Kryptografické algoritmy v rámci technologií blockchain

Petr Nagy

Bakalářská práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Petr Nagy**
Osobní číslo:	**A20371**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Kryptografické algoritmy v rámci technologií blockchain**
Téma práce anglicky:	**Cryptographic Algorithms in Blockchain Technologies**

Zásady pro vypracování

1. Nastudujte a popište problematiku technologie blockchain.
2. Zpracujte srovnání kryptografických algoritmů v technologii blockchain.
3. Sepište možnosti využití kryptografie v blockchainu s ohledem na kvantové počítače.
4. Zvolte vhodné technologie a implementujte ukázkový blockchain.
5. Výsledky vhodně popište a prezentujte.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. DRESCHER, Daniel a Laurence KIRK. Blockchain basics: a non-technical introduction in 25 steps. New York, NY: Apress, ©2017, 1 online zdroj (xv, 255 stran). Dostupné z: doi:9781484226049
2. PECINOVSKÝ, Rudolf. Python: Kompletní příručka jazyka pro verzi 3.10. Grada, 2021, 1 online zdroj (608 stran). ISBN 978-80-271-4423-5. Dostupné také z: https://www.bookport.cz/AccountSaml/SignIn/?idp=https://shibboleth.utb.cz/idp/shibboleth&returnUrl=/kniha/python--10436/
3. LAURENCE, Tiana, 2019. Blockchain For Dummies. 2nd. New Jersey: John Wiley. ISBN 9781119555018.
4. TAPSCOTT, Don a Alex TAPSCOTT, 2018. Blockchain Revolution: How the Technology Behind Bitcoin and Other Cryptocurrencies is Changing the World. New York: PENGUIN UK. ISBN 9780241237861.
5. What is blockchain technology?. IBM [online]. IBM, 20. 6. 2018 [cit. 2022-10-11]. Dostup-né z: https://www.ibm.com/topics/what-is-blockchain

Vedoucí bakalářské práce: **Ing. Petr Žáček, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **26. května 2023**

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

Petr Nagy, v.r.
podpis studenta

ABSTRAKT

Tato bakalářská práce se zabývá kryptografickými algoritmy využívanými v technologii blockchain a jejich možné využití i v době kvantových počítačů. V teoretické části se prvně rozebere téma blockchainu, jednotlivé typy blockchainů a jejich výhody a nevýhody. Nadále se rozeberou kryptografické algoritmy využívané v technologii blockchain a využití blockchainu v praxi. V poslední řadě se rozeberou možnosti blockchainu v době kvantových počítačů a jejich vliv na tuto technologii. V praktické části se rozebere implementace ukázkového blockchainu, který slouží pro lepší pochopení, jak blockchain funguje.

Klíčová slova:

Python, Graphviz, Blockchain, Grafické rozhraní, Blok, Hash, Funkce, Post-kvantová kryptografie

ABSTRACT

This bachelor thesis is about the cryptographic algorithms used in blockchain technology and their possible use in the era of quantum computers. At first, the theoretical part discusses the topic of blockchain, different types of blockchains and their advantages and disadvantages. Then it will discuss the cryptographic algorithms used in blockchain technology and the use of blockchain in practice. Finally, the possibilities of blockchain in the era of quantum computers and their impact on this technology will be discussed. The practical part will discuss the implementation of a sample blockchain, which is used to better understand how blockchain works.

Keywords:

Python, Graphviz, Blockchain, Graphics Interface, Block, Hash, Function, Post-Quantum Cryptographics

Tímto bych chtěl velmi poděkovat vedoucímu práce Ing. Petru Žáčkovi, Ph.D. za cenné rady, konzultace a pevné nervy ohledně vypracovávání bakalářské práce.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD **8**
I TEORETICKÁ ČÁST **9**
1 CO JE TO BLOCKCHAIN **10**
1.1 HISTORIE A VÝVOJ BLOCKCHAINU 11
1.2 FUNKCE BLOCKCHAINU 11
1.2.1 Princip decentralizace 12
1.3 PŘIDÁNÍ BLOKU DO BLOCKCHAINU 12
1.4 VÝHODY A NEVÝHODY BLOCKCHAIN TECHNOLOGIE 13
2 TYPY BLOCKCHAINU **15**
2.1 VEŘEJNÝ BLOCKCHAIN 15
2.1.1 Proof-of-Work a Proof-of-Stake 16
2.2 SOUKROMÝ BLOCKCHAIN 17
2.3 KONSORCIÁLNÍ BLOCKCHAIN 17
2.4 HYBRIDNÍ BLOCKCHAIN 18
2.5 SIDECHAIN 19
2.5.1 Two-Way Peg 19
3 VYUŽÍVANÉ KRYPTOGRAFICKÉ ALGORITMY **20**
3.1 HASH FUNKCE 20
3.2 DIGITÁLNÍ PODPIS 22
3.2.1 ECDSA 23
3.3 SYMETRICKÉ ŠIFROVÁNÍ 24
3.3.1 Blokové šifry 25
3.3.2 Proudové šifry 28
3.4 ASYMETRICKÉ ŠIFROVÁNÍ 28
3.5 MERKLE TREE (HASHOVÝ STROM) 29
3.5.1 Ověření souboru 30
3.6 VYUŽITÍ V TECHNOLOGII BLOCKCHAIN 31
4 VYUŽITÍ TECHNOLOGIE BLOCKCHAIN V PRAXI **33**
4.1 KRYPTOMĚNY 33
4.2 ZDRAVOTNICTVÍ 34
4.3 STÁTNÍ SLUŽBY 35
4.4 DODAVATELSKÝ ŘETĚZEC 36
5 KRYPTOGRAFIE V DOBĚ KVANTOVÝCH POČÍTAČŮ **37**
5.1 SYMETRICKÉ A ASYMETRICKÉ ŠIFROVÁNÍ 38
5.2 HASH FUNKCE 39
5.3 POST-KVANTOVÁ KRYPTOGRAFIE 39
5.3.1 Lattice-Based Cryptography (Kryptografie založená na mřížce) 40
5.3.2 Hash-Based Cryptography (Kryptografie založená na Hashi) 41
5.3.3 Code-Based Cryptography (Kryptografie založená na kódech) 41
5.3.4 Multivariate Cryptography (Vícerozměrná kryptografie) 42

5.4 KVANTOVĚ ODOLNÝ BLOCKCHAIN 42
5.4.1 Hashgraph 43
5.4.2 Quantum Resistant Ledger (QRL) 44
II PRAKTICKÁ ČÁST 45
6 POUŽITÉ TECHNOLOGIE 46
6.1 PROGRAMOVACÍ JAZYK PYTHON 46
6.2 VÝVOJOVÉ PROSTŘEDÍ 46
6.3 GRAFICKÁ VIZUALIZACE POMOCÍ GRAPHVIZ 47
6.4 QT DESIGNER 48
7 IMPLEMENTACE BLOCKCHAINU 49
7.1 GRAFICKÉ ROZHRANÍ APLIKACE 50
7.1.1 Úvodní okno aplikace 50
7.1.2 Vytvoření nového blockchainu 52
7.1.3 Hlavní okno aplikace 53
7.2 IMPLEMENTACE BLOCKCHAINU 58
7.2.1 Funkce pro logování a získání času 58
7.2.2 Vytvoření a přidání bloku do blockchainu 58
7.2.3 Úvodní blok v blockchainu 59
7.2.4 Další přidaný blok v blockchainu 61
7.2.5 Vytváření bloků a přidávání je do blockchainu 62
8 GRAFICKÁ REALIZACE 65
8.1 ÚPRAVA DAT PRO SPRÁVNÉ ZOBRAZENÍ 65
8.1.1 Stylizace Hash hodnoty 65
8.1.2 Stylizace nadpisu a textových dat 66
8.1.3 Vytvoření prvního bloku v blockchainu 67
8.1.4 Přidání dalšího bloku do diagramu 68
8.1.5 Vygenerování posledního přidaného bloku 69
ZÁVĚR 71
SEZNAM POUŽITÉ LITERATURY 72
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK 82
SEZNAM OBRÁZKŮ 83
SEZNAM PŘÍLOH 85

ÚVOD

Technologie se pořád zlepšuje a zrychluje. K tomu navíc teď zažívá obrovskou pozornost médií a veřejnosti kvantové počítače a jejich neskutečný výkon v porovnání s dnešními počítači. I přesto, že je vize kvantových počítačů v každé domácnosti ještě v nedohlednu, tak zabezpečení proti nim musí přijít radši dříve než později. Na druhou stranu není úplně snadné vytvářet zabezpečení na něco, co tak úplně neznáme a nevíme, jak se bude v realitě chovat. Proto se nám může zdát, že se snažíme o něco, co nemůžeme vědět, jak nakonec dopadne. Pravda je tak někde mezi. Již v dnešní době existují kvantové počítače, které zatím dokáží fungovat pouze jen v laboratorních podmínkách. I přes to disponují možností vyzkoušení určitých algoritmů, které jsou sestavené právě na kvantové počítače.

O technologii blockchain slyšel asi každý, který se trochu více zajímá o technologie, jejich vývoj nebo o kryptoměny. Právě kryptoměna Bitcoin byla největším průkopníkem v oblasti blockchainu a mnoho lidí si díky tomu myslí, že Bitcoin a blockchain je totéž. S rozvojem technologie blockchain se ale její využití rozšířilo a její potenciál je v dnešní době obrovský. Blockchain se začíná rozsáhle implementovat i mimo kryptoměny, kvůli jeho výhodám v určitých odvětvích. Rozvíjející se kvantové počítače se stávají potenciální hrozbou právě i pro tuto technologii.

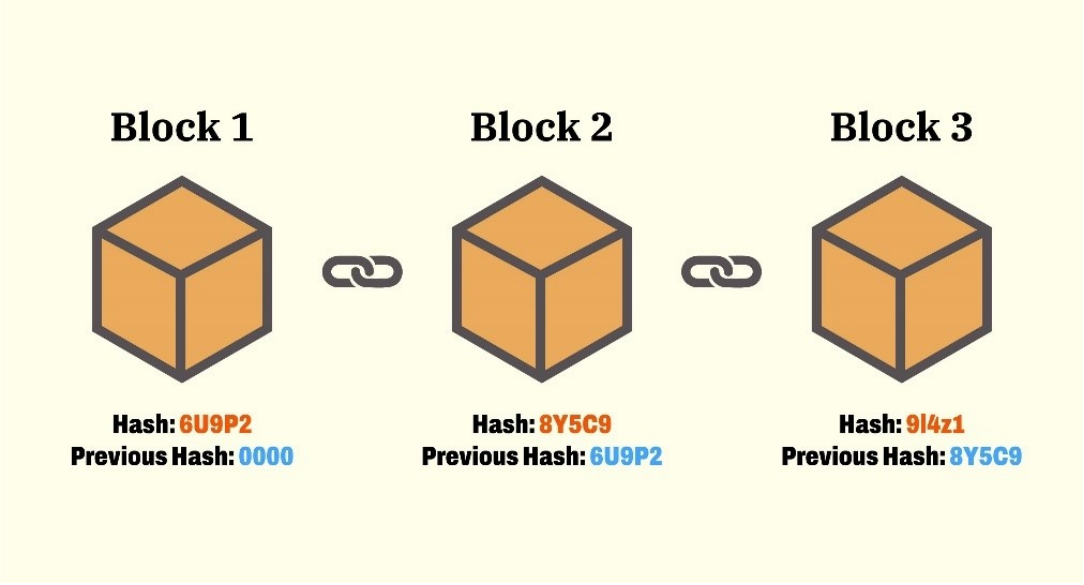
I. TEORETICKÁ ČÁST

1 CO JE TO BLOCKCHAIN

Blockchain je distribuovaná účetní kniha nebo databáze, která je sdílená mezi všechny počítače připojené do stejné sítě. Tento základní princip decentralizace je hlavním základním kamenem skoro každého blockchainu. Hlavním cílem decentralizace je vytvářet zabezpečenou účetní knihu bez potřeby třetí strany, která by data kontrolovala a spravovala. Nejčastěji se blockchain používá v oblasti kryptoměn, ale nově se začíná objevovat i například ve zdravotnictví, umělé inteligenci nebo také ve velkoobchodním či maloobchodním sektoru. [1] [2]

Všechny data v blockchainu jsou ukládány v elektronické podobě do bloků, které se posléze propojí s předchozím přidaným blokem a vytvoří řetězec bloků. Toto představuje také vysokou míru zabezpečení a důvěryhodnost dat přidaných do blockchainu, protože po přidání bloku do řetězce se zároveň k přidanému bloku přidá i časové razítko v době vytvoření bloku. Navíc je vše transparentní a každý si může najít jakýkoliv přidaný blok do řetězce díky jejich propojení mezi sebou. Často se blockchain přirovnává ke klasické databázi. Rozdíl oproti klasické databázi je v tom, že databáze ukládá data do tabulek a blockchain do bloků. [1] [2]

Obsah dat přidaných do blockchainu můžou být různé. Nejznámější využití blockchainu jsou v systémech kryptoměn, jako jsou například Bitcoin nebo Ethereum které uschovávají záznam transakcí. [1] [2]



Obrázek 1 – Propojení bloků v blockchainu [7]

1.1 Historie a vývoj blockchainu

Historie vývoje technologie blockchainu je datována od roku 1991, kdy byla úvodní myšlenka zabezpečit řetězce bloků, kde by nebylo možné upravovat časová razítka vytvoření jednotlivých bloků. První větší vylepšení technologie přislo hned v následujícím roce, kdy byly začleněny Hashové stromy. Toto vylepšení přineslo významné zlepšení efektivity, ale primárně, možnost uložení více dokumentů do jednoho bloku. [3] [4]

Poté až do roku 2008 byl vývoj poklidnější než vývojář nebo skupina vývojářů pod pseudonymem Satoshi Nakamoto vydala whitepaper[1], stanovující základní model této technologie. Rok na to tento vývojář nebo skupina implementovali první veřejnou účetní knihu. Tato veřejná účetní kniha sloužila pro transakce prováděné pomocí kryptoměny Bitcoin. [3] [4]

V roce 2013 přišla další velká aktualizace, která některé části opravila či vylepšila, a dokonce obohatila o nějaké funkce. Tuto aktualizaci přineslo vydání blockchainu Ethereum, který umožnoval oproti původnímu modelu navíc ještě zaznamenávat další aktivity. Blockchain Etherea byl oficiálně vydán až v roce 2015. S touto změnou přišlo rozšíření blockchainu i na vývoj decentralizovaných aplikací mimo kryptoměny. [3] [4]

Od představení blockchainu Etherea vývoj technologie rapidně vzrostl a využití už nachází i mimo kryptoměny, například v oblasti řízení dodávek zboží nebo k přístupu k online nástrojům, uložištím a v mnoha dalších oblastech. [3] [4]

1.2 Funkce blockchainu

Blockchain funguje na základě určitých pravidel, které umožňují vytvářet a přidávat bloky do blockchainu. Je zapotřebí, aby všechny přidané bloky nebylo možné zpětně upravovat nebo odstranit. Toto zabezpečení získáme pomocí decentralizace. Další primární funkce, kterými blockchain disponuje je bezpečnost a důvěryhodnost přidaného záznamu. Jelikož jsou decentralizované blockchainy dostupné pro veřejnost, tak jsou navíc i trans-

[1] Whitepaper (Bíla kniha) – Dokument sloužící pro objasnění a vysvětlení daného složitého problému potencionálním uživatelům

parentní, protože si každý může zobrazit obsah včetně historie. Existují ale i soukromé blockchainy, které jsou dostupné pouze na pozvání nebo oprávnění a každý člen nemusí mít pravomoci si zobrazovat přidaná data do blockchainu.

1.2.1 Princip decentralizace

Hlavním přínosem decentralizace v technologii blockchain je, že kontrola přidávaného bloku a celého blockchainu se předává z centrálního subjektu na všechny účastníky připojené do sítě. Tímto se zabrání, aby se kontrola nedostala do rukou pouze pár lidí, nebo dokonce někomu se špatnými úmysly. Každý uživatel připojený do sítě vlastní přesnou kopii distribuovaného blockchainu a v případě, že nějaký uživatel má vlastní kopii blockchainu poškozenou, ostatní uživatelé po kontrole jeho kopii odmítnou. Tímto se zvýší bezpečnost a důvěryhodnost blockchainu. [5]

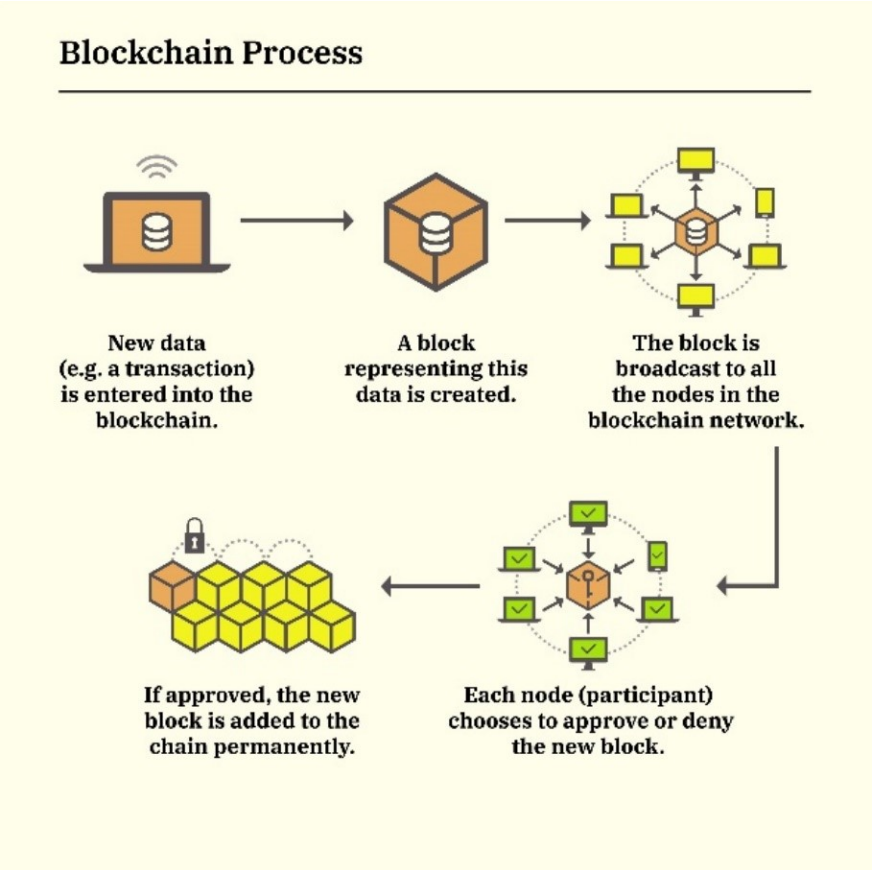
Hlavními výhodami decentralizace jsou vysoká přesnost dat, protože po přidání s nimi nejde nijak manipulovat a všechny přidané bloky se zkopírují mezi všechny uživatele, co zajistí integritu dat. Další majoritní výhodou oproti centralizovanému systému, že neexistuje jediný bod selhání. [5] [6]

Mezi nevýhody decentralizace můžeme uvést náklady na provoz sítě, protože je potřeba více zařízení pro její údržbu a chod. Také velmi často přitahují kriminalitu, kvůli jejich časté úplné anonymitě. [6]

1.3 Přidání bloku do blockchainu

Pro přidání bloku do blockchainu je prvně zapotřebí vytvořit blok obsahující data. Z dat obsažených v bloku se vytvoří Hash, který slouží jako digitální podpis vytvořeného bloku. K propojení jednotlivých bloků mezi sebou, obsahuje každý blok ještě Hash hodnotu předchozího bloku. [7]

Tento vytvořený blok se posléze pošle všem účastníkům sítě, kteří musí ověřit a schválit přidání bloku do blockchainu. Ověření a schválení musí provést všichni uživatelé sítě, aby se zaručila pravost a důvěryhodnost bloku. Jelikož musí tyto operace provést všichni uživatelé sítě, je časté, že čím větší síť je, tím déle trvá proces přidání bloku. Na druhou stranu tento princip přináší zvýšenou bezpečnost přidávaných dat a odolnost proti útočníkům. [7]



Obrázek 2 – Proces přidávání bloku [7]

1.4 Výhody a nevýhody blockchain technologie

Některé výhody a nevýhody jdou odvodit z principu decentralizace, na které většina blockchainu stojí. Důležité je ale podotknout, že blockchain provádí i některé operace navíc, které mohou určité části, jak vylepšit, tak i zhoršit.

Mezi výhody můžeme zařadit: [2] [8]

- **Neměnnost dat** – Data přidané do blockchainu nepodporují, jako klasické databáze, operace CRUD (Create Read Update Delete). Data lze pouze vytvořit v podobě bloků a číst jejich obsah.
- **Dohledatelnost** – Kvůli nemožnosti změnit data a propojení jednotlivých bloků mezi sebou je dohledání jakéhokoliv bloku v sítí snadné, protože propojené bloky vytvářejí cestu.
- **Transparentnost** – Tato výhoda přichází společně s decentralizací, protože každý uživatel sítě má přístup k datům v jakýkoliv čas a může si ověřit jejich pravost oproti centralizovanému systému.

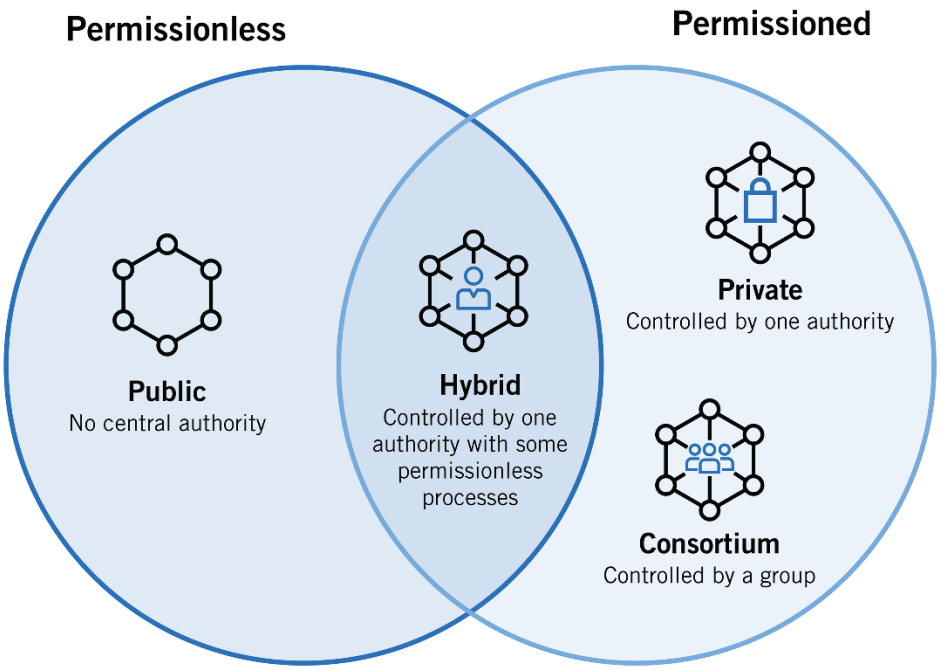
- **Bezpečnost** – Pro přidání bloku dat do blockchainu musí být vystaven souhlas buď všech nebo většiny uživatelů v síti. Navíc všechny přidané bloky jsou trvalé a neměnné Tímto se zaručí větší bezpečnost jak samotných dat, tak i uživatelů připojených v síti.
- **Důvěryhodnost** – Všechna data přidané do sítě blockchain jsou přesná a aktuální. V privátních typech blockchainu jsou navíc data sdílená jen mezi účastníky, kteří mají přístup do sítě.
- **Efektivita práce** – Jelikož jsou data uložená v blockchainu distribuovaná mezi všechny účastníky, není potřeba zpracovávat požadavky od uživatelů. V některých sítích lze také aplikovat určitý soubor pravidel, které se boudou automaticky provádět.

Některé výhody přináší v určitých případech i své nevýhody. Za některé nevýhody můžeme považovat: [2] [8]

- **Neměnnost dat** – Občas je zapotřebí zaznamenané data změnit nebo upravit, ale blockchain nám umožnuje pouze přidávat nové záznamy. Tento problém není v klasické databázi.
- **Rychlost** – Rychlost blockchainu je pomalejší oproti klasickým databázím, protože provádí více operací při přidání bloku dat. Je nutné vypočítat Hash bloku, který se následně musí ověřit. Přidaný blok musí schválit všichni nebo většina uživatelů v sítí blockchainu, aby se mohl blok přidat. Například algoritmus ověřování Proof-of-Work (česky Důkaz o práci), omezuje množství přidaných bloků za určitý čas, protože vyžaduje vysoký výkon po uživatelích sítě. Rychlejší ověřování zajišťuje algoritmus Proof-of-Stake (česky Důkaz o podílu)
- **Škálovatelnost** – Omezení rychlosti zpracovávání požadavků některých blockchainů může mít za následek špatnou škálovatelnost blockchainu.
- **Redundance** – I když je tato vlastnost důležitá pro ověřování přidaných bloků, tak to s sebou nese i určité obtíže. Je zapotřebí mít několikanásobně větší úložné prostory pro data, protože každý uživatel vlastní svojí vlastní kopii dat.
- **Peněžní náklady** – Vyšší náklady jdou ruku v ruce společně s redundancí, kvůli potřebnému hardwaru. Oproti klasické databázi je také nákladnější jej integrovat do již rozběhnutých systémů.

