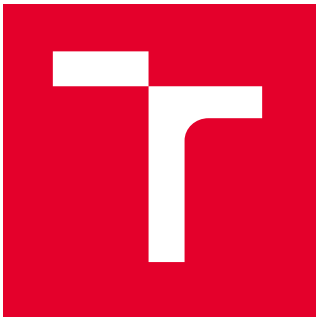


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

POKROČILÉ ZABEZPEČENÍ BLOCKCHAINOVÝCH TRANSAKCÍ

ADVANCED SECURITY FOR BLOCKCHAIN TRANSACTIONS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Minh Tran**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Dzurenda, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Minh Tran ***ID:*** 196013

Ročník: 2 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Pokročilé zabezpečení blockchainových transakcí

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte problematiku blockchainových transakcí a způsobů jejich zabezpečení. Seznamte se s technologiemi multi-signatures, threshold-signatures, Schnorr signature a možnostmi jejich implementace do existujících systémů blockchain. Navrhněte a implementujte systém založený na technologii blockchain a podporující více podpisové transakce a více klíčové peněženky s podporou bezpečného výpočtu více stran (ang. Secure multiparty computations). Implementovaný systém bude využívat chytrá zařízení jako jsou chytré telefony či chytré hodinky.

DOPORUČENÁ LITERATURA:

[1] RICCI, S.; DZURENDA, P.; CASANOVA-MARQUÉS, R.; ČÍKA, P. Threshold Signature for Privacy-preserving Blockchain. In Business Process Management: Blockchain, Robotic Process Automation, and Central and Eastern Europe Forum. Münster, Germany: Springer, 2022. p. 1-15. ISBN: 978-3-031-16167-4.

[2] Android Developers [online]. Google [cit. 2022-09-01]. Dostupné z: https://developer.android.com/

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 19.5.2023

Vedoucí práce: Ing. Petr Dzurenda, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Technologie blockchain, zejména Bitcoin, změnila způsob, jakým přemýšlíme o finančních transakcích a jak je řídíme. S rostoucí poptávkou a používáním technologie blockchain se však bezpečnost kryptoměnových peněženek stala závažnou obavou. Prahové podpisy nabízejí slibné řešení tohoto problému, umožňují více stranám podepsat transakci bez odhalení jejich soukromých klíčů. Tento článek představuje aplikaci pro mobilní Bitcoinovou peněženku pro Android, která používá prahové podpisy založené na Schnorrově podpisu. Aplikace také integruje chytré hodinky pro lepší zabezpečení a použitelnost. Tato integrace poskytuje další vrstvu zabezpečení tím, že vyžaduje fyzické potvrzení od uživatele před schválením jakékoli transakce. Naše implementace poskytuje bezpečnou a efektivní platformu pro správu Bitcoinových aktiv pomocí prahových podpisů a zároveň poskytuje intuitivní a snadno použitelné rozhraní pro interakci s aplikací.

KLÍČOVÁ SLOVA

Android, bezpečnost transakcí, Bitcoin, blockchain, chytré hodinky, prahový podpis, sdílení tajemství

ABSTRACT

Blockchain technology, especially Bitcoin, has revolutionized how we think about and manage financial transactions. However, with the increasing demand and usage of blockchain technology, the security of cryptocurrency wallets has become a critical concern. Threshold signatures offer a promising solution to this problem, allowing multiple parties to sign a transaction without revealing their private keys. This article presents an Android mobile Bitcoin wallet application that uses Schnorr-based threshold signatures. The application also deploys smartwatch integration for enhanced security and usability. This integration provides an additional layer of security by requiring physical confirmation from the user before approving any transaction. Our implementation provides a secure and efficient platform for managing Bitcoin assets using threshold signatures while also providing an intuitive and easy-to-use interface for interacting with the application.

KEYWORDS

Android, security of transactions, Bitcoin, blockchain, smartwatch, threshold signature, secret sharing

Vysázeno pomocí balíčku `thesis` verze 4.07; `http://latex.feec.vutbr.cz`

TRAN, Minh. *Pokročilé zabezpečení blockchainových transakcí*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023, 72 s. Diplomová práce. Vedoucí práce: Ing. Petr Dzurenda, PhD.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Minh Tran
VUT ID autora:	196013
Typ práce:	Diplomová práce
Akademický rok:	2022/23
Téma závěrečné práce:	Pokročilé zabezpečení blockchainových transakcí

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Petru Dzurendovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam výpisů

Úvod

Technologie blockchain je v posledních letech velmi často skloňovaným termínem a každým dnem je čím dál tím více populární. Ať už je popularita dána čímkoliv, faktem je, že tato technologie přináší revoluční řešení mnoha problémů dnešní doby. Nejčastějším příkladem použití blockchainu jsou finance. V říjnu 2022 byla v Evropě průměrná inflace 11,5 % a v některých zemích jako je Estonsko nebo Maďarsko vyšplhala inflace až na 22,5 % [1]. Tak vysoká čísla jsou výsledkem nejen špatné makroekonomie ale také nekontrolovatelného tisknutí peněz. Tento problém může velmi jednoduše vyřešit finanční systém založený na blockchainu. Jelikož jeho vlastnosti jako decentralizace, transparentnost a programovatelnost, které jsou všechny matematicky dokazatelné, dokáží zabránit téměř všem existujícím problémům dnešní finanční sféry a přináší mnohé výhody, kterými dnešní finanční systémy zatím nedisponují.

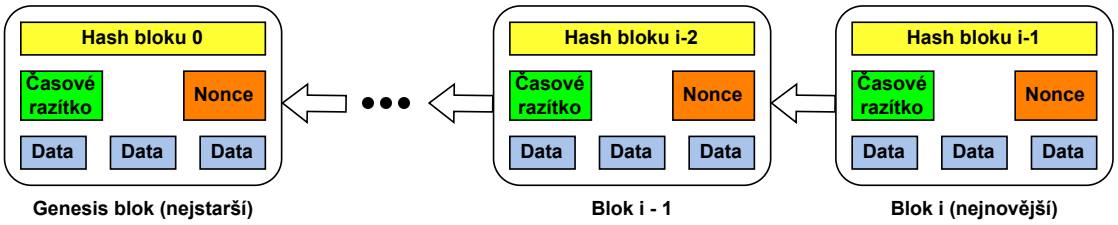
Finance nejsou ale jediným případem užití blockchainu. Příkladem může býti projekt od firmy IBM s názvem *Supply Chain Intelligence Suite: Food Trust* [2]. Projekt je zaměřený na trasování potravin od jejich vzniku až po prodej koncovému zákazníkovi. Cílem je zajistit co největší množství informací o produktu a zaručit důvěru v tyto informace pro všechny účastníky.

Má-li tato technologie být v budoucnu široce používána masovou společností, měla by být škálovatelná, rychlá, cenově dostupná, uživatelsky přívětivá, ekologická a hlavně bezpečná. Oproti klasickému standardu bank zde má klient plnou kontrolu nad svými aktivy, ale také plnou zodpovědnost. Z tohoto důvodu je bezpečnost kritická a s její nedokonalostí přichází také velké znepokojení. Jedním z takových znepokojivých faktů je využívání jednoho páru privátního a veřejného klíče, které jsou většinou uchovávány na jednom zařízení. Tato praktika může způsobit náchylnost na ztrátu a odcizení privátních klíčů, tedy samotných aktiv uživatele. Podle článku od CNN Business News [3] bylo jen za poslední rok ukradeno více než 3,8 miliardy dolarů v kryptoměnách.

Pro řešení tohoto problému mnohé blockchainy, jako například Bitcoin nebo Cardano, již nativně podporují více podpisové peněženky pro zvýšení bezpečnosti. Tedy je nutné mít podpisy od určitého počtu spolumajitelů peněženky, aby bylo možné aktiva v peněžence utrácet. Implementace tohoto přístupu má ale dva zásadní problémy a těmi jsou soukromí a škálovatelnost. Všichni spolumajitelé i podepisující jsou veřejně známí a velikost transakce se zvyšuje dle počtu podepisujících, čímž vznikají vyšší poplatky při užívání blockchainu. Z tohoto důvodu se tato práce věnuje pokročilému zabezpečení blockchainových transakcí. Cílem práce je navrhnout a vytvořit systém založený na technologii blockchain a podporující více podpisové transakce a více klíčové peněženky, které řeší zmíněné problémy.

1 Blockchain

Technologie blockchain ve své podstatě představuje záznam všech dat, která byla vytvořena a přidána do konkrétního blockchainu od počátku jeho vzniku. Data jsou shromažďována do bloků dat, které jsou mezi sebou vzájemně chronologicky spojeny tak, že každý blok obsahuje hash předchozího bloku. Obvykle jsou dále v bloku data časového razítka a náhodného čísla značeného jako *nonce*, které zajišťuje zvýšení obtížnosti podvržení hashe bloku [4]. Díky základní vlastnosti hashů, tj. jedinečnosti, se v blockchainu velmi dobře mitiguje útok podvržení bloků, kde by si útočník mohl pozměnit nebo vytvořit svoje vlastní data ke svému prospěchu. Data jsou tedy neměnná po celou dobu života blockchainu a z toho přirozeně vyplývá, že při užívání této technologie se data do blockchainu mohou jedině přidávat. První blok v blockchainu, který nemá rodičovský blok, se nazývá genesis blok a spojením dalších bloků k němu vzniká řetězec dat a tím i samotný blockchain. Na obrázku 1.1 je vyobrazen obecný diagram blockchainu.



Obr. 1.1: Obecný diagram blockchainu.

Aby byl nový blok zařazen za poslední blok v řetězci, je nutné dosáhnout konsenzu nebo-li shody mezi účastníky. Ten nastane, pokud je správnost všech dat v bloku i samotného bloku ověřena. Správnost definují kryptografické protokoly, které se včetně způsobu ověřování definují na začátku vzniku blockchainu. Není ale dané, že by způsob ověřování nebo kryptografické protokoly měly zůstat navždy stejné. Dojde-li ke konsenzu v oblasti technologické aktualizace, blockchain se bude od té chvíle řídit dle nových pravidel a protokolů. Těmto aktualizacím se dle kompatibility s předchozími verzemi říká buď *soft fork* a nebo *hard fork*. *Soft fork* přináší další funkce k již existujícím a nemění stará pravidla a je zpětně kompatibilní. *Hard fork* mění stará pravidla, a proto již není zpětně kompatibilní. Následkem *hard forku* je rozdělení blockchainu na starou a novou verzi - nezávisle na sobě běžící.

Dle skutečných případů použití technologie blockchain ji lze rozdělit do několika kategorií dle otevřenosti a dle použití tokenů [5]. Dle otevřenosti se blockchainy dělí na veřejné a privátní. U privátních blockchainů je přístup omezen jen pro skupiny

uživatelů, kteří jsou mezi sebou domluveni, že budou jejich členy. Naopak ve veřejných blockchainech neexistuje žádná centrální autorita a nikdo blockchain nevlastní. Blockchain je otevřený pro všechny, kdo chtějí být jeho součástí. V tomto typu se udržuje synchronizace stavu dat blockchainu na všech uzlech. Pro rozhodnutí o aktuálním stavu blockchainu se používá distribuovaný konsenzus. Používá-li blockchain ke svému fungování tokeny, je tzv. tokenizovaný. Příkladem jsou kryptoměny. Tokeny jsou pak používány jako měna nebo platba za používání služeb sítě a nebo jako odměna za čestnou účast v chodu sítě. V blockchainech bez tokenů je hlavní případ užití ukládání dat nebo přenos a sdílení dat mezi uživateli. Nejvíce používané blockchainy jsou kombinace typu veřejného a tokenizovaného blockchainu. Neboť se tímto způsobem nejlépe využijí výhody decentralizace.

1.1 Decentralizace dat a práv

Jeden z hlavních cílů vzniku technologie blockchain je decentralizace dat a práv. Data tedy nejsou uložená jen na jednom zařízení nýbrž na více zařízeních. Hostujícímu zařízení se většinou říká uzel (anglicky *node*). Uzly mezi sebou komunikují, a to za účely synchronizace, verifikace a nebo přidávání dat v blockchainu. Uzly mohou blockchain používat jako uživatelé a nebo jako ověřovatelé. Rozdíl je v tom, že provozovatel verifikuje a přidává bloky, uživatel jen udržuje synchronizovaný stav blockchainu a zadává požadavky pro úkony v síti. Čím více zařízení hostuje (udržuje synchronizovaný stav) konkrétní blockchain, tím více je blockchain decentralizovaný. Obecně je žádaná co nejvyšší možná decentralizace. Z této základní vlastnosti decentralizace vyplývají i výhody a nevýhody decentralizovaného blockchainu. K výhodám patří například [6]:

1. **Peer to peer** - Jelikož zde není žádná centrální autorita a moc nad sítí mají společně uzly sítě, tak si uživatelé mohou mezi sebou posílat data bez jakéhokoliv narušení od třetí strany. Tudíž je pro uživatele možné v rámci pravidel konkrétního blockchainu provádět prakticky vše, co je jim dovoleno.
2. **Nezničitelnost** - Je zde vysoká odolnost proti selhání funkčnosti nebo ztrátě dat. Ke ztrátě dat nebo vypnutí sítě by došlo jen v případě, kdy by byly kompromitovány úplně všechny uzly.
3. **Důvěra** - Díky tomu, že jsou uzly blockchainu na sobě nezávislé, data jsou neměnná, transparentní a každý uzel má schopnost verifikovat jejich pravost, vzniká naprostá důvěra založená na matematických důkazech.
4. **Otevřený vývoj** - Kdokoliv má právo navrhnout softwarové změny nebo vylepšení blockchainu. Změny se projeví po dosažení aktualizace na novou verzi u většiny uzlů.

Naopak k nevýhodám decentralizovaného blockchainu například patří[6]:

1. **Velikost dat** - Každým přidaným blokem se velikost blockchainu zvětšuje. Teoreticky se dá říci, že se velikost dat s časem blíží k nekonečnu. U blockchainů se špatnou architekturou může dojít k moc rychlému růstu velikosti dat oproti technickému růstu objemu datových disků. Hostující zařízení by v budoucnu mohla mít problém vůbec zvládat ukládání blockchainu.
2. **Domluva mezi hosty** - Tím, že má každý uzel stejná práva, je složité vyřešit situaci, kdy se názory liší. Je-li navržena nějaká důležitá a prospěšná změna, není jisté, že by vůbec došlo ke konsenzu a jestli ano, tak není jisté za jakou dobu.
3. **Cena** - Většina blockchainů je open-source a lze je lehce převzít. Pokud je ale nutno vytvořit například na míru privátní ale decentralizovaný blockchain, pak je cena pro vývoj a provoz poměrně vysoká.
4. **Zákony** - Je velmi obtížné vytvořit zákony ohledně kyberprostoru z mnoha důvodů, kterými jsou například extrateritorialita, anonymní identity, vymáhání práv a povinností, mezinárodní vztahy a mnohé další.
5. **Lidská chyba** - Když jsou data zapsána do blockchainu, už je nelze lehce změnit. To znamená, že téměř není možné uživatelské chyby zpětně napravit. Z tohoto důvodu je nutné, aby byly aplikace nasazené na blockchain dobře testovány a uživateli zadávaná data kontrolována.

1.2 Distribuovaný konsenzus

V decentralizovaném systému, kde není centrální autorita rozhodující o stavu systému, je nutné mít nějaký mechanizmus, který by poskytoval bezpečný způsob domluvy o jednom pravém stavu systému mezi většinou nebo všemi uzly systému, a tím je distribuovaný konsenzus. V blockchainu je konsenzus definován jako proces ujednání mezi navzájem si nevěřícími uzly na finálním stavu blockchainu nebo hodnotě. Pro dosažení konsenzu se používají algoritmy konsenzu. Z důvodu bezpečnosti jsou v těchto algoritmech zpravidla mechanizmy pro motivaci uzlů, aby se chovaly čestně. Každý algoritmus konsenzu musí splňovat následující podmínky [5]:

1. **Dohoda (anglicky *Agreement*)** - Všechny poctivé uzly na konci algoritmu konsenzu dochází ke stejnému výsledku.
2. **Zánik (anglicky *Termination*)** - Všechny poctivé uzly jsou schopné ukončit proces konsenzu a nakonec dojít k rozhodnutí.
3. **Platnost (anglicky *Validity*)** - Výsledek, na kterém se shodli všechny poctivé uzly, musí být stejný jako výchozí navržený výsledek od alespoň jednoho poctivého uzlu.
4. **Odolnost vůči chybám (anglicky *Fault tolerant*)** - Algoritmus konsenzu by měl běžet i v případě, že jsou v síti přítomny chybné nebo nepoctivé uzly.

5. **Intergrita (anglicky *Integrity*)** - Žádný uzel nesmí učinit rozhodnutí více než jednou v jednom cyklu konsenzu.

V blockchainu se rozdělují algoritmy konsenzu na dva hlavní typy[5]:

1. **Konsenzus založený na výběru vůdce nebo-li Nakamotův konsenzus** - Nejdůležitější částí Nakamotova konsenzu je role tzv. vůdce, jehož cílem je navrhnout výsledné rozhodnutí. O roli vůdce uzly soutěží pomocí algoritmu, který je definován v blockchainu, a nebo je vůdce pseudonáhodně vybrán.
2. **Konsenzus založený na BFT (*Byzantine Fault Tolerance*)** - Základem této metody jsou zprávy, které uzly digitálně podepisují. Konsenzus vznikne, když je přijat daný počet zpráv.

Nakamotův konsenzus je oproti konsenzu založeném na BFT mnohem lépe škálovatelný, ale jeho nevýhoda je v rychlosti [5]. Ale i přes to je v dnešní době vybírán spíše Nakamotův konsenzus, protože se počítá s tím, že uživatelů v jednom blockchainu může být velmi mnoho, a proto je důležitější škálovatelnost než-li rychlost. V současnosti jsou nejčastěji používány dva algoritmy konsenzu, a to *PoW* (*Proof of Work*) a *PoS* (*Proof of Stake*).

Proof of Work

Úplně první algoritmus konsenzu, který byl použit v kryptoměnách a který se používá dodnes, se nazývá *Proof of Work*. Ten je založený na důkazu vykonání výpočtu určité obtížnosti. Tento důkaz je ale zpětně velmi lehce verifikovatelný. Nejznámějším blockchainem, který používá tento algoritmus, je Bitcoin. Zde je PoW postavený na hashovací funkci *SHA-256*, kde hashe bloků začínají nulovými bity. Počet nul definuje obtížnost výpočtu. Průměrná výpočetní složitost roste exponenciálně s množstvím nulových bitů. Složitost je algoritmicky dána tak, aby každý blok za jakýchkoliv podmínek a změn v blockchainu trval vygenerovat průměrně za 10 minut [7]. Těžař, tedy osoba, která verifikuje blok, se snaží najít hodnotu nonce takovou, aby po výpočtu hashovací funkce, kde jsou vstupem data bloku a nonce, vycházel předdefinovaný formát hashe. Těžař, který tuto úlohu vyřeší jako první, dostává odměnu, která se skládá z poplatků, které platí uživatelé zasílající transakce a z odměn ze samotné sítě. Tato odměna se každých 210000 bloků půlí [7]. Decentralizace zde spočívá v rozdělení výpočetní síly. Velkou nevýhodu tohoto konsenzu přináší jeho kompetitivní naturela. Při použití této techniky tedy vzniká velmi vysoká spotřeba energie. Dalším znepokojivým faktem je možnost útoku, kde by útočník mohl přidávat svoje vlastní bloky za předpokladu, že by ovládal 51 % a více výpočetní síly.

Proof of Stake

Revoluci v konsenzových algoritmech přinesl mechanismus *Proof of Stake*. Zde mizí nutnost pro vysoké výpočetní síly a je nahrazena nutností vlastnit určitý počet tokenů blockchainu a mít je nějakým způsobem uzamčené. Verifikující osoby, zde nazývány jako validátoři, mezi sebou nesoutěží, kdo vyřeší problém jako první, ale je jim pseudonáhodně uděleno právo pro verifikaci bloku. Čím více tokenů má validátor uzamčeno, tím větší pravděpodobnost má, že dostane právo pro verifikaci. Odměnou za verifikaci jsou zde stejně jako u PoW poplatky sítě. Naopak trestem za pokus o nečestné chování není jen spotřeba energie za pokus o útok jako v PoW, ale je jím odebrání části vkladu nečestného uživatele. Decentralizace u PoS spočívá v rozdělení vlastnictví tokenů blockchainu. Nebezpečí útoku u PoS představuje zrcadlově podobný problém jako u PoW jen s rozdílem, že je na něho potřeba vlastnit 51 % a více tokenů blockchainu.

Jedny z nejznámějších blockchainů používajících PoS jsou Cardano nebo Ethereum. Diskutabilně nejlepším PoS je Cardano PoS nazývaný *Ouroboros* [8], kde je vyřešena možná neférovost validátorů vlastnících velký počet tokenů proti validátorům s menším počtem vyřešena tak, že v PoS validují operátoři tzv. vkladových fondů (anglicky *staking pools*). Do těchto fondů může vkládat nejen operátor, ale také jakýkoliv uživatel. Odměny jsou dle množství vložených tokenů rovnoměrně rozděleny mezi členy fondu. Fondy mají nastavenou maximální hranici počtu vložených tokenů. Je-li tato hranice překročena, je pravděpodobnost na udělení práva na validaci snížena, a tím jsou sníženy i celkové odměny. V průměru každý takový fond, který nepřekročí tuto hranici, dostává ročně 5 % z vložených tokenů. Tímto způsobem je zařízena férovost a vyšší decentralizace. Další velkou výhodou je nevazební (anglicky *non-custodial*) způsob vkladu. Tokeny tedy nejsou nijak vázané ani uzamčené a uživatelé je mohou kdykoliv použít.

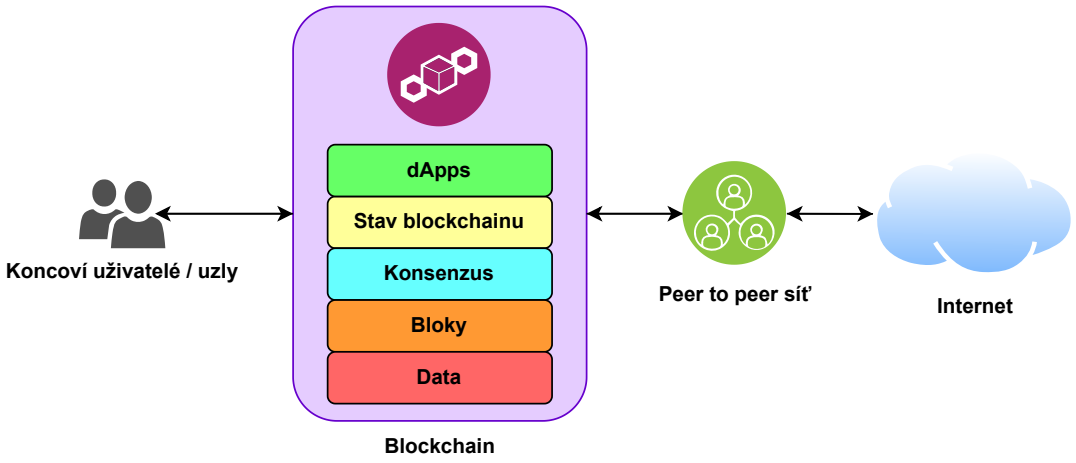
Tab. 1.1: Porovnání výhod PoS a PoW.

