VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ANALYTICKÉ ZPRACOVÁNÍ BLOCKCHAINU KRYPTOMĚN

CRYPTOCURRENCIES BLOCKCHAIN ANALYSIS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. MARTIN OČENÁŠ
AUTHOR

VEDOUCÍ PRÁCE — Ing. VLADIMÍR VESELÝ, Ph.D.
SUPERVISOR

BRNO 2017

Zadání diplomové práce/19354/2016/xocena04

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Očenáš Martin, Bc.**
Obor: Počítačové sítě a komunikace
Téma: **Analytické zpracování blockchainu kryptoměn**
Cryptocurrencies Blockchain Analysis
Kategorie: Počítačové sítě

Pokyny:

1. Analyzujte syntax a sémantiku blockchainů nejvýznamějších kryptoměn jako Bitcoin, Litecoin, Ethereum, Ripple.
2. Seznamte se s existujícími nástroji pro forenzní práci s těmito blockchainy, zejména s nástrojem vyvinutým NES@FIT.
3. Diskutujte potencální informace, které se dají zjistit korelací obsahu blockchainu s veřejně dostupnými informacemi. Navrhněte rozšíření stávající aplikace.
4. Podle doporučení vedoucího implementujte prototyp aplikace realizující primárně forenzní činnost nad obsahem blockchainu (vizualizaci vztahů entit, shlukování dat, tainted analýza, aj.).
5. Ověřte funkčnost prototypu na anotovaném případě, diskutujte jeho škálovatelnost a použitelnost pro jiné kryptoměny.

Literatura:

- Bitcoin Project, *Developer Guide - Bitcoin*,https://bitcoin.org/en/developer-guide
- S. Nakamoto, *Bitcoin:A P2Peer Electronic Cash System*, https://bitcoin.org/bitcoin.pdf
- F. Reid, M. Harrigan, An Analysis of Anonymity in the Bitcoin System
- Nástroje: Blockchain Ltd., *Bitcoin Block Explorer*, https://blockchain.info/; A. Janda, *Wallet Explorer*, https://www.walletexplorer.com/; M. Spagnuolo, *BitIodine project*, https://bitiodine.net/

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3 včetně.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Veselý Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ v BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato práce pojednává o důležitých existujících kryptoměnách a principem na kterých fungují. Zaměřuje se zejména na jejich principiální rozdíly. Dále pojednává o možnostech analýzy obsahu blockchainu kryptoměny Bitcoin. Další část práce popisuje vylepšení nástroje pro analýzu blockchainu. Dále popisuje nástroj pro analýzu kryptoměn a jeho implementovaná rozšíření.

Abstract

This thesis describes important existing cryptocurrencies and their basis principles. Especially it describes differences between this cryptocurrencies and theid basis principles. Also describes posibilites for analysis of Bitcoin blockchain. Next part describes improvments of tool for blockchain analysis. Futher it describes cryptocurrency analyzing tool, and it's implemented extensions.

Klíčová slova

Kryptoměna, protokol, Bitcoin, Litecoin, DogeCoin, DashCoin, Monero, ByteCoin, Ripple, Ethereum, Analýza, Blockchain, Clusterizace, Vizualizace.

Keywords

Cryptocurrency, protocol, Bitcoin, Litecoin, DogeCoin, DashCoin, Monero, ByteCoin, Ripple, Ethereum, Analysis, Blockchain, Clusterization, Visualization

Citace

OČENÁŠ, Martin. *Analytické zpracování blockchainu kryptoměn.* Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Veselý Vladimír.

Analytické zpracování blockchainu kryptoměn

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana doktora Vladimíra Veselého. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Martin Očenáš
23. května 2017

Obsah

1 Úvod **3**

2 Principy a srovnání kryptoměn **4**
2.1 Základní kryptoměnové principy . . . 4
2.1.1 Motivace pro používání kryptoměn . . . 4
2.1.2 Myšlenka fungování kryptoměn . . . 4
2.2 Základní protokoly . . . 5
2.2.1 Bitcoin . . . 5
2.2.2 CryptoNote . . . 7
2.2.3 Ripple . . . 9
2.2.4 Ethereum . . . 11
2.3 Existující kryptoměny . . . 13
2.3.1 Bitcon . . . 14
2.3.2 Litecoin . . . 14
2.3.3 DogeCoin . . . 15
2.3.4 Bytecoin . . . 15
2.3.5 Monero . . . 16
2.3.6 Dash . . . 16
2.3.7 Ripple . . . 17
2.3.8 Ethereum . . . 18
2.4 Srovnání kryptoměn . . . 18
2.4.1 Technické srovnání . . . 19
2.4.2 Finanční srovnání . . . 20

3 Existující nástroj a plány na jeho vylepšení **21**
3.1 CCA a jeho optimalizace . . . 21
3.1.1 Úložiště dat . . . 21
3.1.2 Import dat z blockchainu . . . 22
3.2 Rozšíření CCA . . . 23
3.2.1 Získatelné informace z blockchainu . . . 23
3.2.2 Korelace obsahu blockchainu s dostupnými informacemi . . . 24
3.2.3 Trasování a vizualizace transakcí . . . 25

4 Implementované změny v CCA **26**
4.1 Architektura aplikace . . . 26
4.2 Import blockchainu . . . 28
4.2.1 Blockchain parser . . . 28
4.2.2 Zpracování blockchainu . . . 28

4.3 Databáze Neo4j . . . 29
4.3.1 Schéma databáze . . . 29
4.4 Metody clusterizace . . . 31
4.4.1 Clusterizace podle vstupů transakce . . . 32
4.4.2 Clusterizace podle veřejného klíče . . . 32
4.5 Vizualizace transakcí . . . 32
4.6 Nové obrazovky . . . 33
4.6.1 Detail vstupů a výstupů transakcí . . . 33
4.6.2 Veřejné klíče . . . 33

5 Praktické použití CCA **34**
5.1 Nasazení a ovládání CCA . . . 34
5.2 Výkonnost CCA . . . 35
5.2.1 Parsování blockchainu . . . 35
5.2.2 Vizualizace . . . 35
5.3 Analýza konkrétního případu . . . 36
5.4 Použitelnost pro jiné kryptoměny . . . 37

6 Závěr **38**

Literatura **39**

Přílohy **41**

A Obsah CD **42**

B Popis bitcoin scriptu **43**
B.1 Pay-to-Pubkey (P2PK) . . . 43
B.2 Pay-to-PubkeyHash (P2PKH) . . . 44
B.3 Multisig . . . 45
B.4 Pay-to-ScriptHash (P2SH) . . . 46

C Příklad vizualizace transakce **48**
C.1 Výkonnost zobrazování . . . 49

D Nové obrazovky aplikace **51**

Kapitola 1

Úvod

Tato diplomová práce se zaobírá existujícími kryptoměnami principem jejich fungování a možností jejich praktického využití. Dále popisuje možnosti získávání informací z dat, získaných z blockchainu kryptoměny Bitcoin.

V teoretické části práce jsou popsány kryptografické měny (tzv. kryptoměny), práce představuje základní princip a myšlenku fungování a používání kryptoměn jako celku a následně je zde představeno několik významných existujících kryptoměn, jejich princip fungování a jejich srovnání z technického i finančního hlediska.

Cílem praktické části této práce je vylepšit a rozšířit nástroj pro forenzní analýzu kryptoměn, který vznikl v rámci diplomové práce Tomáše Drozdy [1]. Tento nástroj obsahuje několik chyb a nedostatků. Jedním z cílů této práce je odstranit tyto nedostatky. Druhým cílem je rozšířit funkcionalitu tohoto nástroje o pokročilejší možnosti trasování peněz. V kapitole 3 je popsán tento nástroj a plány na jeho vylepšení. V kapitole 4 jsou popsány implementované změny v nástroji.

V rámci semestrálního projektu jsem prozkoumal existující kryptoměny (popsané v kapitole 2) a navrhl rozšíření existujícího nástroje pro analýzu kryptoměn (popsáno v kapitole 3).

Kapitola 2

Principy a srovnání kryptoměn

V této kapitole jsou popsány základní principy, na kterých pracují kryptoměny. Je zde popsána motivace pro vznik kryptoměn. V další části kapitoly jsou popsány protokoly, které tvoří základ většiny kryptoměn. Následuje popis několika konkrétních význačných kryptoměn.

2.1 Základní kryptoměnové principy

Kryptoměny (neboli kryptografická měna) je platidlo, stejně jako peníze. Od peněz se však zásadně liší ve způsobu, jak pracuje celý finanční systém a jak se s nimi platí. Tato sekce popisuje základní myšlenku kryptoměn, společnou pro většinu jejich zástupců.

2.1.1 Motivace pro používání kryptoměn

Proč vytvářet nějaké zvláštní počítačové měny když tu máme normální peníze, banky umožňující platby online, služby typu *PayPal* a podobně? Všechny tyto systémy totiž mají jednu společnou věc - centrální autoritu, která vše spravuje. Ať už se jedná o vládu, jednotlivé banky nebo jen nějakou komunitu lidí, kteří spravují určitý systém, tak pro všechny tyto systémy vždy existuje skupina lidí, která může danou měnu ovládat. Například může uměle způsobit inflaci nebo deflaci dané měny, přidáváním nebo odebíráním peněz v oběhu, může zmrazit nebo i zabavit určité finanční prostředky, a také má téměř dokonalý přehled o tom, jak se s penězi nakládá a jaké peníze jsou kým vlastněny.

Kryptoměny se tedy snaží vytvořit platební systém, který nemá žádnou centrální správu nebo autoritu (je tzv. *decentralizovaný*) a o veškeré fungování sítě se zasluhují sami uživatelé. U většiny kryptoměn je použit princip tzv. *hrubého konsenzu*, tedy že platí to, na čem se shodne většina uživatelů. Takový systém pak není možné nijak jednostranně řídit, a tak jakékoli změny probíhají postupně. Prakticky se tak nemůže stát, že by kryptoměna velmi rychle zkrachovala nebo byla odepřena určitým uživatelům.

Další motivací je mít měnu, která je anonymní nebo alespoň pseudonymní, tedy není možné spárovat účty s konkrétními osobami.

2.1.2 Myšlenka fungování kryptoměn

V předchozí sekci je popsáno, proč kryptoměny vlastně vznikly, nyní se podíváme na to, jak zhruba fungují. Pokud totiž síť nemá centrální autoritu, pak sice není možné aby autorita měnu uměle ovlivňovala, ale také se tím ztrácí možnost, jak měnu jednoduše spravovat, a

musí tedy existovat systém, který umožní provádět platby a ověřit jejich platnost, také je potřeba nějakým způsobem monitorovat stavy účtů, aby bylo možné ověřit, že uživatel má na danou platbu opravdu dostatek prostředků.

Základním problém je tedy **autentizace**, tedy jak ověřit, že uživatel je opravdu tím pravoplatným disponentem daného účtu, za kterého se vydává. V systémech kryptoměn není účet vázán na jméno, či jiné označení konkrétní osoby, a je tedy nutné vlastníka identifikovat jinak. V kryptoměnových systémech se často pro identifikaci používá asymetrická kryptografie, kdy veřejný klíč je sdělen všem uživatelům a privátní klíč zná pouze vlastník účtu. Každá transakce, provedená ve jménu daného účtu, je pak podepsána privátním klíčem vlastníka a všichni ostatní uživatelé sítě mohou snadno ověřit, že tento podpis je platný. V kryptoměnových systémech je tedy vlastnictví účtu vázáno pouze na vlastnictví privátního klíče, pokud uživatel o daný klíč přijde, nebo je mu odcizen, neexistuje žádná možnost jak znovu získat vládu nad svým účtem.

Dalším problémem je **autorizace a ověření plateb**, tedy možnost ověřit, že danou platbu opravdu provedl vlastník účtu a že na účtu je dostatek prostředků k provedení dané platby. Aby mohl kterýkoli uživatel ověřit dostatek peněz k platbě, je potřeba aby každý uživatel měl přehled o stavech všech účtů. K tomuto se používá koncept tzv. *blockchainu*, kde blockchain je určitá datová struktura, která obsahuje všechny provedené transakce, dalo by se říct, že se jedná o archiv všech transakcí. Kopie blockchainu je uložena u všech uživatelů sítě a v případě provedení platby, musí být informace o jejím provedení též rozdistribuována všem, aby mohli aktualizovat svoji lokální kopii blockchainu. Každý uživatel sítě pak může jednoduše určit stav kteréhokoli účtu v síti, jednoduchým projitím celého blockchainu a počítám příchozích a odchozích transakcí pro daný účet.

Dalším problém je způsob jak v kryptoměnové síti zajistit **vytváření nových peněz**, pokud v síti přibývá počet klientů, je žádoucí, aby byl v oběhu dostatek peněz. Když však neexistuje centrální autorita, která by peníze mohla emitovat, musí peníze vznikat na účtech klientů. Další problém tedy je, jak určit účet, na kterém by měly peníze vzniknout. Je potřeba nějakým způsobem vybrat klienta, který je pro síť jistým způsobem užitečný a zaslouží si nové peníze. K tomuto se v kryptoměnách často používá princip *proof of work* (důkaz o provedení práce), kdy při ověřování každé transakce nebo bloku transakcí je vytvořena matematická úloha, jejíž řešení vyžaduje velký výpočetní výkon počítače, tato úloha je předložena všem klientům. Klient, který jako první poskytne řešení úlohy, je vybrán jako příjemce nových peněz.

2.2 Základní protokoly

V této sekci jsou uvedeny některé význačné protokoly, na kterých staví jednotlivé kryptoměny. Popisuje se zde princip, jak daná síť funguje a co je společné pro všechny kryptoměny, založené na daném protokolu.

2.2.1 Bitcoin

Bitcoin je otevřený protokol, který původně vznikl jako základ pro stejnojmenou kryptoměnu, později byl stejný protokol převzat i do jiných kryptoměn jako je například Litecoin (2.3.2) nebo DogeCoin(2.3.3). Tento dokument dále popisuje jen věci typické pro Bitcoin a protokoly z něj odvozené, zejména na věci, které jsou v jiných protokolech řešeny jinak.

V Bitcoinu se uzly v síti dělí na 2 skupiny: uživatelé a těžaři (angl. miner). Uživatel pouze používá síť jako běžný klient. Miner poskytuje svůj výpočetní výkon pro těžbu bloků (viz. dále) a poskytuje tím prostředky pro běh sítě.

2.2.1.1 Účty

Bitcoin obsahuje platební účty, obdobné jako v každém finančním systému, kde majitel může disponovat s penězi na účtu. K účtu je vytvořena jedna dvojice soukromého a veřejného klíče, které slouží pro ověřování akcí provedených s účtem a pro adresování účtu. Veškeré příchozí transakce jsou přímo nebo nepřímo cíleny na veřejný klíč účtu.

Účet má pouze jednu dvojici klíčů, které jsou použity ve všech transakcích, lze tedy snadno zjistit všechny transakce, které provedl daný účet.

2.2.1.2 Blockchain

Blockchain je základní datová struktura Bitcoinu, ve které jsou uloženy všechny transakce, provedené v celé historii sítě. Blockchain je uložen u každého uzlu sítě a též každá jeho aktualizace se distribuuje všem uzlům.

Uvnitř blockchainu jsou transakce spojeny do tzv. bloků, což je skupina transakcí uzavřena do jedné datové struktury. Samotné bloky jsou v blockchainu seřazeny v lineárním seznamu, kde každý blok obsahuje hash předchozího bloku. Toto zřetězení zabraňuje tomu, aby byl vyměněn blok vytvořený někdy v historii Bitcoinu.

Transakce jsou uvnitř bloku organizovány do tzv. Merkle Tree. Merkle tree je binární strom, kde každý uzel obsahuje hash vypočtený podle hodnot jeho synů. V Bitcoinu transakce tvoří listy stromu, z nich je vypočten hash a každá vyšší úroveň stromu obsahuje hash, získaný z hashe svých synů. Hash kořenového uzlu je uložen v hlavičce bloku. Tento princip umožňuje validovat integritu transakcí, aniž by musely být validovány všechny transakce v bloku.

Pomocí blockchainu se počítají i zůstatky na účtech. Při provádění platby z účtu A je jako zdroj platby, použitá jiná transakce, která byla provedena ve prospěch účtu A, ta je následně označena jako použitá a už nemůže být použita jako zdroj jiné platby. Zůstatek na účtu tedy lze spočítat jako sumu hodnot všech nepoužitých transakcí, provedených ve prospěch daného účtu.

2.2.1.3 Proof-of-work

Proof-of-work je základní princip na kterém stojí protokol Bitcoin. Definuje způsob, jak uživatel dokazuje práci, kterou odvedl pro síť, a jak moc si tedy zaslouží odměnu. Současně je tímto principem chráněna integrita blockchainu, protože změna jakéhokoli bloku, by znamenala přepočtení proof-of-work pro tento a všechny následující bloky.

Procesu vytváření proof-of-work se říká těžení (angl. mining) a od toho je odvozen pojem miner - uživatel, který těží bloky.

Samotný proof-of-work se provádí při ukládání bloku do blockchainu, síť v tomto okamžiku vygeneruje hash H a miner a snaží vyhledat nonce N takový, že hash N a hlavičky bloku je menší nebo roven H. Jakmile je takovýto nonce nalezen, je přidán do bloku, blok je uložen do blockchainu a miner, který nonce objevil je odměněn určitým množstvím peněz dané měny. Samotná hashovací funkce je různá, pro jednotlivé kryptoměny založené na Bitcoinu, proto zde není uvedena.