2 TYPY BLOCKCHAINU

Jelikož neexistuje žádný systém, který by se dal použít v každém případě, tak někdy je zapotřebí, aby existovaly různé modulace systémů, které upřesňují funkcionalitu systému na dané použití. Jiné to není ani v případě systémů blockchain. Blockchain se dělí do dvou základních kategorií, přístupné bez povolení majitele a přístupné pouze s povolením. Dohromady existují čtyři hlavní typy blockchainu, veřejný, soukromý, konsorciální a hybridní. Speciální roli zabírají sidechainy, které můžou být buď veřejné nebo soukromé. [9]



Obrázek 3 – Rozdělení typů blockchainů [9]

2.1 Veřejný blockchain

Veřejný blockchain, jak už název napovídá, je veřejně dostupný, a každý uživatel se může připojit do sítě a provádět všechny operace povolené v síti. Tento typ blockchainu je naprosto decentralizován, protože neobsahuje žádný centrální prvek, a pro přidání všech bloků je zapotřebí povolení od většiny uživatelů v síti. Veřejné blockchainy se v dnešní době nejčastěji používají u kryptoměn. Zde jsou uživatelé obměňováni za svou práci malým podílem ze zisku v podobě kryptoměny. Aby mohli být nějak obměněni, musí nějak prokázat svou práci, a to buď pomocí Proof-of-Work nebo Proof-of-Stake. Tato volba není na uživateli, ale na konfiguraci blockchainové sítě. Mezi nejznámější veřejně blockchainy se řadí Ethereum nebo Bitcoin. [10] [11] [12]

Oproti ostatním typům blockchainu, je nejdůležitější výhodou transparentnost, protože se každý uživatel sítě podílí na jeho řízení. Uživatelé mají tedy důležitější postavení oproti soukromým typům blockchainu. Toto vytváří i vyšší důvěryhodnost přidaných dat mezi uživateli sítě. [10] [11] [12]

Vyšší bezpečnost a důvěryhodnost kvůli potřebě schválení každého přidaného bloku všemi uživateli sítě si s sebou nese nevýhodu v podobě nedostatečné rychlosti přidávání nových bloků do blockchainu. Tímto se výrazně zhorší škálovatelnost blockchainu a zvýší spotřeba energie. [10] [12]

2.1.1 Proof-of-Work a Proof-of-Stake

PoW (Proof-of-Work) a PoS (Proof-of-Stake) jsou algoritmy využívané k ověřování nové přidaných bloků (transakcí) u kryptoměn. Oba algoritmy využívají jiný princip ověření přidaných transakcí, ale pořád slouží ke stejnému účelu, a to udržovaní blockchainu v bezpečí. Ověřování provádí samotní uživatelé sítě, kteří jsou za ověření odměnění peněžní hodnotou v kryptoměně. Tito uživatelé se v případě PoW nazývají těžaři a v případě PoS se nazývají validátoři. Nejznámější kryptoměnou využívající PoW je Bitcoin a v případě PoS je to Ethereum. [13] [14] [15]

Hlavní princip fungování algoritmů je ztížit ověření přidaných bloků, aby bylo složitější a často i finančně nevýhodné zneužití jedincem nebo skupinou se špatnými úmysly, například pomocí získání většiny při ověřování transakce. Algoritmy se liší ve dvou základních principech, energetická náročnost a riziko útoku většiny. [13] [14] [15]

V případě PoW je spotřeba energie oproti PoS extrémně vysoká, protože jsou zapotřebí velmi výkonné počítače pro řešení kryptografických rovnic. Těžař, který rovnici vyřeší jako první, přidá transakci do blockchainu. Finanční odměnu za práci dostane až ostatní účastníci sítě ověří data z přidané transakce pomocí svých kopií blockchainu. V případě PoS se validátoři zavazují za ověření vstupním finančním vkladem a hodnotou, jak dlouho již transakce ověřují. Právě na základě těchto dvou hodnot se vybírají validátoři, kteří budou danou transakci ověřovat. Pokud validátoři ověří transakci správně, jsou jim vstupní vklady vráceny obohacené o odměnu a transakce se přidá do blockchainu. V opačném případě můžou o svůj vklad přijít. [13] [14] [15]

Rizikem útoku získání většiny při ověřování u PoW znamená, že je zapotřebí, aby těžař nebo skupina těžařů ovládala více jak 50 % výkonu celé sítě, což může být finančně

a energeticky velmi náročné. U PoS je zapotřebí, aby každý uživatel měl vstupní vklad, když se chce podílet na ověřování transakcí, který mu může být odebrán v nesprávném ověření transakce. Validátor nebo validátoři, kteří by chtěli získat většinu by museli vlastnit více než 50 % vložené hodnoty financí, která může být velmi vysoká. V případě, že by k takovému útoku došlo, mohou ostatní validátoři hlasovat o ignorování ověřené transakce. [13] [14] [15]

2.2 Soukromý blockchain

Oproti veřejnému blockchainu, je soukromý blockchain dostupný pouze uživatelům, kteří byli pozváni nebo mají povolení od majitele blockchainu. Nejčastěji je tento typ využíván uvnitř nějaké organizace pro interní účely, která jej i zároveň spravuje a kontroluje. Toto znamená, že se nejedná už o zcela decentralizovaný systém. Navíc pro potvrzení přidání bloku dat do blockchainu není častokrát ani zapotřebí schválení všech uživatelů sítě. Většinou stačí pouze schválení vybranými uživateli nebo se používá chytrá smlouva, která obsahuje specifická pravidla pro schvalování přidání bloku. Dále je zde možnost, oproti veřejným blockchainům, upravovat již přidané bloky do blockchainu. U přidaných bloků je možné smazat nebo upravit jejich obsah. [10] [12] [16]

Mezi výhody soukromého blockchainu zařadíme hlavně vyšší rychlost a efektivitu oproti veřejným blockchainům. Protože obsahuje menší počet uživatelů, tak jsou i jednotlivé náklady na provoz sítě nižší a blockchain dokáže lépe škálovat, protože není zapotřebí čekat na schválení od mnoha uživatelů. [10] [12] [16]

Nižší úroveň decentralizace nebo její úplná absence stojí za menším zabezpečením a důvěryhodností dat. Ve většině blockchainů je nastavené, že klasičtí uživatelé nemohou kontrolovat přidané bloky, protože nemají pro tuto akci povolení. Toto povolení mají pouze vysoce postavení uživatelé nebo majitelé blockchainu. [10] [12]

2.3 Konsorciální blockchain

Konsorciální blockchain, nebo také známý jako federativní, je velmi podobný soukromému blockchainu. Nejpoužívanější je totiž mezi jednotlivými organizacemi, které již využívají vlastní soukromý blockchain. Každá organizace, která je připojena do konsorciálního blockchainu má právo se podílet na ověřování, schvalování, přidávání a upravování dat. Jelikož síť spravuje více organizací, oproti jedné jako v případě soukromého blockchainu, je

tato varianta více decentralizovaná, protože je vyžadována shoda organizací na jakékoliv změně, ale stále ne tak moc, jako u veřejného blockchainu. [17] [18]

Konsorciální blockchainy se již využívají například ve finančním sektoru, logistice nebo ve zdravotnictví. Ve finančním sektoru sdílí různé banky data o svých zákaznících na jednom místě. Zde si poté může jiná banka, která má přístup do blockchainové sítě, najít informace o zákazníkovi. V logistice je obecně složité udržovat informace o zboží od původního místa vzniku až do cílové destinace, protože je zde možnost zneužití organizací po cestě. Tento problém by se vyřešil přidáním všech organizací, přes které zboží projde, do společného konsorciálního blockchainu, kde by se evidovaly všechny manipulace se zbožím. V poslední řadě je využití ve zdravotnictví. Zde je začlenění teprve na začátku, ale přináší mnoho výhod jak pro nemocnice či lékaře tak i pro pojišťovny. Veškeré záznamy o pacientovy by mohli být přístupné všem těmto objektům. Například by se zkrátila čekací a vyřizovací doba při vyplácení pojištění nebo při registraci pacientů. [17]

Hlavním principem je přenos dat mezi jednotlivými organizacemi a snížit náklady a čas při předávání dat. Toto je způsobeno menším počtem uživatelů, kteří ještě navíc musí dodržovat smluvená pravidla, které byly vytvořeny společně s vytvořením blockchainové sítě. Dále jsou společně se soukromými blockchainy, díky menším počtům uživatelů, dobře škálovatelné. [17] [18]

Největší nevýhodou tohoto typu blockchainu je, že pro vytvoření této sítě je zapotřebí domluva a shoda uživatelů sítě, ještě před jejím vytvořením. Je totiž nutné sepsat pravidla a vymyslet, jak bude probíhat komunikace mezi organizacemi v rámci sítě. Dalším záporem je jednoznačně větší centralizace než u veřejných blockchainů. [17] [18]

2.4 Hybridní blockchain

Hybridní blockchain kombinuje ty nejlepší vlastnosti z veřejného i soukromého blockchainu. Hybridní blockchain je stejně jako soukromý pouze na pozvání, takže není veřejně přístupný. Je stejně jako soukromý blockchain řízený jednou organizací nebo uživatelem, ale pro schválení, přidání ověření dat je zapotřebí shoda většiny uživatelů sítě nebo se může využívat také chytrá smlouva. Stejně jako ve veřejném blockchainu nelze s přidanými daty provádět žádné operace. Uživatelé nebo organizace stojící za řízením sítě můžou vybírat, které data budou dostupné veřejně nebo pouze uživatelům připojeným do sítě s požadovanými oprávněními. [18] [19] [20]

Jelikož se jedná o uzavřený typ blockchainu, tak náklady na provoz jsou nižší. Dále je rychlejší a efektivnější než veřejný blockchain. Stejně jako soukromý blockchain je také lépe škálovatelný díky menšímu počtu uživatelů. Jedná se o nejlepší možnost při volbě blockchainu, který má chránit soukromí uživatelů a zároveň komunikovat s vnějšími systémy. [18] [19]

Protože je možné vybírat, které data budou veřejně dostupné a které nikoliv, tak tento typ blockchainu není úplně transparentní, jako veřejný blockchain. [18] [19]

2.5 Sidechain

Sidechain je samostatná blockchainová síť připojená k hlavní síti pomocí Two-Way Peg (česky Oboustranného kolíku), který zajišťuje komunikaci mezi sítěmi. Sidechain funguje nezávisle na hlavní síti pomocí vlastního ověřování dat, kde data jsou ověřovány uživateli sidechainové sítě nebo vlastní chytrou smlouvou. Hlavní blockchainové sítě mohou mít k sobě napojené více sidechanů. Využití sidechainů je úzce spjaté se škálovatelností sítě, kterou jednoznačně vylepšuje a zjednodušuje, a zároveň se zlepšuje také i rychlost celé sítě. Dále jsou výhodné pro testování nových funkcí a aktualizací pro hlavní síť, než se na hlavní síti použijí, protože jsou zcela oddělené od hlavní sítě a chyby se detekují předem, než se použijí na hlavním blockchainu. Sidechainy se nevyužívají pouze na rozšíření hlavní sítě, ale mohou poskytovat i decentralizované blockchainové aplikace (DApps). [21] [22]

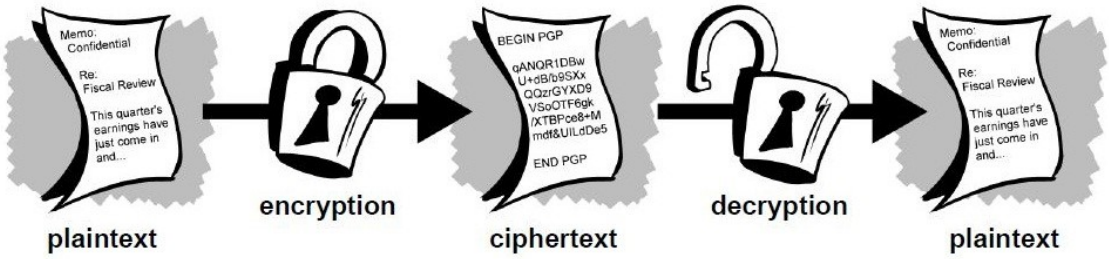
2.5.1 Two-Way Peg

Two-Way Peg slouží k oboustranné komunikaci mezi dvěma blockchainovými sítěmi tak, aby do ní nemohl vstoupit další aktér a manipulovat či zabránit převodu. U blockchainů kryptoměn slouží k převodu kryptoměny mezi hlavní síti a sidechainem. Kryptoměna se ve skutečnosti ale nepřevede, nýbrž se v hlavní síti pouze uzamkne a stejná částka se přičte v sidechainu. Jelikož se kryptoměny reálně nepřevedou, využívají se na validaci a přičtení hodnoty kryptoměny chytré smlouvy, které vše provedou automaticky. [21] [23]

3 VYUŽÍVANÉ KRYPTOGRAFICKÉ ALGORITMY

Kryptografie slouží k ochraně informací a uživatelů v síti před třetí stranou, aby nemohla číst a zneužít soukromá data. Šifrované data jsou více bezpečná a důvěryhodná, protože se zamezuje v přístupu neoprávněným osobám. V případě že by se třetí strana dostala k zašifrovaným datům, tak stejně nedokáže zjistit originální obsah bez znalosti klíče pro dešifrování. [24] [25]

Kryptografické algoritmy mohou být obousměrné, ale i jednosměrné. V případě obousměrného kryptografického algoritmu jde po zašifrování získat zpět původní data pomocí speciálního klíče. Tento princip je využitý v případě symetrických a asymetrických šifrování a existuje mnoho typů algoritmů využívající některý z těchto typů šifrování. U jednosměrných kryptografických algoritmů, jako je Hashovací funkce, je v podstatě nemožné se po zašifrování dostat zpět k původním datům. [24] [25]

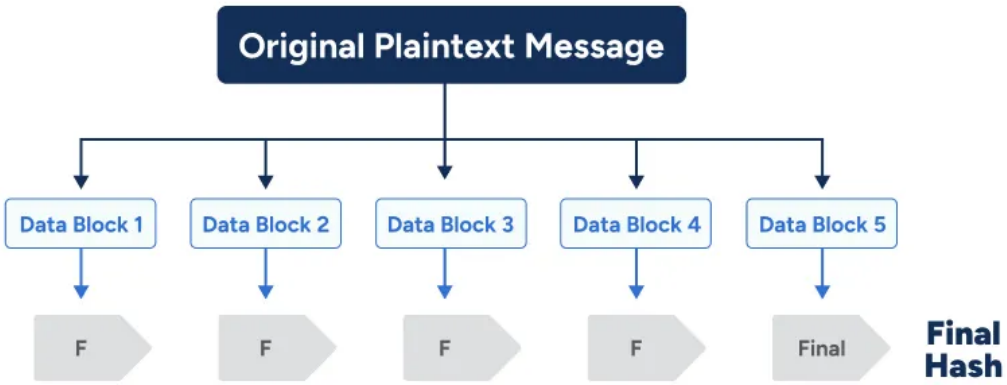


Obrázek 4 – Obousměrné kryptografické algoritmy [26]

U blockchainových technologií se nejčastěji setkáme s Hashovací funkcí, ale vyskytují se zde i ostatní kryptografické algoritmy. Dohromady zajišťují bezpečnost, neměnnost dat a soukromí uživatelů v síti.

3.1 Hash funkce

Hashovací funkce je založená na velmi složité matematické operaci, která z různě dlouhého digitálního vstupu udělá výstup o pevně stanovené délce. Protože Hashovací funkce pracuje s bitovými operacemi, které pracují pouze s daty o určité délce, tak se prvně vstupní data rozdělí na bloky o stejné délce. Pokud poslední blok nemá požadovanou délku, tak se jeho délka doplní. Poté se proces šifrování Hash funkcí použije na všechny bloky, na kolik byly vstupní data rozdělena. [27]



Obrázek 5 – Hash funkce [27]

Hashovací operace je jednosměrná, to znamená, že zašifrovaná data nelze získat zpět v původní podobě pomocí dešifrovacího klíče. Jedna z mála možností, jak se dostat zpět k původním datům, se nabízí průlom brutální silou neboli metodou pokus omyl. Tento proces je velmi náročný na výkon, protože například při délce Hashe 256 bitů je počet kombinací roven 2^{256}. Další vlastností Hashovací funkce je, že je deterministická. To zajišťuje, že stejná vstupní data se zašifrují na vždy stejný výsledný Hash. Naopak jakákoliv změna ve vstupních datech se zobrazí jako veliká změna ve výstupním Hashi. Různé typy Hashovacích funkcí předávají výsledek o různých velikostech, nejčastěji od 128 bitů do 512 bitů. Například funkce SHA-256 (Secure Hash Algorithm) zašifruje vstupní data na délku 256 bitů neboli 64 hexadecimálních znaků, kde oproti tomu funkce MD5 (Message Digest) pouze na délku 128 bitů neboli 32 hexadecimálních znaků. [27]

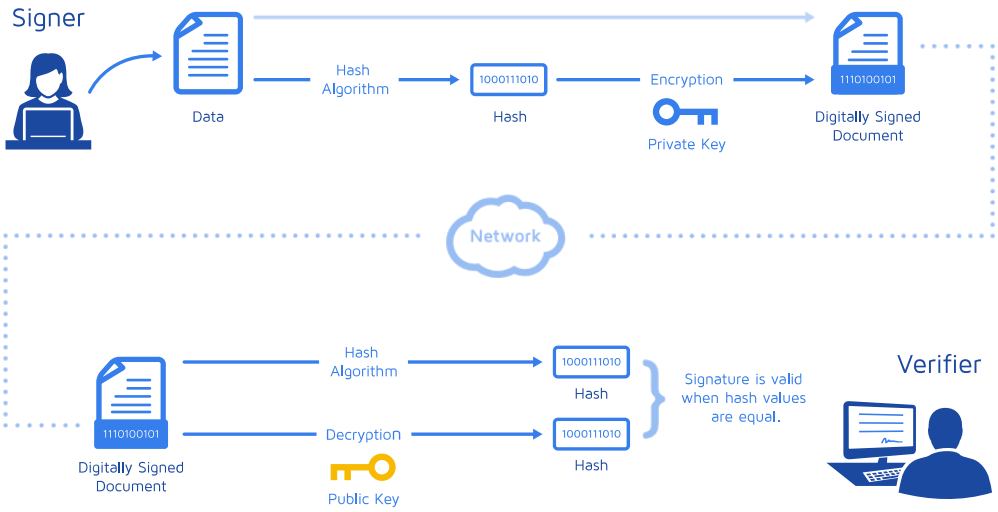
V blockchainu se Hashovací funkce využívá pro ověřování pravosti bloků dat a pro propojení bloků mezi sebou. Každý přidaný blok obsahuje Hash z předchozího bloku a Hash ze svých dat. Jelikož jakákoliv změna dat v bloku se projeví na výsledné hodnotě Hashe, tak je velmi snadné identifikovat neoprávněné změny v blockchainu. Mimo blockchain se hashovací funkce využívají pro kontrolu přihlašovacích údajů, kde se kontroluje Hash hodnota hesla s uloženou Hash hodnotou v databázi. Dále se využívají i v digitálních podpisech pro udržení jejich unikátnosti a bezpečnosti. [27]

Vstupní data	Hashovací Algoritmus	Výsledný Hash
Univerzita Tomáše Bati je otevřenou a flexibilní vysokou školou, kterou rozvíjíme na základě pětice ústředních hodnot.	SHA-256 256 bitů = 32 bytů = 64 znaků	c5af94037c9719a1ef56732a7705f911 dd4a04bbacdd619586cba293bd5ac3e4
Univerzita Tomáše Bati je otevřenou a flexibilní vysokou školou, kterou rozvíjíme na základě pětice ústředních hodnot.	MD5 128 bitů = 16 bytů = 32 znaků	510e42546b5a8935119fd9a8d306cd53
Univerzita Tomáše Bati je otevřenou a flexibilní vysokou školou, kterou rozvíjíme na základě pětice ústředních hodnot.	SHA3-256 256 bitů = 32 bytů = 64 znaků	fc12051d007432b35e66784ba32c0445 33c8adc6c52df8bb8f023b9c6fe3392e

Obrázek 6 – Hashovací funkce SHA a MD (vlastní zdroj)

3.2 Digitální podpis

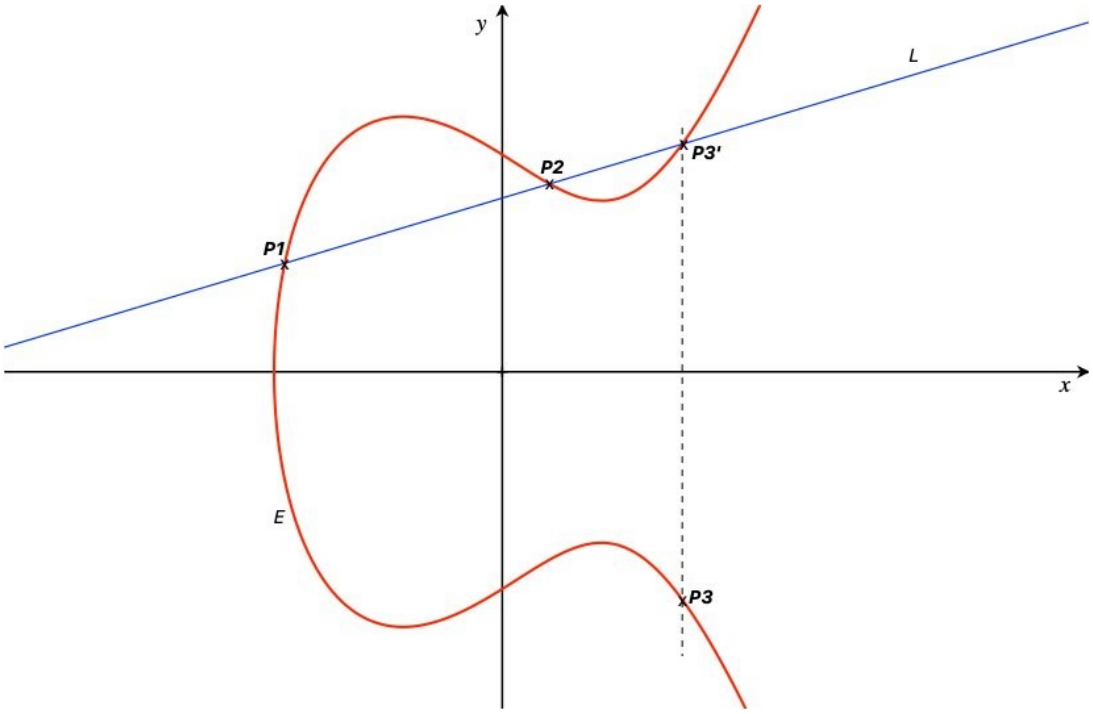
Digitální podpis je typ elektronického podpisu, který nahrazuje fyzický podpis. Přidává se většinou k dokumentům, u kterých zajišťuje pravost identity autora či odesílatele a věrohodnost a originalitu dokumentu. Digitální podpis funguje na principu asymetrického šifrování. Prvně se vytvoří Hash z dokumentu, který se poté zašifruje soukromým klíčem vlastníka a vznikne digitální podpis. Digitální podpis obsahuje navíc ještě časové razítko, ze kdy byl digitální podpis vytvořen. Poté se dokument pošle společně s digitálním podpisem příjemci, který ověří pravost dokumentu tím, že vytvoří z obdrženého dokumentu Hash a veřejným klíčem dešifruje digitální podpis, který obsahuje originální Hash dokumentu ještě před posláním. Pokud je obdržený dokument nezměněný, obě dvě Hash hodnoty jsou totožné. [28] [29]



Obrázek 7 – Digitální podpis [28]

