw8fHkixm6NNyN6nVwxKek4u7qrioRbQmjxac5TVoTtZuot
```

Z veřejného páru (na Tesnetu)

```
$ ku -nT
55066263022277343669578718895168534326250603453777594175500187360389116729240,even

input : 55066263022277343669578718895168534326250603453777594175500187360
                             389116729240,even
network : Bitcoin testnet
public pair x :
55066263022277343669578718895168534326250603453777594175500187360389116729240
public pair y :
32670510020758816978083085130507043184471273380659243275938904335757337482424
 x as hex : 79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
 y as hex : 483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8
y parity : even
key pair as sec : 0279be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
 uncompressed : 0479be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798

483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8
hash160 : 751e76e8199196d454941c45d1b3a323f1433bd6
 uncompressed : 91b24bf9f5288532960ac687abb035127b1d28a5
Bitcoin testnet address : mrCDrCybB6J1vRfbwM5hemdJz73FwDBC8r
 uncompressed : mtoKs9V381UAhUia3d7Vb9GNak8Qvmcsme
```

Z hash160:

```
$ ku 751e76e8199196d454941c45d1b3a323f1433bd6

input : 751e76e8199196d454941c45d1b3a323f1433bd6
network : Bitcoin
hash160 : 751e76e8199196d454941c45d1b3a323f1433bd6
Bitcoin address : 1BgGZ9tcN4rm9KBzDn7KprQz87SZ26SAMH
```

Jako Dogecoin adresa:

```
$ ku -nD 751e76e8199196d454941c45d1b3a323f1433bd6

input : 751e76e8199196d454941c45d1b3a323f1433bd6
network : Dogecoin
hash160 : 751e76e8199196d454941c45d1b3a323f1433bd6
Dogecoin address : DFpN6QqFfUm3gKNaxN6tNcab1FArL9cZLE
```

Nástroj pro transakce (tx)

Nástroj ovládaný z příkazové řádky tx zobrazuje transakce v lidsky čitelné formě, vyzvedává transakce transakční mezipaměti pycoinu nebo z webových služeb (aktuálně jsou podporovány blockchain.info, blockr.io, a biteasy.com), slévá transakce, přidává nebo maže vstupy nebo výstupy transakcí a podepisuje transakce.

Následuje několik příkladů.

Prohlédneme si slavnou transakci "s pizzou" [PIZZA]:

```
$ tx 49d2adb6e476fa46d8357babf78b1b501fd39e177ac7833124b3f67b17c40c2a
warning: consider setting environment variable PYCOIN_CACHE_DIR=~/.pycoin_cache to
cache transactions fetched via web services
warning: no service providers found for get_tx; consider setting environment variable
PYCOIN_SERVICE_PROVIDERS=BLOCKR_IO:BLOCKCHAIN_INFO:BITEASY:BLOCKEXPLORER
usage: tx [-h] [-t TRANSACTION_VERSION] [-l LOCK_TIME] [-n NETWORK] [-a]
          [-i address] [-f path-to-private-keys] [-g GPG_ARGUMENT]
          [--remove-tx-in tx_in_index_to_delete]
          [--remove-tx-out tx_out_index_to_delete] [-F transaction-fee] [-u]
          [-b BITCOIND_URL] [-o path-to-output-file]
          argument [argument ...]
tx: error: can't find Tx with id
49d2adb6e476fa46d8357babf78b1b501fd39e177ac7833124b3f67b17c40c2a
```

Jejda! Nemáme nastavenou webovou službu, uděláme to nyní:

```
$ PYCOIN_CACHE_DIR=~/.pycoin_cache
$ PYCOIN_SERVICE_PROVIDERS=BLOCKR_IO:BLOCKCHAIN_INFO:BITEASY:BLOCKEXPLORER
$ export PYCOIN_CACHE_DIR PYCOIN_SERVICE_PROVIDERS
```

Toto nastavení není uděláno automaticky z důvodu, aby nástroj tx potenciálně neprozradil webové stránce spravované třetí stranou naše soukromé informace o tom, o jaké transakce se zajímáme. Pokud nám toto prozrazení nevadí, můžeme přidat tyto řádky do našeho *.profile*.

Zkuste to znova:

