Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Ekonomická fakulta

Katedra aplikované matematiky a informatiky

Diplomová práce

ROZBOR A NÁVRH APLIKACE PRO DIGITÁLNÍ MĚNU BITCOIN

Vypracoval: Bc. František Drdák

Vedoucí diplomové práce: doc. Ing. Ladislav Beránek, CSc.

České Budějovice 2015

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta ekonomická
Akademický rok: **2013/2014**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. František DRDÁK**
Osobní číslo:	**E13860**
Studijní program:	**N6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Název tématu:	**Rozbor a návrh aplikace pro digitální měnu Bitcoin**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování:

Bitcoin je open source P2P digitální měna, která je v současné době poměrně populární. Cílem práce je popsat vznik, vývoj, dnešní stav této měny a ukázat principy, na jejichž základě tato digitální měna funguje, rozebrat přednosti a slabiny. Dále navrhnout a implementovat aplikaci, pomocí které by bylo možné provést platbu touto měnou prostřednictvím mobilního zařízení.

Metodický postup:

1. Studium odborné literatury, seznámení se s principy digitálních měn, vývoj a dnešní stav.
2. Provedení analýzy, návrh architektury a návrh řešení.
3. Praktická část - vytvoření vlastní aplikace, která bude umožňovat platbu.
4. Závěry a obecná doporučení.

Rozsah grafických prací:

Rozsah pracovní zprávy: **50 - 60 stran**

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

1. **Anynomous.** *BitCoin Made Easy.* **CreateSpace Independent Publishing Platform, 2013. ISBN 978-1484094198.**
2. **FORRESTER, D. a M. SOLOMON.** *Bitcoin Exposed: Today's Complete Guide to Tomorrow's Currency.* **CreateSpace Independent Publishing Platform, 2013. ISBN 978-1489598660.**

Vedoucí diplomové práce: **doc. Ing. Ladislav Beránek, CSc.**
Katedra aplikované matematiky a informatiky

Datum zadání diplomové práce: **7. ledna 2014**

Termín odevzdání diplomové práce: **15. dubna 2015**

doc. Ing. Ladislav Rolínek, Ph.D.
děkan

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13 (26)
370 05 České Budějovice

L.S.

prof. RNDr. Pavel Tlustý, CSc.
vedoucí katedry

V Českých Budějovicích dne 26. února 2014

Prohlašuji, že v souladu s § 47 zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to - v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne 9. 4. 2015

Bc. František Drdák

Poděkování

Na tomto místě bych rád poděkoval celé své rodině a přátelům, kteří mě podporovali po celou dobu mého studia. Dále bych chtěl poděkovat vedoucímu mé diplomové práce doc. Ing. Ladislavu Beránkovi CSc. za jeho cenné rady a připomínky.

OBSAH

1. ÚVOD

Virtuální měny získávají v dnešní době čím dál víc na popularitě a to hlavně mezi uživateli internetu. Je to způsobeno hlavně tím, že se v této oblasti propojuje finanční svět se stále se rozvíjejícími IT technologiemi. Tento trend přináší nové možnosti v problematice bezhotovostního platebního styku a hlavně lze pomocí virtuálních měn razantně snížit transakční náklady spojené s placením či převodem peněžních prostředků. Samozřejmě, že tato možnost přináší pro její potencionální uživatele i nová rizika a je třeba na tyto problémy upozornit.

Za nejznámější virtuální měnu, která je dnes považována za obecně známou a velmi mediálně populární a to jak v pozitivním i negativním slova smyslu je Bitcoin. Tento platební nástroj byl představen již v roce 2009 jako měna, kterou lze zaplatit zboží či služby a dnes jí lze považovat i za prostředek do kterého lze investovat či spekulovat na vývoj kurzu oproti tradičním světovým měnám.

První matematický popis této měny zveřejnil Satoshi Nakamoto ve své práci Bitcoin: A Peer-to-Peer Electronic Cash System. Zde je jsou uvedeny základní principy této měny a požadavky na síť, ve které by bylo možné takovou měnu provozovat. Hlavním záměrem bylo vyloučit z této struktury finanční organizace a to zejména banky, přes které jde drtivá většina dnešních bezhotovostních transakcí. Na rozdíl od tradičních světových měn Bitcoin nemá žádnou centrální regulační organizaci, což je také částečná odpověď proč lze významně snížit náklady na transakci. Řízení a správa sítě je čistě na uživatelích a na robustně postavených základech Bitcoinu samotného.

Tato měna je digitální, což znamená, že nemá žádnou fyzickou formu. Uživatel si udržuje Bitcoiny například na disku, kde je to pouze shluk jedniček a nul. S využitím specializovaného softwaru, který se nazývá příznačně peněženka, může uživatel se svými prostředky nakládat podobně jako například při převodu peněz z internetového bankovnictví.

Bitcoin byla první virtuální měna postavená na základech kryptografie, ale v dnešní době je takových měn více jak 300. Přes 70% transakcí, které byly provedeny nějakou virtuální měnou byly provedeny právě pomocí Bitcoinu. Tržní kapitalizace je v dnešní době přibližně 3,7 miliardy USD a to přibližně při 80 000 transakcích za den.

Diplomová práce je rozdělena do několika hlavních částí, kde první část obsahuje základní informace, principy a historii jak se Bitcoin vyvíjel až do současnosti. Jsou zde také uvedeny zajímavé informace o bezpečnosti této měny a možnosti jakými způsoby lze takovou měnu vlastnit.

V další části se zabývám základním pojmům z oblasti kryptografie. Na úvod této kapitoly jsem zařadil principy a předpoklady kryptografie. Dále je nezbytné popsat fungování například hashovacích algoritmů. S tím souvisí také problematika šifrování. Zde se jedná hlavně o základní popis, jakým způsobem lze šifrovat zprávy a dále rozdíl mezi symetrickým a asymetrickým šifrováním. Poslední podkapitolou kryptografického základu je pak část týkající se digitálních podpisů. Všechny tyto principy je nutné pochopit, protože Bitcoin jako měna je na těchto základech postavena.

V třetí, nejobsáhlejší, části práce se zabývám nejprve vymezením pojmu měna, dále pak obecně funkcí a vlastnostmi peněz. V neposlední řadě se zabývám i vývojem měn až do současnosti. V této kapitole popsán i trend digitalizace peněz a celkové dopady na světovou ekonomiku. V návaznosti na vymezení pojmu digitální měny se pak již zabývám detailní analýzou samotných principů virtuální kryptoměny Bitcoin. Jsou zde popsány principy tvoření adres pro Bitcoinovou síť, dále pak jak se provádějí jednotlivé transakce v síti a následně i proces seskupování transakcí do bloků. Po té je vysvětlena i problematika decentralizované historie tzv. blok chainu a možnosti jak lze, kteroukoliv transakci vždy jednoznačně ověřit. V závěru této části práce je uveden příklad jakým způsobem může uživatel Bitcoinové sítě využívat.

Praktická část této diplomové práce je zaměřena na analýzu a návrh platební aplikace pro mobilní zařízení. Je zde využíváno open-source knihoven např. BitCoinJ. Aplikace je vyvíjena v programovacím jazyce Java s využitím ADT (Android Developer Tools) pro přenosná zařízení, které pracují pod operačním systémem Android. Pro účely analýzy a návrhu grafického uživatelského prostředí je použito UML diagramů. K zpřesnění tohoto UML návrhu byl využit volně dostupný grafický software a výsledné zpracování je též součástí této práce.

Toto téma jsem si vybral, protože se o zajímám o nové trendy v IT technologiích a problematiku virtuálních měn sleduji již více jak dva roky. Dalším důvodem je také fakt, že v současné době není dostatek kvalitní literatury, která by shrnovala principy a

možnosti této virtuální měny. Touto prací bych chtěl tedy umožnit širší veřejnosti pochopit o čem vlastně virtuální měna Bitcoin je a jakým způsobem ji lze používat.

2. CÍLE PRÁCE

Tato diplomová práce má dva hlavní cíle. Za prvé by měla čtenáři poskytnout ucelený náhled na problematiku virtuálních měn a hlavně Bitcoinu. To zahrnuje popis měny, principy fungování, jakým způsobem lze Bitcoin používá a jak funguje software, který je pro využívání nutný. Pro ucelenost tématu jsou v práci popsány i základy kryptografie nutné k pochopení vnitřních principů měny a dále pak podrobná analýza používaných algoritmů. V tomto případě se jedná hlavně o rozbor šifrovacích algoritmů a hashovacích funkcí.

Druhým cílem je návrh a částečná implementace platební aplikace pro virtuální kryptoměnu Bitcoin. Jedná se o navržení základních funkčností takové aplikace, to znamená, že jde hlavně o třídy, které zajišťují generování párů veřejného a soukromého klíče. Dále pak třídy, které umožnují vytvořit peněženku a její následnou transformaci do souboru na pevné úložiště. Další implementované třídy pak slouží k propojení aplikace s Bitcoinovou sítí, dále umožňují nalezení a stažení správného genesis bloku v závislosti na typu sítě a samozřejmě též k provedení platné transakce. V rámci tohoto cíle je též návrh optimálního grafického uživatelského prostředí aplikace.

Tato aplikace bude určena pro přenosná zařízení s operačním systémem Android a v dnešním světě mobilních zařízení by měla umožnit převádět Bitcoiny, bez nutnosti být u PC.

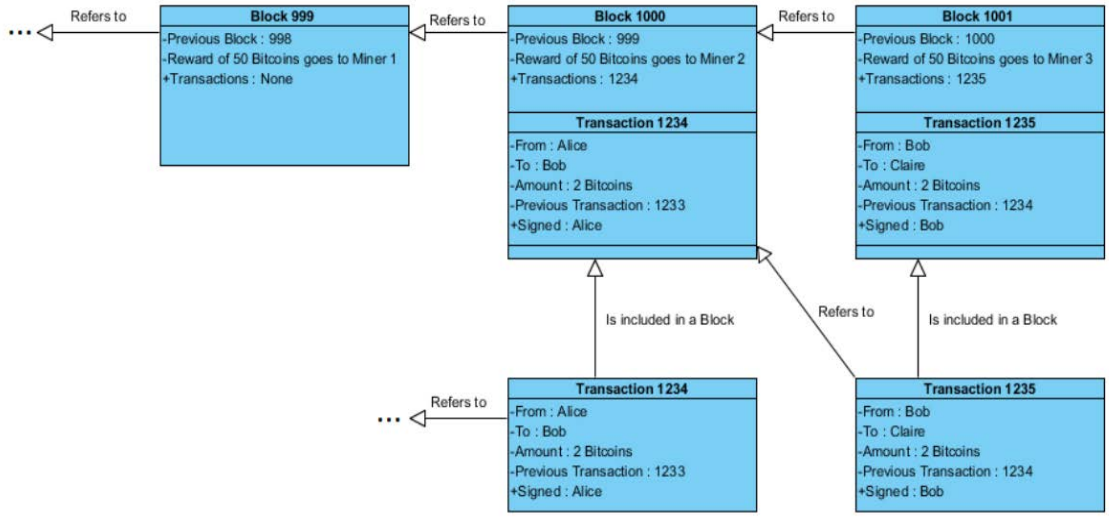
3. ZÁKLADNÍ INFORMACE O BITCOINU

Bitcoin je decentralizovaná virtuální kryptoměna. Není závislá na žádné centrální službě, která by řídila vytváření nebo tok peněz. Spoléhá se na kryptografické algoritmy, které mají za účel zabránění zneužívání systému. Navíc je Bitcoin open-source projekt takže veškeré informace, kódy a další dokumenty včetně vlastní wikipedie jsou volně k dispozici na internetu. (Developers, 2013)

Každý uživatel Bitcoinu vlastní pár privátního a veřejného klíče. Toto je velmi podobné jako například u bankovních účtů. Pokud chce uživatel někomu poslat určitý obnos peněz, vytvoří novou transakci a podepíše jí svým privátním klíčem. Každá takto vytvořená transakce si drží referenci, neboli odkaz, na tu předešlou. Předchozí transakcí se myslí taková transakce, kterou vytvořil někdo jiný a zaslal tak danému uživateli peníze. Z tohoto principu jasně vyplývá, že se Bitcoiny nemohou vytvářet z ničeho. Je to proud nebo lépe řetěz, kde jsou na sebe všechny články, tzn. provedené transakce, pevně navázány a dají se dohledat. Další vlastností, kterou lze odvodit je, že ten samý Bitcoin nemůže být utracen najednou dvakrát tzv. Double spending problém. (Perry, 2014) Každá transakce je vysílána skrz Bitcoin síť a následně je ověřována a nakonec označena jako valid (platná) a „spendable“ (český předklad je ve smyslu peníze povoleny k utracení).

Každých 10 minut jsou všechny transakce spojeny dohromady do tzv. Bitcoin bloku. Jakmile je transakce částí bloku je považována za „safe to spend“ (bezpečná k provedení) a to zejména kvůli, že je velmi těžké bloky zfalšovat nebo padělat. Všechny bloky jsou spojené dohromady do již zmiňovaného řetězu bloků (Block Chain).

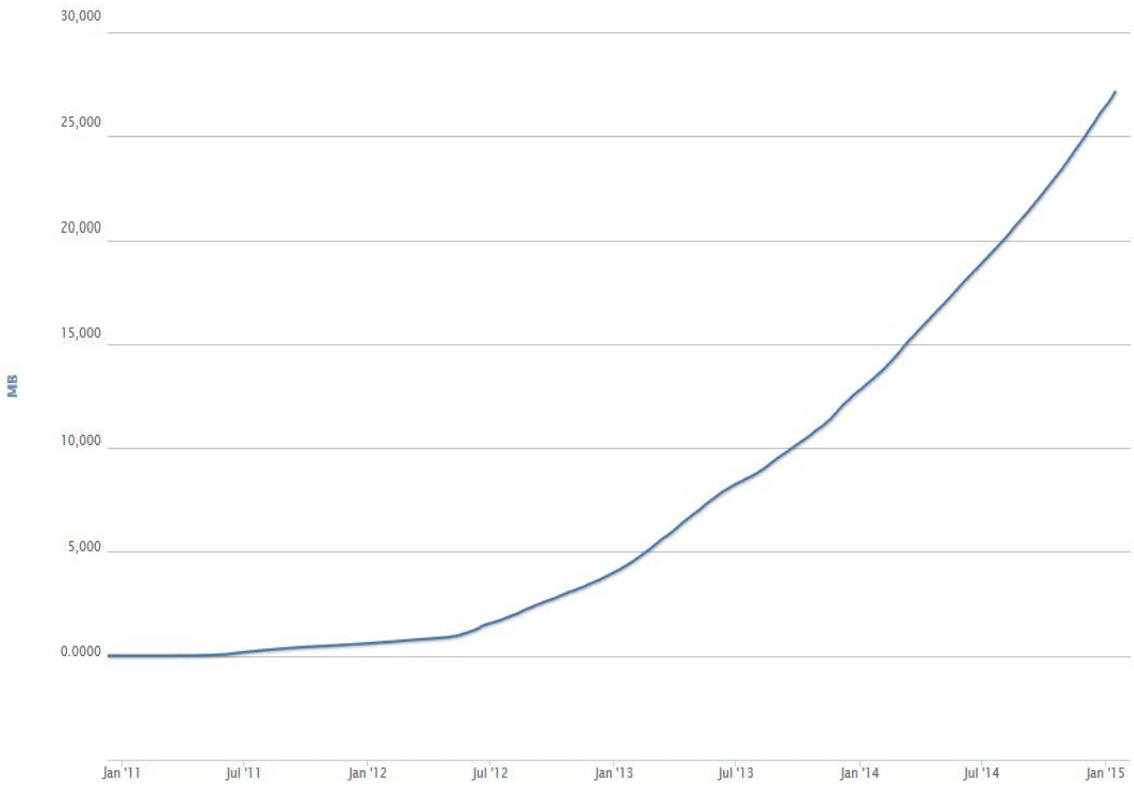
Obrázek 1: Znázornění vztahu mezi transakcí a blokem



Zdroj: Bitcoin.cz (2014)

Řetěz bloků je záznam všech transakcí, které se od spuštění sítě kdy staly. Pro srovnání je to obdoba účetní knihy od založení firmy. Lze dohledat kolik Bitcoinů má každý uživatel v určitém okamžiku na svém účtu nebo lépe na své adrese (ta je navázaná na veřejný klíč uživatele). Řetěz bloků je velmi dobře zabezpečen pomocí kryptografických algoritmů a proto je v dnešní době nemožné změnit jakoukoliv jeho část. Aktuálně (leden 2015) je velikost celého řetězu bloků přibližně 26 GB. (Smith, 2015) Na následujícím obrázku je zřetelně vidět, že velikost samotného blok chainu exponenciálně roste. V lednu roku 2014 byla velikost celého blok chainu pouze 12,5 GB.

Obrázek 2: Velikost blok chainu v MB

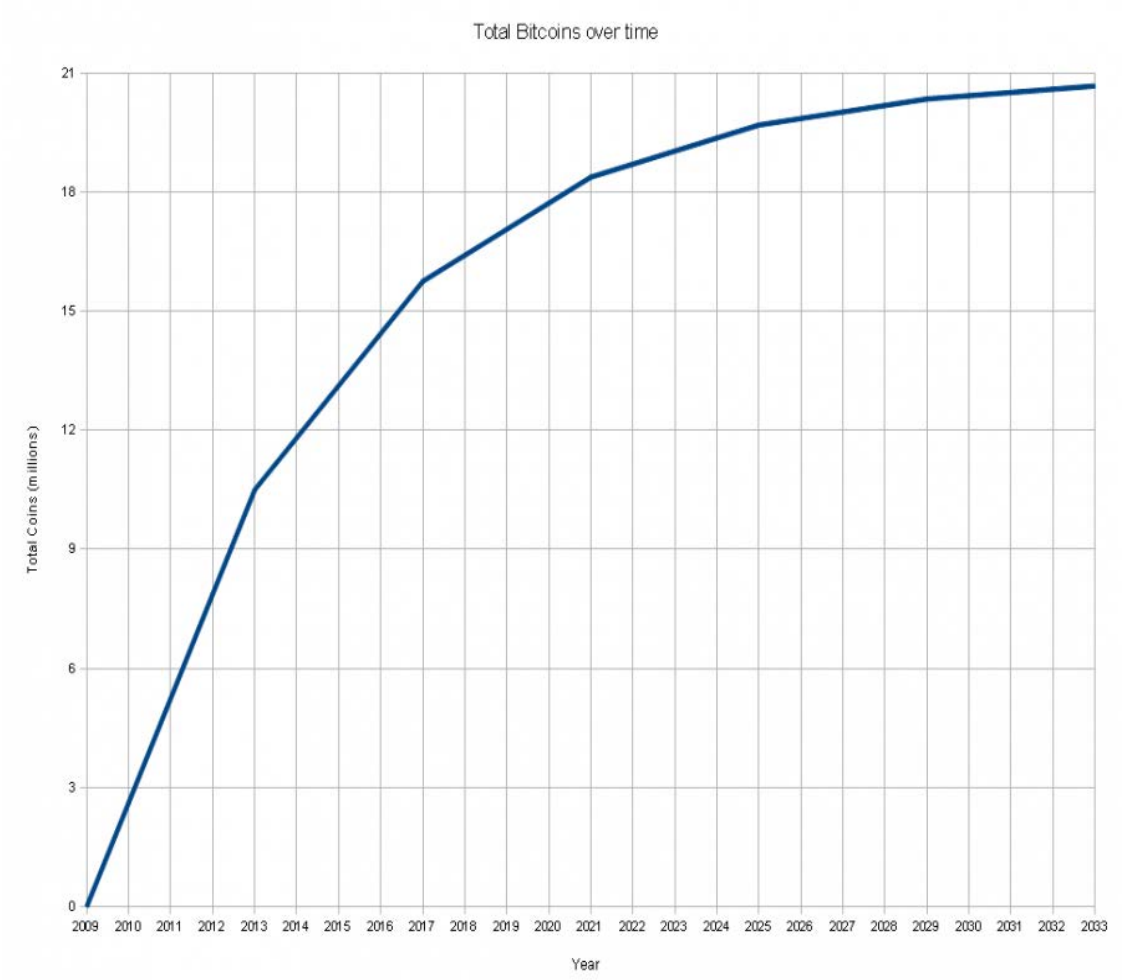


Zdroj: Blockchain.info (2015)

Samotné bitcoiny jsou vytvářeny tak, že někteří uživatelé sítě tzv. minéři (podrobnému vysvětlení tohoto pojmu bude věnována jedna z dalších podkapitol) řeší kryptografický problém spojený s vytvářením nových bloků a poskytováním tzv. proof of work (česky důkaz vykonané práce). Tento proces samozřejmě vyžaduje velký výpočetní výkon použitých strojů a je jednoduše škálovatelný aby bylo možné zaručit, že nový blok bude vytvořen každých 10 minut.

Jako odměnu za vytvoření nového bloku bude jeho tvůrci připsán na jeho Bitcoinovou adresu (lze říct účet) připsán určitý počet Bitcoinů. Tomuto procesu se říká Mining tzv. těžení neboli vytváření nových Bitcoinů. Jinou cestou se totiž nové bitcoiny nedají do oběhu vložit. Obnos takto vytvořených peněz lze velmi dobře předpovědět (to je způsobeno tím, že každý blok musí být vytvořen v určitém časovém intervalu) a celkový počet peněz k vytěžení je omezen na 21 milionů BTC. Tento objem bude vytěžen do roku 2140. A však už v roce 2033 bude vytěženo 99% všech Bitcoinů. Z toho vyplívá, že těžba ale neprobíhá lineárně, zezačátku se těží mnohem rychleji. Tento trend je znázorněn na následujícím obrázku.

