VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

DECENTRALIZOVANÝ AUTENTIZAČNÍ SYSTÉM ZALOŽENÝ NA BLOCKCHAINU

DECENTRALIZED AUTHENTICATION SYSTEM BASED ON BLOCKCHAIN

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ŠTEFAN MRAČNA**
AUTHOR

VEDOUCÍ PRÁCE — **FILIP JANUŠ, Ing.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Mračna Štefan**
Program: Informační technologie
Název: **Decentralizovaný autentizační systém založený na blockchainu**
Decentralized Autentication System Based on Blockchain
Kategorie: Bezpečnost

Zadání:

1. Nastudujte decentralizované a centralizované autentizační biometrické systémy a základy technologie blockchain.
2. Porovnejte decentralizovaný a centralizovaný přístup k autentizaci.
3. Navrhněte decentralizovaný biometrický autentizacni systém, provádějící své operace distribuovaně.
4. Implementujte navržený systém a demonstrujte funkčnost.
5. Diskutujte zranitelnosti a výkonnost navrženého systému.

Literatura:

- Cachin, C., & Vukolić, M. (2017). Blockchain consensus protocols in the wild. arXiv preprint arXiv:1707.01873.
- Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2016). Bitcoin and cryptocurrency technologies: a comprehensive introduction. Princeton University Press.
- Delgado-Mohatar, O., Fierrez J., Tolosana R., & Vera-Rodriguez R. (2020). Blockchain meets Biometrics: Concepts, Application to Template Protection, and Trends

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Januš Filip, Ing.**
Konzultant: Malaník Petr, Ing., UITS FIT VUT
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 3. listopadu 2021

Abstrakt

Tato práce se zabývá zavedením blockchainu do autentizačního systému, konkrétně do biometrického autentizačního systému pro otevírání dveří. Práce analyzuje různé blockchainové technologie a autentizační systémy a určuje, které technologie jsou pro tyto účely vhodné. Porovnává také centralizované a decentralizované autentizační systémy. Na základě analýzy se navrhuje decentralizovaný autentizační systém se zabudovaným blockchainem a zjednodušená verze systému, kde individuální zařízení jsou reprezentovány procesy, se implementuje. Nad tímto systémem je vytvořen program simulující provoz. Nakonec se provádí testování výkonnosti, bezpečnosti, spolehlivosti a paměťové náročnosti systému v různých podmínkách.

Abstract

This work studies the effects of the integration of blockchain technologies into authentication systems, specifically biometric authentication systems for unlocking doors. The work analyzes different blockchain and authentication technologies and determines which of these technologies are appropriate for this purpose. The work also compares centralized and decentralized authentication systems. A decentralized authentication system which includes blockchain technologies is then proposed and a simplified version of the system (where individual devices in the system are represented by processes) is implemented. Testing of effectiveness, security, reliability and memory usage is then conducted under different conditions.

Klíčová slova

Blockchain, autentizace, ethereum, smart contracts, decentralizované systémy, byzantine fault, PBFT

Keywords

Blockchain, authentication, ethereum, smart contracts, decentralized systems, byzantine fault, PBFT

Citace

MRAČNA, Štefan. *Decentralizovaný autentizační systém založený na blockchainu.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Filip Januš, Ing.

Decentralizovaný autentizační systém založený na blockchainu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Filipa Januše, Ing. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Štefan Mračna
8. května 2022

Poděkování

Chtěl bych poděkovat vedoucímu práce, Filipu Janušovi, Ing. za jeho přínosy při vedení této práce.

Obsah

Kapitola 1

Úvod

V dnešní době se převážně používají tzv. centralizované autentizační systémy, kde uživatele autentizuje nějaká centrální autorita. Tyto autentizační systémy mají své problémy, zejména kvůli existenci jednoho bodu selhání (single point of failure), který může potenciálně způsobit výpadek celého systému při útoku na jeden z jejich prvků. Tyto nedostatky by se daly řešit za pomocí decentralizace.

Blockchainové technologie jsou jednou z metod decentralizace. Zaručují integritu transakcí bez potřeby centralizované autority za pomocí tzv. consensus protokolu. Mohou tak zabezpečit určité aspekty autentizačního systému, aniž by spoléhaly na jedno zařízení. Blockchainové technologie a jejich využití není limitováno jen na oblast financí. Byly využity či studovány i v rámci jiných oborů, jako je například zdravotnictví [17], energetika [20], problematika Internet of Things (IoT) [19] nebo ověřování certifikátů [24]. Tyto technologie ale přinášejí určitou sadu problémů a nedostatků, se kterými se musí při návrhu systému počítat. Jedná se především o nedostatek soukromí, a u některých blockchainových technologií i celková vysoká spotřeba energie a s tím spojené zatížení elektrické sítě, které může vést k akci od regulátorů a legislatur na omezení či zákaz technologie. Posledně zmíněný problém pravděpodobně vedl k zákazu kryptoměn v 8 zemích světa (mezi něž patří i Čína, Bangladéš či Egypt), s těžkopádnými regulacemi v některých dalších zemích (např. v Turecku se nesmí provádět komerční transakce s kryptoměnami)[9].

1.1 Cíle bakalářské práce

Práce má za cíl analyzovat integraci blockchainových technologií do autentizačních systémů a na základě této analýzy navrhnout autentizační systém tak, aby byl odolný proti vybraným bezpečnostním hrozbám. Poté by se zjednodušená verze autentizačního systému měla implementovat (kde individuální zařízení jsou reprezentována procesy) a nasimulovat různé scénáře (např. běžný provoz, vysoký provoz, případy, kdy je jeden či více porovnávačů napaden apod.). Na základě výsledků těchto simulací by se pak měl zhodnotit návrh a implementace systému a případně navrhnout některá zlepšení jako podmět pro další práci.

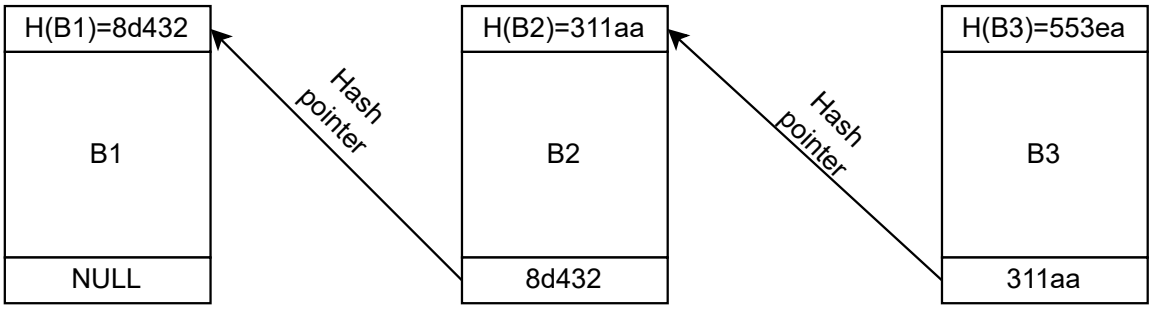
Kapitola 2

Hlavní koncepty

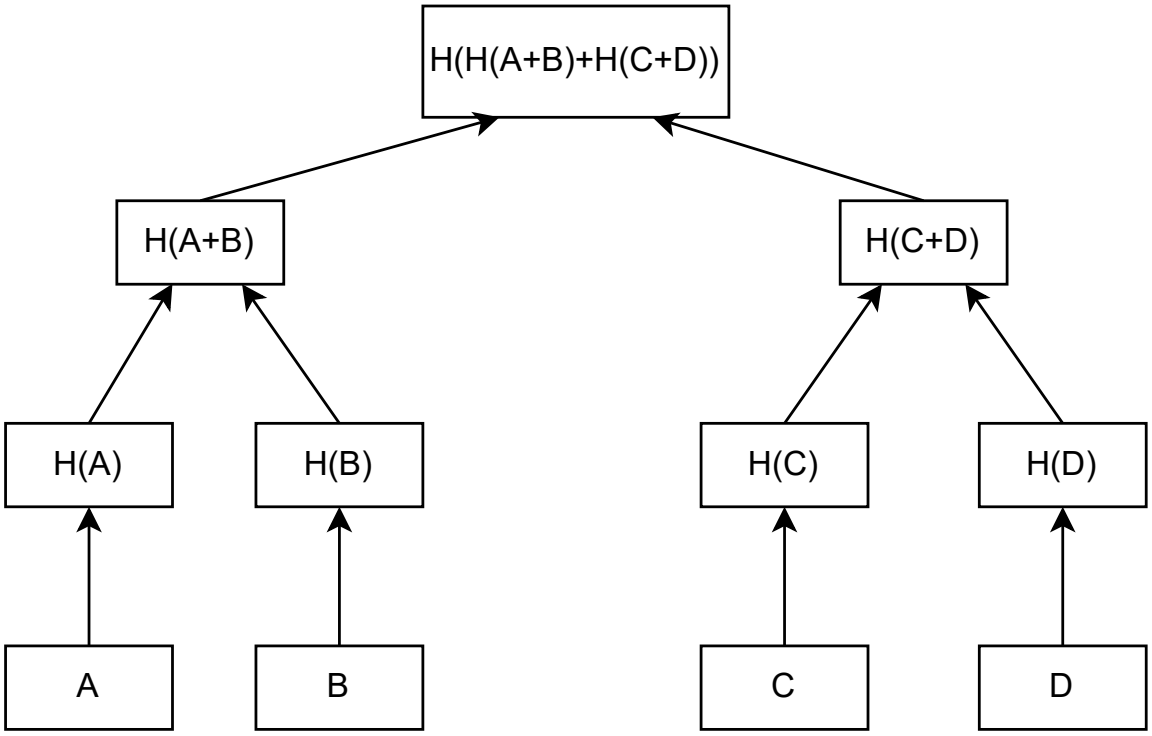
2.1 Blockchainové technologie

Koncept decentralizované autentizace transakcí byl teoreticky rozveden již v osmdesátých letech a první návrh blockchainového protokolu byl vytvořen v roce 1991 v práci Habera a Stornetty [18]. První kryptoměny byly navrhnuty v druhé polovině devadesátých let, ale první úspěšnou kryptoměnou založenou na blockchainových technologiích byl Bitcoin, navrhnutý člověkem (či skupinou lidí), působící pod pseudonymem Satoshi Nakamoto, jehož koncept byl zveřejněn v roce 2008 [22].

Blockchain jakožto struktura je založená na dvou konceptech – struktura lineárního seznamu a kryptografický hash. Kryptograpfický hash je řetězec konzistentní délky, do kterého se pomocí hashovacího algoritmu převede text o jakékoliv délce. V blockchainu se využívá ve formě tzv. hash pointerů, což jsou ukazatele na datovou strukturu, které uchovávají i kryptografický hash dané struktury. Blockchain v základu tyto hash pointery využívá v kombinaci s lineárním seznamem. Každý blok (kromě startovního bloku, tzv. genesis blok) obsahuje hash pointer na předešlý blok, což se opakuje tak dlouho, dokud se nedosáhne genesis bloku. Blockchain tak zajišťuje integritu všech bloků až po genesis block. Pro zvýšení efektivity vyhledávání jsou bloky v blockchainu dále děleny na individuální záznamy, které jsou organizované do tzv. merkelovských stromů (v podstatě binární strom, s ukazately ve formě hash pointerů). Mnohé blockchainové technologie také vyžadují podepisování transakcí, aby se systém zabezpečil proti jejich falšování [22].



Obrázek 2.1: Vizualizace struktury blockchainu



Obrázek 2.2: Vizualizace struktury merklovského stromu

Dalším aspektem blockchainových technologií je uchovávání blockchainu v zařízeních a přidávání bloků do blockchainu. Blockchain je uchováván v tzv. uzlech (nodes). Každý uzel uchovává celý blockchain, což může být problém pro škálovatelnost (každé zařízení fungující jako uzel musí mít dostatečnou paměť na uchování blockchainu, například u bitcoinu se jedná o zkruba 390 GB dat [6]). Přidávání bloků do blockchainu probíhá za pomocí tzv. consensus protokolu, jehož specifická implementace může být různá. Obecně přidávání transakcí funguje tak, že uzel přidá transakci (záznam) do svého bloku. Po určité době na základě specifického consensus protokolu je vybrán uzel, který odvysílá svůj blok. Po příjmutí bloku si uzel ověří, jestli informace na příjmutém bloku jsou konzistentní a jestli byl blok odvysílán podle dohodnutého consensus protokolu. Pokud jsou tyto informace konzistentní, tak daný uzel přidá blok, který přijal do svého blockchainu [22].

Takto nastavený systém by, za předpokladu, že většina bloků bude fungovat správně a čestně, měl zajistit konzistenci transakcí v blockchainu. Existují zde ale určité problémy, které se musí při návrhu blockchainového systému vzít v potaz. Jedním z těchto problémů je tzv. problém byzantinských generálů. Tento teoretický problém je založený na tom, že mohou existovat tři typy uzlů – čestné fungující uzly, nesprávně fungující uzly vysílající nesprávné informace a nepřátelské uzly. Decentralizovaný systém teoreticky může detekovat nesprávně fungující uzly, pokud je jejich počet menší než polovina. Mohou ale existovat i nepřátelské uzly, které se aktivně snaží narušit chod systému. Mohou například podávat špatné informace o tom, jak se chovají, či nesprávně hlásit, které uzly fungují či vysílají nekorektní informace. K tomu, aby systém nebyl schopný detekovat chybně komunikující uzly, je potřeba, aby aspoň třetina uzlů byla nepřátelská, neboť by takový počet nepřátelských uzlů mohl „přehlasovat“ čestné uzly v případech, kdy polovina „čestných“ uzlů nefunguje korektně [13].

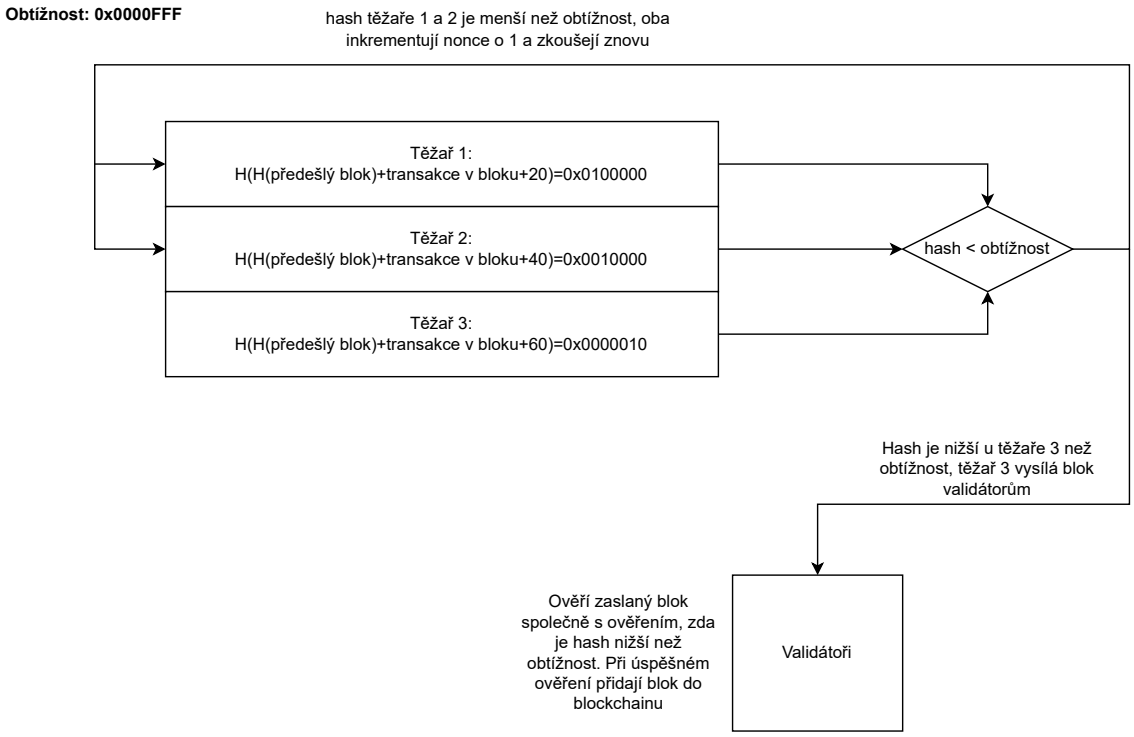
Některé blockchainové protokoly se tomuto problému vyhýbají na základě tzv. pobídek, kdy jsou čestně fungující uzly podporovány v čestném chování nějakou odměnou za práci či za vlastnění nějaké měny. Jedná se téměř výhradně o blockchainové protokoly, které implementují kryptoměny. Tyto protokoly motivují čestně chovající se uzly peněžní odměnou v kryptoměně, a to buď za pomocí práva „danit“ transakce, nebo za pomocí odměn za odvysílaný blok. Každý uzel je tak motivován k tomu, aby jeho bloky byly rozvíjeny v rámci blockchainu, aby tyto získané peníze mohl využít [22].