```
$ tx 49d2adb6e476fa46d8357babf78b1b501fd39e177ac7833124b3f67b17c40c2a
Version: 1 tx hash 49d2adb6e476fa46d8357babf78b1b501fd39e177ac7833124b3f67b17c40c2a
159 bytes
TxIn count: 1; TxOut count: 1
Lock time: 0 (valid anytime)
Input:
  0: (unknown) from
1e133f7de73ac7d074e2746a3d6717dfc99ecaa8e9f9fade2cb8b0b20a5e0441:0
Output:
  0: 1CZDM6oTttND6WPdt3D6bydo7DYKzd9Qik receives 10000000.00000 mBTC
Total output 10000000.00000 mBTC
including unspents in hex dump since transaction not fully signed
010000000141045e0ab2b0b82cdefaf9e9a8ca9ec9df17673d6a74e274d0c73ae77d3f131e000000004a4
93046022100a7f26eda874931999c90f87f01ff1ffc76bcd058fe16137e0e63fdb6a35c2d78022100a61e
9199238eb73f07c8f209504c84b80f03e30ed8169edd44f80ed17ddf451901ffffffff010010a5d4e8000
0001976a9147ec1003336542cae8bded8909cdd6b5e48ba0ab688ac00000000

** can't validate transaction as source transactions missing
```

Poslední řádka se objevila, protože k ověření podpisu transakce potřebujeme zdroj transakcí. Pomocí -a rozšíříme transakci o zdroj informací.

```
$ tx -a 49d2adb6e476fa46d8357babf78b1b501fd39e177ac7833124b3f67b17c40c2a
warning: transaction fees recommendations casually calculated and estimates may be
incorrect
warning: transaction fee lower than (casually calculated) expected value of 0.1 mBTC,
transaction might not propogate
Version: 1 tx hash 49d2adb6e476fa46d8357babf78b1b501fd39e177ac7833124b3f67b17c40c2a
159 bytes
TxIn count: 1; TxOut count: 1
Lock time: 0 (valid anytime)
Input:
  0: 17WFx2GQZUmh6Up2NDNCEDk3deYomdNCfk from
1e133f7de73ac7d074e2746a3d6717dfc99ecaa8e9f9fade2cb8b0b20a5e0441:0 10000000.00000
mBTC sig ok
Output:
  0: 1CZDM6oTttND6WPdt3D6bydo7DYKzd9Qik receives 10000000.00000 mBTC
Total input 10000000.00000 mBTC
Total output 10000000.00000 mBTC
Total fees 0.00000 mBTC

010000000141045e0ab2b0b82cdefaf9e9a8ca9ec9df17673d6a74e274d0c73ae77d3f131e000000004a4
93046022100a7f26eda874931999c90f87f01ff1ffc76bcd058fe16137e0e63fdb6a35c2d78022100a61e
9199238eb73f07c8f209504c84b80f03e30ed8169edd44f80ed17ddf451901ffffffff010010a5d4e8000
0001976a9147ec1003336542cae8bded8909cdd6b5e48ba0ab688ac00000000

all incoming transaction values validated
```

Nyní se podíváme na neutracené výstupy pro specifickou adresu (UTXO). V bloku číslo 1 vidíme mincetvornou transakci ve prospěch adresy 12c6DSiU4Rq3P4ZxziKxzrL5LmMBrzjrJX. Použijeme fetch_unspent k najití všech mincí na této adrese:

```
$ fetch_unspent 12c6DSiU4Rq3P4ZxziKxzrL5LmMBrzjrJX
a3a6f902a51a2cbebede144e48a88c05e608c2cce28024041a5b9874013a1e2a/0/76a914119b098e2e980a229e139a9ed01a469e518e6f2688ac/333000
cea36d008badf5c7866894b191d3239de9582d89b6b452b596f1f1b76347f8cb/31/76a914119b098e2e980a229e139a9ed01a469e518e6f2688ac/10000
065ef6b1463f552f675622a5d1fd2c08d6324b4402049f68e767a719e2049e8d/86/76a914119b098e2e980a229e139a9ed01a469e518e6f2688ac/10000
a66dddd42f9f2491d3c336ce5527d45cc5c2163aaed3158f81dc054447f447a2/0/76a914119b098e2e980a229e139a9ed01a469e518e6f2688ac/10000
ffd901679de65d4398de90cefe68d2c3ef073c41f7e8dbec2fb5cd75fe71dfe7/0/76a914119b098e2e980a229e139a9ed01a469e518e6f2688ac/100
d658ab87cc053b8dbcfd4aa2717fd23cc3edfe90ec75351fadd6a0f7993b461d/5/76a914119b098e2e980a229e139a9ed01a469e518e6f2688ac/911
36ebe0ca3237002acb12e1474a3859bde0ac84b419ec4ae373e63363ebef731c/1/76a914119b098e2e980a229e139a9ed01a469e518e6f2688ac/100000
fd87f9adebb17f4ebb1673da76ff48ad29e64b7afa02fda0f2c14e43d220fe24/0/76a914119b098e2e980a229e139a9ed01a469e518e6f2688ac/1
dfdf0b375a987f17056e5e919ee6eadd87dad36c09c4016d4a03cea15e5c05e3/1/76a914119b098e2e980a229e139a9ed01a469e518e6f2688ac/1337
cb2679bfd0a557b2dc0d8a6116822f3fcbe281ca3f3e18d3855aa7ea378fa373/0/76a914119b098e2e980a229e139a9ed01a469e518e6f2688ac/1337
d6be34ccf6edddc3cf69842dce99fe503bf632ba2c2adb0f95c63f6706ae0c52/1/76a914119b098e2e980a229e139a9ed01a469e518e6f2688ac/2000000