2.2.1.4 Vytvoření a provedení transakce

Transakce je vždy iniciována uživatelem a slouží k přesunu peněz na jiný účet. Pro získaní peněz pro transakci, musí uživatel vzít jednu nebo více nepoužitých transakcí, které přišly ve prospěch jeho účtu, a ty použít jako zdroj peněz pro odchozí platbu. Odchozí platbu je možné rozdělit mezi více příjemců, tudíž nepoužité peníze může odesilatel poslat zpět sám sobě a použít je jako zdroj pro jinou transakci.

Samotná transakce je datová struktura obsahující vstupní a výstupní platby. Jakmile je transakce vytvořena, je podepsána uživatelovým soukromým klíčem a předána ke zpracování. Není nutné, aby výstupní platby pokrývaly všechny peníze ze vstupu, pokud je utraceno méně, než je dáno součtem vstupních transakcí, pak zbytek peněz slouží jako odměna pro minera, který transakci vytěží a zařadí do blockchainu, či se zbytek vrací odesilateli jako rozměrná částka na drobné. Jelikož pořadí vytěžení nových transakcí není ničím dáno, zvetšením odměny pro minery může uživatel upřednostňovat zpracování svojí transakce.

Po odeslání transakce se ke zpracování dostává miner. Jakmile má miner dostatek nezpracovaných transakcí, tak je zformuje do bloku a následně se tento blok snaží vytěžit. Pokud se mu to podaří dříve, než dané transakce vytěží jiný miner, tak uspěl, vytěžený blok se rozešle zbytku sítě a zařadí do blockchainu. Miner, který takto vytěžil blok, obdrží odměnu, kterou stanovuje síť za vytěžení bloku, plus poplatek obsažený v daných transakcích.

2.2.2 CryptoNote

Samotný protokol CryptoNote byl vytvořen s myšlenkou, aby byl evolucí Bitcoinu a vytvořil lepší a anonymnější síť. Původně byl vytvořen pouze jako protokol, až později vznikly jednotlivé měny, založené na tomto protokolu. CryptoNote je od počátku navržen poměrně flexibilní, tak aby umožňoval jednotlivým měnám některé zásadní vlastnosti používat jinak. Mezi významné zástupce měn, založené na tomto protokolu, patří např: Bytecoin (2.3.4), Monero (2.3.5), Dashcoin.

Zásadní novinkou, kterou CryptoNote přináší, je anonymizace transakcí, která v CryptoNote zapnutá a zásadě se používá vždy. Tento systém, popsaný v 2.2.2.4, zabraňuje tomu, aby kdokoli jiný než odesilatel a příjemce věděl který účet byl zdrojem a který příjemcem transakce.

2.2.2.1 Účty

Účty v CryptoNote fungují podobně jako v Bitcoinu, zásadní odlišnost je však v tom, že Bitcoin používal pro každý účet pouze jednu dvojici soukromého a privátního klíče. CryptoNote potřebuje, kvůli anonymizaci transakcí, těchto párů více, v podstatě má jeden pár soukromý a veřejný klíč pro samotný účet a pak jeden pár pro každou příchozí, nepoužitou transakci.

2.2.2.2 Transakce

Transakce jsou v CryptoNote používány téměř stejně jako v rodičovském Bitconu. Jedna transakce se může skládat z více vstupů a výstupů. Kde každý vstup je záznam o již existující transakci, která slouží jako zdroj peněz pro nově vzniklou transakci. Každý vstup je zvlášť podepsán, což zaručuje vlastnictví daných transakcí odesilatelem. Každý výstup je dvojice (množství, adresát), která specifikuje kolik peněz se má poslat kterému příjemci.

Celková suma peněz ve všech výstupech nesmí překročit sumu peněz ve vstupních transakcích.

V transakci jsou uvedeny všechny veřejné klíče, které byly využity k anonymizaci podpisu (viz. anonymizace transakce - 2.2.2.4).

V každé transakci je přiložen obraz soukromého klíče, použitého k podepsání každé vstupní transakce. Tento obraz slouží jako ochrana před dvojím utracením transakce (viz. ochrana před dvojím utracením - 2.2.2.5). Jelikož privátní klíč je jedinečný pro každou transakci, ani tento obraz nijak neškodí anonymitě odesilatele.

Každá transakce má specifikovaný tzv. unlock-time. Jedná se časové razítko, které definuje čas, kdy nejdříve je možné tuto transakci použít jako vstup platby. Do této doby je transakce považována za uzamčenou.

2.2.2.3 Blockchain

Podobně jako v Bitcoinu, i CryptoNote používá datovou strukturu blockchain, která uchovává veškeré záznamy o provedených transakcích. Tato datová struktura je uložena v každém uzlu sítě. Blockchain v CryptoNote v principu funguje stejně jako blockchain v Bitcoin, který byl popsán v kapitole 2.2.1.2.

Změnou oproti Bitcoinu je způsob, jak se určuje nový platný blockchain. Zatímco v Bitcoinu je pomocí konsensu zvolen jako nový blockchain ten nejdelší validní, v CryptoNote se nový blockchain volí pomocí konsensu, ve kterém hlasují všechny uzly sítě rovným hlasem a není specifikováno na jakém základě. Tento systém umožňuje jednotlivým měnám měnit způsob, jak bude volen nový blockchain.

2.2.2.4 Anonymizace transakce

Zásadní funkcí CryptoNote je možnost anonymizovat transakce, tedy skrýt kdo byl odesilatelem a kdo příjemcem dané platby. Cílem je udělat takový systém, kde nebude možné poznat, že 2 platby měly stejného odesilatele nebo stejného adresáta. Myšlenka anonymizace je skrýt odesilatele ve vetší skupině účtů, ze které nejde poznat kdo transakci skutečně odeslal a příjemce skrýt zcela, napříč celou sítí dané měny.

Základním principem pro skrytí identity odesilatele je speciální forma podpisu transakce - tzv. **Ring signature**, který funguje přibližně takto: Existuje skupina účtů, kde každý má vlastní dvojici soukromý a veřejný klíč, jeden z jejich členů vytvořil tento podpis, je výpočetně nezvládnutelné zjistit, který to byl. Podpis je vytvořen pomocí zprávy, jednoho privátní klíče a všech veřejných klíčů. Odesilatel transakce tedy nemůže být jednoznačně identifikován, je známa pouze skupina účtů, kde jeden z nich je odesilatelem.

Na straně příjemce, je pro každou platbu vytvořen nový veřejný klíč a nová adresa, i když se jedná o stejnou dvojici odesilatel-příjemce. Tím pádem, ve více platbách není možné dohledat, zda se jedná o stejného příjemce. Nová adresa je vypočtena na základě soukromého klíče příjemce, takže jen on může disponovat s danou platbou. Nový veřejný klíč vygeneruje příjemce a předá jej odesilateli pomocí protokolu Diffie-Hellman.

Problémem tohoto přístupu je, že na každou transakci, která byla připsána ve prospěch určitého účtu, musí daný účet uchovávat další soukromý klíč.

2.2.2.5 Ochrana před dvojím utracením

Ke každé platbě je dodán obraz (v anglickém originále "image") soukromého klíče odesilatele. Obraz je spočítán pomocí hashovací funkce tak, že není možné z obrazu odvodit

soukromý klíč. Samotný obraz je uložen v blockchainu, obraz je možné validovat pomocí matematických funkcí, které jsou schopny odhalit falešný klíč. Tento obraz je snadno vystopovatelný ve více transakcích. Pokud se uživatel pokusí jednu platbu utratit vícekrát, bude nalezen obraz klíče ve více transakcích a přijata bude však jen jedna transakce.

2.2.2.6 Tvorba nových peněz

Tvorba nových peněz a celý koncept dokazování práce je v CryptoNote vyřešen velmi adaptabilně. Není totiž definován žádný konkrétní systém přidělování odměn, je pouze definováno, že platí to, na čem se shodne většina uzlů v síti. Avšak většina měn, založených na CryptoNote, využívá běžný koncept proof-of-work postavený na hashovacím algoritmu CryptoNight.

CryptoNight je hashovací algoritmus, používaný v CryptoNote sítích. Oproti běžně používaným hashovacím algoritmům byl navržen pro použití v kryptoměnách, a tak není závislý jen na samotném výpočetním výkonu, ale také na náhodných přístupech do velké paměti. Tento přístup velice znesnadňuje tvorbu specializovaného hardware pro výpočet tohoto algoritmu.

2.2.3 Ripple

Ripple se velmi odlišuje od ostatních kryptoměn svojí klientelou, není totiž stavěn jako rozsáhlá síť jednotlivých nedohledatelných uživatelů, kteří chtějí provádět své platby mezi sebou, ale původně byl navržen pro bankovní instituce a mezibankovní převody. Zjednodušeně pak lze říct, že uzly v síti mohou být buď uživatelé, kteří využívají službu, nebo centrální uzly, které služby poskytují.

Stejně jako v ostatních kryptoměnách, i Ripple je založen na asymetrické kryptografii. Každý uživatel má vlastní dvojici veřejného a soukromého klíče a každá transakce musí být podepsána uživatelovým soukromým klíčem.

2.2.3.1 Důležité pojmy sítě:

Gateway Gateway je jedna ze základních částí Ripple sítě, je to uzel, který propojuje síť Ripple se zbytkem světa. Typicky se jedná o bankovní instituci, která vlastní určitý počet peněz v Ripple síti a umožňuje jejich nákup a prodej. Vlastní peníze uživatelů jsou vždy uloženy v určité gateway nebo v několika z nich.

Consensus ledger Consensus ledger, překládáno jako *účetní kniha*, je databáze všech účtů a zůstatků na daných účtech. Kopie této databáze je přístupná všem uzlům sítě. Vždy v případě provedení platby, je vydána nová verze účetní knihy. Validaci plateb provádějí některé ze serverů sítě, o tom, co bude obsahem nové verze účetní knihy, rozhodují tyto servery metodou konsensu.

Trustlines Trustline, přeložitelné jako *řetěz důvěry*, je důležitým prvkem Ripple. Trustline je mezi koncovým uživatelem a gateway a je omezena na určitou měnu a určitou hodnotu. Uživatel tím vyjadřuje svoji důvěru pro danou gateway, čímž umožňuje přijímat platby v dané měně, od dané gateway, ale pouze do té doby, než zůstatek na účtu uživatele v dané gateway přesáhne stanovenou hodnotu.

Důvěra se nenavazuje pro nativní měnu XRP (viz. 2.2.3.2), tam je implicitně ustanovena mezi všemi uživateli a gateway.

Pokud má uživatel ustanovenou důvěru s více gateway pro stejnou měnu, pak Ripple může přesouvat peníze uživatele v této měně, mezi těmito gateway. Zůstatek uživatele zůstane stejný, ale je možné tím provézt jinou transakci, která vyžaduje převod mezi danými gateway.

Rippling Každý uživatel má zůstatek v ne-XRP měnách uložen v určité gateway. Pokud je však potřeba provézt převod mezi uživateli A,B, kde tito uživatelé nemají společnou gateway pro danou měnu, pak je třeba použít rippling. Uvažujme, že uživatel A má zůstatek v gateway *G1*, uživatel B v *G2*, dále uvažujme, že uživatel C má uloženy zůstatky v dané měně ve obou gateway *G1*,*G2* a oběma gateway důvěřuje na dostatečně velkou sumu peněz. *Rippling* je pak proces, kdy pro provedení transakce mezi od uživatele A ve prospěch uživatele B, je uživateli C převedena část zůstatku z *G1* do *G2*, aby mohl mohl být uživateli B připsán odpovídající obnos a obě gateway měli stále stejný celkový zůstatek pro danou měnu.

Aby mohla být důvěra mezi uživatelem a gateway použita pro rippling, musí uživatel explicitně povolit rippling pro danou důvěru.

Market makers Market makers jsou uzly sítě, které mají ustanovenou důvěru s různými gateway a umožňují je tím propojit. V síti Ripple je potom možné, aby dva uživatelé, kteří nemají mezi sebou ustanovenou přímou důvěru, byli propojeni nepřímo skrz uzly, kterým důvěřují. Je žádoucí, aby market maker měl zůstatek ve více měnách a umožnil tak automatickou směnu při provádění transakcí.

2.2.3.2 Měny v Ripple

Ripple má jednu svoji nativní měnu, která se nazývá XRP. Ta platí pouze v síti Ripple. V době vzniku sítě bylo vytvořeno 100 miliard XRP a další už v systému nevznikají. Původní XRP vlastnili autoři sítě, kteří je postupně distribuují.

Důležitou a zajímavou vlastností Ripple je, že umožňuje převod nejen v XRP ale i v jiných měnách, i těch z reálného světa, navíc je možné, aby platba byla odeslána v jedné měně a doručena v jiné. Gateway v síti totiž může mít uloženou hotovost ve více měnách a může umožňovat jejich směnu. Gateway pouze definuje směnný kurz a zbytek je schopna zajistit samotná síť.

2.2.3.3 Provedení transakce

Provedení uživatelské transakce se skládá z několika kroků:

1. Uživatel vytvoří transakci.
2. Uživatel podepíše transakci svým soukromým klíčem.
3. Uživatel odešle podepsanou transakci Rippled serveru.
4. Rippled server rozešle tuto transakci ostatním serverům.
5. Rippled servery v konsensu rozhodnout, které transakce budou započteny v nové účetní knize.
6. Rippled servery jednotlivě začlení dané transakce do své nové účetní knihy a porovnají své výsledky.

7. Pokud se dostatek Rippled serverů shodne na výsledku, je nová účetní kniha prohlášena za platnou.

Jakmile jsou tyto kroky provedeny je transakce zaevidována v účetní knize, bez možnosti zrušení. Při provedení transakce není nutné provádět žádnou formu *proof-of-work*, takže její provedení je rychlé. V Ripple se nová účetní kniha vytváří každých několik sekund.

Každá transakce je zpoplatněna 0.000012 XRP, pokud je síť vytížena, může být tento poplatek vyšší. Navíc je poplatek zvýšen pro uživatele, který vytváří větší množství transakcí. Není určena žádná horní hranice, jak vysoko mohou poplatky vystoupat. Důvodem pro zvyšování poplatků v případě vytížení, je ochrana před DDoS útoky, autora DDoS útoku by pak útok stál velké množství peněz. Poplatek je nenávratně zničen, jakmile je transakce dokončena.

2.2.4 Ethereum

Ethereum je další z protokolů a dalo by se říct konceptů fungování kryptoměn. Protokol prakticky funguje nad jednou kryptoměnou zvanou taktéž Ethereum(viz. 2.3.8). Oproti ostatním kryptoměnám se totiž neomezuje jen na samotnou měnu a posílání peněz v této měně. Zavádí totiž mechanismy, které umožňují vybudovat systémy nad touto měnou.

Motivací pro vznik Etherea bylo vytvořit síť, nad kterou bude možné provádět i složitější, uživatelem definované akce, než jen prosté převedení peněz z jednoho účtu na druhý. Ostatní protokoly (např. Bitcoin) mají skriptovací jazyk, který umožňuje takto transakce řídit, ale jsou velmi omezené. Ethereum v tomto chce mít k dispozici jazyk, který bude výpočetně úplný a bude stavový, tedy bude možné provádění transakce řídit podle jevů, které transakci předcházely.

V konečné podobě tedy Ethereum vytváří síť, nad kterou mohou fungovat rozdílné aplikace, které potřebují provádět bezpečné platby. Definuje kryptoměnovou síť s blockchainem a nad ním výpočetně úplný jazyk pro popis transakcí. Pomocí tohoto jazyka je možné definovat vlastní pravidla pro přístup k penězům a jejich řízení.

2.2.4.1 Účty

Ethereum používá účty podobně, jako kterýkoli finanční systém, ale jsou zde určité věci specifické pro Ethereum. U účtů je obsaženo několik položek:

- **Nonce** - čítač, který slouží pro ochranu proti útoku dvojího utracení,
- **zůstatek** - množství prostředků, které jsou k dispozici na účtu,
- **řídicí kód** - pouze u některých účtů,
- **sklad** - sloužící jako paměť účtu.

Hlavní věcí, která je u Etherea odlišná jsou různé typy účtů. Oproti ostatním kryptoměnovým protokolům, Ethereum definuje 2 typy účtů:

- **Externě vlastněný účet** (externally owned account, dále jen Externí účet).
- **Smluvní účet** (contract account).

Typy účtů jsou zásadně rozdílné. Externí účet je určen pro uživatele identifikovaného pomocí soukromého klíče (tímto uživatelem je přímo řízen), a může běžně posílat transakce. Smluvní účet oproti tomu nemá žádného explicitního vlastníka a je řízen kódem, který slouží jako program, účet reaguje na přijaté zprávy a podle svého kódu, potom vykoná příslušnou akci. I tento účet má vlastní zůstatek se kterým může disponovat a navíc má tzv. sklad, který umožňuje zprávy ukládat a tím pracovat jako stavový systém.

U smluvního účtu je důležité, že účet ani jeho řídicí kód nejsou na konkrétním stroji, ale jsou uloženy v síti a jejich kód vykoná libovolný miner, který bude zpracovávat transakci, která daný kód spustí. Samotný smluvní účet se vytváří pomocí transakce, odeslané z externího účtu, obsahující kód, který bude daný smluvní účet řídit. Od té doby je smluvní účet uložen v blockchainu.

2.2.4.2 Transakce

Ethereum používá transakce, podobně jako ostatní kryptoměny, jako způsob jak předat peníze z jednoho účtu na jiný. Vzhledem k různým typům účtů je omezeno, že transakce mohou vytvářet pouze externí účty. Kromě běžných položek, jako je adresát, podpis odesilatele a suma převedených prostředků, Ethereum přidává ještě další, specifické položky.

Jednou z nich je položka **data**, která není implicitně nijak využitá, není ani definováno, co má být jejím obsahem. Do této položky se mohou umístit data, potřebná pro aplikace, které Ethereum využívají.