Blockchainové protokoly se dělí na základě toho, kdo má k blockchainu přístup. Dělí se tak na tzv. permissionless (bez nutnosti povolení – veřejné (public)) a permissioned (s nutností povolení – soukromé (private)) blockchainy. Permissionless protokoly nijak neregulují, kdo může být uzlem, a jenom definuje pravidla pro komunikaci mezi nimi. Permissioned protokoly regulují, kdo může být uzlem. Na rozdíl od klasických centralizovaných systémů ale pořád samotná správa transakcí není centralizovaná, pouze je centralizovaně (správcem sítě) určeno, která zařízení budou fungovat jako uzly [13].

2.1.1 Bitcoin

Bitcoin je první obecně rozšířenou kryptoměnou založenou na blockchainu. Jeho koncept byl zveřejněn v roce 2008 a genesis blok byl vytěžen v roce 2009. Vývojářem tohoto blockchainového protokolu byl anonymní člověk (nebo skupina lidí) pod pseudonymem Satoshi Nakamoto.

Bitcoin je veřejným blockchainem, jehož consensus protokol se řídí principem Proof of Work (PoW). Takový protokol je založený na „hádání“ správného čísla. Uzel, který chce odvysílat blok, tedy musí najít takové číslo, které po konkatenaci s obsahem bloku a jejím vyhashováním vygeneruje hodnotu, která je menší než cílová hodnota, reprezentující obtížnost. Pokud nedojde k nalezení problémů s bitcoinovým hashovacím algoitmem (SHA-256), tak je hledání cílové množiny hodnot čistě otázkou náhody. Proces hledání cílové množiny hodnot se nazývá v blockchainových kruzích těžba (mining) a v bitcoinu je odměňována právem připsat si určité množství kryptoměny na svůj účet [22].



Obrázek 2.3: Vizualizace consensus protokolu PoW

Výhodou PoW je, že dokáže jednoduše určit, kdo má „právo“ vysílat blok, neboť každý uzel může přiloženou hodnotu a obsah bloku zahashovat a získat tak požadovanou hodnotu. Také se zde dá jednoduše a dynamicky měnit „obtížnost“ práce potřebné k vyhledání bloku pouhou změnou cílové hodnoty. To zaručuje, že si bitcoin ponechává konzistentní block time (dobu potřebnou k vytěžení bloku) nezávisle na výpočetních kapacitách těžařů. Další výhodou je, že tento consensus protokol se nedá jednoduše obejít bez investice velkého množství peněz do výpočetní techniky. Protože PoW je založený na hádání správné hodnoty, a tato hodnota se nedá nijak vyvodit (je založená na náhodě), je potřeba aplikovat hrubou sílu na její správné zjištění. Tato hrubá síla stojí na využívání co největší výpočetní kapacity zařízení k uhodnutí hodnoty. Šance na vytěžení bloku tedy přímo závisí na podílu výpočetní kapacity zařízení vlastněné těžařem a celkové výpočetní kapacitě, která se problému věnuje. Garantuje se tak, že si jeden uživatel nedokáže zajistit neférovou výhodu připojením velkého množství zařízení do blockchainové sítě jakožto uzly [22].

S PoW je bohužel spojena jedna velká nevýhoda, kterou je velká energetická spotřeba a zátěž na produkci určitých výpočetních obvodů. Jak bylo zmíněno dříve, šance na úspěšně vytěžený blok závisí na množství výpočetní techniky nasazené pro účel těžby. Společně s peněžní motivací vykonávat těžbu PoW motivuje k vykupování velkého množství techniky a k vysoké spotřebě energie, což je problém, který se zhoršuje s širším využitím bitcoinu a jeho stoupající cenou. Momentálně se k zajištění bitcoinu využívá celkově 204,5 TWh elektrické energie (což je trojnásobek celkové spotřeby energie v ČR) [2]. Tento problém efektivně znemožňuje využití bitcoinu v širší společnosti, neboť její velká spotřeba energie je neudržitelná a při celospolečenské aplikaci by vedla k masivnímu nárustu ceny energií. S tímto problémem je také spojen negativní dopad těžby bitcoinu na životní prostředí,

zejména emise spojené s generováním elektřiny spotřebované těžbou a elektronický odpad [2].

PoW společně se systémem peněžních odměn za provoz blockchainu a blockchainové sítě umožňuje určitou formu odolnosti proti útokům. Pořád zde ale existují útoky, které mohou být na blockchainovou síť provedeny. Zejména se jedná o tzv. 51% útok a od něho odvozený feather fork útok. 51% útok může nastat, pokud útočník vlastní nadpoloviční množství výpočetního výkonu vynaloženou na těžbu. Útočník tak může získat kontrolu nad určitými aspekty blockchainu, které mohou podrýt důvěru v systém. Může například odmítnout nově příchozí transakce nebo provést tzv. double spending (dvojitá platba se stejnými penězy – většinou jednou na adresu legitimního obchodníka a jednou na adresu vlastněnou útočníkem). Feather forking představuje „odlehčenou“ verzí 51% útoku. Není na něj potřeba většina výpočetních zdrojů, ale takový útok bude úspěšně proveden jen s určitou pravděpodobností, která závisí na podílu vlastněných zdrojů a počet souvislých bloků, který je potřeba současně vytěžit, aby ostatní těžaři označili blok útočníka za věrohodný [20]. Feather forking se především používá na odmítnutí transakcí od určitého zdroje [20].

2.1.2 Ethereum

Ethereum byl původně koncipován v roce 2013 Vitalikem Buterinem a jeho genesis block byl vytěžen v roce 2015. Hlavní novou vlastností etherea je jeho podpora tzv. smart contracts.

Smart contract je transakce reprezentující program. Po jejím vytvoření zůstává uložený v blockchainu a provede se, pokud jsou splněny určité podmínky v něm definované. V ethereu jsou tyto transakce prováděny na straně uzlu ve virtuálním prostředí (Ethereum Virtual Machine – EVM), aby se kód mohl provést stejně pro všechny možné uzly. Pokud chce uživatel využít těchto služeb, musí za ně zaplatit formou transakčního poplatku (měřen v jednotce gas – denominace kryptoměny etherea). Tento poplatek je vyměřován většinou úměrně k výpočetní technice či k paměti, kterou je potřeba vynaložit na provedení smart contracts. Primárním účelem smart contracts je zajistit podmínky dohody mezi dvěma stranami bez potřeby zaručení dohody certifikovanou autoritou. Má ale i mnohé jiné účely, například pro uložení dat, přiřazení certifikátů k duševnímu vlastnictví (ve formě nezaměnitelných tokenů – non-fungible tokens – NFT) nebo vytváření vlastních kryptoměn derivovaných z etherea (což se může například aplikovat u soukromých blockchainů).

Ethereum využívá consensus protokolu proof-of-work, ale jsou již plány na převod etherea na proof-of-stake [10]. U proof-of-stake jsou těžaři nahrazení tzv. validátory, jejichž práce je ověřovat správnost vysílaných bloků. Validátorem může být kterýkoliv uzel, který předloží určité množství vlastněné kryptoměny (v ethereu se jedná o 32 ETH [10]). Pokud je potřeba přidat blok do blockchainu, je vybrán náhodně jeden z validátorů. Ostatní validátoři pak ověřují tento vytvořený blok. Validátoři za tyto aktivity dostávají odměny ve formě množství kryptoměny. Také ale mohou ztratit část či celou předloženou částku, pokud se nebudou chovat tak jak mají (např neověřují bloky nebo se chovají nepřátelsky). Proof-of-stake zajišťuje odolnost systému proti útokům (možná i víc než proof-of-work)[4] u velkých kryptoměn, protože by potenciální útočník musel investovat velké množství kryptoměny (a tím pádem i opravdových peněz), které by pravděpodobně ztratil po provedení útoku. Také není potřeba pro její provádění spotřebovávat velké množství energie. Pro menší kryptoměny mohou ale představovat problém, neboť omezuje množství uzlů, což může vést k jednodušším útokům na blockchainovou infrastrukturu. Proof-of-stake je také přímo vázaný na kryptoměnu, což omezuje její využití v nefinančních oborech.

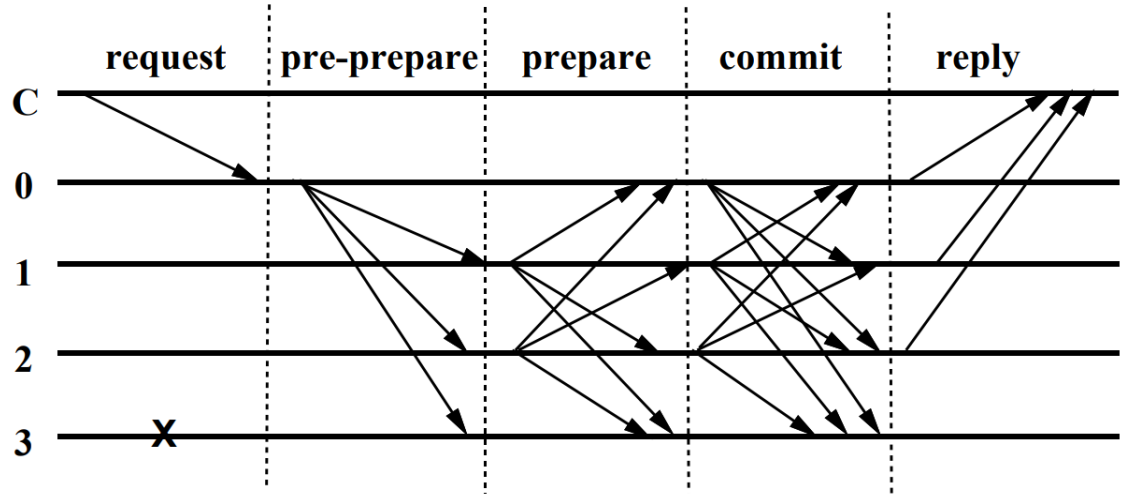
2.1.3 Jiné technologie

Tato kapitola se bude zabývat některými alternativními blockchainovými technologiemi, zejména soukromými blockchainy (které jsou relevantní pro tuto práci). U níže zmíněných technologií figuruje tzv. problém byzantských generálů, který je popsán v kapitole 2.1.

PBFT – Practical byzantine fault tolerance (PBFT) protokol je založený na hlasování a garantuje odolnost proti výpadkům uzlů a proti nepřátelským uzlům. Jedná se o soukromý blockchain. Uzly v PBFT mají dvě role – primární (vedoucí) uzel a sekundární uzly. Primární uzel generuje kandidátní blok. Sekundární uzly kandidátní blok ověří a hlasují o tom, zda daný blok má být přidaný do blockchainu.

Komunikace je složena ze tří zpráv (pokud nepočítáme uživatelský požadavek). Během *requestu* se zašle operace od klienta k vedoucímu uzlu. Vedoucí uzel požadavek ověří a přidává transakce do blockchainu. Po určitém blokovém času odvysílá vedoucí uzel blok ostatním uzlům v rámci zprávy (*pre-prepare*). Ostatní uzly odvysílaný blok validují a v případě správného ověření zašlou *prepare* zprávu ostatním uzlům. Poté každý uzel naslouchá a při získání prepare zpráv od 2/3 uzlů zašle *commit* zprávu ostatním uzlům. Pokud uzel dostane od aspoň dvou třetin uzlů *commit* zprávu potvrzující odvysílaný blok, přidá uzel blok do svého blockchainu. Pokud selže primární uzel, mohou ostatní uzly vyvolat hlasování o změně tokenu. Pokud se najdou dvě třetiny těchto hlasů, tak se vedoucí token změní [14].

Výhodou tohoto protokolu je, že je odolný jak vůči nefunkčním uzlům (dokud je jejich počet menší než třetina celkového počtu uzlů) tak proti nepřátelským uzlům (dokud jich je méně než třetina). Také nevytváří větve (forks). Její hlavní nevýhodou je intenzita komunikace, neboť počet zaslaných zpráv je kvadraticky úměrná s počtem komunikujících uzlů, což může představovat vysokou zátěž sítě [13].



Obrázek 2.4: Schéma komunikace blockchainové sítě v rámci protokolu PBFT. Zdroj: [14]

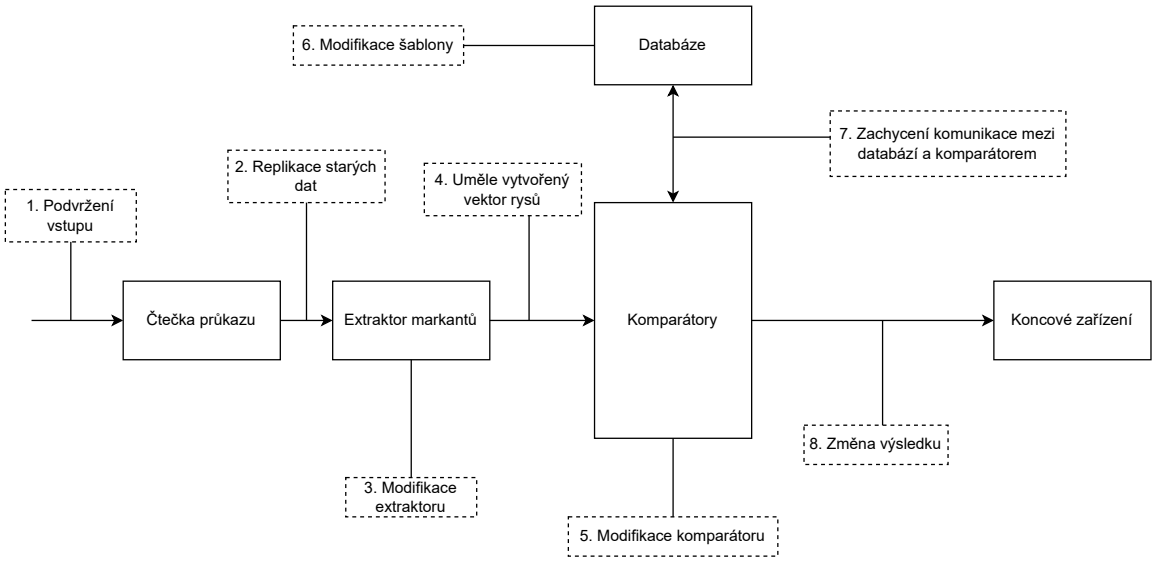
PaLa protokol – Jedná se o úpravu PBFT protokolu. Představuje tzv. epochy, které určují, jak dlouhý blockchain si daný uzel vede. Každý uzel si aktualizuje svůj blockchain, pokud najdou blockchain s vyšší epochou. Dále rozděluje bloky na 3 typy - navrhnuté (navrhuje vybraný uzel, ostatní autentizují jeho pravost), notarizované (2/3 uzlů hlasovaly pro jeho pravost) a finalizované (předek notarizovaného bloku, nedá se dále již v blockchainu měnit). V případě výpadku navrhovatele uzel po timeoutu čeká určitý úsek času, po němž

se sám stane navrhovatelem a navrhne nový blok s vyšší epochou. Tento protokol snižuje intenzitu komunikace na 2 zprávy - návrh bloku a hlasování. Také zachovává požadované vlastnosti PBFT – například odolnost proti 1/3 nepřátelských uzlů [15].

PoA – Proof of Authority (PoA) je modifikace PoS protokolu, kde účastník neinvestuje peníze, ale místo toho investuje svoji reputaci. O verifikaci bloků se stará malá skupina autentizovaných uzlů – validátoři – jejichž identita je veřejně známá. Motivace ke správnému chování je právě podložená identita uživatele, který operuje uzel, neboť nepřátelské chování se dá s daným uživatelem spojit a tak si může získat špatnou reputaci, kterou se vyloučí z účasti na jiných blockchainových sítích s PoA consensus protokolem. PoA přináší lepší škálovatelnost než PoS (neboť validátorů je málo), ale představuje pro validátory potenciální riziko pro soukromí (neboť jejich identita je veřejná), a nejedná se o moc prakticky otestovanou technologii [5].