Obrázek 3: Předpokládaný počet Bitcoinů v čase



Zdroj: Blockchain.info(2015)

S tím souvisí i to, že náročnost řešeného kryptografického problému je čím dál tím větší a proto je potřeba zvyšovat i výpočetní výkon jednotlivých strojů, které se o vytěžení snaží. Dříve bylo možné těžit s pomocí běžných počítačů a to pomocí výkonu jejich procesorů. Ovšem s postupem času se náročnost zvyšovala a už bylo nutné zapojit i výkon grafických karet. Tento krok byl zklamáním pro všechny majitele grafických karet NVidia a naopak uživatelé s kartami od výrobce AMD měli důvod k radosti. Bylo to dáno tím, že grafické karty typu Radeon byly mnohem lépe uzpůsobeny pro výpočet SHA256d hashovacího algoritmu, který Bitcoin používá. Dalším stupněm při těžení byli speciální zakázkové čipy (ASIC) s velmi nízkou spotřebou. Jejích výhoda spočívá v tom, že pro těžení poskytují několikanásobně vyšší výkon než nejmodernější a nejpokročilejší grafické karty. Velký nárůst ASIC čipů v síti znamenal rychlé zvýšení náročnosti a

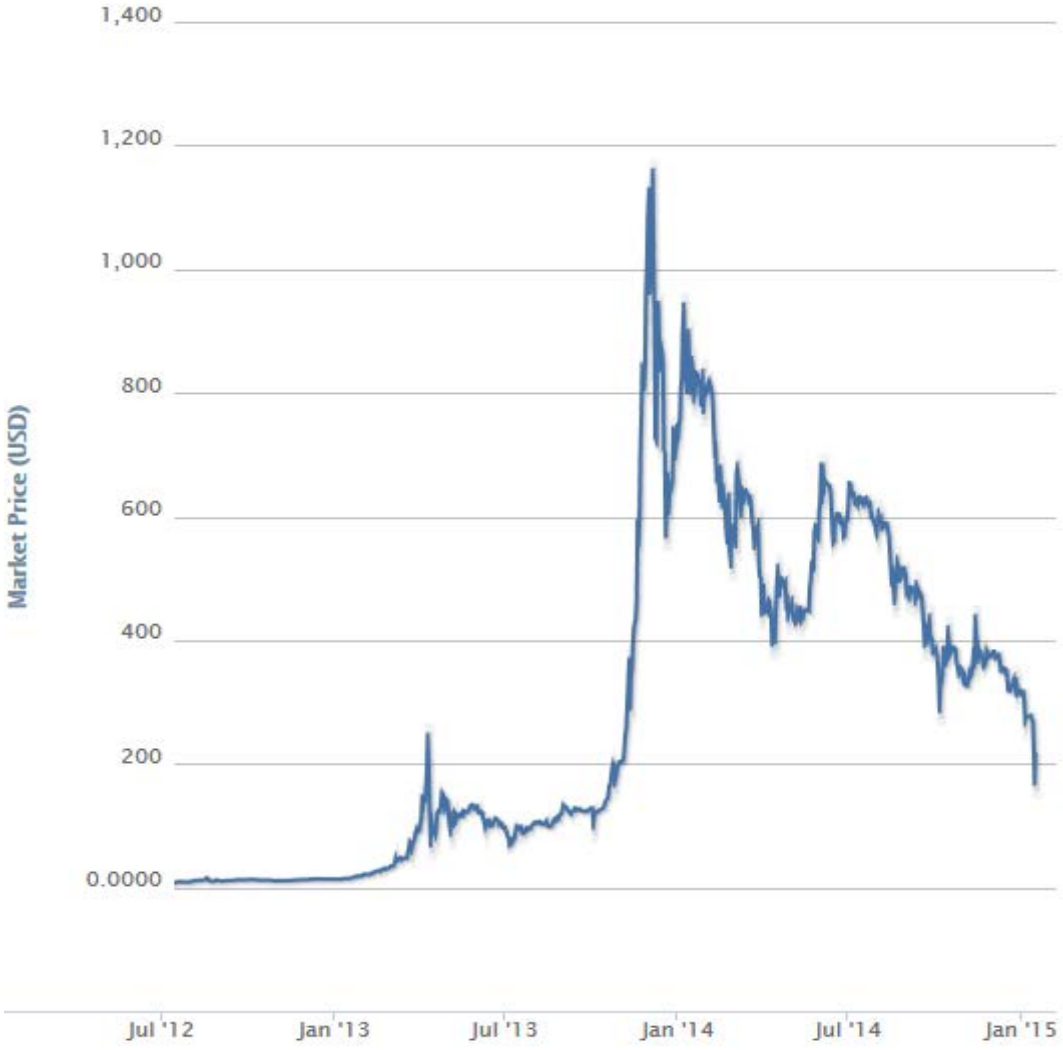
vyřazení efektivního těžení s grafickými kartami. (Javůrek, 2013) Investice do takového čipu se pohybuje v řádech tisíců dolarů.

Protože bitcoiny nemohou být bez těžení tzv. „odpracování“ vytvořeny, je jich pouze omezená zásoba a jsou zabezpečeny kryptografickými algoritmy, lze na ně nahlížet jako na digitální verzi zlata. Můžou být také směněny za různé zboží, služby a v neposlední řadě za tradiční měny. Bitcoin také nabízí ze své podstaty decentralizované měny mnohem menší transakční poplatky při převodu peněz mezi uživateli než standartní bezhotovostní převody mezi bankami a také zaručuje velkou míru anonymity. To je ovšem vykoupeno nemožností vzít transakci zpět.

Pro bitcoiny fungují burzy jako např. pro akciové trhy. Střetá se tu poptávka a nabídka a vytváří se kurz např. vůči americkému dolaru nebo euru. Aktuální kurz je 1 BTC = 219 USD. (Online, 2015) Tento stav znázorňuje obrázek níže. Je zde také vidět velké výkyvy. Například začátkem prosince roku 2013 byl kurz přibližně 1151 USD za 1 BTC. Následní prosincové propady roku 2013 jsou vysvětlovány skutečností, že čínské směnárny přestaly přijímat vklady v čínských Juanech. Podle některých komentářů jde o důsledek zásahu čínských úřadů, kdy bylo zakázáno nabízet služby spojené s virtuální měnou Bitcoin a zaznělo varování před možnými riziky. (Novotný, 2014) V první řadě se jedná o problém že, tato měna není řádně pojištěná tak jako jsou pojištěny např. běžné vklady v komerčních bankách. Je zde také riziko, že vklad může být odcizen a to například hackerským útokem. Tento případ se stal u jedné české bitcoinové peněženky (Bit cash). Věřitelé by neměli dovolání ani v případě neprovedení transakce nebo také kdyby byla celá měna zrušena.

Další velký propad v únoru 2014 zapříčinil krach největší a nejznámější japonské burzy MTGox. Příčinou tohoto krachu byl technický problém, který byl ovšem znám od roku 2011 a lze se před ním chránit. Podle názorů odborníků by každý obchodník, který se zabývá transakcemi s Bitcoiny, ať už jde o převod nebo směnu, měl používat dodatečnou softwarovou ochranu. Tato ochrana by měla mít podobu opětovné kontroly ID transakcí, tak aby se předešlo duplicitám. Burza MTGox takovou ochranu neměla (majitelé burzy zprvu tvrdili, že takovou ochranu burza má, ale nikdy se tento fakt neprokázal) a to zapříčinilo její krach. (Polesný, 2014)

Obrázek 4: Hodnota Bitcoinu v čase (USD)



Zdroj: Blockchain.info (2015)

4. KRYPTOGRAFICKÝ ZÁKLAD

Pro detailnější pochopení všech principů, na kterých je měna Bitcoin založená je nutné zařadit kapitolu týkající se kryptografie a nejdůležitějších algoritmů. V dalších částech této kapitoly jsou podrobněji popsány pojmy jako např. hashovací algoritmus nebo algoritmus digitálního podpisu. Také je zde zařazen slovník nejdůležitějších pojmů, které se v souvislosti s Bitcoinem používají, protože většina pojmů nemá český ustálený překlad a využívají se anglické pojmy nebo zkratky.

4.1. Kryptografie

Kryptografie se zabývá matematickými metodami se vztahem k takovým aspektům informační bezpečnosti, jako je důvěrnost, integrita dat, autentizace entit a původu dat. Ve starším chápání to byla především disciplína, která se zabývala převedením informace do podoby, v níž je obsah této informace skryt. Jejím úkolem je tedy především učinit výslednou zprávu nečitelnou i v situacích, kdy je plně prozrazená, zachycená třetí – nepovolanou – stranou. (Vondruška, 2000)

Kryptografie je věda, která se zabývá psaním zpráv šifrovaným způsobem, tak aby byla zpráva utajena před neoprávněnými osobami. Dělá se obvykle přeložením textu do nějaké formy kódovaného textu. Profesionální kryptografie se snaží ochránit nejen předávanou zprávu, ale také klíč podle, kterého je zpráva šifrována a také celý krypto systém. (Tilborg, 2005)

Podle jiné definice je kryptografie algoritmický proces převádění otevřeného textu nebo vlastní zprávy do šifrovaného textu nebo zprávy na šifře založené. V tomto případě odesílatel i příjemce dopředu zná algoritmus šifrování a tak může být šifrovaná zpráva vrácena opět do své originální podoby. V šifrované podobě nemůže být zpráva čtena nikým, pouze zamýšleným příjemcem. Proces převádění šifrovaného textu do své původní podoby se jmenuje dešifrování. (Afeez, 2015)

4.1.1. Předpoklady kryptografie

Tři základní a nejdůležitější aspekty kryptografie používané měnou Bitcoin:

1. **Šifrování a dešifrování** – algoritmy zaměřující se na převod otevřeného textu na text šifrovaný a obráceně jsou používány k zajištění důvěrnosti dat.
2. **Hašovací funkce** – algoritmy, které vytváří unikátní otisk zpráv a jsou používány jako forma ověření integrity dat.
3. **Schémata digitálních podpisů** – algoritmy které zaručují, že daná osoba opravdu poslala danou zprávu. To znamená, že řeší problém autentifikace.

4.1.2. Šifrovací a dešifrovací algoritmy

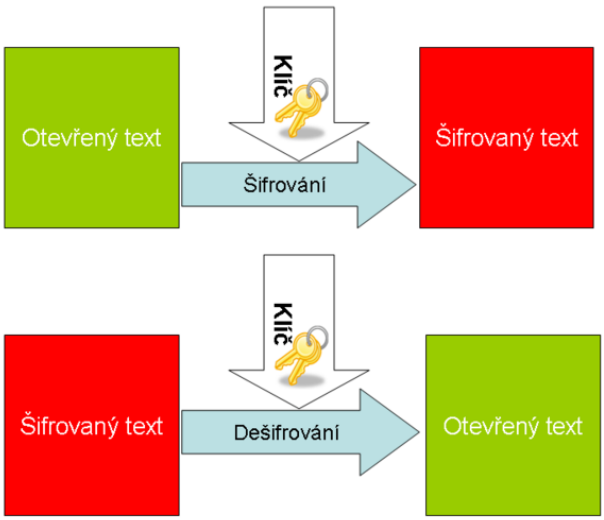
Šifrování je převod otevřeného textu na text šifrovaný, založený na nějakém kódovém textu (slovu) a provádí se postupnou aplikací šifrovacího algoritmu.

Historicky bylo šifrování prvním využitím kryptografie. V dnešních dobách jsou tyto algoritmy široce používány v každodenním životě, tzn. k zabezpečení osobních či důvěrných informací na serverech, dále pak zabezpečení dat při přenosu přes internet a v neposlední řadě i k ochraně dat a souborů na osobních počítačích.

Bez šifrovacích a dešifrovacích algoritmů by jakákoliv ukradená data mohla být zlodějem snadno přečtena a nejspíše i zneužita. Proto bylo sestaveno velké množství šifrovacích algoritmů, ale pouze malá část z nich odolala rozvíjejícím technologiím a strmému nárůstu výpočetního výkonu. Mezi nejúspěšnější algoritmy patří ty, které jsou založeny na klíči. Klíč je jednoduše parametr, který umožnuje proces šifrování a dešifrování. Dnešní moderní na klíči založené kryptografické techniky se dělí na dvě třídy. A to právě podle klíče na symetrické a nesymetrické (veřejný a soukromý).

Šifrování pomocí symetrického klíče je založeno na principu, kdy stejný klíč je použit jak pro zašifrování, tak pro dešifrování. Druhá třída, která využívá nesymetrického principu, využívá jeden klíč k zašifrování a druhý k dešifrování, ten je matematicky odvozen z klíče k zašifrování.

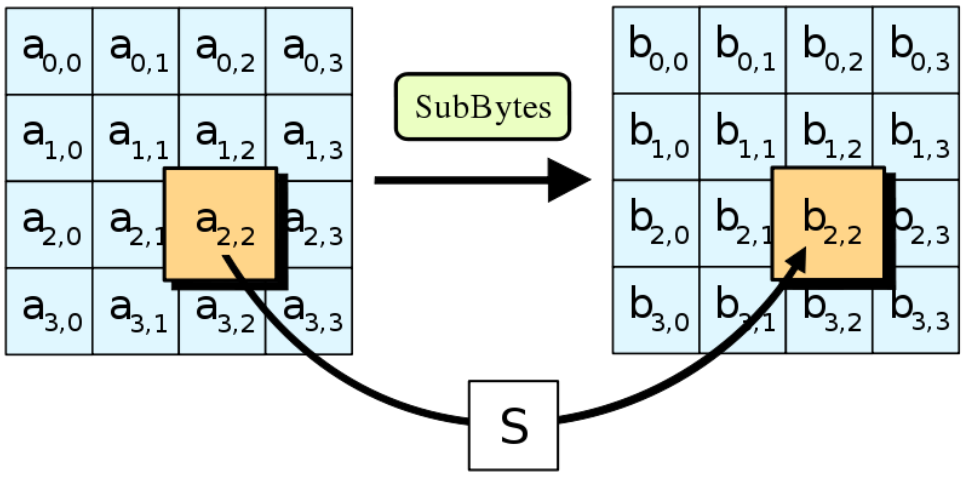
Obrázek 5: Symetrické šifrování



Zdroj: http://home.zcu.cz/~vsyr/

Hlavní podstatou šifrování se symetrickým klíčem je, jeden sdílený klíč, který se využívá, jak k šifrování tak dešifrování dat. Nejpoužívanější současný standard, který se využívá pro symetrické šifrování, je AES algoritmus. Advanced Encryption System je v různých literaturách označován také jako Rijndael. Toto jméno je odvozeno, jako u velkého počtu ostatních šifrovacích algoritmů, ze jména samotného autora. V tomto případě je to spojení jmen dvou odborníků Joan Daemen a Vincent Rijmen. V roce 2001 byl tento algoritmus potvrzen americkým Národním institutem pro standardizaci a technologie (NIST) jako platný současný standard. Do této doby se používal poněkud starší algoritmus DES (Data Encryption Standard). Délka klíčů v tomto algoritmu byla 128, 196 a 256 bitů. (Syrovátková, 2011)

Obrázek 6: Mapování vstupu na výstup



Zdroj: wikimedia.org

Velká výhoda symetrického šifrování spočívá v tom, že je v zásadě velmi rychlé a lze ho využít k šifrování dat velkých objemů a to řádově v GB. Na druhé straně velkou nevýhodou tohoto šifrování je pro velké množství aplikací právě využití sdíleného klíče. Uživatel, který data zašifroval má možnost data i dešifrovat, ale protože spolu komunikují většinou dvě strany, je nezbytné, aby byl klíč nějakým bezpečným způsobem předán. To ovšem nelze vždy zaručit i kvůli neznalosti jednotlivých uživatelů. Dalším aspektem bezpečnosti použité šifry je také závislost na kvalitě zvoleného klíče, měl by být dostatečně komplexní a dostatečně náhodný, pokud není, šifra se stává velmi snadno prolomitelnou.

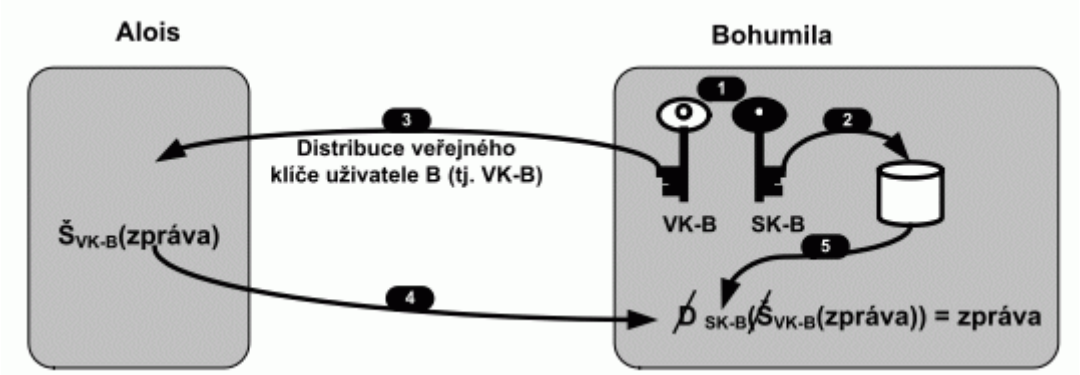
Symetrické šifry lze rozdělit na tyto podkategorie:

1. **Proudové šifry:** V tomto případě se šifruje postupně bit za bitem. Každý bit je tedy zvlášť zašifrován a později i zvlášť dešifrován. Složení do původní podoby je možné až po kompletním rozšifrování.
2. **Blokové šifry**: Tato podkategorie je více rozšířená a spočívá v rozdělení bitového sledu na jednotlivá bitová slova. Ty jsou pak doplněna bitovou šifrou takovým způsobem, aby měla všechna bitová slova stejnou velikost.

Další skupinou šifer jsou šifry asymetrické. Tato část šifer nepoužívá pouze jeden tajný šifrovací klíč, který by byl následně sdílen mezi odesílatelem a příjemcem. V tomto případě se vždy využívá dvou šifrovacích klíčů. První klíč pro zašifrování a další pro dešifrování. Protože u digitálního podpisu můžeme operace zašifrování a dešifrování u

některých šifer zaměnit, nepoužívá se u asymetrických šifer výraz šifrovací a dešifrovací klíč, nýbrž hovoříme o veřejném a soukromém klíči. Pravděpodobně nejznámější asymetrický šifrovací algoritmus je algoritmus RSA.

Obrázek 7: Šifrování pomocí veřejného a soukromého klíče



Zdroj: Floops.cz

Pokud chce odesílatel Adam zašifrovat zprávu příjemci Bobovi asymetrickou šifrou, tak si Bob musí vygenerovat dvojici klíčů. A to veřejný klíč (PublicKey – B) a soukromý klíč (PrivateKey – B). Po té je třeba, aby si Bob uložil svůj klíč na bezpečné úložiště. Muže to být třeba čipová karta, pevný či flash disk. Tento klíč nesmí být prozrazen. Druhý z klíčů, tzn. veřejný klíč (PublicKey – B), může Bob klidně sdílet s okolím. To znamená poslat třeba po třetí osobě Adamovi. V okamžiku kdy Adam disponuje veřejným klíčem Boba, může tímto klíčem zašifrovat zprávu Bobovi. V posledním kroku pak Bob, jako příjemce zprávy, použije svůj soukromý klíč k dešifrování zprávy a získá tak původní zprávu.

Skutečností je že, při šifrování založeném na asymetrických algoritmech je relativně jednoduché za pomoci veřejného klíče zašifrovat text. Naopak jako velmi obtížný se jeví proces, kdy se pomocí veřejného klíče a zašifrované zprávy snaží útočník získat původní zprávu.

Dalším důležitou vlastností tohoto typu algoritmů je délka šifrovacích klíčů. Pro algoritmus RSA je spodní hranice, kdy se délka klíče uznává jako bezpečná 1024 bitů. V dnešní době se používají i mnohem delší klíče a to např. 2048 nebo 4096 bitů. Existují i jiné asymetrické algoritmy. Dnes se často mluví o algoritmu ECC – Elliptic Curve Cryptography (eliptické křivky). Obecně se míní, že z bezpečnostního hlediska odpovídá

1 024 bitů dlouhému RSA klíči 160 bitů dlouhý ECC klíč, přičemž výpočetní náročnost je srovnatelná.

Jiným algoritmem je Diffie-Hellmanův (DH) algoritmus. Ten se vůbec nehodí k nějakému asymetrickému šifrování, ale k bezpečnému ustavení tajných klíčů či sdílených tajemství. Aby Alois s Bohumilou mohli komunikovat symetrickou šifrou, tak se nejprve za využití algoritmu DH dohodnou na tajném klíči, aniž by museli organizovat nějakou tajnou schůzku.

Oba nejprve vygenerují dvojici: veřejné a soukromé DH číslo. Vzájemně si pak vymění (např. volně přes Internet) svá veřejná DH čísla. Obě strany jsou následně ze znalosti svého veřejného a soukromého DH čísla a veřejného DH čísla svého protějšku schopny spočíst sdílené tajemství. Od tohoto tajemství je pak snadno možné nějakou transformací odvodit symetrický šifrovací klíč, který se použije pro šifrování vzájemné komunikace např. algoritmem AES. Diffie-Hellmanův algoritmus hojně využívá např. IPsec.