	Proof of Stake	Proof of Work
Rychlost	✓	✗
Ekologičnost	✓	✗
Decentralizace	✓	✗
Škálovatelnost	✓	✗
Bezpečnost	✗	✓
Cena	✓	✗
Férovost	✓	✗

V tabulce 1.1 je vyobrazeno porovnání výhod PoS a PoW, kde je vidno, že téměř ve všech směrech je lepší PoS. Jediná výhoda PoW je bezpečnost, která ale závisí na složitosti výpočtu hashe. Prakticky to znamená, že čím déle blockchain existuje a je delší, tím více bezpečný je.

1.3 Síťová komunikace

Blockchain stojí v síťové architektuře na stejné úrovni jako protokoly HTTP (*Hypertext Transfer Protocol*) nebo FTP (*File Transfer Protocol*) běžící nad TCP/IP (*Transmission Control Protocol over Internet Protocol*) [5]. Nejnižší vrstvou je internet sloužící jako základní komunikační vrstva pro celou síť. Nad ním běží peer to peer síť, kde jsou všichni klienti schopni společně komunikovat. Další vrstvou je již samotný blockchain obsahující data, bloky dat, mechanismus konsenzu, stav blockchainu a decentralizované aplikace. Nejvyšší vrstvu představují koncoví uživatelé nebo uzly, kteří blockchain používají. Na obrázku 1.2 je zobrazena síťová komunikace blockchainu.



Obr. 1.2: Síťová komunikace blockchainu.

1.4 Generace blockchainů

Technologie blockchain je poměrně nová, ale díky jejímu rychlému vývoji ji lze dle aplikační pokročilosti rozdělit na 3 generace [9], a to:

1. **Gen 1** - První generací se nazývají blockchainy, které započaly blockchainovou éru. Jsou jimi digitální měny nebo-li kryptoměny. Jejich funkce spočívají čistě

ve finanční sféře a tím jsou zasílání a přijímání aktiv. První implementací kryptoměny byl blockchain Bitcoin.

2. **Gen 2** - Druhá generace blockchainů doplňuje první generaci o chytré kontrakty a jejich schopnosti přidávat podmínky jako nadstavbu k transakcím. Díky programovatelným podmínkám se mohou dvě strany a nebo více stran bezpečně, automaticky a decentralizovaně domluvit. Při vyplnění podmínek stanovených chytrými kontrakty se kontrakt sám spustí. Tato funkcionalita přináší schopnost vytvářet tzv. decentralizované aplikace (anglicky *decentralized applications*, zkráceně *dApps*) a s nimi prakticky jakékoliv programovatelné využití. Příkladem mohou býti funkce pro pokročilejší finanční primitivy a funkce přesahující finanční sféru. Finančními funkcemi a primitivy se zde myslí deriváty, opce, dluhopisy, obchodování, půjčky, úvěry, autorský honorář a podobně [5]. Obecně se tato oblast nazývá decentralizované finance (anglicky *decentralized finance*, zkráceně *DeFi*). Druhá generace tedy přináší blockchainovou platformu pro budování jakýchkoliv aplikací. Blockchain, který tuto generaci započal, se nazývá Ethereum.
3. **Gen 3** - Při vysokém počtu uživatelů se zvyšuje čas i cena používaní služeb, a proto je nutná architektura schopná škálovatelnosti. Dalším problémem je nepřítomnost interoperability mezi různými blockchainy. Třetí a zároveň nejnovější generace řeší tyto dva významné problémy a rozšiřuje druhou generaci o interoperabilitu a škálovatelnost, které v dřívějších generacích chyběly. Zástupci blockchainu generace tři jsou například Cardano a Polkadot.
4. **Gen X** - Generace X představuje vizi do budoucnosti, kdy bude existovat blockchain používaný masovou populací, který je veřejný, otevřený, decentralizovaný, autonomní, interoperabilní a regulovaný kódem, ve kterém by byly implementovány zákony a regulace. Blockchain by řešil nejlépe všechny technické i sociální výzvy moderní doby a poskytoval by služby ve všech kategoriích sociálních oblastí. [5]

1.5 Použití blockchainu

I přes nekonečné možnosti použití, které přináší třetí generace technologie blockchain, se v dnešní době blockchain používá nejčastěji jako digitální měna nebo více obecně jako DeFi. Hojné případy užití jsou také v oblasti nefinanční, například umění, hudba a nebo videohry. Všechny tyto případy užití mají společnou jednu věc, a to je užití technologie nezaměnitelných tokenů (anglicky *Non-fungible Tokens*, zkráceně *NFT*). NFT jsou součástí blockchainu a chovají se velmi podobně jako nativní tokeny sítě, jen s tím rozdílem, že mají schopnost býti nezaměnitelnými nebo-li jedinečnými. NFT je používán jako soubor neměnitelných dat, který obsahuje parametry vhodné

pro jednotlivé případy užití. U umění a hudby jsou těmito daty například IPFS (*InterPlanetary File System*) odkaz obrázku nebo nahrávky, autor, datum vzniku a další. Podobně u videoher, kde hráč vlastní digitální herní předměty, jsou data herních předmětů tokenizována a ukládána decentralizovaně v blockchainu a jsou na rozdíl od tradičního způsobu reálně pod kontrolou samotného uživatele. Největší výhodou této technologie je tudíž bezpečný digitální důkaz o vlastnictví a původu, který je velmi těžko podvrhnutelný. Na obrázku 1.3 je vyobrazen příklad užití NFT v oblasti umění.

Obr. 1.3: Příklad užití NFT.

Potenciál blockchainu sahá dále než jen do případů užití aktuální doby. V budoucnu by mohly k případům použití patřit například decentralizované služby orgánů státní správy. V zemích třetího světa je často velmi těžké prokázat vlastnictví pozemků, například když se vláda snaží vyvlastnit své občany. Tento problém lze vyřešit implementací vlastnictví do blockchainu [4].

2 Bitcoin

Bitcoin představuje úplně první peer to peer elektronický platební systém založený na technologii blockchain. Počátek Bitcoinu nastal v roce 2008, kdy zakladatel Satoshi Nakamoto zveřejnil autoritativní zprávu (anglicky *white paper*)[7], která popisuje myšlenky a technologie stojící za jeho systémem. Vytvoření prvního bloku a na něj navazujících, tedy vznik blockchainu, nastal v roce 2009. Účelem vzniku této technologie bylo vyřešit tehdejší a budoucí problémy ve finanční sféře. Autor popsal tyto problémy jako naivní důvěru mezi kupujícím a prodávajícím a nebo ve třetí stranu, která spravuje transakce, zbytečnost existence třetí strany, a také možnost zvrácení platby, kde se počítá s nevyhnutelným procentem podvodů. Cílem bylo tedy vytvořit kryptograficky bezpečný peer to peer finanční systém, který by vytvářel nepřekonatelnou a matematicky ověřenou důvěru mezi uživateli. Tím by se z koloběhu vyřadily třetí strany, čímž by se snížily náklady na provoz systému, doba trvání transakcí a umožnilo by se zasílání malých transakcí. Prvotní myšlenky a názory zakladatele se projevují v prvotní i dnešní architektuře Bitcoinu. Architektura kopíruje vlastnosti decentralizovaného systému, ke kterým jsou přidané další vlastnosti pro zesílení Nakamotovy idey o finančním systému. Těmito vlastnostmi jsou například [10]:

1. **Transparentnost** - Všechna data v blockchainu jsou veřejná. Každý může sledovat všechny transakce a popřípadě je i verifikovat. Velmi diskutabilní je v tomto případě soukromí uživatelů.
2. **Maximální počet mincí** - Bitcoin má nastaven maximální počet mincí na 21 milionů. Každý celý Bitcoin se dá rozdělit na miliontiny, které se nazývají Satoshi. V době psaní této práce je v oběhu 19208256 bitcoinů a další se do oběhu kontrolovatelně dostávají pomocí algoritmu konsenzu PoW. Oproti klasickým měnám není nekontrolovaně inflační a lze se spolehnout na to, že měna nebude kvůli vnějším vlivům nijak ztrácet na hodnotě z hlediska počtu mincí v oběhu. Prakticky je Bitcoin jako měna deflační z důvodu lidské nedokonalosti, která zaviňuje občasné ztráty privátních klíčů nebo zaslání bitcoinů na neznámou adresu. Oboje znamená ztrátu bitcoinů, které nelze již zpátky získat.
3. **Moderní škálovatelné kryptografické protokoly** - Nekonečná snaha vylepšovat blockchain přináší postupem času transakce rychlejší, bezpečnější a s vyšší úrovní soukromí. Příkladem je *Taproot hard fork* [11][12][13].
4. **Non-stop aktivní systém** - Za předpokladu, že existuje alespoň jeden uzel aktivně těžící bloky, je Bitcoin online. S faktem, že čím méně je konkurenčních těžících uzlů, tím více je těžení bloků lukrativní, lze předpokládat, že se bude vždy někdo aktivně podílet na těžení a běhu blockchainu. Z toho vyplývá, že

na rozdíl od tradičních finančních institutů, je možné aktiva zasílat kdykoliv, kdy si osoba přeje.

Prvotní architektura Bitcoinu se skládá z definice transakce a jejího zabezpečení, řešení problému dvojitého utrácení (anglicky *double spending*), chodu sítě, podnětu pro čestné chování a podporu sítě, ukládání dat do disku, zjednodušené verifikace transakcí, kombinování a rozdělování hodnoty, soukromí a pravděpodobnosti útoku [7].

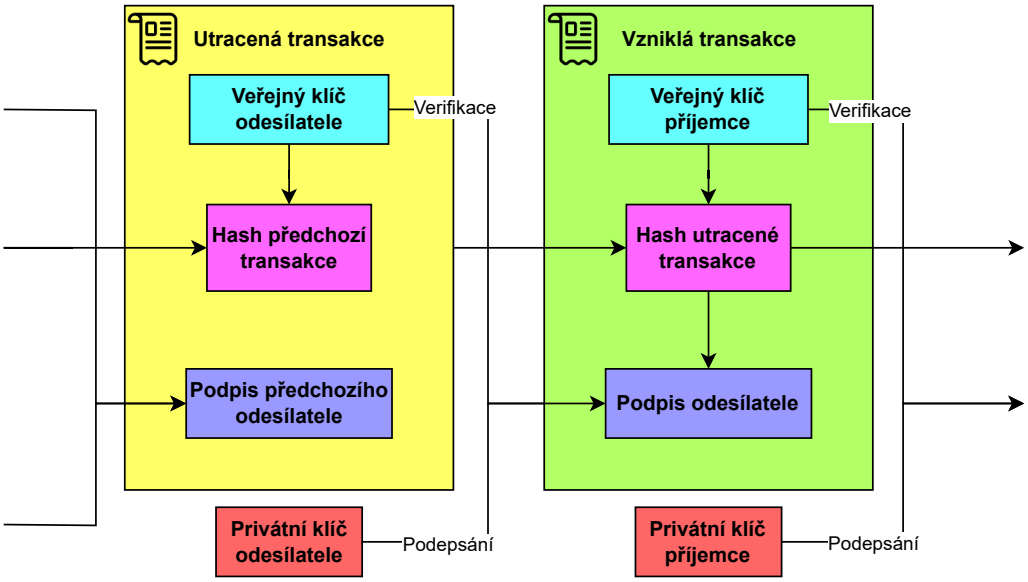
Vlastnictví digitální měny je dáno daty v transakci. To znamená, že ten, kdo vlastní transakci k aktivům, vlastní i samotná aktiva. Transakce měny mezi uživateli se provádí pomocí digitálního podpisu hashe, předchozí transakce a veřejného klíče příjemce. Takto vzniká řetězec podpisů, který představuje historii vlastnictví. Příjemce nebo kdokoliv, kdo vidí transakci, může verifikovat tento řetězec podpisů pro verifikování řetězce vlastnictví. Originálně se pro digitální podpis používal algoritmus *ECDSA* (*Elliptic Curve Digital Signature Algorithm*) s křivkou *secp256k1*. Po *Taproot hardforku* se zde změnil algoritmus podpisu na *Schnorrův podpis* s původní křivkou.

Problém nastává ve dvojitém utrácení, kdy odesílatel zasílá svoje aktiva dvěma či více příjemcům zároveň. Pro kontrolu dvojitého utrácení je zavedeno pravidlo, že je platná jen poslední transakce, lze tedy utratit jen poslední transakci. Z toho důvodu je nutné mít přehled o historii všech transakcí a systém, který by všem účastníkům umožňoval domluvu na jednotné historii transakce postavené na nějakém řádu. Příjemce by měl mít možnost verifikací řetězce získat nejen důkaz o historii vlastnictví, ale také jistotu, že je tato verze shodná s verzí, která je domluvená mezi ostatními účastníky. Na obrázku 2.1 je zobrazen obecný princip provádění transakcí.

První částí řešení problému dvojitého utrácení je server časového razítka. Jeho účelem je důkaz existence transakce v čase. Princip je takový, že server hashuje blok transakcí a k němu přidá časové razítko. Tyto data jsou zveřejněna serverem pro případnou verifikaci. Každé další časové razítko obsahuje časové razítko předcházející. Tudíž s každým časovým razítkem se více zabezpečuje historie časových razítek a tím i historie transakcí. Druhou částí řešení, která kompletuje řešení, je rozšíření serveru časového razítka na bázi peer to peer pro domluvu mezi uzly sítě, čímž je algoritmus konsenzu. U Bitcoinu je vybrán algoritmus PoW.

Další důležitou částí architektury Bitcoinu je jeho chod sítě, který se skládá z následujících kroků [5]:

1. **Nové transakce jsou zasílány všech uzlům.** Není ale nutné, aby transakci dostaly všechny uzly.
2. **Každý uzel sbírá nové transakce do bloků.**
3. **Každý uzel se snaží najít řešení algoritmu PoW.**
4. **Když nějaký uzel najde řešení, tak jej včetně celého bloku zasílá**



Obr. 2.1: Obecný princip provádění transakcí.

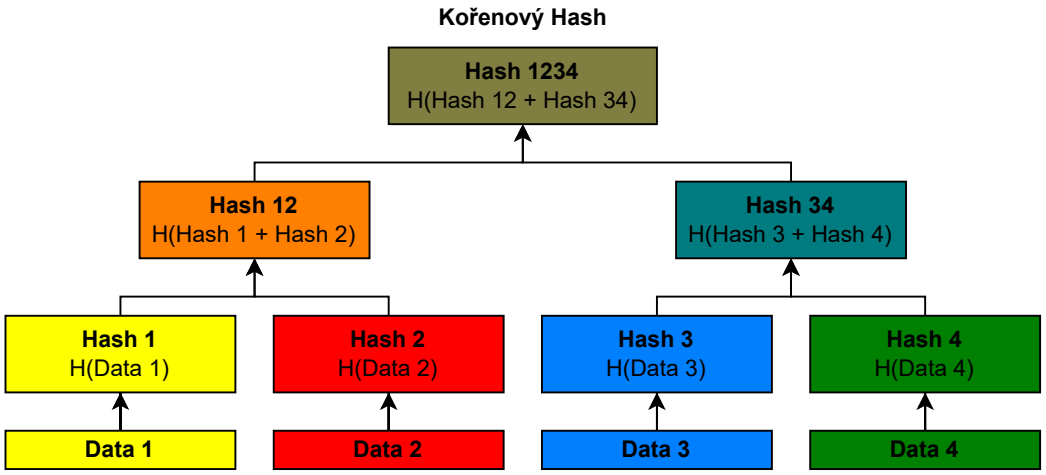
všem ostatním uzlům.

5. **Uzly přijímají blok jen tehdy, když všechny transakce v bloku jsou platné a nejsou již utracené. Tedy že nedošlo k dvojímu utracení.** Může se stát, že uzel nedostane blok. Když si při přijetí dalšího bloku uvědomí, že mu blok chybí, tak o něj požádá.
6. **Uzly vyjadřují přijmutí bloku tím, že pro tvorbu dalšího bloku použijí hash přijatého bloku jako předchozí hash.**

Čestné uzly se vždy snaží přidávat bloky k nejdelší verzi blockchainu, ale může nastat situace, kdy dva čestné uzly zašlou dvě různé ale validní verze blockchainu. Některé uzly dostanou dříve jednu verzi a jiné zase druhou verzi. Tímto vzniká kolize a nejistota v hlavní řetězec. V tomto případě uzly přidávají první přijatý blok do blockchainu, ale zároveň si ukládají i druhou větev pro případ, že by byl pak delší. Tato kolize se vyřeší v moment, kdy je nalezeno následující řešení PoW a je přiřazen nový blok. Tím se stane jedna ze dvou větví delší a ta se stane hlavní větví, ve které všechny uzly dále pokračují. Čestné chování uzlů a podporu sítě podněcuje charakter PoW a jeho odměny.

S novými transakcemi a bloky se zvyšuje objem dat nutných k ukládání. Pro snížení ukládaných dat je v Bitcoinu dáno pravidlo pro vyřazení starých transakcí. Je-li po poslední utracené transakci s aktivy dost bloků, tak jsou všechny utracené transakce před poslední smazány. Aby nebyla porušena celistvost blockchainu a hashe, jsou transakce hashovány v hashovém stromu, také zvaném Merkleův strom. Do dat bloku je zapsán jen kořenový hash. Na obrázku 2.2 je vyobrazen příklad

Merkleova stromu.



Obr. 2.2: Merkleův strom.

Bitcoin umožňuje zjednodušenou verifikaci transakcí bez nutnosti vlastnit celý uzel. Uživatel ke zjednodušené verifikaci potřebuje jen hlavičky bloků a Merkleho větev vztahující se na transakci a blok, ve kterém je transakce uložena. Verifikace je provedena kontrolou, že nějaký uzel sítě akceptoval blok s transakcí, a že existuje spojení s dalšími následujícími bloky. Tato verifikace funguje za předpokladu, že čestné uzly kontrolují blockchain. Zjednodušenou verifikací nelze totiž zjistit, jestli je transakce fabrikovaná útočníkem, a nebo jestli je síť kontrolována útočníkem nebo ne.

Pro manipulaci s digitální měnou uvnitř transakce je v Bitcoinu vytvořena konstrukce umožňující kombinování a rozdělování hodnoty měny. Tato konstrukce udává transakci vstupy (anglicky *inputs*) a výstupy (anglicky *outputs*). Vstupy představují předchozí výstupy, které se v transakci utrácejí. Těmto výstupům transakcí se říká UTXOs (*Unspent transaction outputs*). V transakci je většinou buď jeden vstup s velkou sumou mincí a nebo vyšší počet vstupů s menší sumou mincí. Výstupy jsou maximálně dva. V případě, že je v transakci jen jeden výstup, znamená to, že všechny mince jsou utraceny. V případě, že jsou v transakci dva výstupy, tak je jeden určen pro zaslání mincí příjemci a druhý pro zaslání zbytku peněz zpátky odesílateli. Pravidlem je, že součet všech vstupů se rovná součtu všech výstupů a transakčního poplatku sítě.

Všechny transakce jsou u Bitcoinu veřejné. Tento fakt způsobuje vysoké snížení soukromí. Ale i přesto existuje možnost skrýt některé informace před veřejností. Jestliže není znám veřejný klíč, není možné spojit transakci s uživatelem. Z tohoto důvodu je doporučeno při každé transakci vygenerovat nový pár soukromého a ve-

řejného klíče. Tento přístup ale neřeší situaci, kdy je v transakci vícero vstupů. Při takové transakci je pak jasné, že tyto vstupy vlastnila stejná osoba. Jsou-li v transakci dva výstupy, lze snadno odhadnout, jaký z výstupů je zbytek a tím získat znalost o dalších transakcích s ním spojených.

Útok na síť může nastat v případě, kdy je útočník schopen přidávat bloky rychleji než čestní účastníci. Tedy pokud má útočník vyšší pravděpodobnost na přidání bloku než čestní účastníci, pak je blockchain kompromitován a dále se již nedá nic dělat. V případě, že útočník má menší pravděpodobnost na přidání bloku, se dále výpočet pravděpodobnosti úspěšného útoku počítá pomocí Poissonova rozdělení, které zde udává útočníkův potenciální pokrok. Tato hodnota se počítá jako:

q =pravděpodobnost útočníka přidat další blok

p =pravděpodobnost čestného účastníka přidat další blok

z =počet bloků, o které je útočník pozadu

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

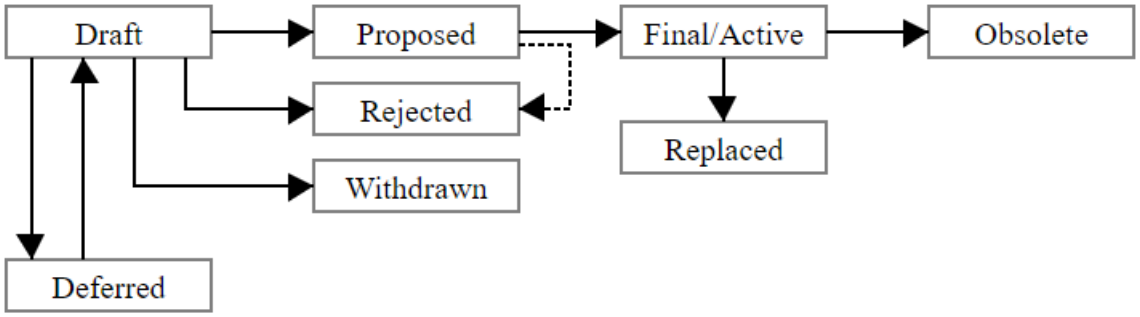
Pravděpodobnost útočníka předstihnout čestné účastníky se pak počítá jako:

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

Za poměrně dlouhou dobu existence Bitcoinu se jeho komunita již zasloužila o mnoho vylepšení, která zlepšují standardizaci, samotný protokol, soukromí a nebo uživatelskou přívětivost. Proces vylepšení začíná dokumentem, kterému se říká BIP (*Bitcoin Improvement Proposal*). Samotný termín, struktura, typy, proces vylepšování a další prvky charakterizující BIP byly prvotně definovány v BIP 1, ten byl později nahrazen BIPem 2, který je dnes aktuální. Dle BIPu 2[14] by měl každý BIP poskytovat informace nebo popisovat novou funkcionalitu nebo její proces nebo prostředí. Měl by poskytovat stručný technický popis funkcionality a odůvodnění pro existenci dané funkcionality. Existují tři typy BIPů:

1. **Standardizující** - Tento BIP navrhuje a popisuje změny, které ovlivňují jakoukoliv funkcionalitu Bitcoinu. Pro aktivaci navrhovaných změn je nutné konsenzus uživatelů.
2. **Informační** - Informační BIP poskytuje různé směrnice, doporučení a nebo jen informuje o některých faktech nebo problémech. Avšak dále nenavrhuje žádné vylepšení nebo řešení. Uživatelé je tedy mohou brát vážně a nebo je ignorovat.
3. **Procesní** - Procesní BIP je podobný jako standardizující BIP. Tedy navrhuje nové funkcionality nebo změny funkcionalit za účelem vylepšení celkové nebo částečné funkcionality Bitcoinu. Rozdílem je, že se neaplikuje na protokol Bitcoinu.