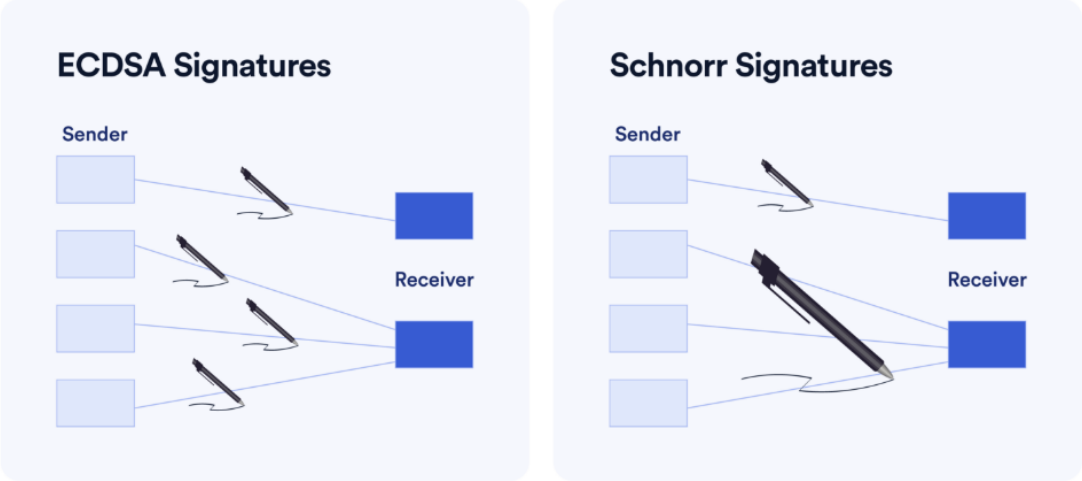
3.2.1 ECDSA

Algoritmus digitálního podpisu eliptickou křivkou (ECDSA) využívá k vytváření klíčů eliptické křivky. Tato metoda se využívá mnohem častěji v technologiích blockchain než například algoritmus RSA (Rivest-Shamir-Adleman), protože používá menší klíče, které zajišťují stejnou úroveň zabezpečení. V porovnání délka klíče 384 bitů u ECDSA zaručuje stejnou úroveň zabezpečení jako klíč o délce 7680 bitů při šifrování pomocí RSA. [30] [31]



Obrázek 8 – Eliptická křivka [31]

Protože u kryptoměn je zapotřebí, aby mohli podepsat jednu transakci vícero uživatelů, tak se v blockchainech kryptoměn používají upravené modely ECDSA algoritmu. Bez tohoto vylepšení by bylo zapotřebí kontrolovat digitální podpisy jeden po jednom, pokud by transakci podepsalo vícero uživatelů. Bitcoin používá algoritmus Schnorr, který komprimuje více podpisů dohromady s určitou mírou náhody a poté je možnost je ověřit dohromady jako celek. Jelikož se podpisy komprimují do jednoho celku, uvolní se místo pro jiné data v transakcích. [30]

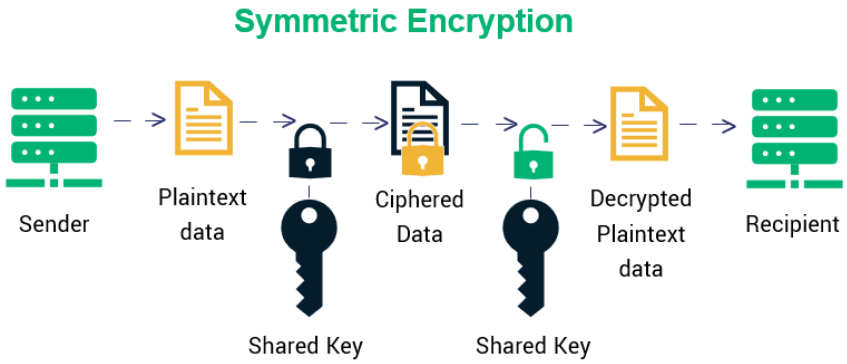


Obrázek 9 – ECDSA vs. Schnorr algoritmus [32]

Ethereum používá algoritmus BLS (Bohen-Lynn-Shacham), jelikož oproti Bitcoinu používá algoritmus PoS namísto PoW. BLS stejně jako Schnorr kombinuje jednotlivé podpisy, ale bez náhody. Tím je jejich tvoření i kontrola jednodušší a rychlejší. [30]

3.3 Symetrické šifrování

Při symetrickém šifrování se používá jeden stejný klíč k šifrování i dešifrování dat. Uživatelé posílající si zprávy nebo dokumenty si musí předem vyměnit tajný klíč, kterým budou data šifrovat a poté zpětně dešifrovat do původní podoby. Jelikož je používán pouze jeden klíč pro šifrování i dešifrování, tak je tento systém náchylnější na útoky odposlechu. U starších šifer platilo, že při častém používání jednoho klíče se zvyšovala šance na jeho vypočítání či reprodukování útočníkem. Redukování této šance je možné používáním sady klíčů, které se budou často měnit. Na druhou stranu tento celý proces vyžaduje značné úsilí ve zprávě klíčů a je zapotřebí domluva obou stran, aby používali pokaždé stejný klíč. Výhodami symetrického šifrování jsou jeho jednoduchost na vytvoření a zpracování, rychlost šifrování a dešifrování a požadování nižšího výkonu strojů. Symetrické šifrování se dělí na dva základní typy, blokové a proudové. [33]

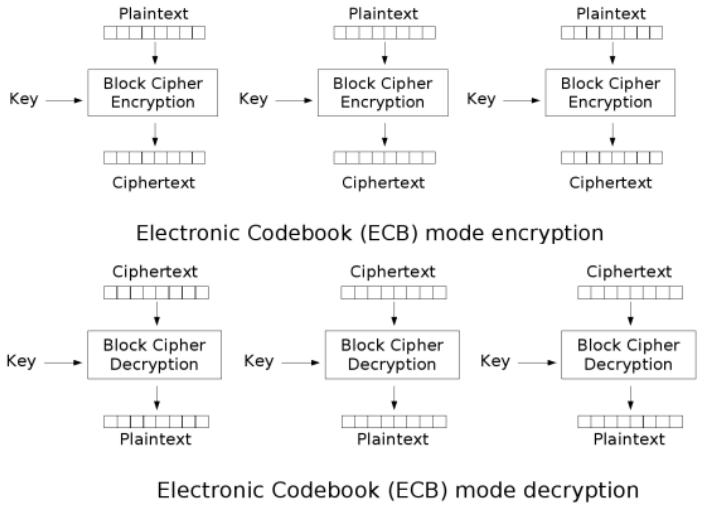


Obrázek 10 – Symetrické šifrování [34]

3.3.1 Blokové šifry

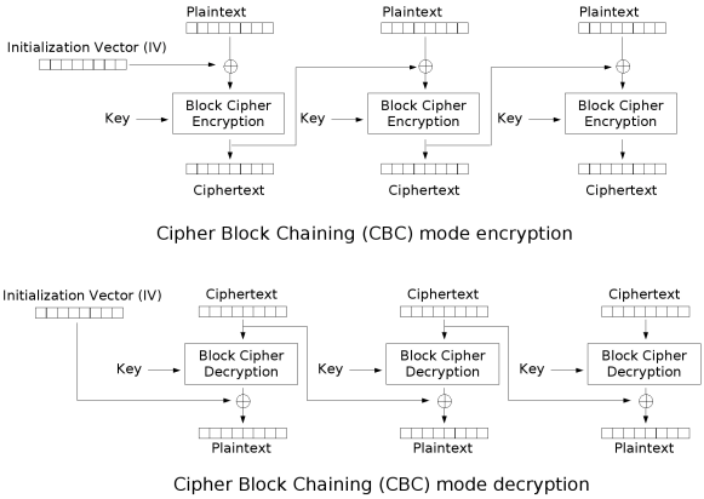
U blokových šifer, jak už název naznačuje, probíhá šifrování a dešifrování po blocích vstupních dat. Všechny bloky mají pevnou velikost, nejčastěji 64 až 256 bitů, a pokud je zapotřebí, tak poslední blok je doplněn na požadovanou velikost. Jednotlivé bloky jsou poté šifrované za pomocí substituční abecedy. Nejznámější blokové šifry jsou Data Encryption Standard (DES) a Advanced Encryption Standard (AES). Hlavním rozdílem mezi těmito dvěma šiframi je délka klíče. Zatímco DES má klíč o velikosti 56 bitů, tak AES má klíče o velikostech 128, 192 a 256 bitů. Blokové šifry se dělí na: [33] [35]

- **Electronic Codebook Mode (ECB)** – jedná se o nejjednodušší model blokové šifry, protože neobsahuje žádný prvek náhody a každý blok dat je šifrován stejným způsobem podle abecedy. Velikost posledního bloku není zapotřebí upravovat, použije se pouze potřebná část velikosti klíče.



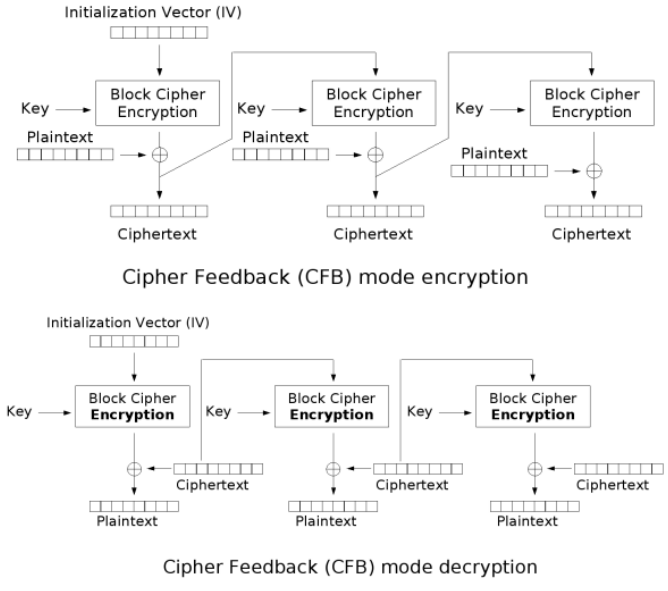
Obrázek 11 – ECB bloková šifra [36]

- **Cipher Block Chaining Mode (CBC)** – tato metoda blokové šifry již obsahuje určitou míru náhody, protože blok dat pro šifrování se zkombinuje se zašifrovanými daty předchozího bloku použitím bitové operace XOR (exkluzivní OR). První blok obsahuje Inicializační vektor (IV) oproti zašifrovaným datům předchozího bloku.



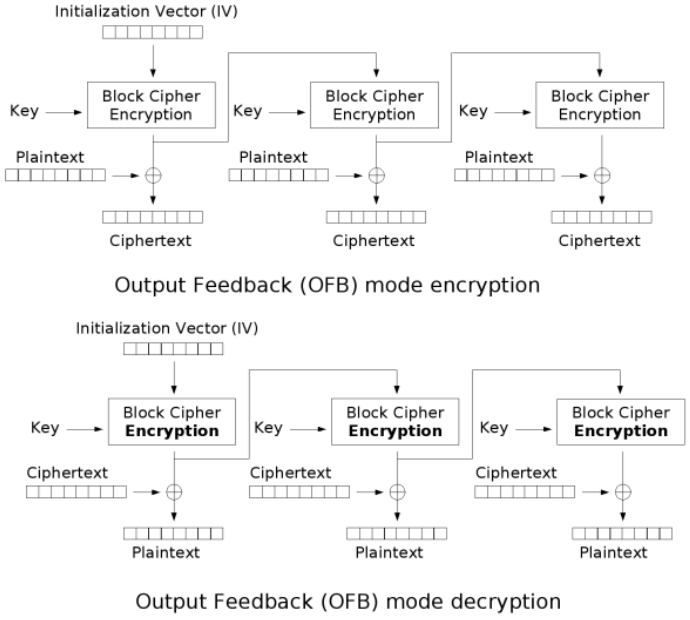
Obrázek 12 – CBC bloková šifra [36]

- **Ciphertext Feedback Mode (CFB)** – CFB bloková šifra je velmi podobná CBC blokové šifře, ale v tomto případě prvně zašifrujeme šifrovaná data zkombinovaná s originálními daty předchozího bloku pomocí operace XOR. Výsledná zašifrovaná data zkombinujeme pomocí operace XOR s originálními daty bloku. První blok použije IV namísto dat z předchozího bloku.



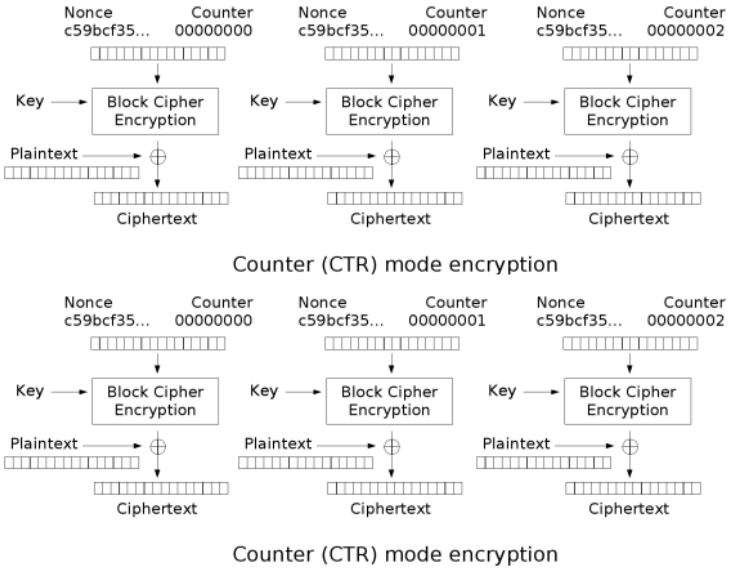
Obrázek 13 – CFB bloková šifra [36]

- **Output Feedback Mode (OFB)** – OFB bloková šifra je rozdílná oproti CFB blokové šifře akorát v tom, že šifruje data předchozího bloku ještě přes zkombinováním s originálními daty.



Obrázek 14 – OFB bloková šifra [36]

- **Counter Mode (CTR)** – CTR bloková šifra je velmi podobná ECB blokové šifře, akorát jako vstupní data pro šifrování používá kombinaci náhodné hodnoty nonce a čísla pořadí bloku. Po zašifrování této hodnoty se výsledek zkombinuje pomocí XOR s originálními daty bloku.

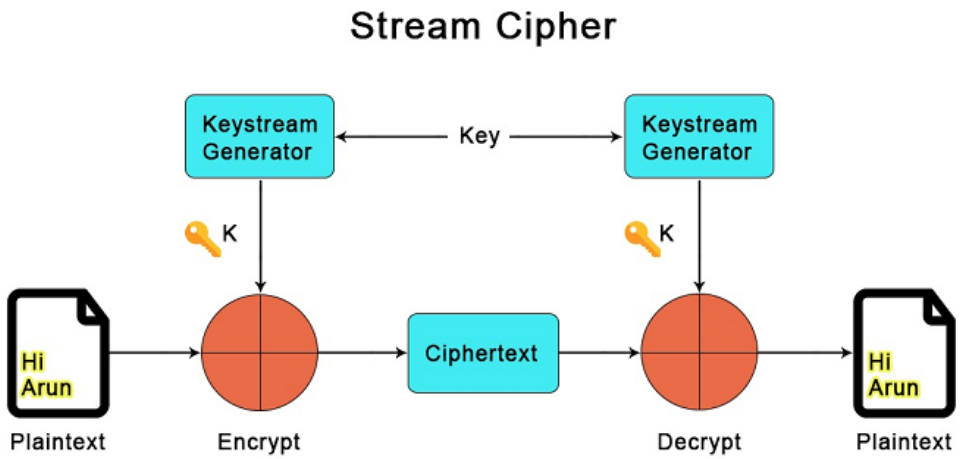


Obrázek 15 – CRT bloková šifra [36]

3.3.2 Proudové šifry

Oproti blokovým šifrám, proudové šifry šifrují vstupní data bit po bitu, kde se používá kombinace 64 až 256 bitů dlouhého klíče a číslici nonce o velikosti 64 až 128 bitů. Kombinací se vytvoří náhodné číslo používané pro šifrování proudu bitů. Tato kombinace může být použita pouze jednou během šifrování, ale hodnota klíče a číslice nonce opakovaně. Jelikož většinou používáme jeden klíč, je zapotřebí neustále měnit hodnotu číslice nonce. Proudové šifry rozdělujeme na dva typy: [33] [35]

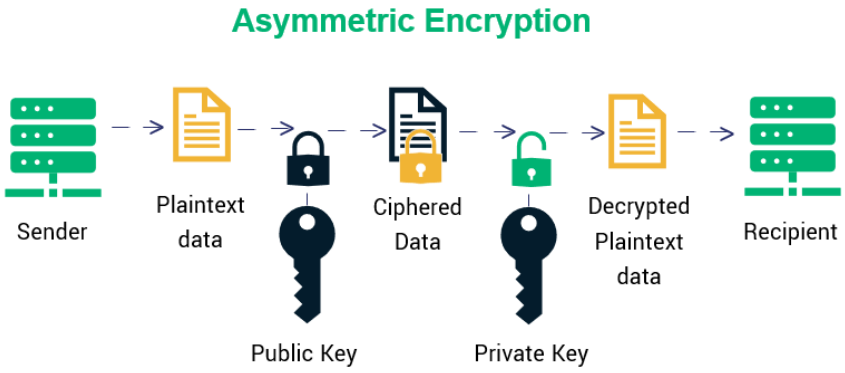
- **Synchronní proudové šifry** – tyto proudové šifry fungují na bázi šifrování bitu po bitu pomocí náhodně generované kombinace klíče a číslice nonce, která se poté zkombinuje pomocí XOR se vstupními daty.
- **Auto-synchronizující/Asynchronní proudové šifry** – u těchto proudových šifer vstupuje do generování šifrovacího klíče velikost již zašifrovaných dat.



Obrázek 16 – Proudová šifra [35]

3.4 Asymetrické šifrování

Asymetrické šifrování používá klíčový pár, soukromý a veřejný klíč, vygenerovaný pomocí matematické funkce. Data se šifrují pomocí veřejného klíče a dešifrovat jdou pouze pomocí matematicky odvozeného soukromého klíče. Zašifrovaná data nelze tedy dešifrovat veřejným klíčem. Oproti symetrickému šifrování není zapotřebí řešit bezpečnost při sdělování klíče, protože pro šifrování se používá veřejný klíč, který je všem dostupný. Vlastníkovi, který klíčový pár vygeneroval, zůstává soukromý klíč pro dešifrování zpráv či dokumentů. Jelikož jsou klíče dva a mají větší délku než při symetrickém šifrování, je tato metoda šifrování pomalejší a nákladnější. [34] [37]



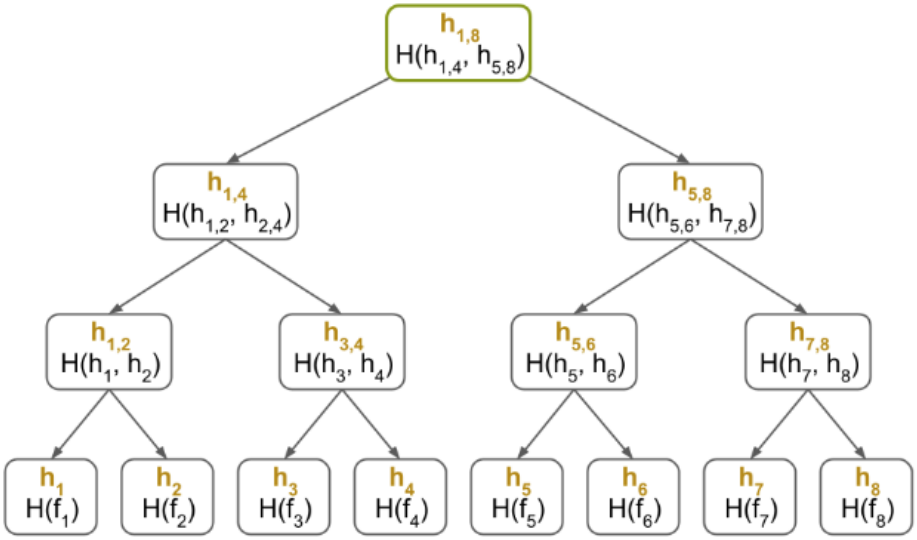
Obrázek 17 – Asymetrické šifrování [34]

Nejznámější příklad asymetrického šifrování je již zmíněný digitální podpis, který se také využívá v systému blockchain. Dále se často asymetrické šifrování využívá v kombinaci se symetrickým šifrováním. Tato kombinace se nejčastěji využívá při end-to-end šifrování v aplikacích pro zasílání zpráv jako jsou například Signal nebo Whatsapp. Asymetrické šifrování se použije pro prvotní inicializaci a sdělení soukromého symetrického klíče mezi dvěma uživateli, který slouží pro šifrování a dešifrování zpráv po celou dobu komunikace. Stejný princip se používá také u komunikace HTTPS (Hypertext Transfer Protocol Secure) mezi klientem a serverem. [38]

3.5 Merkle Tree (Hashový strom)

Merkle Tree slouží pro vytvoření Hashe z vícero vstupních dokumentů a k následnému ověření jednotlivých dokumentů z jednoho celku. Vytvoření Hash hodnoty se všech dokumentů současně by bylo časově náročné, protože by se muselo počkat na všechny dokumenty, než se nahrají na server nebo do bloku dat. Proto se prvně vytvoří Hash z každého přidaného dokumentu zvlášť. Následně se vytvoří Hash ze dvou sousedních hodnot Hashů. Tento proces se opakuje, dokud se nevytvoří finální hodnota Hash z poslední dvojice nazývaná Merkle Root (Hashový kořen). Celý postup je graficky znázorněn na obrázku 18. [39] [40]

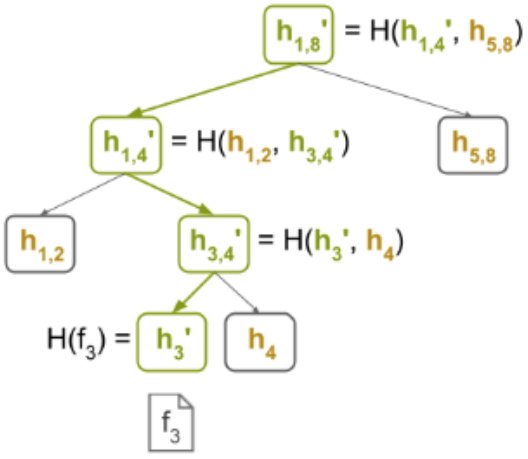
V technologii blockchain se Merkle Tree využívají pro ověřování transakcí v blocích blockchainu kryptoměn. Použitím Merkle Tree se uvolní místo v bloku transakcí, protože namísto velkého počtu Hashů všech transakcí, je uložená pouze jedna hodnota Merkle Root. Tímto se redukuje velikost celého blockchainu. [41]



Obrázek 18 – Konstrukce Merkle Tree [40]

3.5.1 Ověření souboru

Díky použití Merkle Tree je ověření náhodného souboru snazší a rychlejší, protože není zapotřebí stahovat nebo replikovat Hash hodnoty všech souborů, ale pouze potřebných uzlů k jeho ověření. Je ale zapotřebí, aby uživatel, který chce ověřit dokument, vlastnil hodnotu Merkle Root Hashe. Společně se staženým souborem se stáhne i malá část celého Merkle Tree, která se nazývá Merkle Proof. Merkle Proof obsahuje pouze nutné uzly, které jsou zapotřebí k ověření dokumentu. Například pro ověření souboru f_3 z obrázku 18 jsou zapotřebí Hash hodnoty $h_4, h_{1,2}$ a $h_{5,8}$. Kontrola se provede reprodukováním Hash hodnot stejným způsobem, jako při vytváření Merkle Tree. Po zreprodukování Merkle Root se otestuje, jestli je jeho hodnota stejná jako lokálně uložená hodnota originálního Merkle Root. [39] [40]



Obrázek 19 – Ověření souboru v Merkle Tree [40]

3.6 Využití v technologii blockchain

Kryptografické algoritmy se v technologii blockchain využívají primárně pro ochranu údajů uživatelů a dat přidaných do sítě. V různých sítích blockchain se můžeme setkat s různými kryptografickými algoritmy, ale všechny blockchainové sítě obsahují Hashovací funkci, jelikož se jedná o takový základní kámen celé technologie blockchain. Další algoritmus, který by neměl chybět v žádném z blockchainů je digitální podpis, typ asymetrické kryptografie. [42] [43]

Využití hashovací funkce v blockchainu je zejména pro udělení unikátního ID, který zaručuje, že se s daty od jejich vzniku nijak nemanipulovalo. Jelikož nám Hash funkce vytváří výstup o pevné délce z libovolně dlouhého vstupu a výstupní hodnoty jsou nekolizní, kontrola pravosti dat je tedy snadná, protože nám stačí porovnat Hash hodnotu původních a nových dat. Díky unikátnosti Hashovací funkce a propojeností bloků, které obsahují vlastní Hash a Hash předchozího bloku, je také mnohem snazší najít požadovaná data v celém řetězci bloků. V případě blockchainu Bitcoin a Hyperledger Fabric je využíván Hashovací algoritmus SHA-256 a u blockchainu Etherea se používá Keccak256, který byl v roce 2015 standardizován jako SHA3. [43] [44] [45] [46]

Nejznámějším využitím asymetrického šifrování v blockchainu jsou digitální podpisy, jejichž hlavním účelem je zajištění vlastníka (autora) přidaných dat. Vlastník použije svůj soukromý klíč z vygenerovaného klíčového páru pro vytvoření digitálního podpisu a přidá je k datům. Kterýkoliv jiný uživatel sítě si může ověřit identitu autora vytvořeného digitálního podpisu náležícího k datům pomocí veřejného klíče z klíčového páru. Ke generování klíčového páru se nejčastěji používá algoritmus ECDSA, který využívá eliptické křivky a je mnohem bezpečnější než RSA. ECDSA využívají například blockchainy Bitcoin, Ethereum i Hyperledger Fabric. Další možné použití asymetrického šifrování je u blockchainu kryptoměn, které používají soukromý klíč pro šifrování peněženky a pro přístup do ní. Veřejný klíč používají ostatní uživatelé pro získání adresy peněženky a následné zaslaní kryptoměny. [46] [47] [48]