Bez ohledu na délku klíčů obecně platí, že asymetrické šifrovací algoritmy jsou výpočetně mnohem náročnější než symetrické algoritmy. (Online, Základy šifrování: symetrická a asymetrická kryptografie, 2011)

4.1.3. Hashovací algoritmy

Hašovací funkce jsou silným nástrojem moderní kryptologie. Jsou jednou z klíčových kryptologických myšlenek počítačové revoluce a přinesly řadu nových použití. V jejich základu jsou pojmy jednosměrnosti (synonymum: jednocestnost) a bezkoliznosti. Jednosměrnost znamená, že z M lze jednoduše vypočítat h (M), ale obráceně to pro náhodně zadaný hashový kód H je výpočetně nezvládnutelné.

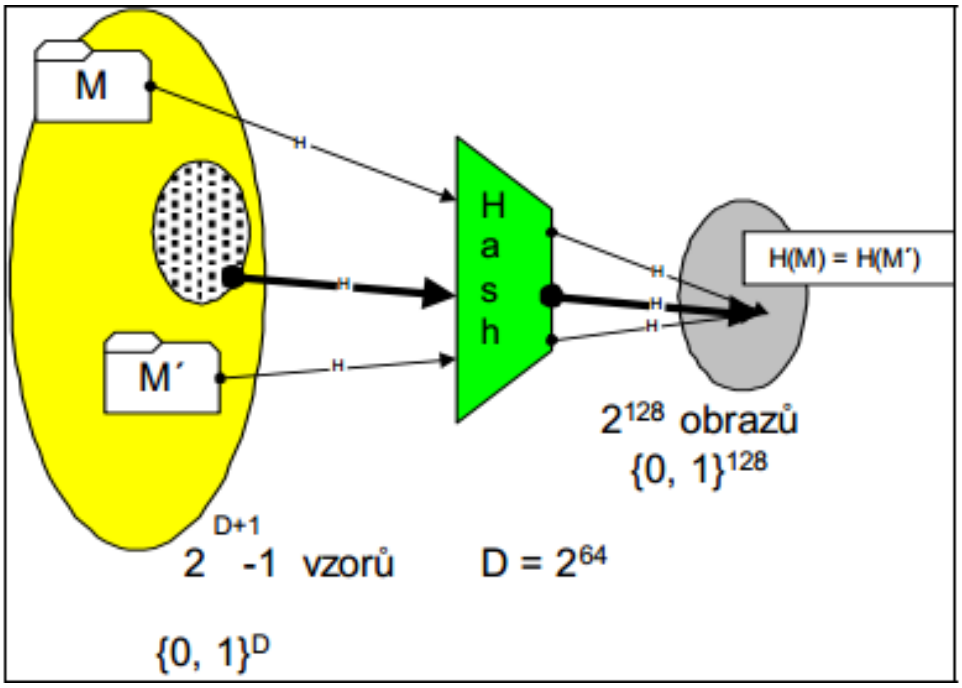
Kryptografické hashovací funkce berou jako vstup řetězce libovolné délky a transformují je na krátké výstupní řetězce pevné délky. V informatice termín hashovací funkce odkazuje na funkci, která stlačuje řetězec libovolné délky do řetězce pevné délky. Například u hashovacích funkcí MD5/SHA-1/SHA-256/SHA-512 je to 128/160/256/512 bitů. (Klíma, 2005)

Obrázek 8: Princip hashovací funkce

Šifrovací hashovací funkce jsou rozděleny do tzv. keyed a unkeyed hashovacích funkcí, podle toho, zda používají tajný parametr nebo tajného klíče. (Klíma, 2005) Tyto hashovací algoritmy jsou často používány k zajištění integrity dat. Pokud se některá část dat nebo souboru změní v důsledku chyb nebo úmyslným zásahem, výsledek hashovací funkce se neshoduje s tím, který byl vytvořen z původních dat.

V ideálním případě by hashovací algoritmy měly být bez kolizí. Bezkoliznost (stručnější označení pro "odolnost proti kolizi") požaduje, aby bylo výpočetně nezvládnutelné nalezení libovolných dvou různých (byť naprosto nesmyslných) zpráv M a M´ tak, že h(M) = h(M´). Pokud se toto stane, říkáme, že jsme nalezli kolizi. Bezkoliznosti, kterou jsme právě popsali, říkáme bezkoliznost 1. řádu nebo jednoduše bezkoliznost. Bezkoliznost se, jak jsme uvedli, zásadním způsobem využívá k digitálním podpisům. Nepodepisuje se přímo zpráva, často velmi dlouhá (u MD5/SHA-1/SHA-256 prakticky až do délky D = 2^{64}-1 bitů), ale pouze její haš. Můžeme si to dovolit, protože bezkoliznost zaručuje, že není možné nalézt dva dokumenty se stejnou hashí. Proto můžeme podepisovat hash. (Klíma, 2005)

Obrázek 9: Transformace vzoru na obraz



Zdroj: (Klíma, 2005)

Možných zpráv je mnoho ($1 + 2^1 + ... + 2^D = 2^{D+1} - 1$) a hashovacích kódů málo (u MD5 například pouze 2^{128}). Musí proto existovat ohromné množství zpráv, vedoucích na tentýž hashový kód - v průměru je to řádově 2^{D-127}. Kolizí tedy existuje ohromné množství. Pointa je v tom, že nalezení byť jediné kolize je nad naše výpočetní možnosti.

Příklad hash funkce SHA256 kdy i změna jednoho znaku ve zprávě z „T“ na „t“ způsobí diametrálně odlišný výstup v podobě hash hodnoty.

Obrázek 10: Příklad hash funkce SHA256

```
Vstup:
    The quick brown fox jumps over the lazy dog
Výstup:
    d7a8fbb307d7809469ca9abcb0082e4f8d5651e46d3cdb762d02d0bf37c9e592

Vstup:
    the quick brown fox jumps over the lazy dog
Výstup:
    05c6e08f1d9fdafa03147fcb8f82f124c76d2f70e3d989dc8aadb5e7d7450bec
```

Zdroj: Vlastní tvorba

Na druhém příkladu můžeme vidět variabilní délku vstupního textu a fixní délku výstupu, který je u této hash funkce roven 64 znakům.

Obrázek 11: Příklad hash funkce SHA256 při variabilní délce vstupu

```
Vstup 11 znaků:
    Lorem ipsum
Výstup 64 znaků:
    a9a66978f378456c818fb8a3e7c6ad3d2c83e62724ccbdea7b36253fb8df5edd

Vstup 280 znaků:
    Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec
    scelerisque enim et nisi commodo, id interdum sapien hendrerit.
    Pellentesque ac elit nunc. Ut erat dui, vulputate et auctor ac,
    egestas eget est. Curabitur ut luctus erat. Maecenas imperdiet vel
    est sit amet volutpat.
Výstup 64 znaků:
    4d115e25bef76e79abb7f94fb19a76c8dd53258d69e29d102bf9385d56d05365
```

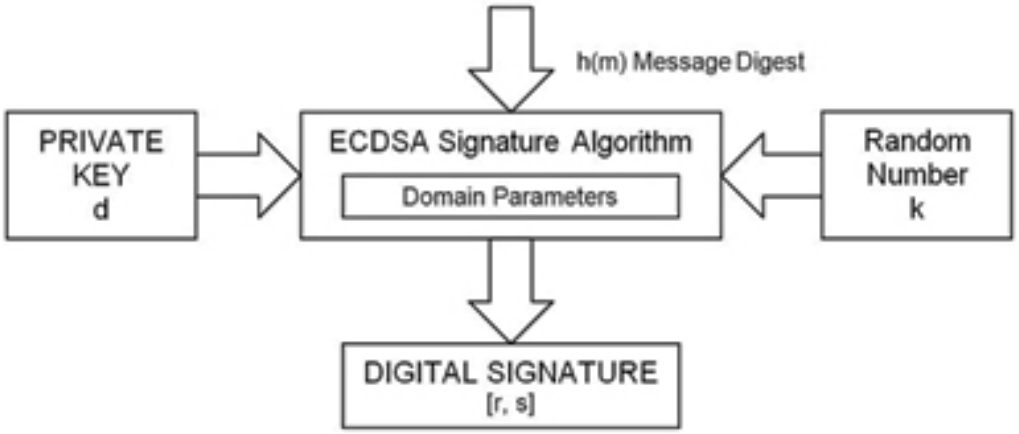
Zdroj: Vlastní tvorba

4.1.4. Digitální podpisy

Schémata digitálních podpisů zahrnují techniky, které se využívají k tomu, aby bylo možné ověřit, kdo danou zprávu odeslal. Typicky má odesílatel privátní a veřejný klíč, který určuje odesílatelovu identitu. Při procesu odesílání je pak vygenerován podpisový znak. Tento znak závisí na zprávě, která ním má být podepsána a také na privátním klíči odesílatele. (Johnson, 2006)

Základní schéma digitálních podpisů, které využívá Bitcoin je tzv. Elliptic Curve Digital Signature Algorithm (ECDSA), je to varianta Digital Signature Algorithm (DSA). Tato varianta používá Elliptic Curve Cryptography (ECC). Schéma kryptografie eliptických křivek bylo představeno v roce 1985 Nealem Koblitzem a Victorem Millerem. Tato varianta vznikla přenesením DSA nad algebru bodů rovinné eliptické křivky. ECDSA algoritmus spočívá v generování náhodného privátního klíče, který se používá pro podepisování zpráv a odpovídajícího veřejného klíče, který je používán pro ověření elektronického podpisu. (Klíma & Rosa, Kryptologie pro praxi – DSA, ECDSA, 2004)

Obrázek 12: Schéma ECDSA algoritmu



Zdroj: embedded.com

Algoritmy digitálních podpisů umožňují uživateli zaručit, že zaslaná data neobsahují žádné chyby, nejsou nijak modifikována a lze ověřit osobu, která zprávu odeslala.

5. BITCOIN

Bitcoin je první decentralizovaná krypto měna. Poprvé byla popsána osobou, která je známá pod pseudonymem Satoshi Nakamoto. Tento auto vydal v listopadu 2008 vlastní práci „Bitcoin: A Peer-to-Peer Electronic Cash System“. Velmi brzo, v lednu 2009, po vydání tohoto teoretického návrhu byla vydána první verze open-source oficiálního Bitcoin klienta.

5.1. Ekonomika a role peněz

Za peníze lze považovat jakýkoliv objekt nebo záznam, který je obecně uznáván k placení za zboží, služby a splacení dluhu v dané zemi nebo socio-ekonomickém kontextu.

Hlavní funkce peněz se rozlišují jako: prostředek směny, zúčtovací jednotka uchovatel hodnoty a občas v minulosti, standardem odložené platby. Jakýkoliv druh objektu nebo prokazatelně zabezpečený záznam, který splňuje tyto funkce, může sloužit jako peníze. (Online, Money, 2012)

Měna obvykle odkazuje na obecně přijímané prostředky směny. Jedná se obvykle o mince a bankovky vydávané konkrétní zemí, které zahrnují fyzické aspekty národní peněžní zásoby. Další částí národní peněžní zásoby jsou vklady u bank. Toto vlastnictví může být převedeno pomocí šeků, debetních karet a dalších forem převodů peněz. Vklady a oběživo jsou obojí peníze a to v tom smyslu, že je lze přijmou jako prostředek platby. (Online, What Are The 6 Characteristics Of Money?, 2014)

V minulosti byly pojmy peníze a měna stejné, měly formu mincí a bankovek. Později, po příchodu částečných rezerv bankovnictví (Online, Fractional-reserve banking, 2015) byly tyto pojmy odděleny. V dnešní době jsou peníze jen malý zlomek toho, co tvoří měnu, většina měny je tvořena úvěry. Dalším důležitým aspektem moderních peněz je to, jak jsou vytvořeny. Původně peníze reprezentovaly hmatatelné zdroje, za které mohli být vyměněny. Byly to například drahé kovy nebo komodity jako obilí či maso. To dávalo penězům jejich unikátní hodnotu. Pokud někdo chtěl vytvořit více peněz, bylo nutné zabezpečit více zdrojů např. těžbou.

Díky příchodu tzv. „Fiat peněz“[1], bylo toto omezení zrušeno. Jakékoli množství peněz mohlo být vytvořeno bez omezení. Hodnota „Fiat peněz“ byla odvozena z nařízení vlády nebo zákonem. V dnešní době je většina, ne-li všechny světové měny, postavena na principu „Fiat měn“.

5.1.1. Funkce a charakteristika peněz

Seznam funkcí peněz podle různých zdrojů. Některé zdroje uvádějí tři základní vlastnosti další čtyři funkce. Následuje seznam charakteristik, které by měly peníze vykazovat.

Základní funkce peněz jsou (poslední funkce je uváděna pouze v některých publikacích): (Anon., 2015)

- Prostředek směny – peníze jsou používány jako přímý nástroj směny za zboží a služby. To umožnuje vyhnout se problémům spojeným s barter systémem.
- Zúčtovací jednotka – peníze jsou používány jako jednotka tržní hodnoty zboží a služeb
- Uchovatel hodnoty – peníze můžou být uchovány a použity bezpečně v budoucnosti
- Standard odložené platby – peníze jsou uznávaným způsobem jak splatit dluhy

Charakteristiky, které by měly peníze vykazovat: (Anon., Peníze, 2015)

- Dělitelnost je jedna z hlavních konkurenčních výhod peněz oproti přímé směně zboží přispívající k jejich univerzálnosti. Každá jednotka může být rozdělena na určitý počet menších jednotek (např. 1 koruna = 100 haléřů, 1 zlatý = 1/2 konvenčního tolaru = 1/20 stříbrné kolínské marky = 60 krejcarů = cca. 11,7g zlata). Z tohoto důvodu se penězi historicky staly právě drahé kovy, které mají takřka dokonalou dělitelnost. Drahé kameny, mušle a další předměty tuto vlastnost postrádají, a proto byly časem jako peníze opuštěny.

[1]Fiat peníze jsou peníze s nuceným oběhem. A označují se jako peníze vytvořené mocí úřední, tedy takové peníze, jejichž hodnota je stanovena zákonem. Slovo fiat je ve významu rozkaz či nařízení (z angličtiny) nebo budiž (z latiny)

- Zaměnitelnost vyjadřuje skutečnost, že účastníkům směny je jedno, v jaké formě či složení peníze vydají/přijmou, dokud platí, že dohromady dají smluvenou cenu. Tuto vlastnost mají například běžné bankovky a mince. Jedna tisícikoruna se rovná dvěma pětistovkám a dvě pětistovky jsou navzájem ekvivalentní. Pamětní mince s uměleckou hodnotou již tuto vlastnost nesplňují.
- Přenositelnost - Peníze (zejména ve formě oběživa) byly historicky navrženy pro snadné přenášení (snadnější než většina hodnot, za něž mohou být směněny). První papírové peněžní substituty se objevily v Číně, kde do té doby jako peníze sloužily měděné mince, kterých bylo pro jejich relativně nízkou hodnotu třeba přenášet i na běžné obchody hodně. To zvyšovalo transakční náklady, a proto došlo k jejich nahrazení přenosnější alternativou.
- Trvalost a trvanlivost - Další vlastností peněz je jejich trvalost a trvanlivost. Peníze nemají navrženu žádnou exspiraci (na rozdíl např. od opčních obchodů), je předpokládána jejich trvalá platnost. Současně jsou vyhotoveny z materiálů, které nepodléhají rychlé zkáze, ale které naopak co nejvíce vzdorují opotřebení.

5.1.2. Digitální měna

Digitální měna je forma peněz, která existuje pouze v elektronické podobě. Typicky to zahrnuje použití počítačových sítí, internetu a systémy k uchování hodnot v digitální podobě.

Elektronické převody peněz, přímé vklady a virtuální měny, to jsou všechno příklady elektronických peněz. V moderním světě jsou digitální měny velmi důležitou částí globální ekonomiky. Mají mnoho různých forem, které jsou přizpůsobené různým potřebám. Nejběžnější digitální měnou jsou elektronické protějšky k reálným světovým měnám jako například Americké dolary nebo Eura, které jsou uloženy v bankách a přeposílány z účtu jednoho subjektu na účet jiného.

Existuje také mnoho tzv. privátních měn (jsou nejčastěji vydávány privátními entitami, které nemají své protějšky z reálného světa. Jako příklady těchto měn lze uvést Lindenovy dolary, Facebookové kredity, nebo dokonce zlaťáky z počítačové hry World of Warcraft. Další důležité rozdělení digitálních měn je založeno na faktu, jestli je měna

centralizovaná či nikoliv tzn., že je známá určitá entita či subjekt (např. stát), která tuto měnu vydává. Pokud taková entita neexistuje, je taková měna označována za decentralizovanou. Typickým příkladem centralizovaných měn jsou např. všechny světové měny (tzv. fiat měny) a také již zmiňované privátní měny. Naproti velkému počtu takových to měn existuje řádově mnohem méně měn decentralizovaných, za zmínku stojí hlavně Bitcoin, Litecoin a Loom. Loom je systém který umožnuje soukromý převod vlastnictví jakéhokoliv aktiva (What is Loom?, 2015).

5.1.3. Využití digitální měny

Funkce peněz v tradiční formě byly již zmíněny v přechozích kapitolách. Digitální měny mají naprosto stejnou funkci jako měny tradiční. Pouze je zde jeden velký rozdíl a to v tom, že digitální měny jsou vlastně nonstop online a proto se také mnohem rychleji a snáze dají převést či s nimi nějak operovat. Samozřejmě jsou zde také problémy, kdy nelze digitální peníze okamžitě převést, protože banky o víkendu platební příkazy neprovádějí a to samé se děje v nočních hodinách, ale výhody těchto peněz jsou nesporné.

Obecně lze říci že, digitální měny či peníze jsou v dnešní době používány úplně stejně jako tradiční peníze. A pokud nelze digitální peníze použít jako např. pokud daný prodejce neakceptuje platbu online nebo nemá platební terminál, lze využít bankomatů nebo přímého výběru v bance k zpětnému převedení digitálních peněz na peníze tradiční (bankovky a mince).

5.1.4. Bitcoin jako kryptoměna

Bitcoin jako digitální měna je revoluční koncept. Projekt vytvořil jedinečnou formu měny, která není závislá na žádném ústředním orgánu. Její hodnota vychází nikoliv z její vlastní hodnoty nebo jakékoliv vyhlášky, ale z její omezené nabídky a matematických algoritmů, které zajišťují její robustnost. Bezpečnost není založena na bezpečném uložení dat v databázích, ale na síle šifrovacích algoritmů.

5.1.5. Základní pojmy spojené s Bitcoinem

- **Bitcoin** – **Bitcoin** je jméno projektu, který byl založen Satoshi Nakamotou s účelem vytvořit první světovou decentralizovanou kryptoměnu. Bitcoin je také název této měny.
- **Bitcoiny** – **Jeden Bitcoin** je označení pro základní jednotku této měny. Bývá zkracována jako BTC podobně jako např. USD nebo EUR. Každý Bitcoin je rozdělitelný na 100 000 000 Satoshi (podle jména zakladatele)
- **Adresa** – Každá **Adresa** je pár ECDSA klíčů používaných uživatelem k přístupu k jeho Bitcoinům
- **Transakce** – **Transakce** je jednotlivá operace pohybu Bitcoinů z jedné adresy na jinou, případně na více adres. Takováto transakce je velmi podobná bankovnímu převodu peněz
- **Blok** – **Blok** je balíček informací obsahující hlavně všechny transakce vytvořené od vzniku předchozího bloku. Obsahuje také referenci na tento předchozí blok.
- **Blok Chain** – **Blok Chain** je seskupení všech propojených bloků od nejnovějšího až po tzv. Genesis blok
- **Genesis Blok** – **Genesis Blok** je blok, který byl tzv. hard-coded do standartního klienta, aby mohl být použit jako startovací bod pro Blok Chain
- **Klient** – Klient je aplikace používaná uživateli k provádění operací v bitcoinové síti
- **Standartní klient** – Standartní klient je aplikace vyvinutá původními vývojáři, kteří pracovali na projektu Bitcoin. Tato aplikace definuje pravidla a standarty jak by měli pracovat a komunikovat ostatní klienti
- **Protokol** – Protokol definuje pravidla, jak klienti komunikují mezi sebou a jak jsou kódovány transakce a bloky.
- **Síť** – Bitcoinová síť je společný název pro všechny aplikace propojené mezi sebou. Tyto aplikace si vyměňují informace o blocích, transakcích a připojených klientech.
- **Peněženka** – je množinou adres vytvořených klientem a uložených lokálně v souboru.
- **Miner** – **Miner** je název pro počítač či výpočetní čip a příslušnou aplikaci, pomocí které lze vytvářet nové bloky

- **Pool** – Pool je webová stránka, která umožnuje Minerům spolupracovat a vytvářet společně nové bloky
- **Směnárna (Exchange)** – Směnárna je webová stránka, která umožnuje směnit bitcoiny za tradiční měny jako dolar nebo euro a naopak.
- **Target** – Target je měřítkem toho, jak těžké je vytvořit nový blok. Haš bloku musí být menší, než je cíl stanovený pro blok

5.2. Jak Bitcoin funguje?

Open-source Bitcoin Project vyvinul aplikaci, která umožňuje provádět všechny operace spojené s virtuální měnou Bitcoin. Tento program je také často uváděn jako standardizovaný klient, tzn. plnohodnotná peněženka. (Bitcoin - P2P digital currency, 2015)

5.2.1. Adresy

Po instalaci je uživateli přidělen pár ECDSA klíčů, který je nazýván Bitcoinovou adresou. Samozřejmě uživatel má možnost vygenerovat si libovolné množství párů takovýchto klíčů tzn. libovolné množství adres. Pro převod Bitcoinů z jedné na jinou adresu je potřeba aby uživatel podepsal specifickou zprávu (tzv. transakci) svým privátním klíčem, který patří k této adrese. Veřejný klíč uživatele je používán kýmkoliv, kdo chce zjistit, jestli daný uživatel má právo disponovat či provádět transakci s danými Bitcoiny.