V BIPu 2 jsou dále graficky specifikovány možné statusy BIPu nebo je rovněž možné říct životní cyklus BIPu. Schéma životního cyklu BIPu je vyobrazeno na obrázku 2.3 [14].



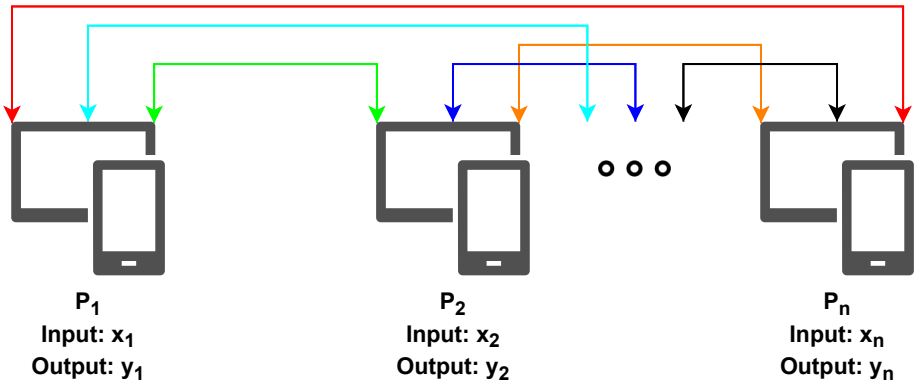
Obr. 2.3: BIP 2 - statusy BIPu.

3 Kryptografické metody a jejich integrace do blockchainu

V této kapitole jsou probírány existující kryptografická primitiva a metody, dále jejich možné užití a integrace do technologie blockchain. Tato primitiva by měla zlepšovat bezpečnost, soukromí a minimalizaci dat, což jsou jedny z nejdůležitějších vlastností týkající se blockchainové technologie.

3.1 Zabezpečené počítání s více stranami

SMPC (*Secure multi-party computation*) nebo-li zabezpečené počítání s více stranami je termín, který umožňuje dvoum či více stranám bezpečně a společně vypočítat jimi zvolené funkce s tím, že nejsou jejich soukromé vstupy ani výstupy prozrazeny. Funkcemi se zde rozumí téměř jakékoliv kryptografické úkony, jako například autentizace, sdílení tajemství, důkaz nulové znalosti, šifrování, schémata závazků a další. Účastníci v SMPC jsou definováni jako $Pi(i = 1, ..., n)$, jejich soukromé vstupy jako x_i a společná funkce jako $f(x_1, x_2, ..., x_n) = (y_1, y_2, ..., y_n)$, které jsou vypočítané pomocí jejich soukromých vstupů. Každý účastník by měl na konci výpočetního procesu dostávat svůj výstup y_i [15]. Diagram popisující SMPC je znázorněn na obrázku 3.1.



Obr. 3.1: Diagram Secure multi-party computation.

Velkou výhodou SMPC je tedy zachování soukromí všech zúčastněných stran. Tuto technologii je vhodné použít například pro elektronické volby, skupinové podpisy, anonymní zpracování a analýzu osobních údajů a nebo strojové učení. Tato práce se věnuje případu užití v kontextu vypočítání společného veřejného výstupu pomocí soukromých vstupů každého z účastněných.

Za hlavní bezpečností hrozbu SMPC se považuje nečestný účastník SMPC. Aby SMPC bylo možné brát jako bezpečné, tak musí prvně splňovat pět základních bezpečnostních vlastností, které ale nemusí rovnou mitigovat nebo úplně eliminovat vektory útoku nečestného účastníka. Těmito vlastnostmi jsou [15]:

1. **Soukromí** - Žádný účastník SMPC by neměl jakýmkoliv způsobem získat soukromá data jiných účastníků. Tedy každý účastník by měl znát jen svoje výstupy a informace, které lze derivovat ze svých vlastních vstupů a výstupů.
2. **Správnost** - Měla by být zajištěna správnost všech výstupů.
3. **Nezávislost vstupů** - Vstupy, které zvolí nečestný účastník SMPC, by měly být nezávislé k výstupům čestných účastníků.
4. **Garantový výstup** - Čestní účastníci by měli mít garantované získání svých výstupů. Útočník by tedy neměl být nijak schopný zabránit procesu SMPC pomocí jakýchkoliv útoků.
5. **Férovost** - Každý účastník SMPC by měl dostat svůj výstup, jestliže ostatní účastníci, včetně nečestných účastníků, dostali svůj výstup.

3.2 Schnorrův podpis

Schnorrův podpis [16] je digitální podpis derivovaný na základě Schnorrově identifikačního protokolu [17] pomocí Fiat-Shamir heuristiky. Toto schéma jednoduše a efektivně vytváří podpisy krátké délky. Bezpečnost je založená na problému diskrétního logaritmu. Schéma je parametrizováno následovně [18]:

$\mathbb{G}$ = grupa řádu p

g = generátor grupy $\mathbb{G}$

$H : \mathbb{G} \times \{0,1\}^k \rightarrow \mathbb{Z}_p$ = hashovací funkce

Schéma má tři fáze, a to:

1. **Generování páru klíčů** - Generování klíčů je vyobrazeno v algoritmu 1.

Algoritmus 1 Schnorr generování páru klíčů

1: Uživatel si náhodně vygeneruje hodnotu $x \leftarrow \mathbb{Z}_p$.
2: Vypočítá se $X = g^x$.
3: Pár veřejného a soukromého klíče je pak (X, x).

2. **Podpis** - Do této fáze vstupuje soukromý klíč x a zpráva m viz algoritmus 2.
3. **Verifikace** - Pro verifikaci jsou zapotřebí hodnoty X, m a podpis (R, y). Verifikující kontroluje, zda je následující rovnice splněna:

$$g^y = R * X^c$$

Algoritmus 2 Schnorr podpis

1: Uživatel si náhodně vygeneruje hodnotu $r \leftarrow \mathbb{Z}_p$.
2: Vypočítá se $R = g^r$.
3: Vypočítá se $c = H(R, m)$.
4: Vypočítá se $y = x * c + r \bmod p$.
5: Podpisem je pak (R, y).

Ve vztahu k blockchainové technologii Schnorrův podpis přináší mnohé výhody díky své schopnosti agregovat klíče. Agregovaný klíč se jeví jako obyčejný klíč podepsaný jedním uživatelem, má tedy stejnou velikost a stejně se verifikuje. Příkladem možnosti použití je [19], kde je Schnorrův podpis využit pro vícenásobné podpisy v blockchainu Bitcoin. I přes funkční implementaci je obyčejný Schnorrův agregovaný podpis považován za slabší možnost využití.

3.3 Shamirův protokol pro sdílení tajemství

Shamirův protokol pro sdílení tajemství (anglicky Shamir's Secret Sharing Protocol) [20][21] je kryptografický algoritmus, který umožňuje rozdělit tajemství na několik částí a distribuovat je mezi skupinu lidí tak, aby bylo nutné mít k dispozici alespoň určitý počet těchto částí pro získání celého tajemství. Tento protokol byl vyvinut v roce 1979 izraelským matematikem Adi Shamirem. V kryptografii má využití hlavně u prahových podpisů.

Protokol je založen na teorii polynomů a teorii konečných těles (anglicky finite fields). Podstatou protokolu je tedy vytvoření polynomu stupně $n-1$, kde n je počet částí, na které je tajemství rozděleno. Tento polynom má náhodně vygenerované koeficienty, z nichž nejvyšší je koeficient, který určuje celkové tajemství. Každá část tajemství je určena jako jeden bod na křivce, kterou tvoří tento polynom. Tyto body jsou poté distribuovány mezi účastníky protokolu. Detailní postup rozdělení tajemství je následující:

1. Volba tajemství S a prahového čísla t.
2. Vytvoření polynomu $P(x)$ stupně $t-1$. Koeficienty $a_0, a_1, \ldots, a_{t-1}$ jsou vybrány náhodně z rozsahu konečného pole, na které se operuje. Koeficient $a_0 = S$.

$$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1}$$

3. Pro každého účastníka s indexem i se vypočítá bod $P(i)$. Tento bod představuje část celkového tajemství.

Pro získání celkového tajemství je zapotřebí znát alespoň t bodů na křivce, kde t je prahový počet. Celkové tajemství je pak získáno interpolací polynomu ze známých

bodů na křivce.

$$S = \sum_{j=1}^{t} P(i_j) * \prod_{m=1, m \neq j}^{t} \frac{i_m}{i_m - i_j}$$

3.4 Paillierův kryptosystém

Paillierův kryptosystém (anglicky Paillier cryptosystem) je pravděpodobnostní asymetrický šifrovací algoritmus založený na problému rozhodování kvadratického residua. Tento kryptosystém byl navržen v roce 1999 Pascalem Paillierem[22]. Jeho hlavní použití spočívá ve využití vlastnosti homomorfního šifrování, tedy je možné provádět operace nad zašifrovanými daty bez nutnosti dešifrování.

Schéma má tři fáze, a to:

1. **Generování páru klíčů** - Generování klíčů je vyobrazeno v algoritmu 3.

Algoritmus 3 Paillier generování páru klíčů

1: Uživatel si náhodně vygeneruje dvě náhodná dostatečně velká a bezpečná prvočísla p a q.
2: Vypočítá se $n = p * q$, $\lambda = (p-1) * (q-1)$ a $g = n + 1$.
3: Veřejným klíčem je pár (n, g) a privátním klíčem je λ.

2. **Šifrování** - Do této fáze vstupuje veřejný klíč (n, g) a zpráva $m < n$, viz algoritmus 4.

Algoritmus 4 Paillier šifrování

1: Uživatel si náhodně vygeneruje hodnotu $r < n$.
2: Šifrovaná zpráva se pak počítá jako $c = g^m * r^n \mod n^2$

3. **Dešifrování** - Pro dešifrování jsou zapotřebí hodnoty c, λ, n a g. Původní zpráva m je pak spočítána jako:

$$m = \frac{L(c^\lambda \mod n^2)}{L(g^\lambda \mod n^2)} \mod n$$

, kde L je Lagrangeova funkce.

3.5 Vícenásobné podpisy

Vícenásobné podpisy (anglicky *multi signatures*) jsou kryptografické primitivum patřící do SMPC, které dávají schopnost skupině uživatelů společně podepsat nějakou zprávu. U tohoto podpisu je nutné, aby se každý ze členů skupiny podílel na tvorbě

podpisu. To je definováno jako *n-of-n*, kde n je počet členů skupiny. Velikost podpisu je vždy stejná při jakémkoliv počtu uživatelů ve skupině. Další výhodou je anonymita uživatelů ve skupině, jelikož je podpis vždy stejně velký, tak není možné říci kolik uživatelů a ani jací uživatelé se podíleli na podpisu. Vícenásobné podpisy jsou více efektivní, jsou-li veřejné klíče uživatelů agregovány do jednoho veřejného klíče, který vypadá jako veřejný klíč jednoho uživatele a podpis se dá verifikovat stejně jako podpis jednoho uživatele [23]. Použití vícenásobných podpisů je vhodné tam, kde je žádoucí vyšší úroveň bezpečí a soukromí v kombinaci s minimalizací dat. Vícenásobné podpisy jsou tudíž velmi dobrou volbou pro použití v technologii blockchain.

V některých textech se o *multi signature* mluví jako o metodě, kde je pro autorizaci potřeba větší množství klíčů, ale na rozdíl od prvního případu se zde nepoužívá agregace klíčů a hodnot, nýbrž je vytvořená struktura, ve které figuruje větší množství klíčů, jsou definovaná pravidla pro verifikaci a zprávu podepisuje každý uživatel zvlášť. Tudíž za každého uživatele je vytvořen jeden podpis. Často jsou i tyto struktury včetně počtu uživatelů a pravidel veřejná. Z toho vyplývá, že v tomto případě sice zůstává nějaká úroveň bezpečnosti, ale mizí výhody soukromí a minimalizace dat. Díky nastavitelným pravidlům lze ale vytvořit verifikaci podobnou prahovým podpisům, není tedy nutné, aby se každý ze skupiny zúčastnil tvorby podpisu. Tento způsob používají například blockchainy Bitcoin[24] a Cardano[25]. Tento způsob je vnímám jako nejslabší varianta k vícenásobným podpisům.

3.5.1 MuSig

MuSig je kryptografické schéma vícenásobných podpisů, které bylo vytvořeno pro účely rozšíření funkcionalit Bitcoinových transakcí. Schéma bylo představeno v [26]. Toto schéma je založeno na Schnorrově podpisu a využívá jeho schopnost agregovat klíče. Schéma je prokazatelně bezpečné v kryptografickém modelu veřejných klíčů, tedy účastníci podpisu mají společný veřejný klíč, ale nemusí prokazovat znalost privátního klíče spojeného se společným veřejným klíčem. Celý proces se skládá ze tří komunikačních rund mezi uživateli. MuSig je parametrizován parametry grupy (G, p, q), kde G je cyklická grupa řádu p a g je generátor G. Dále zde vystupují tři hashe, a to H_{com}, H_{agg} a H_{sig}. H_{com} se používá při fázi závazku, H_{agg} se používá při agregaci klíčů a H_{sig} se používá při vytváření podpisu. Schéma má tři fáze, a to:

1. **Generování klíčů** - Každý uživatel si vygeneruje svůj vlastní pár klíčů. K náhodně vygenerovanému klíči x je vypočítán veřejný klíč jako $X = g^x$.
2. **Podepisování** - Skupina uživatelů s privátním klíčem x_i a veřejným klíčem X_i chtějí podepsat zprávu m. H představuje hashovací funkci.

Algoritmus 5 MuSig podepisování

1: Vypočítá se $L = H(X_1, X_2, \ldots, X_n)$

2: Každý ze skupiny uživatelů vypočítá $a_i = H_{agg}(L, X_i)$. Hodnotu a_i si mezi sebou pošlou. (První runda)

3: Vypočítá se $X = \sum_{i=1}^{n} X_i^{a_i}$, kde X je společný agregovaný veřejný klíč skupiny uživatelů.

4: Každý ze skupiny uživatelů si vygeneruje náhodné číslo r_i a vypočítá $R_i = g^{r_i}$ a $t_i = H_{com}(R_i)$. Hodnotu R_i a t_i si mezi sebou pošlou. (Druhá runda)

5: Po přijetí hodnot R_i a t_i si každý uživatel pro všechny hodnoty zkontroluje, jestli platí $t_i = H_{com}(R_i)$. Je-li vše v pořádku, pokračuje se dál. Pokud nastane neshoda, je protokol ukončen.

6: Vypočítá se $R = \sum_{i=1}^{n} R_i$.

7: Vypočítá se $c = H_{sig}(X, R, m)$.

8: Vypočítá se $s_i = r_i + c * a_i * x_i \ mod \ p$ a pošle se ostatním uživatelům. (Třetí runda)

9: Vypočítá se $s = \sum_{i=1}^{n} s_i$.

10: Podpis představuje $\sigma = (R, s)$

3. **Verifikace** - Verifikace je podobná klasické verifikaci Schnorrova podpisu. Verifikující kontroluje, zda je následující rovnice splněna:

$$g^s = R * \sum_{i=1}^{n} X_1^{a_i c} = R * X^c$$

3.5.2 MuSig2

MuSig2[27] je vylepšený pokračovatel schématu MuSig. Rozdílem mezi nimi je počet rund, které jsou potřeba pro vytvoření vícenásobného podpisu. Tento protokol je prvním, který splňuje všechny následující vlastnosti:

1. Je bezpečný při souběžných podepisovacích relacích.
2. Podporuje agregaci klíčů.
3. Jeho výstupem je klasický Schnorrův podpis.
4. Potřebuje jenom 2 rundy pro vytvoření vícenásobného podpisu.
5. Složitost podepisování je podobná jako u klasického Schnorrova podpisu.

Stejně jako u schématu MuSig má MuSig2 fáze generování klíčů, podpisu a verifikace. Generování klíčů a verifikace jsou shodné, rozdílem je tedy pouze fáze podepisování, které je vyobrazeno v algoritmu 6. Rozdíly jsou zvýrazněny zeleně.

Algoritmus 6 MuSig2 podepisování

Výpočet společného veřejného klíče.

1: Vypočítá se $L = H(X_1, X_2, \ldots, X_n)$.
2: Každý ze skupiny uživatelů vypočítá $a_i = H_{agg}(L, X_i)$. Hodnotu a_i si mezi sebou pošlou. (První runda)
3: Vypočítá se $X = \sum_{i=1}^{n} X_i^{a_i}$, kde X je společný agregovaný veřejný klíč skupiny uživatelů.

První část podpisu.

1: Pro každé $j \in (1, \ldots, \nu)$, kde ν je počet náhodností, si každý podepisující vygeneruje náhodnou hodnotu $r_{1,j}$ a vypočítá $R_{1,j} = g_{1,j}^{r}$.
2: Výstupem jsou ν náhodnosti $(R_{1,1}, \ldots, R_{1,\nu})$, které jsou zaslány agregátorovi.
3: Agregátor vypočítá agregovanou hodnotu $R_j = \prod_{i=1}^{n} R_{i,j}$ pro každé $j \in (1, \ldots, \nu)$. Výstupem jsou $(R_1, \ldots, R_\nu)$.

Druhá část podpisu.

1: Podepisující pro zprávu m vypočítá hodnotu $b = H_{non}(X, (R_1, \ldots, R_\nu), m)$.
2: Dále vypočítá hodnotu $R = \prod_{j=1}^{\nu} R_j^{b^{j-1}}$.
3: Dále vypočítá hodnotu $c = H_{sig}(X, R, m)$.
4: Dále vypočítá hodnotu $s_1 = c * a_1 * x_1 + \sum_{j=1}^{\nu} r_{1,j} * b^{j-1} \mod p$. Hodnota s_1 představuje částečný podpis.
5: Všechny částečné podpisy se zasílají agregátorovi. (Druhá runda)

Agregace částečných podpisů.

1: Agregátor agreguje všechny částečné podpisy jako hodnotu $s = \sum_{i=1}^{n} s_i \mod p$.
2: Společným podpisem je pak $\sigma = (R, s)$.

3.6 Prahové podpisy

Prahové podpisy (anglicky threshold signatures) jsou takové digitální podpisy, kde vystupuje skupina uživatelů vytvářející jeden společný podpis. Obecně se definuje (t, n)-prahový podpis, kde t nebo více členů ze skupiny n uživatelů mohou vytvářet podpisy ve jménu skupiny [28]. Tedy každá podskupina původní skupiny s minimálně t členy je schopna vytvářet podpisy ve jménu skupiny. Identita členů skupiny, kteří vytvořili společný podpis, je skryta podobně jako u vícenásobných podpisů. Vícenásobné podpisy jsou speciální případ prahových podpisů, kdy $t = n$. Dalším speciálním případem je $t, n = 1$. V tomto případě se jedná o obyčejný podpis vytvořený jedním uživatelem. Skupina uživatelů může být spravována nějakou důvěryhodnou autoritou nebo může rozhodovat skupina sama o své správě, kdy každý člen má stejná práva.

Cílem prahových podpisů je zvýšit bezpečnost transakcí tím, že zvýší počet nut-

ných účastníků při podepisování dat, tedy $t > 1$, nebo eliminovat slabé místo v systému, které je kritické pro jeho chod (anglicky *single point of failure*), tedy $n > 1$.

Každý uživatel ve skupině podepisujících má svůj vlastní pár veřejného a soukromého klíče. Do skupiny se ale uživatel zařadí tak, že předloží svůj veřejný klíč a vhodné parametry. Skupina nebo důvěryhodná autorita po získání všech hodnot od uživatelů, kteří chtějí být spolu ve skupině, vypočítá společný veřejný klíč skupiny. Při podepisování každý člen podepisující podskupiny vypočítá svůj částečný podpis a předá ho určené entitě, která z částečných podpisů vytváří výsledný společný podpis. Verifikovat podpis může každý, kdo zná veřejný klíč skupiny.

Celkově mají prahové podpisy 5 operací, a to operaci pro generování dvojic privátního a soukromého klíče člena skupiny, operaci pro generování společných hodnot skupiny, operaci pro správu skupiny, operaci pro vytváření společného podpisu a operaci pro verifikování podpisu.

Prahové podpisy by měly splňovat následující bezpečnostní vlastnosti [28]:

1. **Nefalšovatelnost (anglicky *unforgeability*)** - Nemělo by být možné falšovat podpis výpočetně omezenými útočníky.
2. **Práh (anglicky *threshold*)** - Jedině podskupiny s t nebo více členy z celkového n počtu členů mohou vytvářet podpis.
3. **Robustnost (anglicky *robustness*)** - Jsou-li všechny použité hodnoty členů skupiny k výpočtu společného podpisu validní, je i společný podpis validní.
4. **Proaktivní bezpečnost (anglicky *proactive security*)** - Členové skupiny mohou měnit svoje klíče bez toho, aby byl změněn společný veřejný klíč skupiny.
5. **Sledovatelnost (anglicky *traceability*)** - Tato volitelná vlastnost říká, že nečestné podskupiny uživatelů nemohou vytvořit podpis bez toho, aby byla odhalena jejich identita.

Jelikož jsou prahové podpisy lepší variantou pro vícenásobné podpisy a řeší větší spektrum problémů, tak jsou velmi vhodné pro užití v blockchainové technologii. Prahové podpisy jsou tudíž brány jako nejlepší způsob zvýšení bezpečnosti a anonymity u blockchainových transakcí. S lepší funkcionalitou ale také přichází složitější implementace.

3.7 Srovnání existujících řešení

V této sekci jsou srovnány některé blockchainové peněženky, které nějakým způsobem podporují vícenásobné podpisy nebo prahové podpisy a nebo jakékoliv použití SMPC.

Gnosis-Safe

Velmi známou peněženkou využívající vícenásobné podpisy je Gnosis-Safe peněženka běžící jako webová aplikace, desktopová aplikace a mobilní aplikace. Tato peněženka podporuje blockchainy Ethereum, Polygon, Avalanche, Arbitrum, Optimism a Binance Smart Chain. Kód je open-source a formálně verifikován. Peněženka využívá chytré kontrakty ke tvorbě peněženky a k verifikaci transakcí s vícenásobnými podpisy [29]. Díky chytrým transakcím je zde možné vytvořit pravidlo pro prahovou funkcionalitu. Vzhledem k tomu, že peněženka nepoužívá žádná schémata SMPC, postrádá výhody uchování soukromí a minimalizace dat.