Asymetrická kryptografie se mimo jiné v blockchainech používá i pro zajištění bezpečného přenosu symetrických klíčů, kterými se poté šifruje komunikace mezi dvěma uživateli. Tento proces se využívá častěji v soukromých nebo konsorciálních blockchainech. Symetrická kryptografie se tedy využívá pro zajištění bezpečnosti komunikace mezi dvěma body v blockchainové síti. [47] [48]

Veřejné blockchainy, ale i některé soukromé mohou používat algoritmy pro ověření přidaných bloků do sítě. Například Bitcoin používá algoritmus PoW, Hyperledger Fabric využívá algoritmus Kafka a Ethereum využívá PoS. Tyto algoritmy vyžadují spoluúčast uživatelů sítě, například zapůjčením jejich výpočetního výkonu, nebo peněžitého vkladu jako záruku za správné ověření. V soukromých typech blockchainů se ale často setkáme spíše již s nějakým typem inteligentní smlouvy, která tento proces zjednoduší a zautomatizuje. [44] [49] [50]

Jelikož bloky v blockchainech kryptoměn obsahují vícero transakcí než pouze jednu, využívají se pro jejich ověření Merkle Tree namísto ukládání Hash hodnoty z každé transakce zvlášť. Blockchain Bitcoin, Ethereum ale i Hyperledger Fabric využívají Merkle Tree, nebo jeho různé modifikace. Například blockchain Ethereum využívá modifikovanou verzi Merkle Tree, která se nazývá Merkle Patricia Tree. Oproti klasickému Merkle Tree obsahuje každý blok tři Merkle Tree, kde jeden slouží jako klasický Hash transakcí, další slouží jako ukazovatel stavu transakce a poslední je něco jako účtenka transakcí. [51] [52]

Blockchainová síť	Bitcoin	Ethereum	Hyperledger Fabric
Hash funkce	SHA-256	Keccak256	SHA-256
Digitální podpis = Generování klíčového páru	ECDSA	ECDSA	ECDSA
Konsenzuální mechanizmus	PoW	PoS	Kafka
Šifrování peněženky (uživatelských dat)	AES-256-CBC	AES-128-CTR	AES-256
Hashovaní dat/transakcí	Merkle Tree	Merkle Patricia Tree	Merkle Tree

Obrázek 20 – Používané algoritmy [44] [45] [46] [50] [51] [53] [54] [55]

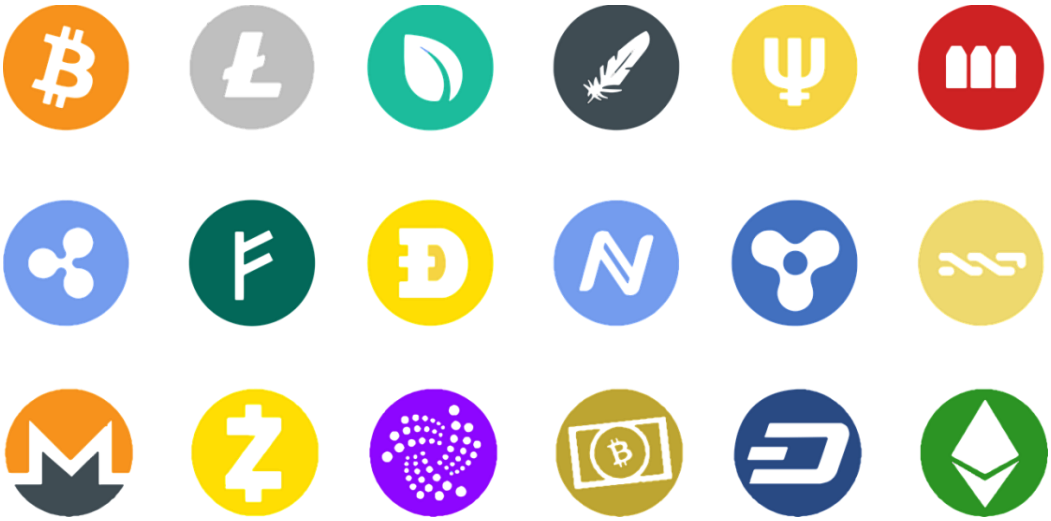
4 VYUŽITÍ TECHNOLOGIE BLOCKCHAIN V PRAXI

Technologie blockchain se proslavila hlavně díky kryptoměnám a převážně díky Bitcoinu a Ethereu. Od té doby se ale vývoj blockchainu výrazně posunul dopředu, a tak se tato technologie rozrostla i do dalších odvětví. Blockchain našel své uplatnění tam, kde je zapotřebí dosáhnout vysoké míry bezpečnosti a transparentnosti a také decentralizaci, nejdůležitější vlastnosti blockchainu. Mezi odvětvími využívající technologii blockchain jsou například digitální měny, zdravotnictví, dodavatelský řetězec, zábavní průmysl nebo zabezpečení informací státu a jejich obyvatelů. [56] [57]

4.1 Kryptoměny

Kryptoměna je typ digitální měny, která využívá různé typy kryptografických algoritmů a šifrování pro zajištění zabezpečení transakcí prováděné s touto měnou. Kryptoměny jsou typu veřejných blockchainů, takže jsou naprosto decentralizované a tím pádem je neovládá žádný centrální subjekt, jako například u fyzické měny centrální banka. Transakce ověřují uživatelé blockchainové sítě. Dříve kryptoměny sloužili spíše jako investice nebo pro placení v určitých částích internetu. V dnešní době jsou kryptoměny rozšířenější, takže je dokonce akceptují jako platbu i v nejrůznějších e-shopech. [58]

Kryptoměny se generují pomocí ověřování transakcí buď pomocí těžby, nebo pomocí validace. Toto záleží na typu ověřovacího algoritmu, který blockchainová síť používá. Hodnota kryptoměn není fyzická, ale pouze digitální a ukládá se v soukromé digitální peněžence. Celková hodnota kryptoměny v oběhu je konečná. Každý pohyb nebo změna v blockchainové síti je nevratně a neměnně zaznamenána. Kryptoměny se dělí do čtyř základních skupin, kterými jsou Altcoiny, Privacy coiny, Stablecoiny a Tokeny. Altcoiny jsou všechny kryptoměny kromě Bitcoinu. Privacy coiny slouží k ještě vyšší ochraně majitele kryptoměny a jeho transakcí. Stablecoiny jsou všechny kryptoměny, které si udržují nějakým způsobem stabilní hodnotu na trhu, nejčastěji tak, že jsou napojené na nějakou komoditu nebo fyzickou měnu. [58] [59]

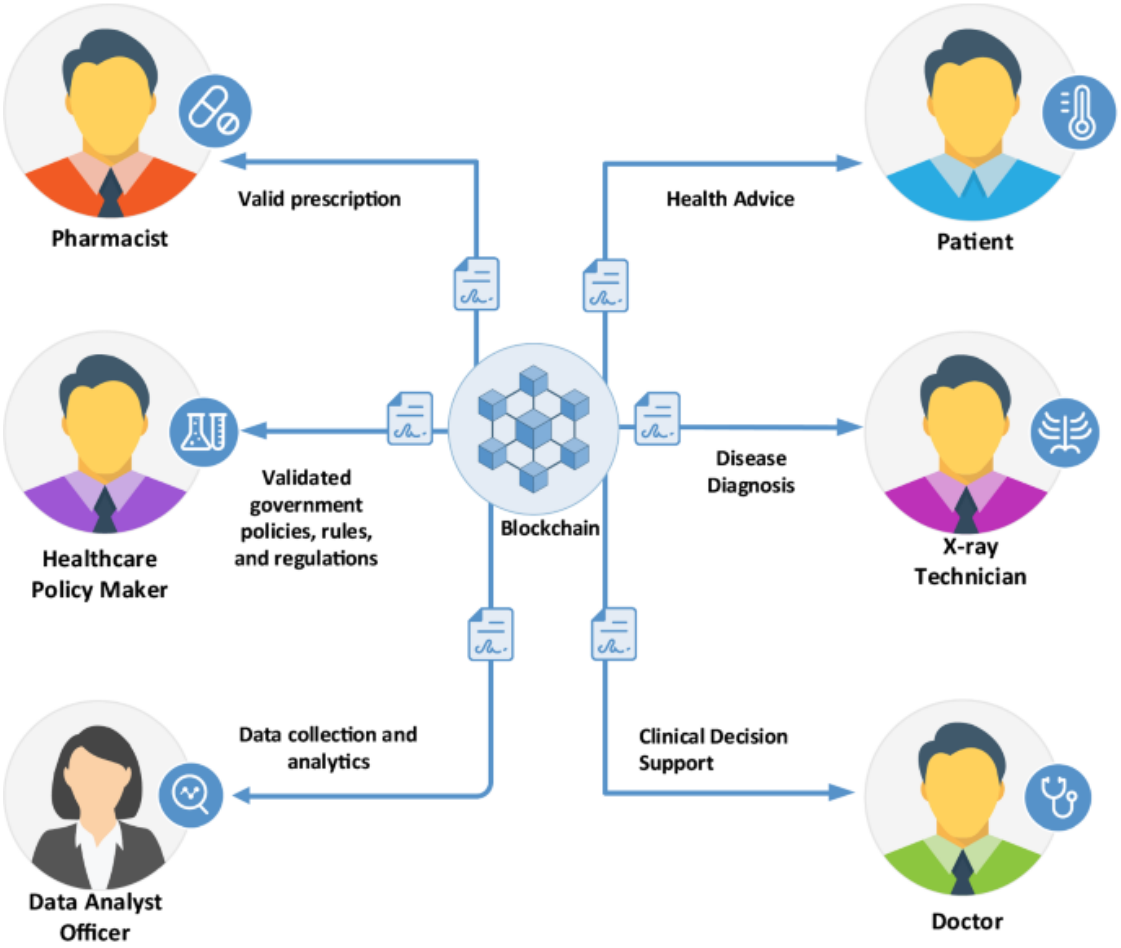


Obrázek 21 – Kryptoměny [59]

4.2 Zdravotnictví

Digitalizace ve zdravotnictví pokročila obrovskými kroky, ale stále je zde vysoká míra centralizace, protože jednotlivé zdravotnické subjekty a zdravotníci používají vytvořené systémy pro zaznamenávání informací o pacientech, které spravují organizace. Tyto systémy většinou mezi sebou nedokáží komunikovat a předávat si informace, takže je zde stále nutnost o manuální zaslání lékařské karty pacienta nebo doručení karty fyzicky na papíře. Technologie blockchain by nejen celý tento proces urychlila a vytvořila by zabezpečenější síť, ale snížila by i celkové náklady oproti ostatním systémům. Každý doktor nebo zdravotnický subjekt by měl okamžitý přístup ke všem důležitým informacím o pacientovi a tím by se urychlila lékařská péče. [60] [61]

Využití ve zdravotnictví není zaměřené pouze na komunikaci mezi doktory, ale dala by se uplatnit i na mezinárodní úrovni ve sledování stavů dostupných léčiv. V dnešní době je časté, že lékárny vydávají namísto předepsaných léků určité alternativy, protože si doktor nemůže zjistit, které léky jsou dostupné a které nikoliv. Zlepšila by se tedy celková komunikace mezi všemi sektory ve zdravotnictví. [60] [61]



Obrázek 22 – Propojení zdravotnictví pomocí blockchainu [62]

4.3 Státní služby

Mnoho státních služeb funguje stále na starém formátu fyzického ukládání dat do zastaralých systémů, které jsou náchylné na prolomení útočníky. Tyto staré systémy jsou navíc centralizované a jejich údržba je stále dražší a dražší. Zároveň je celý systém náchylný na selhání v centrálním bodě, kde je nutné zavést drahé zálohování dat. Počet státních služeb se neustále zvětšuje, a tak je zapotřebí je ideálně nějak propojit, aby měli přístup k požadovaným informacím o občanech. Aby bylo možné jednotlivá data přiřadit k občanům, je nutné, aby každý občan měl svou vlastní digitální identitu. Stát může využít technologii blockchain i na jiné odvětví kromě digitální identity a ukládání dat o občanech. Dalšími možnostmi využití může být hlasování ve volbách, státní administrativa nebo zmírnění korupce ve vládní sféře. [63] [64] [65]

4.4 Dodavatelský řetězec

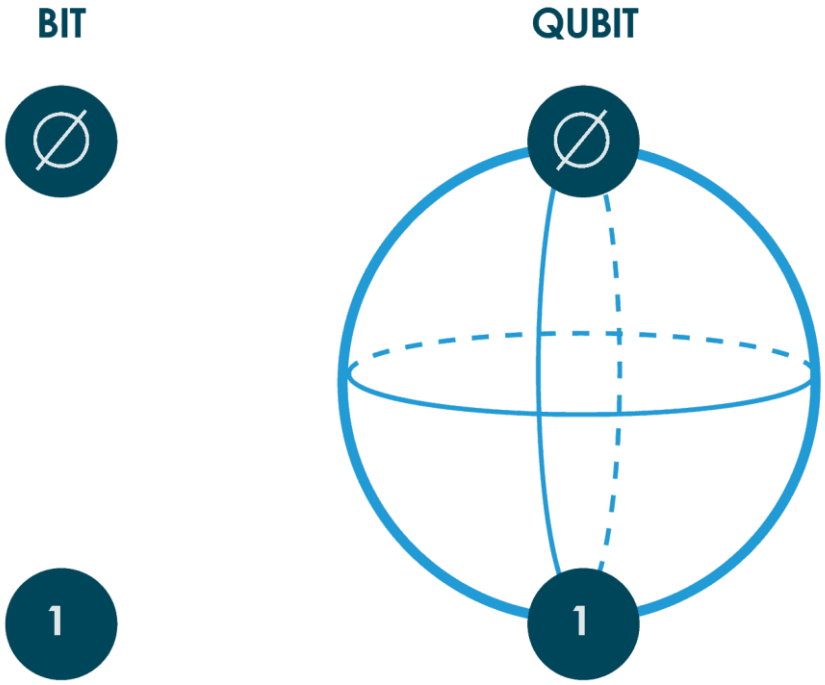
V dnešní době je stále aktuální v dodavatelském řetězci používat klasické papírování, které může kterýkoliv subjekt po cestě, přes které zboží projde, jednoduše přepsat, nebo nějak znehodnotit. Jelikož jsou čím dál větší požadavky na transparentnost, kterou tento systém v plné míře nedisponuje, je zapotřebí vyvinout systém, který bude vyhovovat všem takovýmto požadavkům. Právě zde přichází vhod technologie blockchain, která nabízí řadu výhod, které jsou potřebné při sledování zboží v dodavatelském řetězci od získání materiálů, přes výrobní proces a distributory až k samotnému zákazníkovi. [66] [67]

Technologie blockchain disponuje určitými vlastnostmi, které mohou být velmi užitečné v dodavatelském řetězci. Připojením všech subjektů po cestě získáme výhody jako jsou například: [66] [68]

- **Sledovatelnost –** do sítě blockchain se přidá záznam od každého subjektu, přes které zboží přešlo. Toto nám umožňuje dobrou sledovatelnost zboží až k jeho počátkům. Tuto vlastnost ocení spíše koncoví zákazníci, kteří mohou snadno zjistit například zemi původu zboží
- **Transparentnost –** každý uživatel či subjekt si mohou zkontrolovat všechny informace o zboží, včetně jejich certifikátů
- **Snížení nákladů –** papírování je drahé a časové nákladné. Toto se dá velmi zredukovat, protože stačí zadat data pouze do blockchainové sítě a nutnost tisknutí papírů pro další subjekty nebo uživatele není nutností. Tímto se sníží celkové náklady na logistiku a sníží se časová náročnost o dobu vyřizování těchto papírů
- **Chytré smlouvy –** blockchainové sítě je možné ještě více zrychlit a zpříjemnit pro uživatele i subjekty použitím chytrých smluv. Tyto chytré smlouvy dokáži hlídat zboží jak při přidání, tak i při jakékoliv manipulaci s ním. Celý proces se tímto tedy zautomatizuje a zrychlí

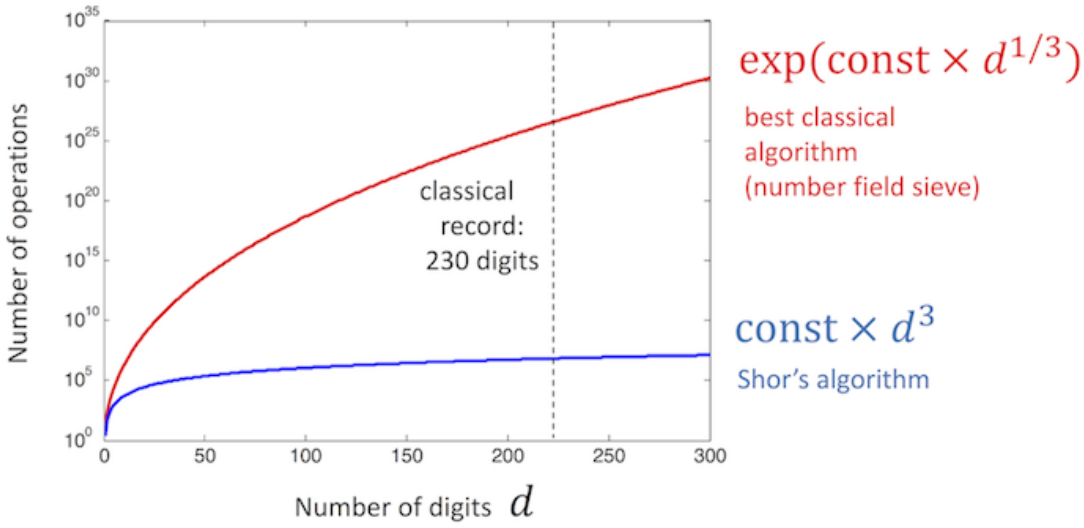
5 KRYPTOGRAFIE V DOBĚ KVANTOVÝCH POČÍTAČŮ

Většina z dnes známých kryptografických algoritmů, by mohli být s nástupem kvantových počítačů a kvantové výpočetní techniky prolomeny, díky vysokému výpočetnímu výkonu. Tento vysoký výpočetní výkon je dosažený právě kvůli využívání kvantové fyziky. Výpočetní výkon stroje je založen na kvantových bitech (Qubit). Dnešní klasické počítače naopak využívají bity. Klasický bit může nabývat hodnoty buď 0 nebo 1 v daný okamžik. Qubit dokáže nabývat těchto dvou hodnot zároveň ve stejnou dobu, a tak se exponenciálně zvyšuje výpočetní výkon. Tento jev se nazývá superpozice, která je jednou ze základních vlastností kvantového počítače. [69] [70]



Obrázek 23 – Bit a Qubit [71]

Nejvýznamnější hrozbu představuje kvantová výpočetní technika pro asymetrickou kryptografii, která je založená na klíčovém páru, které spolu matematicky souvisejí a dají se jeden od druhého matematicky odvodit. V bezpečí nejsou ale ani ostatní typy kryptografie, i přes to, že se některé považují za post-kvantové již dnes. Proti asymetrické kryptografii a kryptografii eliptických křivek (ECC) účinně působí Shorův algoritmus, který je založen na rychlé faktorizaci, rozkladu čísla na součin menších čísel nebo prvočísel. Při pohledu na symetrickou kryptografii má nejvyšší pravděpodobnost uspět Groverův algoritmus, který je založen na velmi rychlém procházení nesetříděné databáze dat za účelem najít hledaný prvek. [72] [73]



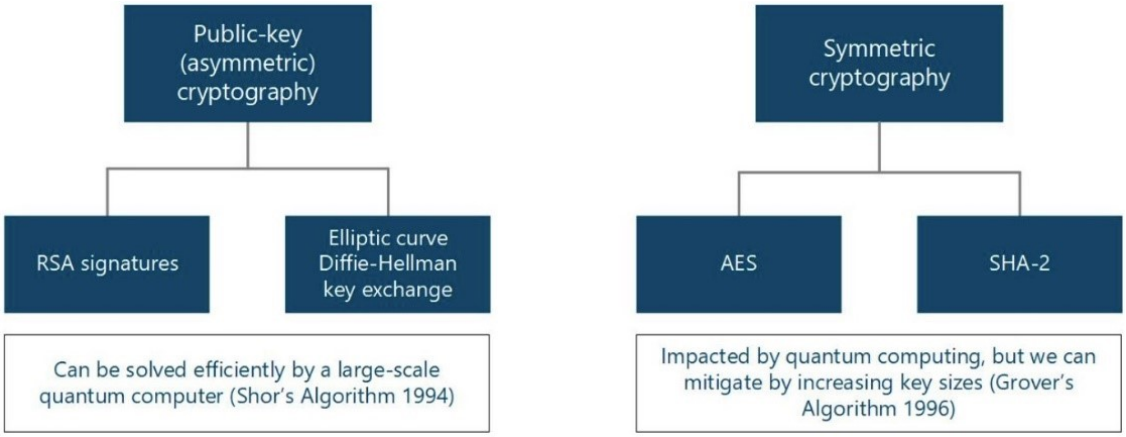
Obrázek 24 – Nejlepší klasický vs. Shorův algoritmus [73]

Jako odpověď na některé známé silné stránky kvantového výpočetního výkonu byli vytvořeny některé post-kvantové algoritmy, které by měli úspěšně odolat útokům pomocí kvantových počítačů. Nejslibnější ze všech vytvořených je kryptografie založená na mřížce. Dalšími slibnými algoritmy jsou kryptografie založená na Hashi, vícerozměrná kryptografie nebo kryptografie založená na kódech. [71] [72]

5.1 Symetrické a asymetrické šifrování

Symetrické šifry jsou obecně bezpečnější proti útokům kvantových počítačů než asymetrické, protože jediná možnost, jak zjistit klíč k šifrování a dešifrování dat zůstává i nadále buď klíč uhodnout nebo jej získat od vlastníka. Závažnější hrozbu představuje Gloverův algoritmus, který zkracuje dobu uhodnutí náhodného klíče zhruba na polovinu. Teoreticky se dá uvažovat, že například šifrování AES-256 bude mít stejnou úroveň zabezpečení v případě kvantových počítačů jako AES-128 proti dnešním počítačům, což se považuje za stále velmi bezpečné. V tomto případě je možné zesílení bezpečnosti šifrování pomocí několikanásobného zvětšení velikosti klíče. [74] [75]

Asymetrické šifry jsou na tom o dost hůře, protože obsahují klíčový pár, kde jeden z klíčů je veřejný. Spoléhají se při tom na složité matematické funkce a násobení velmi velkých prvočísel. Dnešní počítače nemají dostatečný výkon na takovéto výpočty. Tento problém bude velmi snadno řešitelný pomocí Shorova algoritmu, který dokáže takovéto čísla najít velmi rychle. Zranitelnost asymetrického šifrování ale ovlivní i symetrické šifrování, protože se nejčastěji používají obě metody společně. Pomocí asymetrického šifrování si uživatelé předají symetrický klíč, který posléze používají k šifrování a dešifrování dat. [74] [75]



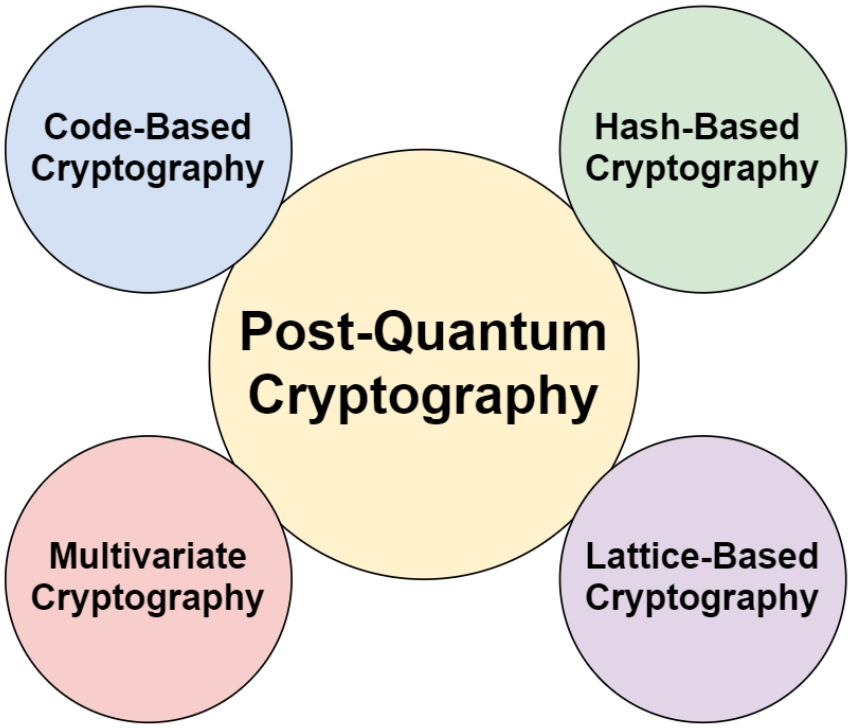
Obrázek 25 – Dopad kvantových počítačů na šifrování [74]