Ukázka Bitcoin adresy vypadá například takto:

1482xENaxQRDRk7hqY6jAL8j7n9BYxSZZK

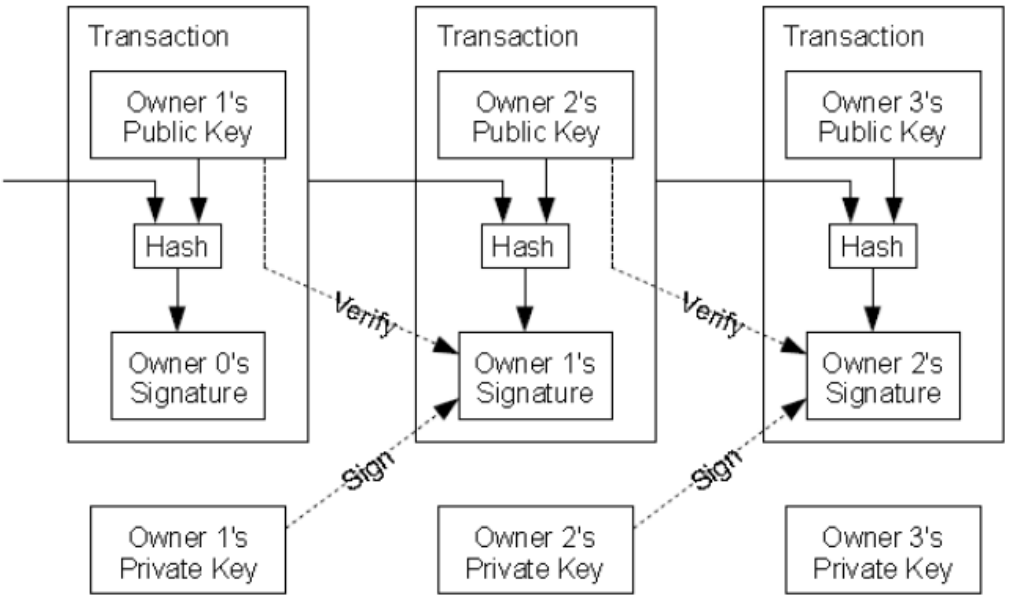
Tato adresa obsahuje informace o tom, pro jakou síť se tato adresa používá, dále pak hash veřejného klíče vlastněného uživatelem a kontrolní součet pro zajištění validity dat [28]. Výše uvedená adresa začíná číslicí 1, a proto lze jednoznačně určit, že se jedná o adresu z „ostrého“ prostředí (tzv. MainNetu). Pokud by adresa začínala písmenem m nebo n je to adresa z testovacího prostředí (tzv. TestNetu).

Taková adresa by vypadala následovně:

msj42CCGruhRsFrGATiUuh25dtxYtnpbTx

5.2.2. Transakce

Obrázek 13: Řetěz transakcí



Zdroj: Nakamoto (2008)

Každá transakce odkazuje na přechozí transakci, obsahuje také veřejný klíč příjemce a podpis odesílatele specifikovaného v předchozí transakci. Transakce může být také definována jako jednoduchá operace, která přesouvá Bitcoiny mezi dvěma různými adresami a dala by se tak přirovnat k bankovnímu převodu peněz.

Při vytváření nové transakce musí napřed uživatel přijmout nějakou transakci na svou adresu. V praxi to znamená, že má na své adrese nějaké Bitcoiny. Dalším krokem je, že uživatel specifikuje adresu příjemce, na kterou chce zaslat Bitcoiny a také samozřejmě také přesnou částku. Klient (viz. 4.1.5) po té vytvoří celou transakci, která obsahuje všechny potřebné údaje. Po té musí být transakce podepsána správným privátním klíčem, který patří k dané adrese. Tento postup zaručuje, že pouze uživatel, který vlastní příslušný privátní klíč přiřazený k dané adrese může Bitcoiny odeslat.

Transakce může vyžadovat jeden nebo více vstupů a obsahuje opět jeden nebo více výstupů. Počet vstupů je limitován pouze počtem starších transakcí a počet výstupů nemusí vůbec korelovat s počtem vstupů. Nejběžnější transakce obsahuje dva výstupy – jedním výstupem je zaslán určitý počet Bitcoinů na adresu zamýšleného příjemce a druhým výstupem je zbývající zůstatek zaslán na adresu uživatele, který celou transakci vytvořil. Pokud obnos Bitcoinů v transakci překročí počet, který je k dispozici

z předchozích transakcí, je takováto transakce považována jako neplatná a síť jí nebere v úvahu. Při zasílání jakékoliv transakce se také vyskytuje tzv. poplatek za transakci. Tento poplatek se automaticky odečte a bude připsán minerovi, který vytvoří blok obsahující danou transakci.

5.2.3. Bloky

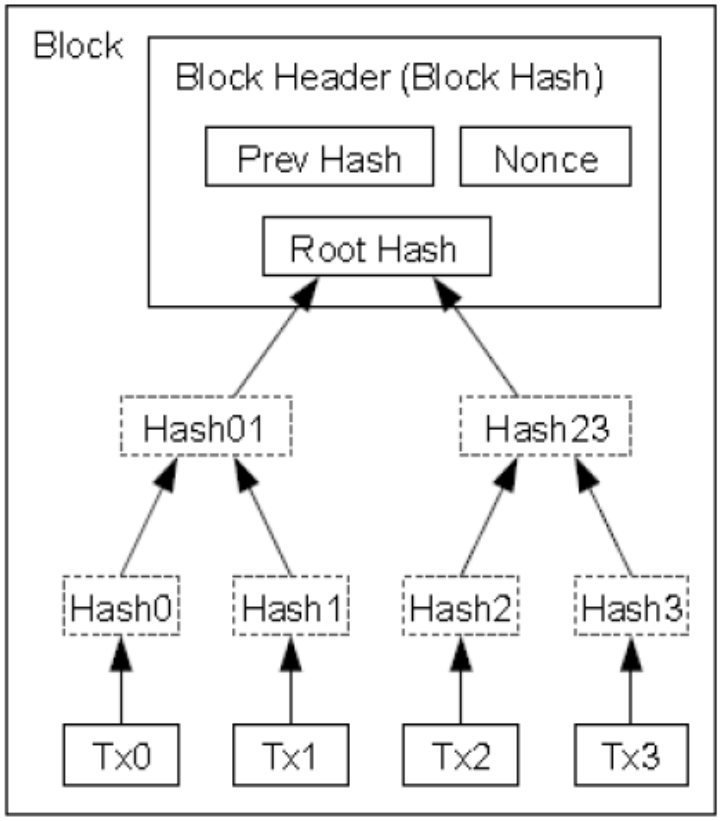
Blok je balík informací obsahující všechny transakce vytvořené napříč Bitcoinovou sítí od okamžiku kdy byl vytvořen předchozí blok. Každý blok obsahuje hlavičku, ve které jsou uloženy všechny meta informace a dále také seznam transakcí, které jsou zakódovány v bloku.

Hlavička bloku obsahuje následující informace (Protocol specification, 2015):

- Verze protokolu bloku (v současné době 1)
- SHA hash hlavičky předchozího bloku
- Tzv. Merkle Root všech transakcí obsažených v bloku
- Časové razítko, kdy byl blok vytvořen
- Hodnota určující target bloku
- Nonce – náhodné znaky, používá se k hashovaní hlavičky bloku

Každý blok odkazuje na blok, který byl vytvořen před ním. To znamená, že všechny bloky dohromady tvoří tzv. Blok Chain (viz. 4.1.5). Z kteréhokoliv bloku se tedy lze dostat až na tzv. Genesis blok, který je prvním a startovním blokem v celé Bitcoinové síti.

Obrázek 14:Struktura Merkle stromu



Zdroj: Nakamoto(2008)

Na obrázku je vidět tzv. Merkle strom (také označován jako hash strom). Všechny listy tohoto stromu obsahují hashe transakcí, které jsou obsaženy v bloku. Díky tomu jak je kořen Merkle stromu počítán je nemožné podvrhnout transakce obsažené v bloku.

Aby mohl být celý blok považován za validní, všechny transakce v obsažené v něm musí být také validní. Dále pak, že musí být propojen s předchozím platným blokem a SHA hash hlavičky bloku musí být menší než hodnota nazývaná target. V návaznosti na poslední podmínku Bitcoinová síť závisí na tzv. proof of work principu aby se zabránilo neoprávněné manipulaci s bloky. Target je hodnota, která je určena jako míra vytváření nových bloků. Průměrně je jeden blok vytvořen každých 10 minut. Target je znovu upravován každých 2016 bloků (tzn. každé 2 týdny). Nejnižší target akceptovaný sítí je:

Obrázek 15: Nejmenší akceptovatelný target

$$0xFFFF * 2^{208}$$

Zdroj: Vlastní tvorba

To znamená, že 1 z každých 2^{32} hashí bloku bude dostatečně malý, aby mohl být přijat. Základní míra targetu bloku je nazývána jako obtížnost bloku. Je odvozena z následujícího vzorce:

Obrázek 16: Vzorec obtížnosti bloku

$$D = \frac{T}{T_{min}}$$

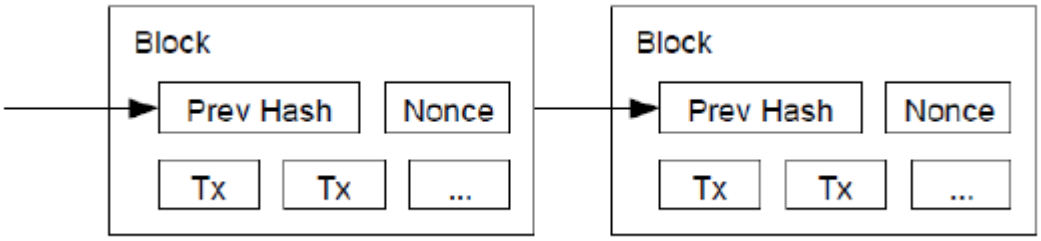
Zdroj: Vlastní tvorba

Kde D je obtížnost, T je současný target a T_{min} je nejnižší možný target. Obtížnost je více vyhovující metrika. Označuje poměr jak těžké je vytvořit blok v porovnání s blokem s nejnižším targetem. Je určeno, že takový blok má obtížnost 1 a poměr targetu je také 1.

5.2.4. Blok Chain

Blok chain je množina bloků navazujících na sebe od prvního tzv. Genesis bloku po nejnovější vytvořený blok. Každý blok v řetězu přesně jednoznačně identifikovatelný svým hashem a každý blok obsahuje hash předešlého bloku. Pokud by tedy mělo dojít ke změně dat v daném bloku, změnil by se také hash bloku. Taková situace ale nemůže nastat, a to právě protože všechny bloky na sebe postupně navazují a obsahují hashe předešlého bloku.

Obrázek 17: Blok chain



Zdroj: Nakamoto(2008)

Protože Bitcoin využívá blok chain lze udržovat decentralizované „účetnictví“. Jednotliví klienti nemusí důvěřovat jeden druhému, a přesto můžou provádět platné transakce. Každý si kdykoliv v blok chainu může ověřit, jestli se někdo nesnaží podvádět či jinak manipulovat s daty.

5.2.5. Mining

Proces vytváření nových bloků se nazývá Mining. Po vytvoření řádného bloku je celá hlavička bloku hashována pomocí funkce SHA256. Pokud je výsledek menší než aktuální hodnota targetu, blok je propagován přes síť, aby byl zařazen jako další platný blok do blok chainu. Pokud ne Miner změní hodnoty in hlavičce bloku, zejména jde o nonci, časové razítko a Merkle root a počítá nový hash.

Jelikož proces Miningu vyžaduje velký výpočetní výkon je vytvoření řádného bloku odměněno určitým počtem Bitcoinů. První transakce v každém bloku je transakcí speciální a je nazývána jako tzv. coinbase transakce. Nevyžaduje žádné vstupy, ale její výstup je roven sumě všech poplatků zaplacených ve všech transakcích, které jsou součástí bloku. Je zde také připočítána dodatečná odměna, která se s časem snižuje. Proto je systém odměn za vytvoření bloku také zodpovědný za počáteční distribuci mincí prostřednictvím sítě. První 4 roky fungování Bitcoinu bylo vytvoření bloku ohodnoceno vytěžením 50 nových Bitcoinů. Po uplynutí této doby byla odměna o polovinu snížena. Tento princip se opakuje každé 4 roky. Tato odměna podporuje rychlý růst výpočetního výkonu sítě a s tím i velkou stabilitu sítě.

Je zajímavé poznamenat, že v počátcích byla velká část výpočetního výkonu zajišťována OpenCL grafickými kartami počítačů (Mining hardware comparison, 2014). V dnešní době už těžení zajišťují FPGA (Programovatelné hradlové pole - Field Programmable Gate Array) zařízení. (Perry, Bitcoin alternative designed for NVIDIA",

http://bitcoin.stackexchange.com/a/1525/323, 2013) V omezeném množství se vyskytují na trhu už i ASIC (zákaznický integrovaný obvod - Application Specific Integrated Circuit) čipy, které mají řádově ještě vetší výpočetní výkon než FPGA zařízení.

5.2.6. Peer-to-peer výměna informací

Každý uzel v Bitcoin síti si vyměňuje informace o všech transakcích, blokách, upozorněních a IP adresách dalších známých uzlů ve všemi ostatními uzly (peery). Standartní klient ale propaguje pouze náhodnou podmnožinu transakcí a to většině připojených uzlů v intervalu jedné minuty. Pouze jeden uzel (peer) obdrží celou množinu informací. (Moore, 2013) Jelikož transakční zprávy neobsahují informace o žádném uzlu (tyto informace jsou mezi uzly odesílány, ale neukládá se seznam IP adres) a pouze náhodná skupina uzlů (peerů) obdrží informaci o nové transakci lze obecně říci, že šíření zprávy v síti je nedeterministické.

Všechny informace (kromě ECDSA klíčů) jsou sdíleny každému uzlu, který o to požádá, což zaručuje, že všechny uzly mají relevantní informace pro zajištění bezpečnosti sítě.

Z přechozích tvrzení je tedy možné vyvodit, že Bitcoinová síť je plnohodnotný peer-to-peer systém, který může fungovat sám o sobě a bez použití jakékoliv centrální autority.

5.3. Infrastruktura Bitcoinu

Infrastruktura Bitcoinu je složena z několika základních komponent. Jádrem této infrastruktury je Bitcoinová síť, která je složená z množství uzlů (aktivních Bitcoinových klientů). Každý uzel je zodpovědný za uchovávání své vlastní kopie blok chainu a seznamu transakcí, které ještě nejsou zařazeny do bloku. Uzel také ověřuje zprávy, které přijímá, a to zejména jejich platnost.

Druhou základní částí Bitcoinové infrastruktury jsou mineři. Tyto aplikace komunikují přímo s uzlem připojeným do sítě, ale neukládají žádné informace spojené s Bitcoinovou sítí. Spoléhají pouze na informace, které jim poskytne uzel, ke kterému jsou právě připojeny. Jejich hlavní činností je vytváření nových bloků pomocí brute-force výpočtu možných hodnot Nonce. Po nalezení správné hodnoty předávají hodnoty zpět uzlu, který sestaví řádný blok a tuto skutečnost rozesílá broadcastem do Bitcoinové sítě.

Opět je dobré uvést, že většina minérů je připojena do sítě skrz tzv. pool. To znamená webovou aplikaci, která sdružuje minery a tím i jejich výpočetní výkon. Tato aplikace má pak na starost i přerozdělení odměny za vytvoření nového bloku a to podle poměru vykonané práce.

Třetí částí této infrastruktury jsou všechny aplikace, které jsou využívány k převodu nebo uchování hodnoty Bitcoinu a využívají ostatní informace o Bitcoinu. Tyto aplikace nejsou potřebné k tomu, aby Bitcoinová síť fungovala, ale jsou potřebné pro její samotné uživatele. Za zmínku stojí hlavně aplikace zabývající se výměnou Bitcoinu za tradiční měny nebo ostatní kryptoměny. Těmto aplikacím se souhrnně říká burzy nebo směnárny.

5.3.1. Bitcoin ekosystém

Bitcoin ekosystém lze definovat jako všechno co obklopuje infrastrukturu Bitcoin. Hlavně se jedná o subkulturu vytvářenou uživateli Bitcoinu. Nejvíce využívaná místa k výměně a sdílení informací jsou např. Bitcoin fórum (Anon., 2015), Bitcoin Wiki (Anon., Bitcoin, 2015), IRC kanály a v neposlední řadě Bitcoin StackExchange (Bitcoin Beta - Stack Exchange , 2015).

5.3.2. MainNet vs.TestNet

Protokol, který zajišťuje funkčnost Bitcoinu může být s úpravami také použit pro chod i jiných kryptoměn s podobnými charakteristikami. Tyto měny se pak nazývají jako tzv. AltCoins. V takovém případě je pak nutné změnit software klienta a vytvořit nový Genesis blok pro alternativní měnu. Pak už je možné začít vytvářenou novou měnu. Protože už bylo vytvořeno mnoho měn na podobném principu jako je Bitcoin standartní Bitcoin blok chain a celá síť je nazývána jako MainNet. (Anon., Testnet, 2015)

Je ovšem nutné aby každá taková měna měla i svůj TestNet. TestNet Bitcoinu je jedna ze dvou nejpopulárnějších AltCoins. Sdílí většinu stejných charakteristik jako má MainNet, ale její účel je úplně jiný. Hlavním účelem je totiž testování nových funkčností a vlastností před vlastním uvedením do MainNetu. Ze zmíněných vlastností vyplývá že TestNet Bitcoiny nemají žádnou hodnotu a jsou volně převáděny mezi uživateli bez nějaké protihodnoty nebo služby. Navíc TestNet je upraven a nastaven tak bych měl výrazně větší hodnoty targetu než samotný MainNet. To v realitě znamená, že tester nebo vývojář s průměrným výpočetním výkonem má možnost vytvořit nový blok. V MainNetu taková možnost už dnes neexistuje, protože potřebný výkon na vytvoření bloku je tak

veliký, že jsou uživatelé obvykle seskupení do poolu, jak už bylo popsáno v kapitole 4. 3. (Anon., Testnet, 2015)

Dále existuje velké množství AltCoins, které byli od příchodu Bitcoinu vytvořeny. Jedná se hlavně o LiteCoins, SolidCoins, GeistGelt, I0coins a další. Velká část z nich se liší od Bitcoinu jen ve velmi malých detailech jako např. časem mezi vytváření jednotlivých bloků nebo jiným použitým algoritmem pro hashování bloků.

5.4. Příklady využití Bitcoinu

V této kapitole je uveden základní popis softwaru využívaného pro manipulaci a uchovávání Bitcoinů. Protože je mnoho dostupných klientů byl vybrán oficiální BitcoinQt.

5.4.1. Instalace a nastavení Bitcoin klienta

Standartní klient je volně ke stažení na domovské stránce celého projektu Bitcoin tzn. http://bitcoin.org/. K dispozici je v několika variantách, záleží, jaký operační systém používá PC, na které chceme klienta instalovat. V tomto příkladu je zvolen instalační soubor .exe v 64-bitové verzi, protože na testovacím PC je naistalován Windows 7 v 64-bitové kopii. Velikost instalačního souboru je v řádech MB a je oproti aktuální velikosti celého blok chainu naprosto zanedbatelná. Velikost blok chainu je aktuálně něco kolem 26 GB a přenosová rychlost při stahování blok chainu (viz. níže) je relativně malá (stovky Kb/s), to znamená, že synchronizace klienta může trvat minimálně hodiny.

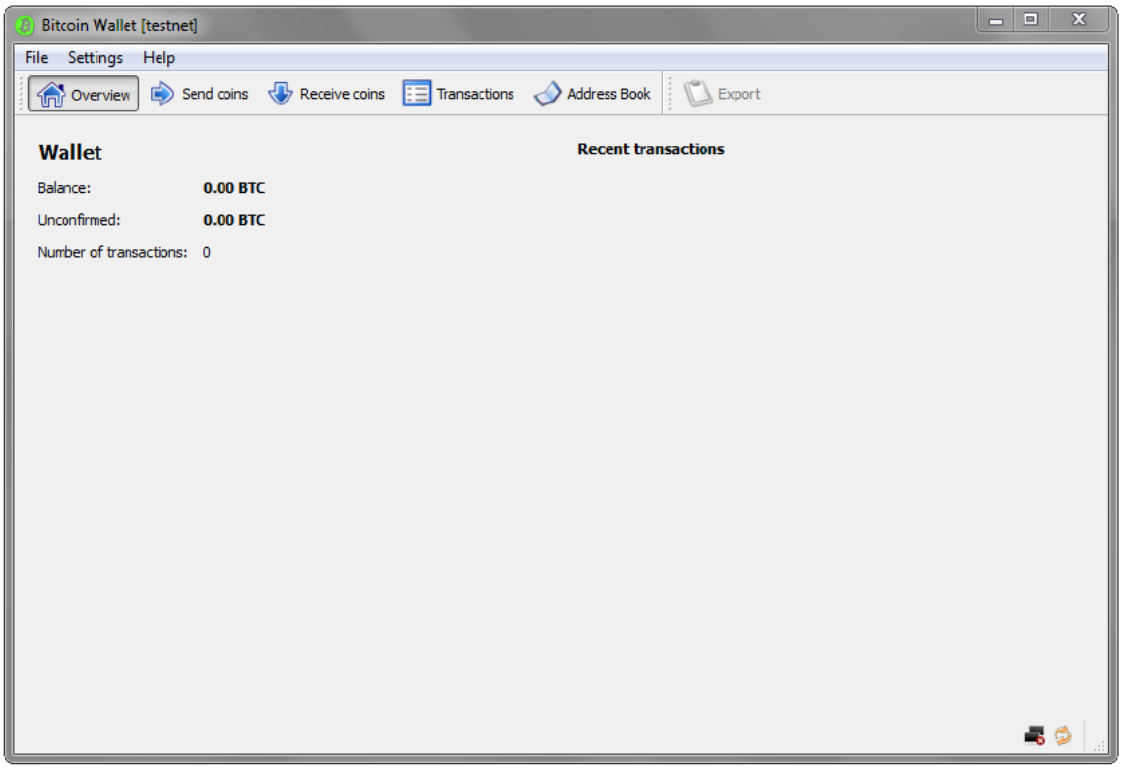
Po stažení instalačního souboru vypadá úvodní obrazovka instalace asi takto:

Obrázek 18: Instalační obrazovka

Zdroj: Vlastní tvorba

Po instalaci se tato aplikace sama otevře a uživatel je přivítán základním oknem kde může vidět přehled svých prostředků a další záložky. V celé této kapitole je použit standartní klient, ale při spouštění této aplikace je doplněn přepínač –„testnet“. To znamená (viz. 4.3.2), že je využito testovací prostředí a testovací Bitcoiny, které nemají hodnotu.