Proof of Elapsed Time (PoET) – Jedná se o consensus protokol použitelný jak pro soukromé, tak i veřejné blockchainy. Je založený na čekání po náhodnou periodu času, následovanou broadcastem vytvořeného bloku společně s důkazem o tom, že danou periodu času zařízení počkalo. Čekání se provádí za pomocí speciálního hardwarového modulu od firmy Intel – Software Guard Extensions (SGX), který vygeneruje náhodnou periodu čekání, během této periody čeká a po uběhnutí periody se prohlásí za vůdce a dodá atestaci, za pomocí které se dá u ostatních uzlů ověřit, že daný uzel opravdu počkal dané množství času [13]. Výhodou PoET u veřejných blockchainů je, že není energeticky náročný jako PoW a vázaný na kryptoměnu jako PoS. Pořád je náročný na produkci určitých výpočetních modulů (protože k získání většího vlivu na blockchainu je pouze potřeba nakoupit velké množství SGX modulů a napojit je na síť) a může být náchylný na hardwarové útoky (rollback útoky, key extraction...) [13]. U soukromých blockchainů se mohou tyto moduly autentizovat, ale klasické Byzantine fault tolerant (BFT) metody bývají rychlejší a efektivnější než PoET a navíc nabízejí ochranu proti nepřátelským uzlům [13].

2.2 Autentizační systémy



Obrázek 2.5: Obecné schéma autentizačního systému s popisem možných útoků

Autentizační systémy jsou obecně složeny z pěti komponentů – senzor (snímá autentizační data), extraktor markantů (převede nasnímaná data do strukturované podoby, například do vektoru rysů), porovnávač (porovnává nasnímaná data se šablonami v databázi), databáze (uchovává šablony – strukturovaná data reprezentující vzory registrovaných uživatelů) a konečného zařízení (na popud porovnávače kontroluje zařízení spojené s autentizací).

Na autentizační systém lze provést 8 různých typů útoků [16]:

1. Podvržení vstupu – útočník zfalšuje vstup, ať už se jedná o biometrii či jiné autentizační údaje. Příkladem může být například ukradení citlivých dat od legitimního registrovaného uživatele a její použití pro autentizaci. Mezi metody prevence může být například vyžadování hesla, striktnější nároky na biometrické ověřování apod.

2. Replikace starých dat – útočník použije zastaralá data pro autentizaci. Řadí se sem převážně replay útoky a pokusy o autentizaci za pomocí již neplatných dat. Prevencí je timestamping či využití session ID.

3. Modifikace extraktoru – útočník se pokouší změnit vnitřní fungování extraktoru tak, aby pro jeho vstup mohl porovnávač vrátit pozitivní výsledek. Prevencí je fyzické zajištění extraktoru či kryptografické hashování zdrojového kódu extraktoru, aby se dalo ověřit, že nebylo manipulováno se zdrojovým kódem extraktoru.

4. Uměle vytvořený vektor rysů – útočník se naboural do komunikace mezi extraktorem a porovnávačem a snaží se porovnávači zaslat data tak, aby vrátil pozitivní odpověď. Většinou se jedná o man-in-the-middle (MITM) útoky. Útok může sloužit i k narušení provozu autentizačního systému, například k tzv. blacklistování (znemožnění úspěšné autentizace legitimního uživatele). Mezi metody prevence patří například šifrování s bezpečnou výměnou klíčů.

5. Modifikace porovnávače – porovnávač je útočníkem podvržen či je již existující porovnávač modifikován. Útok slouží k zajištění úspěšné autentizaci útočníka bez ohledu na dostatečnou podobnost šablon v databázi a na vstupu. Při existenci jednoho porovnávače lze použít i pro některé útoky narušovacího charakteru, např. DoS či DDoS útoky. Prevencí je zajištění porovnávače (např. zahashováním zdrojového kódu a monitorováním, jestli tento hash nebyl změněn, nebo fyzické zajištění).

6. Modifikace šablony – útočník se pokusí vytvořit/změnit záznamy přímo v databázi tak, aby v ní existoval záznam o útočníkově vstupu. Útok také může sloužit k modifikaci údajů registrovaných uživatelů, jejich mazání z databáze či krádeži citlivých dat. Mezi metody prevence patří například autentizace zařízení, které s databází pracují, či logování transakcí a monitorování těchto logů (může sloužit ke zjištění útoku).

7. Zachycení komunikace mezi databází a porovnávačem – Podobný princip jako v útoku 4. Jedná se především o MITM a replay útoky. Cíle útočníka mohou být různé. Může se jednat o změnu databázové odpovědi, změnu databázového dotazu (aby databáze vrátila pozitivní odpověď) či využití útoku k provedení útoku 6 (neboť by se mohl útočník tvářit jako legitimní porovnávač). Prevencí je šifrování komunikace.

8. Změna výsledku – MITM útok na komunikaci mezi porovnávačem a koncovým zařízením. Slouží ke změně odpovědi přicházející koncovému zařízení. Mezi metody prevence patří primárně šifrování komunikace.

2.2.1 Centralizované autentizační systémy

Centralizoavné autentizační systémy jsou založeny na tzv. centrální autoritě (CA). Kromě centrální autority je složen z vlastníků dat (uživatelé, kteří data nabízejí serveru), uživatelů dat (uživatelé, kteří z určitého důvodu chtějí přístup k datům) a databázového či cloudového serveru (který uchovává data a zajišťuje přenos mezi vlastníky a uživatele dat). Tento přenos se zajišťuje podle pevně definované přístupové politiky definované centrální autoritou. Centrální autorita je také zodpovědná za rozdělování klíčů mezi individuálními účastníky systému [11].

Autentizace v těchto systémech se provádí za pomocí autentizační techniky definované centrální autoritou (ať už se jedná o heslo, biometriku či certifikát). Centrální autorita rozděluje klíče, za pomocí kterých zajistí bezpečnou komunikaci mezi individuálními účastníky systému. Centrální autorita může také na popud klienta předložit digitální certifikát od certifikační autority, která dokazuje, že centrální autorita je opravdu to, za co se vydává.

Tento způsob autentizace byl dlouhou dobu výhradně aplikován k autentizaci uživatele u služby kvůli relativně snadně implementaci. Centralizovaná autentizace také přináší mnohé jiné výhody, například nízká redundance autentizační operace (neboť autentizaci zpravidla provádí jedno zařízení). Také se jedná o autentizaci založenou na prokázaných a standardizovaných protokolech, což zvyšuje jejich důvěryhodnost. Pokud je centrální autorita plně důvěryhodná a její infrastruktura je odolná proti útokům, tak může efektivně garantovat bezpečnost uložených dat a zajistit bezpečnou práci s ní.

Samozřejmě důvěryhodnost centrální autority je jednou z největších nevýhod těchto systémů. Nelze úplně garantovat, že centrální autorita je stoprocentně důvěryhodná a čestná. Může dojít k zneužití těchto dat (například komercializace či prodej soukromých dat). V některých legislativních podmínkách může být takové zneužití dat dokonce legální. Centrální autorita může vydat špatný certifikát, a individuální certifikáty mohou být obětí útoku na infrastrukturu certifikační autority, padělání certifikátů, či falzifikaci certifikátů. Také tato centrální autorita představuje jediný bod selhání (single point of failure), a s výpadkem této služby dojde k výpadku celého autentizačního systému. Systém je tak náchylný vůči některým útokům, jako jsou DDoS útoky znemožňující provoz systému, nebo úniky soukromých dat, které mohou být pro mnohé uživatele velmi problematické. Také jsou zde určité problémy se škálovatelností, neboť s vyšším množstvím uživatelů a dat ztrácí centrální autorita přehled o individuálních uživatelích a jejich přístupových právech [28][26].

Mnohé z těchto problémů se snaží řešit decentralizované autentizační systémy.

2.2.2 Decentralizované autentizační systémy

Decentralizované autentizační systémy jsou založeny na 4 základních prvcích (podobně jako u centralizované autorizace) – uživatelé dat, vlastníci dat, distribuované centrum a databázový čí cloudový server. Hlavním rozdílem je, že decentralizované autentizační systémy nahrazují centrální autoritu distribuovaným centrem, které může být buď sítí uzlů využívajících blockchain, distribuovanou certifikační autoritou, či distribuovanou atributní autoritou. Vlastník dat předá své informace databázovému serveru a zašle přístupová práva distrubuovanému centru. Databázový server pak na základě těchto práv předává data uživatelům dat [11].

Decentralizace autentizace je primárně koncipována z důvodu eliminování nutnosti důvěry mezi uživatelem a autoritami definující pravidla použití a poskytujících certifikáty. Měla by tak být zvýšená odolnost proti útokům (jako jsou DDoS útoky). Také se díky vyšší kontrole nad daty ze strany uživatele zabrání únikům dat (případně dopady těchto úniků by

se mohly snížit). Decentralizace také eliminuje jediný bod selhání, čímž snižuje náchylnost na selhání. S touto metodou autentizace existují ale i určité problémy. Zejména se jedná o nahrazení důvěry v jednu centrální autoritu důvěrou v distribuovanou síť zajišťujících definici pravidel pro sdílení dat. Jak již bylo zmíněno výše, v některých distribuovaných modelech (například v určitých distribuovaných systémech založených na blockchainu, jako je bitcoin) mohou nastat některé útoky (například 51 % útok, feather fork útok apod.), které mohou podlomit důvěru v distribuované centrum. Také jsou zde problémy se škálovatelností (decentralizované autentizační systémy se často spoléhají na velkou míru redundance) a výkonností takových systémů. Tyto problémy mohou být v některých případech extrémní (například u proof-of-work blockchainů) [8][21][28].

2.2.3 Porovnání centralizovaných a decentralizovaných autentizačních systémů

Výhody centralizovaných autentizačních systémů jsou následující:

- Nižší problémy se škálovatelností [21].
- Nižší redundance při vykonávání a vyšší výkonnost [21].
- Založen na důvěryhodných, otestovaných a standardizovaných praktikách, narozdíl od decentralizovaných autentizačnách systémů [8].
- Snadnější implementace a nasazení.

Nevýhody centralizovaných autentizačních systémů jsou následující:

- Single point of failure a s tím spojené problémy – Náchylnost na DDoS útoky, úniky dat jsou více problematické pro uživatele, falzifikace certifikátů [28].
- Nutnost důvěry v centrální autoritu, popřípadě v certifikační autority [28].
- Uživatel má nízkou kontrolu nad svými vlastními daty [8]

Výhody decentralizovaných autentizačních systémů jsou následující:

- Uživatel má vyšší míru kontroly nad svými daty – Snižuje se tak dopad problémů jako jsou zneužití dat či jejich úniky [8].
- Eliminuje se single point of failure – systém je tak odolný proti DDoS útokům či selháním části systému [28].
- Eliminuje se potřeba jedné centrální či certifikační autority – Systém je tak odolný vůči padělání certifikátů nebo vůči selháním na straně centrální autority (například vydání špatných certifikátů) [28].
- Možnost distribuovat výkonnostní nároky na větší množství zařízení.

Nevýhody decentralizovaných autentizačních systémů jsou následující:

- Nižší škálovatelnost [21].

- Vysoká redundance komunikace mezi uzly, nižší výkon systému [21]. Řešení těchto problémů často bývá založeno na kompromisech s centralizací.
- Založeno na méně otestovaných a ověřených praktikách [8].
- Možnost určitých útoků na systém (např. 51 % útok, feather fork apod.).

Ukazuje se zde, že jsou centralizované autentizační systémy vhodné, pokud je centrální autorita důvěryhodná a pokud je schopná zabránit útokům na její infrastrukturu (zejména DDoS útoky). Decentralizované autentizační systémy tuto nutnost důvěry eliminují, čímž mohou zabránit některým útokům (např. DDoS útoky či útoky na některý z uzlů distribuovaného centra), ale existují u nich problémy se škálovatelností a výkonností.

2.2.4 Biometrika

Biometrickou autentizací se především rozumí převzetí biometrických dat od uživatele (otisk prstů, tvář, duhovka apod.) a její porovnání s uloženým vektorem rysů. Má velké množství výhod nad tradičními autentizačními praktikami. Zejména se jedná o těžké padělání těchto údajů a jednoduchost použití pro uživatele (nemusí si nic pamatovat či u sebe něco nosit). Jsou zde ale problémy se sbíráním těchto informací a s jejich zajištěním (datové úniky mohou znamenat únik velmi citlivých údajů). Mnohé z prací zabývajících se zavedením blockchainu do decentralizovaných autentizačních systému se zabývaly biometrickými daty. Je tedy příhodné analyzovat rozdíly mezi zpracováváním biometrických dat a běžných dat, neboť jsou zde určité překážky, které se musí identifikovat a překonat. Mezi takové překážky patří biohashing a bezpečnostní hrozby (například biometrické dilema).

Jedním z problémů při zavedení biometriky do systému je hashovaní, které se často používá k uchovávání soukromých dat a bude velmi důležitý pro zavedení blockchainu do systému. Klasický kryptografický hash nelze v biometrice použít kvůli vlastnosti, která je u ní žádoucí – avalanche effect. Tato vlastnost zaručuje, že pro malé změny v originálním textu se vygeneruje hash, který je naprosto odlišný. U biometriky se ale musí tolerovat určité nepřesnosti ve snímání, jako je například jiné osvětlení, zdravotní stav uživatele, který může mít vliv na danou biometrickou vlastnost, či úhel a vzdálenost snímání. Tím pádem porovnávání musí tolerovat určitou chybu a hashovací funkce musí reflektovat podobnost biometrických záznamů. To znamená, že by biometrický hash měl pro podobné výsledky vytvářet podobné hashe, a měl by brát v potaz určité aspekty biometrického snímání (například podobnější hashe pro jinak otočené či posunuté záznamy, méně podobné pro samotné biometrické vlastnosti) [27]. Touto problematikou se zabývalo několik různých prací, které častokrát musely problematiku přizpůsobit určitému aspektu biometriky (např. otisky prstů [27], či rozpoznání obličeje [23]).

Biometrické systémy představují i unikátní bezpečnostní hrozby. Jednou z hrozeb je *biometrické dilema*. Je založeno na konceptu, že biometrika v autentizačních systémech sice může v lokálním časově omezeném hledisku zlepšit bezpečnost, ale v globálním hledisku kvůli úniku biometrických informací (podvody, útoky, data leaky, zneužití citlivých údajů společnostmi či vládami apod.) může naopak dojít ke kompromitaci bezpečnosti uživatele. Uniklé biometrické údaje totiž nelze zahodit či nahradit jinými, a tak při úniku biometrických dat se může útočník vydávat za legitimního uživatele, a to i v jiných autentizačních systémech. Další bezpečnostní hrozbou je tzv. *doppelganger attack*. Doppelganger attack zneužívá vlastnosti biometrického systému, který musí mít zahrnutou určitou míru tolerance v rozdílech mezi uživatelem předloženými biometrických daty a uloženými biometrickými

daty. Při velkému úniku biometrických dat tak může útočník úspěšně napodobit legitimního uživatele v nekompromitovaném autentizačním systému, neboť může najít biometrické údaje, které jsou při autentizaci dostatečně podobné k údajům legitimního uživatele [12].

K řešení těchto problémů se používají tzv. *cancelable Biometrics*, neboli *biotokeny* [12]. Cancelable biometrics jsou založeny na zkreslování biometrických dat podle nějakého schématu, aby při úniku biometrických dat se schéma zkreslení dalo zrušit a případně nahradit jiným. Musí podporovat dostatečné množství různých zkreslovacích schémat (tokenů) tak, aby nebyly stejné či vzájemně závislé mezi databázemi. Dále musí zajistit dostatečnou ochranu dat a přesnost, aby se zabránilo doppelganger útokům [12]. Mezi techniky vytváření biotokenů patří biometrické solení, fuzzy schémata nebo biometrické generování klíčů [12].

2.3 Minulé pokusy o zavedení blockchainu do decentralizovaných autentizačních systémů

Práce *Blockchain meets biometrics: Concepts, Application to template protection and trends* [16] se zabývá aplikací biometriky a blockchainových technologií na obecné úrovni. Studuje převážně efekt zakomponování blockchainu do biometrických systémů na síťovou odezvu, rychlost zpracování, cenu v rámci veřejného blockchainu a biometrický výkon. Také se zde studovaly různé metody uchovávání dat v blockchainu (např. uchovávání šablon v blockchainu, uchovávání hashů šablon apod.). K analýze využívá experimentování se smart contracts implementovaném v Ethereu. Biometrika se v studii implementuje za pomocí biometrického hashování a testy se provádí nad databází tváří a nad dynamickými podpisy. Studie zjistila, že přímé uchovávání biometrických informací v blockchainu nebo přímé datové hashování není vhodné pro biometrické systémy. Také je ve studii vyzdvyhnuta potřeba využití Merklovských stromů, které snížují dobu provedení o 10-20 sekund. Ve studii bylo demonstrováno, že porovnávání na blockchainu je proveditelné a efektivní na jednoduchých komparátorech založených na Hammingově vzdálenosti a že dostatečná ochraná vektoru biometrických rysů zajistí kompatibilitu se zákony chránící soukromí (např. GDPR).