5.2 Hash funkce

Hashovací funkce jsou na tom velmi podobně, jako symetrické šifrování. Hlavní zranitelností může být uhodnutí originální zprávy stylem pokus omyl, protože vytvořit reverzní funkci je velmi složité a ve své podstatě i nemožné. Jelikož Hash funkce vytvářejí výstup o pevné velikosti, tak jsou také náchylné proti Groverově algoritmu. Stejně jako u symetrického šifrování jsou Hashovací funkce o délce výstupu 256 bitů stále velmi bezpečné i v případě kvantových počítačů. Pokud by ale byla potřeba zvýšit bezpečnost, tak jedna z možností je použít Hashovací funkci s delším výstupem, například 384 nebo 512 bitů. [55] [76]

5.3 Post-kvantová kryptografie

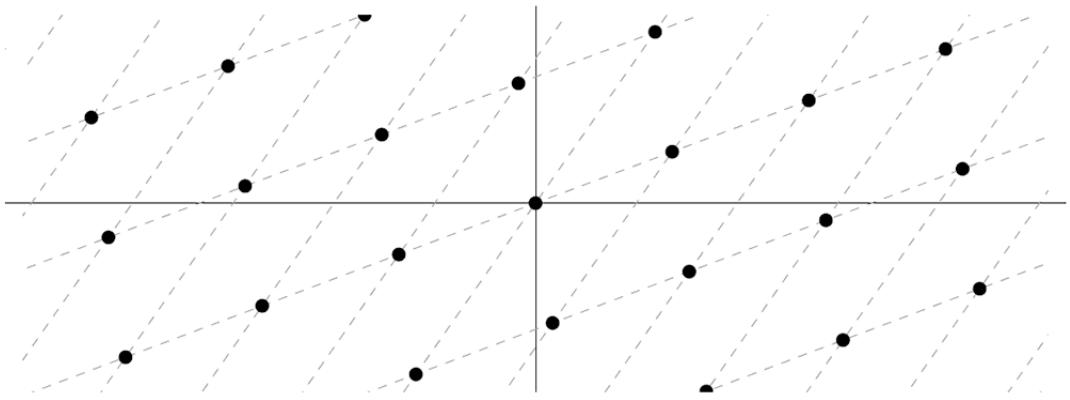
Post-kvantová kryptografie, označována také jako kvantově odolná nebo kvantově bezpečná, je speciální vývoj kryptografie na dnešních počítačích, které by měli být bezpečné i v době kvantových počítačů. Jelikož kvantové počítače nejsou ještě normálně dostupné a neustále se vyvíjí, tak nelze jistě zaručit bezpečnost všech dnešních post-kvantových algoritmů. Například Shorův algoritmus již byl úspěšně vyzkoušený na kvantovém počítači, kde úspěšně našel činitele z čísla 15. Dalším významným krokem kvantových počítačů v oblasti výpočetního výkonu je z měření od firmy Google. Jejich kvantový počítač dokázal provést velmi obtížnou matematickou operaci za 200 sekund, která by nejvýkonnějšímu superpočítači v té době trvala přibližně 10 000 let. Tento výzkum jen dokazuje, jak kvantové počítače dokážou být výkonné a že dokáží provádět i takové výpočty, které jsou pro dnešní počítače v reálném čase nemožné. [70] [77]



Obrázek 26 – Typy post-kvantové kryptografie [77]

5.3.1 Lattice-Based Cryptography (Kryptografie založená na mřížce)

Kryptografie založená na mřížce využívá k šifrování a dešifrování dat složité matematické problémy založené na geometrických mřížkách. Tato mřížka vypadá stejně jako reálná mřížka, ale oproti té reálné je nekonečně velká, protože se udává akorát tvar, jakým se mřížka generuje do nekonečna a lze použít jakoukoliv část mřížky. Mezi nejslibnější algoritmy založené na kryptografii založené na mřížce jsou CRYSTALS-Dilithium jako algoritmus pro digitální podpisy a CRYSTALS-KYBER jako algoritmus pro šifrování s veřejným klíčem. [78] [79]



Obrázek 27 – Příklad mřížky u kryptografie založené na mřížce [79]

Správné fungování zajišťují tři základní pojmy, mřížka, vektor a báze: [78]

- **Mřížka –** nekonečná množina bodů, může být periodická, ale i náhodná
- **Vektor –** bod v mřížce. Jeho poloha je určená souřadnicemi
- **Báze –** počítače nedokáží uložit celou mřížku do paměti, protože nemají nekonečně velikou paměť, proto se používá báze, která za pomocí vektorů určuje počátek a velikost uložené mřížky do paměti

Flexibilita, snadná implementace a různorodost použití mřížek jsou základními výhodami kryptografie založené na mřížkách. Právě díky flexibilitě a snadné implementaci je tento typ kryptografie snadno dostupný a levný. I přes to, že jsou založené na velmi složitých matematických problémech, tak jsou ve své podstatě velmi srozumitelné, protože nevyžadují velmi širokou základnu matematických znalostí pro pochopení základního konceptu. Protože jsou výpočty na úrovni mřížek rychlé tak tyto algoritmy vykazují vysoký výpočetní výkony a nízkou spotřebu energie. [78]

5.3.2 Hash-Based Cryptography (Kryptografie založená na Hashi)

Kryptografie založená na Hashi využívá vlastnosti Hashe, aby se zaručila bezpečnost v době kvantových počítačů. Jedná se o jeden z typů post-kvantové kryptografie, který má sloužit primárně na podepisování digitálních dokumentů. Prvním algoritmem pro podepisování dokumentů využívající Hashovací funkce byl Merkle Tree. Od jeho vzniku bylo vynalezeno mnoho algoritmů, kde nejslibnější jsou eXtended Merkle Signature Scheme (XMSS), FALCON nebo SPHINCS+. Výhodami kryptografie založené na Hashi jsou využití již známé, přesto velmi bezpečné metody Hashování. Díky tomuto není zapotřebí nutně nový hardware pro správnou funkcionalitu algoritmů. [80] [81]

5.3.3 Code-Based Cryptography (Kryptografie založená na kódech)

Kryptografie založená na kódech je typ kryptografie, která obsahuje kódy, které dokážou detekovat chyby v přenesených datech a opravit je. Nejznámější metoda kryptografie založené na kódech je McEliecova metoda. Tato metoda se spoléhá na to, že do původní zašifrované zprávy vloží v určité míře šum, který znehodnotí původní zašifrované data. McEliecova metoda používá speciální binární kód Goppa, který společně jako součin s lineární transformací dává veřejný klíč. Tento klíč je mnohem větší oproti algoritmu RSA, kterému je principiálně podobný. Navíc binární kód Goppa dokáže odstranit šum ze zašifrované zprávy. [82]

5.3.4 Multivariate Cryptography (Vícerozměrná kryptografie)

Vícerozměrná kryptografie je určitá forma asymetrické kryptografie, protože využívá klíčový pár pro šifrování a dešifrování. Oproti klasické asymetrické kryptografii je ale založena na vícerozměrnými polynomiálními rovnicemi nad konečnými poli. Oba klíče se skládají ze soustavy polynomiálních rovnic, kde soukromý klíč obsahuje navíc ještě informace k řešení těchto polynomiálních rovnic. [83]

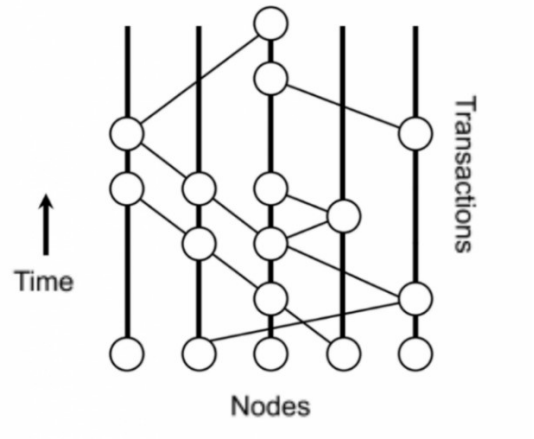
5.4 Kvantově odolný blockchain

Příchod kvantových počítačů může technologii blockchain uškodit, ale i přilepšit. Úkolem všech existujících i nových blockchainů bude zajistit zabezpečení proti útokům pomocí kvantových počítačů. Protože většina dnešních známých blockchainů využívá dnešní kryptografické algoritmy, mezi kterými je nejpoužívanější asymetrické šifrování. Tento typ kryptografie je podle všech dostupných informací a měření náchylný proti útokům ze strany kvantových počítačů, takže je zapotřebí pozměnit nebo implementovat nové zabezpečení. I když některé blockchainy už používají algoritmy, které se považují za kvantově bezpečné, jako je například Merkle Tree, tak stále bude s největší pravděpodobností potřeba zvětšit velikost výsledné Hashovací funkce na několikanásobek. Další obavou kvantových počítačů je jejich vysoký výpočetní výkon. Většina dnešních blockchainů se používá v oblasti kryptoměn, kde je zapotřebí veliké množství uživatelů s výkonnými stroji na ověřování transakcí. Jelikož kvantové počítače mají být několikanásobně výkonnější, je zde riziko, že by mohli nahradit dnešní počítače a tím potencionálně převzít kontrolu nad veřejnými blockchainy, které využívají algoritmus ověřování PoW. Tento problém aktivně řeší druhý z algoritmů ověřování PoS, na který už přešli některé z kryptoměn, jako například jedna z nejznámějších kryptoměn Ethereum. [84] [85] [86]

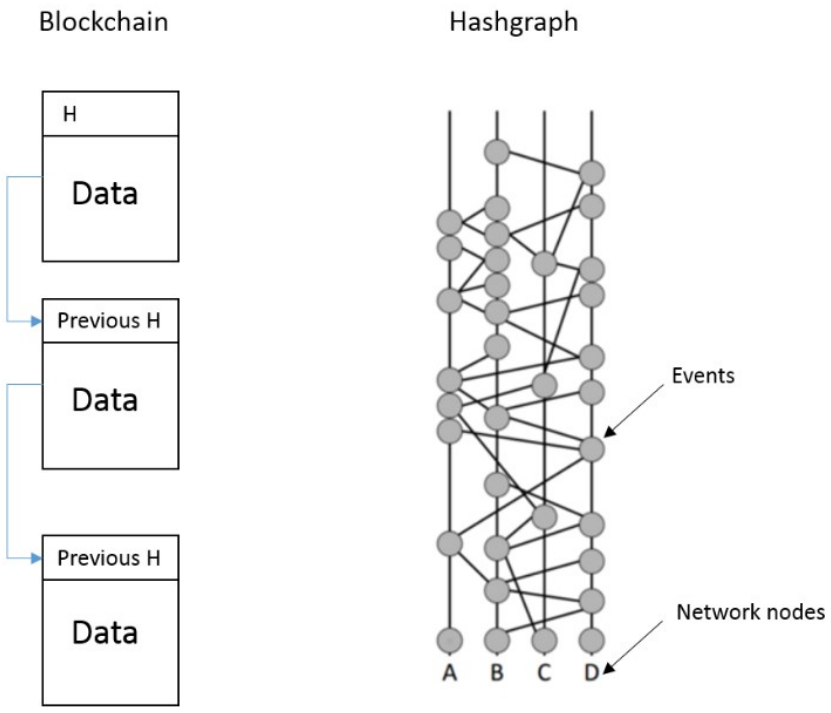
Vysoký výpočetní výkon kvantových počítačů může být ale technologii blockchain i přínosný. Vyšší výkon značně zlepšuje vlastnosti v oblasti vytváření a ověřování bloků a celkové přenosové rychlosti v celé síti blockchain. V případě blockchainů kryptoměn, by toto navýšení rychlosti sítě umožňovalo rychlejší přidávání a ověřování transakcí uživateli sítě. Již dnes existují některé typy blockchainů, které jsou považované za post-kvantové. Mezi ně se řadí například Hashgraph nebo Quantum Resistant Ledger (QRL) [84] [85] [86]

5.4.1 Hashgraph

Hashgraph je speciální typ blockchainu, který používá speciální přístup k ukládání dat a k jejich následnému ověřování. Hashgraph totiž neukládá data do řetězce chronologicky za sebou, ale využívá grafickou strukturu, kde jsou data uložená jako události. Událost obsahuje Hashe z aktuální události a předchozí dvě transakce. Jednotlivé události jsou mezi sebou propojené, čím se ověřují jednotlivé bloky. Ověření je platné pouze v případě, že jej více než $^{2}/_{3}$ uzlů zaznamenalo a schválilo. Tento model nám také umožňuje paralelní přidávání vícero událostí zároveň. [87] [88]



Obrázek 28 – Hashgraph [87]



Obrázek 29 – Porovnání Hashgraph a Blockchain [87]

Nelze jednoznačně určit, jestli je lepší Hashgraph nebo typický blockchain, protože vždy záleží na typu použití a jaké vlastnosti od systému očekáváme. Blockchain je výhodnější pro aplikace, kde je zapotřebí vysoká úroveň decentralizace a zabezpečení. Toto využijeme například ve zdravotnictví, kde je potřeba ochránit data pacientů. Naopak Hashgraph se vyplatí více v aplikacích, kde je zapotřebí vyšší rychlost přidávání dat a celková rychlost sítě. Příkladem může být distribuovaná účetní kniha pro pohyb financí, například v bankovnictví nebo energetice. Mezi značné výhody Hashgraphu můžeme zařadit následující: [87] [88] [89]

- **Rychlost –** jelikož je umožněno přidávat události paralelně, zvýší se tím rychlost a zároveň i efektivita celé sítě
- **Rychlé ověření událostí –** díky speciálním ověřovacím algoritmům umožňuje Hashgraph velmi rychlé ověření přidaných událostí

Jako hlavní nevýhodou oproti blockchainu je pouze částečná decentralizace. Většinou Hashgraph používá pouze pár jedinců, kteří jsou dopředu vybráni a celý systém spravuje buď jedna organizace nebo nějaká skupina. Vývoj Hashgraphu směřuje k tomu, aby byl tento systém v budoucnu co nejvíce decentralizovaný. [87] [89]

5.4.2 Quantum Resistant Ledger (QRL)

QRL je jedním z mála kvantově odolných typů technologie blockchain. Svoji úroveň zabezpečení zajišťuje používáním podpisového schématu XMSS. Společně s XMSS používá pro zabezpečení klíčů technologii založenou na mřížkách. Skupina, která vytvořila QRL vydala mimo jiné i vlastní kryptoměnu se stejným názvem QRL, která je podle tvůrců první post-kvantovou kryptoměnu. Díky velmi dobré a rozsáhlé dokumentaci je vývoj aplikací na QRL snadná a velmi rychlá. [84] [85]

II. PRAKTICKÁ ČÁST

6 POUŽITÉ TECHNOLOGIE

Pro zpracování praktické části byl vybrán programovací jazyk Python 3.11, software pro grafické zobrazení Graphviz a software na vytváření jednoduchých grafických uživatelských rozhraní (GUI) QT Designer. Jako vývojové prostředí jsem zvolil PyCharm Community od JetBrains.

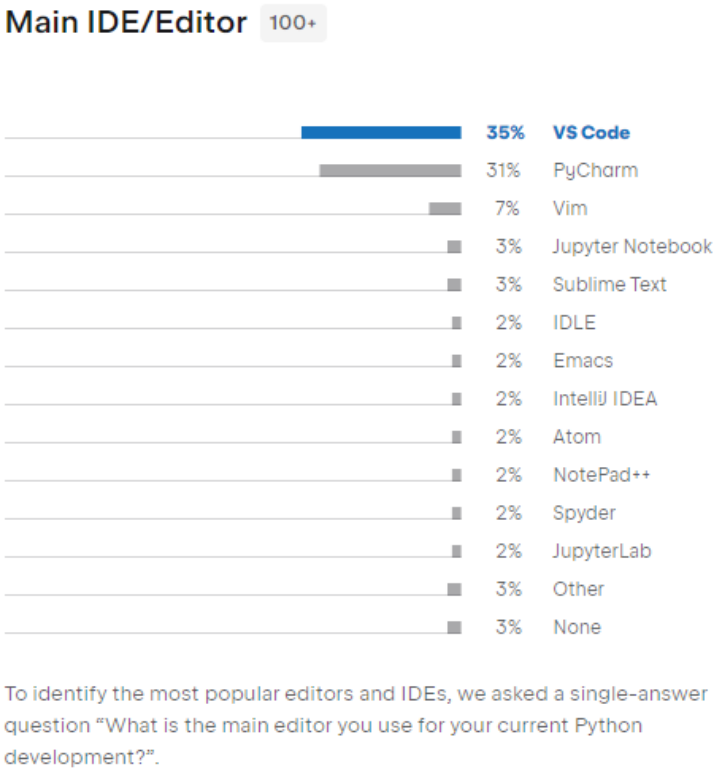
6.1 Programovací jazyk Python

Programovací jazyk Python jsem zvolil, kvůli jeho jednoduchosti, množství použitelných knihoven, které dokáži velmi ulehčit práci, a hlavně kvůli mé oblíbenosti. Python je i velmi oblíbený ve společnosti, a proto lze na internetu najít mnoho zajímavých věcí, návodů a dokumentací které při programování v jazyce Python použít.

Díky množství knihoven, kterými Python disponuje, je velmi snadné vytvářet a upravovat různé textové dokumenty, zpracovávat data a upravovat je podle potřeby, které jsou pro mojí praktickou část důležité. Navíc všechny knihovny mají kvalitní dokumentaci a na internetu je mnoho ukázek, ve kterých lze najít řešení v podstatě na každý problém. Zvolil jsem nejaktuálnější verzi Python 3.11, protože je přibližně o 10-60% rychlejší než předchozí verze 3.10 [90].

6.2 Vývojové prostředí

Jako vývojové prostředí jsem zvolil Pycharm Community od firmy JetBrains. Jedná se o jedno z neznámějších a nejoblíbenějších vývojových prostředí pro programování v programovacím jazyce Python. Z průzkumu z roku 2021 od firmy JetBrains vyplývá, že oblíbenější než PyCharm, bylo jen vývojové prostředí VS Code od firmy Microsoft. Dohromady tyto dvě vývojová prostředí dají skoro 70% využívání na trhu mezi programátory. [91].

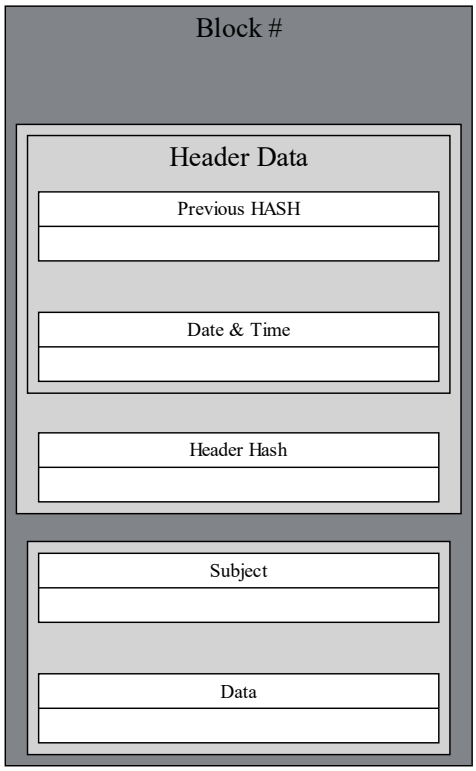


Obrázek 30 – Průzkum oblíbenosti vývojových prostředí pro Python [91]

6.3 Grafická vizualizace pomocí Graphviz

Graphviz je open source software, který slouží pro generování nejrůznějších diagramů a grafů. Grafy a diagramy se navrhují v programovacím jazyku DOT, který obsahuje pár hlavních klíčových slov, jako jsou například graph, subgraph a node. Tyto hlavní klíčová slova slouží k vytvoření hlavní struktury celého grafu nebo diagramu. Jednotlivé části jde vizuálně upravovat podle potřeby pomocí atributů. Pomocí atributů lze měnit například barvu pozadí i textu, velikost a styl písma, velikost jednotlivých buněk a mnoho dalšího.

Jelikož jsem potřeboval vytvořit velmi specifickou kostru pro zobrazení jednotlivých bloků v naprogramovaném blockchainu, tak byl tento software výbornou možností díky jeho rozsáhlým možnostem. Software Graphviz dokáže vygenerovat obrázky výstupu v mnoha formátech, ze kterých mi nejvíce vyhovoval formát SVG (Scalable Vector Graphics), právě kvůli různým velikostem jednotlivých bloků a škálovatelnosti celého blockchainu. Formát SVG umožňuje přibližovat či oddalovat výsledný obrázek bez jakékoliv degradace kvality. To je výhodné, pokud je v blockchainu mnoho bloků, který díky tomu nabere na velikosti.



Obrázek 31 – Kostra bloku v blockchainu (vlastní zdroj)

6.4 QT Designer

Pro tvorbu GUI pro můj projekt byla jasná volba QT Designer, jelikož se jedná o tvoření GUI za pomocí grafického rozhraní. Ovládání a tvoření jednotlivých GUI je velmi jednoduché a ušetří mnoho práce a času oproti vytváření GUI za pomocí kódu. Vytvořená GUI jsem používal v kombinaci s Python knihovnou PyQt6. QT Designer obsahuje mnoho různých widgetů, pomocí kterých můžeme vytvořit velmi uživatelsky přívětivé GUI. V nabídce widgetů nalezneme například widgety pro zobrazení obsahu či textu, widgety tlačítkové, widgety pro uživatelské vstupy, kontejnery pro rozdělení obsahu na více stránek nebo například widgety pro uspořádání ostatních widgetů na obrazovce.

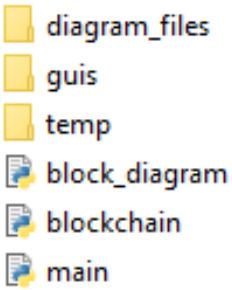
Pro moji tvorbu tří jednodušších GUI stačili pouze základní widgety, kterými jsou QLabel, QPushButton, QTextEdit, QLineEdit QMenuBar se dvěma QAction a Line. První GUI slouží pro vytvoření nového nebo otevření již existujícího blockchainu. Druhé GUI slouží pouze pro zadání názvu nového blockchainu. Třetí a poslední GUI slouží pro přidávání bloků do blockchainu.

7 IMPLEMENTACE BLOCKCHAINU

Celý projekt je strukturován do tří Python skriptů a tří složek, ze kterých dvě slouží jako složky pro výstupní data. Ve třetí zmíněné složce s názvem guis se nachází tři soubory pro grafické rozhraní aplikace. Soubor startwindow_gui.ui je grafické rozhraní pro úvodní obrazovku při zapnutí aplikace a slouží ke zvolení možnosti buď na vytvoření nového blockchainu, nebo k otevření již existujícího blockchainu. Účel druhého souboru new_blockchain_gui.ui je pouze ke stanovení jména nového blockchainu. Poslední a hlavní soubor blockchain_gui.ui je hlavní okno aplikace. Slouží k přidávání jednotlivých bloků do blockchainu, vytváření a otevírání jednotlivých blockchainů přes horní lištu aplikace.

Obě výstupní složky slouží pro výstup obrázků a zdrojových kódů diagramů. Zatímco ve složce temp se nachází pouze vždy poslední přidaný blok do jakéhokoliv blockchainu, tak složka diagram_files obsahuje data ke každému vytvořenému blockchainu. Každý vytvořený blockchain vytvoří stejnojmennou složku se zdrojovým kódem diagramu a samotný diagram celého blockchainu.

Hlavní skript main.py, slouží k ovládání aplikace, zpracování uživatelského vstupu a přepínání jednotlivých GUI. Navíc se zde uchovávají data aktuálně používaného blockchainu. Druhý skript blockchain.py obsahuje všechnu logiku vytváření blockchainu, vytváření bloků a přidávání vytvořených bloků do aktuálně otevřeného blockchainu. Ovládá i vytváření diagramů, kde volá funkce z posledního skriptu block_diagram.py. Skript obsahuje pouze funkce pro vytváření diagramů a úpravu dat zobrazených v jednotlivých blocích v diagramu.