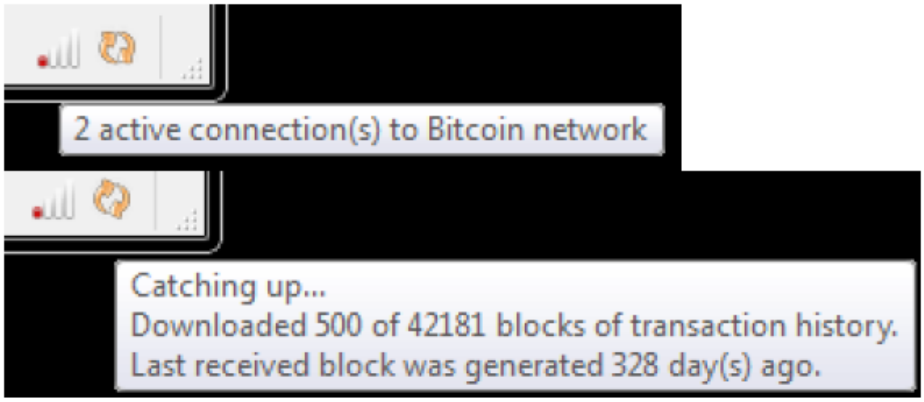
Obrázek 19: Přehledové okno aplikace



Zdroj: Vlastní tvorba

V prvním kroku se klient připojí k síti a začne se synchronizací blok chainu. Tento proces je poměrně zdlouhavý, protože velikost blok chainu pořád narůstá. Aktivní synchronizace je znázorněna na následujícím obrázku.

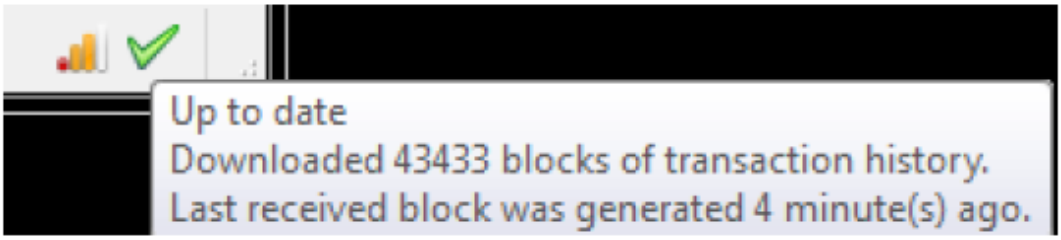
Obrázek 20: Synchronizace blok chainu



Zdroj: Vlastní tvorba

Jakmile bude klient plně synchronizován se sítí, uživatel může začít posílat nebo přijímat transakce. Uživatel tento stav jednoznačně pozná podle zeleného ukazatele v pravém dolním rohu aplikace.

Obrázek 21: Úspěšně dokončená synchronizace blok chainu



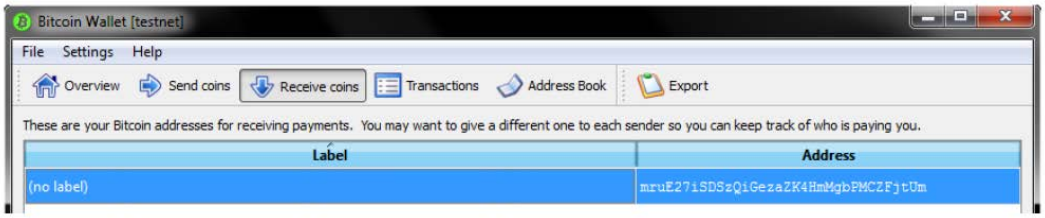
Zdroj: Vlastní tvorba

5.4.2. Příjem transakcí

Pro příjem prvních Bitcoinů je nutné vytvořit a vybrat jednu adresu v peněžence. Tato adresa musí být uvedena v příslušné kartě „Receive coins“. Na následujícím obrázku je tato situace zachycena na příkladu adresy:

mruE27iSDSzQiGezaZK4HmMgbPMCZFjtUm

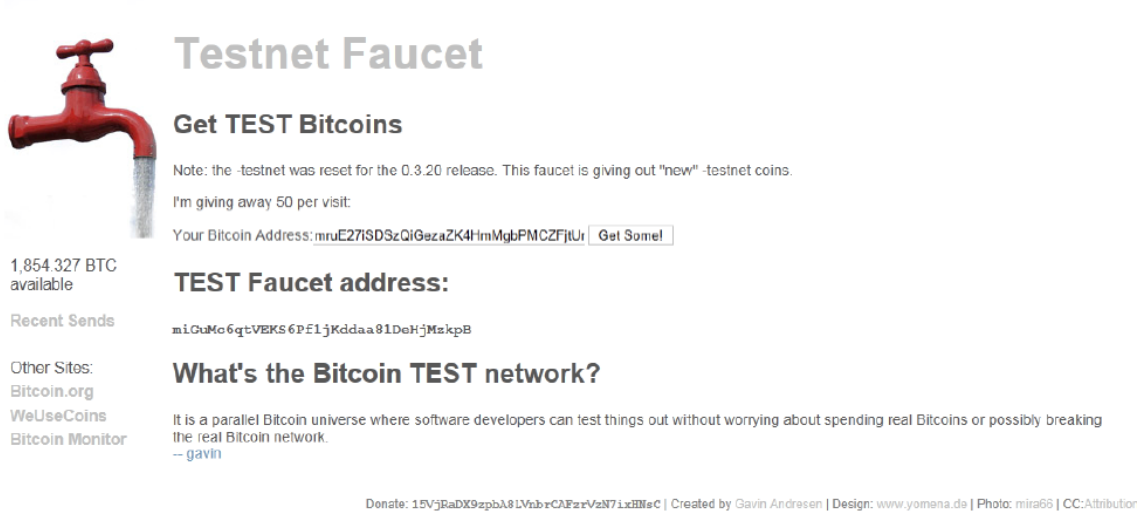
Obrázek 22: Příklad adresy pro příjem Bitcoinů



Zdroj: vlastní tvorba

Tato adresa může být používána pro příjem Bitcoinů z různých zdrojů – lze je nakoupit na burzách či směnárnách a dále pak získat těžením (Mining). V tomto příkladu budou Bitcoiny získány z tzv. TestNet Bitcoin Faucet webové stránky (Andresen, 2015). Tuto situaci znázorňuje následující obrázek.

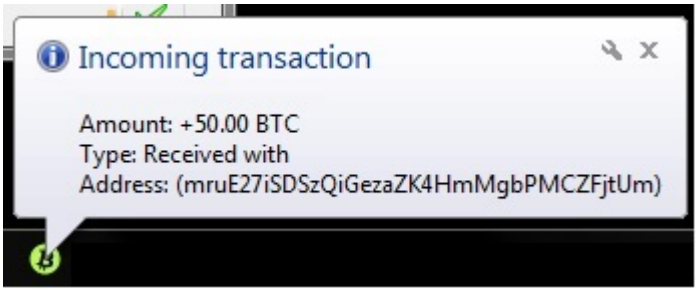
Obrázek 23: TestNet Faucet



Zdroj: testnet.freebitcoins.appspot.com

Pro získání Bitcoinů touto cestou musí uživatel pouze vyplnit svou příjmovou adresu a pomocí této webové stránky je mu zasláno 50 BTC. V několika sekundách je pak zobrazena notifikace o příchozí transakci viz následující obrázek. Podmínkou zobrazení této notifikace je zapnutý klient na pozadí.

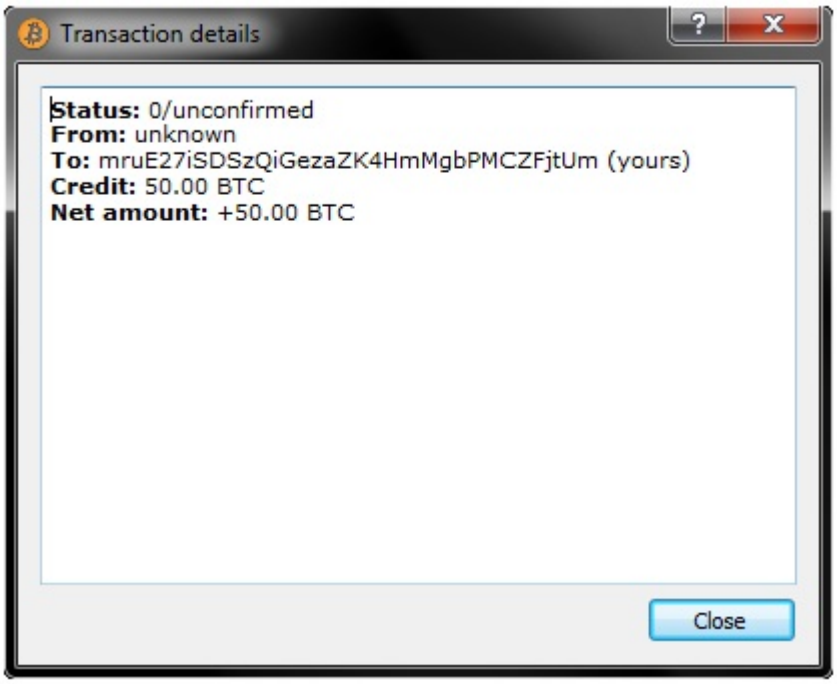
Obrázek 24: Notifikace příchozí transakce



Zdroj: Vlastní tvorba

Detaily transakce si uživatel může prohlédnout v klientovi pod záložkou „Transactions“. Na dalším obrázku je vidět, že z počátku bude mít transakce status „0/unconfirmed“.

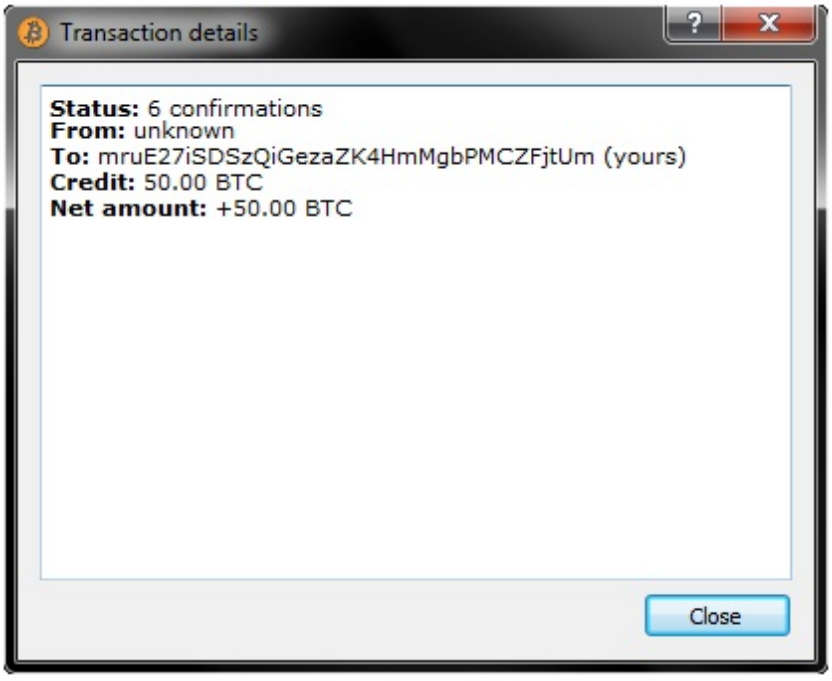
Obrázek 25: Detail nepotvrzené transakce



Zdroj: Vlastní tvorba

Postupně bude probíhat potvrzení dalšími uzly a po dosažení 6 a více potvrzení lze považovat za transakci schválenou tzv. „safe and spendable“.

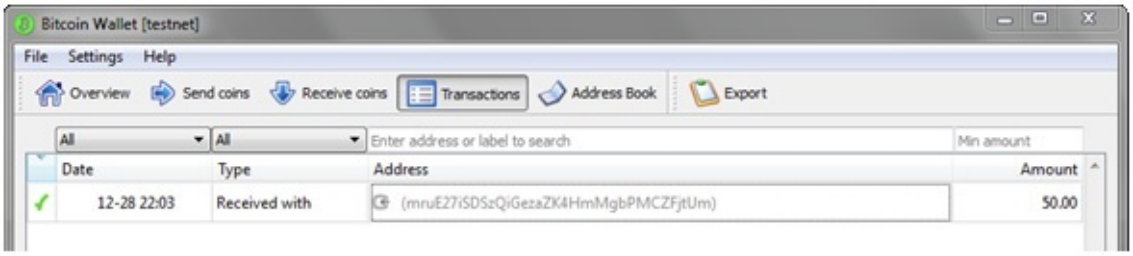
Obrázek 26: Detail potvrzené transakce



Zdroj: Vlastní tvorba

Na posledním obrázku této podkapitoly o přijímání Bitcoinů je vidět konečný stav po úspěšném provedení transakce. V přehledné tabulce jsou k dispozici všechny důležité informace, tzn. datum a čas přijetí transakce, typ transakce, adresa protistrany a částka.

Obrázek 27: Přehled úspěšných transakcí

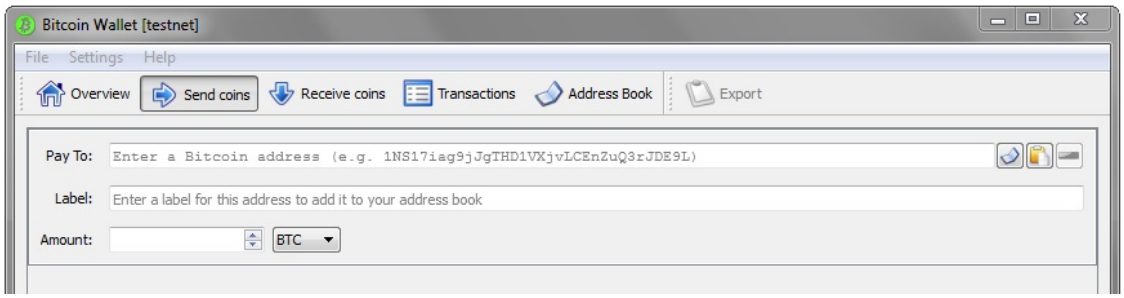


Zdroj: Vlastní tvorba

5.4.3. Odeslání transakce

Opačným způsobem se provádí odesílání Bitcoinů. Jak je vidět z následujícího obrázku musí uživatel vyplnit formulář, který je v záložce „Send Coins“. Tento formulář obsahuje adresu, na kterou chceme Bitcoiny zaslat, dále pak nepovinný popisek transakce a samozřejmě obnos, který chceme zaslat. U tohoto pole lze ještě změnit jednotky z BTC např. na mBTC (0,001 BTC).

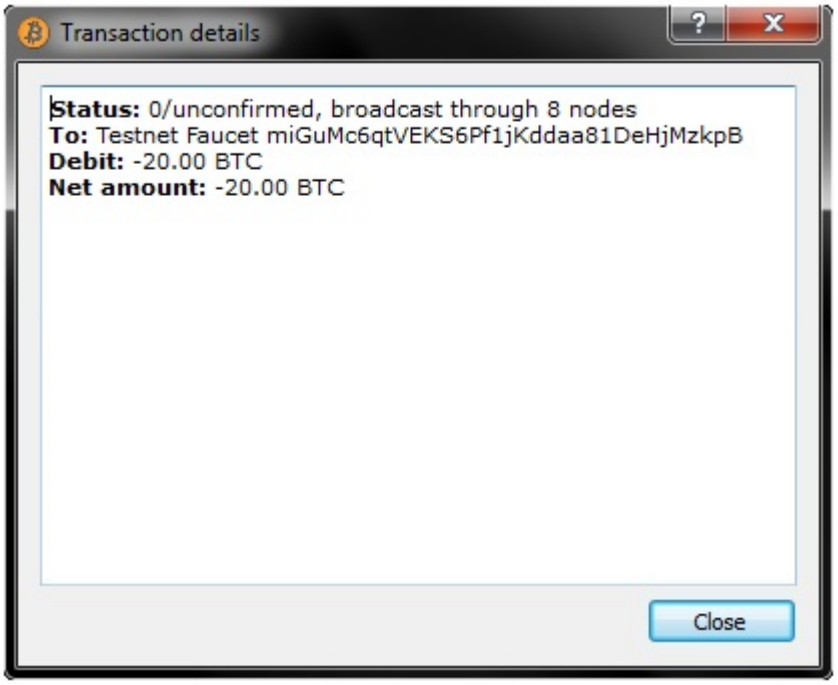
Obrázek 28: Formulář pro odeslání Bitcoinů



Zdroj: Vlastní tvorba

Odchozí transakce proběhne velmi podobným způsobem jako příchozí. V první řadě bude zobrazena notifikace o odchozí transakci. Uživatel si po té může opět prohlédnout kartu „Transactions“, kde lze zjistit podrobné informace o této transakci. Z dalšího obrázku je vidět, že transakci je opět třeba potvrdit. Dále si uživatel může všimnout, kolika připojeným uzlům byla tato transakce odeslána (k broadcastu sítí).

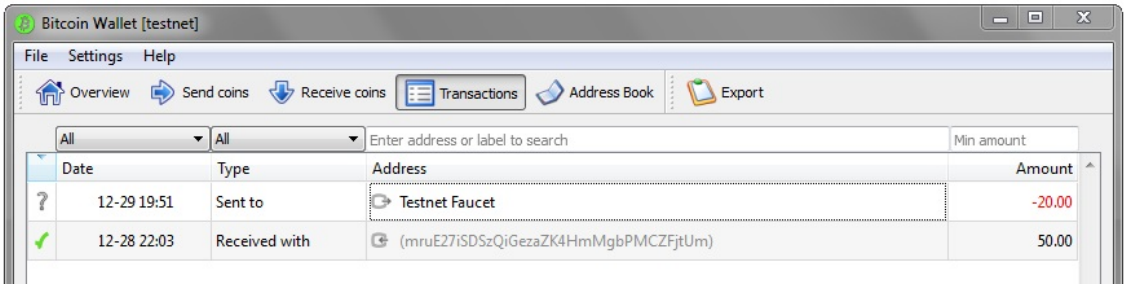
Obrázek 29: Detail nepotvrzené odchozí transakce



Zdroj: Vlastní tvorba

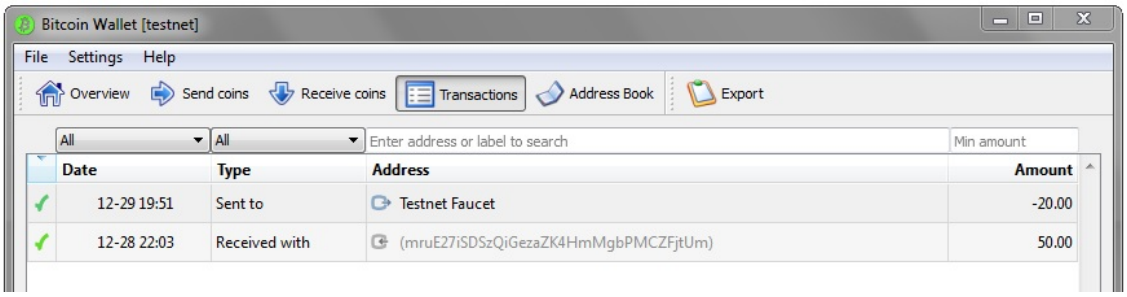
Další obrázek znázorňuje přehled transakcí a otazník u odeslané transakce indikuje, že tato transakce dosud nebyla potvrzena.

Obrázek 30: Přehled transakcí s nepotvrzenou odchozí transakcí



Po dosažení alespoň 6 potvrzení je odchozí transakce opět označována jako „safe to spend“.

Obrázek 31: Přehled transakcí s potvrzenou odchozí transakcí



Zdroj: Vlastní tvorba

6. METODIKA

Hlavním cílem praktické části diplomové práce je analýza a návrh platební aplikace pro virtuální měnu Bitcoin. Tato aplikace by měla pracovat pod operačním systémem Android od verze 4.0. Aby bylo možné takovouto aplikaci možno navrhnout, je nutné správně zvolit software pro analýzu a vývoj aplikace.