Electrum

Jednou z nejstarších Bitcoinových peněženek je desktopová aplikace Electrum [30]. Peněženka podporuje vícenásobné podpisy ve smyslu neagregovaného klíče. Tudíž stejně jako u Gnosis-Safe peněženky nevylepšuje soukromí ani neminimalizuje data, ale rovněž přidává možnost prahové funkcionality.

ZenGo

Velmi zajímavou možností na trhu je peněženka ZenGo [31]. Peněženka běží jako mobilní aplikace a podporuje přes 60 kryptoměn. Zajímavostí této mobilní peněženky je, že používá SMPC, kdy uživatel nedostává při registraci žádné privátní klíče ani mnemonic kód pro obnovu svých aktiv. Pro obnovu je zapotřebí 3D biometrický scan obličeje, který uživatel provedl při registraci. Transakce jsou zabezpečeny pomocí 2-of-2 vícenásobného podpisu založeného na ECDSA. Jeden klíč je uložen v mobilním zařízení a druhý je uložen na serveru ZenGo. Všechna kryptografie je dle ZenGo open-source [32]. Tím, že je ke všem akcím s peněženkou potřeba centralizovaný server, který kontroluje firma ZenGo, je peněženka ZenGo v rozporu se základní myšlenkou blockchainu, a to decentralizací, kde by měl mít uživatel plnou kontrolu nad svými aktivy.

4 Návrh a implementace systému podporujícího více klíčové blockchain peněženky

Cílem implementace systému podporujícího více klíčové blockchain peněženky je zvýšení bezpečnosti a soukromí v oblasti blockchainových transakcí. U běžných transakcí je přítomen jen jeden pár privátního a veřejného klíče. K podepsání transakce je tudíž nutný jen jeden privátní klíč. To znamená, že při kompromitaci privátního klíče útočníkovi nic nebrání v autentizaci transakcí a zneužití všech zdrojů, které jsou vázány na tento privátní klíč. Více klíčová možnost přináší řešení k tomuto problému. Vyšší bezpečnost zde spočívá v rozdělení práv k autentizaci transakcí pomocí účasti většího množství privátních klíčů. Tímto způsobem je možné vytvořit dvou nebo i více faktorovou autentizaci a nebo skupinové vlastnictví peněženky. V prvním případě se jedná o jednoho uživatele kontrolujícího několik zařízení, ve kterých jsou jednotlivě uloženy privátní klíče. K autentizaci je pak třeba účast všech a nebo určitého počtu zařízení obsahujících privátní klíče. V druhém případě jsou v obrazu dva či více uživatelů, kteří společně sdílí blockchainový účet. K autentizaci je pak nutná participace všech nebo určitého počtu uživatelů. U participace všech se tedy jedná o vícenásobné podpisy a u participace podmnožiny celkové množiny se naopak jedná o prahové podpisy. U obou těchto případů je důležitým přínosem zachování anonymity všech členů skupiny, neboť nelze z výsledné transakce určit kdo, ani kolik lidí se transakce zúčastnilo.

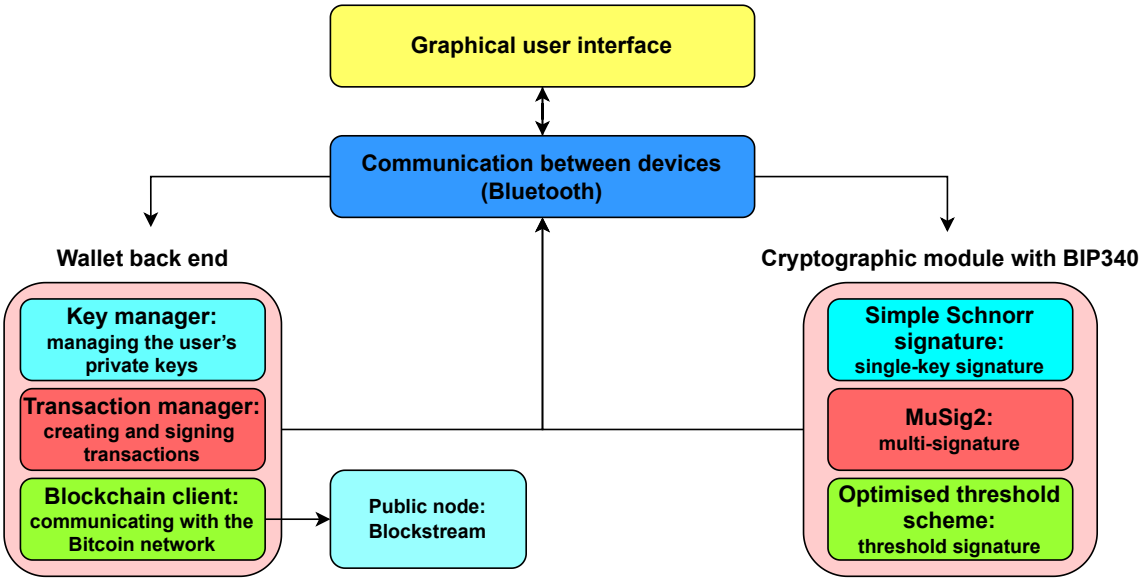
Pro účely demonstrace zabezpečení blockchainových transakcí pomocí více privátních klíčů byla v této práci implementována aplikace blockchainové peněženky běžící na mobilním systému Android. Aplikace je psaná v programu Android Studio v jazyce Kotlin s minimální verzí *API 21: Android 5.1 Lollipop*, která by měla spolehlivě fungovat na 99,2 % všech Android zařízení [33]. Jako cílová verze je zvolena *API 31: Android 12 Snow Cone* kvůli požadavku od firmy Google konstatující minimální cílovou verzi API 31, aby uživatelé novějších verzí Androidu mohli aplikaci najít na *Google Play* a nainstalovat ji [34]. Aplikace podporuje blockchain Bitcoin. Jsou podporované sítě *Testnet* a *Mainnet*. Bitcoin byl vybrán kvůli jeho světové rozšířenosti, dobré dokumentaci a především kvůli podpoře Schnorrova podpisu, bez kterého nelze uskutečnit implementaci více klíčových transakcí.

Aplikace podporuje všechny tři zmíněné úrovně zabezpečení transakcí. Pro základní úroveň je možné vytvářet klasické transakce pomocí jednoho klíče Schnorrova podpisu. Pro pokročilou úroveň zabezpečení je možné vytvářet transakce pomocí kombinace *n-of-n* částečných podpisů, tedy transakce s vícenásobným podpisem. K tomuto se zde používá schéma MuSig2. U nejvyšší úrovně zabezpečení, tedy transakce vytvořené kombinací *t-of-n* částečných podpisů, je použito schéma pro prahové

podpisy od Ricci a spol [35], které bylo upraveno pro funkčnost v Bitcoin blockchainu včetně splnění jeho standardů.

4.1 Architektura systému

Celkový systém se skládá ze čtyř hlavních částí, a to: *back end peněženky*, *kryptografický modul*, *komunikace mezi zařízeními pomocí bluetooth* a *grafické uživatelské rozhraní*. Propojení těchto částí lze vidět na obrázku 4.1.



Obr. 4.1: Architektura systému dle MVVM.

Back end peněženky slouží k řízení všech operací týkajících se dat peněženky a komunikace s Bitcoin blockchainem. Skládá se z následujících komponentů:

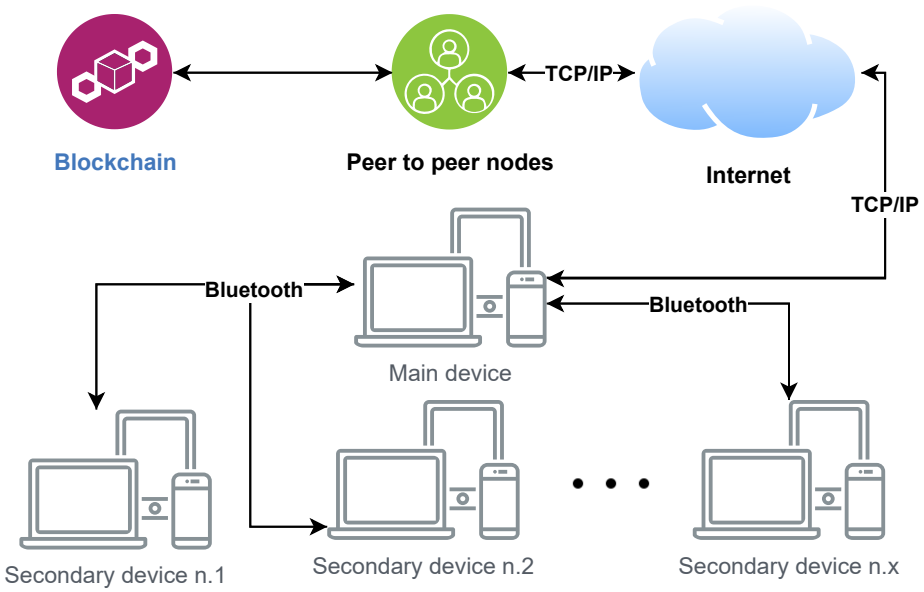
1. **Manažer klíčů** - Stará se o generování klíčů dle Bitcoinových standardů, zajišťuje jejich bezpečné ukládání a vytváří architekturu peněženky.
2. **Manažer transakcí** - Ovládá vytváření transakcí a společně s kryptografickým modulem je podepisuje.
3. **Blockchainový klient** - Jeho účelem je komunikace s Bitcoinovým blockchainem. Získává data o transakcích a také zasílá data do blockchainu. Je zde využit veřejný uzel.

Stěžejní část, tedy kryptografický modul, se skládá ze třech jednotlivých podepisujících algoritmů.

1. **Jednoduchý Schnorrův podpis** - Je používán u základní úrovně zabezpečení využívající jeden podpis.

2. **MuSig2** - Je používán u pokročilé úrovně zabezpečení využívající n-of-n částečných podpisů.
3. **Optimalizovaný prahový podpis** - Je používán u nejvyšší úrovně zabezpečení využívající t-of-n částečných podpisů.

Na obrázku 4.2 jsou vyobrazeny komunikační kanály vyskytující se v systému. Android zařízení spolu komunikují pomocí Bluetooth připojení a při získávání a zasílání dat do Bitcoin blockchainu je využit Internet, tedy TCP/IP.



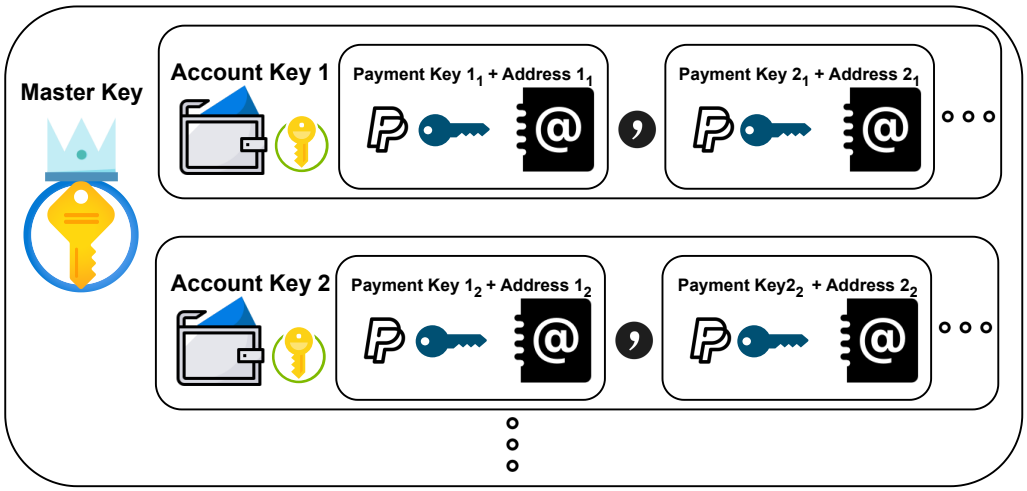
Obr. 4.2: Komunikace mezi zařízeními.

4.2 Architektura peněženky a typ transakcí

Jako peněženku lze brát jeden pár privátního a veřejného klíče, kde veřejný klíč slouží jako adresa peněženky a privátní klíč slouží k dokazování vlastnictví aktiv v peněžence. Chce-li uživatel používat více peněženek pro různé účely, musí vygenerovat další páry klíčů a každý pár jednotlivě ukládat. Tímto způsobem je ale uživatel nucen zaručit bezpečné uložení velkého množství privátních klíčů a při ztrátě jakéhokoliv z nich se tím ztratí i spojená peněženka. Tento problém řeší Bitcoinový standard BIP32, který definuje architekturu hierarchicky deterministické peněženky. Tato architektura používá derivování dětských klíčů (anglicky *Child Key Derivation*), tedy odvozování více potomků klíčů z jednoho rodičovského klíče. Tento princip je použit i v rámci architektury peněženky v této práci. Na obrázku 4.3 je vyobrazena architektura peněženky.

Nejvyšším klíčem v architektuře je *Master Key*, z kterého jsou všechny ostatní klíče odvozeny. K vytvoření tohoto klíče je využit standard BIP39, který definuje ukládání privátních klíčů jako tzv. *mnemonic slova*, což je soubor předem definovaných slov, které při složení určitého počtu v určitém pořadí reprezentuje privátní klíč. Hlavním využitím je uložení privátního klíče takovým způsobem, které je pro lidské oko lehce pochopitelné a tudíž snižuje pravděpodobnost chyby uživatele při ukládání. V této aplikaci je využita bezpečnost 24 slov s možností přidání jednoho jakéhokoliv uživatelem definovaného slova. Toto slovo slouží jako přídavná náhodnost a může sloužit jako heslo k peněžence.

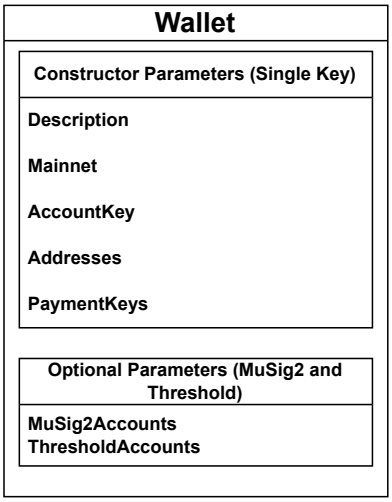
Z Master Key se dále derivuje *Account Key*, který reprezentuje účet, ze kterého se dále derivují nejnižší klíče v architektuře, a to jsou již klasické páry privátního a veřejného klíče, které tvoří Bitcoin peněženku. Díky využití této architektury je možné definovat použití různých účtů či párů klíčů. V případě této aplikace je tedy možné určit, jaký účet nebo pár klíčů bude použit pro jednotlivé úrovně zabezpečení.



Obr. 4.3: Architektura peněženky.

V aplikaci je v peněžence možné vytvořit tři typy účtů. Jimi jsou Single Key, MuSig2 a Threshold účty. Přesněji je možné k Single Key účtu vytvořit MuSig2 a Threshold účty, které jsou na něj vázané. Na obrázku 4.4 lze vidět konstrukci peněženky s povinnými a volitelnými parametry.

Aplikace používá v transakcích klasický způsob utrácení aktiv pomocí dvojice privátního klíče a veřejného klíče (anglicky *key path spending*) oproti způsobu používání skriptu. Využívá se zde nejnovější verze Bitcoinu, kterou je Taproot. Peněženka i transakce jsou tedy vytvářeny na základě Schnorrova podpisu.



Obr. 4.4: Konstrukce peněženky.

4.3 Struktura kódu

Pro účely dodržení zásad vývoje na Android je celkový systém stavěn podle architektury *Model View ViewModel (MVVM)*. Má tudíž tři vrstvy, které spolu navzájem komunikují. První vrstvou je soubor tříd a objektů obsluhující logiku počítání podpisů, vytváření peněženek a transakcí, ukládání dat, komunikace s blockchainem, komunikace se zařízeními a pomocné funkce. Druhá vrstva obsahuje třídy ViewModel, které působí jako mezi vrstva mezi vrstvou grafického uživatelského rozhraní, tedy poslední vrstvou. Účelem druhé vrstvy je oddělení logiky front endu a back endu a udržování stavů jednotlivých aktivit aplikace.

V aplikaci je vytvořeno mnoho tříd, objektů, datových tříd, funkcí a komponentů pro různé účely. Následně jsou uvedeny ty nejpodstatnější.

1. **AndroidBluetoothController** - Třída obsluhující Bluetooth komunikace. Řídí skenování, připojení, odpojení, odesílání a přijímání zpráv.
2. **BlockstreamAPI** - Objekt sloužící ke komunikaci mobilní aplikace s Bitcoin blockchainem pomocí zasílaní požadavků na server třetí strany. Je zde tedy nutné připojení k Internetu.
3. **CryptoLib** - Objekt obsahující všechny kryptografické funkce potřebné k práci s eliptickými křivkami a vícenásobným podpisem. Knihovna je napsána ručně z důvodu minimalizace velikosti aplikace. Importovaná knihovna se všemi funkcemi okolo eliptických křivek by zbytečně zabírala místo v paměti.
4. **MainActivity** - Hlavní aktivita, ze které se volají další aktivity.
5. **MuSig2** - Objekt obsahující funkce pro vytváření vícenásobného podpisu, účtu a peněženky pro vícenásobný podpis. Využívá funkce z objektu *CryptoLib*.

6. **MuSig2Account** - Třída definující parametry účtu vícenásobného podpisu. Dále obsahuje funkce pro výpočet vícenásobného podpisu.
7. **MuSig2Wallet** - Třída definující parametry společné peněženky vícenásobného podpisu.
8. **Paillier** - Objekt sloužící k vytváření klíčů, šifrování a dešifrování u Paillierova kryptosystému.
9. **Tests** - Objekt určený k testování modulů.
10. **TransactionBuilder** - Objekt sloužící k vytváření, podpisu s jedním klíčem a zaslání transakcí.
11. **ThresholdSignature** - Objekt obsahující funkce pro vytváření prahových účtů, peněženek a podpisů. Využívá funkce z objektu *CryptoLib*.
12. **ThresholdAccount** - Třída definující parametry účtu prahového podpisu. Dále obsahuje funkce pro výpočet prahového podpisu.
13. **ThresholdWallet** - Třída definující parametry společné peněženky prahového podpisu.
14. **Utils** - Objekt obsahující pomocné funkce pro práci s daty.
15. **Wallet** - Třída definující parametry obecné peněženky. Součástí této peněženky jsou všechny ostatní peněženky, jsou-li vytvořené.
16. **WalletCreation** - Objekt obsahující funkce pro vytvoření obecné peněženky. Dále obsahuje funkce pro ukládání a čtení peněženek.

V částech aplikace, ve kterých se používají Bitcoinové standardy pro obsluhu vytváření a derivování klíčů, formátování adres, vytváření transakcí a jednoduchý Schnorrův podpis, se využívá externí knihovna *bitcoin-kmp* od firmy ACINQ [36]. Tato knihovna byla vybrána kvůli její dostupnosti a rychlosti. V době vytváření této práce jiná knihovna pro Kotlin podporující Taproot neexistovala. Nativní funkce knihovny jsou všechny psané v jazyce C pro co nejvyšší rychlost.

4.4 Komunikace s blockchainem

První výzva při vývoji Bitcoinových mobilních aplikací je nereálnost hostování svého vlastního uzlu kvůli jeho velikosti jako nejvíce bezpečnou možnost. Existují ale dvě možnosti řešení. První možností je hostovat svůj vlastní uzel na nějakém serveru, ke kterému by byla aplikace připojena. Tímto ale vznikají další povinnosti spojené s během serveru a uzlu samotného. Oproti tomu přináší nejvyšší jistotu, že data získávaná a zasílaná ze serveru jsou opravdu data z uzlu, a tedy blockchainu. Dále je zde jasný přehled o stavu serveru a uzlu. Druhou možností je použít server s uzlem třetí strany. Velkou nevýhodou je nutnost důvěry ve třetí stranu, která při nečestném chování může předávat pozměněná, necelá a nebo i žádná data. Nemůže ale využít uživatelova aktiva, protože aplikace nikdy nezasílá soukromé hodnoty ven

z lokální paměti mobilní aplikace. Na druhou stranu přináší využití třetí strany velké zjednodušení vývoje a snížení množství záležitostí, o které by bylo nutné se při provozu aplikace postarat.

Z důvodu praktičnosti využití třetí strany byl vybrán *Esplora HTTP API* (Application Programming Interface) od organizace *Blockstream* [37]. Tato API umožňuje získávat informace o transakcích, poplatcích, adresách a blocích z Bitcoin mainnetu i testnetu. Dále je možné s touto API zasílat vytvořené transakce přímo do Bitcoin blockchainu. Všechny funkce se obsluhují pomocí požadavků na jejich server. Pro tvorbu HTTP požadavků je v aplikaci využívaná knihovna *Fuel*, díky které se jednoduše vytváří celá struktura požadavků.

Pro potřeby aplikace byly vytvořeny následující funkce:

1. **getUTXO(address: String)** - Tato funkce slouží ke zjištění, jaká UTXO má zadaná adresa k dispozici.
2. **getHexUTXO(txID: String)** - Funkce má za účel vrátit hexadecimální vyjádření k zadanému identifikačnímu číslu transakce.
3. **getEstematedFees(level: String)** - Funkce slouží ke zjištění aktuální výše poplatků dle uvedené rychlosti provedení transakce.
4. **postTx(tx: String)** - Tato funkce zasílá zadanou transakci na blockchain. Při úspěchu vrací ID transakce.

4.5 Účet s jedním klíčem

Na počátku vzniku peněženky se automaticky vytváří Single Key účet, tedy klasický účet s jedním klíčem. Způsob vytváření tohoto účtu kopíruje Bitcoinové standardy pro případnou interoperabilitu mezi jinými aplikacemi. Proces vytváření prvotní nebo Single Key peněženky je řízen funkcí *createWalletAccount()* z objektu *WalletCreation*. Kroky vytváření Single Key účtu jsou následující:

1. **Generování Master Key** - V prvním kroku procesu vytváření peněženky se generuje *Master Key*. Z tohoto klíče se dále derivují všechny ostatní klíče. Proces začíná získáním entropie, ze které se pak generuje *Master Key*. Entropii zde představují náhodně vygenerovaná data, která jsou pro průměrného člověka nezapamatovatelná. Z tohoto důvodu je v aplikaci dle dokumentu BIP 39 použit Mnemonic kód, který entropii převádí na člověku srozumitelná slova, která si uživatel při vytváření peněženky zapíše pro případ ztráty peněženky nebo importu peněženky do jiné aplikace. Jako jazyk seznamu slov je použita angličtina. Počet slov je nastaven na maximální, a to 24, s možností přidání vlastního hesla, které přidává další entropii a tudíž i bezpečnost k náhodně vygenerované hodnotě. V případě použití vlastního hesla toto heslo musí uživatel

mít, stejně jako 24 slov, k dispozici při importování a nebo při obnově peněženky. Po dokončení generování slov je nutné vyčistit paměť s daty Mnemonic kódu, aby nebyla citlivá data dostupná útočníkovi. Tento proces je znázorněn na výpisu 4.1, kde je znázorněná funkce *mnemonic()*.

```
1  fun mnemonic(password: String): ByteArray {
2      val count = Mnemonics.WordCount.COUNT_24
3      val mnemonicCode = Mnemonics.MnemonicCode(count)
4      val passphrase = password.toCharArray()
5      val seed: ByteArray = mnemonicCode.toSeed(passphrase)
6      val listOfWords = mnemonicCode.words
7      printMnemonic()
8      mnemonicCode.clear()
9      return seed
10     }
```

Výpis 4.1: Funkce mnemonic().