0e3e2357e806b6cdb1f70b54c3a3a17b6714ee1f0e68bebb44a74b1efd512098/0/410496b538e853519c726a2c91e61ec11600ae1390813a627c66fb8be7947be63c52da7589379515d4e0a604f8141781e62294721166bf621e73a82cbf2342c858eeac/5000000000
```

Appendix A: Operátory, konstanty a symboly transakčního skryptovacího jazyka

Vkládání hodnot na zásobník zobrazuje operátory pro vkládání hodnot na zásobník.

Table 1. Vkládání hodnot na zásobník

Symbol	Hodnota (hex)	Popis
OP_0 nebo OP_FALSE	0x00	Na zásobník je vloženo prázdné pole
1-75	0x01-0x4b	Na zásobník je vloženo N bytů, kde N je od 1 do 75
OP_PUSHDATA1	0x4c	Další byte skryptu obsahuje N. Na zásobník je vloženo následujících N bytů.
OP_PUSHDATA2	0x4d	Následující dva byty skriptu obsahují N. Na zásobník je vloženo následujících N bytů.
OP_PUSHDATA4	0x4e	Následující čtyři byty skriptu obsahují N. Na zásobník je vloženo následujících N bytů.
OP_1NEGATE	0x4f	Na zásobník je vložena hodnota "–1".
OP_RESERVED	0x50	Zastaví výpočet - Neplatná transakce, pokud se nenachází v nevykonané klauzuli OP_IF.
OP_1 or OP_TRUE	0x51	Na zásobník je vložena hodnota "1"
OP_2 to OP_16	0x52 to 0x60	Pro OP_N je na zásobník vložena hodnota N. Například OP_2 vloží "2".

Podmíněné řízení toku zobrazuje podmíněné operátory řízení toku.

Table 2. Podmíněné řízení toku

Symbol	Hodnota (hex)	Popis
OP_NOP	0x61	Nedělá nic

Symbol	Hodnota (hex)	Popis
OP_VER	0x62	Zastaví výpočet - Neplatná transakce, pokud se nenachází v nevykonané klauzuli OP_IF.
OP_IF	0x63	Pokud není na vrcholu zásobníku 0 vykoná následující příkazy
OP_NOTIF	0x64	Pokud je na vrcholu zásobníku 0 vykoná následující příkazy
OP_VERIF	0x65	Zastaví výpočet - Neplatná transakce (i když se nachází v nevykonané klauzuli OP_IF)
OP_VERNOTIF	0x66	Zastaví výpočet - Neplatná transakce (i když se nachází v nevykonané klauzuli OP_IF)
OP_ELSE	0x67	Vykoná se pouze pokud předchozí příkaz nebyl vykonán
OP_ENDIF	0x68	Ukončuje blok OP_IF, OP_NOTIF, OP_ELSE
OP_VERIFY	0x69	Zkontroluje vrchol zásobníku, pokud na něm není TRUE, výpočet se zastaví, transakce je neplatná.
OP_RETURN	0x6a	Zastaví výpočet - Neplatná transakce

Zásobníkové operace zobrazuje operátory použité pro práci se zásobníkem.

Table 3. Zásobníkové operace

Symbol	Hodnota (hex)	Popis
OP_TOALTSTACK	0x6b	Odebere položku zvrcholu z zásobníku a vloží ji do alternativního zásobníku
OP_FROMALTSTACK	0x6c	Odebere položku z vrcholu alternativního zásobníku a vloží ji do zásobníku
OP_2DROP	0x6d	Odebere z vrcholu zásobníku dvě položky.

Symbol	Hodnota (hex)	Popis
OP_2DUP	0x6e	Zdvojí dvě položky na vrcholu zásobníku
OP_3DUP	0x6f	Zdvojí tři položky na vrcholu zásobníku
OP_2OVER	0x70	Okopíruje třetí a čtvrtou položku v zásobníku na jeho vrchol
OP_2ROT	0x71	Okopíruje pátou a šestou položku v zásobníku na jeho vrchol
OP_2SWAP	0x72	Prohodí dvě položky na vrcholu zásobníku
OP_IFDUP	0x73	Zdvojí položku na vrcholu zásobníku, pokud není 0
OP_DEPTH	0x74	Na vrchol zásobníku vloží počet položek v zásobníku
OP_DROP	0x75	Odebere položku z vrcholu zásobníku