Blockchain Technology for Healthcare: Facilitating the Transition to Patient-Driven Interoperability [17] se zabývá využitím blockchainových technologií k zajištění kontroly pacienta nad svými soukromými daty v oblasti zdravotnictví. Studují se zde především efekty zavedení blockchainu na pravidla přístupu k datům, agregaci dat, likviditu dat, identitu pacienta a neměnnost dat. Efekty aplikace blockchainu práce zpracovává teoreticky. Ve studii se identifikují problémy s nasazením blockchainu pro daný případ. Studie vyzdvyhla určité výzvy pro implementaci blockchainu do prostředí zdravotnictví a možná řešení. Prvním vyzdvyhnutím problémem je škálovetelnost, která by se podle studie dala řešit za pomocí soukromých blockchainových technologií či minimalizací dat na shrnutí. Dalším problémem je soukromí, kde studie opět navrhuje soukromý blockchain a také navrhuje, aby se data ukládala mimo blockchain a v blockchainu by byla uložena primárně metadata. Mezi ostatní problémy, které jsou analyzované v rámci studie, patři ochota uživatelů se podílet na systému, pobídky k účasti v systému a problém výměny dat mezi různými institucemi v rámci decentralizovaného systému.

Feather Forking As a Positive Force: Incentivising Green Energy Production in a Blockchain-based Smart Grid [20] se zabývá využitím blockchainových technologií, zejména specifického útoku na blockchainovou síť – feather forking, k demotivaci využití zdrojů z fosilních paliv ke generování elektrické energie soukromými akcionáři na energetickém trhu.

Tohoto by, podle studie, šlo dosáhnout za pomocí vynucení vyšších transakčních poplatků na využití fosilních paliv. Studie kombinuje centralizovanou autoritu s distribuovanými uzly reprezentující akcionáře trhu s energiemi. Studie se zabývá problematikou v návrhové rovině, není zde konkrétní implementace. Také se snaží řešit některé problémy s mnohými blockchainovými technologiemi, například vysokou spotřebu energie. Řešením problému vysoké spotřeby energie podle studie není přechod z proof-of-work na proof-of-stake, ale modifikace proof-of-work omezením intenzity hashování. Omezení chce studie zajistit prováděním blockchainových operací ve speciálním hardwaru poskytnutém centrální autoritou. Modifikovaný proof-of-work vybrala studie zejména kvůli tomu, že proof-of-stake představuje určité problémy pro korektní implementaci a správné využití feather forking útoku. Studie se zabývá problematikou v návrhové rovině, není zde konkrétní implementace.

Bubbles of Trust: A decentralized blockchain-based authentication system for IoT [19] se snaží aplikovat blockchainové technologie k vytvoření autentizačního systému pro Internet of Things (IoT). Motivací této studie je, že kvůli velikosti sítě věcí (IoT) je velmi těžké či dokonce i nemožné využít centralizované autentizační systémy a to, že se individuální zařízení v IoT musí vzájemně autentizovat bez zásahu lidí. K zajištění těchto cílu práce zavádí koncept tzv. *Bubbles of Trust*, neboli bezpečné virtuální zóny, kde se individuální zařízení mohou identifikovat a vzájemně si důvěřovat. Studie také dodává implementaci teoretického konceptu v jazyce C++ a v knihovně Ethereum a analyzuje časovou a energetickou náročnost implementace na Raspberry Pi mikropočítači. V práci je demonstrována také dostatečná bezpečnost a odolnost proti útokům. Mezi zmíněnými problémy ve studii patří zejména neadoptovatelnost v real-time aplikacích, potřeba inicializační fáze a proměnná cena kryptoměny, na které je technologie založená.

Towards a blockchain-based certificate authentication system in Vietnam [24] je pokus vytvořit systém založený na blockchainu, jejímž cílem je se vypořádat s padělanými certifikáty. Činí se tak primárně z toho důvodu, že blockchain je z tohoto hlediska žádoucí kvůli její schopnosti decentralizovaně verifikovat transakce. Studie analyzuje blockchainové technologie a navrhuje autentizační systém založený na blockchainu pro účely detekce falešných vzdělávacích certifikátů. Studie implementuje blockchainový systém v platformě Hyperledger Fabric a přes výkonnostní testy se snaží prokázat praktickou využitelnost navrhnutého systému. Mezi budoucí plány pro expanzi systému patří nasazení systému na více uzlech v různých lokacích či vytvoření systému pro vyhledávání v transakcích.

Kapitola 3

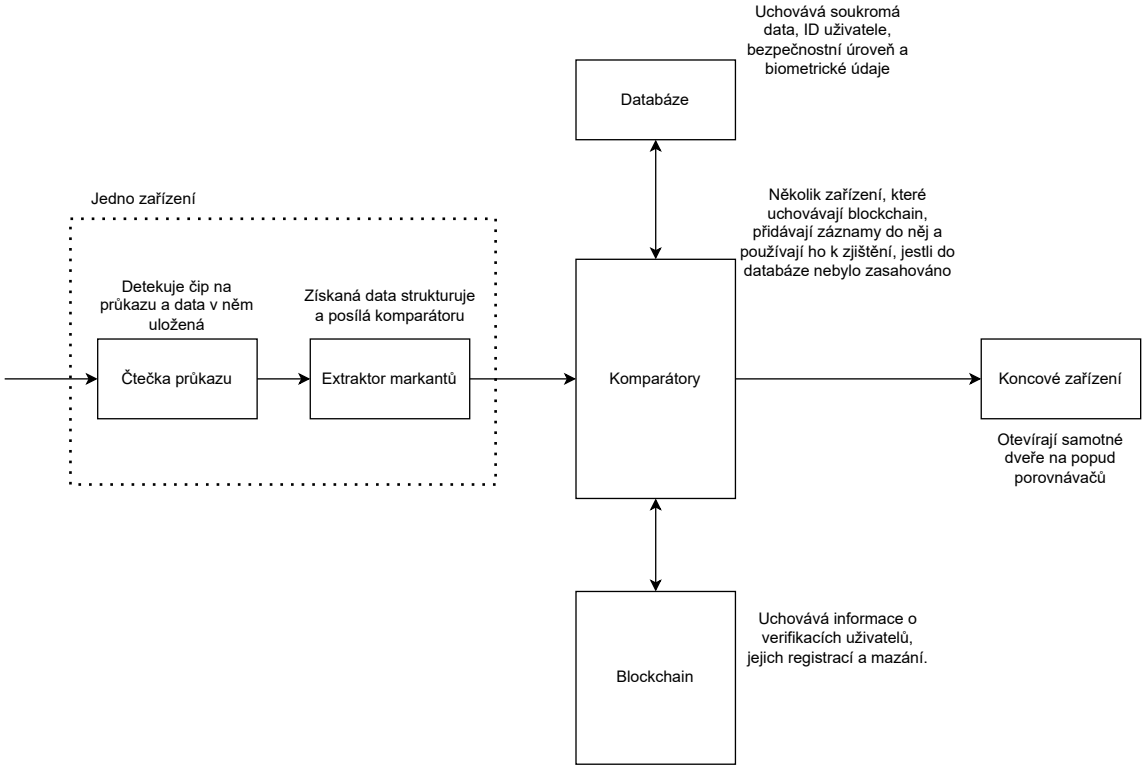
Návrh

V této kapitole se bude navrhovat decentralizovaný autentizační systém, který autentizaci provádí za pomocí blockchainu. Autentizační systém, jak již bylo zmíněno v kapitole 2.2, je složen z 5 komponent – senzoru, extraktoru markantů, porovnávače, databáze a koncového zařízení. V návrhu se bude vybírat specifické zařízení, které bude decentralizováno, které bude uchovávat blockchain a chovat se jako uzel consensus protokolu. Vytyčí se, jaké role budou individuální komponenty hrát v konečném systému. Dále se vybere vhodný consensus protokol pro udržování blockchainu a k provedení verifikace. Specifikují se některé aspekty consensus protokolu a blockchainu (např. block time, kapacita bloku či hashovací algoritmus). Na závěr se popíše základní koncepce fungování navrženého systému. Aspekty návrhu se pak vysvětlují a ospravedlňují v kapitole 3.1

Blockchain bude zaveden do porovnávače. Každý porovnávač (počítá se zhruba s 4 až 10) bude uzlem zachovávajícím blockchain a zodpovědným za přidávání záznamů do něj. Data budou v blockchainu zahashovaná. Blockchain bude fungovat na základě Practical byzantine fault tolerant consensus protokolu (PBFT). Blockchain bude hashován v SHA-256 a individuální bloky budou uloženy v merklovském stromu, který bude mít záznamy organizované podle času přidání. Nový blok se vytvoří zaplněním předešlého bloku a kapacita bloku bude 128 záznamů. Ověřovat pomocí PBFT se bude každý záznam. K tomu, aby koncové zařízení otevřelo dveře, bude muset k tomu získat signál od aspoň dvou třetin porovnávačů. Do blockchainu se budou ukládat informace o procesu který proběhl (verifikace, přidání, či odstranění), hash uživatelských informací, timestamp provedené operace, případný časový odkaz na minulou operaci, která se prováděla s daným uživatelem, a digitální podpis záznamu. V databázi bude uloženo ID uživatele, hash verifikačních údajů uživatele, úroveň zabezpečení uživatele a informace o něm.

Porovnávač při získání ID uživatele, bezpečnostní úrovně skeneru a hashe verifikačních údajů získá z databáze na základě ID uživatele jeho bezpečnostní úroveň a hash jeho verifikačních údajů. Tyto údaje porovná s údaji získanými z databáze. Pokud se shodují verifikační údaje a uživatel má korespondující bezpečnostní úroveň, tak se porovnávač podívá do historie přístupů daného uživatele v blockchainu a zjistí, jestli se informace uložené v blockchainu shodují s informacemi uložené v databázi. Pokud se tyto informace neshodují (např. uživatel uložený v databázi nebyl nikdy registrován, uživatelovi uloženém v databázi skončila platnost průkazu, ale v databázi pořád uložený je. . .)., je uživatelův přístup zamítnut. Pokud se informace shodují, tak je přístup povolen.

Senzor a extraktor markantů bude v jednom zařízení. Senzor bude detekovat čip a číst údaje na něm uvedené. Extraktor markantů bude předělávat tyto údaje do strukturované podoby, a posílat je porovnávačům společně s timestampem.



Obrázek 3.1: Schéma návrhu decentralizovaného systému

3.1 Analýza návrhu

Problémem je zakomponování blockchainu do senzoru, neboť zařízení musí vysílat data, která se mají dále zpracovat do vektoru rysů, což bez narušení soukromí uživatele nelze udělat. Také je nemožné databázi uchovávat v blockchainu, respektive by uchovávání databáze muselo být podmíněno zahashováním citlivých údajů, což by znemožnilo jejich použití pro jiné účely. I logy databáze není možné zavést do blockchainu, neboť by pro zahashovaná data SŘBD nevěděl, jak je správně obnovit. Zavedení blockchainu do extraktoru je možné, ale může se ukázat jako nepřiměřeně náročné. Je to především kvůli implementaci autentizace s biometrickými údaji, které mají vysoké nároky na zpracování. Takové nároky na zpracování mohou vytvářet problémy při implementaci extraktoru v rámci veřejného blockchainu, jako je například ethereum [16], vyšší nároky na uchovávání dat (záznamy reprezentující prováděné operace budou více paměťově náročné), či náročnější síťová komunikace.

Nejvhodnější bude zavést blockchain do porovnávače. Zavedením logů práce s databází do blockchainu se dá při správné implementaci zabránit útokům na databázi (modifikace šablony) a zachycení komunikace mezi databází a porovnávačem [16], neboť jakákoliv práce s databází bude uvedena v blockchainu a bude snadno a spolehlivě dohledatelná, což znemožní útočníkovi nedetekovaně změnit databázi bez kontroly blockchainové sítě.

V blockchainu budou uložena zahashovaná data zejména z důvodu soukromí. Blockchain je totiž veřejný a je potřeba zajistit, aby citlivé informace nebyly snadno čitelné neoprávněným uživatelům. Hashování údajů umožňuje zařízením ověřit integritu údajů, aniž by měla přístup k specifickým citlivým údajům uživatele. PBFT protokol byl vybrán zejména kvůli jeho odolnosti vůči nepřátelským a chybně fungujícím uzlům a nekomplikovanosti z hlediska struktury blockchainu (nevytváří větve (forks) a nemusí se zde řešit, na které z těchto větví bude uzel stavět). Vysoká intenzita přenesených zpráv nebude v našem případě velký problém, protože se bude jednat o lokální síť, která nebude mít velké množství uzlů. Počet přenesených zpráv tak nebude moc limitující (bude se jednat zhruba o nižší stovky zpráv přenesené v rámci celé sítě pro 10 uzlů). Možnost připojení dalších uzlů, které původně v síti nebyly zahrnuty, budou řešeny tak, že se bude modelovat privátní síť, která nebude umožňovat napojení více uzlů bez explicitního svolení správce sítě.

SHA-256 hashovací algoritmus byl zvolen, neboť se již osvědčil v jiných blockchainových technologiích (zejména u bitcoinu). Nebyla zatím pro něj nalezena kolize a tím pádem je (aspoň v době psaní této práce) vhodným algoritmem, který dokáže zajistit integritu vstupních dat. Merklovský strom primárně slouží k ověřování integrity blockchainu (neboť merklovský strom potřebuje k ověření projít méně uzly než běžný lineární seznam).

Relativně nízká velikost bloku (v porovnání například s bitcoinem) byla vybrána kvůli tomu, že se neočekává moc vysoká náročnost na provoz blockchainu, což znamená, že by se větší bloky musely stavět potenciálně velmi dlouho, zatímco práce s menšími bloky by byla z hlediska správy blockchainu v případě menší sítě robustnější. V rámci PBFT se bude ověřovat každý záznam, neboť ověřování záznamu je spojeno s povolením přístupu do bezpečnostního objektu, čímž se vytváří nároky na rychlost ověření. Povolení přístupu do budovy bez ověření či s ověřením pouze jedním porovnávačem by mohlo představovat bezpečnostní riziko (kvůli výše zmíněnému útoku na porovnávač), a zjištění nepovoleného přístupu by v případě ověřování celých bloků mohlo trvat desítky minut až hodiny (záleží, jaký by se vybral block time, krátký block time by mohl vytvářet velké množství prázdných či málo obsazených bloků, což by prodlužovalo vyhledávání a verifikaci, zatímco dlouhý block time by znamenal, že reálné ověření uživatele by mohlo trvat dlouhou dobu). Ověřování každého záznamu umožňuje spojit decentralizované ověřování uživatele s povolením přístupu uživatele k objektu v teoreticky rozumném čase. Ověřování každého záznamu zvyšuje náročnost na komunikaci mezi porovnávacími uzly. Tato vlastnost může vytvářet problém u větších sítí s vysokým provozem. U malé sítě v kombinaci s menším provozem by dané řešení nemělo činit nepřiměřeně velký problém z hlediska časové odezvy na uživatelův požadavek k přístupu do objektu, a to ani přesto, že je tento princip kombinován s PBFT (náročný protokol z hlediska množství zaslaných zpráv).

Blok se bude přidávat s vyčerpáním kapacity bloku zejména kvůli výše zmiňovanému ověřování každého záznamu v rámci consensus protokolu. Vytváření block timů a problémy ohledně jejího dodržování a synchronizace blockchainů u individuálních porovnávačů by naprosto zbytečně komplikoval protokol, pokud se stejně ověří každý záznam. Navíc kvůli iregularitě využívání objektu by block time vytvářel velké problémy v rámci blockchainové struktury (vytvářelo by se velké množství prázdných bloků, zatímco u špiček provozu by daný block time nestačil a protokol by musel přejít k přidávání a ověřování bloků po jejím naplnění). Z těchto důvodů se blok bude přídávat do struktury po naplnění kapacity a nebude se samostatně ověřovat (integrita tohoto bloku se bude ověřovat při ověřování individuálních záznamů).

Timestampování komunikace a symetrické šifrování zpráv mezi účastníky systému nebude implementován. Je to zejména z toho důvodu, že analýza man-in-the-middle a replay

útoků není náplní této studie – jsou tak opomíjeny kvůli zachování jednoduchosti implementace. V reálném autentizačním systému ale tyto praktiky musí být použity.