Další položky se vztahují k tzv. **gas**, což je jednotka, kterou se v Ethereu platí za provádění výpočtů. Celý tento koncept je vytvořen proto, aby se zabránilo nekonečným smyčkám při provádění výpočtů, vzhledem k tomu, že Ethereum umožňuje vytvoření vlastního kódu pro zpracování transakce, jsou nekonečné smyčky možné. Proto odesilatel transakce specifikuje kolik gas povolí transakci spotřebovat a kolik odesilatel zaplatí, za každý spotřebovaný gas. Miner, který transakci zpracovává pak po každém výpočetním kroku ubere transakci určitý gas a pokud gas dojde, je transakce zamítnuta. Množství spotřebovaného gas se liší pro různé výpočetní operace a je přímo definováno ve standardu Etherea. Gas se kromě výpočtů spotřebovává i při přenosu transakce mezi uzly, a to přímo úměrně velikosti transakce. Veškerý spotřebovaný gas pak platí odesilatel transakce a to bez ohledu, jestli je transakce dokončena úspěšně nebo ne.

Další, spíše terminologickou, specifičností Etherea jsou tzv. **Zprávy**. Což je v Ethereu způsob, jakým se zbytkem sítě komunikují smluvní účty. V zásadě jsou velmi podobné jako transakce, liší se hlavně terminologicky. Nejdůležitější rozdíl je v tom, že zpráva je vygenerována kódem, který řídí smluvní účet, nikoli uživatelem. Může to tedy být způsob, jak se mezi sebou mohou dorozumívat různé smluvní účty.

Pokud určité zprávě dojde gas, tak je tato zpráva zrušena a taktéž jsou zrušeny všechny zprávy, které byly spuštěny danou zrušenou zprávou.

2.2.4.3 EVM

Pro řízení výpočtů má Ethereum definovaný vlastní výpočetní jazyk, nazvaný EVM kód, který je interpretován pomocí tzv. Ethereum virtual machine (EVM). EVM kód je výpočetně úplný jazyk, používající nízkoúrovňové instrukce, který má k dispozici zásobník a paměť s náhodným přístupem, navíc může využívat sklad, který je součástí účtu a který je persistentní.

EVM kód je využíván pro řízení smluvních účtů, kde definuje reakce na přijaté zprávy nebo transakce.

Samotný řídicí kód je pak tvořen běžnou posloupností instrukcí. Kromě běžných instrukcí, jako jsou aritmetické operace apod., obsahuje instrukce pro vytváření a modifikaci zpráv potažmo transakcí.

2.2.4.4 Blockchain

Ethereum používá blockchain velmi podobně jako Bitcoin, ukládá v něm záznamy o proběhlých transakcích. Oproti Bitcoinu Ethereum ukládá do blockchainu i stav vykonávání transakce v době, kdy byla do blockachainu uložena. Navíc je u bloku uloženo i jeho číslo a obtížnost.

Nový blockchain je určen pomocí konsensu, přičemž princip podkladů pro konsensus se v průběhu verzí Etherea liší.

Do verze 1.0 používá Ethereum běžný proof-of-work a blok, který je úspěšně vytěžen, je v konsensu zvolen jako vítězný a zařazen do blockchainu.

Od verze 1.1 plánuje Ethereum používat tzv. **proof-of-stake**. Proof-of-stake je koncept, kdy každý uzel vsadí určité množství peněz na to, který blok bude jako další zařazen do blockchainu. Uzel který vyhraje získá peníze navíc, uzel který prohraje peníze ztrácí. Účastnit se může kterýkoli uzel v síti. Uzel v podstatě vsadí na určitý blok a současně pro tento blok hlasuje tolika hlasy, kolik peněz na blok vsadil. Blok, který získá největší počet hlasů, je vítězem.

2.2.4.5 Provedení transakce

Transakce se vytváří podobně jako u ostatních kryptoměn. Samotnou transakci však může vytvořit pouze externí účet, další zprávy mohou vzniknout od smluvních účtů jako reakce na tuto transakci. Samotné provedení transakce je v Ethereu složitější. Po odeslání transakce ke zpracování provede miner přibližně tyto kroky:

1. Ověř validitu transakce.
2. Ověř, že odesilatel má dostatek peněz na to, aby zaplatil maximální možný spotřebovaný gas.
3. Proveď převod peněz, pokud je adresátem smluvní účet, vykonej jeho kód.
4. Pokud v průběhu vykonávání dojde gas, zruš transakci a všechno, co bylo transakcí spuštěno, a vrať změny.
5. Strhni odesilateli poplatek za spotřebovaný gas.
6. Ulož změny do blockchainu.

Transakce je do blockchainu uložena vždy minimálně proto, že je třeba uživateli strhnout spotřebovaný gas. Stejně jako u ostatních kryptoměn je transakce, která je jednou uložena v blockchainu, již nevratná.

2.3 Existující kryptoměny

Tato kapitola popisuje některé význačné kryptoměny. Kryptoměn existuje velké množství (v současnosti více něž 650), ale většina z nich je málo používána. Navíc velká část kryptoměn je založena na protokolech Bitcoin (viz. 2.2.1) nebo CryptoNote (viz. 2.2.2) a příliš se neodlišují, proto jsou zde popsány pouze některé významné.

2.3.1 Bitcon

Bitcoin, který při svém vzniku definoval stejnojmenný protokol (viz. kapitola 2.2.1), je v současnosti nejrozšířenější a neznámější kryptoměna světa. Mimo jiné je to také zcela první kryptoměna, která vznikla v roce 2009. Bitcoin je také považován za průkopníka a tvůrce oblasti kryptoměn.

Pro vytváření proof-of-work Bitcoin využívá dva průchody algoritmu *SHA256*. Ověřování transakce a samotný důkaz o provedení práce pak vypadá přibližně takto:

- Při vytvoření bloku transakcí síť určí hodnotu hash H.
- Miner musí nalézt takovou hodnotu T, aby platilo, že výsledek hashe bloku transakcí a T byl menší nebo roven H.

Hashovací funkce *SHA256* je jedna z funkcí rodiny SHA2, která byla původně zamýšlena pro kryptografické účely. Funkce SHA2 se totiž dá velmi jednoduše a efektivně implementovat v hardwaru. Tato vlastnost je pro kryptoměny nežádoucí, protože základními kameny sítě mají být uživatelé a jejich počítače, které by měly mít řádově stejnou výpočetní sílu. Jelikož specializovaný hardware je schopen počítat funkci *SHA256* nesrovnatelně rychleji než CPU nebo GPU tak klient, který užívá takovýto hardwarový akcelerátor, se tak stává význačným bodem v síti. Pro klienty s běžným počítačem pak nemá příliš smysl se snažit o těžbu Bitcoinů a síť se tak stává částečně centralizovanou.

2.3.1.1 Platby

V síti Bitcoin je základní platební jednotkou 1 Bitcoin (zkratka BTC). Jeden BTC je možno dělit až na 8 desetinných míst. Z důvodu praktických omezení, je minimální částka 0.00000546 BTC.

Jako odměna za vytěžení bloku bylo původně uděleno 50 BTC, aktuálně 12,5 BTC a odměna se snižuje na polovinu, každých 210 000 bloků. Celkový počet BTC, který bude dohromady vytvořen je 21 000 000 BTC. Vytěženo je přibližně 15 000 000 BTC.

Bitcoin je navržen tak, aby těžba jednoho bloku trvala průměrně 10 minut. Pro dosažení tohoto času se každých 2016 bloků mění obtížnost těžby.

Hodnota jednoho BTC je v současné době přibližně 800 amerických dolarů, historické maximum hodnoty jednoho BTC je více něž 1000$. Hodnota celé měny Bitcoin je v současné době více než 12 miliard dolarů, tímto číslem je v současnosti jednoznačně nejhodnotnější kryptoměnou světa. Hodnoty jsou převzaty z [21].

Počet uživatelů měny nelze, vzhledem k anonymitě sítě, reálně změřit, ale počet aktivních Bitcoinových adres je v současnosti zhruba 400 000.

2.3.2 Litecoin

Litecoin vznikl jako odnož Bitcoinu a jeho myšlenkou bylo vytvořit tzv. stříbro vedle zlatého Bitcoinu (z originálu *silver next to Bitcoin gold*). Jak už název napovídá, Litecoin vznikl jako kryptoměna která má být lehčí než Bitcoin, tedy kryptoměna která bude rychlejší.

Základním rozdílem oproti Bitcoinu je jiný hashovací algoritmus, namísto SHA256 použitého v Bitcoinu, Litecoin využívá algoritmus *Scrypt*, který byl vytvořen speciálně tak, aby jej nešlo efektivně realizovat v hardwaru. Toho docílí tím, že jednotlivé kroky algoritmu na sebe přímo navazují, výpočet jednoho hashe tedy nelze efektivně paralelizovat. Výpočet

algoritmu *Scrypt* také vyžaduje velké množství paměti oproti SHA256, takže na specializované hardwaru není možné počítat větší množství hashů současně, bez přidání velké a rychlé paměti.

2.3.2.1 Platby

V síti Litecoin je základní platební jednotka jeden Litecoin (zkratka LTC). Obdobně jako BTC i LTC je možné při platbě dělit na 8 desetinných míst.

Litecoin uděluje za každý blok odměnu 50 LTC, která se snižuje na polovinu každých 840 000 bloků. Celkově bude v síti Litecoin existovat 84 000 000 LTC. Tento limit je podstatně větší než u BTC, Litecoin vznikl mimo jiné s cílem, aby měny v oběhu bylo dostatek a bylo tak možné platit v celých číslech, která jsou pro lidi opticky příjemnější.

Těžba jednoho bloku Litecoinu trvá průměrně 2,5 minuty. Tento čas je mnohem kratší než u Bitcoinu, čímž se Litecoin snaží vytvořit dynamičtější měnu, u které není nutné čekat desítky minut, než bude možné transakci skutečně považovat za potvrzenou.

V současné době je hodnota jednoho LTC přibližně 3,8$. Hodnota celé sítě Litecoin je přibližně 189 milionů dolarů. Hodnoty jsou převzaty z [21].

2.3.3 DogeCoin

DogeCoin je měna odvozená od Litecoinu, původně vznikla pouze jako vtip, což se projevovalo například tím, že odměna za vytěžení bloku byla náhodná. Později, když se tato měna uchytila, tak byly některé principy předělány tak, aby mohla reálně fungovat.

Oproti Litecoinu nepřináší DogeCoin žádnou velkou technickou inovaci, princip obou sítí je téměř stejný.

2.3.3.1 Platby

V síti DogeCoin je platební jednotka jeden DOGE, i zde je možné tuto jednotku dělit na desetinné části, ale vzhledem k počtu a hodnotě DOGE to není příliš potřeba.

Odměna za vytěžení jednoho bloku byla původně 1 000 000 DOGE, tato odměna se snižuje na polovinu každých 100 000 bloků až do celkového počtu 600 000 bloků, pak bude odměna konstantní - 10 000 DOGE. Původně mělo být celkově vytěženo 100 miliard DOGE, ale později bylo rozhodnuto, že měna nebude mít horní hranici počtu DOGE a bude růst do nekonečna.

Těžba jednoho bloku trvá přibližně 1 minutu. Což, ve spojení s velkými odměnami na počátku existence měny, zajistilo rychlé vygenerování obrovského množství DOGE.

V současné době je vytěženo přes 31 miliard DOGE, přičemž tržní hodnota jednoho DOGE je přibližně 0,0002$. Celková hodnota měny je přibližně 21 milionů dolarů.

2.3.4 Bytecoin

Bytecoin je první z kryptoměn založených na protokolu CryptoNote (2.2.2). Jedná se o typickou kryptoměnu, která používá princip proof-of-work a hashovací algoritmus CryptoNight. Jakožto první kryptoměna na bázi CryptoNote, nepřináší žádné zvláštní technické inovace nebo modifikace.

Zásadní nevýhoda Bytecoinu je, že většina měny byla vytěžena úzkou komunitou lidí, v době blízké vzniku měny, což devalvuje její hodnotu, navíc naprostá většina peněz je již vytěžena, čímž klesá popularita této měny.

2.3.4.1 Platby

V síti Bytecoin je platební jednotka jeden Bytecoin (zkratka BNC), tuto měnu je možné dělit na 8 desetinných míst.

Odměna za vytěžení bloku se mění s každým blokem a to podle vzorce:

$$Reward = (Supply - A)/2^{18} \tag{2.1}$$

Kde supply je maximální množství peněz v síti, A je počet již vytěžených peněz. Maximální množství peněz v síti je 2^{64}, což je přibližně 184 miliard BNC.

Změna obtížnosti těžby se taktéž mění s každým vytěženým blokem. Přičemž doba těžby jednoho bloku by měla být přibližně 2 minuty. V současné době je odměna za vytěžení jednoho bloku zhruba 17 000 BNC. Celkově je vytěženo přibližně 179 miliard BNC.

Hodnota jednoho BNC je v současné době zhruba 0,000033 $. Hodnota celé měny Bytecoin je zhruba 5 milionů dolarů.

2.3.5 Monero

Monero je kryptoměna odvozená od Bytecoinu, a to takovým způsobem, že se jedná téměř o její klon. Oproti Bytecoinu by Monero mělo opravovat několik těchnických chyb, ale jinak žádnou zásadní technickou změnu nezavádí.

Zásadní rozdíl je však v popularitě měny, na rozdíl od Bytecoinu, Monero je od počátku těženo rovnoměrněji a transparentněji vůči všem uživatelům. V podstatě je Monero pouze Bytecoin, realizovaný lépe po stránce vztahu k uživatelům.

2.3.5.1 Platby

Platební jednotkou v Monero je XMR, při platbách je možné ji dělit na 8 desetinných míst. Odměna za vytěžení jednoho bloku se počítá stejným způsobem jako v Bytecoin (viz. 2.3.4.1).

Doba těžby bloku je stanovena na 1 minutu. Obtížnost těžby se mění po každém bloku tak, aby bylo dosaženo tohoto času. Odměna za vytěžení jednoho bloku je v současné době 6,4 XMR. V síti Monero bude celkově vytvořeno přibližně 11 milionů XMR.

Hodnota jednoho XMR je v současné době přibližně 0,9 $. Hodnota celé měny je zhruba 10 milionů dolarů.

2.3.6 Dash

Dash je další z kryptoměn. Název je odvozen od anglických slov "Digital cash". Dříve byl znám pod jmény XCoin a DarkCoin, nyní je znám jako Dash.

Zásadním změnou v Dashi je koncept tzv. **masternode**. Masternode je uzel sítě, fungující jako každý jiný běžný uzel, který navíc poskytuje ostatním uzlům speciální služby, jako například mixování transakcí.

Aby mohl uživatel provozovat masternode, musí jeho stroj splňovat několik podmínek, ty to po celou dobu fungování jako masternode. Pokud některá z podmínek přestane platit, přestane být masternodem. Tento souhrn podmínek se nazývá proof of Service a zajišťuje, že mezi masternode uzly budou pouze uzly schopné tuto činnost opravdu vykonávat. Mezi podmínky patří zejména, že uzel musí po celou dobu mít:

- 1000 DASH na účtu.

- Aktivní a dostupný stroj.
- Statickou IP adresu.

Dodržování dostatku peněz hlídá systém sám, dostupnost stroje hlídají ostatní masternode tím, že pravidelně posílají ping na ostatní uzly.

V síti existuje globální seznam masternode uzlů. Pokud chce klient využít služeb některého z nich, může zvolit kterýkoli ze seznamu. Provozovatel masternode je za své služby odměňován. Jakmile nějaký miner vytěží blok, všechny masternode si rozdělí zhruba 45 procent odměny za vytěžení bloku. Vzhledem k tomu, že provoz masternode stojí určité prostředky a odměna za jeho provozování se snižuje s jejich počtem, tak počet masternode reguluje samotný trh a není třeba jej omezovat technicky.

Další inovací Dashe je *zamykání transakce*. Jedná se o způsob, jak zabránit dvojímu utracení jedné transakce. Systém funguje tak, že každá transakce musí být zamčena, než může být použita v jedné konkretní platbě, přičemž po dobu zamčení nemůže být použita v žádné jiné platbě. Vykonaní samotné platby řeší skupina masternode, která je deterministicky vybrána.

Pro demonstraci systému předpokládejme, že uživatel dříve přijal transakci A, kterou chce využít jako vstup pro odchozí transakci B. Uživatel tedy vytvoří zámek na transakci A, ve kterém specifikuje, že tato transakce bude použita v transakci B. Tento zámek je následně rozdistribuován k masternode uzlům, řešícím danou transakci. Teprve jakmile je ode všech oslovených uzlů potvrzeno, že zámek přijali, je možné použít transakci A jako vstup pro transakci B. Následně je transakce B předána k zařazení do bloku a vytěžení.

Dash pro proof-of-work používá hashovací algoritmus X11, který je sám o sobě kombinací několika hashovacích algoritmů, a to konkrétně: blake, bmw, groestl, jh, keccak, skein, luffa, cubehash, shavite, simd, echo. Algoritmus X11 je poměrně rychlý na CPU i GPU, avšak kvůli složitosti je těžké jej implementovat v hardware, což je pro kryptoměny velmi žádaný efekt.

2.3.6.1 Platby

Základní platební jednotkou v Dashi je jeden DASH. Který je možné v platbách rozdělit až na 8 desetinných míst.

V současné době je odměna za vytěžení bloku přibližně 1,7 DASH. Odměna i obtížnost vytěžení bloku se mění po každém bloku. Vytěžení jednoho bloku trvá přibližně 2 minuty.

Hodnota jednoho DASH je v současné době přibližně 12,28 $. Vytěženo je přibližně 7 milionů DASH. Celkově bude vytěženo přibližně 22 milionů DASH. Celková hodnota měny je v současné době přibližně 86 milionů dolarů.