Obrázek 32 – Struktura projektu (vlastní zdroj)

7.1 Grafické rozhraní aplikace

Po spuštění aplikace se zavolá podmínka, která zkontroluje, jestli je skript spuštěn jako hlavní program. Pokud je, tak se vytvoří proměnná aplikace, nastaví se grafické rozhraní, které se má zobrazit a aplikace se spustí.

```
# Check if script is run as main program
if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)  # Create an application
    window = StartWindow()  # Set starting window
    window.show()  # Show window GUI
    sys.exit(app.exec())  # Execute application
```

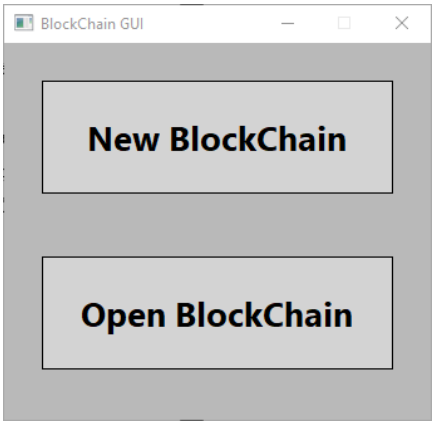
Obrázek 33 – Kontrola, jestli skript je spuštěn jako hlavní program (vlastní zdroj)

7.1.1 Úvodní okno aplikace

Jako úvodní grafické rozhraní je zvolená třída StartWindow, u které se při jejím zavolání zavolá inicializační funkce. Úvodní inicializace slouží k nastavení velikosti zobrazeného okna grafického rozhraní, propojení jednotlivých tlačítek s funkcemi a případné zavolání jiných potřebných funkcí.

```
def __init__(self):
    """ First initialization and linking function in window GUI """
    self.window = None
    QtWidgets.QMainWindow.__init__(self)  # Initialization function of the parent QMainWindow class
    Ui_MainWindow.__init__(self)  # Initialization function of the Ui_MainWindow class to set up GUI elements
    self.setupUi(self)  # setupUi function to initialize the UI
    self.setFixedSize(340, 300)  # Set window size
    self.label_openWarning.hide()  # Hide warning
    # Link buttons to functions
    self.pushButton_newBC.clicked.connect(self.create_new)
    self.pushButton_openBC.clicked.connect(self.open_file)
    create_folders()
```

Obrázek 34 – Úvodní inicializace třídy StartWindow (vlastní zdroj)



Obrázek 35 – Grafické rozhraní úvodního okna (vlastní zdroj)

Na obrázku 34 vidíme kód úvodní inicializace třídy StartWindow, která slouží k nastavení grafického zobrazení úvodního okna, které lze vidět na obrázku 35. Z kódu vyčteme, že rozlišení okna aplikace je 340 pixelů na šířku a 300 pixelů na výšku. Dále se dočteme, že

upozornění, které signalizuje nesprávný soubor, má být skryto. Toto upozornění se zobrazí přesně mezi oběma tlačítky ve chvíli, když se uživatel pokusí otevřít nesprávný soubor. Následující dva řádky stanovují, které funkce se budou volat při stisku tlačítka. Poslední řádek je statická funkce, která pouze zkontroluje, jestli existují obě složky pro výstupní obrázky diagramů a v případě jejích absence je vytvoří. Její kód lze vidět na obrázku 36.

```
def create_folders():
    """ Creates folders if needed """
    if not os.path.exists("diagram_files"):
        os.mkdir("diagram_files")
    if not os.path.exists("temp"):
        os.mkdir("temp")
```

Obrázek 36 – Kontrola existence výstupních složek (vlastní zdroj)

```
def open_file(self):
    """ Open existing blockchain """
    home_dir = os.path.dirname(os.path.abspath(__file__))  # Current file directory
    # Open file dialog
    file_name = QFileDialog.getOpenFileName(self, "Select Directory", home_dir, "DOT File (*.dot)")
    print(f'Opening file: {file_name[0]}')  # Log
    file_name_short = os.path.basename(file_name[0][:-4])  # Only name of file without extension

    if os.path.getsize(file_name[0] or file_name[0] == ""):  # Check file size
        # Store file first line
        file = open(file_name[0], 'r')
        line = file.readline()
        file.close()
        # Check file first line
        if re.search(r'^#\[.', line) and re.search(r']$', line):
            self.label_openWarning.hide()  # Hide warning
            print(f'File {file_name[0]} is not empty')  # Log
            if self.window is None:
                self.window = BlockChainGUI(file_name_short)  # Switch window GUI
            self.close()  # Close old GUI
            self.window.show()  # Open new GUI
        else:  # File is wrong
            print(f'File {file_name[0]} is wrong')  # Log
            self.label_openWarning.show()  # Show warning
    else:  # Selected file is empty
        print(f'File {file_name[0]} is empty')  # Log
        self.label_openWarning.show()  # Show warning
```

Obrázek 37 – Otevření blockchainu z úvodní obrazovky (vlastní zdroj)

Na obrázku 37 vidíme funkci, která se zavolá v případě, že stiskneme tlačítko pro otevření, již existujícího blockchainu. Funkce otevře dialogové okno pro otevření souboru. Dialogové okno se otevře ve složce, kde je aktuální Python skript spuštěn. Tuto absolutní cestu dostaneme pomocí třetího řádku funkce. Třetí a poslední parametr upřesňuje koncovou příponou souboru, které uživatel může vybrat. V tomto případě může uživatel vybrat pouze soubory s příponou DOT. Dialogové okno vrátí tuple s absolutní cestou ke zvolenému souboru a jeho koncovku z nabízených ve třetím parametru.

Po otevření souboru otestujeme, jestli je zvolený soubor validní pomocí otestování velikosti souboru. Pokud by byl soubor prázdný, nebo by uživatel nevybral soubor žádný, mezi tlačítky se objeví již zmíněné varování o špatném souboru. V případě že je zvolený

soubor validní, zobrazíme grafické rozhraní BlockChainGUI pro přidávání jednotlivých bloků. Při změně okna předáme parametr s názvem otevřeného souboru bez přípony.

```
class StartWindow(QtWidgets.QMainWindow, Ui_StartWindow):
    def create_new(self):
        """ Switch GUI to another one where blockchain name is selected """
        if self.window is None:
            self.window = CreateNewBlockChain()  # Switch window GUI
        self.close()  # Close old GUI
        self.window.show()  # Open new GUI
```

Obrázek 38 – Vytvoření nového blockchainu z úvodní obrazovky (vlastní zdroj)

Na obrázku 38 vidíme možnost vytvoření nového blockchainu. Tato funkce nám akorát změní grafické rozhraní na rozhraní CreateNewBlockChain, pro stanovení názvu nového blockchainu.

7.1.2 Vytvoření nového blockchainu

Okno aplikace pro vytvoření nového blockchainu se skládá pouze z textového pole a tlačítka pro potvrzení názvu nového blockchainu.

Obrázek 39 – Grafické rozhraní při vytváření nového blockchainu (vlastní zdroj)

```
class CreateNewBlockChain(QtWidgets.QMainWindow, Ui_CreateNewWindow):
    def create_blockchain(self):
        """ Create blockchain with given name """
        name = self.lineEdit_blockChainName.text()  # Get the name from line edit element
        if name.replace(" ", "") == "":  # Name is not valid
            print("Enter valid name!")
            self.label_newWarning.show()  # Show warning
        else:  # Name is valid
            name = name.strip().replace(" ", "_")  # Replace spaces with underlines
            self.label_newWarning.hide()  # Hide warning
            if self.window is None:
                self.window = BlockChainGUI(name)  # Switch window GUI
            self.close()  # Close old GUI
            self.window.show()  # Open new GUI

    def __init__(self):
        """ First initialization and linking function in window GUI """
        self.window = None
        QtWidgets.QMainWindow.__init__(self)  # Initialization function of the parent QMainWindow class
        Ui_MainWindow.__init__(self)  # Initialization function of the Ui_MainWindow class to set up GUI elements
        self.setupUi(self)  # setupUi function to initialize the UI
        self.setFixedSize(340, 300)  # Set window size
        self.label_newWarning.hide()  # Hide warning
        self.pushButton_submitCreateNewBC.clicked.connect(self.create_blockchain)  # Link button to function
```

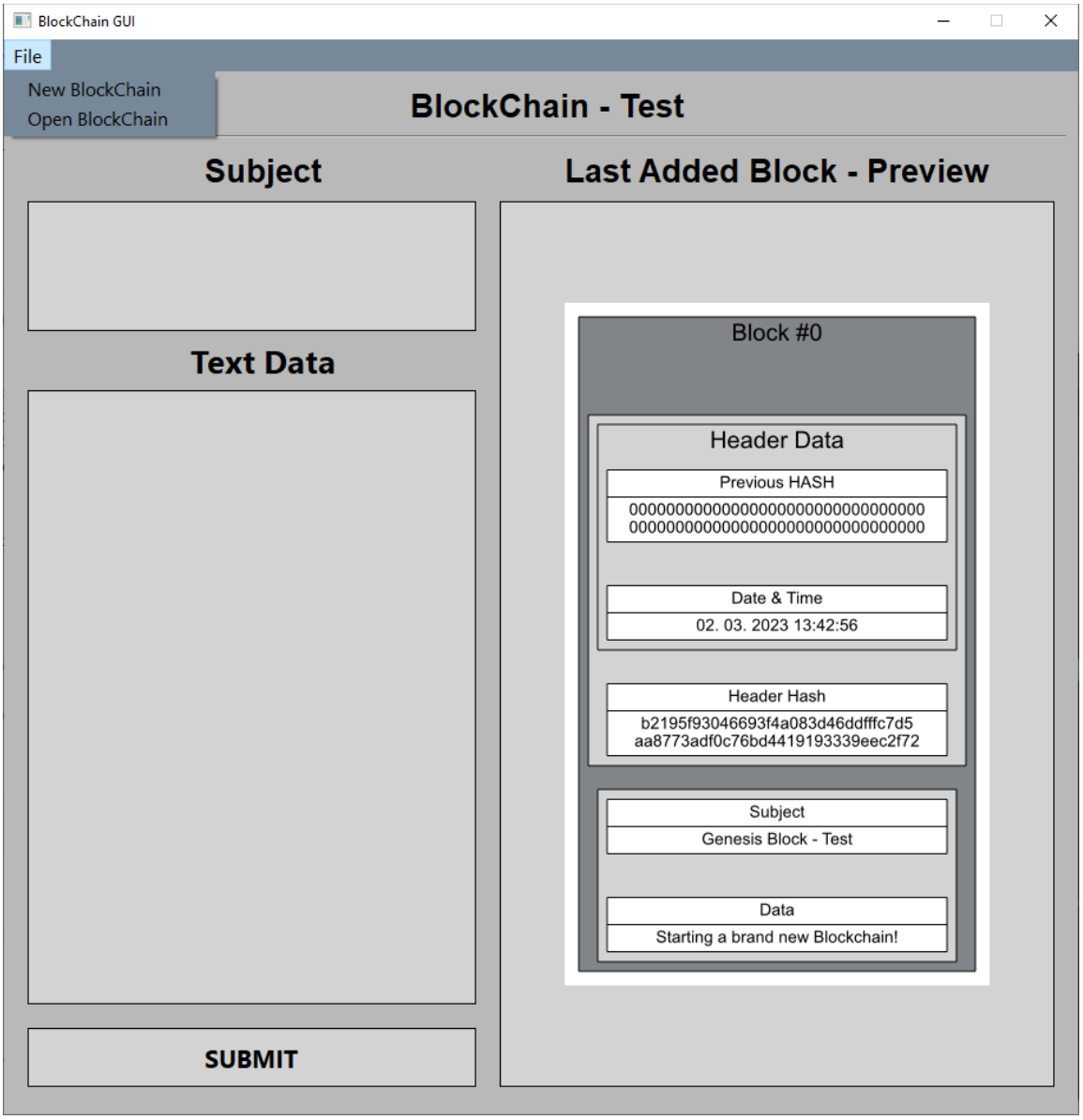
Obrázek 40 – Zdrojový kód pro GUI vytvoření nového blockchainu (vlastní zdroj)

Obrázek 38 zobrazuje vzhled okna aplikace pro vytvoření nového blockchainu. Okno obsahuje pouze textové pole, pro zadávání názvu nového blockchainu, a tlačítko pro potvrzení vytvoření. Na obrázku 40 je celý zdrojový kód, který ovládá okno aplikace na obrázku 39. Úvodní inicializace třídy CreateNewBlockChain je téměř totožná jako při úvodním okně aplikace na obrázku 34. Jediným rozdílem je, že namísto dvou tlačítek se napojuje na funkci pouze jedno tlačítko.

Při stisku tlačítka se zavolá funkce pro vytvoření nového blockchainu. Funkce si uloží uživatelský vstup z textového pole pro název blockchainu, a otestuje, jestli není uživatelský vstup prázdný, nebo jestli neobsahuje jen samé mezery. V tomto případě se akorát v okně aplikace zobrazí varování o nepovoleném názvu. V případě validního názvu blockchainu se odstraní všechny mezery před a za názvem a poté se všechny mezery nahradí podtržítkem. Následně se změní grafické rozhraní na BlockChainGUI s názvem blockchainu jako parametrem a zobrazíme hlavní okno aplikace pro přidávání bloků do blockchainu.

7.1.3 Hlavní okno aplikace

Hlavní okno aplikace slouží pro přidávání bloků do blockchainu, se kterým uživatel právě pracuje. Dále může i založit nový blockchain nebo otevřít již existující. Po vytvoření se v okně aplikace vždy zobrazí právě vytvořený blok.



Obrázek 41 – Hlavní GUI pro přidávání bloků do blockchainu (vlastní zdroj)

Na obrázku 41 je zobrazené rozvržení hlavního okna aplikace. Hlavní okno aplikace se skládá ze dvou textových polí v levé části okna, jedno pro nadpis a druhé pro samotná textová data. Pod textovými poli je tlačítko, pro potvrzení přidání bloku. V pravé části je náhled posledního přidaného bloku do blockchainu. Pro rozeznání, jaký blockchain je aktuálně otevřen, slouží nadpis, který obsahuje název otevřeného blockchainu. Z horního menu je možné taky vytvořit nový blockchain nebo otevřít již existující.

```
def __init__(self, file_name):
    """
    First initialization and linking function in window GUI
    :param file_name: Blockchain name
    """
    self._file_name = file_name  # Local variable for file name. This changes when opening file from main window
    self.window = None
    QtWidgets.QMainWindow.__init__(self)  # Initialization function of the parent QMainWindow class
    Ui_MainWindow.__init__(self)  # Initialization function of the Ui_MainWindow class to set up GUI elements
    self.setupUi(self)  # setupUi function to initialize the UI
    self.setFixedSize(910, 910)  # Set window size
    self.label_subjectWarning.hide()  # Hide subject warning
    self.label_textDataWarning.hide()  # Hide data warning
    self.label_openWarning.hide()  # Hide open file warning
    self.label_title.setText(f'BlockChain - {self._file_name}')  # Set text label
    # Start blockchain and store blockchain list and last added block
    self.blockchain_data, self.previous_block = start_blockchain(self._file_name)
    last_block_image = QPixmap('temp/temp.svg')  # Store image to variable
    self.label_lastBlockImage.setPixmap(last_block_image)  # Show SVG image of last added block
    # Link buttons to functions and
    self.action_New_BlockChain.triggered.connect(self.create_new)
    self.action_Open_BlockChain.triggered.connect(self.open_file)
    self.pushButton_submit.clicked.connect(lambda: self.submit_block(self._file_name, self.blockchain_data,
                                                                      self.previous_block))
```

Obrázek 42 – Úvodní inicializace třídy BlockChainGUI (vlastní zdroj)

Funkce úvodní inicializace na obrázku 42 je trochu odlišná od inicializačních funkcí uvedených na obrázcích 34 a 40. Hlavním rozdílem je, že zdejší úvodní inicializace přebírá název blockchainu jako vstupní parametr. Předaný název blockchainu se poté uloží do lokální proměnné. Název využívá pro nastavení nadpisu a pro vytvoření nebo otevření samotného blockchainu. Dalším hlavním rozdílem je načítání obrázku posledního přidaného bloku a zobrazení jej v pravé části okna aplikace. Dále je rozdílná i velikost okna, která je nastavená na 910 pixelů na šířku i na výšku. Poslední řádek využívá lambda funkci pro volání funkce při stisku tlačítka pro sestavená a přidání bloku do blockchainu. Lambda je zapotřebí, protože se při stisku tlačítka předávají parametry, které by bez lambda funkce nemohli být předané. Bez lambda funkce se předává pouze název funkce a aplikace by spadla.

```
class BlockChainGUI(QtWidgets.QMainWindow, Ui_MainWindow):
    def create_new(self):
        """ Switch GUI to another one where blockchain name is selected """
        if self.window is None:
            self.window = CreateNewBlockChain()  # Switch window GUI
        self.close()  # Close old GUI
        self.window.show()  # Open new GUI
```

Obrázek 43 – Vytvoření nového blockchainu z hlavního okna (vlastní zdroj)

Funkce na obrázku 43 se zavolá po stisknutí tlačítka pro vytvoření nového blockchainu z horního menu aplikace. Jediným účelem funkce, je změnit grafické rozhraní aplikace na CreateNewBlockChain, které lze vidět na obrázku 39.

```
    def clear_everything(self):
        """ Clear elements after valid submit """
        self.label_lastBlockImage.clear()  # Clear image label
        self.textEdit_subject.clear()  # Clear subject text edit
        self.textEdit_textData.clear()  # Clear data text edit

    def disable_input(self, inp_disable_enable: bool):
        """ Disable or enable user input """
        self.textEdit_subject.setEnabled(not inp_disable_enable)  # Disable subject text edit
        self.textEdit_textData.setEnabled(not inp_disable_enable)  # Disable data text edit
        self.pushButton_submit.setEnabled(not inp_disable_enable)  # Disable button

    def set_file_name(self, new_file_name):
        """ Setter for local variable for file name """
        self._file_name = new_file_name
```

Obrázek 44 – Funkce pro vizuální úpravu hlavního okna (vlastní zdroj)

Všechny tři funkce na obrázku 44 slouží pouze pro upravování grafického rozhraní hlavního okna při otevírání blockchainu. První funkce vyčistí okno od uživatelského vstupu a odstraní obrázek posledního přidaného bloku. Druhá funkce zakáže nebo povolí uživatelský vstup včetně stisku tlačítka. Poslední funkce je setter na lokální proměnou, která udržuje název blockchainu, se kterým se aktuálně pracuje.

```
def open_file(self):
    """ Open existing blockchain """
    home_dir = os.path.dirname(os.path.abspath(__file__))  # Current file directory
    # Open file dialog
    file_name = QFileDialog.getOpenFileName(self, "Select Directory", home_dir, "DOT File (*.dot)")
    print(f'Opening file: {file_name[0]}')  # Log
    file_name_short = os.path.basename(file_name[0][:-4])  # Only name of file without extension
    self.label_subjectWarning.hide()  # Hide subject warning
    self.label_textDataWarning.hide()  # Hide text data warning

    if os.path.getsize(file_name[0] or file_name[0] == ""):  # Check file size
        # Store file first line
        file = open(file_name[0], 'r')
        line = file.readline()
        file.close()
        # Check file first line
        if re.search(r'^#\[.', line) and re.search(r']$', line):
            print(f'File {file_name[0]} is not empty')  # Log
            self.label_title.show()  # Show title text
            self.label_openWarning.hide()  # Hide warning
            self.disable_input(False)  # Enable user input
            self.set_file_name(file_name_short)  # Set _file_name local variable to open file name
            self.label_title.setText(f'BlockChain - {file_name_short}')  # Set text label
            # Start existing blockchain
            self.blockchain_data, self.previous_block = start_blockchain(file_name_short)
            last_block_image = QPixmap('temp/temp.svg')  # Store image to variable
            self.label_lastBlockImage.setPixmap(last_block_image)  # Show SVG image of last added block
        else:    # File is wrong
            print(f'File {file_name[0]} is wrong')  # Log
            self.label_title.hide()  # Hide title text
            self.label_openWarning.show()  # Show warning
            self.clear_everything()  # Clear edit elements
            self.disable_input(True)  # Disable user input
    else:  # File is empty
        print(f'File {file_name[0]} is empty')  # Log
        self.label_title.hide()  # Hide title text
        self.label_openWarning.show()  # Show warning
        self.clear_everything()  # Clear edit elements
        self.disable_input(True)  # Disable user input
```

Obrázek 45 – Otevření již existujícího blockchainu z hlavního okna (vlastní zdroj)

Princip ověření validity již existujícího blockchainu na obrázku 45 je stejný, jako při ověřování na úvodní obrazovce, který je k vidění na obrázku 36. Prvním rozdílem je, že se nevolá změna grafického rozhraní, ale nastaví se pomocí setteru hodnota lokální proměnné

pro název blockchainu a poté se zavolá funkce pro start blockchainu s názvem otevřeného blockchainu. Poté se vytvoří a načte obrázek posledního přidaného bloku do právě otevřeného blockchainu a zobrazí se v pravé části okna.

Druhým rozdílem je, že v případě otevření nevalidního souboru se zobrazí varování v místě nadpisu v horní části obrazovky Následně se vymažou data z polí pro uživatelský vstup a zakáže se do nich cokoliv psát. V poslední řadě se zakáže i tlačítko pro potvrzení přidání bloku do blockchainu. V opačném případě, když je soubor validní, se varování skryje a zobrazí se nadpis s název otevřeného blockchainu. Povolí se uživatelský vstup do textových polí a tlačítko pro vytvoření a přidání bloku do blockchainu.

```
def submit_block(self, inp_file_name, inp_blockchain_data, inp_previous_block):
    """
    Submit block to blockchain
    :param inp_file_name: Name of blockchain
    :param inp_blockchain_data: List of blockchain
    :param inp_previous_block: Last added block
    """
    subject = self.textEdit_subject.toPlainText()  # Get subject from text edit element
    data = self.textEdit_textData.toPlainText()  # Get data from text edit element

    # Show warnings base on user input
    if subject.strip() == "" and data.strip() == "":  # Subject and data are empty
        self.label_subjectWarning.show()
        self.label_textDataWarning.show()
    elif subject.strip() == "" and not data.strip() == "":  # Subject is empty
        self.label_subjectWarning.show()
        self.label_textDataWarning.hide()
    elif not subject.strip() == "" and data.strip() == "":  # Data are empty
        self.label_subjectWarning.hide()
        self.label_textDataWarning.show()
    else:  # Subject and data are valid
        self.label_subjectWarning.hide()
        self.label_textDataWarning.hide()
        # Add block to blockchain
        self.blockchain_data, self.previous_block = start_blockchain(inp_file_name, subject.strip(),
                                                                     data.strip(),inp_blockchain_data,
                                                                     inp_previous_block)
        last_block_image = QPixmap('temp/temp.svg')  # Store image to variable
        self.label_lastBlockImage.setPixmap(last_block_image)  # Show SVG image of last added block
        # Clear subject and data text edit elements
        self.textEdit_subject.clear()
        self.textEdit_textData.clear()
```

Obrázek 46 – Potvrzení přidání bloku do blockchainu (vlastní zdroj)

Funkce na obrázku 46 slouží k přidání bloku do blockchainu. Funkce obsahuje tři vstupní parametry: název blockchainu, list obsahující seznam bloků v blockchainu a poslední přidaný blok do blockchainu. Prvně se uloží uživatelský vstup z obou textových polí a otestuje se, jestli jsou oba dva validní. Validní jsou v případě, že obsahují alespoň jeden znak mimo mezery. V případě, že je nějaký uživatelský vstup nevalidní, zobrazí se varování na místě nadpisu daného textového pole. Pokud jsou oba vstupy validní, obě varování se skryjí a zavolá se funkce pro vytvoření nebo otevření blockchainu podle předaných parametrů. Po tomto kroku se zobrazí obrázek posledního přidaného bloku do blockchainu a obsah textových polí pro uživatelský vstup se vymaže.

7.2 Implementace blockchainu

Implementace hlavní logiky vytváření blockchainu a bloků je ve skriptu blockchain.py. Skript obsahuje části pro vytvoření bloku z předaných informací, vytvoření prvního bloku v blockchainu, vytvoření dalšího bloku v blockchainu, získání času a stylistickou úpravu jeho zápisu, logování přidaných bloků do blockchainu na konzoli a funkci, která ovládá přidávání bloků podle vstupních parametrů.

7.2.1 Funkce pro logování a získání času

Při vytváření bloků v blockchainu využíváme dvě funkce. Jedna slouží pro získání časového razítka a jeho následnou úpravu do požadovaného formátu. Druhá funkce zajišťuje logování přidaných bloků na konzoli.

```
def get_time_now():
    """
    Only to get timestamp and styled datetime
    :return: String of timestamp and styled timedate for blockchain diagram
    """
    time_now = datetime.now()  # Timestamp now
    time_datetime = time_now.strftime("%d. %m. %Y %H:%M:%S")  # Styled date and time for better view
    return str(time_now), time_datetime


def log(inp_id, inp_timestamp, inp_prev_hash, inp_head_hash, inp_sub, inp_data):
    """
    Log blocks to console
    :param inp_id: Block ID
    :param inp_timestamp: Block timestamp
    :param inp_prev_hash: Previous block header hash
    :param inp_head_hash: Block header hash
    :param inp_sub: Block subject
    :param inp_data: Block data
    """
    print('#' * 80)  # Print 80 times hashtag symbol
    print(f'\nIndex: {inp_id}\nTimestamp: {inp_timestamp}\nPrevious Hash: {inp_prev_hash}\n'
          f'Header Hash: {inp_head_hash}\nSubject: {inp_sub}\nData: {inp_data}\n')
    print('#' * 60)
```

Obrázek 47 – Podpůrné funkce (vlastní zdroj)

Vrchní funkce get_time_now na obrázku 47 slouží pro získání aktuálního časového razítka a jeho následnou úpravu do evropského formátu datumu s časem. K získání aktuálního časového razítka využívá knihovnu datetime a z ní funkci now. Získané časové razítko nakonec před vrácením ještě přetypuje do formátu string. Druhá funkce slouží pro logování dat přidaných bloků do blockchainu na konzoli.