Kapitola 4

Implementace

4.1 Výběr jazyka a knihoven

K implementaci systému byl využit jazyk Python. Jazyk Python byl vybrán především kvůli snadné a přehledné implementaci programů v něm vytvořených. Python také nabízí různé knihovny pro sledování časové a paměťové náročnosti prováděného kódu, společně s jejím profilováním. To usnadní testování funkčnosti systému a měření některých jejích požadovaných vlastností, jako je paměťová náročnost, doba odezvy, apod. Hlavní nevýhodou jazyka Python je, že je vysoce abstrahovaný a tím pádem konstrukce naprogramované v tomto jazyce nejsou plně optimalizované. Tato nevýhoda by ale neměla být velkou překážkou, neboť úkolem této práce není vytvořit plně optimalizovaný systém, ale vytvořit proof-of-concept autentizačního systému, tj. odzkoušet autentizační systém z bezpečnostního hlediska, zejména odolnost proti vybraným útokům, a přibližně zjistit výkonnostní nároky na systém a časovou složitost operací prováděných systémem.

Pro implementaci systému byly využity primárně následující knihovny:

- `hashlib` – Slouží k hashování řetězců.
- `datetime` – Slouží k vytváření timestampů pro účely získávání a řazení záznamů v blockchainu a ukládání do databáze. Podle timestampu se vyhledávají záznamy v blockchainu a identifikují se vstupy do systému.
- `mysql.connector` – Slouží k připojení k SQL databázi a k práci s ní. Tuto knihovnu využívá především manažer databáze.
- `cryptography` – Slouží k zajištění funkcí spojené s kryptografií – zejména se jedná o podepisování záznamů a zpráv a ověřování těchto podpisů.
- `random` – Bude primárně sloužit k náhodnému výběru senzorů či registrátorů pro účely testování.
- `multiprocessing` – Zajišťuje paralelní běh zařízení v rámci jednoho systému. Bude se využívat struktury fronta, zejména kvůli jednoduché práci s ní. Více informací ohledně multiprocessingu viz kapitola 4.4.
- `time` – Z této knihovny se bude využívat funkce `sleep`, která uspí proces na určitý úsek času. Bude sloužit společně s funkcí knihovny multiprocessing `get` k zajištění pasivního čekání procesu v určitých situacích (například čekání na signál od jiného

procesu, uspání testovacího procesu na určitou délku času za účelem simulování časového rozložení provozu objektu apod.). Také se z této knihovny využívá funkce `time.time` na zjištění momentálního času pro účely testovacího programu

- `openCV-python (cv2)` – Využívá se zde primárně ukládání a načítání souborů s otisky prstů, extrakci markantů z těchto obrázků a jejich porovnávání
- `sys` se používá pro ukončování testovacího programu při chybě vstupních parametrů.
- `os` se používá k práci se soubory s otisky prstů.
- `getopt` – Zpracovávání vstupních parametrů
- `names` – Používá se ke generaci náhodných jmen a příjmení
- `memory_profiler` – Používá se k profilování paměťové náročnosti systému.

4.2 Databáze

V rámci systému se využívá mySQL databáze. K databázi se program připojuje za pomocí funkce `mysql.connector.connect` na uživatelské jméno `user` s heslem `sample-password` na localhostu. Konkrétně se připojuje do databáze pojmenované `mydatabase`. Databáze se upravuje nastavením kurzoru na kurzor databázového připojení, provedení databázového dotazu s ním, potvrzením změn za pomocí příkazu `commit` a zavřením kurzoru. Databáze využíva autorizační plugin `mysql_native_password`.

4.3 Implementace blockchainu a merklovského stromu

K implementaci blockchainu a merklovského stromu nebyly použity jakékoliv knihovny. U dvou individuálních struktur je tak z různých důvodů. U blockchainu samotného se jedná o to, že z hlediska našeho systému nemusí blockchain vykonávat velké množství funkcí, a importováním knihovny pro práci s ní by, společně s obeznámením s knihovnou a naučení se práce s ní, znamenalo velké vynaložení času, který by se dal využít k vypracování těchto funkcí samostatně. U knihoven implementujících merklovský strom se objevil jiný problém, a to takový, že existující knihovny pro merklovský strom buď nebyly dostatečně zdokumentované, nebo byly nedostatečné z hlediska potřeb našeho systému (například jedna z těchto knihoven ukládala do individuálních listů stromu pouze kryptografické hashe, zatímco pro účely našeho systému bylo potřeba ukládat celé záznamy).

Merklovský strom je reprezentován objektem `MerkleTree`. Struktura je složena ze dvou typů uzlů. Prvním typem je uzel `LeafNode`, který obsahuje informace o záznamu samotném (objekt `Record`, který obsahuje timestamp, předešlý timestamp (odkazující se tak na jiné záznamy), typ operace a hash informací o uživatelovi). V rámci struktury je `LeafNode` uložen jako pole, které se vytváří při inicializaci objektu a postupně se do něj přidávají prvky. Druhým typem je uzel `HashNode`, který hashuje konkatenované hashe od pravého a levého potomka a ukládá si je. `LeafNode` má jednoduché metody, především gettery a settery. `HashNode` obsahuje metodu `Rehash`, která zahashuje oba konkatenované hashe svých potomků a iterativně přepočítá hashe svých předků. Oba typy uzlů dědí metody od třídy `Node`. Mezi tyto metody patří settery a gettery ukazatelů na potomky a na předka uzlu.

Merklovský strom má tři důležité metody. První metodou je přidání záznamu do struktury `add_leaf`. Tato funkce appenduje pole listových uzlů a modifikuje struktury tak, aby byla zajištěna konzistence stromové struktury. Pokud je množství listových uzlů po přidání sudé, tak se pouze nastaví nově přidaný uzel jako potomek hashového uzlu a zavolá se nad tímto uzlem funkce `Rehash`. Pokud je množství listových uzlů liché, najde se nejpravější uzel bez pravého potomka a naváže se na něj nový uzel. Během tohoto procesu se přidávají další hashové uzly tak, aby byla stromová struktura konzistentní a spojitá. Pokud neexistuje hashový uzel bez pravého potomka, tak se vytvoří nový kořenový uzel a listový uzel se přes hashové uzly naváže na pravý slot tohoto uzlu.

Druhou metodou je metoda `Verify`, která na základě indexu vyhledá specifický listový záznam v merklovském stromu. Pro každého předka tohoto uzlu je pak provedeno hashovaní dvou hashů od potomků a výsledek se porovnává s hodnotou uloženou v uzlu. Pokud se neshodují, vrátí metoda chybu. Třetí metodou je vyhledávání záznamů podle timestampu, který sekvenčně prohledává pole listových uzlů a vrací všechny listové uzly, které mají stejný timestamp.

Blockchain je reprezentován objektem `Block`, který obsahuje ukazatel na předešlý blok, timestamp, kapacitu bloku, počet záznamů v bloku, merklovský strom se záznamy hashových informací uloženém v bloku a hash předešlého bloku. Přidání záznamu se provádí přidáním záznamu do merklovského stromu a ověřením, jestli se blok nenaplnil (pokud ano, vrátí se chyba, kterou volající funkce obslouží). Po ověření se aktualizuje timestamp posledního záznamu. Vyhledávání podle timestampu funguje na základě vyhledání prvního bloku se starším nebo stejným timestampem než blok předchozí. U předchůdce tohoto bloku se pak začne vyhledávat podle timestampu v příslušném merklovském stromu. Pokud se najde více bloků se stejným timestampem, tak se prohledají merklovské stromy u více bloků.

4.4 Multiprocessing

Víceprocesové fungování systému zajišťuje knihovna `multiprocessing`. Pro každé zařízení (senzor, registrátor, komparátor, manažer databáze a otevírač dveří) se vytváří samostatný proces. Procesy se budou inicializovat v testovacím programu. Procesy se aktivují voláním metody `run`. Metoda `run` obsahuje nekonečnou smyčku, na jejímž začátku pasivně čeká na vstup od jiných procesů. U senzorů se bude jednat o vstup od testovacího programu, u komparátorů se bude jednat o vstup od senzorů či jiných komparátorů a u správce databaze a otevíračů dveří se bude jednat o vstup z komparátorů. Komunikace mezi procesy se bude vykonávat za pomocí front. Každý proces má svou frontu, na které naslouchá za pomocí metody `get`. Tuto frontu sdílí s procesy, se kterými může komunikovat. Tyto procesy zasílají cílovému procesu zprávy za pomocí metody `put`. Fronta byla vybrána z toho důvodu, že se u ní nemusí řešit problémy se synchronizací (např. data race).

4.5 Senzory, správce databáze, otevírač dveří

Senzory zpracovávají vstupní informace, převádějí je do vhodné podoby a předávají je porovnávačům k autentizaci. Program obsahuje dva typy senzorů. Běžné senzory (`Sensor`), které umí zaslat jenom požadavky na verifikaci, a registrátory (`Registrator`), které dědí metody od běžného senzoru a který umí zasílat i požadavky na odstranění či přidání uživatele do databáze. Při získání signálu z testovacího programu zpracuje senzor informace a pošle signál všem komparátorům, kde uvede informace o uživatelovi, typ operace (běžný

senzor umí jen verifikaci, registrátoři umí i více operací), bezpečnostní úroveň, timestamp získaný z testovacího programu, ID senzoru a obrázek otisku prstu.

Správce databáze je objekt, který na popud porovnávačů provádí změny v databázi. Signál k činnosti získává od porovnávačů. Pokud informace o uživatelovi s daným timestampem nejsou již u správce databáze uložené, tak vytvoří nový slovník (`user_info_object`, do kterého se uloží informace o uživatelovi, timestamp, operace (vše pocházející od zprávy z porovnávače) a signály (jedná se o list veřejných klíčů, které souhlasí s provedením operace nad databází). Správce databáze verifikuje za pomocí dodaného veřejného klíče zprávu a přiřazený záznam. Při úspěšné verifikaci přidá veřejný klíč do signálů, pokud se jedná o jeden ze známých veřejných klíčů a pokud již není veřejný klíč uložen. Pokud množství uložených veřejných klíčů překročí dvě třetiny celkového počtu komparátorů, provede správce databáze operaci s databází podle typu operace uvedené ve zprávě. V případě verifikace upraví správce databáze položku `last_operation` na timestamp uvedený ve zprávě, u registrace přidá správce záznam o danému uživatelovi a u odstranění správce odstraní položku s uživatelem z databáze.

Otevírač dveří je proces který vyhodnocuje zprávy pocházející z komparátorů a „otevírá dveře“ (vypíše na výstup, že specifické dveře byly otevřeny). Funguje v principu stejně jako správce databáze, akorát místo práce s databází provede již zmíněný výpis.

4.6 Komparátor

Komparátor na popud zpráv od senzorů zpracovává uživatelské údaje a za pomocí consensus protokolu je potvrzuje. Mezi jeho atributy patří ID, proměnná určující, jestli se komparátor považuje za vedoucí uzel, blockchain, fronta, pomocí které přijímá od senzorů zprávy, fronta sdílená s ostatními komparátory, fronty sdílené s otevíracími mechanismy a se správcem databáze, počet připojených komparátorů, množina uživatelských informací, ve kterých porovnávač vyhledává timestamp a uživatelské informace z dané zprávy, množina uživatelských informací, které ještě nebyly nalezeny, ID komparátoru, který daný komparátor považuje za vedoucí, množina hlasů pro změnu vedoucího komparátoru, seznam známých veřejných klíčů, doba posledního vyčištění přídavných datových struktur, doba poslední změny vedoucího uzlu, jestli se považuje uzel za nepřátelský a celkový počet známých komparátorů.

Komparátor má mimo metodu `run` šest důležitých dílčích funkcí. První metodou je `add_record`, který přidá záznam do blockchainu. Pokud je blok v rámci blockchainu plný, vytvoří se nový blok, nastaví se na něj ukazatel pro předešlý blok, přidá se do něj záznam a ukazetel bloku pro komparátor se nastaví na nově vzniklý blok. `Fetch_record` provede dotaz výběru záznamů na základě hashe informací na kartě a vrátí první záznam z odpovědí na dotaz. Pokud žádný záznam není nalezen, vrátí se chyba. `Verify_user` zajistí, že informace získané z databáze korespondují k uživatelským informacím na vstupu. Testuje se, zda doba expirace není starší než momentální doba, zda uživatel má dostatečnou bezpečnostní úroveň pro dané dveře, a zda se zbytek informací zhoduje s uživatelskými informacemi na vstupu. Pokud tak není, vrátí se chyba. `Verify_history` ověřuje v blockchainu, zda jsou uložené záznamy v blockchainu korespondující k uživatelovi konzistentní se záznamem získaném z databáze. Zejména se jedná o situace, kdy je nalezen ostraňovací záznam, pokud záznam, která má timestampový ukazatel na jiný záznam, neobsahuje záznam se stejnými uživatelskými informacemi, nebo pokud se nenajde registrovací záznam pro uživatele uloženého v databázi. Pokud se najde nekonzistence v blockchainu nebo uživatel není v blockchainu vůbec uložen, vrátí metoda chybu.

Metoda `verify_fingerprint` je popsána v kapitole 4.8. Metody `sign_message` a `verify_message` jsou popsány v kapitole 4.7.

Metoda `run` čeká na frontě `conn` na zprávu od senzorů. Každý komparátor má u čekání timeout, po jejimž vypršení se inicializuje změna vedoucího uzlu v rámci consensus protokolu. Případně se změna vedoucího uzlu provede po 30 sekundách běhu komparátoru. Každá zpráva je složena z informací o uživatelovi, prováděné operaci, úrovni senzoru, timestampu, ID senzoru, záznamu veřejného klíče, obrázku otisku prstu a podpisu zprávy. Metoda provádí consensus protokol PBFT. V případě, že se v rámci vyslané zprávy jedná o operaci `Request` (v programu se jedná o operace `Verify`, `Register` a `Delete`), se vytvoří `user_info_object`, obsahující informace o uživatelovi, operaci, která se provedla, timestampu a počtu signálů, a připojí se k frontě `user_info_queue`. Tato struktura se bude v pozdějších fázích protokolu používat k potvrzení, že komparátor již dostal předešlé zprávy ohledně uživatelových informací s daným timestampem, a také se bude využívat ke sledování množství signálů k určení, jestli bylo dosáhnuto prahu k zaslání určitých zpráv. Co se přesně provede při ověření uživatelova požadavku záleží na specifické operaci (registrace, verifikace či odstranění). U verifikace a odstranění se pomocí metody `fetch_record` získá záznam z databáze. Poté se pomocí metod `verify_user`, `verify_history` a `verify_fingerprint` provede samotná verifikace. U registrace proces jenom vytvoří nový záznam. Při úspěšném provedení těchto operací pak vždy vedoucí komparátor rozešle `Pre-prepare` zprávu ostatním komparátorům

U zpráv `Pre-prepare`, `Prepare` a `Commit` se provede prohledávání fronty s uživatelskými informacemi. Pokud se najdou položky korespondující k danému uživatelovi a timestampu uvedeném ve zprávě, přidá se daný `user_info_object` do pole a dále se pracuje s první položkou tohoto pole. Pokud se nenajde žádná položka korespondující k danému uživatelovi a timestampu, prohledá metoda pole `not_found`, které obsahuje informace o uživatelích a operacích, na jejichž zprávy proces ještě čeká. Pokud není objekt korespondující s uživatelem uvedeném ve zprávě nalezen, vytvoří se nový. Proces poté počká určitou délku času, po jejímž vypršení zašle sám sobě stejnou zprávu. U každého objektu `not_found` je ještě zahrnutý atribut `ttd`, který začíná na nastavené hodnotě a při každé iteraci výše zmíněné procedury se dekrementuje. Při dosažení hodnoty 0 už proces sám sobě nepřeposílá zprávu. `Ttd` v proceduře existuje proto, aby nedošlo k nekonečnému přeposílání zprávy ze strany komparátoru při nějaké přenosové chybě.

Pre-prepare fáze komunikace slouží především k ověření, že informace poslané od vedoucího uzlu nejsou zfalšované (tj. shodují se s informacemi získané ze senzoru). Komparátor verifikuje podpis zprávy a podpis záznamu. Pokud je autenticita informací z vedoucího uzlu potvrzena, rozešlou nevedoucí uzly `Prepare` zprávu ostatním komparátorům.

Prepare fáze slouží k opětovné verifikaci uživatele a jeho historie v rámci blockchainu a k zajištění, že dostal zprávy od více jak dvou třetin nevedoucích uzlů potvrzující autenticitu zprávy od vedoucího uzlu. Pro provedení operací v prepare fázi je potřeba dostat k tomu popud aspoň od dvou třetin ostatních komparátorů. Po verifikaci podpisu zprávy a podpisu přiřazeného záznamu se provedou operace pro verifikaci uživatele popsané v `Request` fázi při získání popudu k této akci od více jak 2/3 komparátorů. Po úspěšné verifikaci se pak pošle všem ostatním komparátorům `Commit` zpráva.