2.3.7 Ripple

Jak již bylo popsáno v 2.2.3, Ripple je síť, která se svojí filozofií výrazně odlišuje od ostatních kryptoměn. Převážně je totiž určen pro mezibankovní převody. Jsou zde určité uzly s rozdílnými funkcemi, není potřeba nijak dokazovat práci. Mimo jiné také umožňuje převody v různých měnách.

V sekci protokolu Ripple (viz. 2.2.3) byly popsány základní funkcionální prvky sítě, nyní se podíváme na prvky sítě, z hlediska instalovaného software:

- Ripple connect - Software, instalovaný na straně bank, který zajišťuje propojení interního systému banky do sítě Ripple. Tyto systémy připojují do sítě klienty.

- Ripple stream - Klientský software, který umožňuje správu účtu a transakcí.

2.3.7.1 Platby

Ripple má svoji nativní měnu XRP. Oproti jiným kryptoměnám, XRP se netěží, ale bylo vytvořeno v době vzniku sítě, kdy jej bylo 100 miliard. V současné době je cena jednoho XRP zhruba 0,0064$.

Ripple je možné používat jako most mezi jinými měnami, mezi kterými umí automaticky směňovat. V současné době, je v Ripple možné platit s reálnými měnami (např. USD, EUR, GBP a další), kryptoměnami (např. XRP, BTC, LTC a další) i některými komoditami (zlato, stříbro, platina). Konkrétní možnosti použitelných platidel nejsou pevné a záleží na tom, co podporují gateway a uživatelé.

Protože Ripple nepoužívá proof-of-work, není zde možné měnu nijak těžit. Současně však není třeba čekat s prováděním platby na vytěžení bloku a transakce tak může být zaúčtována téměř okamžitě, reálně provedení transakce trvá několik sekund.

Celková hodnota sítě Ripple je v současné době přibližně 237 milionů dolarů.

2.3.8 Ethereum

Jak bylo popsáno v 2.2.4, Ethereum se snaží být více než jen kryptoměnou. S lehkou nadsázkou lze říct, že Ethereum může fungovat jako framework pro další aplikace, které potřebují mít kryptoměnové platby jako svoji součást.

Další zvláštností je, že v průběhu verzí Etherea se má změnit systém pro určování dalších bloků v blockchainu. V současných verzích je použit běžný proof-of-work, který používá hashovací algoritmus Ethash. Ethash je algoritmus, který byl navržen speciálně pro Ethereum a je závislý na paměťových operacích, takže je obtížné jej realizovat v hardware.

2.3.8.1 Platby

V této části je popsán systém plateb, jak funguje přímo v Ethereu, ostatní aplikace, které existují nad Ethereem mohou fungovat jinak.

Základní platební jednotkou v Ethereu je jeden Ether (zkratka ETH), tuto měnu je možné dělit na 18 desetinných míst. Další platební jednotkou v Ethereu je gas, který není zůstatek na účtu, ale používá se jako poplatek při provádění transakcí, přičemž spotřebovaný gas se nakonec platí v ETH. Kurz pro převod ETH na gas určuje odesilatel transakce.

Nový blok by měl být přidán do blockchainu každých 12 sekund. Odměna za vytěžení bloku je staticky nastavena na 5 ETH, navíc miner dostane od odesilatele transakce proplacený veškerý spotřebovaný gas. Obtížnost je přepočítána s každým vytěženým blokem.

Jelikož je odměna za vytěžení bloku statická, není v Ethereu žádné maximální množství peněz, které vznikne.

Cena jednoho ETH je zhruba 9,75 $. Zatím je vytěženo přibližně 87 milionů ETH. Aktuálně je Ethereum druhou nejcennější kryptoměnou s celkovou hodnotou přes 850 milionů dolarů.

2.4 Srovnání kryptoměn

V této kapitole je uvedeno srovnání parametrů jednotlivých kryptoměn, popsaných v předchozí kapitole. Jsou zde jak technické tak finanční vlastnosti jednotlivých kryptoměn.

2.4.1 Technické srovnání

Následující tabulka zobrazuje rozdíly mezi jednotlivými měnami.

Název	Měna	Proof-of-work algoritmus	Počet soukromých klíčů pro n transakcí	Doba provedení platby	Vytěžení bloku potřebuje zejména
Bitcoin	BTC	double-SHA256	1	10 minut	výpočetní výkon
Litecoin	LTC	Scrypt	1	2,5 minuty	paměť
DogeCoin	DOGE	Scrypt	1	1 minuta	paměť
Bytecoin	BNC	CryptoNight	n	1 minuta	paměť
Monero	XMR	CryptoNight	n	1 minuta	paměť
Dash	DASH	X11	n	2 minuty	výpočetní výkon
Ripple	XRP	-	1	jednotky sekund	konektivita
Ethereum	ETH	Ethash	1	12 sekund	paměť

2.4.2 Finanční srovnání

Následující tabulky ukazují tržní hodnoty jednotlivých měn a jiné parametry, důležité z obchodního hlediska, údaje jsou aktuální k lednu 2017. Aktuální hodnoty jsou k získání z [21].

Název	Měna	Hodnota měny	Maximální množství měny	Doba provedení platby
Bitcoin	BTC	801 $	21 000 000	10 minut
Litecoin	LTC	3,85 $	84 000 000	2,5 minuty
DogeCoin	DOGE	0,0002 $	Neomezeně	1 minuta
Bytecoin	BNC	0,000042 $	184 miliard	1 minuta
Monero	XMR	11,29 $	11 milionů	1 minuta
Dash	DASH	12,28 $	22 milionů	2 minuty
Ripple	XRP	0,006 $	100 miliard	jednotky sekund
Ethereum	ETH	9,75 $	Neomezeně	12 sekund

Název	Odměna za vytěžení bloku	interval snížení odměny	Platby napříč měnami	Hodnota celé sítě
Bitcoin	12,5 BTC	210 000	ne	12,9 miliard $
Litecoin	50 LTC	840 000	ne	189 milionů $
DogeCoin	10 000 DOGE	nikdy	ne	22 milionů $
Bytecoin	17 000 BNC	1	ne	7,5 milionů $
Monero	6,4 XMR	1	ne	155 milionů $
Dash	1,7 DSH	1	ne	86 milionů $
Ripple	-	-	ano	237 milionů $
Ethereum	5 ETH + poplatek	-	ne	856 milionů $

Kapitola 3

Existující nástroj a plány na jeho vylepšení

V rámci diplomové práce Tomáše Drozdy [1] vznikl nástroj pro forenzní analýzu kryptoměn Bitcoin a Litecoin nazvaný *CryptoCurrency Analyzer* (dále jen CCA). Primárně se zaměřuje pouze na síť Bitcoin. Tento nástroj umožňuje načíst a procházet obsah blockchainu a umí trasovat jednoduché vazby mezi jednotlivými transakcemi. Dokáže také evidovat existující adresy a přiřazovat je určitým uživatelům. Ve výsledku je v CCA možné zjistit, jaké účty vlastní konkrétní uživatel a jakými na nich disponuje prostředky. Samostatnou problematikou je zjišťování reálných identit uživatelů a jejich propojení s Bitcoinovými adresami, ale to není předmětem této práce. CCA je určen zejména pro bezpečnostní složky, kvůli možnosti trasování kryptoměnových peněz získaných trestnou činností.

3.1 CCA a jeho optimalizace

CCA je implementován v jazyce PHP a funguje primárně jako webová aplikace. K webové části aplikace, je přidána konzolová část, která slouží pro řízení načítání obsahu blockchainu do aplikace. Samotná data jsou uložena v databázi. Největším problémem CCA je pomalý import dat do databáze a následně jejich pomalé procházení. Problém spočívá v použitých technologiích a architektuře aplikace. Tyto problémy jsou rozepsány v následujících sekcích.

3.1.1 Úložiště dat

Jako původní úložiště dat byl v CCA použit databázový systém *MongoDB*. Jedná se o NoSQL, dokumentovou databázi. MongoDB sama o sobě pracuje s daty velmi rychle, ale umožňuje data ukládat pouze jako kolekce objektů, které mezi sebou nemají žádné vazby. Tento systém tedy neumožňuje nijak provázat objekty na úrovni databáze. Veškeré spojování objektů je tedy nutné dělat přímo v aplikaci, což je problematické např. při trasování plateb, kdy je potřeba najít objekt identifikovaný podle hashe. V případě, kdy je v aplikaci velké množství dat, tento přístup způsobí značné zpomalení.

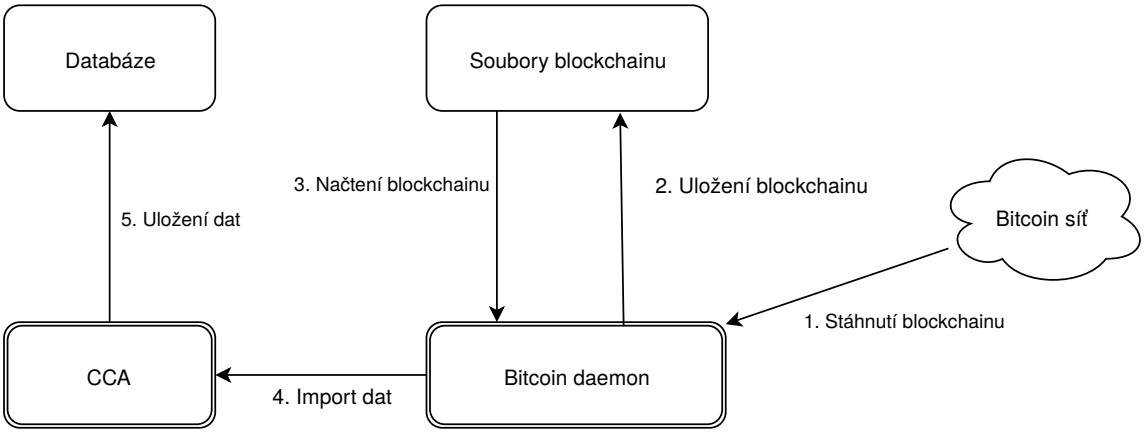
Náhradou za MongoDB se stane databázový systém *Neo4j*. Jedná se o nestrukturovanou, NoSQL, grafovou databázi, která jednotlivé záznamy ukládá jako uzly grafu. Neo4j je přímo optimalizovaná na hledání vazeb mezi uzly a dohledávání takto spojených uzlů. Tato změna databázového systému by měla výrazně zrychlit trasování transakcí. Nedostatkem Neo4j je, že neumožňuje uzly žádným způsobem nativně rozřazovat do skupin, nebo jinak třídit (jako

je možné např. přiřazení do jednotlivých tabulek v SQL databázích). Tento nedostatek CCA řeší pomocí dalšího atributu uloženého uzlu, který určuje typ záznamu.

3.1.2 Import dat z blockchainu

V původním CCA sloužil jako zdroj dat *Bitcoin daemon*, tedy klient sítě Bitcoin. Tento klient musí být nainstalován a CCA k němu musí mít přístup. K tomuto klientovi se CCA připojoval přes *Bitcoin RPC API*, přes které stahoval obsah blockchainu. Tento přístup je problematický z pohledu rychlosti importu dat. Bitcoin daemon běží jako samostatná aplikace, která sama o sobě přináší do zpracování zpoždění. Další problém je, že s Bitcoin daemonem se komunikuje přes počítačovou síť a tedy každý požadavek na načtení bloku, musí jít přes síťovou vrstvu, v horším případě i přes síť na jiný stroj.

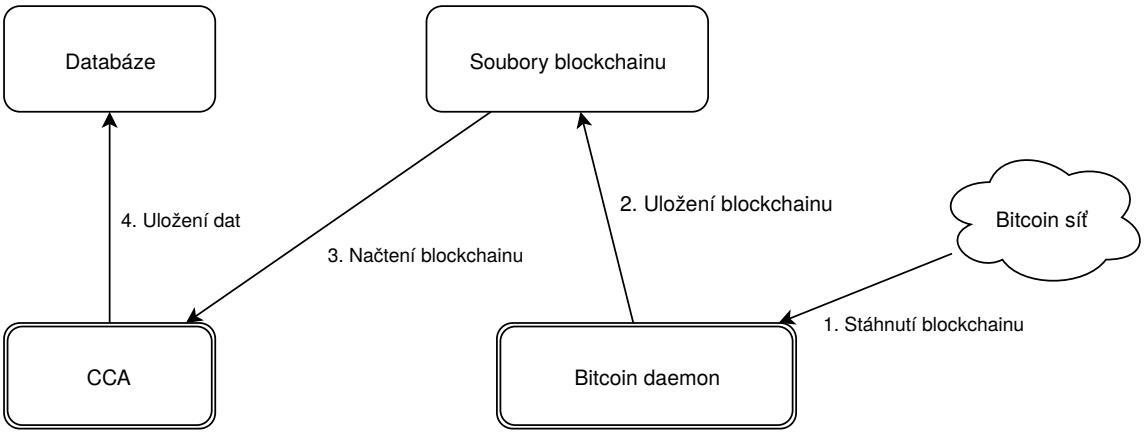
Všechny kroky, které byly nutné pro import dat, ukazuje následující diagram:



Obrázek 3.1: Schéma importu starým způsobem

Při optimalizaci importu je snaha zejména o jeho zrychlení, a pokud by to bylo možné, tak i zjednodušení a eliminací věcí, které nejsou v systému naprosto nutné. Vzhledem k tomu, že CCA má pracovat s aktuálními daty, tak v systému musí zůstat Bitcoin daemon aby zprostředkovával stahování aktuálních dat ze sítě Bitcoin. Bitcoin deamona však lze vypustit z procesu importu dat z lokální kopie blockchainu do databáze CCA, jelikož CCA může přímo číst a parsovat soubory blockchainu.

Způsob importu dat po optimalizaci znázorňuje následující diagram:



Obrázek 3.2: Schéma importu novým způsobem

Výhodou tohoto řešení je výrazně rychlejší import dat a jednoduší nasazování aplikace, jelikož není třeba zprovozňovat komunikaci mezi CCA a Bitcoin daemonem.

Nevýhodou tohoto řešení je, že CCA musí mít přístup k souborům blockchainu, nejlépe aby byl blockchain uložen na stejném stroji jako CCA. Oproti tomu, při starém způsobu importu mohl být Bitcoin daemon umístěn na odděleném stroji. Další nevýhodou nového řešení je ztráta informací, které k blockchainu dokáže dodat Bitcoin daemon. Jelikož blockchain Bitcoinu byl navržen tak, aby obsahoval co nejmenší množství dat a tedy jen data nezbytně nutná, některé potřebné informace je třeba dopočítávat. Mezi takovéto informace patří např. hash bloku, výška (*height*) bloku, ID transakce, apod.

3.2 Rozšíření CCA

V této sekci jsou popsány informace, které lze získat z blockchainu kryptoměny Bitcoin, a způsoby, jak tyto informace navázat na informace známé z reálného světa. Následně jsou zde popsány možnosti, jak tyto informace využít.

V další části sekce jsou popsány konkrétní plánované rozšíření nástroje CCA, které budou implementovány v rámci této práce.

3.2.1 Získatelné informace z blockchainu

Samotný blockchain je navržen tak, aby obsahoval naprosté minimum dat, avšak některé další informace je možné z těchto dat dopočítat. Tato část práce popisuje pouze zajímavé informace, které lze prakticky využít, samotný CCA však zobrazuje všechny dostupné informace.

3.2.1.1 Blok

Blok kromě transakcí obsahuje převážně servisní informace nutné pro funkčnost sítě. Mezi tyto informace patří například: verze bloku, hash předchozího bloku, hash kořenu stromu transakcí a další. Dopočítat je možné například hash bloku. Tyto informace jsou důležité pro funkčnost sítě, avšak nejsou příliš důležité pro analýzu.

Mezi analyticky zajímavé informace patří:

Čas vytvoření bloku - Udává přibližný čas, kdy byl blok vytvořen.

Hash bloku - Hash bloku se používá jako jeden z možných identifikátorů bloku.

Height - Výška vyjadřuje pořadové číslo bloku v blockchainu a je dalším z možných identifikátorů bloku.

3.2.1.2 Transakce

Transakce obsahují převážně informace ohledně převodu finančních prostředků. Samotná transakce obsahuje 0-n vstupů a 1-n výstupů. Z těchto dat je možné určit:

- Které adresy byly použity jako zdroje peněz pro tuto transakci.
- Na které adresy byly peníze převedeny.
- Kolik peněz bylo převedeno na které adresy.
- Z rozdílu množství vstupních a výstupních prostředků je možné určit velikost poplatku.
- Hash transakce (Txid), který slouží jako identifikátor transakce.

Z bloku, uvnitř kterého je tato transakce uložena, je možné získat čas vzniku. Blok vznikl vždy později než transakce, ale velmi pravděpodobně to není později více než o jednu hodinu.

Speciálním případem transakce je tzv. *coinbase* transakce. Tyto transakce slouží pro emitování nových Bitcoinů do sítě. V každém vytěženém bloku je právě jedna a příjemcem nově vzniklých peněz jsou mineři, kteří blok vytěžili. Coinbase transakce nemá žádné vstupy a na výstupu jsou peníze, které v součtu tvoří odměnu za vytěžení bloku (v současné době 12,5 BTC) plus poplatky z transakcí.

3.2.2 Korelace obsahu blockchainu s dostupnými informacemi

Tato sekce ukazuje, jak je možné prakticky využít informace, popsané v předchozí sekci.

Ohledně informací získaných z bloku, je zajímavý pouze čas vytvoření bloku, ze které lze odvodit, že všechny transakce uvnitř tohoto bloku vznikly v přibližně stejnou dobu jako blok, nebo alespoň byly v této době zveřejněny. Lze tedy usuzovat, že v době vytvoření bloku, byli autoři vložených transakcí připojeni k internetu. Ostatní informace nejsou analyticky zajímavé.