V první řadě bylo nutné zvolit vhodný programovací jazyk, ve kterém by se dala takováto aplikace vytvořit. Po důkladné analýze, kdy bylo nutné zohlednit několik aspektů jako např. robustnost, nezávislost na architektuře, přenositelnost a objektová orientace, byl jako vhodný programovací jazyk vybrána Java samozřejmě v kombinaci s ADT (Android Development Tools).

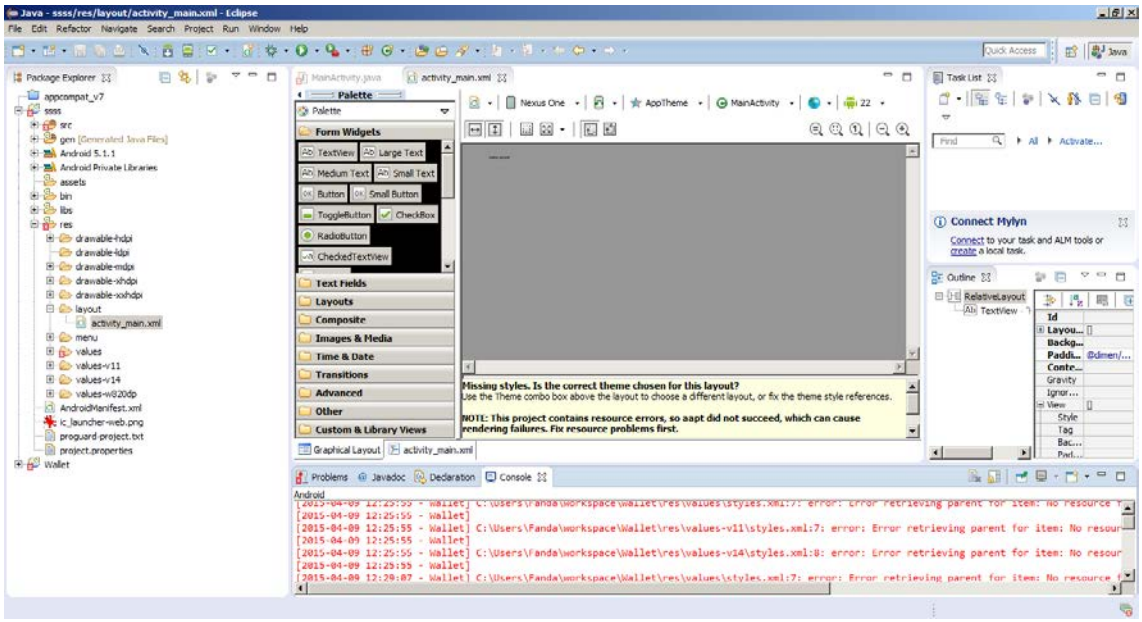
Obrázek 32: Logo Java

Zdroj: thaicreate.com (2015)

Na výběr programovacího jazyka samozřejmě navazuje volba vývojového prostředí. Tato volba byla spíše na předchozí zkušenosti s vývojovými prostředími. Tedy na osobní preferenci, samozřejmě, že toto prostředí musí umožňovat doinstalovat již zmíněný ADT (Android Development Tools). Na výběr máme k dispozici mnoho vývojových prostředí. Eclipse, NetBeans, Intellij IDEA a Android Studio. Mezi nejrozšířenější vývojové prostředí patří Android Studio a Eclipse. Pro nespočetné výhody vývojového prostředí Eclipse, např. Eclipse se nemusí zdlouhavě instalovat (stačí pouze rozbalit .zip soubor) a také je velmi přehledné, bylo zvoleno jako nejlepší a k účelům diplomové práce nejvhodnější. Jediným nedostatkem, který bych mohl uvést je to, že Eclipse není lokalizované do češtiny (i když pro programátory to asi problém není) a dále

pak trochu horší kompatibilita s operačním systémem Windows 7. Je zde problém s nastavováním systémových proměnných, ale tento problém je velmi dobře zdokumentován a lehce opravitelný.

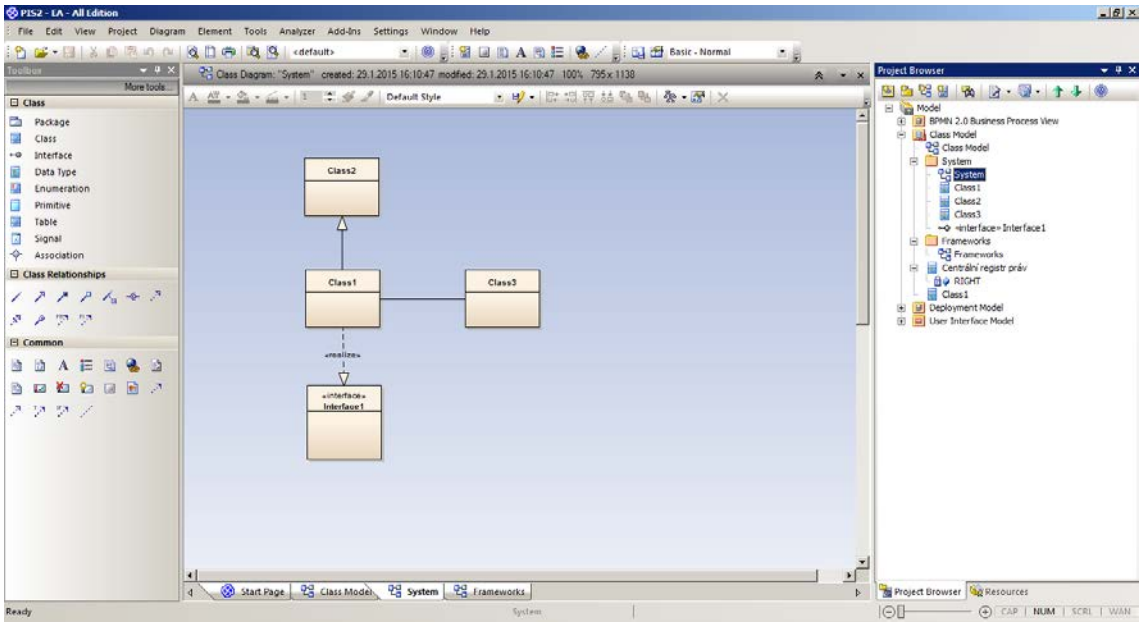
Obrázek 33: Ukázka prostředí Eclipse



Zdroj: Vlastní tvorba

Pro důkladnou analýzu celé aplikace bylo také naprosto nezbytně vybrat komplexní modelovací nástroj. Pro analýzu a návrh takovýchto aplikací se běžně používá nástroj, který podporuje UML notaci (Unified Modeling Language). Opět po předchozí zkušenosti a zvážení kladů a záporů byla zvolena aplikace Enterprise Architect 8. Tato aplikace od společnosti Sparx Systems je kompletní nástroj pro systémovou analýzu a návrh, který pokrývá celý životní cyklus vývoje systému, tzn. od zadání požadavků přes analýzu stavů, návrh modelů, testování a údržbu, vše s využitím diagramů v UML. (Enterprise Architect, 2015)

Obrázek 34: Ukázka modelovacího prostředí Enterprise Architect 8



Zdroj: Vlastní tvorba

6.1. Shrnutí použitých metod

V této podkapitole je shrnutí všech využívaných metod.

1. Pro analýzu a návrh aplikace je využita notace UML. V první fázi jsou definovány požadavky na danou aplikaci, které budou doplněny o slovní komentáře. Na základě tohoto modelu je poté implementována aplikace.
2. Aplikace bude navržena ve vývojovém prostředí Eclipse v programovacím jazyce Java s využitím ADT.
3. Při vývoji aplikaci bude použito některých open-source částí dostupných na internetu. Jedná se hlavně o volně dostupnou knihovnu BitCoinJ, kterou lze při vývoji aplikace peněženky využít.
4. Pomocí User Interface diagramů bude navrženo grafické prostředí aplikace.

7. ANALÝZA A NÁVRH VLASTNÍ APLIKACE

Praktická část diplomové práce je rozdělena do několika kapitol. První kapitola se zabývá analýzou potřebných komponent pro návrh samotné aplikace. Jedná se hlavně bezpečnostní protokoly, šifrovací algoritmy, hashovací funkce a další části nezbytně nutné k samotnému chodu aplikace. Další kapitola se pak věnuje samotnému popisu jednotlivých tříd, v této části jsou také uvedeny požadavky na aplikaci. V dalších kapitolách se pak zabývám návrhem částí aplikace v programovacím jazyce Java. A v neposlední řadě je zde uveden příklad možného grafického zpracování celé aplikace a znázornění v UML pomocí nástroje Enterprise Architect.

7.1. Analýza potřebných komponent – BitCoinJ

BitCoinJ je open source Java implementace Bitcoinového protokolu. Je to velmi dobrý nástroj k navrhování Java aplikací, které by měli komunikovat s Bitcoinovou sítí. Je zde naimplementováno mnoho užitečných částí, které by jinak programátor musel psát od začátku. Pro začátek je nutné naimportovat tuto knihovnu do projektu vlastní aplikace, kterou vyvíjím v Elipse. Po té budu pomocí této knihovny vytvořit základní funkčnosti aplikace. Jedná se zejména o vytváření Bitcoinové adresy, dále pak její uchovávání v peněžence a uložení peněženky do úložiště mobilního zařízení. Další klíčovou funkcionalitou, kterou se budu snažit naimplementovat je navázání připojení k Bitcoinové síti a získání tzv. genesis bloku. V neposlední řadě zde bude ještě ukázka, jakým způsobem lze provést transakci a odeslat tak Bitcoiny na adresu v TestNetu (testovací síť Bitcoinu). Ještě je dobré zmínit, že knihovna BitCoinJ je stále ve vývoji (aktuálně ve verzi 0.3) a z doporučení vývojářů plyne, že by neměla být využívána pro převod velkého počtu Bitcoinů. Ovšem pro účely mé diplomové práce plně dostačuje.

Z dokumentace této knihovny lze celkem dobře vyčíst, že API je rozděleno do čtyř hlavních balíčků.

- **Discovery** – zabývá se komunikací peer-to-peer
- **Store** – obsahuje datové struktury pro uchovávání bloků a blok chainu
- **Examples** – zahrnuje užitečné jednoduché aplikace založené na BitCoinJ (tyto příklady mne inspirovali při vývoji vlastní aplikace)

- **Core** – tato část obsahuje hlavní BitCoinJ třídy a funkcionality, zahrnující třídy pro komunikaci s peer uzly, stahováním blok chainu a odesílání a přijímání transakcí

7.2. Vytvoření Bitcoinové adresy

Pro zaslání nebo příjem Bitcoinů je nutná adresa. Jak už bylo popsáno v teoretické části práce, adresa je odvozena od páru veřejných a privátních klíčů. V tomto specifickém případě jde o specifické klíče podle tzv. ECDSA algoritmu. Ten je založen a obtížnosti najít diskrétní logaritmus eliptické křivky. Pokud by tedy programátor chtěl programovat celý tento algoritmus, musel by mít přesně velmi důkladně nastudovanou příslušnou část algebry, která se tímto problémem zabývá. Naštěstí proto je to tu BitCoinJ knihovna, kde tento algoritmus je již připravený. Tuto funkčnost zajišťuje třída ECKey. Pomocí této třídy lze vygenerovat pár příslušných klíčů. Na následujícím výřezu kódu (obr. 35) je vidět, že nejprve naimportujeme potřebné funkčnosti. Jedná se o ECkey, Address a NetworkParameters.

Ve třídě CreateAddress pak vytvoříme nový EC (elliptic curve) klíč vytvořením instance objektu typu ECkey. Výchozí metoda této třídy toString() je pak přepsaná tak, aby vracela veřejný a privátní klíč v hex notaci.

Po analýze jsem zjistil, že veřejná část klíče generovaná tímto kódem, ale vůbec nepřipomíná platnou Bitcoinovou adresu, kterou zobrazuje klient v uživatelském rozhraní. Formát adresy, který je zobrazován klientem je výsledkem opakovaných hash operací prováděných na veřejném klíči. Aby se z tato část klíče stala platnou adresou je potřeba provést správnou sekvenci kryptografických hashovacích funkcí. Opět je zde již připravená funkce v rámci třídy ECKey, kterou jsem využil. Jedná se o metodu toAddress(). Po vyvolání této metody a určením typu sítě, pro kterou se adresa bude vytvářet, skrze objekt NetworkParemeters, metoda toAddress() vrátí objekt Address. Metoda objektu toString() následně sestaví platnou Bitcoinovou adresu pro zadanou síť. Po zkompilování a vykonání této třídy jsem dostal následující adresu.

mxc4u5LkmC2oAGVkUHpr1WtdJxq8wyuqyJ

Poslední poznámkou k tomuto kódu je výběr příslušné sítě. Ten je zde realizován proměnou „net“, která tzv. „ natvrdo“ nastavena na hodnotu „test“. Pokud bych zde přepsal hodnotu na „prod“ třída by generovala adresu pro MainNet, která začíná jedničkou.

Obrázek 35: Příklad vytvoření nové Bitcoin adresy

```
import com.google.bitcoin.core.ECKey;
import com.google.bitcoin.core.NetworkParameters;
import com.google.bitcoin.core.Address;

public class CreateAddress {

    public static void main(String[] args) throws Exception {

        // použij TestNet
        String net = "test";

        if (args.length >= 1 && (args[0].equals("test") || args[0].equals("prod"))) {
            net = args[0];
            System.out.println("Using " + net + " network.");
        }

        // vytvoř nový klíč
        ECKey key = new ECKey();

        // vypiš klíč
        System.out.println("created key:\n" + key);


        final NetworkParameters netParams;

        if (net.equals("prod")) {
            netParams = NetworkParameters.prodNet();
        } else {
            netParams = NetworkParameters.testNet();
        }

        // Získej platnou adresu z klíče
        Address addressFromKey = key.toAddress(netParams);

        System.out.println(net + "\n" + addressFromKey);
    }
}
```

Zdroj: Vlastní tvorba

7.3. Peněženky a klíče

V předchozí kapitole jsme úspěšně vytvořili adresu potřebnou k zasílání nebo obdržení nějaké sumy Bitcoinů. V tomto kroku je potřeba někde tyto adresy ukládat. Pro ukládání vytvořím peněženku. Peněženka není nic jiného než lokální datový soubor, který obsahuje „serialized“ objekty. Tyto objekty představují ve skutečnosti všechny

Bitcoinové transakce a seznam využitých adres. Na následujících ukázkách kódu bude vysvětleno vytvoření peněženky za využití BitCoinJ knihovny. Dále vytvořím pět adres a vložím je do peněženky. Celá peněženka se pak uloží do souboru na lokální úložiště.

Třída Wallet implementuje rozhraní Serializable. To mi umožnuje ukládat peněženku na disk či jiné trvalé úložiště dat. V konkrétním případě jde o metody loadFromFile(File) a saveToFile(File), které mohou číst a zapisovat do souboru peněženky. Já využiji metody loadFromFile(File), abych zapsal nově vytvořený objekt peněženky do souboru.

7.3.1. Ukládání klíčů pomocí metody addKey

Třída Wallet obsahuje objekt keychain, je to vlastně ArrayList typu ECKey, který je využíván k uchovávání všech párů klíčů v peněžence. Metoda addKey(ECKey) je používaná pro přidávání párů klíčů. Zajímavé je, že v momentální implementaci neexistuje metoda, která by dokázala tyto klíče odebírat. Ono to ale při pohledu z druhé strany dává smysl, protože by pro uživatele či programy nemělo výt jednoduché tyto klíče mazat, speciálně privátní klíče. Privátní klíč je nezbytně nutný k tomu aby se uživatel dostal ke svým peněžním prostředkům. Bez celého páru klíčů uložených v peněžence nebo zálohovaných na jiném médiu jsou všechny Bitcoiny uživatele nenávratně ztraceny.

Kdykoliv si lze ověřit jestli hash nebo daný veřejný klíč má k sobě v keychainu svůj privátní protějšek. To lze za použití následujících metod.

Obrázek 36: Metody pro ověření veřejného klíče a hashe

```
findKeyFromPubHash(byte[] pubkeyHash)
isPubKeyHashMine(byte[] pubkeyHash)
findKeyFromPubKey(byte[] pubkey)
isPubKeyMine(byte[] pubkey)
```

Zdroj: Vlastní tvorba

Opět za zmínku stojí jedna zajímavá věc. Jde o to, že jenom konstruktor třídy Wallet bere objekt NetworkParameters jako argument. Tato věc zajišťuje, že nelze kombinovat transakce z „ostrého“(MainNet) a testovacího (TestNet) prostředí v jedné peněžence. V praxi se toto řeší, viz kapitola 4.4, spouštěním aplikace s různými přepínači. (např. -testnet)

7.3.2. Vytvoření peněženky

Na dalším obrázku je vidět, že výchozí síť je zvolena jako TestNet (řádka 13) a dále je deklarovaný objekt Wallet (řádka 16). Tento objekt bude potřebovat objekt typu File (řádka 17) aby se mohl uložit na disk či jiné datové úložiště. V hlavním try-catch bloku pak inicializuji objekt Wallet tak, aby používal TestNet (řádka 20). Dalším krokem je smyčka, kterou zde píši proto, abych mohl vytvořit oněch pět adres. V podstatě to znamená, že při každé smyčce se generuje nový pár klíčů ve formě ECKey objektu, který je pak přidáván do peněženky (řádka 26).

Obrázek 37: Ukázka třídy CreateWallet

```
003|import java.io.File;
004|import java.io.IOException;
005|
006|import com.google.bitcoin.core.*;
007|
008|public class CreateWallet {
009|
010|    public static void main(String[] args) {
011|
012|        // pracuj s testNetem
013|        final NetworkParameters netParams = NetworkParameters.testNet();
014|
015|       // pokus se načíst peněženku z úložiště, jinak vytvoř novou
016|        Wallet wallet = null;
017|        final File walletFile = new File("test.wallet");
018|
019|        try {
020|            wallet = new Wallet(netParams);
021|
022|            // vytvoř 5 párů klíčů
023|            for (int i = 0; i < 5; i++) {
024|
025|                // vytvoř nový pár klíču a přidej je do peněženky
026|                wallet.addKey(new ECKey());
027|            }
028|
029|             // ulož peněženku na disk
030|            wallet.saveToFile(walletFile);
031|
```

Zdroj: Vlastní tvorba

V této fázi už mám vytvořeno 5 adres, mám je přidané v peněžence a posledním krokem je uložení. Tuto proceduru zajištuje řádek 30 s využitím metody saveToFile(). Nově vytvořený soubor se jmenuje test.wallet.

V na dalších řádkách jsou aplikovány metody, které jsem již dříve zmínil a to například ověření páru klíčů pomocí hashe veřejného klíče. Například na řádce 37 vezmu první klíč z keychain ArrayListu a vypíšu ho (řádka 40), posléze vypisuji všechno co je v této peněžence (řádka 43).

V posledních řádkách (47 – 50) jsem vytvořil malý test, ten mi pak dovolí zjistit, jestli určitý klíč byl přidán do peněženky nebo ne. Nad objektem Wallet tedy spustím metodu isPubKeyHashMine(), která porovná se zadaným hashem všechny hashe veřejných klíčů v mé peněžence. Jelikož jsem použil hash svého veřejného klíče, který jsem si předtím nechal vyzvednout a pojmenoval jako firstKey skončím kladným výpisem Yep, that´s my key. (Ano, to je můj klíč)

Obrázek 38: Ukázka generování klíčů v peněžence

```
031|
032|        } catch (IOException e) {
033|            System.out.println("Unable to create wallet file.");
034|        }
035|
036|        // vyzvedni první klíč z keychain ArrayListu
037|        ECKey firstKey = wallet.keychain.get(0);
038|
039|        // první klíč v peněžence
040|        System.out.println("First key in the wallet:\n" + firstKey);
041|
042|         // celý obsah
043|        System.out.println("Complete content of the wallet:\n" + wallet);
044|
045|        // použití hashe veřejného klíče k ověřenení
046|        // jestli takový pár klíčů mám ve své peněžence
047|        if (wallet.isPubKeyHashMine(firstKey.getPubKeyHash())) {
048|            System.out.println("Yep, that's my key.");
049|        } else {
050|            System.out.println("Nope, that key didn't come from this wallet.")
051|        }
052|    }
053|}
```

Zdroj: Vlastní tvorba

Na dalším obrázku je vidět výstup po kompilaci a spuštění třídy CreateWallet.