Nakonec se pomocí funkce *generate()* z objektu *DeterministicWallet* z knihovny bitcoin-kmp vygeneruje pomocí funkce *HMAC* finální naformátovaný *Master Key*.

2. **Derivace Account Key** - Dalším krokem je derivace *Account Key* z *Master Key*. Teoreticky je tento krok i další následující kroky možné přeskočit a použít *Master Key* jako privátní klíč pro podepisování transakcí. Takovým způsobem ale uživatel přijde o možnost využít výhod hierarchisticky deterministických peněženek. Nebude mít tudíž možnosti vytvářet stromovou konstrukci klíčových párů a tím ani větší počet účtů či adres pro účely soukromí, bezpečnosti, sdílení s ostatními uživateli apod. Z tohoto důvodu je tato i další derivace implementována dle dokumentu BIP 32. O derivaci se stará funkce *derivePrivateKey()* z objektu *DeterministicWallet*. Tuto funkci i cestu pro CKD (*Child Key Derivation*) *Account Key* lze vidět na výpisu 4.2.

```
1  val accountKey = DeterministicWallet.derivePrivateKey(masterKey
       , KeyPath("m/86'/"+ if (mainnet) "0'" else "1"+"/0'"))
```

Výpis 4.2: CKD Account Key.

Jak lze zpozorovat z cesty pro CKD *Account Key*, index je definován vždy na číslo 0. Aplikace tedy podporuje vytvoření jen jednoho účtu z jednoho *Master Key*. Tento způsob byl vybrán pro zajištění vyšší bezpečnosti na úkor uživatelského pohodlí. Kdyby měl jeden uživatel nebo více uživatelů větší množství účtů svázané k jednomu *Master Key* a ten by byl kompromitován, tak jsou potom kompromitovány i všechny účty pod ním. Dále při sdíleném používání více účtů pod jedním *Master Key* není možná obnova účtu bez důvěry v třetí stranu.

3. **Generování dvojic adres a Payment Keys** - Pomocí *Payment Key* se podepisují transakce. Je to tedy klasický privátní klíč, tudíž se také derivuje pomocí CKD. *Payment Key* se v aplikaci derivuje z *Account Key*. Při prvotním vzniku peněženky je vytvořený jen jeden *Payment Key* s indexem 0. To znamená i jedna adresa k tomuto klíči. Dále při užívání aplikace je ale možné přidávat další páry klíče a adresy. Na výpisu 4.3 je vyobrazen CKD *Payment Key*.

```
1 val paymentKey = DeterministicWallet.derivePrivateKey(
    accountKey, listOf(0L, index))
```

Výpis 4.3: CKD Payment Key.

Adresu je možné získat klasickým způsobem pomocí výpočtu obyčejného veřejného klíče k *Payment Key* a následně formátování dle dokumentu BIP 350. Doporučeno je ale použít upravený veřejný klíč dle dokumentu BIP 341. Aplikace tedy používá upravený veřejný klíč pro získání adresy. Ve výpisu 4.4 lze vidět postup generování adresy z upraveného veřejného klíče.

```
1 val internalKey = XonlyPublicKey(paymentKey.publicKey)
2 val outputKey = internalKey.outputKey(Crypto.SchnorrTweak.
    NoScriptTweak).first
3 val address = Bech32.encodeWitnessAddress(if (mainnet) "bc"
    else "tb", 1, outputKey.value.toByteArray())
```

Výpis 4.4: Generování adresy z upraveného veřejného klíče.

4. **Uložení peněženky** - Posledním krokem je uložení vytvořené peněženky. Peněženka je strukturovaná jako objekt, tudíž k ukládání peněženky byla vybrána knihovna Gson od firmy Google [38], která jednoduše serializuje a deserializuje Java objekty na JSON (*JavaScript Object Notation*) objekty a zpátky. Funkčnost ukládání spravuje funkce *saveWalletToFile()*, která má parametry *context: Context, wallet: Wallet a filename: String*. Data jsou ukládána v módu *MODE_PRIVATE*, tudíž žádná aplikace kromě této aplikace nemá přístup k souboru, kde jsou ukládaná data peněženek.

4.5.1 Tvorba a podepisování transakcí

Hlavní funkcí pro řízení tvorby transakcí je funkce *buildPay2TRTx()* z objektu *TransanctionBuilder*. Jejími parametry jsou *myaddress: String, ammount: Long, address: String, feesLevel: String* a vrací vytvořenou transakci a její hash. Je tedy nezbytné zadat adresu, ze které se zasílá, jaké množství aktiv má být zasláno, na jakou adresu mají být aktiva zaslána a jakou rychlostí. Před tím, než by chtěl uživatel vytvářet a zasílat transakce, musí prvně zjistit, jestli nějaké transakční výstupy

vlastní. K tomu slouží funkce *getUTXO()* z objektu *BlockstreamAPI*, která vrací ID transakcí. Dále je nutné z vybraných UTXOs získat hexadecimální vyjádření pro možnost čtení transakce. Tato hodnota se získává pomocí funkce *getHexUTXO()* z objektu *BlockstreamAPI* s parametrem ID transakce, která byla získaná z funkce *getUTXO()*. Dále je nutné kvůli možnosti většího počtu výstupů transakce z transakce vyčíst, jaké výstupy uživatel vlastní. Toto se provede pomocí porovnání scriptu veřejného klíče uživatele oproti scriptu obsaženého ve výstupu. Script veřejného klíče se vytváří pomocí funkce *addressToPublicKeyScript()* z objektu *Bitcoin* z knihovny *bitcoin-kmp*. Poslední hodnotou, kterou je nezbytné získat, je výše poplatku v danou dobu. Ta se získává pomocí funkce *getEstematedFees()* z objektu *BlockstreamAPI*. Z důvodu, že jsou všechny funkce s HTTP požadavkem asynchronní a některé na sebe navazují, se zde používají *Kotlin Coroutines* pro řízení životního cyklu asynchronních funkcí.

Po získání všech potřebných hodnot nastává samotná stavba transakce. O stavbu transakce se stará třída *Transaction* z knihovny bitcoin-kpm. V jejím konstruktoru je nutné specifikovat verzi transakce, která určuje jaká pravidla transakce splňuje. V aplikaci je použita verze 2 z důvodu případného využití relativního času uzamknutí (anglicky *relative lock-time*) transakcí. Dále je nutné specifikovat vstupy a výstupy. Vstupy jsou vybrány z listu UTXOs dle množství zasílaných aktiv. List výstupů může obsahovat výstup pro příjemce s hodnotou zasílaných aktiv a výstup zbytků zaslaný zpátky odesílateli, a nebo může obsahovat jen jeden výstup pro příjemce. Ve výstupu je také nutné přidat script příjemce pro verifikaci vlastnictví pomocí scriptu. Script se v aplikaci vytváří pomocí funkce *addressToPublicKeyScript()* ze třídy *Bitcoin* z knihovny bitcoin-kmp, kde vstupem funkce je adresa příjemce. Poplatky jsou dané dle rozdílu hodnot vstupů a hodnot výstupů. V tomto momentě je transakce úspěšně vytvořená, ale k úspěšnému zaslání do blockchainu jí chybí ještě podpis. Pro podpis je nutné vytvořit z transakce hash, to se provádí pomocí funkce *hashForSigningSchnorr()* z třídy *Transaction*. Ve výpisu 4.5 je ukázána část tvorby nepodepsané transakce ve funkci *buildPay2TRTx()*.

```
1 val tx = Transaction(
2     2,
3     listOf(TxIn(OutPoint(txToSpend, myOutputs[0]), TxIn.
    SEQUENCE_FINAL)),
4     listOf(TxOut((ammount).sat(), outputScript)),
5     0
6 )
```

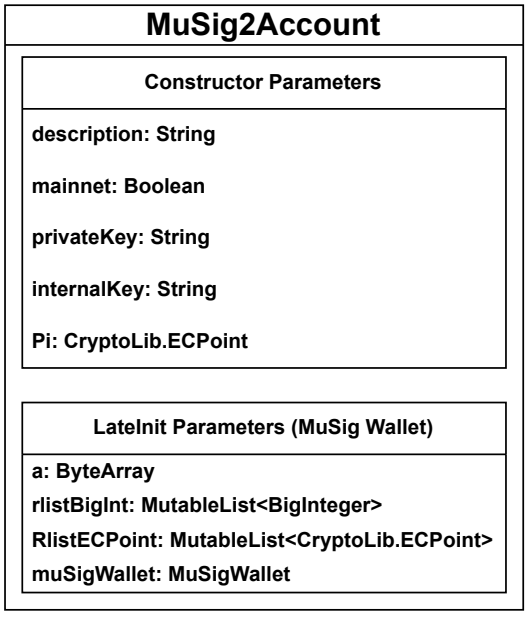
Výpis 4.5: Tvorba transakce.

Podepisování transakcí s jedním klíčem obsluhuje funkce *singTx()* z objektu *TransactionBuilder*. Jejími parametry jsou *data: ByteVector32, tx: Transaction, pay-*

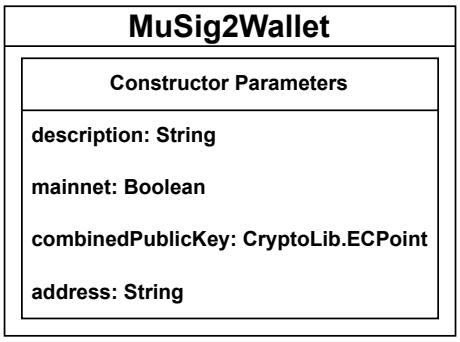
mentKey: String, Tweak: Crypto.SchnorrTweak a vrací podepsanou transakci. Hlavní je zde funkce *signSchnorr()* ze třídy *Crypto* z knihovny *bitcoin-kmp*, která provádí Schnorrův podpis. Při podpisu se předává hash transakce, platební klíč a specifikace o úpravě klíčů. Zde u jednoduchého podpisu se vždy používá Taproot úprava klíčů dle dokumentu BIP 341. Nakonec se transakce musí doplnit podpisem. To se provádí pomocí funkce *updateWitness()* ze třídy *Transaction*. Pro zaslání podepsané transakce na blockchain stačí již jen použít funkci *postTx()* s parametrem transakce v hexadecimálním formátu.

4.6 MuSig2 účet

Předpokladem pro vytvoření MuSig2 účtu je existence Single key účtu s alespoň jedním volným párem platebního klíče a adresy. Při vytváření uživatel vybírá, který pár se použije pro vznik MuSig2 účtu, tedy jakým párem bude vytvářet společnou více podpisovou peněženku a podepisovat více podpisové transakce. Tento pár je možno využít i jako obyčejný účet pro podpis jedním klíčem. MuSig2 účet definuje datová třída *MuSig2Account*. Pro vytvoření účtu se používá funkce *createMuSig2Account()* z objektu *MuSig2*. Na obrázku 4.5 je vizualizována struktura účtu MuSig2.



Obr. 4.5: Konstrukce MuSig2 účtu.



Obr. 4.6: Konstrukce MuSig2 peněženky.

Inicializace společné peněženky se provádí v aktivitě *MuSig2WalletCreationActivity* využívající ViewModel *MuSig2WalletCreationViewModel*. ViewModel zachytává příchozí zprávy a předává je do aktivity. Uživatel, který inicializuje tvoření peněženky, posílá inicializační zprávu obsahující jeho veřejný klíč ostatním uživatelům, se

kterými si chce vytvořit společnou peněženku. Každý z uživatelů odpovídá na tuto zprávu svým veřejným klíčem. V moment, kdy mají všichni všechny veřejné klíče, je volána funkce *muSig2WalletGen()*, která vytvoří společnou peněženku. Přesné výpočty, které se zde dějí, jsou popsány v teoretické části v sekci MuSig2. Na obrázku 4.6 lze vidět parametry MuSig2 peněženky.

Transakce se u MuSig2 účtu vytváří úplně stejně jako u Single Key účtu, tedy skrze třídu *Transaction* z knihovny bitcoin-kpm. Podpis se zde vytváří v aktivitě *MuSig2SpendingActivity* využívající ViewModel *MuSig2SpendingViewModel*. Uživatel, který inicializuje podepisování, zasílá ostatním uživatelům inicializační zprávu obsahující vytvořenou transakci pomocí funkce *spendingInit()*. Uživatelé poté pomocí funkce *muSig2Sign()* vytváří své částečné podpisy, které jsou zaslány jako odpověď k inicializační zprávě. Inicializátor nakonec agreguje všechny částečné podpisy pomocí funkce *finalSign()* a tím vzniká finální podepsaná transakce, která je dále odeslaná do blockchainu pomocí funkce *sendTx()*. Přesné výpočty, které se zde dějí, jsou popsány v teoretické části v sekci MuSig2.

4.7 Účet prahového podpisu

Další úrovní bezpečnosti po MuSig2 účtu, tedy účtu vícenásobného podpisu, je účet prahového podpisu.

4.7.1 Základní schéma prahového podpisu a jeho modifikace

Jádrem účtu prahového podpisu je schéma prahového podpisu od Ricci a spol [35]. Toto schéma používá 3 kryptografická primitiva: Schnorrův podpis, Shamirův protokol pro sdílení tajemství a Paillierův kryptosystém. Schéma je dále složeno ze tří fází: nastavení, podpis a verifikace. Podepisující entity lze rozdělit jako hlavní(MD) a sekundární(SD) podepisující. Hlavní podepisující iniciuje proces podepisování a má na konci procesu celkový podpis. Sekundární podepisující odpovídá na požadavky od hlavního podepisujícího a má jen svůj vypočítaný částečný podpis. V Algoritmu 7 je vyobrazena fáze nastavení. V první fázi nastavení si každé zařízení vygeneruje svoje veřejné a soukromé hodnoty. V druhé fázi se počítají hodnoty Shamirova protokolu sdílení tajemství. V této fázi spolu zařízení komunikují a přenášejí mezi sebou data. Zabezpečení přenášených dat a schopnost počítat se zašifrovanými hodnotami umožňuje Paillierův kryptosystém a jeho homomorfní vlastnosti.

Na obrázku 4.7 jsou vyobrazeny etapy nastavení a podepisování základního prahového schématu. Před samostatnou etapou podepisování je nutné rozhodnout, kdo se účastní podepisování. Etapa podepisování má dvě části. V první části každý podepisující, který byl rozhodnut k účasti podepisování, vypočítá svůj relační klíč

Algoritmus 7 `Nastavení`

Fáze 1 - generování veřejných a soukromých hodnot

1: vygenerovat náhodně $d_1^{(j)}, \ldots, d_{t-1}^{(j)}$

2: vygenerovat Paillierův klíčový pár $(pk_{p,j}, sk_{p,j})$

3: vygenerovat náhodně k_j in $\mathbb{Z}_{q_{EC}}$

4: vypočítat $pk_j = g^{k_j}$

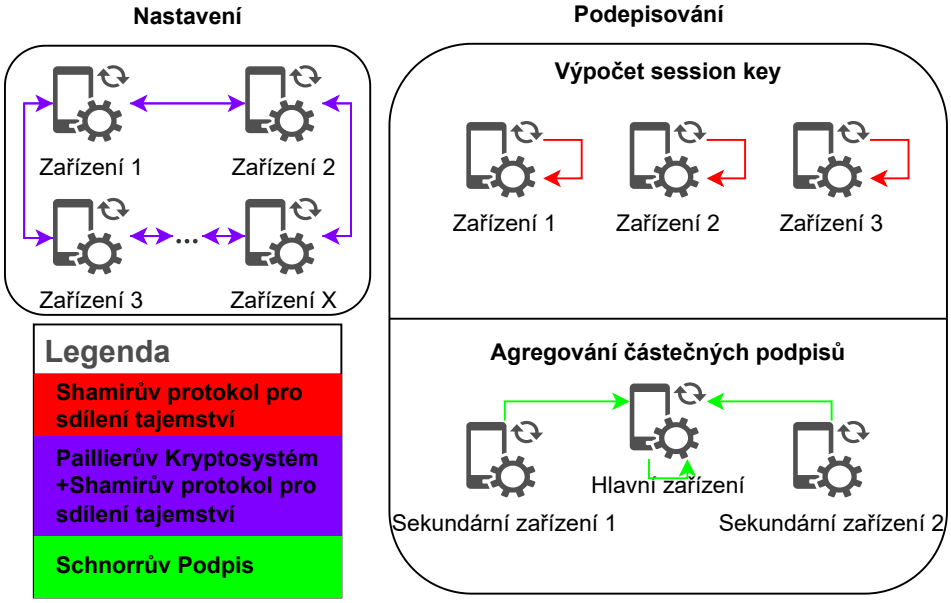
Fáze 2 - výpočet hodnot Shamirova protokolu sdílení tajemství

1: D_h vygenerovat náhodnou hodnotu $r_{j,h}$ a vypočítat $e_{j,h} = Enc_{pk_{p,j}}(\alpha_j, r_h)$

2: D_h vygenerovat náhodnou hodnotu $v_{j,h}$ a vypočítat $c_h = e_{j,h}^{\alpha_j^{t-2} * d_{t-1}^h} * e_{j,h}^{\alpha_j^{t-3} * d_{t-2}^h} * \cdots * e_{j,h}^{d_1^h} * Enc_{pk_{p,j}}(k_j, v_{j,h})$

3: je-li $h = j$, pak D_j vypočítá $f(\alpha_j) = Dec_{sk_{p,j}}(c_j - 1) + d_{t-1}^j * \alpha_j^{t-1} + \cdots + d_1^j * \alpha_j + k_j$

(anglicky *session key*). Výpočet využívá vlastnosti Shamirova protokolu pro sdílení tajemství, přesněji tedy interpolaci. V druhé a poslední části podepisování každý podepisující vypočítá závazek, který zasílá hlavnímu podepisujícímu, který všechny závazky sečte a agregované závazky pošle zpátky všem podepisujícím. Každý podepisující má v tuto chvíli všechny potřebné hodnoty k výpočtu svého částečného podpisu. K výpočtu je použit Schnorrův podpis. Podepisující zasílají své částečné podpisy hlavnímu podepisujícímu, který je spolu se svým agreguje a tím získává finální podpis.



Obr. 4.7: Fáze základního schématu prahového podpisu.

Finální etapa verifikace je prováděna Bitcoin blockchainem. Verifikace dle Bitcoinového standardu je popsána v algoritmu 8.

Algoritmus 8 `Verifikace`

1: výpočet $e = taggedHash(\text{"}BIP0340/challenge\text{"}, r_{sig} + pk + msg)\%n$
2: výpočet $R = G * s_{sig} + pk * (n - e)$
3: **if** $(R == r)$ **return true** else **return false**

Základní schéma prahového podpisu, na kterém je tato práce postavena, bylo prvotně vytvořeno jako samostatný protokol a nebral v úvahu specifické požadavky a vlastnosti blockchainu Bitcoin. Kvůli tomu bylo nutné modifikovat schéma tak, aby bylo optimalizované dle Bitcoinových standardů. Přesněji bylo nutné modifikovat část schématu, kde se vytvářejí částečné Schnorrovy podpisy. V prvotním schématu se závazek c počítá jako g^r, kde r je náhodné číslo. V Bitcoinu není hodnota r zcela náhodná. Prvně je vypočítaná hodnota t jako *xor* tajného klíče podepisujícího a tzv. *tagged hashe* náhodného čísla. Hodnota c je pak vypočítána jako tagged hash zřetězení hodnot t, veřejného klíče podepisujícího a zprávy. Výzva e se také počítá rozdílně a to jako tagged hash zřetězení hodnot c, veřejného klíče podepisujícího a zprávy. Důkaz s je počítán stejně u Bitcoinu i u základního schématu. Finální podpis je dvojice c a s. V algoritmu 9 je vyobrazena modifikovaná fáze podepisování. Části zvýrazněné červeně jsou části, které byly modifikovány.

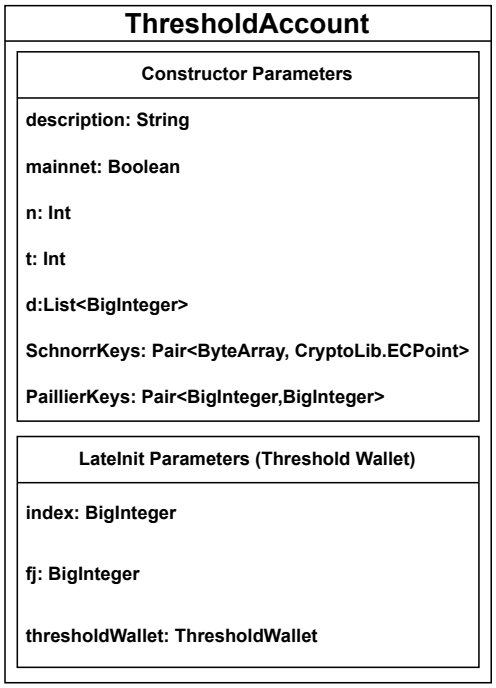
Algoritmus 9 Podepisování - upravené

1: **for** $j \in \mathcal{J}_t$ **do**: ▷ run privately by each D_j (i.e., MD and SDs)
2: $t_j = k_j$ xor $taggedHash(\text{"}BIP0340/aux\text{"}, aux)$
3: $c_j = taggedHash(t_j||pk_j||m) \bmod q_{EC}$ ▷ SDs send c_j to MD
4: **end for**
5: $c = \prod_{j \in \mathcal{J}_t} c_j$ ▷ run by MD, c and m sent to SDs
6: **for** $j \in \mathcal{J}_t$ **do**: ▷ run privately by each D_j (i.e., MD and SDs)
7: $e = taggedHash(\text{"}BIP0340/challenge\text{"}, c||pk||m)$
8: $z_j = r_j - e * s_j \bmod q_{EC}$ ▷ SDs send z_j to MD
9: **end for**
10: $z \leftarrow \sum_{j \in \mathcal{J}_t} z_j \bmod q_{EC}$ ▷ run by MD
11: **return** $\sigma = (c, z)$

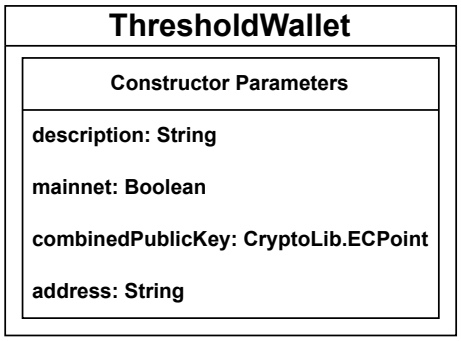
4.7.2 Tvorba společné peněženky prahového podpisu a podepisování transakcí

Stejně jako u MuSig2 je prvně nutné vytvořit účet. Toho se docílí použitím funkce *createThresholdAccount()* z objektu *ThresholdSignature*. Tato funkce vytváří všechna

data potřebná k vytvoření společné peněženky, tedy provedení fáze nastavení ve schématu prahového podpisu. Jsou tudíž vygenerovány páry klíčů Schnorrova podpisu, Paillierova kryptosystému a $d_1^{(j)}, \ldots, d_{t-1}^{(j)}$. Dále jsou připraveny kontejnery dat pro hodnoty následně vygenerované peněženky. Data threshold účtu drží datová třída *ThresholdAccount*, jak je vidět na obrázku 4.8.