Oproti tomu transakce obsahují mnohem více analyzovatelných informací.

Z každého vstupu a výstupu transakce je možné vyčíst adresu Bitcoinové peněženky. Takto je možné zjistit, že určité adresy existují a můžeme je dohledávat na internetu nebo jiných zdrojích. Některé instituce zveřejňují svoje Bitcoinové adresy a je teoreticky možné zjistit reálnou identitu účastníků této transakce.

Vzhledem k tomu, že jednu transakci obvykle vytváří jedna entita, pak pravděpodobně všechny vstupy, použité v jedné transakci, pocházejí z adres, které vlastní jedna fyzická osoba nebo organizace. Při analýze celého blockchainu je takto teoreticky možné zjistit, které adresy mají společného vlastníka.

Obsahem výstupů transakce je množství převedených BTC, z dostupných kurzů mezi měnami je možné zjistit, kolik reálných prostředků bylo touto transakcí převedeno.

Identifikátor transakce (txid) se používá pro identifikování a odkazování se na danou transakci.

3.2.3 Trasování a vizualizace transakcí

Plánovaným rozšířením CCA je zejména trasování BTC napříč transakcemi a vizualizace takto získaných informací. Cílem je vytvořit systém, ve kterém si uživatel může zobrazit určitou transakci a bude schopen snadno zjistit, odkud pocházejí prostředky použité v dané transakci a kam byly prostředky dále posílány.

Aktuálně CCA umí vizualizovat pouze jednu transakci a její výstupy, nedokáže zobrazit vazby na jiné transakce, ani trasovat původ peněz vstupujících do transakce.

Kapitola 4

Implementované změny v CCA

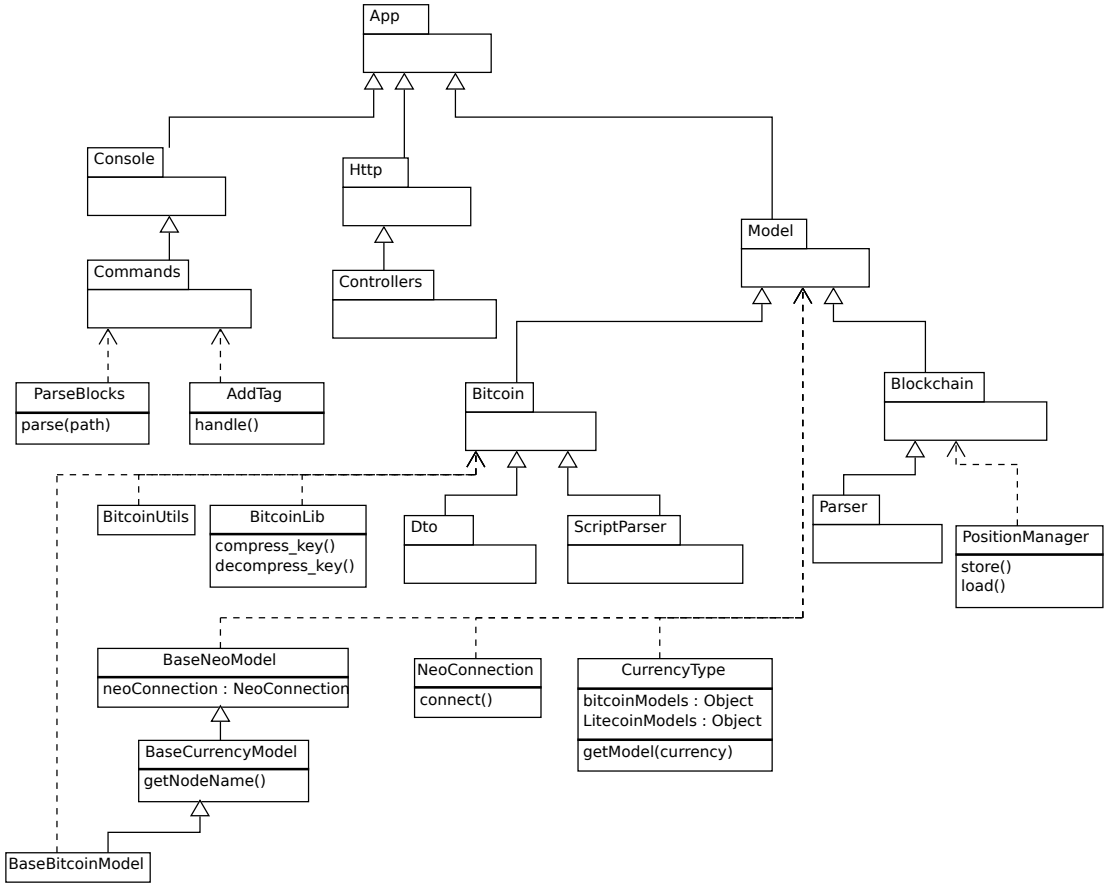
V rámci této práce došlo v CCA k mnoha změnám a rozšíření. V této sekci je popsána architektura aplikace a hlavní změny, které byly provedeny v rámci této práce.

4.1 Architektura aplikace

CCA je vytvořen v PHP frameworku Laravel, který používá běžný návrhový vzor MVC. V této sekci jsou popsány důležité části vlastní aplikace a kde je najít ve zdrojovém kódu. Diagram 4.1 zobrazuje vybrané části struktury namespace a tříd aplikace. Diagram není zdaleka úplný, obsahuje pouze hlavní části struktury CCA, je určen pouze pro základní orientaci. Namespace uvedené v diagramu odpovídají adresářové struktuře aplikace.

Zde následuje stručný popis namespace a tříd, uvedených v diagramu:

- Console/Commands - Tento namespace obsahuje implementaci příkazů, které je možné spustit pomocí konzolového rozhraní a skriptu *artisan*, který je součástí frameworku Laravel.
- Console/Commands/ParseBlocks - Tato třída realizuje konzolový příkaz pro parsování bloků z blockchainu. Řídí parsování bloků a ukládání informací do databáze
- Console/Commands/AddTag - Tato třída je vytvořena pro usnadnění budoucí integrace jiných nástrojů, které budou dodávat značky k Bitcoinovým adresám.
- Http/Controllers - Tento namespace obsahuje třídy, které obsluhují samotné HTTP požadavky.
- Model - Tento namespace obsahuje veškeré třídy, které realizují aplikační logiku.
- Model/Blockchain - Zde je umístěn parser blockchainu a třídy, pomáhající při parsování blockchainu.
- Model/Blockchain/Parser - Zde je samotný parser blockchainu.
- Model/Blockchain/PositionManager - Tato třída slouží pro persistentní ukládání aktuální pozice v blockchainu.
- Model/BaseNeoModel - Tato třída implementuje konstrukci veškerých dotazů, posílaných do databáze.



Obrázek 4.1: Abstraktní architektura CCA

- Model/NeoConnection - Tato třída spravuje spojení s Neo4j databází.
- Model/CurrencyType - Tato třída uchovává instance modelů, které pracují s daty BTC a LTC. Zbytku aplikace předává příslušné modely, podle požadované kryptoměny.
- Model/Bitcoin - Tento namespace obsahuje modely pro práci s BTC entitami.
- Model/Bitcoin/Dto - Tento namespace obsahuje třídy pro přenos dat jednotlivých entit.
- Model/Bitcoin/ScriptParser - Tento namespace obsahuje třídy, které zpracovávají *scriptPubkey* a *scriptSig* z blockchainu.
- Model/Bitcoin/BitcoinLib - Tato třída implementuje matematické operace s BTC daty.
- Model/Bitcoin/BitcoinUtils - Tato třída implementuje pomocné funkce pro práci s BTC daty.
- Model/Bitcoin/BaseBitcoinModel - Rodičovská třída pro všechny třídy, které pracují s BTC entitami.

4.2 Import blockchainu

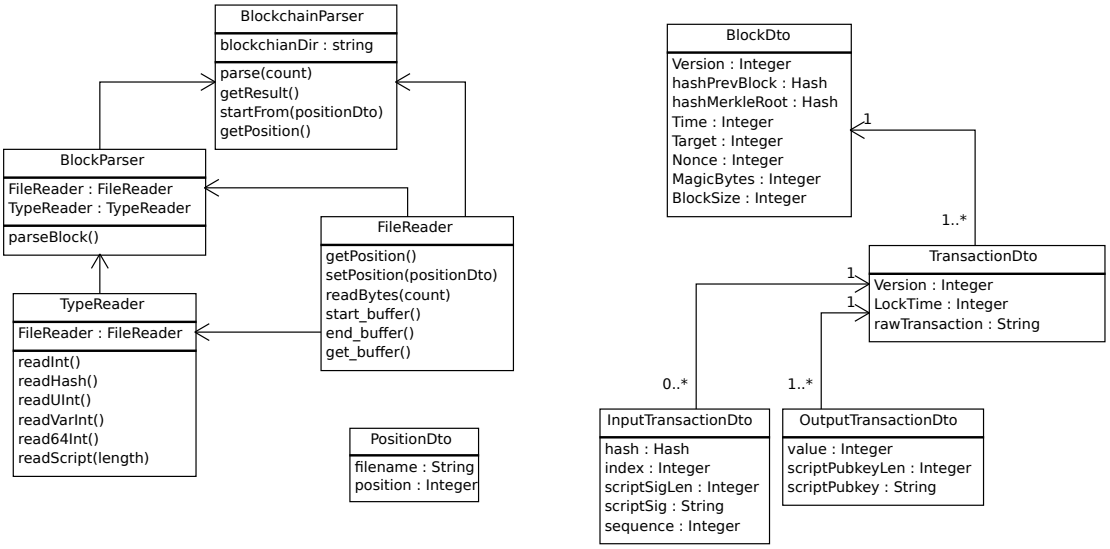
4.2.1 Blockchain parser

V rámci této práce vznikl parser blockchainu pro měnu Bitcoin, napsaný v jazyce PHP, který byl následně integrován do CCA. Snahou bylo co nejvíce přiblížit načítání dat k samotné aplikaci CCA tak, aby nebyla potřeba žádná komunikace s dalšími programy. Navíc je parser napsán tak, aby jej bylo možné použít jako knihovnu a snadno jej zaintegrovat do jiné aplikace. Parser se nachází v namespace *app/Model/Blockchain/Parser*.

Parser pouze načítá data, která jsou uložena v blockchainu, nedopočítává k nim žádné další informace. Strukturu blockchainu, včetně významu jednotlivých polí, je možné nalézt v [2].

Pro správnou funkci napříč více běhy aplikace, parser potřebuje ukládat pozici, na které přestal v předchozím běhu. V CCA se tato pozice ukládá do databáze, do uzlu s označením *BTC_POSITION*. Ukládání a načítání pozice obstarává třída *PositionManager*.

Strukturu parseru zachycuje diagram tříd 4.2.



Obrázek 4.2: Architektura parseru blockchainu

4.2.2 Zpracování blockchainu

Import a zpracování blockchainu zajišťuje třída *app/Console/Commands/ParseBlocks*, řídí komunikaci s parserem, dopočítávání potřebných dat a ukládání dat do databáze. Dopočítává se Hash a výška bloku, identifikátor transakce, adresy které participují v transakcích a další.

Bloky v blockchainu jsou spojeny do seznamu, kde každý blok obsahuje hash hlavičky předchozího bloku. Bloky však v blockchainu nemusí být fyzicky uloženy tak, jak jdou logicky za sebou. V případě že se načte blok, jehož předchůdce ještě není uložen, je takový blok umístěn do dočasného uzlu v databázi, kde zůstane, dokud nejsou zpracovány všechny uzly, které jej předchází. Vždy při uložení nového bloku do databáze se kontroluje jestli není uložen takovýto nezpracovaný blok, navazující na právě uložený. Pokud ano, pak je vyjmut z dočasného uzlu a zpracován běžným způsobem.

V rámci zpracování je třeba parsovat *ScriptPubkey* a *ScriptSig*, jedná se skripty v jazyce *Script*, které řídí zacházení s platbami. *ScriptPubkey* je položka v jednom výstupu transakce, která specifikuje, za jakých podmínek je možné tento výstup utratit. *ScriptSig* je položka ve vstupu transakce, která obsahuje taková data, se kterými je možné splnit *ScriptPubkey* použitého výstupu. CCA umí zpracovat pouze běžně používané skripty, jejich popis je umístěn v B. *Script* sice není výpočetně úplný, ale poskytuje mnoho způsobů, jak ztížit identifikaci adresáta platby. Popis *Scriptu* je možné nalézt v [24].

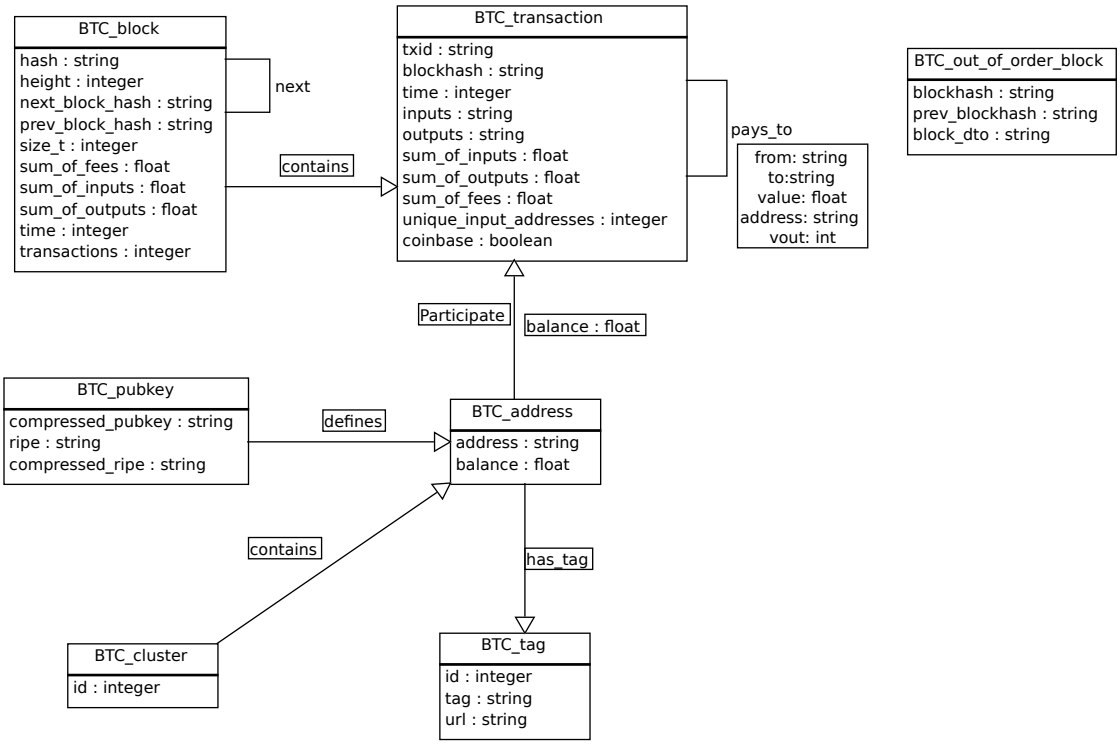
4.3 Databáze Neo4j

Jedním z hlavních cílů této práce je změna databázového systému z MongoDB na Neo4j. Neo4j je grafová databáze která by měla rychleji pracovat s větším množstvím uzlů a hledáním vazeb mezi nimi.

V implementaci CCA byla velmi slabá abstrakce databázové vrstvy, tedy značná část aplikace přímo používala služby databázové vrstvy. Oproti původním předpokladům pak změna databázového systému vyžadovala značný zásah do architektury aplikace.

4.3.1 Schéma databáze

V rámci změn databáze se změnilo schéma, aby bylo vhodnější pro grafovou databázi a reflektovalo požadované změny v aplikaci. Současné schéma zobrazuje diagram 4.3. Neo4j má oproti relačním databázím drobnou výhodu v tom, že umožňuje ukládat atributy i do vazeb mezi uzly. Neo4j také nemá striktně definované schéma, tak jako např. SQL databáze, ukládání správných atributů je nutné řešit na aplikační úrovni.



Obrázek 4.3: Schéma databáze

4.3.1.1 Popis entit

V této sekci jsou popsány jednotlivé entity uložené v databázi a význam jejich atributů.

BTC_block ukládá informace o jednom bloku Bitcoinu, který je nyní uložen v databázi a zobrazuje se v CCA. Jeho položky obsahují hodnoty:

hash - Hash hlavičky bloku, slouží jako identifikátor.

height - Výška bloku, reprezentuje pořadové číslo bloku.

next_block_hash - Hash bloku, který následuje za tímto blokem.

prev_block_hash - Hash předchozího bloku.

size_t - Počet bajtů, které blok zabírá v blockchainu.

sum_of_fees - Suma poplatků všech transakcí v bloku.

sum_of_inputs - Suma všech vstupů transakcí v bloku.

sum_of_outputs - Suma všech výstupů transakcí v bloku.

time - Unix timestamp, čas kdy byl tento blok zařazen do blockchainu.

transactions - Počet transakcí v bloku.

Jeden uzel bloku má vazby na uzel následujícího bloku a na všechny transakce, které do bloku spadají.

BTC_transaction reprezentuje jednu transakci provedenou v Bitcoinu. Význam jejich políček je:

txid - Hash transakce, slouží jako její identifikátor.

blockhash - Hash bloku, ve kterém je transakce uložená.

time - Čas vzniku transakce.

inputs - Serializované pole vstupů transakce.

outputs - Serializované pole výstupů transakce.

sum_of_inputs - Suma hodnot všech vstupů transakce.

sum_of_outputs - Suma hodnot všech výstupů transakce.

sum_of_fees - Suma všech poplatků v transakci.

unique_input_addresses - Počet všech unikátních adres, použitých na vstupu transakce.

coinbase - Identifikuje, zdali se jedná o coinbase transakci nebo ne.