Obrázek 39: Výstup při spuštění třídy CreateWallet

```
First key in the wallet:
pub:04cf1b56e809dd615663d918824688264d81dbd3642550a82545df5bfa0dfeb5c1e1ba0d97fa278853e121678d13eb2f7e061281957933e9d4bd03893e80e14e0e
priv:00e657901ce15ec38c2340dd99cda6f1d82ef4b3c90da65c84d6e9ce06a9e

Complete content of the wallet:
Wallet containing 0.00 BTC in:
  0 unspent transactions
  0 spent transactions
  0 pending transactions
  0 inactive transactions
  0 dead transactions

Keys:
  addr:n1QCia54L1MDR4xjQStM48vGnfQf62kh8p pub:04cf1b56e809dd615663d918824688264d81dbd3642550a82545df5bfa0dfeb5c1e1ba0d97fa278853e121678d13eb2f7e061281957933e9d4bde80e14e0e
                                          priv:00e657901ce15ec38c234060e79dd99cda6f1d82ef4b3c90da65c84d6e9ce06a9e
  addr:moMovkB3uGTZrf66f5WddcNVhq6MuZUajJ pub:0423a79d67d612e44d06228bc47c9ac1ead8124929280ac827c2aa301075e27909ec2b5502eee7be40e08fba42a7e5c2daa91b1641fe1c036806a14c27d60
                                          priv:00862ef8769d848ae56eebd211652f72667b353093f5db6139e81b7a787544a039
  addr:mrukZip4D6TG8TxEhhBLkKZvQY4fCPZxZn pub:04c11cdfbc6fbf838a03bd4d49df1957330a4105529054002f87a83d726a632ad7da257c0997aa52fa484382c675d6835f9f78b3c65366a2f3692404de63f
                                          priv:460343ee1e983fd0ad0ca0023f1b1aa4269ba34c7f17e8463856085669c0bc23
  addr:mgro3Ned1KakmVbP8hPf4iDDo6KDTpCzGL pub:046212ef2216273af27185493de1197ee4a21fef46dee0c27f026b765473c0cd1e1d62ea9f1ea2dbb16103b3c96bb31d78e102b2387ac0b03d6726dbb2117
                                          priv:5a42597d63515df1a60ac0090ae55f52e0a8e6ec74e320041cb52605a643c178
  addr:myQqSQpk5mN1e4it1r3dxdyB6Cuu1tWjjo pub:04222d33fb1f89a217804a0400d34b48c08808fbbd22af5e408f04e45597ca3c061ca6dcfe04e4b483c3099caa718e4e5ba6260501b7d24ae9dc44afffe13
                                          priv:00ec4b1abee48a43fe4b12eff1e8fdacabd392554dc12f1cf1156467218cd92189

Yep, that's my key.
```

Zdroj: Vlastní tvorba

Po spuštění této třídy jsem zjistil, že metoda objektu Wallet toString() také vypisuje různé typy transakcí jako např. unspent (neprovedeno), spent (provedeno), pending (probíhající), inactive (neaktivní), a dead (permanentně neaktivní).

7.4. Získání genesis bloku

Dalším krokem je připojení aplikace k peer uzlu Bitcoinové sítě a vyžádání si od něho jeden blok. V mém případě se připojím na lokálně běžící Bitcoinový uzel a získám genesis blok TestNetu. Jen pro připomenutí genesis blok je první v sekvenci blok chainu, který má podobnou strukturu jako spojový seznam (linked list) a obsahuje bloky tak jak šli postupně za sebou. Tyto bloky samozřejmě i samotné transakce.

Každá hlavička bloku obsahuje hash, který je ukazatelem na předchozí blok v blok chainu. Tento ukazatel využívám, když si chci vyžádat blok od peera v síti. Tento princip mi umožnuje procházet celým blok chainem a získat všechny bloky, abych mohl zrekonstruovat celou transakční historii celého systému Bitcoin od jeho založení. Jediný blok, který nemá žádný platný hash ukazatel je genesis blok a to protože žádný přecházející blok nemá. V další kapitole budu popisovat, jakým způsobem lze takový genesis blok získat podle jeho hashe. Samozřejmě tímto způsobem je možné získat jakýkoliv jiný blok nebo procházet blok chainem.

Po analýze jsem zjistil, že lepším přístupem než nechat si vracet od sousedního uzlu pouze jeden blok, je stažení všech bloků a to ve správné sekvenci. Tím pak vlastně zrekonstruuji celý blok chain. Analýza potřebných tříd je též popsána v následují kapitole.

7.4.1. Třída BlockChain a BlockStore

Třída BlockChain sama o sobě umí pracovat přesně s takovým typem dat jako je blok chain. Při použití metody add(Block) je možné přidat jakékoliv množství bloků a třída BlockChain sama ověří platnost bloků a pokusí se přiřadit blokům správné místo v blok chainu. Jak již bylo vysvětleno v přechozích kapitolách s postupem času a se zvyšujícím se celkovým počtem transakcí je blok chain neustále delší. Klient si musí stáhnout, uložit a mít k dispozici vždy celý blok chain. Protože požadavky na volné místo na datovém úložišti a také na operační paměť jsou značné, rozhodli se vývojáři, že BitCoinJ bude mít oddělené skladování bloků od logiky a datové struktury, tímto vzniklo rozhraní BlockStore

Rozhraní BlockStore obsahuje 4 metody, které musí být implementovány každou třídou, která zajišťuje uchovávání bloků. První metodou je put(StoredBlock), druhou pak get(Sha256Hash). Tyto metody umožnují vývojářům přidat a získat bloky. Respektive k tomuto účelu slouží také ty dvě další metody getChainHead() a setChainHead(StoredBlock). Pomocí těchto metod lze přistupovat k nejaktuálnějšímu (nejpozději vytvořenému) bloku v blok chainu a zejména jeho hlavičce. (Jak jsem již zmiňoval na začátku kapitoly 6, BitCoinJ není plnou implementací Bitcoinového protokolu. Zde je vidět markantní rozdíl a to v tom, že ukládá pouze hlavičky bloků a zahazuje informace o transakcích, které jsou běžně částí bloků.)

Aktuálně implementují rozhraní BlockStore tři třídy. MemoryBlockStore uchovává hlavičky bloků v paměti, což je sice rychlé, ale ne úplně nejvhodnější s přihlédnutím k tomu jak blok chain roste. DiskBlockStore má podobou funkci jako MemoryBlockStore, ale už umí ukládat hlavičky bloků na datové úložiště. Poslední třída je BoundedOverheadBlockStore.

Jelikož nebudu stahovat celý blok chain MemoryBlockStore bude pro mě velmi dobrou volbou. Všechny předcházející implementace BlockStore jsou již zahrnuty v projektu Bitcoin a na velkém množství dalších se ještě pracuje. Některé jsou založené na SQLite a další část zase na NoSQL.

7.4.2. Třída Peer

Další třídou, kterou potřebuji pro úspěšné získání genesis bloku je třída Peer. Ta se stará o high-level komunikaci mezi vlastním klientem a určitým uzle v síti Bitcoinu. Při důkladnější analýze této třídy jsem zjistil, že vlastně hodně záleží na dalších třídách jako je například NetworkConnection, Block a BlockChain, aby tato třída Peer mohla plnit svůj účel.

Při standartním používání knihovny BitCoinJ se aplikace připojuje k neomezenému počtu uzlů. Záleží pouze na počtu Peer objektů. Po zavolání metody connect() na objektu Peer kód, který má na starosti zprávy z ostatních uzlů spustí smyčku a musí běžet jako vlákno. Tento případ budu demonstrovat v další ukázce kódu.

7.4.3. Požadavek na genesis blok prostřednictvím peeru sítě

Díky znalosti vnitřního fungování knihovny BitCoinJ můžu již přejít k samotné ukázce, jak takové získávání genesis bloku funguje v praxi. Na následujícím obrázku začínám vytvořením instance objektu, který reprezentuje parametry pro TestNet (řádka 19) a využívám ho k inicializaci objektu MemoryBlockStore (řádka 22). Poté deklaruji objekt BlockChain (řádka 25) a dále ho inicializuji uvnitř try-catch bloku (řádka 30). Objekt BlockChain vyžaduje oba objekty (NetworkParameters a BlockStore) jako argumenty.

Obrázek 40: Vyzvednutí genesis bloku

```
003|import java.io.IOException;
004|import java.net.InetAddress;
005|import java.net.UnknownHostException;
006|import java.util.concurrent.ExecutionException;
007|import java.util.concurrent.Future;
008|
009|import com.google.bitcoin.core.*;
010|import com.google.bitcoin.store.BlockStore;
011|import com.google.bitcoin.store.BlockStoreException;
012|import com.google.bitcoin.store.MemoryBlockStore;
013|
014|public class FetchGenesisBlock {
015|
016|    public static void main(String[] args) {
017|
018|        // Použij TestNet
019|        final NetworkParameters netParams = NetworkParameters.testNet();
020|
021|        // datová struktura pro uložení blok chainu
022|        BlockStore blockStore = new MemoryBlockStore(netParams);
023|
024|        // deklarace objektu blok chainu
025|        BlockChain chain;
026|
027|        try {
028|
029|            // inicializace objektu
030|            chain = new BlockChain(netParams, blockStore);
031|
```

Zdroj: Vlastní tvorba

Potom založím instanci objektu Peer (viz obr. 40) a definuji, že má využít TestNet, připojit se k uzlu na localhostu a využiji už vytvořenou instanci objektu BlockChain pro uchování blok chainu (řádek 33). Po té se připojuji k peer uzlu (řádek 36). Dále je vykonávána smyčka objektu Peer (řádek 39). Typičtější případ připojení je takový, že bych měl několik připojení k peer uzlům. V tomto případě by pak pravděpodobně každý objekt Peer, lépe řečeno smyčka zajišťující výměnu zpráv, běžela ve vlastním vlákně.

Protože mám pouze připojení pouze k jednomu uzlu, nemusím zde problematiku vláken uvažovat. Jelikož jsem si našel hash genesis bloku TestNetu pomocí webové stránky Bitcoin Block Explorer, založím Sha256Hash objekt (řádek 50). Dále pak objekt

Peer pošle požadavek na blok s tímto hashem připojenému uzlu sítě (řádek 54). Po přijetí vyžádaného bloku (řádek 58) ho vypíši. Metoda toString() je u této třídy dává velmi přesné údaje o bloku. (řádek 59).

Obrázek 41: Vyzvednutí genesis bloku - pokračování

```
031|
032|            // instance objektu Peer
033|            final Peer peer = new Peer(netParams, new PeerAddress(InetAddress.getLocalHost()), chain);
034|
035|
036|            peer.connect();
037|
038|
039|            new Thread(new Runnable() {
040|                public void run() {
041|                    try {
042|                        peer.run();
043|                    } catch (PeerException e) {
044|                        throw new RuntimeException(e);
045|                    }
046|                }
047|            }).start();
048|
049|
050|            Sha256Hash blockHash = new Sha256Hash("00000007199508e34a9ff81e6ec0c477a4cccff2a4767a8eee39c11db367b008")
051|
052|
053|
054|            Future<Block> future = peer.getBlock(blockHash);
055|            System.out.println("Waiting for node to send us the requested block: " + blockHash);
056|
057|
058|            Block block = future.get();
059|            System.out.println("Here is the genesis block:\n" + block);
```

Zdroj: Vlastní tvorba

Obrázek 42: Odchytávané výjimky

```
060|
061|                // hotovo odpojit
062|                peer.disconnect();
063|
064|            // některé odchycené výjimky
065|            } catch (BlockStoreException e) {
066|                e.printStackTrace();
067|            } catch (UnknownHostException e) {
068|                e.printStackTrace();
069|            } catch (PeerException e) {
070|                e.printStackTrace();
071|            } catch (IOException e) {
072|                e.printStackTrace();
073|            } catch (InterruptedException e) {
074|                e.printStackTrace();
075|            } catch (ExecutionException e) {
076|                e.printStackTrace();
077|            }
078|    }
079|}
```

Zdroj: Vlastní tvorba

V přechozím obrázku už je pouze příklad odchytávaných výjimek. Samozřejmě nejsou zde uvedeny všechny a pro bezpečné používání by bylo třeba celý kód otestovat např. napsáním JUnit testů.

7.5. Odeslání Bitcoinů

K poslední ukázce funkčnosti aplikace založené na BitCoinJ knihovně vytvořím třídu, která bude umožňovat zaslat Bitcoiny z již vytvořené peněženky na libovolnou Bitcoinovou adresu. Pro tento úkol budu využívat metodu třídy Wallet sendCoins(Peer peer, Address to, BigInteger nanocoins)

Na rozdíl od předešlých případů budu veškerý vstup zadávat z příkazové řádky. Tato aplikace bude vyžadovat následující argumenty v tomto pořadí:

- Jaká síť se bude využívat („test“ nebo „prod“)
- Jméno souboru peněženky, která obsahuje Bitcoiny
- Počet nano Bitcoinů, které chceme odeslat
- Bitcoinová adresa příjemce

Obrázek 43: Odeslání Bitcoinů

```
003|import java.io.File;
004|import java.io.IOException;
005|import java.math.BigInteger;
006|import java.net.InetAddress;
007|import java.net.UnknownHostException;
008|
009|import com.google.bitcoin.core.*;
010|import com.google.bitcoin.store.BlockStore;
011|import com.google.bitcoin.store.BlockStoreException;
012|import com.google.bitcoin.store.MemoryBlockStore;
013|
014|public class SendCoins {
015|
016|    public static void main(String[] args) {
017|
018|        if (args.length != 4) {
019|        System.out.println("Usage: java SendCoins prod|test wallet amount recipient");
020|        System.exit(1);
021|        }
022|
023|
024|
025|        String network          = args[0];
026|        String walletFileName   = args[1];
027|        String amountToSend     = args[2];
028|        String recipient        = args[3];
029|
030|         // jaká se použije síť
031|        final NetworkParameters netParams;
```

Zdroj: Vlastní tvorba

V přechozím obrázku nastavím argumenty příkazové řádky (řádek 18 až 29) a inicializuji některé komponenty, které jsem již v předchozích kapitolách zmiňoval. Hlavně se to týká instancí tříd NetworkParameters, BlockChain, Wallet a Peer. Částka k odeslání je specifikovaná jako BigInteger, kde každá jednotka reprezentuje 1/100000000 Bitcoinu (řádek 59). Dále vytvořím instanci objektu Address založeném na Bitcoinové adrese poskytnuté standartním klientem (řádek 71, obrázek 44). Po té využiji metodu třídy Wallet sendCoins() k započetí transakce (řádek 74, obrázek 44). Pokud se mi vrátí hodnota null, tak to značí, že nebyl dostatek prostředků v peněžence zvolené k transakci. V opačném případě je zasláno potvrzení o tom, jaká částka byla odeslána.

Obrázek 44: Odeslání Bitcoinů - pokračování

```
032|
033|         // pokud uživatel zadal do popisu "ostrou" síť
034|        if (network.equalsIgnoreCase("prod")) {
035|            netParams = NetworkParameters.prodNet();
036|          // jinak TestNet
037|        } else {
038|            netParams = NetworkParameters.testNet();
039|        }
040|
041|
042|        BlockStore blockStore = new MemoryBlockStore(netParams);
043|
044|
045|        BlockChain chain;
046|
047|
048|        Wallet wallet;
049|
050|        try {
051|
052|
053|            final File walletFile = new File(walletFileName);
054|
055|
056|            wallet = Wallet.loadFromFile(walletFile);
057|
058|
059|            BigInteger btcToSend = new BigInteger(amountToSend);
060|
```

Obrázek 45: Odeslání Bitcoinu - dokončení transakce a odchytávané výjimky

```
061|
062|            chain = new BlockChain(netParams, wallet, blockStore);
063|
064|
065|            final Peer peer = new Peer(netParams, new PeerAddress(InetAddress.getLocalHost()), chain);
066|
067|
068|            peer.connect();
069|
070|
071|            Address recipientAddress = new Address(netParams, recipient);
072|
073|
074|            Transaction sendTxn = wallet.sendCoins(peer, recipientAddress, btcToSend);
075|
076|
077|            if (sendTxn == null) {
078|                System.out.println("Cannot send requested amount of " + Utils.bitcoinValueToFriendlyString(btcToSend)
079|                                + " BTC; wallet only contains " + Utils.bitcoinValueToFriendlyString(wallet.getBalance()) + " BTC.");
080|            } else {
081|
081|
083|
084|                System.out.println(Utils.bitcoinValueToFriendlyString(btcToSend) + " BTC sent. You can monitor the transaction here:\n"
085|                                + "http://blockexplorer.com/tx/" + sendTxn.getHashAsString());
086|            }
087|
088|
089|            wallet.saveToFile(walletFile);
090|
091|
092|        } catch (BlockStoreException e) {
093|            e.printStackTrace();
094|        } catch (UnknownHostException e) {
095|            e.printStackTrace();
096|        } catch (PeerException e) {
097|            e.printStackTrace();
098|        } catch (AddressFormatException e) {
099|            e.printStackTrace();
100|        } catch (IOException e) {
101|            e.printStackTrace();
102|        }
103|    }
104|}
```

Zdroj: Vlastní tvorba

Výstup z této třídy při zadání následujících údajů příkazové řádky:

java com.FWallet.bitcoinj.SendCoins prod sendcoins-prodnet.wallet 2000000 1MVNTqKKpnSyx3tpztWX5p8oySTRRPmknd

Obrázek 46: Potvrzení o provedení platby

```
0.02 BTC sent.
```

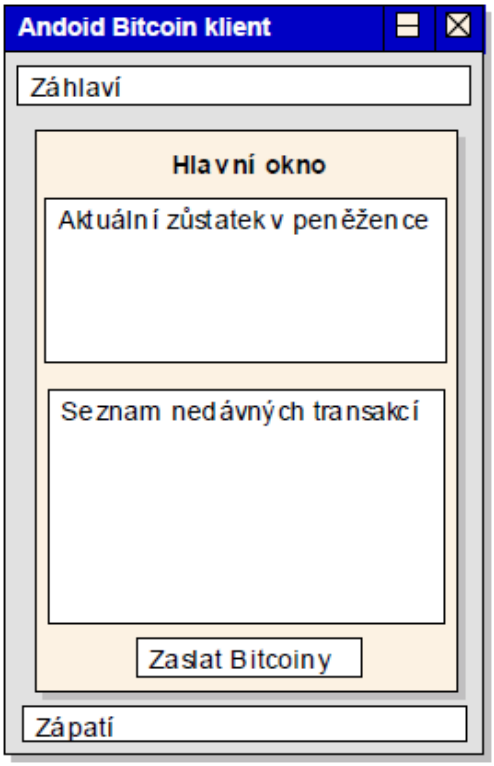
Zdroj: Vlastní tvorba

7.6. Návrh grafického prostředí

Pro návrh uživatelského grafického prostředí jsem zvolil nástroj již zmiňovaný nástroj Enterprise Architect od společnosti Sparx Systems a to konkrétně ve verzi 8. Tento nástroj podporuje notaci UML, která se pro analýzu při vývoji software běžně používá.

Jako první část uživatelského rozhraní jsem navrhnul uvítací obrazovku. Po instalaci aplikace a jejím spuštění by měl uživatel základní přehled. To určitě znamená celkový aktuální zůstatek v peněžence. Dále by zde měla být jasná navigace, tím myslím dobře viditelní odkazy na nejdůležitější operace, které bude chtít uživatel dělat. V tomto případě peněženky jde o možnost zaslat Bitcoiny (provést platbu neboli také vytvořit transakci). V neposlední řadě by zde mělo být také nějaké záhlaví, kde by bylo logo aplikace a její jméno. V tomto záhlaví by pak bylo i menu, které by se dalo rozkliknout (srolovalo by se dolů). Prostřednictvím tohoto panelu by uživatel mohl získat informace o verzi aplikaci a o autorovi, dále pak přístup do adresáře, celkové historie transakcí a případně by menu sloužilo k dalším doplňkovým funkčnostem, které by se časem mohli do této aplikace přidat (např. aktuální směnný kurz vůči USD či EUR)

Obrázek 47: UML návrh úvodní obrazovky



Zdroj: Vlastní tvorba

Z přechozího obrázku je také vidět, že jsem do spodní poloviny návrhu také zařadil prvek, kterém by mohl uživatel vidět poslední provedené transakce. V tomto základním zobrazení by byla vidět pouze adresa a počet Bitcoinů transakci. Pro rozlišení jestli se jedná o transakci příchozí či odchozí by bylo použito šipek s různým barevným rozlišením (to bude součástí konkrétního grafického návrhu). Poslední částí je zápatí, kde by byla umístěna informace o copyrightu a že používání této aplikace je na vlastní riziko každého uživatele.

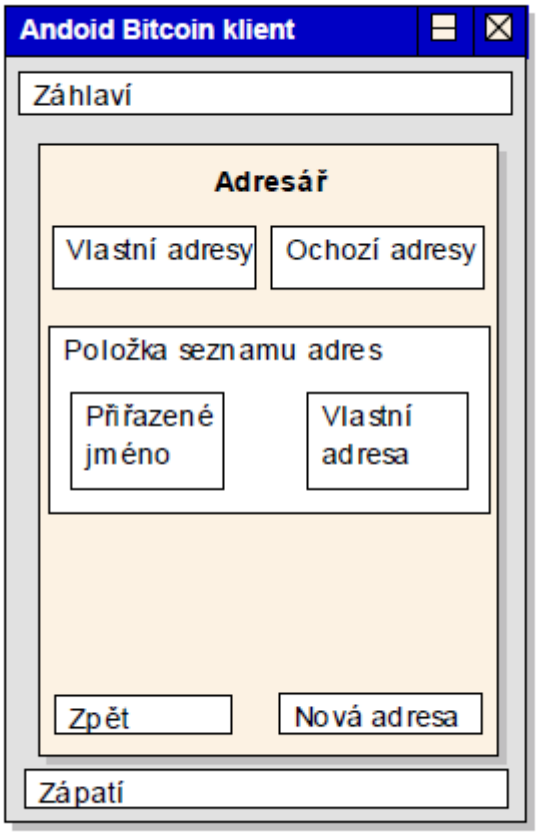
Obrázek 48: UML návrh zasílání Bitcoinů

Zdroj: Vlastní tvorba

Obrázek č. 48 zobrazuje návrh obrazovky, která by sloužila k provedení transakce, tzn. zaslání Bitcoinů na jinou adresu. Opět je zde záhlaví, které je stejné jako v případě přechozího návrhu dále pak hlavní část okna, ve které je jasně vidět jakou činnost jako uživatel provádím. Dále jsou zde dvě pole, do kterých může uživatel vepsat potřebné údaje. V první poli vyplňuje uživatel adresu příjemce. Po ťuknutí do tohoto pole je uživateli zobrazena softwarová klávesnice a lze do tohoto pole psát. V případě dlouhého

stisku lze již dříve zkopírovanou adresu vložit. Druhé pole má obdobné vlastnosti, ale zde uživatel vyplňuje částku, kterou chce zaslat. Důležité jsou přepínače, které se nacházejí bezprostředně pod tímto polem. Ty umožnují měnit jednotky z BTC (Bitcoinů) na mBTC (miliBitcoiny). Tyto přepínače jsou podle mne velmi důležité vzhledem, že aktuální hodnota jednoho Bitcoinu je přibližně je aktuálně asi 5800 Kč. Pokud tedy provádí uživatel menší transakce, může využít zadávání částky v miliBitcoinech (jedna tisícina BTC) a nemusí používat desetinného čísla. Po jsou zde již pouze dvě tlačítka, které umožňují buď potvrdit transakci, nebo jí zrušit. Při kliknutí na tlačítko odeslat se vyvolá ještě varovné okno vyžadující opětovné potvrzení transakce.