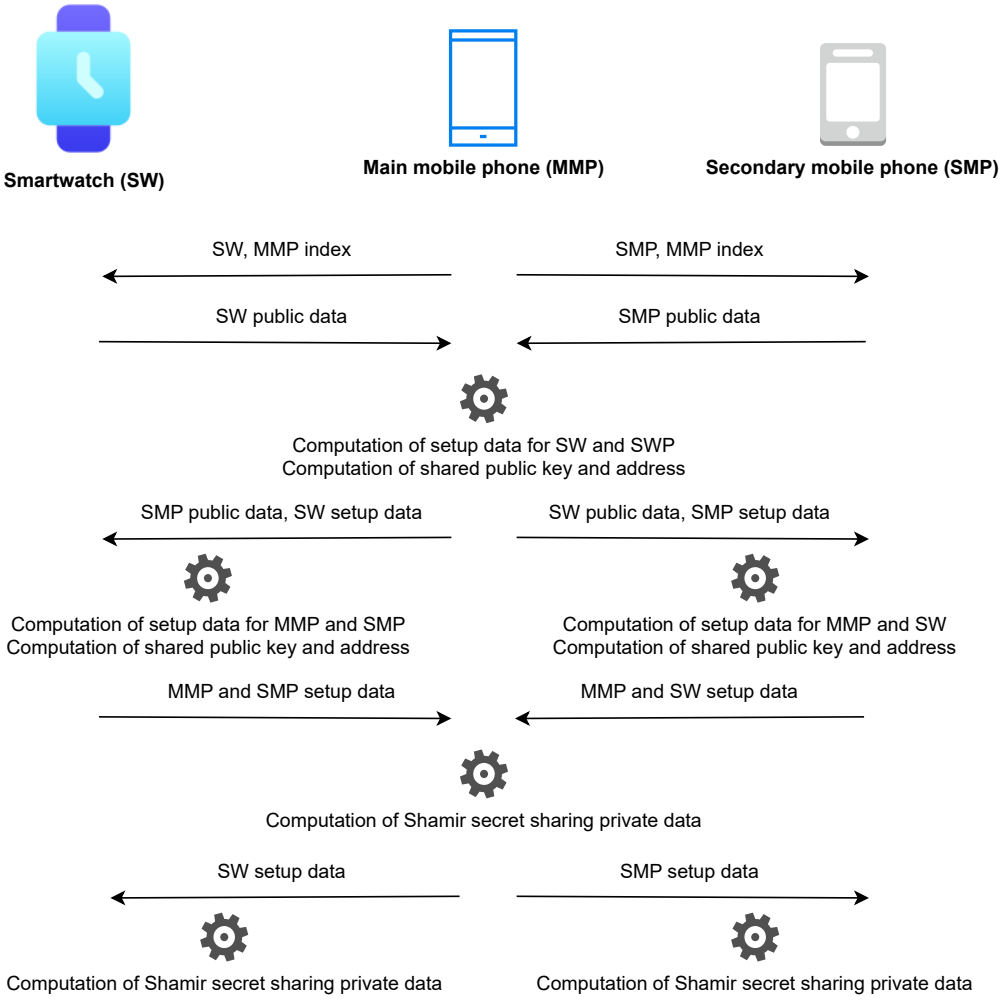
Obr. 4.8: Konstrukce Threshold účtu.



Obr. 4.9: Konstrukce Threshold peněženky.

Generování společné peněženky probíhá v aktivitě *ThresholdWalletCreationActivity* společně s ViewModelem *ThresholdWalletCreationViewModel*. ViewModel používá dvě různé třídy pro Bluetooth komunikaci. Pro komunikaci s chytrými hodinkami využívá *WearOSDataTransferController* a pro komunikaci s android mobilními telefony využívá *AndroidBluetoothController* společně s *BluetoothViewModel*. Generování společné peněženky začíná v moment, kdy jsou MD a všechny SD společně připojeni. MD zasílá všem SD inicializační zprávu obsahující svůj a jejich indexy. SD jako odpověď zasílají svá veřejná data, tedy Schnorrův a Paillierův veřejný klíč. MD má nyní dostatek informací na spočítání společného veřejného klíče a adresy. Dále počítá nastavovací data pro vypočítání privátních dat Shamirova sdíleného tajemství. MD přeposílá nastavovací data a veřejná data SD příslušným SD. SD vykonají stejné výpočty jako MD v předchozím kroku a zasílají MD nastavovací data. MD nastavovací data přeposílá příslušným SD. V tento moment mají všichni účastníci data k vypočítání privátních dat Shamirova sdíleného tajemství (fj). Data se uloží

a vytvoří si společnou peněženku. Výsledná peněženka je popsána na obrázku]4.9. Komunikace mezi zařízeními je vyobrazena na obrázku 4.10.



Obr. 4.10: Komunikace mezi zařízeními ve fázi nastavení peněženky.

Jak je z komunikace vidno, data jsou přeposílána z centrálního prvku, což není originální způsob řešení a přináší potenciální bezpečnostní hrozby, byl by MD nečestný. V případě této práce je ale předpoklad, že jsou aplikace navzájem navázané, nelze provést reverzní inženýrství na již zkompilovanou aplikaci a nebo je aplikace použita v bezpečném prostředí.

Vytváření transakcí a podepisování je obdobné jako u MuSig2 účtu. MD vytváří transakci pomocí třídy *Transaction* a zasílá na podepsání SD v podobě inicializační zprávy. Využívá se zde aktivity *ThrersholdApproveSpendingBluetoothActivity* společně s ViewModelem *ApproveSpendingBluetoothViewModel*. SD vypočítá svůj závazek c_j pomocí funkce *approveSpending()* a zasílá ho MD. MD vypočítá výzvu e pomocí funkce *calculateE()* a zasílá jej SD. SD nyní vypočítá svůj částečný podpis

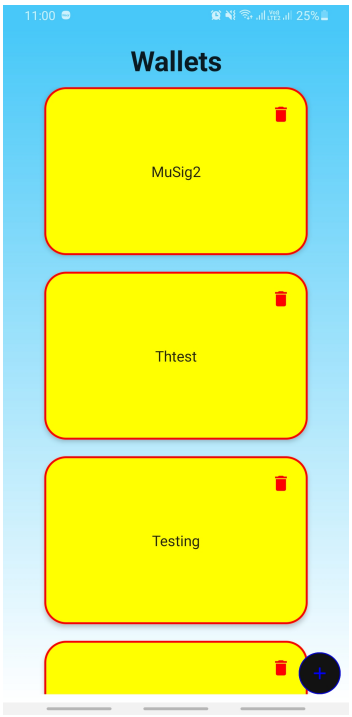
pomocí funkce *calculatePartialSig()* a zasílá jej MD. MD agreguje svůj i SD částečné podpisy do jednoho finálního pomocí funkce *calculateSig()*, aktualizuje transakci s finálním podpisem pomocí funkce *updateTx()* a může jej poté zaslat do blockchainu pomocí funkce *postTx()*. Detailní výpočty podpisu jsou popsány výše v algoritmu 9.

4.8 Grafické uživatelské rozhraní

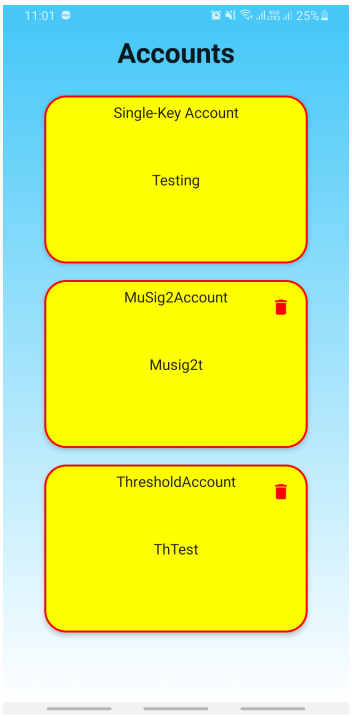
Pro vytváření grafického uživatelského rozhraní je v mobilní aplikaci použito kombinace staré a moderní architektury. U chytrých hodinek je použita stará architektura. Starší architekturu reprezentují *.xml* soubory v kombinaci s aktivitou. Moderní architekturu představuje *Jetpack Compose* v kombinaci s aktivitou a ViewModelem. U *.xml* architektury jsou vytvářeny *.xml* soubory, které představují obrazovky, které jsou promítány uživateli. Tyto soubory jsou volány v aktivitách a zde se přistupuje k jejich komponentům při nutnosti změny. Naopak u architektury *Jetpack Compose* jsou vytvářeny tzv. *composable* komponenty, které vytvářejí promítanou obrazovku. Tyto komponenty jsou volány taktéž v aktivity, ale změny dat jsou prováděny pomocí stavů, které jim přináší ViewModel.

Z uživatelského hlediska byly vytvořeny grafické komponenty pro hlavní obrazovku obsahující všechny peněženky, obrazovku zobrazující tři typy účtů, které lze v aplikaci vytvořit, obrazovku pro jednotlivé typy účtů, obrazovku pro vytváření MuSig2 a účtů prahového podpisu, obrazovku pro připojení Bluetooth, obrazovku pro odesílání transakcí a obrazovku pro schválení transakcí.

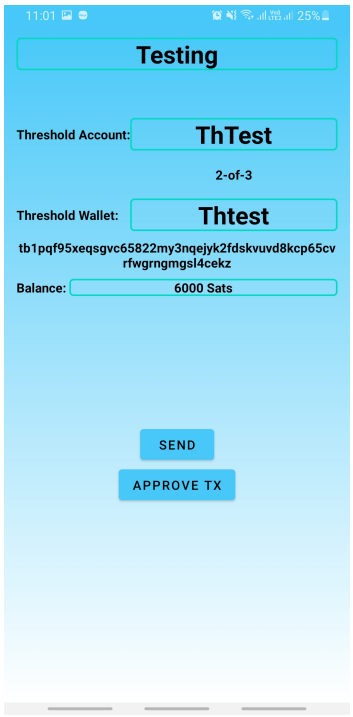
Hlavní aktivitou, tedy aktivitou, která se spustí při otevření aplikace, je *MainActivity*, která obsahuje seznam všech peněženek, viz obrázek 4.11. Uživatel si v této aktivitě může vytvářet peněženky a nebo odstranit existující. Při výběru některé z peněženek v seznamu se uživatel dostává do aktivity *AllAccountsActivity*, kde jsou uvedeny všechny tři typy účtů, viz obrázek 4.12. Zde má uživatel automaticky vytvořený klasický single key účet. Zbylé dva účty si musí uživatel nastavit. Na obrázku 4.13 je vyobrazen příklad plně nastaveného účtu prahového podpisu. Při odesílání aktiv z účtů se objeví obrazovka pro odesílání transakcí příslušného účtu. Na obrázku 4.15 je vyobrazena obrazovka pro odeslání transakce na potvrzení u účtu prahového podpisu z mobilní aplikace. Při nutnosti připojení přes Bluetooth se zobrazí obrazovka pro připojení Bluetooth, viz obrázek 4.14. U chytrých hodinek jsou obrazovky minimalistické kvůli omezenému prostoru. Na obrázcích 4.16 a 4.17 jsou vyobrazené obrazovky pro vytváření peněženky a potvrzení transakce na chytrých hodinkách.



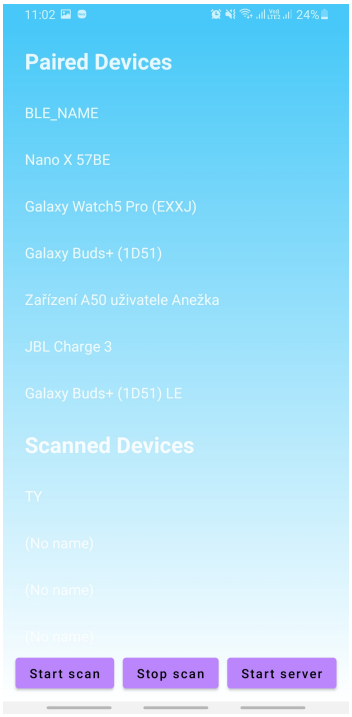
Obr. 4.11: Obrazovka s peněženkami.



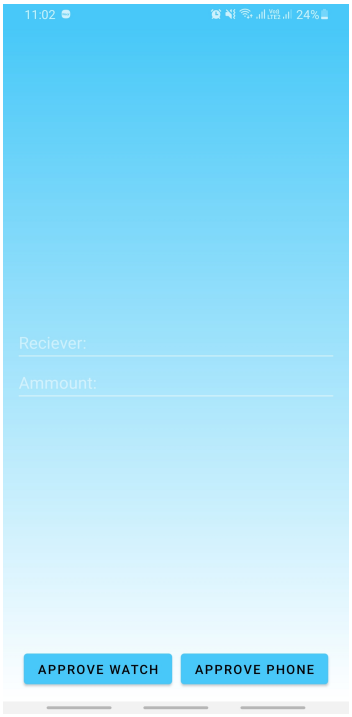
Obr. 4.12: Obrazovka s účty.



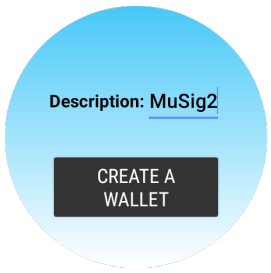
Obr. 4.13: Obrazovka účtu prahového podpisu.



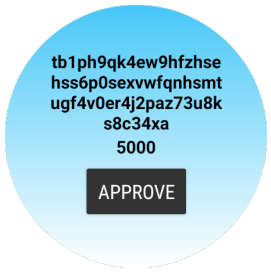
Obr. 4.14: Obrazovka pro připojení Bluetooth.



Obr. 4.15: Obrazovka pro odeslání transakce.



Obr. 4.16: Obrazovka pro vytváření peněženky.

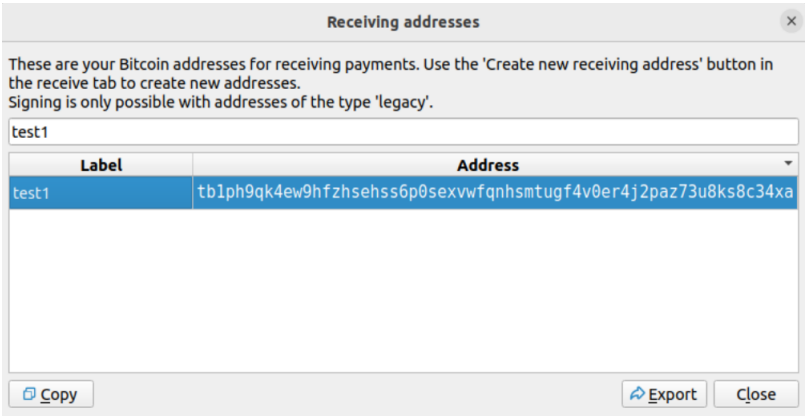


Obr. 4.17: Obrazovka pro potvrzení transakce.

5 Testování aplikace

Pro testování transakcí byla vybrána síť *Testnet*, jelikož je to síť určená pro testovací účely v blockchainu Bitcoin. Používají se zde testovací Bitcoiny, které nemají žádnou reálnou peněžní cenu, tudíž by při neúspěchu testování nebyla ztracena žádná reálná aktiva. Testovací tokeny byly získány na Bitcoin *faucet* stránkách, které jsou určené k zasílání testovacích Bitcoinů pro komunitu. V práci byla využita stránka `https://bitcoinfaucet.uo1.net/`. Aplikace je psaná tak, aby podporovala *Testnet* i *Mainnet*, kde jediný rozdíl je CKD cesta a předpona u adresy, tudíž při úspěšném testování transakcí na testovací síti se předpokládá úspěšné testování i v druhé síti.

Peněženky vytvořené aplikací slouží jako role odesílatele. Pro roli příjemce byl vytvořen reálný uzel v Bitcoinové síti včetně bitcoin-QT, což je grafické rozhraní pro Bitcoinový uzel, kde byla vytvořena Taproot peněženka s jednou adresou v síti Testnet. Uzel s peněženkou běží na notebooku Lenovo Legion 7. Důvodem pro zprovoznění oficiální Bitcoinové peněženky běžící na Bitcoinovém uzlu je, aby byla vytvořena jistota ze strany příjemce a dále peněženka slouží pro následnou kontrolu, jestli transakce z aplikace skutečně přišly a nebo ne. Na obrázku 5.1 je vyobrazen screenshot z aplikace bitcoin-QT, kde lze vidět přijímací adresu.

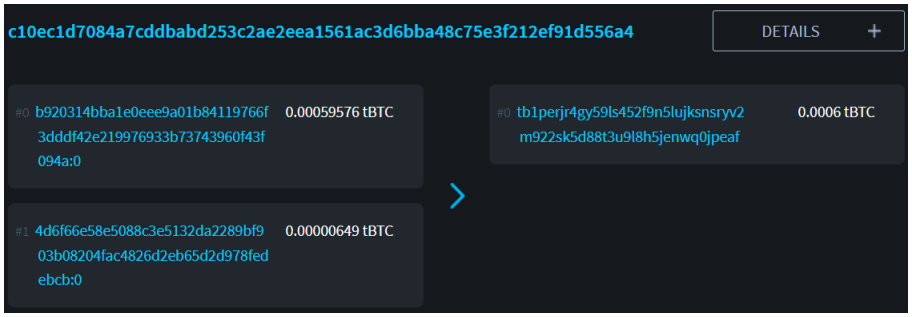


Obr. 5.1: Přijímací adresa v bitcoin-QT.

5.1 Transakce s jedním podpisem

U transakce s jedním podpisem byl použit mobilní telefon Samsung Galaxy S9+ s verzí androidu 10. Test byl proveden vytvořením peněženky s výběrem sítě Testnet. V seznamu účtů je automaticky vytvořen standardní účet včetně jeho adresy, tedy účet s jedním podpisem. Adresa byla dále pomocí *faucet* stránky nabita testovacími

Bitcoiny. Na obrázku 5.2 je vyobrazena transakce z Bitcoin *faucetu* na vygenerovanou adresu ze stránky BlockStream. Po provedení této transakce je bilance testovací peněženky 1 UTXO s 0.0006 testovacími Bitcoiny.



Obr. 5.2: Screenshot transakce z faucetu.

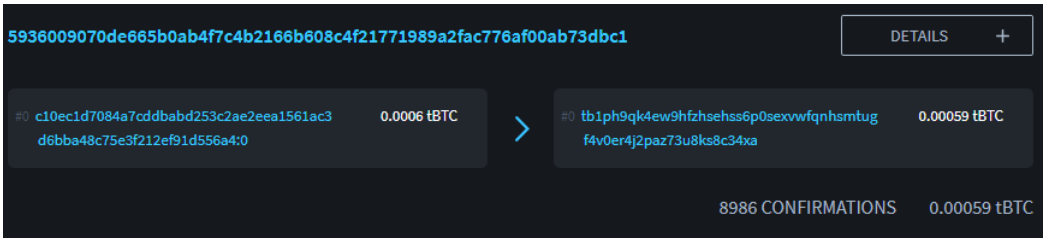
Jako test byla vytvořena transakce s 0.00059 testovacími Bitcoiny. A po podepsání transakce funkcí *singTx()* vrátila funkce hodnotu hexadecimální formy vytvořené podepsané transakce. Tuto hodnotu lze přeložit do člověkem čitelné podoby. Na překlad byla použita stránka Blockcypher. Ve výpisu 5.1 je vyobrazena přeložená transakce ze stránky Blockcypher.

Výpis 5.1: Překlad hexadecimální formy transakce.