Commit fáze slouží k předání zpráv otevíračům dveří a správci databáze. Pokud komparátor dostane commit zprávy od více jak 2/3 komparátorů, zašle správci databáze zprávu s příslušnou operací, kterou správce nad databází provede. V případě `Verify` ještě tuto zprávu zašle příslušnému otevírači dveří.

Pokud dojde k vypršení timeoutu při čekání na zprávu na začátku metody `run`, nebo pokud od poslední změny kontextu vyprší lhůta 30 sekund, zašle komparátor vedoucímu komparátoru a jeho následníkovi `Change View` zprávu a změní ID komparátoru, kterého považuje za vedoucího. Vedoucí a následnický komparátor, který dostane tento signál si pak ukládá hlasy a pokud jich dostane více než dvě třetiny, změní daný komparátor svůj vedoucí status (vedoucí se nebude považovat za vedoucího, zatímco následnický komparátor se začne považovat za vedoucího) a změní ID komparátoru, který považuje za vedoucí, na ID následovnického komparátoru.

Nefunkční komparátory jsou komparátory, které v metodě `run` jenom pasivně čekají. Nepřátelské komparátory místo ukončení zpracovávání zprávy při neúspěšné verifikaci pokračují dále. Funkční a nefunkční komparátory nemají u sebe uloženy veřejné klíče nepřátelských komparátorů, zatímco nepřátelské komparátory mají uložené veřejné klíče všech komparátorů.

4.7 Kryptografie

Kód pro generaci klíčů byl převzat z webové stránky nitratine [1]

K verifikaci záznamu se používá knihovna `cryptography`, zejména funkce pro podpis a pro verifikaci podpisu. Každý komparátor si generuje dvojici klíčů (soukromý a veřejný klíč), pomocí kterých podepisuje a verifikuje zprávy. Generace klíčů se děje ve funkcí `generate_keys`. V této funkci se vygeneruje soukromý RSA klíč s exponentem 65537 a velikostí 2048 bitů. Tyto klíče jsou poté serializovány. Podpis zprávy se děje s pomocí deserializovaného soukromého klíče s defaultním backendem. Poté se tento klíč aplikuje na hash informací uvedených ve zprávě. Zpráva je podepsaná za pomocí podpisového schématu RSA-PSS s MGF1 paddingem ve funkci knihovny `cryptography` `privatekey.sign`. Verifikace se provádí po zahashování informací ve zprávě pomocí funkce `publickey.verify`.

Podepisování a verifikace se aplikují na dva typy struktur. Jedním z těchto struktur je záznam. Podepisuje se a verifikuje hash všech informací uvedených v záznamu. Druhou strukturou je zpráva posílaná mezi komparátory. Zde se podepisují a verifikují všechny elementy zprávy kromě klíče a záznamu. Obrázek se pomocí knihovny openCV zakóduje. K verifikaci zprávy i záznamu dojde při každé příchozí zprávě, k podepisování zprávy dojde vždy před zasláním zprávy ostatním komparátorům a k podepisování záznamu dojde před zasláním zprávy ostatním komparátorům, otevíračům dveří či správci databáze.

4.8 Biometrika

Kód funkce pro ověřování otisku prstu byl převzat z videa „Fingerprint Matching in Python“ [3] a přizpůsoben potřebám implementovaného systému.

Biometrické informace se ověřují za pomocí knihovny `openCV-python`. Pro účely tohoto systému byly vybrány otisky prstů díky snadné a rychlé implementaci. Obrázek otisku prstu se načte v testovacím programu z příslušného souboru a přepošle se senzoru. Ten obrázek přepošle komparátorům, které s ním v rámci PBFT protokolu pracují. Během `Request` fáze a `Prepare` fáze PBFT protokolu se provádí pro verifikaci a odstranění uživatele ověřování otisku prstu v metodě `verify_fingerprint` V této metodě se dokládá obrázek s otiskem prstu, který má být verifikován, a obrázek s otiskem prstu získaném z databáze (zde reprezentované složkou `db`, kde příslušný soubor má jméno stejné, jako je hash uživatelových informací).

Ve funkci se vyextrahují keypointy a deskriptory pro otisky prstů za pomocí funkcí `SIFT_create` a `detectAndCompute` a to u obou obrázků. Poté se zavolá funkce `FlannBasedMatcher`, který obsahuje sbírku algoritmů optimalizovanou pro rychlé vyhledávání nejbližších sousedů. Jako algoritmus je vybrán `FLANN_INDEX_KDTREE`, který využívá K-D stromů. V rámci funkce bude využito 10 paralelních stromů. Poté je vygenerováno k nebližších sousedů (v našem případě je k nastaveno na hodnotu 2 pro každý deskriptor vygenerovaný předešlou funkcí. Na základě vzdálenosti od nejbližších sousedů se určuje, jestli daný bod je v rámci porovnávání relevantní. Dále se na základě vyextrahovaných bodů z funkce `detectAndCompute` získá počet klíčových bodů. Poté se na základě podílu relevantních bodů z celkových klíčových bodů získá skóre (v procentech). Pokud je skóre menší než 50, vyvolá se verifikační chyba.

Kapitola 5

Experimentální fáze

5.1 Testovací program

5.1.1 Použití

Testovací program se spouští z příkazového řádku s následujícími parametry:

- `-h` nebo `--help` – vypíše návod k používání programu.
- `-s (počet)` nebo `--sensors=(počet)` – nastaví počet běžných (neregistrujících) senzorů na hodnotu uvedenou v položce `(počet)`. Při chybějícím parametru je počet sensorů nastaven na 4.
- `-r (počet)` nebo `--registrators=(počet)` – nastaví počet registrátorů na hodnotu uvedenou v položce `(počet)`. K fungování testovacího programu musí existovat aspoň jeden registrátor, aby mohl zaregistrovat nové uživatele. Při chybějícím parametru je počet registrátorů nastaven na 1.
- `-c (počet)` nebo `--comparators=(počet)` – nastaví počet běžných (korektně fungujících) komparátorů na hodnotu uvedenou v položce `(počet)`. Při chybějícím parametru je počet komparátorů nastaven na 10.
- `-m (počet)` nebo `--malicious=(počet)` – nastaví počet nepřátelských (snažících se aktivně odmítnout korektní vstupy) komparátorů na hodnotu uvedenou v položce `(počet)`. Při chybějícím parametru je počet nepřátelských komparátorů nastaven na 0.
- `-n (počet)` nebo `--nonfunctional_comparators=(počet)` – nastaví počet nefunkčních (neodpovídajících na vstup) komparátorů na hodnotu uvedenou v položce `(počet)`. Při chybějícím parametru je počet nefunkčních komparátorů nastaven na 0.
- `-u (počet)` nebo `--users=(počet)` – nastaví počet uživatelů, kteří se v úvodní fázi testovacího programu zaregistrují do databáze. Kvůli limitované kapacitě testovacích otisků prstů jejich počet nesmí překročit 500. Při chybějícím parametru je počet uživatelů nastaven na 20.
- `-d (počet)` nebo `--duration=(počet)` – nastaví dobu, po kterou bude trvat verifikační fáze testovacího programu. Údaj je uveden v minutách. Při chybějícím parametru je doba trvání nastavena na 5 minut.

- `-i (počet)` nebo `--interval=(počet)` – nastaví interval (v sekundách) mezi individuálními vstupy verifikační fáze testovacího programu. Při chybějícím parametru je interval nastaven na 5 sekund.

- `-w (počet)` nebo `--chance_wrong_input=(počet)` – nastaví šanci na generaci vstupu, který se bude snažit identifikovat za pomocí neplatného otisku prstu. Údaj je uveden v procentech. Nesmí překročit hodnotu 100. Při chybějícím parametru je šance nastavena na 10 %.

- `-e (počet)` nebo `--chance_illegal_entry=(počet)` – nastaví šanci na generaci vstupu od uživatele, který je v databázi přidán bez registrace přes komparátory. Údaj je uveden v procentech. `--chance_wrong_input=(počet)` a `--chance_illegal_entry=(počet)` se vyhodnocují kumulativně a jejich celková hodnota nesmí překročit 100 procent. Při chybějícím parametru je šance nastavena na 10 %.

K správnému fungování programu je potřeba mít správně stažený dataset otisků prstů. Dataset použitý v testovacím programu byl stažený ze Sokoto Coventry datasetu[7][25], který obsahuje otisky prstů od celkem 600 afrických dospělých lidí. Obsahuje jak neupravené referenční otisky prstů tak i zkreslené otisky (například lokální rotace, z-cut nebo lokální snížení rozlišení). Tento dataset musí být v rámci programu rozložen správně, aby mohl fungovat. Ve složce `to-be-added` jsou obsaženy neupravené otisky prstů. Z této složky se za pomocí registrující operace budou ukládat soubory do složky `db`. Ve složkách `input` a `fake-input` pak jsou uloženy upravené otisky prstů. Ve složce `input` jsou uloženy otisky prstů identifikovaných čísly 1-500. Bude se jednat o otisky prstů, pomocí kterých se budou uživatelé identifikovat korektně. Ve složce `fake-input` jsou uloženy otisky prstů identifikovaných čísly 501-600. V daném případě se bude jednat o nekorektní otisky prstů, které má systém odmítnout. Testovací program bude pracovat jen s otisky palce pravé ruky. K převodu původního datasetu na dataset požadovaný testovacím programem slouží skript `script.py`.

V rámci testovacího programu se bude ještě využívat měření paměťové náročnosti celého systému. K získání informací o paměťovém vytížení systému se musí nainstalovat knihovna `memory_profiler`. K spuštění programu se sledováním paměťového vytížení se program musí spustit za pomocí příkazu `mprof run -C python <program>`, kde `-C` je parametr určující, že se má měřit celková paměťová náročnost hlavního procesu společně s potomkovými procesy. V rámci knihovny `memory_profiler` lze i procesy monitorovat individuálně, ale nejde v něm filtrovat individuální procesy a tak případný graf bude silně nepřehledný. Výsledky monitorovaní se poté zobrazí za pomocí příkazu `mprof plot`. Je potřeba vzít v potaz, že `memory_profiler` ve velké míře snižuje celkovou výkonnost a rychlost programu, což může přídavně ovlivnit konečné výsledky. Je proto doporučené spouštět profiler jen pro účely měření paměťové náročnosti systému a během jejího testování nehledět na ostatní výsledky programu.

5.1.2 Fungování testovacího programu

Testovací program získá za pomocí `getopts` vstupní parametry a případně změní výchozí hodnoty na hodnoty uvedené v parametrech. V případě nekorektně uvedených informací vypíše testovací program příslušnou chybovou zprávu. Poté dojde k přidání jednoho uživatele do databáze přímo bez využití komparátorů společně s přesunutím souboru s otiskem prstu do složky `db`. Dalším krokem je inicializace procesů. Vytvoří se komunikační fronty

mezi procesy a pak se vytvoří procesy pro dveře, komparátory, sensory, registrátory a pro manažer databáze společně s příslušnými komunikačními frontami. Dále jsou uloženy pro komparátory, manažer databáze a pro otevírače dveří veřejné klíče komparátorů, které tyto procesy budou akceptovat jako „hlas“ pro určování, jestli postupovat ve zpracovávání příslušné zprávy dál. Čestné procesy nebudou mít u sebe uložené veřejné klíče nepřátelských uzlů, zatímco nepřátelské uzly budou mít uložené klíče všechny. Po uložení klíčů proběhne zapnutí individuálních procesů.

Další fází (tzv. úvodní či registrační) fází testovacího programu je úvodní registrace uživatelů do databáze přes blockchain. Registrace uživatelů probíhají s meziintervalem 5 sekund. Je tak učiněno především k účelům nezahlcení front procesů a zajištění, že se individuální uživatelé přidají do databáze. Registrace proběhne v množství určeném parametrem `--users` a otisky prstů pro účely úvodní registrace se získávají ze složky `to-be-added` sekvenčně podle ID subjekta otisku prstů. Po poslední registraci program počká 5 sekund na dokončení registrací přes blockchain. Znova se tak činí kvůli bezpečnosti.

Další fází programu je testovací (nebo verifikační) fáze. Tato fáze trvá po dobu uvedenou v parametru `--duration`. Pro každou verifikaci je vygenerováno náhodné číslo v intervalu od 0 do 1. Pokud je vygenerované číslo menší než setina čísla uvedeného v parametru `--chance_wrong_input`, provede se verifikace pro špatný otisk prstů nad náhodným sensorem. Tato verifikace vytáhne náhodný otisk pravého palce ze složky `fake-input` a předloží ho legitimně zaregistrovanému uživatelovi. Pokud je vygenerované číslo menší než setina součtu `--chance_wrong_input` a `--chance_illegal_entry` a větší než setina
`--chance_wrong_input`, je provedena verifikace pro nekorektně vložený záznam v databázi. Pro účely verifikace je použit soubor s ID 600 ze složky `fake-input`, jehož nepoškozená verze byla uložena do složky `db` bez využití komparátorů a bez uložení záznamu do blockchainu. Pro ostatní náhodně vygenerované hodnoty se provádí verifikace pro korektně zaregistrované uživatele se správným otiskem prstu nad náhodným senzorem. Otisky prstů se budou získávat v daném případě ze složky `input`.

Testovací program je napojen na otevírače dveří za pomocí fronty `test_queue`. Otevírač dveří při získání dostatečného množství signálů k otevření dveří zašle testovacímu programu zprávu, kde je uveden počáteční timestamp transakce a momentální čas otevření samotných dveří. Před zasláním transakce senzoru testovací program do pole `test_vector` uloží informace o počátečním timestampu transakce, momentálním času a o očekávaném výsledku (false pro nekorektní vstupy, true pro korektní). Po ukončení verifikační fáze pak program z fronty `test_queue` získává informace od otevíračů dveří o úspěšných otevření dveří. Program si podle počátečního timestampu vyhledá, ke kterému záznamu pole `test_vector` informace patří a uloží do něj položku momentálního času otevírání dveří. Po uložení všech položek vyhodnotí na základě položky o očekávaném výsledku a výskytu položky času otevírání dveří, jestli operace byla provedena korektně, či ne. Korektnost provedených operací pak vyjádří jako procento z celkového počtu verifikačních operací. Pro operace, u kterých se dveře otevřely, pak testovací program spočítá rozdíl mezi časem vytvoření vstupu do systému a otevřením dveří. Tyto rozdíly se poté zprůměrují. Po výpočtu procenta úspěšně provedených operací a odezvy a jejích výpisu na standardní výstup pak program ukončí všechny procesy za pomocí příkazu `process.terminate()`

5.2 Prováděné testy

Testy budou prováděny s výchozími hodnotami (zmíněné v kapitole 5.1.1 s rozdílem jednoho nebo dvou parametrů, jejichž účelem bude testovat určitý aspekt systému. Verifikační fáze každého testu bude trvat 5 minut, aby každý uzel mohl být v určité fázi testování vedoucím uzlem (vyjímky budou testy pro 15 a 20 komparátorů, kde nebudou nepřátelské či nefungující uzly. Dobá trvání těchto dvou testů bude také 5 minut). Testy budou prováděné na Windows subsystem for Linux (WSL). Testy, které budou nad systémem prováděny, jsou následující:

- Testy různých intervalů vstupů – budou testovány nad hodnotami 2, 5, 10 a 20 sekund. Účelem testu bude zjistit, jestli dokáže systém čelit špičkám uživatelských vstupů. Test pro hodnotu 5 sekund bude také sloužit jako referenční test pro ostatní testy
- Testy různého množství uživatelů – budou testovány nad hodnotami 10, 50, 100 a 200. Účelem testu bude zjistit, jestli zvládne systém zajistit otevírání dveří pro velké množství zaregistrovaných uživatelů, společně s testováním odezvy otevírání těchto dveří
- Testy různého množství komparátorů – budou testovány nad hodnotami 5, 10, 20 a 30. Účelem testu bude zjistit, jaký dopad bude mít zvýšený počet komparátorů na fungování systému, především na odezvu.
- Testy různého množství nefunkčních komparátorů – budou testovány nad hodnotami 1, 2, 3 a 4. Celkový počet komparátorů se bude vždy rovnat 10. Účelem testu bude zjistit, jestli systém funguje i s nefunkčními komparátory.
- Testy různého množství nepřátelských komparátorů – budou testovány nad hodnotami 1, 2, 3 a 4. Celkový počet komparátorů se bude vždy rovnat 10. Účelem testu bude zjistit, jestli systém funguje i s nepřátelskými komparátory.