OP_DUP	0x76	Zdvojí položku na vrcholu zásobníku
OP_NIP	0x77	Odebere druhou položku z vrcholu zásobníku
OP_OVER	0x78	Okopíruje druhou položku z vrcholu zásobníku a vloží ji na vrchol
OP_PICK	0x79	Odebere hodnotu N z vrcholu zásobníku, poté okopíruje N-tou položku od vrcholu zásobníku na jeho vrchol
OP_ROLL	0x7a	Odebere hodnotu N z vrcholu zásobníku, poté přesune N-tou položku od vrcholu zásobníku na jeho vrchol
OP_ROT	0x7b	Provede rotaci tří prvků na vrcholu zásobníku (123 na 231)
OP_SWAP	0x7c	Prohodí dva prvky na vrcholu zásobníku

Symbol	Hodnota (hex)	Popis
OP_TUCK	0x7d	Okopíruje položku z vrcholu zásobníku a vloží ji mezi vrchol a druhou položku.

Řetězcové spojovací operace zobrazuje řetězcové operátory.

Table 4. Řetězcové spojovací operace

Symbol	Hodnota (hex)	Popis
OP_CAT	0x7e	Neplatné (spojí dvě položky z vrcholu zásobníku)
OP_SUBSTR	0x7f	Neplatné (vrátí podřetězec)
OP_LEFT	0x80	Neplatné (vrátí levý podřetězec)
OP_RIGHT	0x81	Neplatné (vrátí pravý podřetězec)
OP_SIZE	0x82	Spočte délku řetězce na vrcholu zásobníku a výsledek vloží na vrchol

Binární aritmetika a podmínky zobrazuje binární aritmetické a logické operátory.

Table 5. Binární aritmetika a podmínky

Symbol	Hodnota (hex)	Popis
OP_INVERT	0x83	Neplatné (neguje bity v položce na vrcholu zásobníku)
OP_AND	0x84	Neplatné (bitový AND vrchních dvou položek zásobníku)
OP_OR	0x85	Neplatné (bitový OR vrchních dvou položek zásobníku)
OP_XOR	0x86	Neplatné (bitový XOR vrchních dvou položek zásobníku)
OP_EQUAL	0x87	Pokud se vrchní dvě položky na zásobníku rovnají vloží na vrchol zásobníku TRUE (1), jinak FALSE (0)
OP_EQUALVERIFY	0x88	Stejné jako OP_EQUAL, ale navíc spustí OP_VERIFY

Symbol	Hodnota (hex)	Popis
OP_RESERVED1	0x89	Zastaví výpočet - Neplatná transakce, pokud se nenachází v nevykonané klauzuli OP_IF.
OP_RESERVED2	0x8a	Zastaví výpočet - Neplatná transakce, pokud se nenachází v nevykonané klauzuli OP_IF.

Aritmetické operátory zobrazuje aritmetické operátory.

Table 6. Aritmetické operátory

Symbol	Hodnota (hex)	Popis
OP_1ADD	0x8b	Přičte 1 k položce na vrcholu zásobníku
OP_1SUB	0x8c	Odečte 1 od položky na vrcholu zásobníku
OP_2MUL	0x8d	Neplatné (vynásobí položku na vrcholu zásobníku hodnotou 2)
OP_2DIV	0x8e	Neplatné (vydělí položku na vrcholu zásobníku hodnotou 2)
OP_NEGATE	0x8f	Obrátí znaménko položky na vrcholu zásobníku
OP_ABS	0x90	Změní znaménko položky na vrcholu zásobníku na kladné
OP_NOT	0x91	Pokud je položka na vrcholu zásobníku 0 nebo 1, prohodí je, jinak vrátí 0.
OP_0NOTEQUAL	0x92	Pokud položka na vrcholu zásobníku je 0 vrátí 0, jinak vrátí 1.
OP_ADD	0x93	Vyzvedne dvě vrchní položky ze zásobníku, sečte je a výsledek vloží na vrchol zásobníku
OP_SUB	0x94	Vyzvedne dvě vrchní položky ze zásobníku, odečte první od druhé a výsledek vloží na vrchol zásobníku
OP_MUL	0x95	Neplatné (vynásobí dvě vrchní položky zásobníku)

Symbol	Hodnota (hex)	Popis
OP_DIV	0x96	Neplatné (vydělí druhou položku z vrcholu zásobníku první položkou)
OP_MOD	0x97	Neplatné (zbytek po vydělení druhé položky z vrcholu zásobníku první položkou)