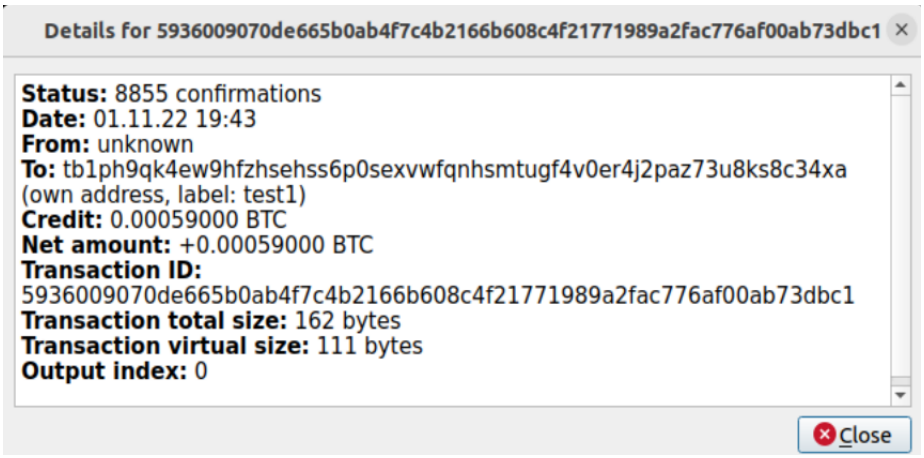
```
1  Hex value: 02000000000101a456d591ef12f2e3758ca4bbd6c31a56
     a1eee22a3c25bdbadd7c4a08d7c10ec10000000000ffffffff0178
     e60000000000000225120b9416ae5c5ba457866f08682f864cc7241
     3bc36be213563f23ac941e8bd1e1ed014060de73c2198032e299a6
     9f73634506cedabee03a0291b8fbc2810d2fb2c299eaf45a7a9f18
     617ee90c8b3407a95869c9741fe7d18ba8b5380546bfe42501fae3
     00000000
2  "addresses": [
3          "tb1perjr4gy59ls452f9n5lujksnsryv2m922sk5d88t3u9l
             8h5jenwq0jpeaf",
4          "tb1ph9qk4ew9hfzhsehss6p0sexvwfqnhsmtugf4v0er4j2
             paz73u8ks8c34xa"
5      ],
6      "block_height": -1,
7      "block_index": -1,
8      "confirmations": 0,
9      "double_spend": false,
10     "fees": 1000,
```

```
    "hash": "5936009070de665b0ab4f7c4b2166b608c4f21771989
      a2fac776af00ab73dbc1",
    "inputs": [
        {
            "addresses": [
                "tb1perjr4gy59ls452f9n5lujksnsryv2m922sk5
                  d88t3u9l8h5jenwq0jpeaf"
            ],
            "age": 2378490,
            "output_index": 0,
            "output_value": 60000,
            "prev_hash": "c10ec1d7084a7cddbabd253c2ae2eea
              1561ac3d6bba48c75e3f212ef91d556a4",
            "script_type": "pay-to-taproot",
            "sequence": 4294967295
        }
    ],
    "outputs": [
        {
            "addresses": [
                "tb1ph9qk4ew9hfzhsehss6p0sexvwfqnhsmtugf4
                  v0er4j2paz73u8ks8c34xa"
            ],
            "script": "5120b9416ae5c5ba457866f08682f864cc
              72413bc36be213563f23ac941e8bd1e1ed",
            "script_type": "pay-to-taproot",
            "value": 59000
        }
    ],
    "preference": "low",
    "received": "2023-05-04T13:11:52.51147099Z",
    "relayed_by": "44.202.158.188",
    "size": 162,
    "total": 59000,
    "ver": 2,
    "vin_sz": 1,
    "vout_sz": 1,
    "vsize": 111
```

Jak lze ve výpisu vidět, figurují zde dvě adresy, a to odesílající a přijímající, které odpovídají vytvořeným adresám z aplikace a z bitcoin-QT. Dále transakce obsahuje svůj hash, který odpovídá identifikačnímu číslu transakce, je-li transakce úspěšně zapsána v blockchainu. Do vstupu transakce je vložena transakce obdržená z *faucetu* a do výstupu je vložena adresa příjemce, script adresy a odesílaná hodnota. Výstup je jen jeden, to znamená, že celá transakce bude utracena a všechny zbytky transakce, tedy rozdíl mezi hodnotou vstupu a výstupu, poslouží jako transakční poplatek. V tomto případě je odesíláno 59000 Satoshi a k dispozici je 60000 Satoshi, poplatek je tudíž 1000 Satoshi, jak lze také vidět v transakci. Dále je vidět, že se používá transakce typu *pay-to-taproot*, což indikuje zasílání na Taproot adresu. Transakce byla dále odeslána na blockchain. A jak lze vidět ze screenshotů 5.3 ze stránky BlockStream a z bitcoin-QT 5.4, tak vskutku transakce prošla a v bitcoin-QT peněžence byla transakce přidána.



Obr. 5.3: Screenshot testovací transakce.

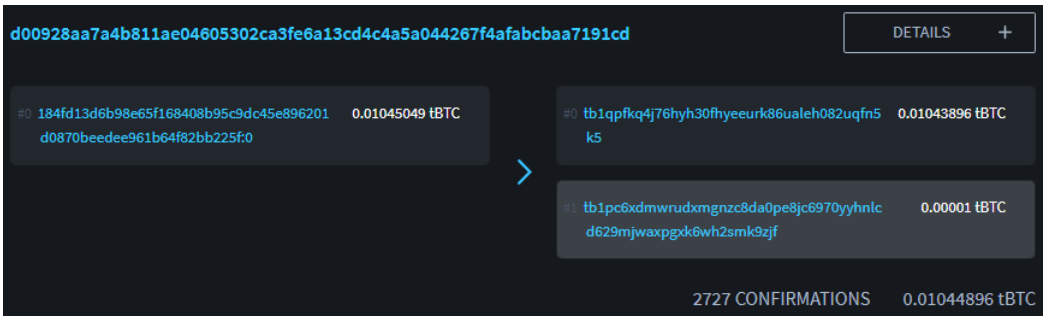


Obr. 5.4: Screenshot testovací transakce z bitcoin-QT.

5.2 Transakce s MuSig2 podpisy

Testování transakcí s MuSig2 podpisy proběhlo na dvou zařízeních, a to na mobilním telefonu Samsung Galaxy S9+ s verzí androidu 10 a chytrých hodinkách Samsung Galaxy Watch5 Pro. Testování bylo rozděleno do dvou částí. První z nich bylo testování samotného schématu MuSig2. Tento test je proveden funkcí *testMuSig2BIP340()* v objektu *Tests*. Při testu bylo vytvořeno 10 obyčejných účtů obsahujících MuSig2 účty, které byly dále agregovány do jedné MuSig2 peněženky. Dále byla náhodná data podepsána funkcí *musig2Sign()* a podpis byl ověřen pomocí funkce *verifySignatureSchnorr()* z objektu *Crypto* z knihovny bitcoin-kmp. Podpis byl úspěšně ověřen, tudíž je zde předpoklad k úspěchu v další druhé fázi, kde je již testováno na blockchainu.

V druhé části byla vytvořena MuSig2 peněženka ze dvou MuSig2 účtů. Případem použití je zde dvou faktovoré ověření při vlastnictví obou zařízení jedním vlastníkem a nebo jako sdílená peněženka v případě dvou uživatelů vlastnících každý jedno zařízení. Stejně jako v testu s jedním klíčem byla adresa společné peněženky nabita testovacími Bitcoiny. Na screenshotu 5.5 je zobrazena transakce z *faucetu* na MuSig2 peněženku.



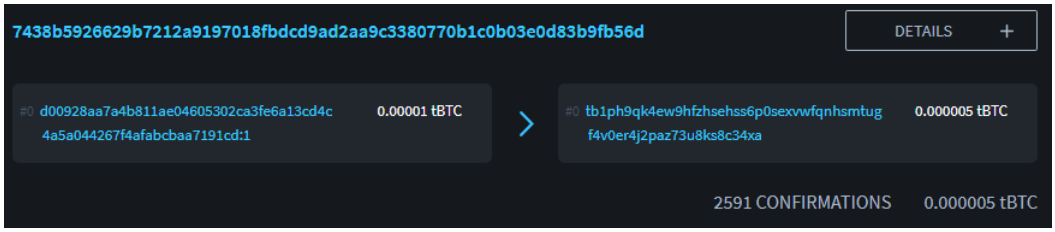
Obr. 5.5: Screenshot transakce z faucetu na MuSig2 peněženku.

MuSig2 peněženka má k dispozici transakci s 0.00001 testovacími Bitcoiny, neboli 1000 testovacími Satoshi, jak lze vyčíst ze screenshotu. Dále byla vytvořena transakce pro odeslání 0.000005 testovacích Bitcoinů na bitcoin-QT peněženku. Zbytek, tedy 0.000005 testovacích Bitcoinů je dáno jako poplatek za transakci. Tudíž stejně jako u transakce s jedním podpisem byla využita celá neutracená transakce. Podpis transakce byl vytvořen pomocí aktivity *MuSig2SpendingActivity* společně s *MuSig2SpendingViewModel*, jak je popsáno v sekci 4.6. Vytvořená transakce v hexadecimální formě, kterou lze přeložit stejně jako v případě transakcí s jedním podpisem, je vyobrazena ve výpisu 5.2.

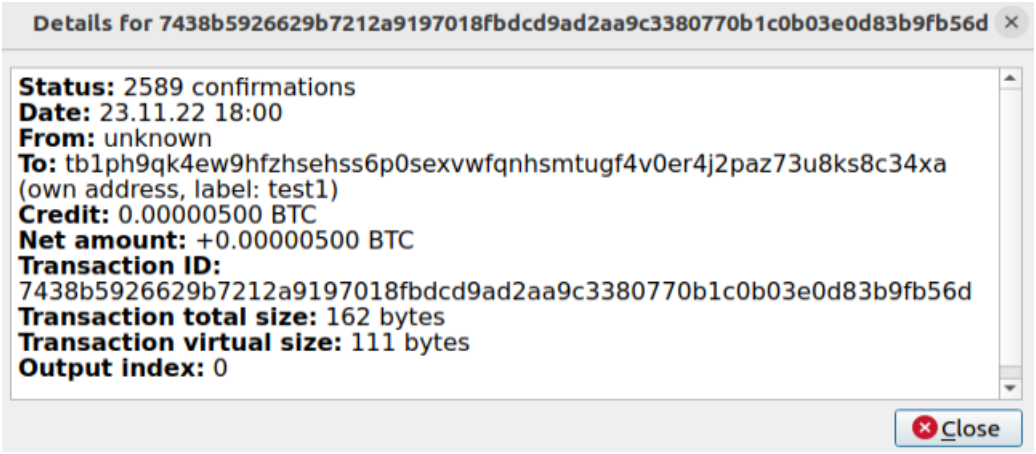
Výpis 5.2: Překlad hexadecimální formy transakce MuSig2 účtu.

```
Hex value: 02000000000101cd9171aacbabaff46742045a4a4ccd13
   6afea32c300546e01a814b7aaa2809d00100000000ffffffff01f4
   0100000000000225120b9416ae5c5ba457866f08682f864cc7241
   3bc36be213563f23ac941e8bd1e1ed0140c5c2a37106a70fd06600
   f171cc3dbe913b869d93bcb850c60d99049d21af998291bad7c029
   9e20ab4366df32361e19d0ffa0dd6d75ac418398af2429cc7f9f13
   00000000
```

Jak lze zpozorovat z transakcí ve výpisech 5.1 a 5.2, transakce jsou formátově úplně stejné a nelze z nich vyčíst, jsou-li transakce podepsané více uživateli a nebo jedním. Po následném odeslání transakce na blockchain byla transakce úspěšně připsána na adresu bitcoin-QT peněženky, jak je vyobrazeno na obrázcích 5.6 a 5.7.



Obr. 5.6: Screenshot MuSig2 testovací transakce.

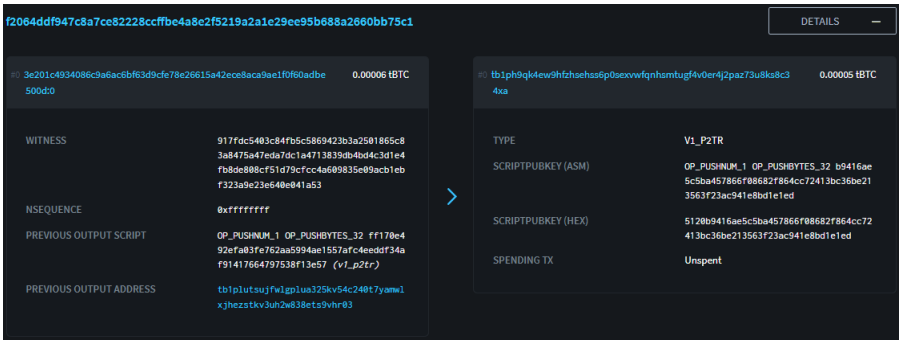


Obr. 5.7: Screenshot MuSig2 testovací transakce z bitcoin-QT.

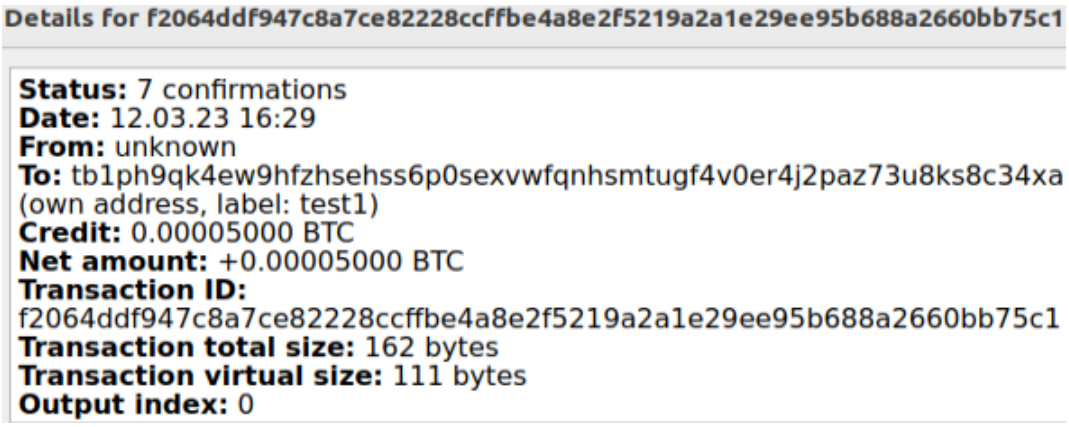
5.3 Transakce s prahovým podpisem

Pro testovací scénář s transakcemi s prahovým podpisem byly zvoleny tři zařízení a to dva mobilní telefony (Samsung Galaxy S9+ a Samsung A50) a jedny chytré hodinky (Samsung Galaxy Watch5 Pro). Práh je nastaven na dva ze tří. Roli MD má mobilní zařízení, které je připojeno k chytrým hodinkám. Zbytek zařízení je tedy SD. Jak již bylo zmíněno v sekci 4.7, MD se v této aplikaci chová jako server, který přeposílá data ostatním SD. Tato varianta byla zvolena kvůli nemožnosti navázání spojení hodinek s více než jedním zařízením bez toho, aby bylo nutné hodinky resetovat. Kvůli bezpečnostním pochybnostem je využití tohoto scénáře doporučeno v bezpečném prostředí, kde si uživatelé věří. Příklad užití by mohla být rodičovská kontrola, kde MD vlastní dítě a dvě SD vlastní rodiče. Dítě může utrácet z peněženky jen při souhlasu alespoň jednoho rodiče. Další možností může být MD jako jeden rodič a SD jako dvě děti. Dítě tedy musí mít souhlas rodiče pro utrácení z peněženky. Společně by si děti mohly podepsat transakci, ale není zde existující komunikační kanál. Jedním z dalších případů užití může být vylepšená verze dvou faktorového ověření jako u MuSig2 ale s rozdílem, že uživatel může pomocí MD a jakéhokoliv SD podepsat transakci. Nemusí mít tedy u sebe přesně danou sestavu zařízení. Kdyby z nějakého důvodu ztratil privátní klíče jednoho z SD, tak má ještě druhý a může si aktiva přesunout do nové peněženky.

Stejně jako u testování MuSig2 účtu byla prvně otestována funkcionalita lokálně pomocí funkce *testThresholdBIP340()* z objektu *Tests*. Po úspěšném lokálním testu byla vytvořena společná peněženka ze zmíněných tří zařízení. Peněženka byla nabita testovacími Bitcoiny a byla vytvořena transakce zasílaná na bitcoin-QT peněženku. Podpisu se zúčastnilo zařízení Samsung Galaxy S9+ jako MD a Samsung Galaxy Watch5 Pro jako SD. Na obrázku 5.8 je vyobrazen screenshot zaslané transakce a na obrázku 5.9 je vyobrazen screenshot přijaté transakce v bitcoin-QT peněžence. Podepsaní transakce pomocí prahového podpisu tedy proběhlo úspěšně.



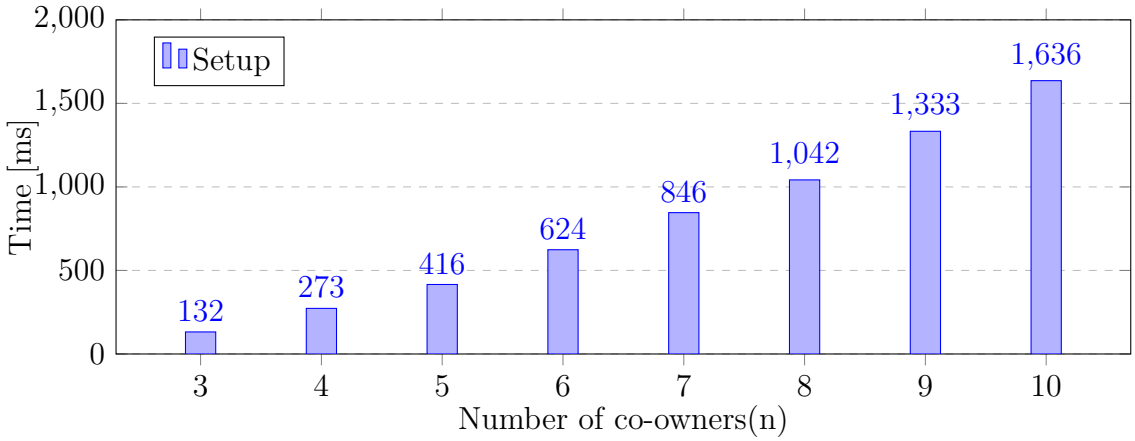
Obr. 5.8: Screenshot testovací transakce z prahové peněženky.



Obr. 5.9: Screenshot testovací transakce z prahové peněženky z bitcoin-QT.

5.3.1 Rychlostní testy

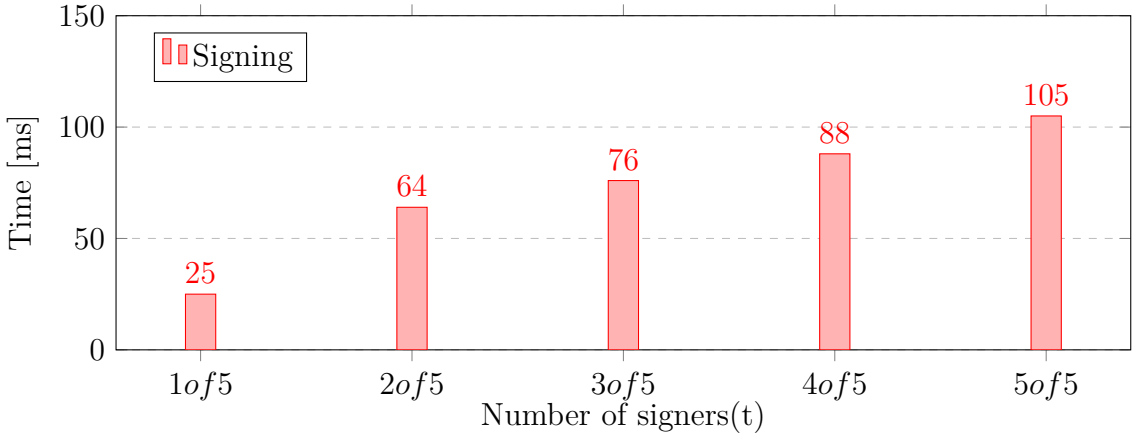
Součástí testování bylo také měření rychlosti implementovaného schématu prahového podpisu. Jak lze vidět z obrázku 5.10, fáze nastavení peněženky rapidně roste s rostoucím množstvím společných uživatelů. Hodnoty jsou u vyšších počtů uživatelů poměrně vysoké, ale tato fáze probíhá jen jednou při vytváření peněženky. Tudíž reálné použití u vyššího počtu uživatelů je proveditelné.



Obr. 5.10: Rychlostní test fáze nastavení peněženky.

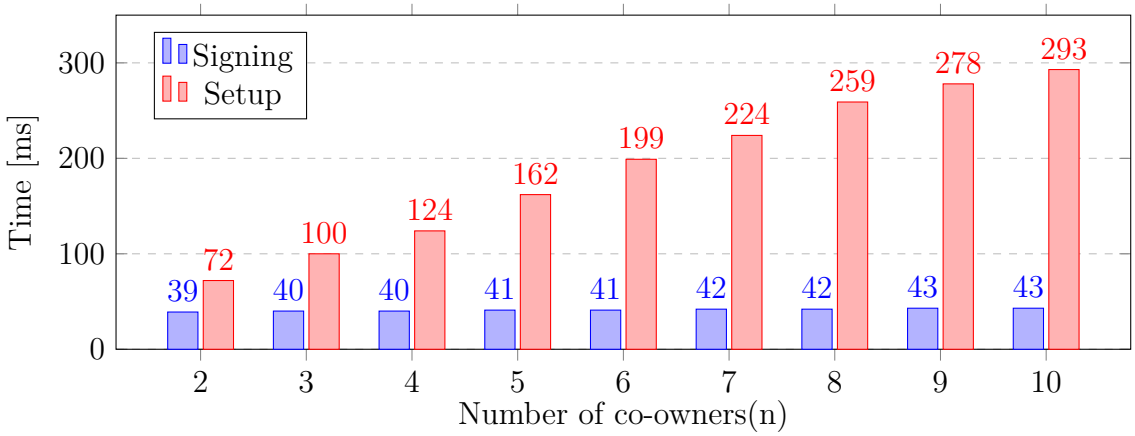
V obrázku 5.11 je vyobrazena rychlost podepisování u účtu s pěti uživateli dle nastaveného prahu. Časy jsou zde přijatelné u všech počtů podepisujících. I při přidání zpoždění kvůli komunikaci mezi zařízeními zde není problém pro reálný provoz. Zpoždění v rámci Bluetooth komunikace v testovacím prostředí této práce

bylo kolem jedné sekundy. Kdyby spolu zařízení komunikovala pomocí TCP/IP, bylo by komunikační zpoždění zanedbatelné.



Obr. 5.11: Rychlostní test fáze podepisování u účtu s pěti uživateli.

Pro porovnání rychlosti schémat MuSig2 a prahového podpisu byla také změřena rychlost nastavení peněženky a podepisování transakcí u MuSig2. V grafu 5.12 lze vidět tyto hodnoty pro dva až deset uživatelů. Z grafu je patrné, že při fázi nastavení peněženky se rychlost lineárně zvedá a je mnohem nižší než u schématu prahového podpisu. Podepisování je téměř konstantní a velmi rychlé i přes to, že je část podepisování psaná v jazyce Kotlin. Z toho vyplývá, že v případě použití *n-of-n* částečných podpisů je vhodnější použít účet MuSig2.



Obr. 5.12: Rychlostní test MuSig2.

Závěr

V této práci byla navržena a implementována Bitcoinová peněženka běžící na systémech Android, peněženka podporuje tři úrovně bezpečnosti. Peněženka tedy umožňuje vytvářet tři typy účtů, a to účty podporující klasické transakce s jedním podpisem (*Single Key*), které jsou vhodné pro běžného uživatele. Dále účty podporující pokročilejší transakce s vícenásobným podpisem (*MuSig2*) založené na protokolu MuSig2, které jsou vhodné pro pokročilejší uživatele. A nakonec účty prahového podpisu jako nejvyšší možnost zabezpečení v této aplikaci, které jsou vhodné pro nejpokročilejší uživatele či firmy hledající nejvyšší možné zabezpečení. Aplikace podporuje chytré hodinky, které se chovají jako hardware peněženka, a tím přidávají dodatečnou bezpečnost. Aplikaci lze využít pro více faktorové ověření, jako společnou peněženku více uživatelů a nebo jako rodičovskou kontrolu.