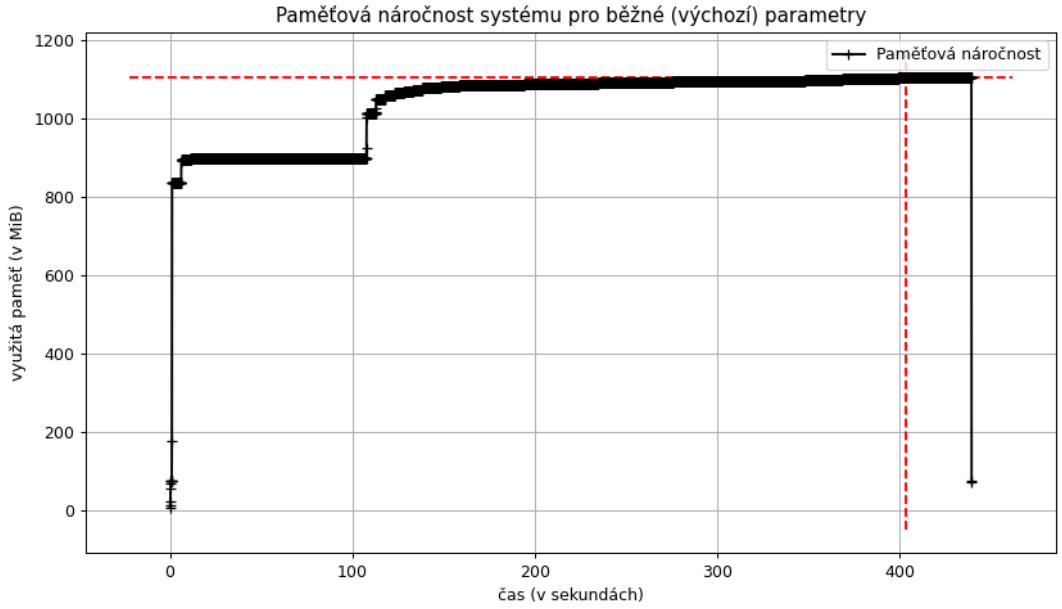
Testy na měření paměťové náročnosti systému se budou vykonávat nezávisle na výše zmíněných testech. Je to z toho důvodu, že `memory_profiler` zpomaluje vykonávání samotného programu, a tak může ovlivňovat konečné výsledky. Testy na měření paměťové náročnosti se budou vykonávat v následujících scénářích:

- Test pro výchozí hodnoty (viz kapitola 5.1.1)
- Test pro interval vstupu 2 sekund
- Test pro 200 uživatelů
- Test pro 30 komparátorů
- Test pro 3 nefunkční komparátory
- Test pro 3 nepřátelské komparátory

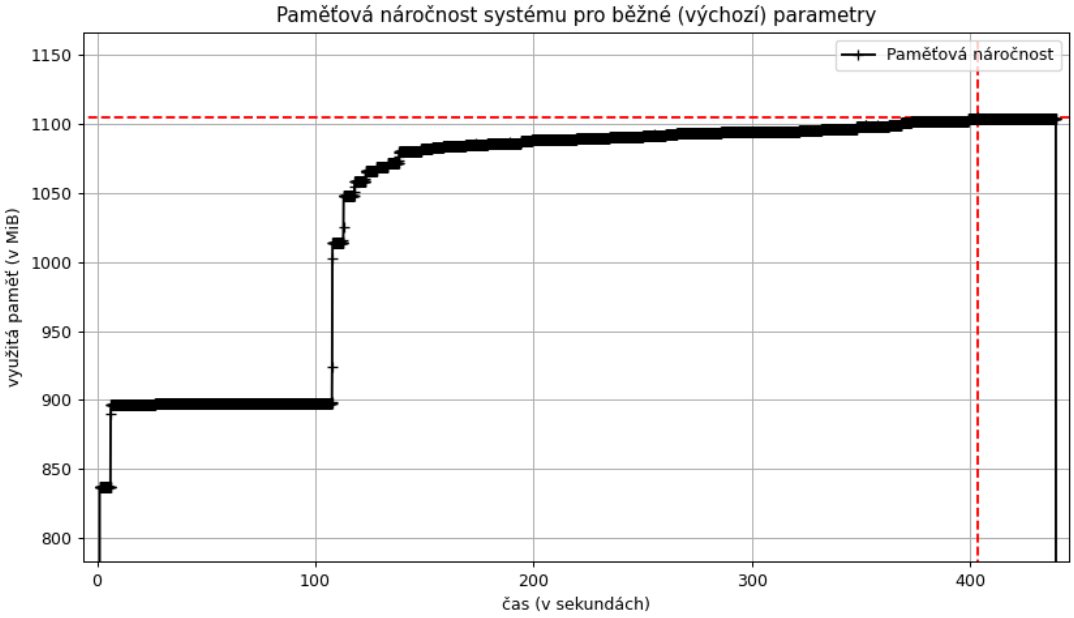
5.3 Výsledky testů

Interval vstupu (s)	Průměrná odezva (s)	Úspěšnost operací (v %)
2	0,21	97,2
5	0,17	94,92
10	0,20	100
20	0,21	93,33

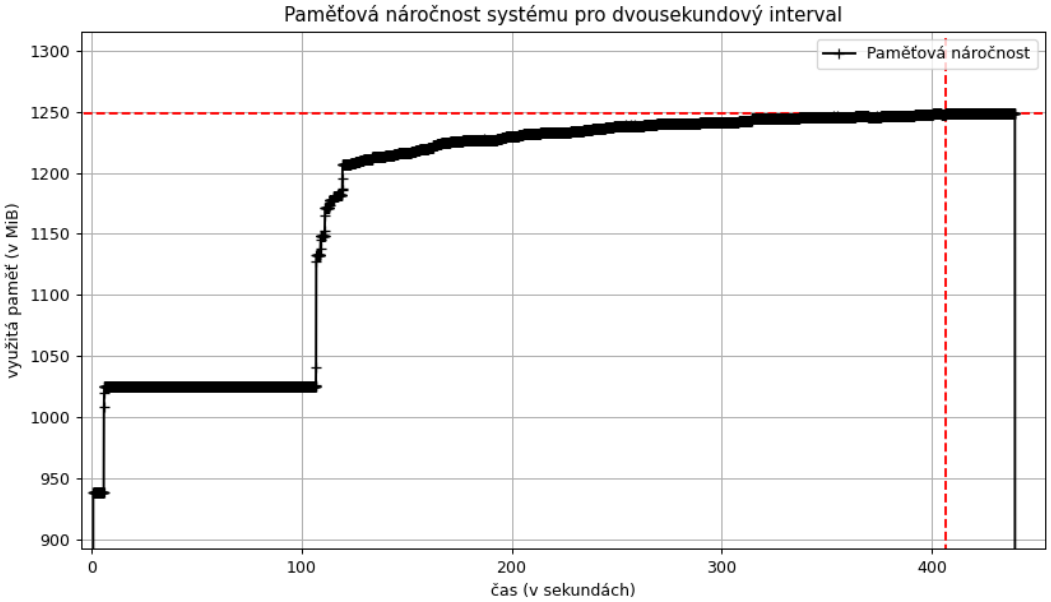
Tabulka 5.1: Odezva a úspěšnost verifikačních operací pro různé intervaly vstupu



Obrázek 5.1: Paměťová náročnost systému pro běžné vstupy



Obrázek 5.2: Paměťová náročnost systému pro běžné vstupy (po inicializaci procesů)



Obrázek 5.3: Paměťová náročnost systému pro dvousekundový interval vstupu (po inicializaci procesů)

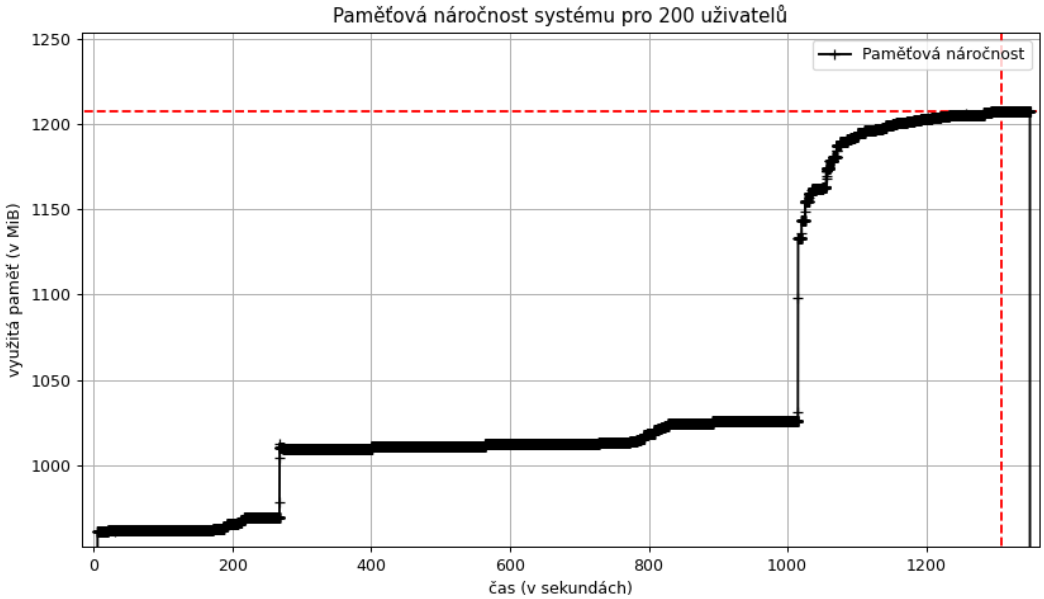
Systém je schopný zpracovat a vhodně odpovědět na většinu požadavků s dostatečnou odezvou pro všechny intervaly využití. Existují zde ale problémy se spolehlivostí implementovaného systému. Pravděbodobně se tak systém chová ze dvou důvodů. Prvním důvodem

jsou problémy při současné změně kontextu a verifikací informací ze senzoru. Může se tak stát, že při získávání `request` zprávy se žádný z uzlů nepovažuje za vedoucí, a tím pádem nerozešle `pre-prepare` zprávy ostatním uzlům. Druhým důvodem je pravděpodobně přehlcení systému zprávami, což může způsobit, že se některé zprávy nepošlou. Může tak dojít například k tomu, že malé množství uzlů nezíská dostatečné množství `commit` zpráv, čímž nepřidá příslušný záznam do blockchainu. Pokud se takový uzel stane vedoucím uzlem, může vzniknout situace, kdy uzel vyhodnotí uživatelský vstup jako nekompatibilní se záznamy v blockchainu a odmítne ho. Mezi možná řešení takového problému patří snížení intenzity zaslaných zpráv prodloužením čekání či snížením TTD. Dalším řešením by bylo zajistit, aby záznamy v blockchainu byly mezi uzly rozpoložené ve stejném pořadí za pomocí sekvenčního čísla, což umožní danému uzlu si vyžádat specifický záznam při výpadku. Odezva na systému nezávisí na intervalu mezi individuálními verifikacemi. Je možné, že pro velmi krátké intervaly (např. 0,1 sekund) se odezva prodlouží kvůli vyššímu množství současně zpracovávaných operací.

Kvůli vysoké paměťové náročnosti inicializace procesů se budou v následujících tabulkách ukazovat pouze výsledky paměťové náročnosti po inicializaci úvodních procesů. Graf 5.2 se bude používat jako referenční graf. V porovnání s referenčním grafem má graf 5.3 vyšší paměťovou náročnost kvůli vyššímu množství provedených verifikací.

Množství uživatelů	Průměrná odezva (s)	Úspěšnost operací (v %)
10	0,19	91,53
50	0,21	96,61
100	0,22	98,31
200	0,19	98,31

Tabulka 5.2: Odezva a úspěšnost verifikačních operací pro různé množství zaregistrovaných uživatelů

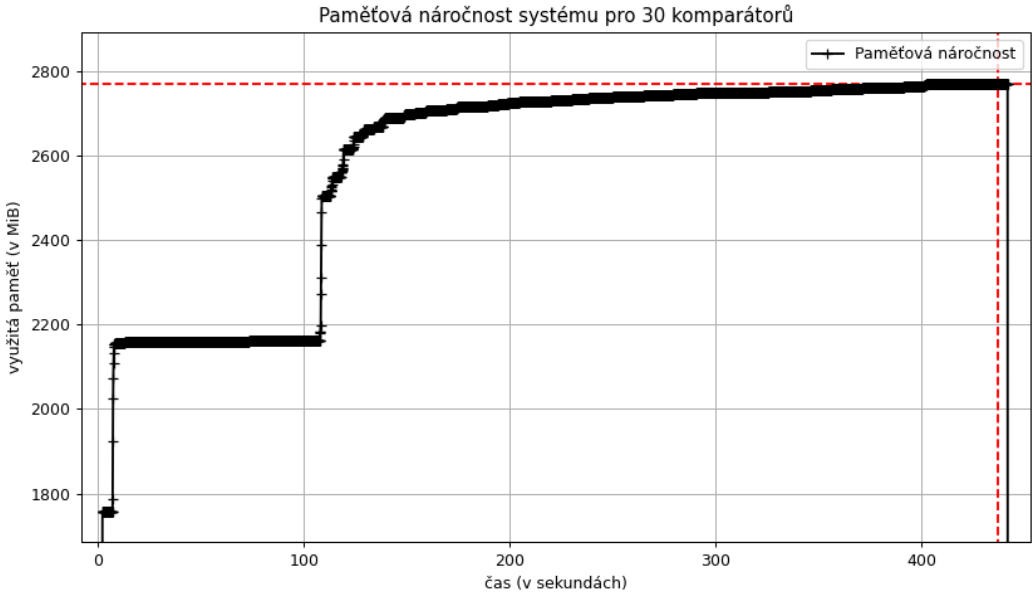


Obrázek 5.4: Paměťová náročnost systému s 200 registrovanými uživateli (po inicializaci procesů)

Množství uživatelů nemá výrazný vliv na odezvu systému. Neúspěšnost zpracování se zvyšuje s nižším počtem zaregistrovaných uživatelů. Děje se tak pravděpodobně z toho důvodu, že u nižšího množství uživatelů existuje vyšší šance, že se bude verifikovat uživatel, u kterého má vedoucí uzel nekonzistentně uložené záznamy v blockchainu. U grafu využití paměti pro 200 uživatelů není nic neočekávaného, celková spotřeba paměti je vyšší kvůli vyššímu množství registrovacích operací prováděném nad systémem.

Množství komparátorů	Průměrná odezva (s)	Úspěšnost operací (v %)
5	0,08	93,33
15	0,31	94,92
20	0,48	95
30	0,9	91,67

Tabulka 5.3: Odezva a úspěšnost verifikačních operací pro různé množství komparátorů

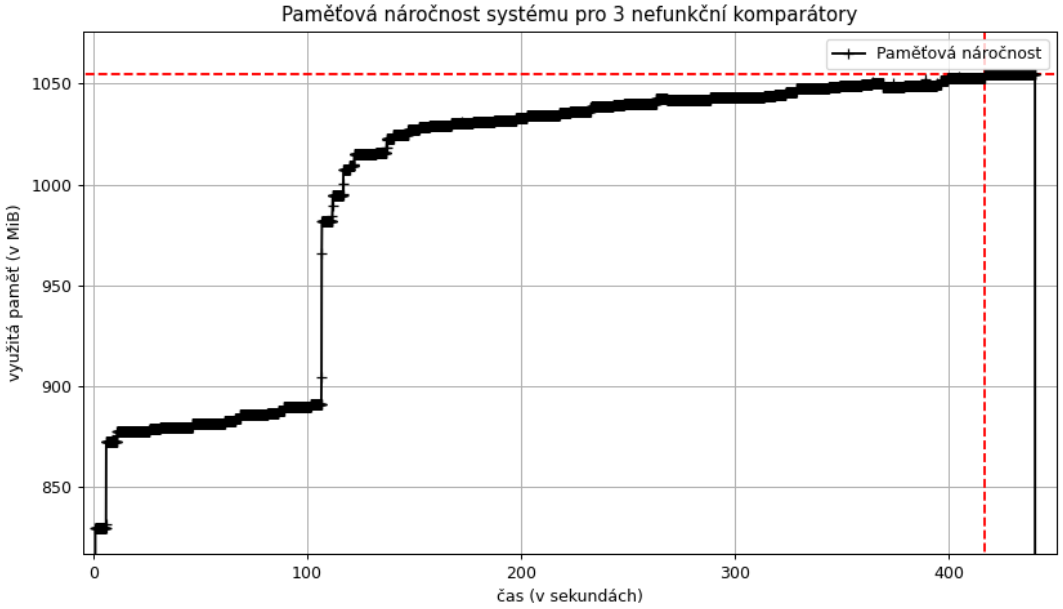


Obrázek 5.5: Paměťová náročnost systému s 30 komparátory (po inicializaci procesů)

Množství komparátorů má rozhodující vliv na odezvu systému. Takové výsledky se daly čekat, neboť intenzita zpráv u protokolu PBFT roste velmi rychle s vyšším počtem komunikujících uzlů. Odezva je ale pořád v přijatelných mezích. Z hlediska paměti je kvůli inicializaci nových procesů paměťová náročnost systému s 30 komparátory téměř 3 GiB. Zvýšení paměťové náročnosti systému během běhu programu je zhruba trojnásobně větší než u verze systému s 10 komparátory, což je očekávané.