Obrázek 49: UML návrh adresáře

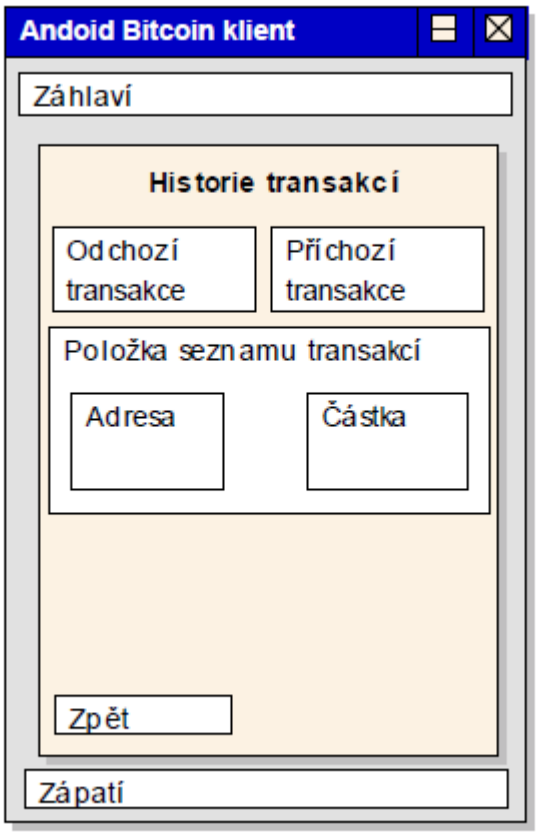


Zdroj: Vlastní tvorba

Přechozí obrázek znázorňuje, jak vypadá adresář neboli seznam adres. Nejprve je nutné výběrem správného tlačítka určit, jaký typ adres chce uživatel zobrazit. Jsou to buď adresy vlastní a adresy odchozí. Podle vybraného tlačítka pak lze zobrazit vybraný seznam položek. Tyto položky sami o sobě obsahují vlastní adresu a též přiřazené jméno

či jiné označení. Bez dodatečného označení adresy by pak adresář neměl žádný význam, protože by uživatel stejně nevěděl, komu tato adresa patří. Ve spodní části zobrazení je pak dvojice dalších tlačítek. Pomocí jednoho se lze vrátit na úvodní obrazovku a druhé má různou funkci podle toho, jaký seznam adres je vybrán. Pokud je vybrán seznam vlastních adres, pak toto tlačítko umožňuje vygenerovat novou adresu a té pak přiradit jméno či jinou identifikaci. V případě že je vybrán seznam odchozích adres, je tímto tlačítkem možné vytvořit ručně nový záznam v adresáři. Tentokrát musí uživatel vyplnit adresu a její označení.

Obrázek 50: UML návrh seznamu transakcí



Zdroj: Vlastní tvorba

Jako poslední UML návrh grafického prostředí jsem vytvořil historii transakcí. Každý uživatel chce mít určitě přehled o proběhlých transakcích a to jak o příchozích tak odchozích. V tomto případě stačí, aby uživatel vybral na úvodní obrazovce položku menu, kde pak nalezne položku historie transakcí. Po té musí zvolit, jestli chce zobrazit seznam odchozích nebo příchozích transakcí. Pokud zvolí odchozí transakce, zobrazí se

mu seznam položek. Každá položka obsahuje cílovou adresu, případně její identifikaci, a také částku, která byla v této transakci odeslána. Pro přehlednost a možnost kontroly v jakém stavu se transakce nachází, je zde i indikátor stavu. Analogicky pokud uživatel vybere tlačítkem příchozí transakce, je mu zobrazen seznam těchto transakcí, kde opět vidí z jaké adresy a jaká částka přišla. Tlačítkem zpět se uživatel dostane opět na hlavní obrazovku aplikace.

7.6.1. Příklad grafického zpracování UML návrhu

Pro vytvoření konkrétního grafického zpracování jsem použil volně dostupného softwaru GIMP. Na následujících obrázcích je příklad, jak by výsledné uživatelské prostředí mohlo vypadat přímo na přenosném zařízení.

Obrázek 51: Grafický návrh - okno přehledu

Zdroj: Vlastní tvorba

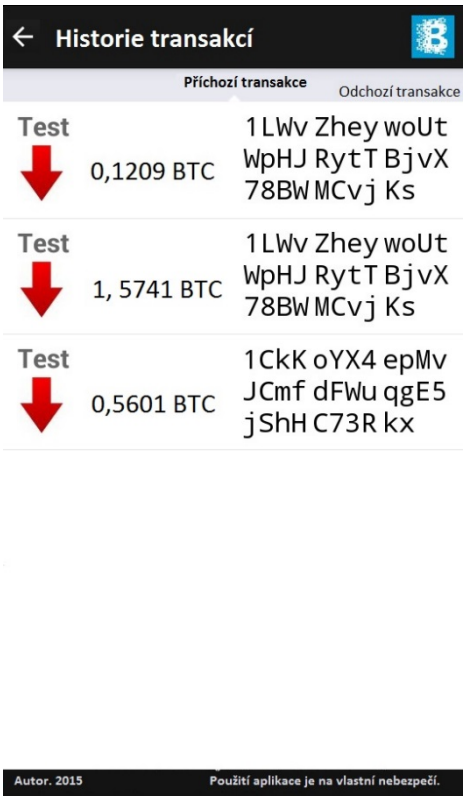
Obrázek 52: Grafický návrh - Odesílání Bitcoinů

Zdroj: Vlastní tvorba

Obrázek 53: Grafický návrh - adresář

Zdroj: Vlastní tvorba

Obrázek 54: Grafický návrh - historie transakcí



Zdroj: Vlastní tvorba

8. ZÁVĚR

Závěrem bych shrnul obsah celé mé diplomové práce a přidal doporučení, jakým způsobem by mohla být práce ještě rozšířena. Prvním cílem mé diplomové práce bylo podrobně rozpracovat téma virtuálních měn a zejména Bitcoinu. Po nastudování tématu jsem se rozhodl v první řadě popsat hlavní vlastnosti této měny. Základní vlastností je, že tato měna je vytvořena jako tzv. open-source projekt, veškeré zdrojové kódy tohoto projektu jsou veřejně k dispozici a tato měna se neustále vyvíjí. Další vlastností je fakt, že se jedná o decentralizovanou měnu, která nemá žádnou centrální autoritu, která by měnu vydávala či nějakým způsobem regulovala. Tento fakt byl v prvních fázích mé analýzy hlavním bodem, který mně zaujal, protože jsem si nedokázal představit, že by takto postavený systém mohl fungovat. Po detailnějším rozboru a studiem principu P2P sítí jsem ovšem velmi rychle přišel na velké výhody, které z decentralizace měny plynou. Například při srovnání převodu Bitcoinů se standartním elektronickým bankovním převodem peněz není pro uživatele žádný velký rozdíl. Uživatel vytvoří příkaz, vyplní částku a adresu kam chce tyto prostředky poslat a příkaz odešle k vyřízení. O tohoto bodu se pak projevují výhody Bitcoinu jako platebního systému. Nejprve je dobré zmínit, že standartní příkazy k úhradě jsou bankou zpracovány v horizontu jednoho až dvou dní. U Bitcoinu je to řádově v minutách tzn., že celá transakce je dokončena v minutách. Nezáleží ani na datu či hodině, kdy chceme převod provést, což může být u bankovního převodu velký problém. Další výhodou jsou poplatky na převod, které jsou takřka nulové a také anonymita uživatele. V tomto bodě je možná diskuze o tom, jestli je to vlastně výhoda či nevýhoda např. kvůli praní špinavých peněz atd. Jedinou čistou nevýhodou, která plyne z decentralizace této měny, vidím v nemožnosti vrátit transakci. To znamená, že pokud uživatel splete adresu či vyplní špatnou částku a transakci odešle, nemá pak už žádnou možnost jak tuto transakci zrušit. V případě bankovního převodu by se obrátil s požadavkem na bankovní instituci, ale v rámci tohoto platebního systému se není kam obrátit. Tyto informace jsou asi tím nejdůležitějším, co popisuje první kapitola mé práce.

Protože, jde o měnu založenou na kryptografických principech je následně zařazena kapitola shrnující kryptografické techniky, které je nutné pro znát pro následnou analýzu této měny. Zde se jedná hlavně o popis šifrovacích algoritmů a zejména pak o ECDSA (Elliptic Curve Digital Signature Algorithm) a následně hashovacích funkcí jako je např. SHA 256. Tato kapitola by mohla být mnohem rozsáhlejší a určitě by mohla být

samostatným tématem diplomové práce, proto jsou zde s ohledem na vlastní téma popsány jen základy.

V další kapitole se pak již zabývám vnitřním fungováním tohoto platebního systému, například problémem generování adres v Bitcoinové síti. Při srovnání opět s tradičním bankovním převodem má Bitcoin tu výhodu, že jako uživatel si mohu vytvořit neomezený počet adres (tedy účtů). Na rozdíl od banky kde mi banka přidělí jedno číslo účtu. Při tomto principu bylo nutné analyzovat, jestli není možné, aby byla adresa v rámci duplicitní, protože adresy je možné generovat i off-line bez připojení do sítě a ani po připojení není autorita, která by kontrolovala duplicitu. S využitím znalostí z předešlé kapitoly jsem zjistil, že taková situace nemožná a to hlavně kvůli samotnému procesu generování adres, který využívá mnohonásobného hashování klíčů. Dále se zabývám vytvářením transakcí a jejich následném zatřiďování do bloků. Zde je základní informací fakt, že každá transakce si udržuje referenci na transakci předchozí a celá struktura se pak chová podobně jako spojový seznam. Po určitém časovém intervalu se všechny tyto transakce spojí v jeden blok a ten se zatřídí do tzv. blok chainu. Zatřídí se nakonec a opět každý blok si v hlavičce drží referenci na blok předchozí. Blok chain je pak podle tohoto principu nazýván jako tzv. otevřené účetnictví celého Bitcoinu. Jsou v něm obsaženy všechny transakce od spuštění a je veřejně dostupný pro každého. Ukázka jak taková transakce v realitě probíhá je pak zachycena v příkladu, který zakončuje teoretickou část mé diplomové části.

Cílem praktické části bylo analyzovat na navrhnout platební aplikaci pro virtuální měnu Bitcoin, která by umožnovala platbu na přenosných zařízeních s operačním systémem Android. Prvním úkolem byla analýza vhodného programovacího jazyka a výběr vývojového prostředí. Výsledkem této analýzy byl výběr programovacího jazyka Java a vývojového prostředí Eclipse. Důvodem pro výběr této kombinace byla dobrá kompatibilita vývojového prostředí s ADT (Android Development Tools). Tento balíček knihoven a dalších utilit je původně určen pro vývojové prostředí Android Studio, ale kvůli jeho špatné přehlednosti jsem zvolil raději Eclipse. Toto prostředí je dlouhodobě nejpoužívanější pro vývoj Java aplikací a je také velmi přehledné. V neposlední řadě má také výhodu, že se nemusí složitě konfigurovat a instalovat. Další části přípravy implementace samotné aplikace byla analýza open-source knihovny BitCoinJ, která v sobě obsahuje již připravené části, které jsem pak při implementaci používal. Samotná implementace platební aplikace pak obsahuje několik základních funkčností. První z nich

je vytváření – generování adresy pro Bitcoinovou síť. Tuto funkčnost zajišťuje třída CreateAddress, která využívá již zmíněné knihovny BitCoinJ. Pro vytvoření adresy je napřed nutné vygenerovat pár klíčů, což je zajištěno objektem typu ECKey. Pomocí metody toAddress() pak získávám platnou adresu. Další základní funkčností je vytvoření peněženky, aby měl uživatel kde své vygenerované adresy ukládat. V tomto případě se jedná o třídu CreateWallet. V zásadě jde třídu, která vytvoří soubor, který má datovou strukturu peněženky a lze do ní ukládat neomezené množství adres. Při implementaci jsem ovšem zjistil, že v rámci implementace BitCoinJ úplně chybí metoda pro mazání jednotlivých adres z peněženky. Proto jsem tuto funkčnost doplnil. Ovšem pro uživatelovu vlastní bezpečnost, nebude tato funkčnost využita a to z toho důvodu, že po smazání adresy už nelze adresu nijak obnovit a veškeré prostředky na této adrese jsou nenávratně ztraceny, což může mít dost fatální následky. Třetí podstatná třída řeší připojení k Bitcoinové síti a také komunikaci s ostatními uzly (peery). Aplikace musí být schopna se k síti připojit a synchronizovat se tzn. stáhnout celý blok chain a přesně proto jsem navrhl třídu FetchGenesisBlock. Při synchronizaci je totiž nutné začít od prvního bloku blok chainu tzv. Genesis bloku. Poslední implementovanou funkčností je provedení transakce, to znamená odeslání Bitcoinů na jinou adresu v síti. Zajišťuje ji třída SendCoins. V této třídě se využije objektu Wallet pro načtení obsahu peněženky a následně se využije objektu Peer pro komunikaci s ostatními uzly v síti. Samostatnou transakci pak provádím pomocí objektu Transaction, kterému předávám parametry zadané uživatelem.

Poslední kapitolou je pak návrh grafického uživatelského prostředí, kde jsem použil nejprve UML a to konkrétně User Interface Diagramu a následně pak grafického softwaru pro finální grafický návrh aplikace. Vytvořil jsem několik základních obrazovek např. úvodní obrazovka peněženky, dále pak formulář pro vyplnění transakce, seznam provedených transakcí nebo adresář.

Celkově tedy byla navržena a implementována hlavní funkcionalita peněženky a připraveno vhodné grafické prostředí aplikace. Práci by bylo možné rozšířit o implementaci grafického prostředí a tím by vznikla plně funkční aplikace.

9. SUMMARY

The thesis is focused on the topic of virtual cryptocurrency Bitcoin. The theoretical part is divided into several chapters. First of all, there is a chapter containing basic information about the currency's history, its development to the present, as well as principles of operation of this currency.

Because it is a currency based on cryptographic principles next chapter summarizing cryptographic techniques, which is necessary to know for subsequent analysis of this currency. In other parts is elaborated a detailed analysis of the principles of Bitcoin. The main point of this analysis is a system of transactions, decentralized history, then joining the transactions into blocks and in the end creating blockchain. It also explains the principle of P2P networks.

The practical part of this work focuses on the analysis and design of payment applications for mobile devices. There used open-source libraries such BitcoinJ. The application is developed in Java programming language with plugin for portable devices, which work under the operating system Android. For the purposes of analysis and design of graphical user interface are used UML diagrams. The specification of the UML diagrams was utilized by freely available graphics software and the result is also included in this work.

Keywords: bitcoin, P2P network, cryptocurrency, money, block, blockchain, transaction, address, protocol, wallet, client, miner, pool, target, peer, network node, cryptography, encryption, decryption, key, hashing algorithm, digital signature, BitcoinJ.

10. SEZNAM POUŽITÝCH ZDROJŮ

Afeez, O. (16. Leden 2015). *ENCRYPTION AND DECRYPTION ALGORITHM*. Načteno z University of East London: http://homepages.uel.ac.uk/u0430614/Encryption%20index.htm

Andresen, G. (2015). *Testnet Faucet*. Načteno z http://testnet.freebitcoins.appspot.com/

Anon. (16. Leden 2015). *Bitcoin*. Načteno z Bitcoin Wiki: https://en.bitcoin.it/wiki/Main_Page

Anon. (2015). *Bitcoin Forum - Index,* . Načteno z Bitcoin: https://bitcointalk.org/

Anon. (19. Leden 2015). *Peníze*. Načteno z Wikipedia: http://cs.wikipedia.org/wiki/Pen%C3%ADze

Anon. (11. Leden 2015). *Testnet*. Načteno z Bitcoin Wiki: https://en.bitcoin.it/wiki/Testnet

Bitcoin - P2P digital currency. (16. Leden 2015). Načteno z Bitcoin Project: http://bitcoin.org

Bitcoin Beta - Stack Exchange . (18. Leden 2015). Načteno z Stack Exchange inc.: http://bitcoin.stackexchange.com/

Developers, B. (2013). *Bitcoin GitHub*. Načteno z GitHub: https://github.com/bitcoin

Enterprise Architect. (2015). Načteno z Dataprojekt s.r.o.: http://www.enterprise-architect.cz/

Javůrek, K. (15. Srpen 2013). *Těžba kryptoměn*. Načteno z Živě.cz: Těžba kryptoměn je novodobá zlatá horečka Více na: http://www.zive.cz/clanky/tezba-kryptomen-je-novodoba-zlata-horecka/sc-3-a-170137/default.aspx#utm_medium=selfpromo&utm_source=zive&utm_campaign=copylin#utm_medium=selfpromo&utm_source=zive&utm_campaign=c

Johnson, D. (2006). *The Elliptic Curve Digital Signature Algorithm (ECDSA).* Načteno z University of Waterloo: http://cs.ucsb.edu/~koc/ccs130h/notes/ecdsa-cert.pdf

Klíma, V. (19. Březen 2005). *Hašovací funkce, principy, příklady a kolize.* Načteno z CryptoFest: http://cryptofest.cz/slides/klima_cryptofest_2005.pdf

Klíma, V., & Rosa, P. (2004). *Kryptologie pro praxi – DSA, ECDSA.* Načteno z CryptoWorld: http://crypto-world.info/klima/2004/st_2004_04_21_21.pdf

Mining hardware comparison. (20. Prosinec 2014). Načteno z Bitcoin Wiki: https://en.bitcoin.it/wiki/Mining_hardware_comparison

Moore, C. (2013). *When does the main client relay its knowledge of TXs in INV message?* Načteno z Stackexchange.com: http://bitcoin.stackexchange.com/a/3277/323

Novotný, R. (13. Prosinec 2013). *Evropský bankovní úřad varuje před riziky bitcoin*. Načteno z Investujeme.cz: http://www.investujeme.cz/evropsky-bankovni-urad-varuje-pred-riziky-bitcoinu/

Novotný, R. (6. Leden 2014). *Bitcoin: Sen rentiéra, nebo bublina?* Načteno z Investujeme.cz: http://www.investujeme.cz/bitcoin-sen-rentiera-nebo-bublina/

Online. (30. Duben 2011). *Základy šifrování: symetrická a asymetrická kryptografie*. Načteno z Floops.cz: http://www.flops.cz/zaklady-sifrovani-symetricka-a-asymetricka-kryptografie

Online. (28. Květen 2012). *Money*. Načteno z Wikipedia: http://en.wikipedia.org/wiki/Money

Online. (10. Prosinec 2014). *What Are The 6 Characteristics Of Money?* Načteno z BlurtIt: http://money.blurtit.com/q5102689.html

Online. (14. Leden 2015). *Fractional-reserve banking*. Načteno z Wikipedia: http://en.wikipedia.org/wiki/Fractional-reserve_banking

Online. (15. Leden 2015). *Market Price*. Načteno z Blockchain.info: https://blockchain.info/charts/market-price

Perry, D. (12. Září 2013). *Bitcoin alternative designed for NVIDIA", http://bitcoin.stackexchange.com/a/1525/323*. Načteno z Stackexchange.com: http://bitcoin.stackexchange.com/a/1525/323

Perry, D. (2014). *Bitcoin Wiki*. Načteno z Double spending: https://en.bitcoin.it/wiki/Double-spending

Polesný, R. (10. Únor 2014). *Bitcoin se otřásá*. Načteno z Živě.cz: http://www.zive.cz/bleskovky/bitcoin-se-otrasa-nejvetsi-bitcoinova-burza-

mtgox-ma-problemy/sc-4-a-172408/default.aspx#utm_medium=selfpromo&utm_source=zive&utm_campaign=copylink

Protocol specification. (8. Leden 2015). Načteno z Bitcoin Wiki: https://en.bitcoin.it/wiki/Protocol_specification

Smith, P. (15. Leden 2015). *Blocks size*. Načteno z Blockchain.info: https://blockchain.info/charts/blocks-size

Syrovátková, V. (15. Listopad 2011). *Symetrické šifrování*. Načteno z Západočeská univerzita v Plzni: http://home.zcu.cz/~vsyr/

Tilborg, H. C. (2005). *ENCYCLOPEDIA OF CRYPTOGRAPHY AND SECURITY*. New York: Springer Science+Business Media, Inc.

Vondruška, P. (Květen 2000). Cesta kryptografie do nového tisíciletí. *ComputerWorld*, stránky 37 - 40.

What is Loom? (16. Leden 2015). Načteno z Loom: https://loom.cc/help#what_is_loom

11. SEZNAM OBRÁZKŮ

12. SEZNAM PŘÍLOH

Příloha č. 1: Zdrojové kódy vytvořených tříd

13. PŘÍLOHY

Přílohy jsou v elektronické podobě na CD.