BTC_out_of_order_block ukládá bloky, které byly zpracovány mimo pořadí a ještě nemohou být zařazeny mezi zpracované bloky, protože chybí bloky jim předcházející. Obsahuje položky:

blockhash - Hash bloku.

prev_blockhash - Hash předchozího bloku. Jakmile bude předchozí blok zpracován, je možné zpracovat i tento blok.

block_dto - Serializovaný objekt, který uchovává obsah bloku.

BTC_address obsahuje informace o jedné adrese v Bitcoinu. Přímo v uzlu jsou uloženy pouze informace:

address - Bitcoinová adresa.

balance - aktuální použitelný zůstatek, na dané adrese.

Kromě těchto atributů, má vazby na uzly reprezentující transakce, kde atributem vazby je množství peněz, které adresa v transakci získala nebo ztratila.

BTC_pubkey obsahuje informace o jednom známém veřejném klíči. Jelikož jeden veřejný klíč může definovat více Bitcoinových adres, jsou k těmto uzlům připojeny vazby na korespondující adresy. Samotný uzel obsahuje atributy:

compressed_pubkey - Komprimovaná verze veřejného klíče. Komprese je bezztrátová, takže je možné znovu získat i nekomprimovaný klíč.

ripe - Obsahuje nekomprimovaný veřejný klíč, po projití hashovacími algoritmy SHA256 a RIPEMD160. Tento hash se může vyskytovat přímo v blockchainu.

compressed_ripe - Hashovaná verze komprimovaného veřejného klíče.

Uzel **BTC_cluster** reprezentuje jednu entitu, která vlastní několik samostatných adres. Samotný uzel obsahuje pouze identifikátor, k uzlu jsou vazbou připojeny adresy, které patří dané entitě.

BTC_tag umožňuje uložit označení dané adresy. Jedná se o informace o adrese, získané z externích zdrojů. Každý tag může být přiřazen k jedné nebo více adresám. Tag obsahuje informace:

id - Jednoznačný identifikátor.

tag - Popis tagu, např. jméno vlastníka adresy.

url - URL adresa, kde byl tag získán.

4.3.1.2 Indexy

Pro zrychlení vyhledávání, jsou atributy některých uzlů vybaveny indexem. Indexy nezaručují jednoznačnost, slouží pouze pro zrychlení vyhledávání. Jedná se o:

BTC_block - atributy *hash* a *height*.

BTC_transaction - atribut *txid*.

BTC_address - atribut *address*.

BTC_pubkey - atribut *compressed_pubkey*.

4.4 Metody clusterizace

V rámci clusterizace se zjišťuje, které adresy mají společného vlastníka, případně jsou pod jednotnou správou. Tyto adresy se spojují do tzv. clusterů. CCA používá pro clusterizaci pouze informace dostupné z blockchainu. CCA clusterizuje adresy podle dvou věcí: vstupních adres transakce a veřejného klíče adres. Obě metody jsou popsané v následujících sekcích.

Clusterizačních metod existuje více, ale ostatní používají i informace, které nemá CCA k dispozici, jako např. chování konkrétních Bitcoinových klientů nebo interní informace Bitcoinových služeb. Z toho důvodu CCA implementuje pouze tyto dvě základní metody.

V CCA může jedna adresa být maximálně v jednom clusteru. V případě že má dojít ke spojení dvou adres, přičemž každá je obsažena v jiném clusteru, dojde ke spojení obou clusterů. Operace s clustery jsou implementovány ve třídě *BitcoinClusterModel*.

4.4.1 Clusterizace podle vstupů transakce

Tato clusterizační metoda vychází z předpokladu, že pokud se na vstupu jedné transakce objeví více adres, pak pravděpodobně tyto adresy mají stejného vlastníka. Na to, aby jedna transakce mohla použít více vstupů, je nutné tyto vstupy vytvořit a spojit, ještě před uveřejněním transakce v blockchainu. Proto můžeme předpokládat, že pokud má jedna transakce vstupy z více adres, pak majitelé těchto adres museli být ve spojení ještě před vytvořením transakce. Tato metoda shlukuje všechny adresy, které jsou na vstupu jedné transakce.

4.4.2 Clusterizace podle veřejného klíče

Tato clusterizační metoda využívá toho, že z jednoho veřejného klíče je možné vytvořit různé adresy. Protože jsou však adresy vázány na stejný veřejný klíč, pak mají zcela jistě stejného vlastníka. Při zjištění nového veřejného klíče CCA vypočte tyto možné adresy a uloží je do společného clusteru.

Bitcoin jako algoritmus pro podepisování používá eliptické křivky, u kterých platí, že veřejný klíč je možné bezztrátově komprimovat na zhruba poloviční velikost. Tím vzniknou dva, na první pohled nezávislé, klíče: komprimovaný a nekomprimovaný. Přičemž v Bitcoinu je možné plnohodnotně používat obě varianty klíče.

Dále jsou možnosti jak vytvořit více adres z jednoho klíče, zejména použitím jiné metody platby. Metody P2PK (Pay to Pubkey) a P2SH (Pay to ScriptHash) produkují adresy ze stejného veřejného klíče, avšak výsledné adresy není možné spojit (o tvorbě adres se můžete dočíst více v [2]). Jak bylo zmíněno v sekci 4.2.2, ve Scriptu je možné tvořit komplikované způsoby identifikace adresáta platby, metoda P2SH tak může produkovat teoreticky nekonečně mnoho adres, patřící stejnému veřejnému klíči.

4.5 Vizualizace transakcí

V rámci této práce se v CCA zcela změnila vizualizace transakcí. Oproti původní verzi, která využívala knihovnu *d3*, je nově pro vykreslování použita javascriptová knihovna *vis* [23]. V obou verzích je nejdřív zvolena jedna konkrétní transakce, která se zobrazí a od ní se zobrazují další transakce. Nová verze umožňuje zobrazovat transakce následující i předcházející vybrané transakci, také umožňuje zobrazovat transakce, které jsou ve vztahu s libovolnou, již zobrazenou transakcí.

Vizualizace nově zobrazuje graf, ve kterém uzly reprezentují transakce a hrany reprezentují adresy, na kterých byly Bitcoiny uloženy mezi transakcemi. Příklady vizualizace jsou v příloze C. Vizualizace je schválně orientována na transakce nikoli na adresy, protože cílem CCA je trasovat Bitcoiny. Pro sledování toku Bitcoinů je přehlednější sledovat transakce chronologicky.

4.6 Nové obrazovky

Tím, že CCA nově parsuje blockchain, se dostává k více technickým informacím o platbách a adresách v Bitcoinu. Jedná se o informace, které jsou nutné pro provoz Bitcoinu, ale pro uživatele nejsou podstatné. V případě detailního zkoumání určitých transakcí však mohou hrát důležitou roli. CCA nově zobrazuje i některého z těchto informací, zejména detaily vstupů a výstupů transakcí a veřejné klíče.

4.6.1 Detail vstupů a výstupů transakcí

Mezi nové obrazovky detaily vstupů a výstupů transakcí. Příklady těchto obrazovek lze najít v příloze D. Tyto obrazovky ukazují zejména Script, řídicí daný vstup nebo výstup, použité veřejné klíče, podpisy, způsoby platby apod. Díky zobrazení Scriptu, který řídí danou transakci, je možné provézt manuální analýzu nestandardních použití Scriptu.

Zobrazení veřejných klíčů a metody platby umožňuje také zjistit, kdo může použít výstup multisign transakce (transakce která vyžaduje M z N podpisů).

4.6.2 Veřejné klíče

Na obrazovce o veřejném klíči lze nalézt informace jak daný klíč vypadá v různých reprezentacích a jaké známé adresy k němu přísluší. Zobrazuje se veřejný klíč v komprimovaném i nekomprimovaném tvaru a oba tyto klíče po hashování algoritmy SHA256 a RIPEMD160. Ve Scriptu se běžně veřejný klíč používá v některém ze zobrazených formátů, při ruční analýze Scriptu je možné takto dohledat odpovídající veřejný klíč a odpovídající adresy.

Kapitola 5

Praktické použití CCA

Tato kapitola popisuje různé aspekty praktického použití nástroje CCA. Obsahuje stručný návod na zprovoznění CCA, popisuje možné problémy při jeho používání. Dále je zde rozebrán jeden konkrétní příklad analýzy transakce. Nakonec je zde popsána možnost využití CCA pro jiné kryptoměny.

5.1 Nasazení a ovládání CCA

Tato sekce obsahuje stručný návod na nasazení, spuštění a ovládání CCA. CCA pro běh vyžaduje pouze webový server s nainstalovaným PHP ve verzi 5.5.9 nebo vyšší a databázi neo4j verze 3 nebo vyšší. Pro parsování blockchainu je navíc třeba, aby na hostitelském stroji byly k dispozici soubory blockchainu.

Na nasazení aplikace je třeba na webovém serveru vytvořit pro CCA vlastní virtual host, do které se zkopírují zdrojové soubory CCA. Document root aplikace je třeba nastavit do adresáře *public* ze zdrojových kódu CCA. Následně je třeba vytvořit soubor *.env* (jako šablona slouží soubor *.env.example*) a upravit v něm hodnotu *NEO4J_DB*, na připojení do databáze. Tato položka by měla být vyplněna ve formátu:

```
NEO4J_DB=http://<neo4j login>:<neo4j heslo>@<adresa>:<port>
```

Tím by CCA měl být nasazen a mělo by fungovat jeho webové rozhraní. Dalším krokem je naplnění databáze. Následující příkazy se vykonávají přes konzoli v kořenovém adresáři CCA.

Nejdřív je třeba inicializovat databázi, což se provede příkazem:

```
php artisan blocks:parse --initDb
```

Tento příkaz vytvoří v databázi indexy a jiné potřebné věci. Poté je možné zajít parsovat blockchain. Následující příkaz parsuje a uloží do databáze požadované množství bloků:

```
php artisan blocks:parse <cesta k~blockchainu>
--number_of_blocks=<počet bloků, které se mají zpracovat>
```

Pokud se příkaz spustí opakovaně, pak při dalším běhu bude parsování pokračovat od místa, kde v předchozím běhu skončilo.

Pro vymazání blockchainu z databáze je možné použít příkaz:

```
php artisan blocks:parse --clear
```

Tento příkaz však v databázi zanechá nalezené adresy a clustery. Pokud vyžadujeme smazání i těchto, je třeba využít příkaz:

```
php artisan blocks:parse --totalClear
```

Tento příkaz z databáze vymaže všechny uzly.

5.2 Výkonnost CCA

Tato sekce popisuje možné výkonnostní problémy CCA a použitých technologií.

5.2.1 Parsování blockchainu

Při parsování blockchainu je kladen důraz na rychlost čtení z disku, zpracování dat a uložení dat do databáze. Pro reálnou analýzu je třeba rozparsovat celý blockchain Bitcoinu, který zabírá v současné době více než 100GB. Při zpracování dat je časově nejnáročnější clusterziace adres, při které se dohledávají různé adresy a spojují se do clusterů.

Dalším problémem je prostor pro uložení zpracovaných dat. Blockchain zabírá více než 100GB, přičemž obsahuje jen nezbytné minimum informací a to v úsporném formátu. Databáze ukládající zpracované údaje tedy bude zabírat více datového prostoru.

Kvůli náročnosti clusterizace jsem se rozhodl vyzkoušet parsování blockchainu se zapnutou i s vypnutou clusterizací.

Při zapnuté clusterizaci se rychlost zpracování ukázala jako nepřijatelně pomalá. Zpracovat prvních 100 000 bloků blockchainu zabralo přibližně týden. Zpracování bylo nutno ukončit u zhruba 150 000 bloků, kdy databáze přestávala rozumně fungovat. Databáze přestávala odpovídat na dotazy a v některých případech nebylo možné ji ani korektně vypnout. Prvních 150 000 bloků zabírá zhruba 0,5GB prostoru blockchainu, tedy méně než 1%. Databáze však zabírala přibližně 200GB, což je nepřijatelně mnoho, na tak malé množství dat.

Při vypnuté clusterizaci se zpracování výrazně zrychlilo. Prvních 100 000 bloků bylo zpracováno za zhruba 8 hodin. Problémy se stabilitou databáze a množstvím datového prostoru však přetrvaly.

5.2.2 Vizualizace

Další částí, která je zajímavá z pohledu výkonnosti je vizualizace transakcí. Samotné vykreslování samozřejmě záleží na výkonu klientského počítače a použitém prohlížeči. Samotná vykreslovací knihovna *vis* [23] používá grafickou akceleraci a zvládá pracovat i s velkým počtem uzlů, viz. C.1. S výkonným klientským počítačem je tedy možné vizualizovat i velké množství transakcí.

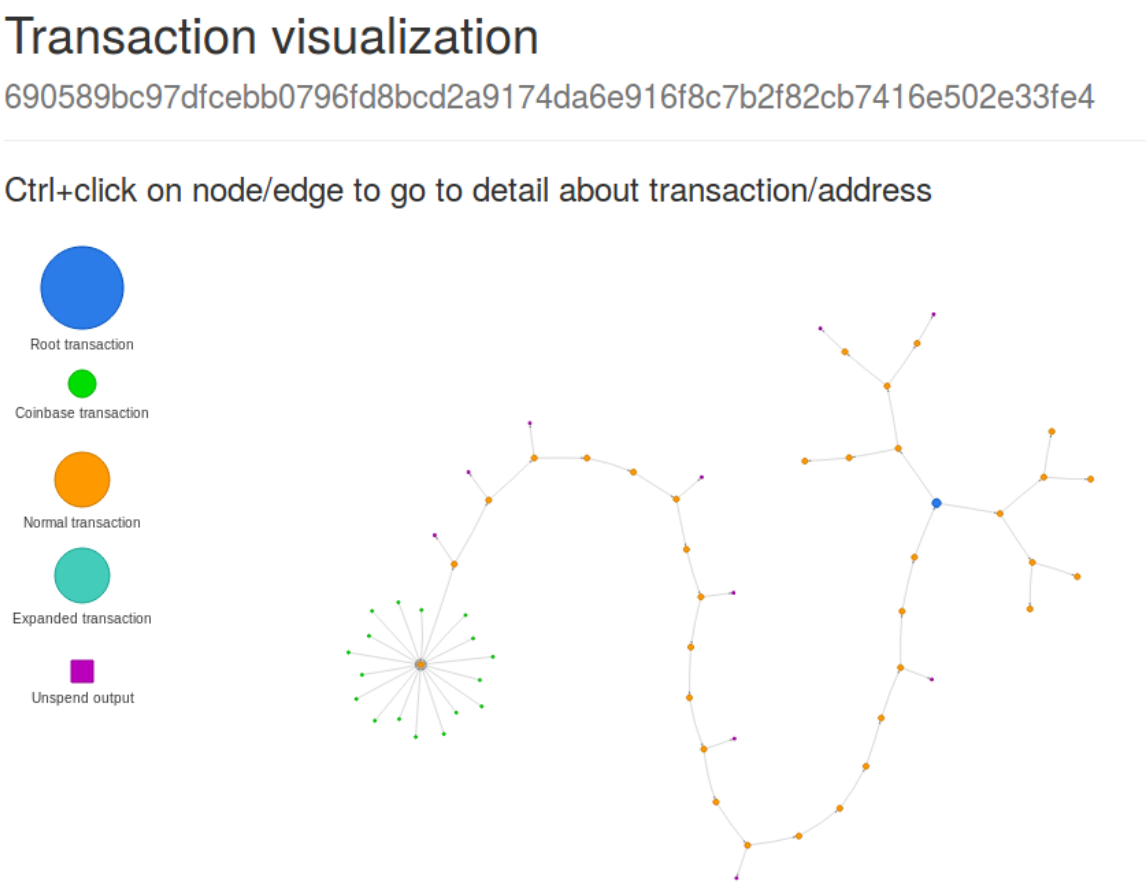
Problém s vykreslováním nastane v případě, že má být jednorázově zobrazeno velké množství dat, typicky při zobrazení transakce s velkým množstvím vstupů resp. výstupů a současném zobrazení několika následujících resp. předcházejících kroků. V takovémto případě knihovna *vis* hlásí chybu a odmítne graf vykreslit. V případě že jsou transakce v grafu expandovány postupně, tak tento problém nenastává.

Dalším problémem může být načítání a zobrazování rozsáhlých transakcí, kdy CCA může narážet na systémové limity. Například při vizualizaci transakce *abfb109b88700d25 a0cb66c6ce0061784191ffb8a4a9564e043bc668511e89b7*, která má 205 vstupů a zobrazení pouze jednoho kroku zpět, CCA alokuje více než 500 MB paměti, což může narazit na limit, nastavený na webovém serveru.