Nefunkční komparátory	Průměrná odezva (s)	Úspěšnost operací (v %)
1	0,16	91,53
2	0,14	84,75
3	0,13	81,67
4	N/A	25,42

Tabulka 5.4: Odezva a úspěšnost verifikačních operací pro různé množství nefunkčních komparátorů



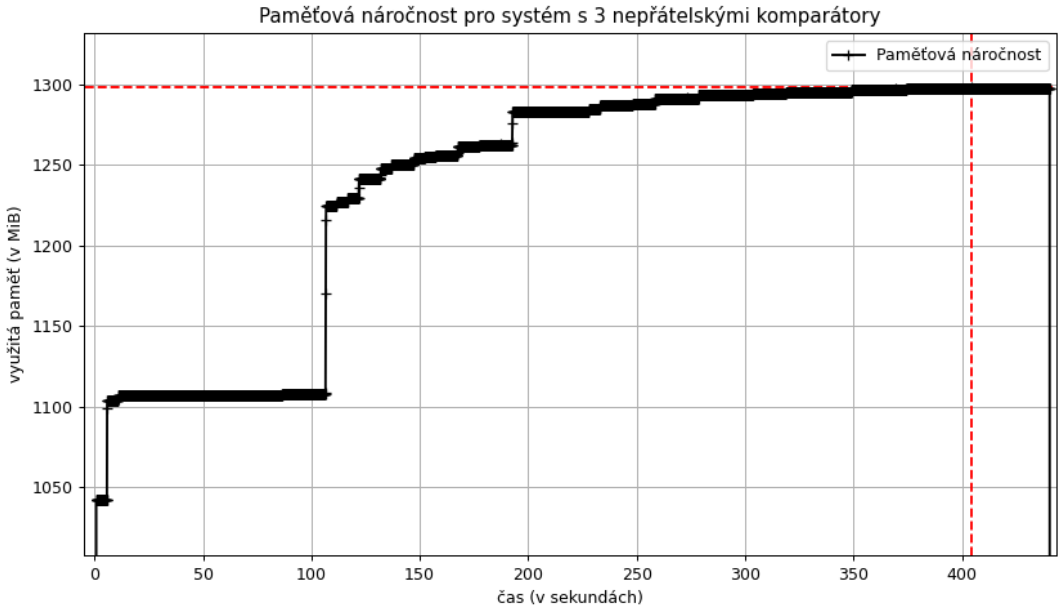
Obrázek 5.6: Paměťová náročnost systému s 3 nefunkčními komparátory (po inicializaci procesů)

Pro 1, 2 a 4 nefunkční komparátory se systém chová podle očekávání. U 4 nefunkčních komparátorů je systém schopný „detekovat“ jen nesprávné vstupy tím, že nezašle příslušné množství signálů otevíračům dveří. Nižší úspěšnost operací u komparátorů 1 a 2 je očekávaná, neboť díky návrhu programu se stanou na určitou délku času vedoucím uzlem, čímž znemožní dočasně fungování systému, neboť nefunkční komparátory nezasílají pre-prepare zprávy. U 3 nefunkčních komparátorů se toto chování ale neprojevuje, což indikuje problém se špatnou implementací změny kontextu. Průměrná odezva operací se mírně snižuje s vyšším množstvím nefunkčních komparátorů, což se pravděpodobně děje kvůli nižší intenzitě komunikace mezi fungujícími komparátory. Také se ukazuje, že pro vyšší množství nefunkčních komparátorů se zvyšuje neúspěšnost verifikačních operací (u 3 nefunkčních komparátorů se jedná o 82 procent, a to i přesto, že změna kontextu se pro tento případ neprovádí), pravděpodobně kvůli vyššímu dopadu ztráty zpráv, neboť zde neexistuje redundance u systémů s vyšší mírou fungujících komparátorů.

Paměťová náročnost pro 3 nefunkční komparátory je nižší než pro referenční příklad, což je očekáváný výsledek, neboť si nefunkční komparátor neukládá výsledky operací do blockchainu.

Nepřátelské komparátory	Průměrná odezva (s)	Úspěšnost operací (v %)
1	0,17	88,33
2	0,17	78,33
3	0,16	78,33
4	N/A	25

Tabulka 5.5: Odezva a úspěšnost verifikačních operací pro různé množství nepřátelských komparátorů



Obrázek 5.7: Paměťová náročnost systému s 3 nepřátelskými komparátory (po inicializaci procesů)

Chování pro nepřátelské uzly je podobné chování pro nefungující uzly. Opakuje se zde neočekáváné chování u tří nepřátelských komparátorů. Také se zde stejně projevuje úspěšnost operací s vyšším množstvím nepřátelských uzlů. Z hlediska využité paměti je paměťová náročnost nepřátelských komparátorů vyšší než u nefunkčních komparátorů a nižší než u správně fungujících komparátorů.

5.4 Vyhodnocení

Systém je schopný správně registrovat a verifikovat uživatele systému. Je schopný korektně odpovědět na bezpečnostní hrozby, jako je přiložení špatných biometrických informací k legitimnímu uživatelovi. Také je schopný za pomocí ověřování historie blockchainu zjistit, jestli byl uživatel přidán do databáze korektně přes distribuovanou síť komparátorů nebo ne.

Během testování se projevily některé nedostatky systému. Zejména se jedná o přehlcení multiprocesorových front zprávami, a také se jedná o chyby v implementaci změny kontextu v rámci PBFT. Opravení těchto problémů bude důležité pro správné a spolehlivé fungování systému.

Kvůli vysoké paměťové náročnosti pro vytvoření nového procesu a kvůli nemožnosti přehledně monitorovat paměťovou náročnost individuálních procesů přineslo měření využité paměti procesů jen malé množství užitečných informací. Z naměřených dat lze ale vidět, že systém využívá nepřiměřené množství paměti. Zčásti se jedná o rušivý element testovacího programu (například testovací vektor, který zabírá nemalé množství paměti, či fronta požadavků mezi otevíračem dveří a testovacím programem). Také se jedná o paměť zabranou vektorem uživatelských informací, které se využívají u čekání na příslušnou zprávu v případech, kdy došla nějaká ze zpráv předčasně, a která se pak vyčistí po určité době.

Je ale možné, že bude potřeba opravit některé aspekty garbage cleaningu a popřípadě také zefektivnit ukládání dat do blockchainu.

Kapitola 6

Závěr

V práci byly analyzovány centralizované a decentralizované autentizační systémy a vyzdvyhnuty přínosy a nedostatky obou typů systémů. Také byla analyzována technologie blockchain a její konkrétní implementace (Bitcoin, Ethereum, PBFT apod.). Byly vyhodnoceny jejich výhody, nevýhody a vhodnost pro implementaci do mechanismu otevírání dveří na základě biometriky v rámci jedné budovy. Dále se v rámci práce analyzovalo zavedení biometriky do blockchainových technologií a problémy s nimi spojené. Na základě analýzy byl navržen biometrický decentralizovaný autentizační systém založený na blockchainovém consensus protokolu PBFT.

Na základě návrhu byla implementována zjednodušená verze autentizačního systému v jazyce Python, kde individuální zařízení byla reprezentována procesy. Také byl vytvořen simulátor provozu nad tímto autentizačním systémem. V testech byla demostrována funkčnost a výkonnost implementace. Systém je schopný provést základní operace (přidání, verifikace, odstranění uživatele) a je schopný zabránit základním útokům (využití špatných biometrických dat k ověření uživatele, útok na databázi apod.). Existují zde ale určité problémy ve specifické implementaci protokolu. Zejména se jedná o problém přehlcování systému zprávami (které způsobí, že se některé zprávy nepošlou), či implementační problémy u změny kontextu (které v některých případech znemožní její správné fungování). Vyřešení těchto problémů je popud k další práci.

Bylo demonstrováno, že za předpokladu vyřešení implementačních nedostatků se dá decentralizace za pomocí blockchainu použít k biometrické autentizaci uživatelů. Program se tak dá s některými úpravami použít jako základ pro reálné nasazení autentizačního systému k otevírání dveří. Úprava systému pro reálné nasazení je předmětem další práce.

Mezi potenciálními aspekty další práce patří některá rozšíření systému. Například jedním z těchto rozšíření by mohla být schopnost otevíračů dveří aktivně odmítnout otevření dveří (na rozdíl od nynější implementace systému, která při nesprávném ověření signál dveřím vůbec nepošle). Mezi dalšími potenciálními rozšířeními funcionality systému by také mohlo patřit vyvolání poplachu pro některé detekované hrozby (například nekorektně přidaný záznam do databáze, nekorektní odstranění legitimního uživatele z databáze apod.), či zavedení symetrického šifrování a timestampování do komunikace mezi individuálními objekty. Mezi další aspekty budoucí práce patří otestování dlouhodobějšího fungování systému či optimalizace systému (zejména z hlediska využité paměti). Také by bylo vhodné otestovat systém pro chytřejší nepřátelské uzly.

Literatura

[1] *Asymmetric Encryption and Decryption in Python* [online]. Nitratine [cit. 2022-05-04]. Dostupné z: https://nitratine.net/blog/post/asymmetric-encryption-and-decryption-in-python/#:~:text=Asymmetric%20encryption%20uses%20two%20keys,key%20can%20decrypt%20the%20message.

[2] *Bitcoin Energy Consumption Index* [online]. Digiconomist [cit. 2022-02-14]. Dostupné z: https://digiconomist.net/bitcoin-energy-consumption/.

[3] *Fingerprint Matching in Python* [online]. Youtube [cit. 2022-05-04]. Dostupné z: https://www.youtube.com/watch?v=IIvfqfKkiio.

[4] *PoS: Advantages and Disadvantages of Proof-of-Stake* [online]. Profolus [cit. 2022-05-04]. Dostupné z: https://www.profolus.com/topics/pos-advantages-and-disadvantages-of-proof-of-stake/.

[5] *Proof of Authority Explained* [online]. Binance [cit. 2022-05-05]. Dostupné z: https://academy.binance.com/en/articles/proof-of-authority-explained.

[6] *Size of the Bitcoin blockchain from January 2009 to April 4, 2022* [online]. Statista [cit. 2022-05-04]. Dostupné z: https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/.

[7] *Sokoto Coventry Fingerprint Dataset (SOCOFing)* [online]. Kaggle [cit. 2022-02-25]. Dostupné z: https://www.kaggle.com/datasets/ruizgara/socofing.

[8] *What you Need to Know about Centralized vs Decentralized Identity Management* [online]. Pingidentity [cit. 2022-02-26]. Dostupné z: https://www.pingidentity.com/en/resources/blog/posts/2021/centralized-decentralized-identity-management.html.

[9] *Regulation of Cryptocurrency Around the World: November 2021 Update* [online]. loc.gov, 2021 [cit. 2022-02-14]. Dostupné z: https://tile.loc.gov/storage-services/service/ll/llglrd/2021687419/2021687419.pdf.

[10] *PROOF-OF-STAKE (POS)* [online]. etereum.org, leden 2022 [cit. 2022-02-14]. Dostupné z: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/.

[11] Almutairi, S., Alghanmi, N. a Monowar, M. M. Survey of Centralized and Decentralized Access Control Models in Cloud Computing. *International Journal of Advanced Computer Science and Applications*. The Science and Information Organization. 2021, sv. 12, č. 2. DOI: 10.14569/IJACSA.2021.0120243. Dostupné z: http://dx.doi.org/10.14569/IJACSA.2021.0120243.

[12] Boult, T. E., Scheirer, W. J. a Woodworth, R. Revocable fingerprint biotokens: accuracy and security analysis. *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007, s. 1–8.

[13] Cachin, C. a Vukolic, M. Blockchain Consensus Protocols in the Wild. *CoRR*. 2017, abs/1707.01873. Dostupné z: http://arxiv.org/abs/1707.01873.

[14] Castro, M. a Liskov, B. Practical Byzantine Fault Tolerance. *Proceedings of the Third Symposium on Operating Systems Design and Implementation, New Orleans, USA*. Únor 1999, s. 1–8.

[15] Chan, T.-H. H., Pass, R. a Shi, E. PaLa: A Simple Partially Synchronous Blockchain. *IACR Cryptol. ePrint Arch.* 2018, sv. 2018, s. 981.

[16] Delgado Mohatar, O., Fierrez, J., Tolosana, R. a Vera Rodriguez, R. Blockchain meets Biometrics: Concepts, Application to Template Protection, and Trends. Březen 2020.

[17] Gordon, W. J. a Catalini, C. Blockchain Technology for Healthcare: Facilitating the Transition to Patient-Driven Interoperability. *Computational and Structural Biotechnology Journal*. 2018, sv. 16, s. 224–230. DOI: https://doi.org/10.1016/j.csbj.2018.06.003. ISSN 2001-0370. Dostupné z: https://www.sciencedirect.com/science/article/pii/S200103701830028X.

[18] Haber, S. a Stornetta, W. S. How to Time-stamp a Digital Document. *Journal of Cryptology*. 1991, sv. 3, s. 99–111.

[19] Hammi, M. T., Hammi, B., Bellot, P. a Serhrouchni, A. Bubbles of Trust: A decentralized blockchain-based authentication system for IoT. *Computers & Security*. 2018, sv. 78, s. 126–142. DOI: https://doi.org/10.1016/j.cose.2018.06.004. ISSN 0167-4048.

[20] Magnani, A., Calderoni, L. a Palmieri, P. Feather Forking as a Positive Force: Incentivising Green Energy Production in a Blockchain-Based Smart Grid. In:. New York, NY, USA: Association for Computing Machinery, 2018, s. 99–104. CryBlock'18. DOI: 10.1145/3211933.3211951. ISBN 9781450358385. Dostupné z: https://doi.org/10.1145/3211933.3211951.

[21] Masdari, M., Jabbehdari, S., Ahmadi, M., Hashemi, S. M., Bagherzadeh, J. et al. A survey and taxonomy of distributed certificate authorities in mobile ad hoc networks. *EURASIP Journal on Wireless Communications and Networking*. Prosinec 2011, sv. 2011. DOI: 10.1186/1687-1499-2011-112.

[22] Narayanan, A., Bonneau, J., Felten, E., Miller, A. a Goldfeder, S. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016. ISBN 9780691171692. Dostupné z: https://books.google.cz/books?id=Ncu4jwEACAAJ.

[23] Ngo, D., Teoh, A. a Goh, A. Biometric hash: high-confidence face recognition. *IEEE Transactions on Circuits and Systems for Video Technology*. 2006, sv. 16, č. 6, s. 771–775. DOI: 10.1109/TCSVT.2006.873780.

[24] Nguyen, B. M., Dao, T.-C. a Do, B.-L. Towards a blockchain-based certificate authentication system in Vietnam. *PeerJ Computer Science*. PeerJ Inc. 2020, sv. 6, s. e266.

[25] Shehu, Y. I., Ruiz Garcia, A., Palade, V. a James, A. Detection of Fingerprint Alterations Using Deep Convolutional Neural Networks. In: *Proceedings of the International Conference on Artificial Neural Networks (ICANN 2018)*. 2018. DOI: 10.48550/arXiv.1807.10609.

[26] Sinnott, R. O., Chadwick, D. W., Doherty, T., Martin, D., Stell, A. et al. Advanced Security for Virtual Organizations: The Pros and Cons of Centralized vs Decentralized Security Models. In: *2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*. 2008, s. 106–113. DOI: 10.1109/CCGRID.2008.67.

[27] Tulyakov, S., Farooq, F., Mansukhani, P. a Govindaraju, V. Symmetric Hash Functions for Secure Fingerprint Biometric Systems. *Pattern Recognition Letters*. Prosinec 2007, sv. 28, s. 2427–2436. DOI: 10.1016/j.patrec.2007.08.008.

[28] Zhaofeng, M., Jialin, M., Jihui, W. a Zhiguang, S. Blockchain-Based Decentralized Authentication Modeling Scheme in Edge and IoT Environment. *IEEE Internet of Things Journal*. 2021, sv. 8, č. 4, s. 2116–2123. DOI: 10.1109/JIOT.2020.3037733.