7.2.2 Vytvoření a přidání bloku do blockchainu

Blockchain je tvořen ze dvou částí, bloku a chainu (česky řetězce). Blok obsahuje všechny potřebná data a chain je způsob propojení jednotlivých bloků. V jiném slova smyslu by se dalo říct, že se jedná o řetěz, na který se postupně zavěšují bloky od nejstaršího po

nejnovější podle času vytvoření. Každý blok udržuje informaci o předchozím bloku v podobě jeho Hash hodnoty.

```
class Block:
    def __init__(self, index, timestamp, subject, data, previous_hash):
        """
        Create Block of Blockchain
        :param index: Index of Block
        :param timestamp: Timestamp from when the Block is created
        :param subject: Subject of the block
        :param data: Data text
        :param previous_hash: Hash value of previous Block
        """
        self.index = index
        self.timestamp = timestamp
        self.subject = subject
        self.data = data
        self.previous_hash = previous_hash
        self.header_hash = self.hash_block()

    def hash_block(self):
        """
        Creates Hash from Block Header
        :return: Hash value of Block Header
        """
        header_hash = hashlib.sha256()  # Using SHA256 hashing algorithm
        # Updating hash with header info
        # Strings need to be encoded before hashing
        header_hash.update((str(self.index) +
                            str(self.timestamp) +
                            str(self.previous_hash)).encode('utf-8'))
        return header_hash.hexdigest()
```

Obrázek 48 – Vytvoření bloku v blockchainu (vlastní zdroj)

Na obrázku 48 je kód třídy Block, která obsahuje dvě funkce, pro vytvoření bloku ze vstupních dat. Po zavolání třídy se jako první zavolá inicializační funkce, která vytvoří blok z předaných informací. Data potřebná k vytvoření bloku jsou: ID bloku, časové razítko z doby vytvoření bloku, nadpis pro textová data, textová data obsahu bloku, Hash hodnotu předchozího bloku a Hash hodnotu hlavičky samotného bloku. Pro získání Hash hodnoty hlavičky bloku slouží Hashovací algoritmus SHA-256 z knihovny hashlib, která vrací Hash o pevně stanovené délce 256 bitů neboli 64 znaků. Pro správnou funkci je nutné všechny vstupní parametry převést na datový typ string, a poté vše zakódovat. Pro správné zobrazení Hash hodnoty hlavičky bloku, je zapotřebí použít funkci hexdigest, která nám Hash hodnotu zobrazí v hexadecimálním formátu.

7.2.3 Úvodní blok v blockchainu

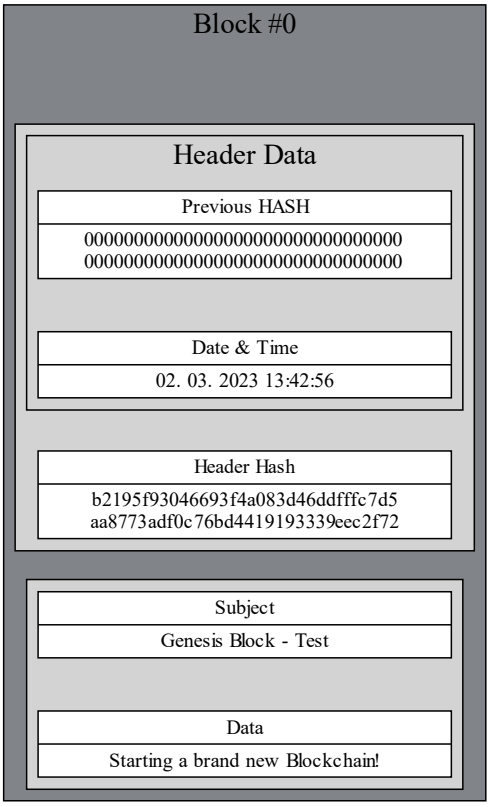
Úvodní blok se od ostatních bloků liší tím, že neobsahuje Hash hodnotu předchozího bloku, protože žádný předchozí blok v blockchainu není. V mém provedení se úvodní blok generuje automaticky s přednastavenými hodnotami.

```python
def genesis_block_blockchain(file_name):
    """
    Creates first Block in Blockchain
    :param file_name: Name of block diagram output file
    :return: Genesis block of blockchain
    """
    index = 0
    timestamp, time_datetime = get_time_now()  # Get original timestamp and styled timedate
    subject = f'Genesis Block - {file_name}'
    data = "Starting a brand new Blockchain!"
    prev_hash = "0" * 64  # Set previous hash of genesis block to zeros

    genesis_block = Block(index, timestamp, subject, data, prev_hash)  # Creates Block
    # Creates first block in diagram output
    first_block_diagram(index, prev_hash, timestamp, time_datetime, genesis_block.header_hash,
                        subject, data, file_name)
    return genesis_block
```

Obrázek 49 – Vytvoření úvodního bloku v blockchainu (vlastní zdroj)

Funkce na obrázku 49 vytvoří úvodní blok v blockchainu s jediným vstupním parametrem, kterým je název blockchainu. Všechny hodnoty jsou pevně nastavené, protože se blok vytváří bez uživatelského vstupu. Každý blok obsahuje hodnotu ID nastavenou na 0, datum a čas kdy byl vytvořen. V nadpisu textu je obsažen název blockchainu, aby bylo možné rozeznat, o jaký blockchain se jedná i v případě, že uživatel nemá přístup k celé složce. Dále obsahuje pevně stanovená textová data a pevně nastavenou Hash hodnotu předchozího bloku, která je nastavená na samé nuly. Poté se vytvoří blok předáním potřebných parametrů do třídy Block, která má na starosti vytváření bloků. V poslední řade, ještě před vrácením vytvořeného úvodního bloku, se zavolá funkce pro vytvoření úvodního bloku v diagramu blockchainu. Výsledný obrázek úvodního bloku diagramu lze vidět na obrázku 50.



Obrázek 50 – Obrázek úvodního bloku diagramu (vlastní zdroj)

7.2.4 Další přidaný blok v blockchainu

Všechny další přidané bloky do blockchainu jsou již s uživatelským vstupem, kde uživatel musí vyplnit nadpis pro textové data a textová data samotné. Pro vytvoření bloku musí následně uživatel potvrdit stiskem tlačítka v hlavním okně aplikace.

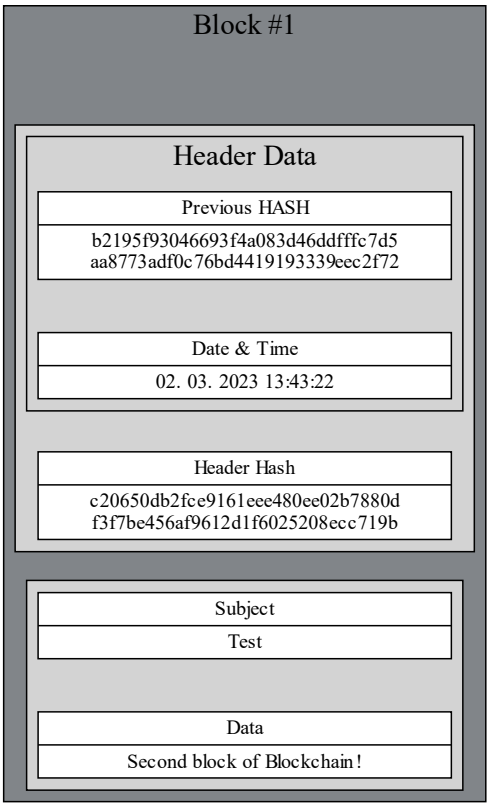
```
def next_block_blockchain(last_block, subject, data, file_name):
    """
    Add next block to Blockchain
    :param last_block: Last added block to Blockchain
    :param subject: Subject of the block
    :param data: Data text
    :param file_name: Name of block diagram output file
    :return: New block of blockchain
    """
    this_index = last_block.index + 1
    this_last_block_header_hash = last_block.header_hash
    this_timestamp, this_time_date = get_time_now()  # Get original timestamp and styled timedate

    new_block = Block(this_index, this_timestamp, subject, data, this_last_block_header_hash)  # Creates Block
    add_block_diagram(this_index, last_block.index, this_last_block_header_hash, this_timestamp, this_time_date,
                      new_block.header_hash, subject, data, file_name)
    return new_block
```

Obrázek 51 – Přidání dalšího bloku do blockchainu (vlastní zdroj)

Zdrojový kód na obrázku 51 slouží k přidání dalšího bloku do blockchainu již s uživatelským vstupem jako vstupními parametry. Dalšími vstupními parametry této funkce jsou

název blockchainu a poslední přidaný blok do blockchainu. Z předchozího bloku inkrementujeme ID a uložíme si hodnotu Hash hlavičky bloku. Poté získáme aktuální řasové razítko a vytvoříme blok, který zároveň přidáme do diagramu blockchainu. Nakonec vrátíme vytvořený blok.



Obrázek 52 – Obrázek dalšího přidaného bloku do diagramu (vlastní zdroj)

Na obrázku 52 je zobrazen další přidaný blok do blockchainu. Lze vidět, že ID se inkrementovalo a že Hash hodnota předchozího bloku opravdu sedí s předchozím blokem, který je zobrazen na obrázku 50.

7.2.5 Vytváření bloků a přidávání je do blockchainu

Jedna funkce ovládá, jaké data bude blok obsahovat a jak se upraví vygenerovaný diagram blockchainu. Toto se rozhoduje podle vstupních parametrů, které se funkci předají. Funkce buď vytvoří první blok, nebo přidá další blok.

```
def start_blockchain(inp_name, inp_subject=None, inp_data=None, blockchain=None, previous_block=None):
    """
    Creates blockchain folder and create a blockchain diagram
    :param inp_name: Name of blockchain
    :param inp_subject: Subject of the block
    :param inp_data: Data text
    :param blockchain: Blockchain list
    :param previous_block: Last added block to blockchain
    :return: Updated blockchain and recent added block to blockchain
    """
    if blockchain is None:
        blockchain = []
    if inp_subject is None and inp_data is None:  # Check for user input
        if not os.path.exists(f'diagram_files/{inp_name}'):  # Check if folder with blockchain name don't exist
            os.mkdir(f"diagram_files/{inp_name}")  # Make folder with blockchain name
            blockchain.append(genesis_block_blockchain(inp_name))
            previous_block = blockchain[0]
            # Console log
            print(f'{" FIRST ADDED BLOCK ":#^60}')
            log(blockchain[0].index, blockchain[0].timestamp, blockchain[0].previous_hash,
                blockchain[0].header_hash, blockchain[0].subject, blockchain[0].data)
            print("\n")
        else:  # If folder exist
            if os.path.getsize(f'diagram_files/{inp_name}/{inp_name}.dot') == 0:  # Check file size
                blockchain.append(genesis_block_blockchain(inp_name))
                previous_block = blockchain[0]
            else:
                file = open(f'diagram_files/{inp_name}/{inp_name}.dot', 'r', encoding='utf-8')
                first_line = file.readline()
                file.close()
                previous_block = first_line[1:]
                previous_block = literal_eval(previous_block)  # Creates a list from string
                # Creates SVG image of last added block
                create_last_block(previous_block[0], previous_block[4], previous_block[6], previous_block[5],
                                  previous_block[2], previous_block[3])
                blockchain = [Block(previous_block[0], previous_block[1], previous_block[2],
                                    previous_block[3], previous_block[4])]
                previous_block = blockchain[0]
                # Console log
                log(blockchain[0].index, blockchain[0].timestamp, blockchain[0].previous_hash,
                    blockchain[0].header_hash, blockchain[0].subject, blockchain[0].data)
    else:  # Add block with user input
        block_to_add = next_block_blockchain(previous_block, inp_subject, inp_data, inp_name)  # Creates a block
        blockchain.append(block_to_add)
        previous_block = block_to_add

        # Console log
        # Set maximum of 3 block written to console log
        blockchain_len = len(blockchain) if len(blockchain) < 3 else 3
        for block in blockchain[-blockchain_len:]:
            log(block.index, block.timestamp, block.previous_hash, block.header_hash, block.subject, block.data)
        print("\n" * 2)
    return blockchain, previous_block
```

Obrázek 53 – Vytváření bloků a přidávání je do blockchainu (vlastní zdroj)

Funkce na obrázku 53 slouží k vytváření bloků na základě vstupních parametrů, kde je pouze název souboru povinný. Ostatní parametry mají nastavenou základní hodnotu pro případ, že uživatel vytváří nový nebo otevírá existující blockchain.

Funkce prvně otestuje vstupní parametry, a poté se na základě jejich hodnoty rozhodne, jestli se jedná o přidání bloku s uživatelským vstupem nebo ne. V případě, že byla funkce zavolána s uživatelským vstupem, vytvoří se pouze nový blok a přidá se do listu bloků v blockchainu. V opačném případě proběhne kontrola, jestli existuje složka s názvem blockchainu, která se v případě absence vytvoří. Pokud složka existuje, tak se zkontroluje existence dokumentu obsahující zdrojový kód pro generování diagramu. Pokud soubor exis-

tuje, funkce zkopíruje první řádek, který drží informace o posledním přidaném bloku a vygeneruje obrázek posledního přidaného bloku. Poté blok přidá do listu bloků a nastaví ho jako poslední přidaný blok.

8 GRAFICKÁ REALIZACE

Grafické zobrazení jednotlivých bloků a celého blockchainu probíhá za pomoci open source softwaru Graphviz. Prvotní návrhy kostry bloku jsem vytvořil v online editoru na stránkách Graphvizu. Poté jsem naprogramovanou kostru vzal a upravil ji tak, aby se soubor s kódem automaticky vyplňoval podle informaci z generovaného bloku. Následné generování obrázku nebo diagramu probíhá pomocí příkazu v příkazové řádce. V příkazu se specifikuje formát obrázku, kódování souboru a udává se cesta k souboru pro generování a kam chceme výsledný obrázek uložit.

V mém provedení, se jedná o tři funkce, kde jedna vytváří první blok v blockchainu, druhá přidává každý další blok a třetí vytváří obrázek pouze posledního přidaného bloku od právě otevřeného blockchainu. Ke správnému generování je zapotřebí i úprava vstupních dat do diagramu, protože by se jinak nemusela zobrazit správně a výsledek by vypadal hodně chaoticky.

8.1 Úprava dat pro správné zobrazení

Pro správné zobrazení dat je zapotřebí upravit kolonky Hash, Subject a Data. Zapotřebí je upravit hlavně šířku textu, aby se daný blok moc neroztáhnul a zobrazení nebylo pak velmi nepřehledné. Navíc pro zobrazení posledního přidaného bloku, je zapotřebí upravit i celkovou délku textu Dat a Subjectu, protože by se nám obrázek bloku nemusel správně zobrazit v okně aplikace a mohl by být vyšší a širší, než je velikost okna pro zobrazení.

8.1.1 Stylizace Hash hodnoty

```
def style_hash(inp_hash):
    """
    Style input Hash string value for displaying them on .SVG image properly
    :param inp_hash: Hash value input
    :return: Styled Hash value
    """
    # Split 64 character in 2 pieces to display them correctly on diagram
    split_hash = [(inp_hash[i:i+32]) for i in range(0, len(inp_hash), 32)]
    out_hash = "\\n".join(split_hash)
    return out_hash
```

Obrázek 54 – Vizuální úprava zobrazení Hash hodnoty (vlastní zdroj)

Jelikož hashovací funkce SHA-256 vždy vrací Hash o délce 64 znaků, tak je potřeba daný Hash rozpůlit, aby se v bloku zobrazil přehledně. Funkce na obrázku 54 vstupní hodnotu Hash rozpůlí po 32 znacích, kde mezi obě části vloží nový řádek. Ve vygenerovaném bloku se Hash hodnota zobrazí na dva řádky pod sebou. Funkce funguje i v případě, kdyby

byl použitý jiný Hashovací algoritmus. Například SHA-512 vrací Hash o délce 128 znaků, takže by se v obrázku bloku zobrazil vygenerovaný Hash na čtyři řádky.

8.1.2 Stylizace nadpisu a textových dat

Jelikož máme nastavenou nějakou šířku bloku, kterou nechceme přesáhnout, tak je zapotřebí vstupní data od uživatele určitým způsobem upravit tak, aby se při grafickém zobrazení vlezli do daného bloku. Zde se vyskytli dva hlavní problémy, kde jedním byl velmi dlouhý text a druhým slovo delší než šířka řádku.

```
def style_split_data(inp_data):
    """
    Split text into words and split very long words into chunks
    :param inp_data: Text to split into words and splitting very long words
    :return: Split text in list
    """
    split = inp_data.split(" ")  # Split text into words
    # Check if any of words is longer than 36 characters
    if any(len(word) ⩾ 37 for word in split):
        for word in split:
            if len(word) ⩾ 37:
                # Split words longer than 36 characters
                split_word = [(word[i:i+35]) for i in range(0, len(word), 35)]
                word_index = split.index(word)  # Store index of the long word
                split.remove(word)  # Remove the long word
                for word2 in split_word:
                    if len(word2) == 35:
                        word2 = f'{word2}-'  # Update split long word
                    split.insert(word_index, word2)  # Update base list of words
                    word_index += 1
    return split
```

Obrázek 55 – Rozdělení textu a dlouhých slov (vlastní zdroj)

Funkce na obrázku 55 upraví text tím, že ho rozdělí na jednotlivá slova. Slova, která jsou delší než 36 znaků se rozdělí po 35 znacích. Následně se slovo odstraní z původního listu. Pokud má část rozděleného slova přesně 35 znaků, přidá se na konec pomlčka. Všechny části se poté přidají na místo smazaného slova.

```
def style_data(inp_data):
    """
    Style input string data for displaying them on .svg image properly
    :param inp_data: Text to style
    :return: Styled text
    """
    split_data = style_split_data(inp_data)  # Split text into words
    out_data = ""  # Output string
    tmp_data = ""  # Temporary string
    for word in split_data:
        tmp_data += f'{word} '
        # Put a new line right before word if tmp string is long
        if len(tmp_data) ⩽ 39:
            out_data += f'{word} '
        else:
            out_data += f'\\n{word} '
            tmp_data = f'{word} '  # Reset temporary string
    return out_data
```

Obrázek 56 – Spojení slov a rozdělení textu na řádky (vlastní zdroj)

Funkce na obrázku 56 je druhou částí úpravy textu, která se stará o spojení slov z první funkce podle. Při spojování se kontroluje délka spojených slov, aby nepřekročila

délku 39 znaků. Tato úprava je zapotřebí, aby se text zobrazoval správně ve vygenerovaném obrázku bloku.

8.1.3 Vytvoření prvního bloku v blockchainu

Vytvoření prvního bloku v diagramu je odlišné, protože při vytváření prvního bloku vytváříme i hlavičku celého dokumentu pro nastavení vzhledu a bloků a celého diagramu.

```
def first_block_diagram(block_num, prev_hash, time_timestamp, time_datetime, header_hash, subject,
                        data, file_name):
    """
    Creates a diagram in graphviz and add first block to it
    :param block_num: Block ID
    :param prev_hash: Hash value of previous block
    :param time_timestamp: Timestamp from when block was created
    :param time_datetime: Styled datetime for displaying it in diagram
    :param header_hash: Hash of actual block
    :param subject: Subject of block
    :param data: Text data
    :param file_name: Blockchain name
    """
    # List of data from recent block added
    this_block = [block_num, time_timestamp, subject, data, prev_hash, header_hash, time_datetime]
    # Create SVG image of recent block added
    create_last_block(block_num, prev_hash, time_datetime, header_hash, subject, data)
    # Styling data for displaying them od diagram
    prev_hash = style_hash(prev_hash)
    header_hash = style_hash(header_hash)
    subject = style_data(subject)
    data = style_data(data)
```

Obrázek 57 – Úprava dat před vytvořením prvního bloku (vlastní zdroj)

Na úvod ve funkci na obrázku 57 se upraví vstupní data předtím, než se vytvoří soubor se zdrojovým kódem pro vygenerování diagramu blockchainu. Po úpravě dat se vytvoří soubor již naplněný upravenými daty a vygeneruje se diagram s prvním přidaným blokem. Druhá část zdrojového kódu funkce z obrázku 57 k vytvoření takového diagramu je na obrázku 58.

```
# Write a file for graphviz diagram
file = open(f'diagram_files/{file_name}/{file_name}.dot', 'w', encoding='utf-8')
file.write(
    f'#{str(this_block)}\n'  # Store data of recent added block
    f'digraph G {{\n'  # Creates a diagram
    f'    rankdir = "TB";\n\n'  # Sorts nodes in subgraph from Top to Bottom
    f'    node [\n'  # Style node
    f'        shape = record;\n'
    f'        style = filled;\n'
    f'        fillcolor = white;\n'
    f'        color = black;\n'
    f'        width = 4;\n'
    f'    ];\n\n'
    f'    graph [\n'  # Style graph
    f'        style = filled;\n'
    f'        fillcolor = "#818589";\n'
    f'        color = black;\n'
    f'        fontsize = 20;\n'
    f'    ];\n\n'
    f'    subgraph cluster_{block_num} {{\n'  # Creates subgraph which symbolizes block of blockchain
    f'        label = "Block #{block_num}";\n'  # Block ID
    f'        align_block_{block_num} [style = invis];\n'  # Invisible node to align other nodes
    f'        subgraph cluster_{block_num}00 {{\n'  # Subgraph for Header Hash
    f'            label = ""\n'
    f'            fillcolor = lightgrey;\n'
    f'            subgraph cluster_{block_num}01 {{\n'  # Subgraph for Header Data
    f'                label = "Header Data"\n'
    f'                "node{block_num}1" [\n'  # Previous block Hash node
    f'                    label = "{{<f0> Previous HASH | <f1> {prev_hash}}}"\n'
    f'                ]\n\n'
    f'                "node{block_num}2" [\n'  # Datetime node
    f'                    label = "{{<f0> Date & Time | <f1> {time_datetime}}}"\n'
    f'                ]\n'
    f'            }}\n\n'
    f'            "node{block_num}3" [\n'  # Header Data Hash node
    f'                label = "{{<f0> Header Hash | <f1> {header_hash}}}"\n'
    f'            ]\n'
    f'        }}\n\n'
    f'        subgraph cluster{block_num}10 {{\n'  # Subgraph for subject and data
    f'            label = ""\n'
    f'            fillcolor = lightgray;\n'
    f'            "node{block_num}4" [\n'  # Subject node
    f'                label = "{{<f0> Subject | <f1> {subject}}}"\n'
    f'            ]\n\n'
    f'            "node{block_num}5" [\n'  # Data node
    f'                label = "{{<f0> Data | <f1> {data}}}"\n'
    f'            ]\n'
    f'        }}\n\n'
    f'        align_block_{block_num} -> node{block_num}1 -> node{block_num}2 -> node{block_num}3 -> '
    f'node{block_num}4 -> node{block_num}5 [style = invis];\n'  # Connecting all nodes and hide connectors
    f'    }}\n'
    f'}}'
)
file.close()
create_svg_image(file_name)  # Creating an SVG image of diagram
```

Obrázek 58 – Zdrojový kód vytvoření prvního bloku diagramu (vlastní zdroj)

8.1.4 Přidání dalšího bloku do diagramu

Přidání dalšího bloku do diagramu probíhá úpravou zdrojového kódu diagramu a znovu jeho vygenerováním. Zdrojový kód se celý kromě prvního a posledního řádku překopíruje a poté se dopíše část pro vložení dalšího bloku. Tento krok se opakuje po každém přidání bloku.

```
# Copy old data from graphviz diagram file
file = open(f'diagram_files/{file_name}/{file_name}.dot', 'r', encoding='utf-8')
lines = file.readlines()
file.close()
# Add new data to graphviz diagram file
file = open(f'diagram_files/{file_name}/{file_name}.dot', 'w', encoding='utf-8')
file.write(f'#{str(this_block)}\n')  # Store data of recent added block
for line in lines[1:-1]:
    file.write(line)  # Rewriting old data
file.write(
    f'\n'
    f'    subgraph cluster_{block_num} {{\n'  # Creates subgraph which symbolizes block of blockchain
    f'        label = "Block #{block_num}";\n'  # Block ID
    f'        align_block_{block_num} [style = invis];\n'  # Invisible node to align other nodes
    f'        subgraph cluster_{block_num}00 {{\n'  # Subgraph for Header Hash
    f'            label = ""\n'
    f'            fillcolor = lightgrey;\n'
    f'            subgraph cluster_{block_num}01 {{\n'  # Subgraph for Header Data
    f'                label = "Header Data"\n'
    f'                "node{block_num}1" [\n'  # Previous block Hash node
    f'                    label = "{{<f0> Previous HASH | <f1> {prev_hash}}}"\n'
    f'                ]\n\n'
    f'                "node{block_num}2" [\n'  # Datetime node
    f'                    label = "{{<f0> Date & Time | <f1> {time_datetime}}}"\n'
    f'                ]\n'
    f'            }}\n\n'
    f'            "node{block_num}3" [\n'  # Header Data Hash node
    f'                label = "{{<f0> Header Hash | <f1> {header_hash}}}"\n'
    f'            ]\n'
    f'        }}\n\n'
    f'        subgraph cluster{block_num}10 {{\n'  # Subgraph for subject and data
    f'            label = ""\n'
    f'            fillcolor = lightgray;\n'
    f'            "node{block_num}4" [\n'  # Subject node
    f'                label = "{{<f0> Subject | <f1> {subject}}}"\n'
    f'            ]\n\n'
    f'            "node{block_num}5" [\n'  # Data node
    f'                label = "{{<f0> Data | <f1> {data}}}"\n'
    f'            ]\n'
    f'        }}\n\n'
    f'        align_block_{block_num} -> node{block_num}1 -> node{block_num}2 -> node{block_num}3 -> '
    f'node{block_num}4 -> node{block_num}5 [style = invis];\n'  # Connecting all nodes and hide connectors
    f'    }}\n\n'
    # Creates a connector from node 'Previous HASH' of recent block to node 'Header HASH' of previous block
    f'node{block_num}1 : f1 -> node{prev_block_num}3 : f1 [color = red, penwidth = 3];\n'
    f'{{rank = same; align_block_{prev_block_num}, align_block_{block_num}}}\n'  # Align all block to same row
    f'}}\n'
)
file.close()
create_svg_image(file_name)  # Creating an SVG image of diagram
```

Obrázek 59 – Přidání dalšího bloku do diagramu (vlastní zdroj)

8.1.5 Vygenerování posledního přidaného bloku

Upravení dat pro vygenerování posledního přidaného bloku je rozdílná pouze v případném zkrácení nadpisu a textových dat. Zkrácení je zapotřebí pro správné zobrazení v hlavním okně aplikace. Následné vygenerování obrázku bloku na obrázku 59 je totožné, jako na obrázku 58, kde jediným rozdílem je místo uložení zdrojového kódu a vygenerovaného obrázku.

```
def create_svg_image(img_name):
    """
    Creates a .svg image base with name img_name
    :param img_name: Name of output image file
    """
    os.system(f'cmd /c "dot -Tsvg -Gcharset=utf8 diagram_files/{img_name}/{img_name}.dot -o '
              f'diagram_files/{img_name}/{img_name}.svg"')


def create_temp_svg_image():
    """ Creates a .png image of last added block """
    os.system(f'cmd /c "dot -Tsvg -Gcharset=utf8 temp/temp.dot -o temp/temp.svg"')
```

Obrázek 60 – Funkce pro generování diagramu a obrázku (vlastní zdroj)

Horní funkce na obrázku 60 slouží k vygenerování diagramu. Vstupní parametr je název blockchainu a udává název diagramu po vygenerování. Spodní funkce je pro vygenerování posledního přidaného bloku. Obě funkce používají pro vygenerování příkazovou řádku, která se po provedení příkazu sama uzavře.