5.3 Analýza konkrétního případu

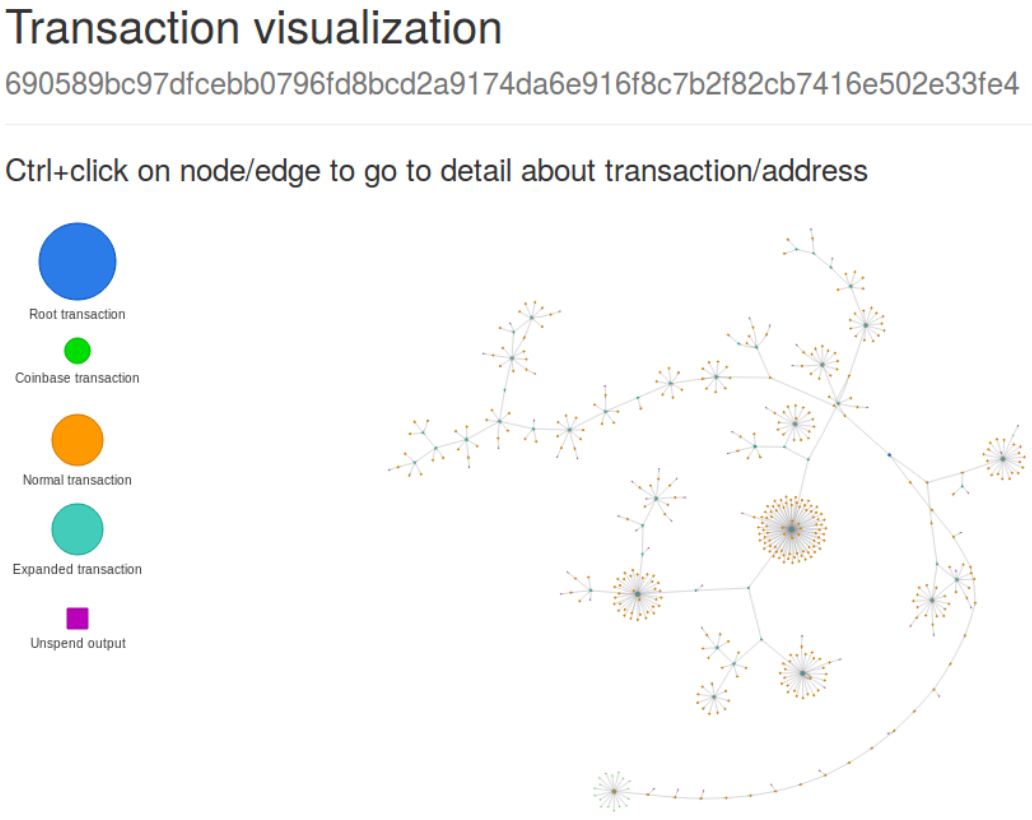
V této sekci popisuji analýzu jedné konkrétní transakce a informace, které je o ní možné zjistit z CCA. Vybraná transakce má txid *690589bc97dfcebb0796fd8bcd2a9174da6e916f8c7b2f82cb7416e502e33fe4*.

V CCA mohu rychle zjistit, že tato transakce je součástí bloku *00000000000012d8427a36b4cb598dfcc0ea82938974a31ea0dd0c8d5b73a4e6*, který má výšku 129929. Současně zjistím že tato transakce byla do blockchainu zařazena 2011-06-11 v 02:04:30. Dále zjistím, že tato transakce čerpá peníze z adresy *1FtgrggtABDTtjHd4nD5rtuD4YZoAysPyz*. Z vizualizace této transakce mohu zjistit (viz. obrázek 5.1) , že všechny peníze použité na vstupu pocházejí z transakce *0f5b4e0ad45737762ad3c1a134d133570e25e8bbc09086dd8678adf0b81f4c16*, která peníze posbírala z několika coinbase transakcí. Přičemž mezi oběma výše zmíněnými transakce je dalších 20 transakcí.



Obrázek 5.1: Dohledání zdroje transakce

Dohledání aktuální pozice předaných Bitcoinů je v tomto případě složité, neboť výstup zkoumané transakce se záhy mísí s mnoho jinými transakcemi, viz. obrázek 5.2.



Obrázek 5.2: Dohledání konečného cíle transakce

5.4 Použitelnost pro jiné kryptoměny

CCA je primárně navržen pro Bitcoin, ale je možné jej snadno přizpůsobit pro jiné kryptoměny, založené na stejném protokolu. Architektura CCA již obsahuje částečnou podporu pro Litecoin (2.3.2), tato však není dostačující k tomu aby CCA mohl pracovat s Litecoinem. Dokončení podpory pro Litecoin by však mělo být relativně snadné, obdobně pak přidání podpory pro jiné kryptoměny založené na Bitcoinu, jako třeba DogeCoin (2.3.3).

Pro kryptoměny založené na jiných protokolech je CCA teoreticky také použitelné, záleží však na míře, jakou se daný protokol liší od Bitcoinu. Pravděpodobně by bylo možné do CCA přidat podporu pro kryptoměny založené na protokolu CryptoNote (2.2.2), znamenalo by to však implementaci dalšího parseru blockchainu a další rozšíření. Pro kryptoměny založené na Ripple (2.2.3) či Ethereu (2.2.4) není CCA vhodný, tyto protokoly se diametrálně odlišují od Bitcoinu a jejich podpora by znamenala velké zásahy do CCA.

Kapitola 6

Závěr

V rámci diplomové práce jsem nastudoval základní principy kryptoměnových protokolů CryptoNight, Ripple a Ethereum, a podrobněji jsme nastudoval protokol Bitcoin. Vytvořil jsem parser blockchainu pro kryptoměnu Bitcoin napsaný v jazyce PHP. Tento nástroj jsem integroval do programu CCA. V CCA jsem předělal ukládání dat z databázového systému MongoDB do Neo4j a zprovoznil import dat z parseru blockchainu. Dále jsem rozšířil nástroj CCA o několik nových obrazovek, vylepšil grafické zobrazení trasování transakcí a přidal další metodu clusterizace. Taktéž jsem opravil drobné chyby v CCA.

Po implementaci se ukázalo že databáze Neo4j není pro CCA vhodná. Nezvládá práci s velkými objemy dat, uložená data zabírají příliš mnoho diskového prostoru a vkládání nových záznamů je pomalé. Pro praktické použití doporučuji CCA postavit spíše nad strukturovanou SQL databází, vhodná by mohla být například PostgreSQL. Taková změna databáze by sice omezila funkčnost CCA, ale měla by pomoci vyřešit výše zmíněné problémy.

V rámci dalšího rozšíření nástroje je možné implementovat lepší analýzu jazyka Script, kde nyní CCA zvládá analyzovat pouze základní konstrukce. Další možností je analýza podpisů u multisign transakcí, z nichž je možné určit, kdo použil výstup dané transakce, dosud je možné pouze určit, které veřejné klíče mohou transakci využít, zatím však CCA neumí zjistit které klíče byly použity pro využití transakce. Dále je možné CCA rozšířit o další metody clusterizace.

Literatura

[1] Drozda, T.: *Forenzní analýza sítě Bitcoin*. Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, Brno, 2016, vedocí príce Veselý Vladimír.

[2] Okupski, K.: Working Paper. *Bitcoin Developer Reference*, 2015.

[3] WWW stránky: The Differences between Bitcoin, Litecoin, Peercoin, Dogecoin and other AltCurrencies. `http://blog.gocoin.com/bitcoin-litecoin-peercoin-dogecoin/`, 2010-06-08 [cit. 2016-03-25].

[4] WWW stránky: Grafová databáze Neo4j. `https://www.zdrojak.cz/serialy/grafova-databaze-neo4j/`, 2013-10-21 [cit. 2016-11-4].

[5] WWW stránky: What is the Difference Between Litecoin and Bitcoin. `http://www.coindesk.com/information/comparing-litecoin-bitcoin/`, 2014-04-02 [cit. 2016-04-10].

[6] WWW stránky: The Pros and Cons of the X11 Algorithm. `https://www.cryptocoinsnews.com/pros-cons-x11-algorithm/`, 2014-04-10 [cit. 2016-04-15].

[7] WWW stránky: Litecoin vs. Darkcoin: Can the Coins Compete? `https://www.cryptocoinsnews.com/litecoin-vs-darkcoin-can-coins-compete/`, 2014-06-24 [cit. 2016-04-14].

[8] WWW stránky: Proof of stake: How I Learned to Love Weak Subjectivity. `https://blog.ethereum.org/2014/11/25/proof-stake-learned-love-weak-subjectivity/`, 2014-11-25 [cit. 2016-04-29].

[9] WWW stránky: Bitcoin multisig the hard way: Understanding raw P2SH multisig transactions. `http://www.soroushjp.com/2014/12/20/bitcoin-multisig-the-hard-way-understanding-raw-multisignature-bitcoin-transactions/`, 2014-12-20 [cit. 2017-03-02].

[10] WWW stránky: CryptoNote - Bitcoin. `https://en.bitcoin.it/wiki/CryptoNote`, 2014-12-24 [cit. 2016-04-27].

[11] WWW stránky: Bitcoin Vs. Litecoin: What's The Difference? `http://www.investopedia.com/articles/investing/042015/bitcoin-vs-litecoin-whats-difference.asp`, 2015-04-05 [cit. 2016-04-08].

[12] WWW stránky: Monero MRO Information. https://cryptocointalk.com/topic/11933-monero-mro-information/, 2015-06-10 [cit. 2016-04-18].

[13] WWW stránky: Technology - Ripple. https://ripple.com/technology/, 2015-06-10 [cit. 2016-04-20].

[14] WWW stránky: Contracts and Transactions - Ethereum. https://github.com/ethereum/go-ethereum/wiki/Contracts-and-Transactions, 2015-08-25 [cit. 2016-04-29].

[15] WWW stránky: CryptoNote Technology. https://cryptonote.org/inside.php, 2015-10-12 [cit. 2016-04-20].

[16] WWW stránky: Block hashing algorithm. https://en.bitcoin.it/wiki/Block_hashing_algorithm, 2015-12-12 [cit. 2016-12-28].

[17] WWW stránky: Ripple (playment protocol). https://en.wikipedia.org/wiki/Ripple, 2016-04-10 [cit. 2016-04-20].

[18] WWW stránky: Ethereum Project. https://www.ethereum.org/, 2016-04-10 [cit. 2016-04-29].

[19] WWW stránky: White Paper - Ethereum. https://github.com/ethereum/wiki/wiki/White-Paper, 2016-04-13 [cit. 2016-04-29].

[20] WWW stránky: CryptoNote. https://en.wikipedia.org/wiki/CryptoNote, 2016-04-23 [cit. 2016-04-28].

[21] WWW stránky: Crypto-Currency Market Capitalizations. https://coinmarketcap.com/, 2016-04-28 [cit. 2016-04-28].

[22] WWW stránky: Ethash - Ethereum. https://www.ethereum.org/, 2016-24-03 [cit. 2016-04-29].

[23] WWW stránky: vis.js - A dynamic, browser based visualization library. http://visjs.org/, 2017.

[24] WWW stránky: Script. https://en.bitcoin.it/wiki/Script, 2017-01-04 [cit. 2017-03-02].

[25] WWW stránky: Technical background of version 1 Bitcoin addresses. https://en.bitcoin.it/wiki/Technical_background_of_version_1_Bitcoin_addresses, 2017-01-29 [cit. 2017-02-27].

[26] WWW stránky: List of address prefixes. https://en.bitcoin.it/wiki/List_of_address_prefixes, 2017-03-23 [cit. 2017-03-27].

Přílohy

Příloha A

Obsah CD

Přílohou diplomové práce je CD s následujícím obsahem:

app Zdrojové kódy projektu,

text-src zdrojové kódy textové části práce,

README.txt popis instalace a užívání aplikace,

xocena04.pdf textová část práce.

Příloha B

Popis bitcoin scriptu

V této kapitole je popsán jazyk Script, který je použit v Bitcoinu pro určování podmínek pro využití transakce a její samotné využití. Zejména je zde kladen důraz na binární reprezentaci běžně používaných konstrukcí scriptu. Popisy základních konstrukcí Scriptu je možná nalézt v [24]. Tato kapitola se věnuje pouze některým praktickým použitím Scriptu.

Pojem *ScriptPubkey* označuje využití Scriptu ve výstupu transakce. V tomto případě Script definuje funkci, která může mít určité vstupní parametry. Daný výstup pak může využít libovolný uživatel, který dokáže dodat takové vstupy, aby definovaná funkce byla vyhodnocena kladně. V případě že *ScriptPubkey* nemá žádné vstupy, může tento výstup použít každý resp. nikdo, podle toho, co Script definuje.

Pojem *ScriptSig* označuje část Scriptu používanou ve vstupech transakce. Cílem *ScriptSig* je dodat taková vstupní data, se kterými bude *ScriptPubkey* použitého výstupu vyhodnocen kladně. *ScriptSig* může obsahovat podpis, veřejný klíč i definici další Script funkce, záleží na použitém *ScriptPubkey*.

Tato kapitola se dále zabývá konkrétními platebními metodami, které se v síti Bitcoin běžně používají.

B.1 Pay-to-Pubkey (P2PK)

Při použití této metody odesilatel platby ve *ScriptPubkey* specifikuje veřejný klíč, kterým identifikuje adresáta výstupu. Korespondující *ScriptSig* obsahuje podpis vytvořený pomocí korespondujícího soukromého klíče.

Příklad *ScriptPubkey*:

```
4104678afdb0fe5548271967f1a67130b7105cd6a828e03909a67962e0ea1f61
deb649f6bc3f4cef38c4f35504e51ec112de5c384df7ba0b8d578a4c702b6bf1
1d5fac
```

Jednotlivá pole Scriptu mají tento význam:

```
41 - následujících 0x41 = 65 B ulož na zásobník
04678afdb0fe5548271967f1a67130b7105cd6a828e03909a67962e0ea1f61
deb649f6bc3f4cef38c4f35504e51ec112de5c384df7ba0b8d578a4c702b6b
f11d5f - veřejný klíč
ac - proveď operaci OP_CHECKSIG
```

ScriptSig pro vyzvednutí této platby obsahuje pouze podpis a související pole. Může vypadat například takto:

```
493046022100e26d9ff76a07d68369e5782be3f8532d25ecc8add58ee256da6
c550b52e8006b022100b4431f5a9a4dcb51cbdcaae935218c0ae4cfc8aa903f
e4e5bac4c208290b7d5d01
```

Kde jednotlivá pole mají význam:

```
49 - Celý podpis je dlouhý 0x49 = 73B
30 - Indikátor začátku podpisu
46 - zbytek podpisu má 0x46 = 70B
02 - Indikátor začátku celého čísla
21 - Část R podpisu je dlouhá 0x21 = 33B
00e26d9ff76a07d68369e5782be3f8532d25ecc8add58ee256da6c550b52e8
006b - Část R podpisu
02 - Indikátor začátku celého čísla
21 - Část S~podpisu je dlouhá 0x21 = 33B
00b4431f5a9a4dcb51cbdcaae935218c0ae4cfc8aa903fe4e5bac4c208290b
7d5d - Část S~podpisu
01 - Typ podpisu SIGHASH_ALL
```

B.2 Pay-to-PubkeyHash (P2PKH)

V této platební metodě odesilatel specifikuje hash. Při použití tohoto výstupu musí uživatel dodat veřejný klíč, jehož hash odpovídá zadanému, a odpovídající podpis.

Příklad *ScriptPubkey*:

```
76a91412ab8dc588ca9d5787dde7eb29569da63c3a238c88ac
```

Rozbor jednotlivých polí:

```
76 - OP_DUP
a9 - OP_HASH160
14 - Ulož 0x14 = 20B na zásobník
12ab8dc588ca9d5787dde7eb29569da63c3a238c - Hash veřejného klíče
88 - OP_EQUALVERIFY
ac - OP_CHECKSIG
```

Příklad *ScriptSig*:

```
4930460221009f8aef83489d5c3524b68ddf77e8af8ceb5cba89790d31d2d2d
b0c80b9cbfd26022100bb2c13e15bb356a4accdd55288e8b2fd39e204a93d84
9ccf749eaef9d8162787014104f9804cfb86fb17441a6562b07c4ee8f012bdb
2da5be022032e4b87100350ccc7c0f4d47078b06c9d22b0ec10bdce4c590e0d
01aed618987a6caa8c94d74ee6dc
```

Rozbor jednotlivých polí:

```
49 - Délka podpisu = 0x49 = 73B
30460221009f8aef83489d5c3524b68ddf77e8af8ceb5cba89790d31d2d2db0
c80b9cbfd26022100bb2c13e15bb356a4accdd55288e8b2fd39e204a93d849c
cf749eaef9d816278701 - podpis
41 - Délka veřejného klíče = 0x41 = 41B
04f9804cfb86fb17441a6562b07c4ee8f012bdb2da5be022032e4b87100350c
cc7c0f4d47078b06c9d22b0ec10bdce4c590e0d01aed618987a6caa8c94d74e
e6dc - veřejný klíč
```

Rozbor podpisu je stejný jako v kapitole B.1.

B.3 Multisig

Při použití metody multisig odesilatel specifikuje několik veřejných klíčů a také určí kolik z nich je třeba pro vyzvednutí tohoto výstupu. Příslušný *ScriptSig* pak musí obsahovat požadovaný počet podpisů od daných klíčů.