I přes to, že se při konstrukci vícenásobných podpisů a prahových podpisů v aplikaci využívá několika částečných podpisů, tak má finální podpis stejnou velikost jako jednoduchý Schnorrův podpis. Podpisy jsou od sebe nerozeznatelné. Tato vlastnost tedy řeší problém soukromí a škálovatelnosti. Při testování funkčností aplikace byly ověřovány všechny tři typy účtů. Testování bylo provedeno v síti *Testnet*. Všechny tři účty úspěšně vytvořily a podepsaly transakce, které byly dále akceptovány blockchainem a aktiva byla připsána příjemci transakce.

Díky využívání Bitcoinových standardů je tato aplikace kompatibilní ve všemi ostatními aplikacemi, které taktéž využívají dané standardy. Tudíž lze peněženky mezi těmito aplikacemi migrovat. Avšak *MuSig2* účty a účty prahového podpisu nejsou zatím nijak standardizované, a proto se jejich tvorba nedá automaticky převést do jiných peněženek.

Podle autorových znalostí se jedná o první aplikaci peněženky s vícenásobnými podpisy a prahovými podpisy na trhu v době psaní této práce, která používá čistou kryptografii, nikoli chytré kontrakty a nemá problémy se škálovatelností a soukromím uživatelů.

Aplikaci Bitcoinové peněženky je vhodné rozšířit o vylepšení schématu prahového podpisu pomocí implementace BIP341 *(Taproot SegWit)* pro snížení velikosti transakce a tím docílit menších poplatků. Dále je vhodné implementovat lepší grafické uživatelské rozhraní, lepší manipulaci s transakcemi a nebo přidat podporu dalších blockchainů.

Literatura

[1] Veronika Lang and Razvan Cristian. Annual inflation up to 10.6% in the euro area. [online], 11 2022. [cit. 2022-12-11]. URL: https://ec.europa.eu/eurostat/documents/2995521/15265521/2-17112022-AP-EN.pdf/b6953137-786e-ed9c-5ee2-6812c0f8f07f.

[2] IBM. Supply chain intelligence suite: Food trust. [online], 2022. [cit. 2022-12-11]. URL: https://www.ibm.com/products/supply-chain-intelligence-suite/food-trust.

[3] Jennifer Korn. Record 3.8 billion stolen in crypto hacks last year, report says. [online], 1 2023. [cit. 2023-05-01]. URL: https://edition.cnn.com/2023/02/01/tech/crypto-hacks-2022/index.html.

[4] Oliver Hinz Michael Nofer, Peter Gomber and Dirk Schiereck. Blockchain – a disruptive technology. *Business & Information Systems Engineering*, 59:183–187, 2017. URL: https://link.springer.com/article/10.1007/s12599-017-0467-3, doi:https://doi.org/10.1007/s12599-017-0467-3.

[5] I. Bashir. *Mastering Blockchain*. Packt Publishing, 2017. URL: https://books.google.cz/books?id=dMJbMQAACAAJ.

[6] lastbitcoder. Advantages and disadvantages of blockchain. [online], 2022. [cit. 2022-12-11]. URL: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-blockchain/.

[7] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. [online], 10 2008. [cit. 2022-12-1]. URL: https://bitcoin.org/bitcoin.pdf.

[8] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 357–388, Cham, 2017. Springer International Publishing.

[9] Kirsty Moreland. The blockchain generations. [online], 8 2022. [cit. 2022-12-1]. URL: https://www.ledger.com/academy/blockchain/web-3-the-three-blockchain-generations.

[10] Decentralized Dog. Four advantages and disadvantages of bitcoin. [online], 2022. [cit. 2022-12-15]. URL: https://coinmarketcap.com/alexandria/article/four-advantages-and-disadvantages-of-bitcoin.

[11] Pieter Wuille, Jonas Nick, and Tim Ruffing. Schnorr signatures for secp256k1. [online], 1 2020. [cit. 2022-12-10]. URL: https://github.com/bitcoin/bips/blob/a099f43ce2013e5dc17e9ad4aa92b5c53fd69da7/bip-0340.mediawiki.

[12] Pieter Wuille, Jonas Nick, and Tim Ruffing. Taproot: Segwit version 1 spending rules. [online], 1 2020. [cit. 2022-12-10]. URL: https://github.com/bitcoin/bips/blob/a099f43ce2013e5dc17e9ad4aa92b5c53fd69da7/bip-0341.mediawiki.

[13] Pieter Wuille, Jonas Nick, and Tim Ruffing. Validation of taproot scripts. [online], 1 2020. [cit. 2022-12-10]. URL: https://github.com/bitcoin/bips/blob/a099f43ce2013e5dc17e9ad4aa92b5c53fd69da7/bip-0342.mediawiki.

[14] Luke Dashjr. Bip process, revised. [online], 2 2016. [cit. 2022-12-6]. URL: https://github.com/bitcoin/bips/blob/master/bip-0002.mediawiki.

[15] Chuan Zhao, Shengnan Zhao, Minghao Zhao, Zhenxiang Chen, Chong-Zhi Gao, Hongwei Li, and Yu an Tan. Secure multi-party computation: Theory, practice and applications. *Information Sciences*, 476:357–372, 2019. URL: https://www.sciencedirect.com/science/article/pii/S0020025518308338, doi:https://doi.org/10.1016/j.ins.2018.10.024.

[16] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.

[17] C. P. Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 239–252, New York, NY, 1990. Springer New York.

[18] Nils Fleischhacker, Tibor Jager, and Dominique Schröder. On tight security proofs for schnorr signatures. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 512–531, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[19] William J Buchanan. Using the schnorr signature method to aggregate signers (secp256k1). [online], 2022. [cit. 2022-11-14]. URL: https://asecuritysite.com/schnorr/schnorr_test.

[20] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, nov 1979. doi:10.1145/359168.359176.

[21] Ed Dawson and Diane Donovan. The breadth of shamir's secret-sharing scheme. *Computers Security*, 13(1):69–78, 1994. URL: https://www.

sciencedirect.com/science/article/pii/0167404894900973, doi:https://doi.org/10.1016/0167-4048(94)90097-3.

[22] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology—EUROCRYPT'99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2–6, 1999 Proceedings 18*, pages 223–238. Springer, 1999.

[23] Meenakshi Kansal and Ratna Dutta. Round optimal secure multisignature schemes from lattice with public key aggregation and signature compression. In Abderrahmane Nitaj and Amr Youssef, editors, *Progress in Cryptology - AFRICACRYPT 2020*, pages 281–300, Cham, 2020. Springer International Publishing.

[24] Anon. Multi-signature. [online], 7 2021. [cit. 2022-12-8]. URL: https://en.bitcoin.it/wiki/Multi-signature.

[25] Jordan Millar, Piotr Napierala, Duncan Coutts, and Olga Hryniuk. Simple scripts. [online], 9 2022. [cit. 2022-12-10]. URL: https://github.com/input-output-hk/cardano-node/blob/master/doc/reference/simple-scripts.md.

[26] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. Cryptology ePrint Archive, Paper 2018/068, 2018. URL: https://eprint.iacr.org/2018/068.

[27] Jonas Nick, Tim Ruffing, and Yannick Seurin. Musig2: Simple two-round schnorr multi-signatures. Cryptology ePrint Archive, Paper 2020/1261, 2020. https://eprint.iacr.org/2020/1261. URL: https://eprint.iacr.org/2020/1261.

[28] Gerrit Bleumer. *Threshold Signature*, pages 611–614. Springer US, Boston, MA, 2005. doi:10.1007/0-387-23483-7_429.

[29] Lukas Schor. What is gnosis safe? [online], 2022. [cit. 2022-12-10]. URL: https://help.gnosis-safe.io/en/articles/3876456-what-is-gnosis-safe.

[30] Thomas Voegtlin. Multisig wallets. [online], 2017. [cit. 2022-12-10]. URL: https://electrum.readthedocs.io/en/latest/multisig.html.

[31] ZenGo. What is mpc? [online], 2022. [cit. 2022-12-10]. URL: https://zengo.com/mpc-wallet/.

[32] Elichai Turkel, Omer Shlomtz, and Matan Hamilis. Zengo x. [online], 2022. [cit. 2022-12-10]. URL: https://github.com/ZenGo-X.

[33] Eugene Belinski. Android api levels. [online], 11 2022. [cit. 2022-12-6]. URL: https://apilevels.com/.

[34] Google. Target api level requirements for google play apps. [online], 11 2022. [cit. 2022-12-6]. URL: https://support.google.com/googleplay/android-developer/answer/11926878?hl=en.

[35] Sara Ricci, Petr Dzurenda, Raúl Casanova-Marqués, and Petr Cika. Threshold signature for privacy-preserving blockchain. In Andrea Marrella, Raimundas Matulevičius, Renata Gabryelczyk, Bernhard Axmann, Vesna Bosilj Vukšić, Walid Gaaloul, Mojca Indihar Štemberger, Andrea Kő, and Qinghua Lu, editors, *Business Process Management: Blockchain, Robotic Process Automation, and Central and Eastern Europe Forum*, pages 100–115, Cham, 2022. Springer International Publishing.

[36] Fabrice Drouin, Bastien Teinturier, Salomon Brys, Richard Myers, and Romain Boisselle. Kotlin multiplatform bitcoin library. [online], 10 2022. [cit. 2022-12-8]. URL: https://github.com/ACINQ/bitcoin-kmp/.

[37] Nadav Ivgi, Matthew Haywood, Dimitris Tsapakidis, Benthe Carman, and Hyunhum Cho. Esplora http api. [online], 11 2022. [cit. 2022-12-8]. URL: https://github.com/Blockstream/esplora/blob/master/API.md.

[38] Google. A java serialization/deserialization library to convert java objects into json and back. [online], 11 2022. [cit. 2022-12-9]. URL: https://github.com/google/gson/.

Seznam symbolů a zkratek

BFT	Byzantova chybová tolerance – Byzantine Fault Tolerance
PoW	důkaz provedení práce – Proof of Work
PoS	důkaz hodnoty – Proof of Stake
HTTP	hypertextový transportní protokol – Hypertext Transfer Protocol
FTP	protokol pro přenos souborů – File Transfer Protocol
dApps	decentralizované aplikace – Decentralized Applications
DeFi	decentralizované finance – Decentralized Finance
NFT	nezaměnitelné tokeny – Non-fungible Tokens
IPFS	meziplanetární souborový systém – InterPlanetary File System
ECDSA	protokol digitálního podpisu s využitím eliptických křivek – Elliptic Curve Digital Signature Algorithm
UTXOs	neutracené transakční výstupy – Unspent transaction outputs
BIP	návrh na vylepšení Bitcoinu – Bitcoin Improvement Proposal
SMPC	zabezpečené počítání s více stranami – Secure Multi-party Computation
API	rozhraní pro programování – Application Programming Interface
CKD	odvození dětského klíče – Child Key Derivation
JSON	zápis objektů JavaScript – JavaScript Object Notation

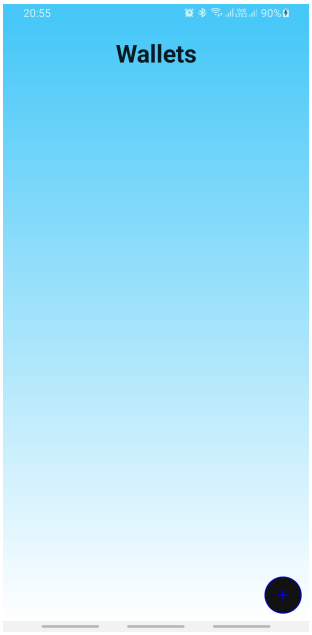
A Manuál

Instalace

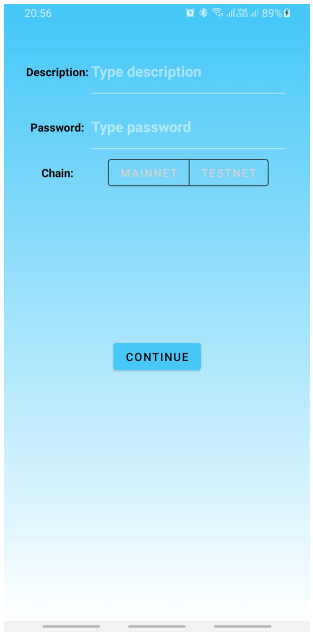
Pro instalaci aplikace je třeba stáhnout *.apk* soubor do mobilního telefonu a chytrých hodinek a nainstalovat dle pokynů výrobců daných zařízení. Aplikace dále již potřebují pouze povolení k využívání internetu a komunikace Bluetooth. Pro využití kombinace mobilního telefonu a chytrých hodinek je zapotřebí, aby byly hodinky spárované s telefonem. Aplikace podporuje jen Android zařízení.

Vytvoření peněženky

Po instalaci a spuštění aplikace na mobilním telefonu se objeví obrazovka se seznamem peněženek. V tento moment je seznam prázdný, jak lze vidět na obrázku A.1. Pro vytvoření je nutné stisknout tlačítko *plus* v pravém dolním rohu. Po stisknutí se objeví obrazovka pro vytvoření peněženky, viz obrázek A.2. Zde je nutné doplnit kolonku *.apk* pro identifikaci peněženky. Kolonka *Password* je volitelná a může a nemusí být vyplněna. Poslední částí je výběr sítě, kterou bude peněženka podporovat. Pro testovací účely je vhodné vybrat síť *Testnet*. Poté se pro potvrzení stiskne tlačítko *Continue*. Po stisknutí tlačítka se objeví obrazovka s mnemonic kódem, který představuje privátní klíč, viz A.3. Tento seznam si uživatel zapíše pro případnou migraci peněženky nebo při ztrátě zařízení je možné peněženku obnovit. Po potvrzení tlačítkem *OK* by se měla na obrazovce ukázat nově vytvořená peněženka.



Obr. A.1: Prázdný seznam peněženek.



Obr. A.2: Vytváření peněženky.

Obr. A.3: Mnemonic kód.

Vytváření peněženky u hodinek je postupově podobné. Po otevření aplikace se zobrazí obrazovka pro vytvoření peněženky, viz obrázek A.4. Po vytvoření se objeví hlavní obrazovka s MuSig2 účtem a účtem prahového podpisu, viz obrázek A.5.

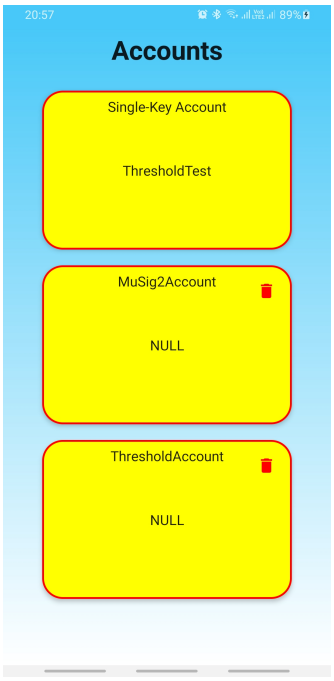
Obr. A.4: Vytváření peněženky na hodinkách.

Obr. A.5: Obrazovka s MuSig2 účtem a účtem prahového podpisu.

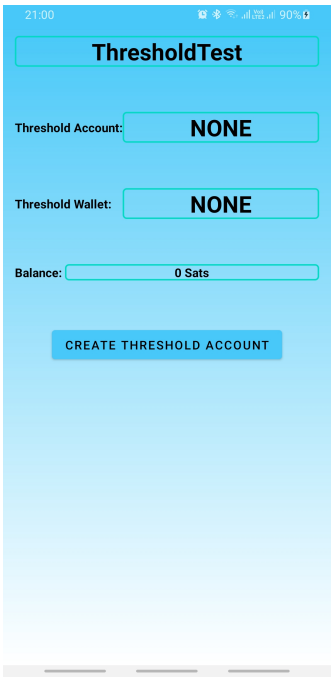
Vytvoření účtů a společné peněženky

Po vytvoření peněženky je automaticky vytvořený účet s jedním klíčem a ostatní účty jsou prázdné, jak je vidět na obrázku A.6. Pro výběr některého z účtu stačí stisknout kartu s daným účtem. Na obrázku A.7 je zobrazen prázdný účet s prahovým podpisem. Pro vytvoření tohoto účtu se stiskne tlačítko *Create Threshold Account*. Po stisknutí tlačítka se objeví obrazovka pro vytváření tohoto účtu, viz obrázek A.8. Je nutné zadat všechny parametry. Parametr n zadává celkový počet vlastníků peněženky a parametr t udává práh, tedy počet částečných podpisů nutných pro vznik podpisu. Pro testování s hodinkami a jedním mobilním telefonem je doporučen $n = 2$ a $t = 2$. Toto platí i u MuSig2 účtu. Pro testování s dvěma mobilními telefony a jedněmi hodinkami je doporučen $n = 3$ a $t = 2$.

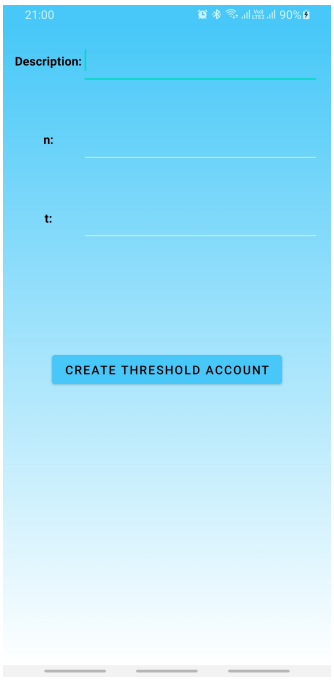
Při vytváření například společné peněženky prahového podpisu s hodinkami a dvěma telefony je nutné mít na všech zařízeních správně vytvořeny účty prahového podpisu. Dále při vytváření musí být na hodinkách otevřená aplikace v účtu prahového podpisu a oba mobily v obrazovce pro připojení Bluetooth, což se provede stisknutím tlačítka *Create Threshold Wallet*. V této obrazovce, viz obrázek A.9, jeden mobilní telefon startuje server pomocí tlačítka *Start Server* a druhý ho musí najít v naskenovaných zařízeních a výběrem se k němu připojí. Nyní hlavní zařízení (má připojené hodinky a druhý mobilní telefon) zadá název vytvářené peněženky a potvrdí stisknutím tlačítka. Na hodinkách se objeví obrazovka pro zadání názvu a po potvrzení tlačítkem *Create THWallet*, viz obrázek A.10, započne komunikace



Obr. A.6: Účty peněženky.



Obr. A.7: Prázdný účet prahového podpisu.



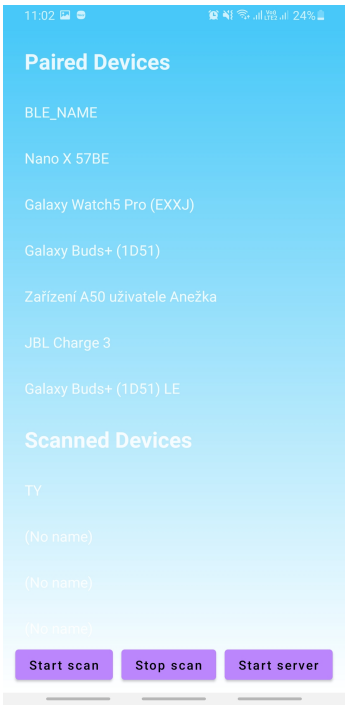
Obr. A.8: Vytváření účtu prahového podpisu.

mezi zařízeními a následně se vytvoří společná peněženka. (Pozn. Bluetooth bylo implementováno a otestováno pouze pro zařízení s Android 10 (API 29), tudíž nemusí fungovat na novějších zařízeních.)

Vytváření a podepisování transakcí

Před vytvářením transakcí je nutné peněženku nabít testovacími Bitcoiny. K tomu lze využít veřejných faucetů, například tento Faucet. Pro testovací účely je doporučeno požádat například o 6000 Satoshi. Pro testovací účely je doporučeno odesílat z přijatých 6000 Satoshi z faucetu rovnou 5000 Satoshi z důvodu, že transakce využijí celou UTXO a nezasílá zbytky zpátky do peněženky.

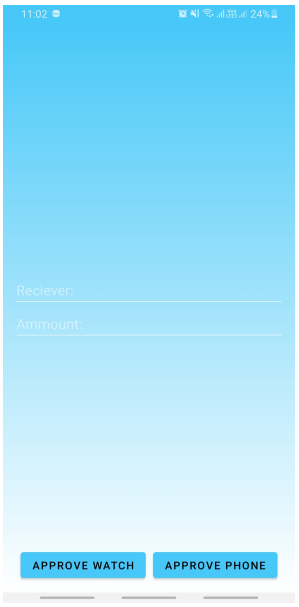
Po nabití lze nyní tvořit transakce. Na obrázku A.11 je vyobrazena obrazovka pro vytvoření transakce u účtu prahového podpisu. Zde se zadá komu a kolik se bude zasílat. Pro zaslání požadavku o schválení transakce, tedy požadavek pro částečný podpis sekundárního zařízení, se stiskne buď tlačítko *Approve Watch* a nebo *Approve Phone* dle toho, s kým byla vytvořena peněženka a od jakého zařízení chce uživatel částečný podpis. V případě hodinek je nutnost mít otevřenou aplikaci v daném účtě a u mobilního telefonu mít otevřené okno pro schválení pomocí tlačítka *Approve TX*. Na obrázku A.12 je vyobrazen příklad stisknutí tlačítka *Approve Watch*. Po stisknutí se na hodinkách objeví okno s informacemi komu a kolik se zasílá. Tlačítkem *Approve* hodinky potvrdí schválení a zasílají částečný podpis. Po přijetí a agregování



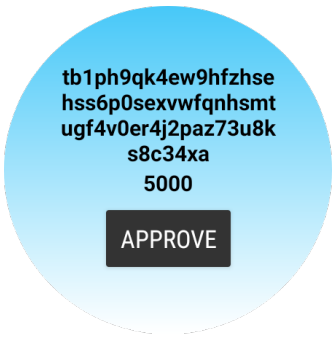
Obr. A.9: Obrazovka pro připojení Bluetooth.

Obr. A.10: Vytváření společné peněženky prahového účtu v hodinkách.

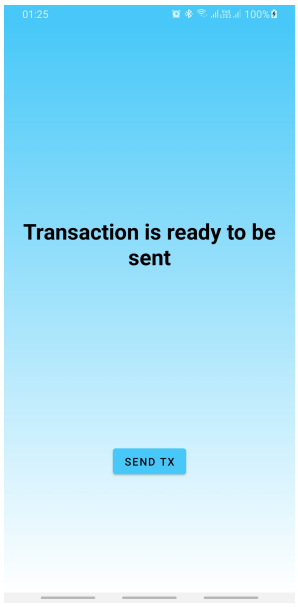
částečných podpisů do jednoho finálního se objeví obrazovka pro zaslání transakce, viz obrázek A.13. Pomocí tlačítka *Send TX* je transakce zaslána na blockchain.



Obr. A.11: Obrazovka pro vytvoření transakce.



Obr. A.12: Obrazovka pro schválení transakce v hodinkách.



Obr. A.13: Obrazovka pro zaslání transakce.