OP_LSHIFT	0x98	Neplatné (posune doleva druhou položku z vrcholu zásobníku o počet bitů daný první položkou)
OP_RSHIFT	0x99	Neplatné (posune doprava druhou položku z vrcholu zásobníku o počet bitů daný první položkou)
OP_BOOLAND	0x9a	Výsledek logického AND dvou položek z vrcholu zásobníku
OP_BOOLOR	0x9b	Výsledek logického OR dvou položek z vrcholu zásobníku
OP_NUMEQUAL	0x9c	Vrací TRUE, pokud dvě položky na vrcholu zásobníku mají shodné číselné hodnoty
OP_NUMEQUALVERIFY	0x9d	Stejné jako NUMEQUAL, ale následně zavolá OP_VERIFY
OP_NUMNOTEQUAL	0x9e	Vrací TRUE pokud dvě položky na vrcholu zásobníku nemají shodné číselné hodnoty OP_LESSTHAN
0x9f	Vrací TRUE pokud druhá položka z vrcholu zásobníku je menší než první	OP_GREATERTHAN
0xa0	Vrací TRUE pokud druhá položka z vrcholu zásobníku je větší než první	OP_LESSTHANOREQUAL
0xa1	Vrací TRUE pokud druhá položka z vrcholu zásobníku je menší nebo rovna než první	OP_GREATERTHANOREQUAL
0xa2	Vrací TRUE pokud druhá položka z vrcholu zásobníku je větší nebo rovna než první	OP_MIN

Symbol	Hodnota (hex)	Popis
0xa3	Vrátí menší z dvou vrchních položek na zásobníku	OP_MAX
0xa4	Vrátí větší z dvou vrchních položek na zásobníku	OP_WITHIN

Kryptografické a hašovací operace zobrazuje operátory kryptografických funkcí.

Table 7. Kryptografické a hašovací operace

Symbol	Hodnota (hex)	Popis
OP_RIPEMD160	0xa6	Vrací RIPEMD160 haš položky z vrcholu zásobníku
OP_SHA1	0xa7	Vrací SHA1 haš položky z vrcholu zásobníku
OP_SHA256	0xa8	Vrací SHA256 haš položky z vrcholu zásobníku
OP_HASH160	0xa9	Vrací RIPEMD160(SHA256(x)) haš položky z vrcholu zásobníku
OP_HASH256	0xaa	Vrací SHA256(SHA256(x)) haš položky z vrcholu zásobníku
OP_CODESEPARATOR	0xab	Označí začátek dat, která budou kontrolována na jejich podepsanost
OP_CHECKSIG	0xac	Z vrcholu zásobníku odebere veřejný klíč a podpis a ověří, zda podpis odpovídá hašovaným datům transakce, vrací TRUE pokud se shodují
OP_CHECKSIGVERIFY	0xad	Stejné jako CHECKSIG, ale následně zavolá OP_VERIFY
OP_CHECKMULTISIG	0xae	Zavolá CHECKSIG pro každý poskytnutý pár podpisu a veřejného klíče. Všechny musí souhlasit. Kvůli chybě v implementaci je vrácena přebytečná hodnota označená prefixem ON_NOP.
OP_CHECKMULTISIGVERIFY	0xaf	stejné jako CHECKMULTISIG, ale následně zavolá OP_VERIFY

Neoperátory zobrazuje neoperátory "Script language","symbols")

Table 8. Neoperátory

Symbol	**Hodnota (hex)**	**Popis**
OP_NOP1-OP_NOP10	0xb0-0xb9	Nedělá nic, ignorováno

[tx_script_ops_table_internal] zobrazuje kódy operátorů vyhrazených pro vnitřní použití parserem skriptu.

1. Rezervované OP kódy pro vnitřní použití parserem

Symbol	Hodnota (hex)	Popis
OP_SMALLDATA	0xf9	Reprezentuje malé datové pole
OP_SMALLINTEGER	0xfa	Reprezentuje malé číselné datové pole
OP_PUBKEYS	0xfb	Reprezentuje položky veřejného klíče
OP_PUBKEYHASH	0xfd	Reprezentuje položku haše veřejného klíče
OP_PUBKEY	0xfe	Reprezentuje položku veřejného klíče
OP_INVALIDOPCODE	0xff	Reprezentuje jakýkoliv OP kód dosud nepřiřazený