Příklad *ScriptPubkey*

```
524104a882d414e478039cd5b52a92ffb13dd5e6bd4515497439dffd691a0f1
2af9575fa349b5694ed3155b136f09e63975a1700c9f4d4df849323dac06cf3
bd6458cd41046ce31db9bdd543e72fe3039a1f1c047dab87037c36a669ff90e
28da1848f640de68c2fe913d363a51154a0c62d7adea1b822d0503507741826
7b1a1379790187410411ffd36c70776538d079fbae117dc38effafb33304af8
3ce4894589747aee1ef992f63280567f52f5ba870678b4ab4ff6c8ea600bd21
7870a8b4f1f09f3a8e8353ae
```

Rozbor jednotlivých polí:

```
52 - OP_2 - na zásobník ulož 2 = počet klíčů pro vyzvednutí
41 - délka veřejného klíče = 0x41= 65B
04a882d414e478039cd5b52a92ffb13dd5e6bd4515497439dffd691a0f12af9
575fa349b5694ed3155b136f09e63975a1700c9f4d4df849323dac06cf3bd64
58cd- první veřejný klíč
41 - délka veřejného klíče = 0x41= 65B
046ce31db9bdd543e72fe3039a1f1c047dab87037c36a669ff90e28da1848f6
40de68c2fe913d363a51154a0c62d7adea1b822d05035077418267b1a137979
0187 - druhý veřejný klíč
41 - délka veřejného klíče = 0x41= 65B
0411ffd36c70776538d079fbae117dc38effafb33304af83ce4894589747aee
1ef992f63280567f52f5ba870678b4ab4ff6c8ea600bd217870a8b4f1f09f3a
8e83 - třetí veřejný klíč
53 - OP_3 - na zásobník ulož 3 = celkový počet klíčů
ae - OP_CHECKMULTISG
```

Příklad *ScriptSig*:

```
00483045022100e6fbdd61e6c59508a000603a5cc092aad3b404cf80d9ab04c
b790815e3160c3702205700c1a5ba6f53620f3ee776ca452243cdac2e6430ec
4c4777020ed8be3a6aa201483045022100dad5dc893fce871cd9630c3e4e21a
d0bf03367f1b065a1e58c58162b95290e0302205aede8141dd30e5adc24331e
f3297bbab071751e4736cbf34e7dc1ae014f5b2201
```

rozbor jednotlivých polí:

```
00 - OP_FALSE
48 - Délka prvního podpisu = 0x48 = 72B
3045022100e6fbdd61e6c59508a000603a5cc092aad3b404cf80d9ab04cb790
815e3160c3702205700c1a5ba6f53620f3ee776ca452243cdac2e6430ec4c47
77020ed8be3a6aa201 - první podpis
48 - Délka druhého podpis = 0x48 = 72B
45022100dad5dc893fce871cd9630c3e4e21ad0bf03367f1b065a1e58c58162
b95290e0302205aede8141dd30e5adc24331ef3297bbab071751e4736cbf34e
7dc1ae014f5b2201 - druhý podpis
```

B.4 Pay-to-ScriptHash (P2SH)

U této metody odesilatel platby specifikuje hash. Při použití výstupu musí uživatel uvnitř *ScriptSig* definovat *ScriptPubkey* odpovídající danému hashi a k tomu dodat data pro splnění daného Scriptu.

Příklad *ScriptPubkey*:

```
a914e9c3dd0c07aac76179ebc76a6c78d4d67c6c160a87
```

Rozbor jednotlivých polí:

```
a9 - OP_HASH160
14 - Délka hashe = 0x14 = 20B
e9c3dd0c07aac76179ebc76a6c78d4d67c6c160a - hash scriptu
87 - OP_EQUAL
```

Příklad *ScriptSig*:

```
0047304402205ae53c9d31db9f75d0d92e6d7670f0e835d753323dcb8b92511
b187f96b38fa40220448acbc3965594e8ea6c2e95cadeeb86e5baa8d82bb8b0
92ee213b630af4315501255121030ec111fb923515ba4747f3c7005b4398e81
d816a66ba50306aacac2f405ac72651ae
```

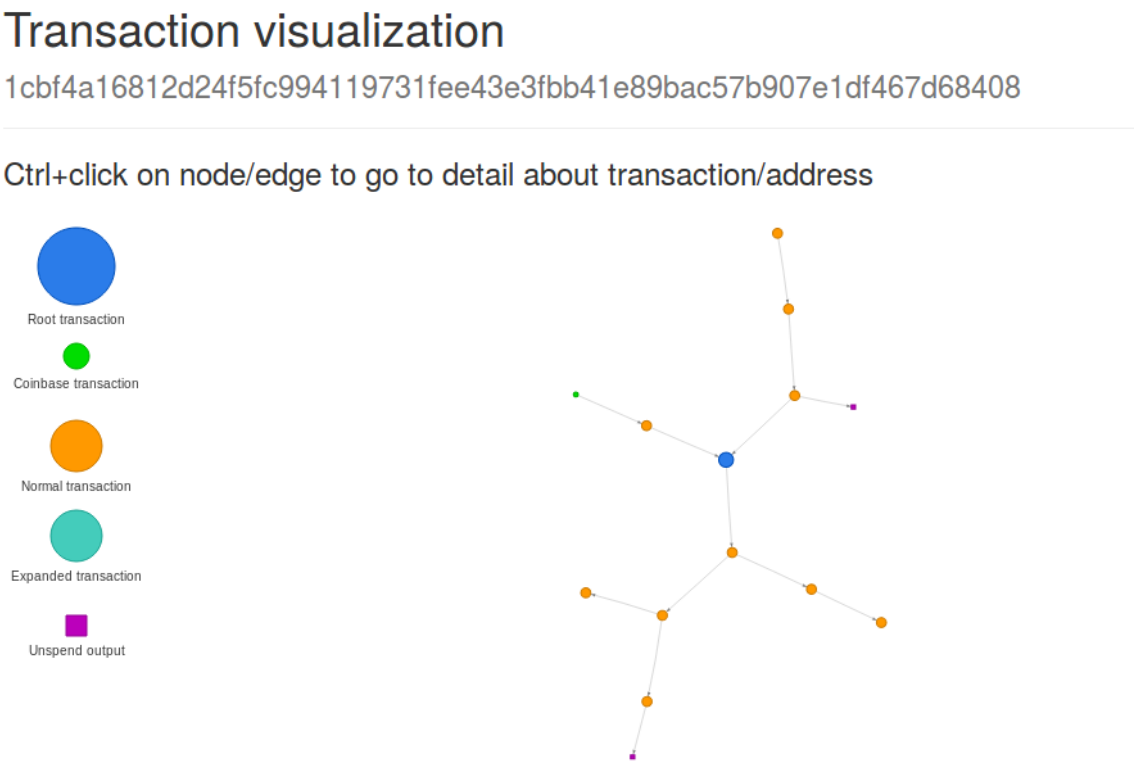
Rozbor jednotlivých polí:

```
00 - OP_FALSE
47 - Ulož na zásobník 0x47 = 71B (délka podpisu)
304402205ae53c9d31db9f75d0d92e6d7670f0e835d753323dcb8b92511b18
7f96b38fa40220448acbc3965594e8ea6c2e95cadeeb86e5baa8d82bb8b092
ee213b630af4315501 - podpis
25 - Délka Scriptu 0x25 = 37B
51 - OP_1 - ulož na zásobník číslo 1
21 - Délka veřejného klíče 0x21 = 33B
030ec111fb923515ba4747f3c7005b4398e81d816a66ba50306aacac2f405a
c726 - Komprimovaný veřejný klíč
51 - OP_1 - ulož na zásobník číslo 1
ae - OP_CHECKSIG
```

Příloha C

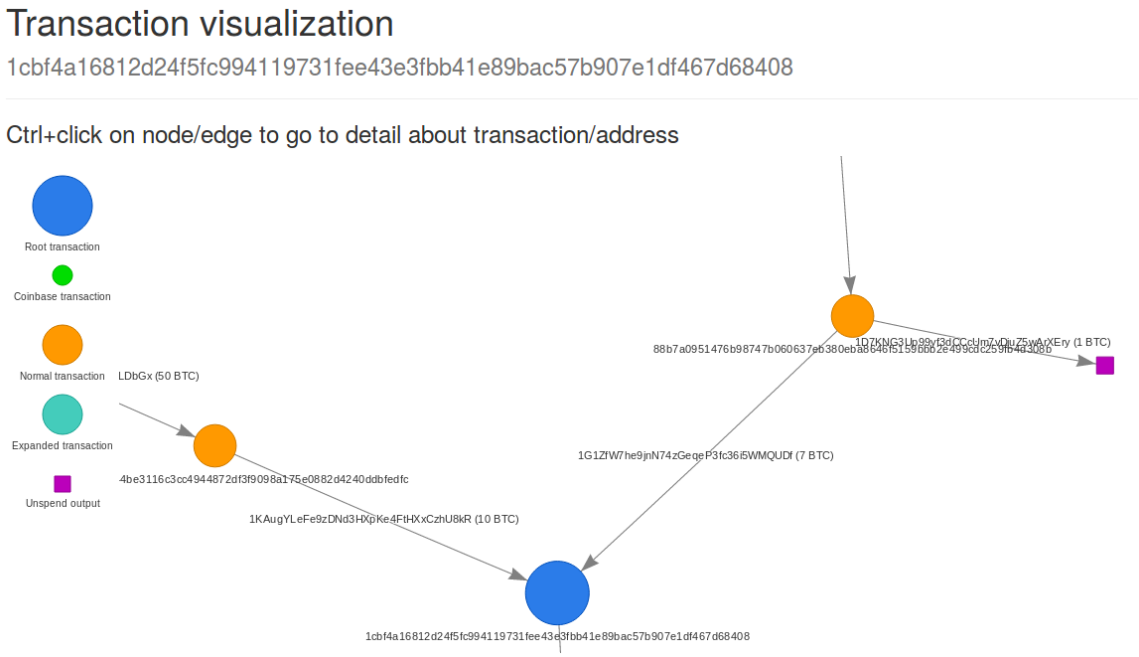
Příklad vizualizace transakce

V této příloze je příklad vizualizace jedné konkrétní transakce. Když si uživatel zobrazí vizualizace jedné transakce, zobrazí se mu zhruba obrázek C.1.



Obrázek C.1: Vizualizace transakce

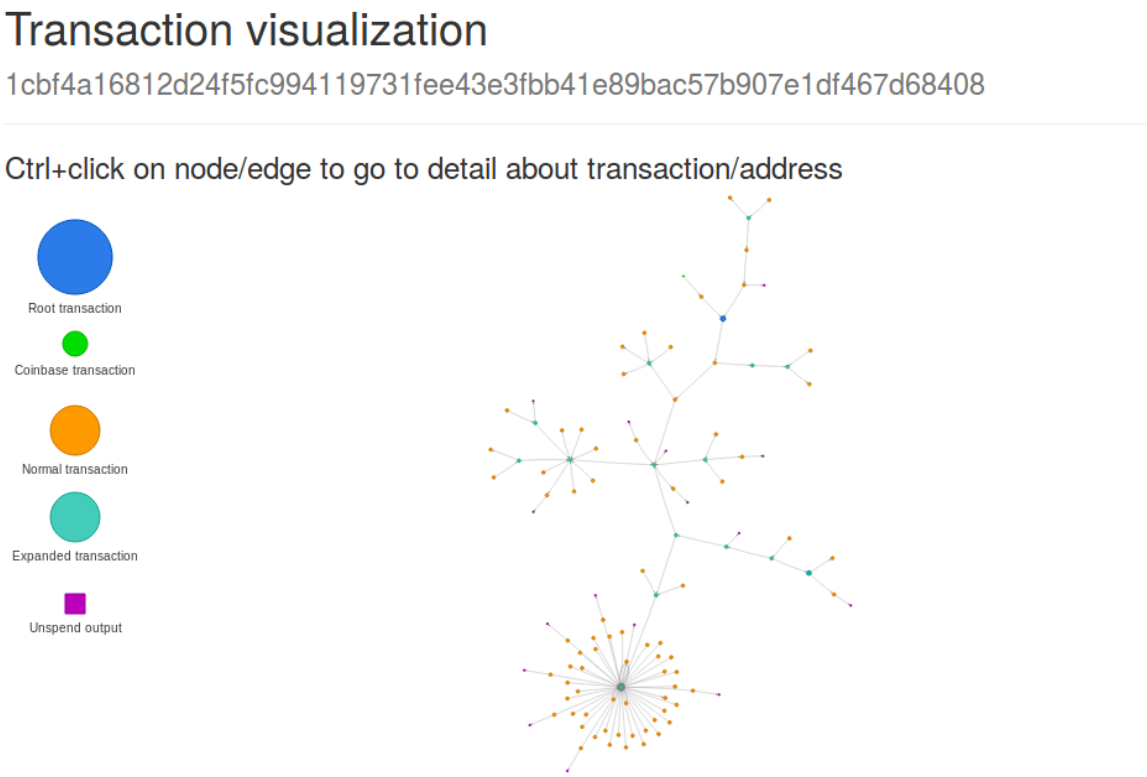
Přiblížení na požadovanou transakci je vidět na obrázku C.2.
Obrázek C.3 zobrazuje trasování Bitcoinů napříč větším množstvím transakcí.



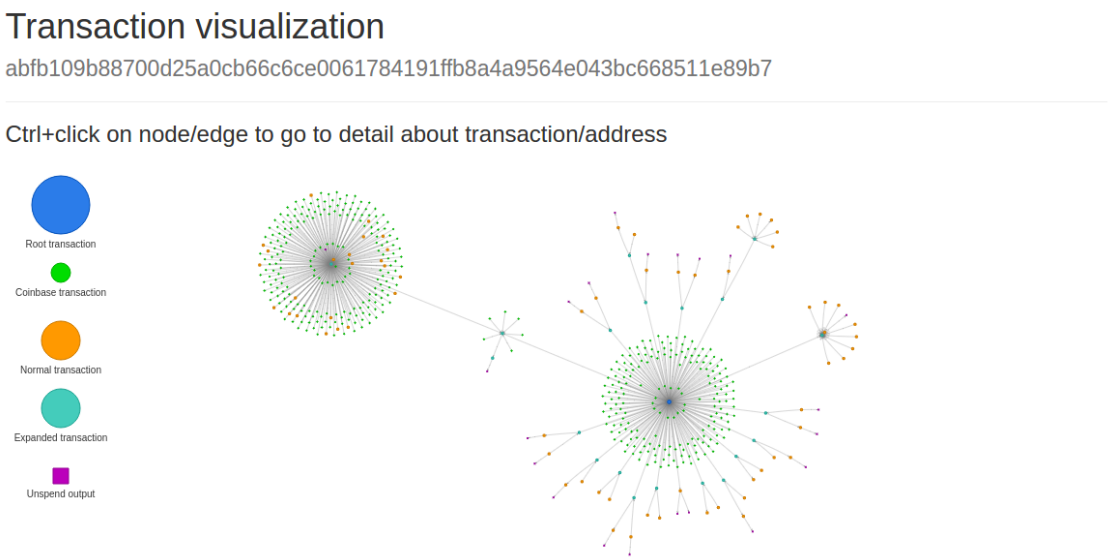
Obrázek C.2: Přiblížení vizualizace

C.1 Výkonnost zobrazování

V této sekci je uveden příklad vykreslovaní při zobrazení velkého počtu uzlů. Graf v obrázku C.4 zobrazuje 516 uzlů a 531 hran. Pro zobrazení byl použit prohlížeč *Google chrome* ve verzi 58. Manipulace s uzly je stále relativně plynulá a bez větších problémů.



Obrázek C.3: Větší množství transakcí



Obrázek C.4: Velké množství transakcí

Příloha D

Nové obrazovky aplikace

Transaction: ac85e00b880e08ab85223be9b09bd305e645009ea4058983887b2d2e7c0c3adc

Input number: 0

Input detail

Source transaction	ceb9628f2de0747bc1d15a33f8ce7425b39a367f691d7fa06eed680742b5ea84
Source output	1
Value	0.0785
Input address	1JQwwU6TiSUeQhHo5H7T14WMMmWj3jveoz
ScriptSig	473044022034f707c6c0f7027d189ae39d457db456a0f6f14c3b99e7266e3a7f325160c6270220045f34841cb1acff93b8d845e4dd50fb551305436a6222cf8cf860d744055285014104fb2caf8566ef0a44dc109b5789cd3bfae18155550c69500a56b76df68a8b2b027148e1f3bf22828770d046c861e6a1b748cc829a82c3220c2d22173666d474f2
Pubkey script present in input	

Public keys present in ScriptSig

04fb2caf8566ef0a44dc109b5789cd3bfae18155550c69500a56b76df68a8b2b027148e1f3bf22828770d046c861e6a1b748cc829a82c3220c2d22173666d474f2

ScriptSig Signatures

R field	S field	Signature type
34f707c6c0f7027d189ae39d457db456a0f6f14c3b99e7266e3a7f325160c627	045f34841cb1acff93b8d845e4dd50fb551305436a6222cf8cf860d744055285	SIGHASH_ALL

Obrázek D.1: Detail vstupu transakce

Transaction: ac85e00b880e08ab85223be9b09bd305e645009ea4058983887b2d2e7c0c3adc

output number: 0

Output detail

Value	0.048
Output address	17ww93e6A17iSsqifoSRHtzNhtBJt6wqsj
ScriptPubkey	76a9144c3468cc0741864c2403e1658e5d6e8bca52227688ac
Spent?	Yes
Spent in transaction	e91af81a06bca9122690c7f19b5904114cd426898be81c2bd0dd508cddb21fcb
Spent time	2011-08-14 16:54:39

Payment information

Payment type	PAY TO HASH OF PUBLIC KEY (P2PKH)
Text data in output	

Addresses that can to redeem this output

17ww93e6A17iSsqifoSRHtzNhtBJt6wqsj

Public keys used in output

Obrázek D.2: Detail výstupu transakce

Public key: 041ada81ea00c11098d2f52c20d5aa9f5ba13f9b583fda66f2a478dd7d95a7ab615159d98b63df2e6f3ecb3ef9eda138e4587e7afd31e7f434cbb6837e17feb0c5

Summary

Uncompressed public key	041ada81ea00c11098d2f52c20d5aa9f5ba13f9b583fda66f2a478dd7d95a7ab615159d98b63df2e6f3ecb3ef9eda138e4587e7afd31e7f434cbb6837e17feb0c5
Compressed public key	031ada81ea00c11098d2f52c20d5aa9f5ba13f9b583fda66f2a478dd7d95a7ab61
Ripemd160 hash of uncompressed key	49c0e836a0bb1e176097e6408efe6cb4c11a98f5
Ripemd160 hash of compressed key	e22833ee5ab6fa938a46f703e8f6c4f10885b48e

Known adresses

33qfmRm4tR5mnkJ5fsjpK7GmQeisgLS21y

3PGsRTvcfNaJHb4Bw7av1u3qdeLv1MCoAp

17iyRRXBHJKbv5DKPPkttWewm3CHdNPGQd

1McoshRvL7xQQ6FnNFrZriHvb5ftRAVQPg

Obrázek D.3: Detail veřejného klíče