ZÁVĚR

Cílem práce bylo probrat jednotlivé kryptografické algoritmy využívané v technologii blockchain a jejich rozbor a účel používání. Posléze rozebrat jak používané kryptografické algoritmy, tak i možnosti nových kryptografických algoritmů v době kvantových počítačů a dopad kvantových počítačů na ně.

Teoretická část se zabývá technologií blockchain. Prvně se rozebere význam technologie blockchain, její funkčnost, jednotlivé části, a nakonec výhody a nevýhody této technologie. Následně se rozeberou typy blockchainů a jejich možné využití. Posléze se práce zaměří na kryptografické algoritmy využívané v technologii blockchain a jejich rozbor. V poslední řadě se rozeberou možné využití technologie blockchain v praxi a kryptografické algoritmy v době kvantových počítačů.

Praktická část se zabývá vytvořením jednoduchého blockchainu pro přidávání textových dat. Je zde popsaný celý postup vytvoření aplikace včetně popisu grafických rozhraní a grafického zpracování a zobrazení přidaných dat do blockchainu. Grafické rozhraní umožňuje uživatelsky přívětivější práci a grafické zobrazení přináší uživatelský komfort a usnadňuje pochopení funkčnosti základního principu technologie blockchain.

SEZNAM POUŽITÉ LITERATURY

[1] HAYES, Adam. Blockchain Facts: What Is It, How It Works, and How It Can Be Used. *Investopedia* [online]. Investopedia, [2022], 27 September 2022 [cit. 2023-04-07]. Dostupné z: https://www.investopedia.com/terms/b/blockchain.asp

[2] What is blockchain technology?. *IBM* [online]. IBM, [2018] [cit. 2023-04-07]. Dostupné z: https://www.ibm.com/topics/blockchain

[3] IREDALE, Gwyneth. History Of Blockchain Technology: A Detailed Guide. *101 Blockchains* [online]. 101 Blockchains, © 2023, 3 November 2020 [cit. 2023-04-07]. Dostupné z: https://101blockchains.com/history-of-blockchain-timeline/

[4] History of blockchain. *ICAEW* [online]. ICAEW, ©2023 [cit. 2023-04-08]. Dostupné z: https://www.icaew.com/technical/technology/blockchain-and-cryptoassets/blockchain-articles/what-is-blockchain/history

[5] What is Decentralization in Blockchain?. *AWS* [online]. Amazon Web Services, © 2023 [cit. 2023-04-07]. Dostupné z: https://aws.amazon.com/blockchain/decentralization-in-blockchain/

[6] PATRIZIO, Andy. Definition: Blockchain decentralization. *TechTarget* [online]. TechTarget, © 2007-2023 [cit. 2023-04-07]. Dostupné z: https://www.techtarget.com/searchcio/definition/blockchain-decentralization

[7] RODRIGUEZ, Gabriel, RODRÍGUEZ HAMILTON, Claudia, ed. What Is Blockchain?. *Money* [online]. Money, © 2023, 02 Jun 2022 [cit. 2023-04-08]. Dostupné z: https://money.com/what-is-blockchain/

[8] BUDHI, Veera. Advantages And Disadvantages Of Blockchain Technology. *Forbes* [online]. Forbes Technology Council, [2022], 20 Oct 2022 [cit. 2023-04-08]. Dostupné z: https://www.forbes.com/sites/forbestechcouncil/2022/10/20/advantages-and-disadvantages-of-blockchain-technology/?sh=a57f44e34537

[9] WEGRZYN, Kathleen E. a Eugenia WANG. Types of Blockchain: Public, Private, or Something in Between. *Foley & Lardner LLP* [online]. Foley & Lardner LLP, © 2023, 19 August 2021 [cit. 2023-04-08]. Dostupné z: https://www.foley.com/en/insights/publications/2021/08/types-of-blockchain-public-private-between

[10] SETH, Shobhit. Public, Private, Permissioned Blockchains Compared. *Investopedia* [online]. Investopedia, [2022], 28 July 2022 [cit. 2023-04-09]. Dostupné z:

https://www.investopedia.com/news/public-private-permissioned-blockchains-compared/

[11] GERONI, Diego. What Is A Public Blockchain? Beginner's Guide. *101 Blockchains* [online]. 101 Blockchains, © 2023, 10 November 2020 [cit. 2023-04-09]. Dostupné z: https://101blockchains.com/public-blockchain/

[12] KUMAR SHARMA, Toshendra. Types Of Blockchains Explained: Public Vs. Private Vs. Consortium. *Blockchain Council* [online]. Blockchain Council, 2022, 12 October 2022 [cit. 2023-04-09]. Dostupné z: https://www.blockchain-council.org/blockchain/types-of-blockchains-explained-public-vs-private-vs-consortium/

[13] HETLER, Amanda. Proof of work vs. proof of stake: What's the difference?. *TechTarget: WhatIs.com* [online]. TechTarget, © 1999 - 2023, 01 Aug 2022 [cit. 2023-04-10]. Dostupné z: https://www.techtarget.com/whatis/feature/Proof-of-work-vs-proof-of-stake-Whats-the-difference

[14] FRANKENFIELD, Jake. What Does Proof-of-Stake (PoS) Mean in Crypto?. *Investopedia* [online]. Investopedia, [2023], 22 April 2023 [cit. 2023-04-10]. Dostupné z: https://www.investopedia.com/terms/p/proof-stake-pos.asp

[15] FRANKENFIELD, Jake. What Is Proof of Work (PoW) in Blockchain?. *Investopedia* [online]. Investopedia, [2023], 09 February 2023 [cit. 2023-04-10]. Dostupné z: https://www.investopedia.com/terms/p/proof-work.asp

[16] ANWAR, Hasib. What Is A Private Blockchain?: Beginner's Guide. *101 Blockchains* [online]. 101 Blockchains, © 2023, 02 June 2021 [cit. 2023-04-09]. Dostupné z: https://101blockchains.com/what-is-a-private-blockchain/

[17] HAIDER, Fatima. A Complete Guide to Consortium Blockchain And Its Features. *Analytics Vidhya* [online]. Analytics Vidhya, © 2013-2023, 7 January 2023 [cit. 2023-04-09]. Dostupné z: https://www.analyticsvidhya.com/blog/2023/01/a-complete-guide-to-consortium-blockchain-and-its-features/

[18] CAMPBELL, Christine. What are the 4 different types of blockchain technology?. *TechTarget* [online]. TechTarget, © 2007-2023, 03 Mar 2023 [cit. 2023-04-09]. Dostupné z: https://www.techtarget.com/searchcio/feature/What-are-the-4-different-types-of-blockchain-technology

[19] AFOLABI, Oluwademiladei. What Is a Hybrid Blockchain, and How Does It Differ from a Regular Blockchain?. *Make Use Of* [online]. Make Use Of, © 2023, 21 Mar 2023 [cit. 2023-04-09]. Dostupné z: https://www.makeuseof.com/what-is-hybrid-blockchain-how-differ-from-regular-blockchain/

[20] GERONI, Diego. Hybrid Blockchain: The Best Of Both Worlds. *101 Blockchains* [online]. 101 Blockchains, © 2023, 28 January 2021 [cit. 2023-04-10]. Dostupné z: https://101blockchains.com/hybrid-blockchain/

[21] ROTH, Stephan. An Introduction to Sidechains. *CoinDesk* [online]. CoinDesk, © 2023, 7 Mar 2022 [cit. 2023-04-10]. Dostupné z: https://www.coindesk.com/learn/an-introduction-to-sidechains/

[22] What Are Sidechains?: Scaling Blockchain on the Side. *Crypto.com* [online]. Crypto.com, © 2018 - 2023, 4 Feb 2021 [cit. 2023-04-10]. Dostupné z: https://crypto.com/university/what-are-sidechains-scaling-blockchain

[23] WESTON, Georgia. A Detailed Guide On Sidechains. *101 Blockchains* [online]. 101 Blockchains, © 2023, 28 November 2022 [cit. 2023-04-10]. Dostupné z: https://101blockchains.com/sidechains-blockchain/

[24] SINGH CHOUDHARY, Anurag. Concept of Cryptography in Blockchain. *Analytics Vidhya* [online]. Analytics Vidhya, © 2013-2023, 29 September 2022 [cit. 2023-04-12]. Dostupné z: https://www.analyticsvidhya.com/blog/2022/09/concept-of-cryptography-in-blockchain/

[25] RICHARDS, Kathleen. Cryptography. *TechTarget* [online]. TechTarget, © 2000-2023 [cit. 2023-04-12]. Dostupné z: https://www.techtarget.com/searchsecurity/definition/cryptography

[26] ACHARYA, Sunil. Encrypt and Decrypt a String in ASP.NET. In: *C# Corner* [online]. C# Corner, © 2023, 30 Jan 2023 [cit. 2023-04-12]. Dostupné z: https://www.c-sharpcorner.com/blogs/encrypt-and-decrypt-a-string-in-asp-net1

[27] LOO, Andrew. Hash Function. *CFI* [online]. CFI, © 2015 - 2023, 5 April 2023 [cit. 2023-04-12]. Dostupné z: https://corporatefinanceinstitute.com/resources/cryptocurrency/hash-function/

[28] What are digital signatures?. *DocuSign* [online]. DocuSign, © 2023 [cit. 2023-04-15]. Dostupné z: https://www.docusign.com/how-it-works/electronic-signature/digital-signature/digital-signature-faq

[29] SOTO, Leyre. What is a digital signature?. *Signaturit* [online]. Signaturit, © 2022, 19 June 2018 [cit. 2023-04-15]. Dostupné z: https://blog.signaturit.com/en/what-is-a-digital-signature

[30] Why digital signatures are essential for blockchains. *Coinbase* [online]. Coinbase, © 2023, 26 January 2022 [cit. 2023-04-15]. Dostupné z: https://www.coinbase.com/cloud/discover/dev-foundations/digital-signatures

[31] Exploring Elliptic Curve Cryptography. *STEM Hash* [online]. STEM Hash, © 2023, 24 Jun 2021 [cit. 2023-04-15]. Dostupné z: https://stemhash.com/exploring-elliptic-curve-cryptography/

[32] CHAINLINK. What Is a Schnorr Signature?. *Chainlink: Blog* [online]. Chainlink Foundation, © 2023, 26 March 2023 [cit. 2023-04-15]. Dostupné z: https://blog.chain.link/schnorr-signature/

[33] SMIRNOFF, Peter a Dawn M. TURNER. Symmetric Key Encryption: Why, where and how it's used in banking. *Cryptomathic* [online]. Denmark: Cryptomathic, © 1986-2023, 3 January 2020 [cit. 2023-04-15]. Dostupné z: https://www.cryptomathic.com/news-events/blog/symmetric-key-encryption-why-where-and-how-its-used-in-banking

[34] THAKKAR, Jay. Types of Encryption: What to Know About Symmetric vs Asymmetric Encryption. *INFOSEC INSIGHTS: by SECTIGO Store* [online]. 25 April 2020 [cit. 2023-04-15]. Dostupné z: https://sectigostore.com/blog/types-of-encryption-what-to-know-about-symmetric-vs-asymmetric-encryption/

[35] Difference Between Block Cipher and Stream Cipher. *Javatpoint* [online]. Javatpoint, © 2011-2021 [cit. 2023-04-15]. Dostupné z: https://www.javatpoint.com/block-cipher-vs-stream-cipher

[36] Block cipher mode of operation. *Wikipedia: the free encyclopedia*. San Francisco (CA): Wikimedia Foundation, 2023. Dostupné také z: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

[37] Asymmetric Encryption: Definition, Architecture, Usage. *Okta* [online]. Okta, © 2023, 14. 02. 2023 [cit. 2023-04-16]. Dostupné z: https://www.okta.com/identity-101/asymmetric-encryption/

[38] ARAMPATZIS, Anastasios. What Are the Best Use Cases for Symmetric vs Asymmetric Encryption?. *Venafi* [online]. Venafi, ©2023, 16 September 2019 [cit. 2023-

04-16]. Dostupné z: https://venafi.com/blog/what-are-best-use-cases-symmetric-vs-asymmetric-encryption/

[39] FRANKENFIELD, Jake. Merkle Tree in Blockchain: What it is and How it Works. *Investopedia* [online]. Investopedia, 26 July 2021 [cit. 2023-04-16]. Dostupné z: https://www.investopedia.com/terms/m/merkle-tree.asp

[40] TOMESCU, Alin. What is a Merkle Tree?. *Decentralized Thoughts* [online]. Decentralized Thinkers, [2023], 22 December 2020 [cit. 2023-04-16]. Dostupné z: https://decentralizedthoughts.github.io/2020-12-22-what-is-a-merkle-tree/

[41] RAVIKIRAN, A. S. Merkle Tree in Blockchain: What is it, How does it work and Benefits. *Simplilearn* [online]. Simplilearn Solutions, © 2009-2023, 17 Jan 2023 [cit. 2023-04-16]. Dostupné z: https://www.simplilearn.com/tutorials/blockchain-tutorial/merkle-tree-in-blockchain

[42] GERONI, Diego. Blockchain Security Algorithms Used To Protect The Blockchain Security. *101 Blockchains* [online]. 101 Blockchains, © 2023, 20 June2021 [cit. 2023-04-24]. Dostupné z: https://101blockchains.com/blockchain-security-algorithms/

[43] IREDALE, Gwyneth. Blockchain Cryptography: Everything You Need To Know. *101 Blockchains* [online]. 101 Blockchains, © 2023, 23 April 2021 [cit. 2023-04-24]. Dostupné z: https://101blockchains.com/blockchain-cryptography/

[44] MARK. Bitcoin Algorithm Explained. *Mycropedia: Educating the World on Cryptocurrency* [online]. MYC Signals, © 2023, 3 May 2022 [cit. 2023-04-24]. Dostupné z: https://www.mycryptopedia.com/bitcoin-algorithm-explained/

[45] SOARES, Lais. Understanding Keccak256: The Cryptographic Hash Function Behind Ethereum. *Linkedin* [online]. Linkedin, © 2023, 28. 3. 2023 [cit. 2023-05-22]. Dostupné z: https://www.linkedin.com/pulse/understanding-keccak256-cryptographic-hash-function-soares-m-sc-

[46] Fabric-common: Working with an offline private key. *Hyperledger Fabric SDK for Node.js* [online]. [2023], 14. 2. 2023 [cit. 2023-04-24]. Dostupné z: https://hyperledger.github.io/fabric-sdk-node/release-2.2/tutorial-sign-transaction-offline.html

[47] GERONI, Diego. How Does Blockchain Use Public Key Cryptography?. *101 Blockchains* [online]. 101 Blockchains, © 2023, 5 May 2021 [cit. 2023-04-24]. Dostupné z: https://101blockchains.com/public-key-cryptography-in-blockchain/

[48] POSTON, Howard. Blockchain and asymmetric cryptography. *Infosec* [online]. Infosec Institute, ©2023, 9 March 2021 [cit. 2023-04-24]. Dostupné z: https://resources.infosecinstitute.com/topic/blockchain-and-asymmetric-cryptography/

[49] KORE, Akshay. A brief introduction to Consensus mechanisms, Smart contracts and distributed apps on the…. *Hackernoon* [online]. Colorado: Hackernoon hq, 8 January 2018 [cit. 2023-04-24]. Dostupné z: https://hackernoon.com/a-brief-introduction-to-consensus-mechanisms-smart-contracts-and-distributed-apps-on-the-a94453d16c3a

[50] Hyperledger Architecture: Introduction to Hyperledger Business Blockchain Design Philosophy and Consensus. *Hyperledger: Blockchain Technologies for Business* [online]. (1), 6-14 [cit. 2023-04-24]. Dostupné z: https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf

[51] What Is a Merkle Tree & What Is Its Role in Blockchain?. *Bybit: Learn* [online]. Bybit.com, © 2018-2022, 9 Dec 2022 [cit. 2023-04-24]. Dostupné z: https://learn.bybit.com/blockchain/what-is-merkle-tree/

[52] Updating a channel configuration. *Hyperledger Fabric* [online]. Hyperledger, © 2020-2022 [cit. 2023-04-24]. Dostupné z: https://hyperledger-fabric.readthedocs.io/en/release-2.4/config_update.html?highlight=merkle#channel

[53] Wallet encryption. *Bitcoin Wiki* [online]. 2010, 17 September 2017 [cit. 2023-04-24]. Dostupné z: https://en.bitcoin.it/wiki/Wallet_encryption

[54] Encryption at Rest for Hyperledger Fabric on Managed Blockchain. *AWS* [online]. Amazon Web Services, © 2023 [cit. 2023-04-24]. Dostupné z: https://docs.aws.amazon.com/managed-blockchain/latest/hyperledger-fabric-dev/managed-blockchain-encryption-at-rest.html

[55] NAKOV, Svetlin. *Practivcal Cryptography for Developers* [online]. Sofia: The MIT License, 2018 [cit. 2023-04-20]. ISBN 978-619-00-0870-5. Dostupné z: https://cryptobook.nakov.com/

[56] BAILEY, David. 19 Blockchain application use cases that will surprise you. *Supplain* [online]. Supplain, 30. 4. 2022 [cit. 2023-04-22]. Dostupné z: https://supplain.io/news/blockchain-applications-use-cases

[57] Today's Best Real-World Blockchain Use Cases. *Kriptomat: Crypto But Simple* [online]. Kriptomat, © 2023 [cit. 2023-04-22]. Dostupné z: https://kriptomat.io/blockchain/real-world-blockchain-use-cases/

[58] GLOVER, Ellen. Cryptocurrency: What Is Cryptocurrency? How Is It Used?. *Built In* [online]. Built In, © 2023, 9 Nov 2022 [cit. 2023-04-22]. Dostupné z: https://builtin.com/cryptocurrency

[59] THOMPSON, David. Different Types of Cryptocurrency. *Tech Times* [online]. Tech Times, © 2023, 2 November 2021 [cit. 2023-04-22]. Dostupné z: https://www.techtimes.com/articles/267423/20211102/different-types-of-cryptocurrency.htm

[60] Blockchain in Healthcare and the Life Sciences. *CONSENSYS* [online]. CONSENSYS, © 2023 [cit. 2023-04-22]. Dostupné z: https://consensys.net/blockchain-use-cases/healthcare-and-the-life-sciences/

[61] Blockchain in Healthcare: 17 Examples to Know. *Built In* [online]. Built In, © 2023, 16 Feb 2023 [cit. 2023-04-22]. Dostupné z: https://builtin.com/blockchain/blockchain-healthcare-applications-companies

[62] YAQOOB, Ibrar, Khaled SALAH, Raja JAYARAMAN a Yousof AL-HAMMADI. Blockchain for healthcare data management: opportunities, challenges, and future recommendations. In: *Springer* [online]. Springer Nature, © 2023, 7 January 2021 [cit. 2023-04-22]. Dostupné z: https://link.springer.com/article/10.1007/s00521-020-05519-w#citeas

[63] Government and public services. *SettleMint* [online]. SettleMint, © 2023 [cit. 2023-04-22]. Dostupné z: https://www.settlemint.com/government-blockchain-use-cases/#text-3

[64] Blockchain in Government and the Public Sector. *CONSENSYS* [online]. CONSENSYS, © 2023 [cit. 2023-04-22]. Dostupné z: https://consensys.net/blockchain-use-cases/government-and-the-public-sector/

[65] How blockchain can transform government sector: E-government, e-voting, e-identities and e-documents. *Cointelegraph* [online]. Cointelegraph, © 2013 - 2023 [cit. 2023-04-22]. Dostupné z: https://cointelegraph.com/learn/how-blockchain-can-transform-government-sector

[66] Blockchain in Supply Chain Management. *CONSENSYS* [online]. CONSENSYS, © 2023 [cit. 2023-04-23]. Dostupné z: https://consensys.net/blockchain-use-cases/supply-chain-management/

[67] SINGH, Onskar. How blockchain technology is used in supply chain management?. *Cointelegraph* [online]. Cointelegraph, © 2013 - 2023, 26 Nov 2022 [cit. 2023-04-23]. Dostupné z: https://cointelegraph.com/explained/how-blockchain-technology-is-used-in-supply-chain-management

[68] ASHCROFT, Sean. Top 10 uses of blockchain in supply chain. *SuplyChain* [online]. SuplyChain, 1 March 2023 [cit. 2023-04-23]. Dostupné z: https://supplychain-digital.com/top10/top-10-uses-of-blockchain-in-supply-chain

[69] What Is Quantum Computing?. *AWS* [online]. Amazon Web Services, © 2023 [cit. 2023-04-19]. Dostupné z: https://aws.amazon.com/what-is/quantum-computing/

[70] VIDAR. Google's Quantum Computer Is About 158 Million Times Faster Than the World's Fastest Supercomputer. *Medium* [online]. Medium, 28 Feb 2021 [cit. 2023-04-19]. Dostupné z: https://medium.com/predict/googles-quantum-computer-is-about-158-million-times-faster-than-the-world-s-fastest-supercomputer-36df56747f7f

[71] The impact of quantum computing on cryptography. *Senetas: Security without compromise* [online]. Australia: Senetas, © 2023 [cit. 2023-04-19]. Dostupné z: https://www.senetas.com/the-impact-of-quantum-computing-on-cryptography/

[72] AREL, Ryan. Explore the impact of quantum computing on cryptography. *TechTarget* [online]. TechTarget, © 2000-2023, 12 May 2023 [cit. 2023-04-19]. Dostupné z: https://www.techtarget.com/searchdatacenter/feature/Explore-the-impact-of-quantum-computing-on-cryptography

[73] Shor's algorithm. *IBM Quantum* [online]. IBM Quantum, [2022] [cit. 2023-04-19]. Dostupné z: https://quantum-computing.ibm.com/composer/docs/iqx/guide/shors-algorithm

[74] OLZAK, Tom. Will Symmetric and Asymmetric Encryption Withstand the Might of Quantum Computing?. *Spiceworks Inc.* [online]. Spiceworks, © 2006 - 2023, 14 June 2021 [cit. 2023-04-19]. Dostupné z: https://www.spiceworks.com/it-security/data-security/articles/will-symmetric-and-asymmetric-encryption-withstand-the-might-of-quantum-computing/

[75] XU, Tammy. Cryptographers Are Racing Against Quantum Computers. *Built In* [online]. Built In, © 2023, 30 Apr 2021 [cit. 2023-04-19]. Dostupné z: https://builtin.com/cybersecurity/post-quantum-cryptography

[76] State of Symmetric & Hash Algorithms after Quantum Computing. *REAL security* [online]. Slovenia: REAL security, © 2021, 31 Aug 2019 [cit. 2023-04-20]. Dostupné z: https://www.real-sec.com/2019/08/state-of-symmetric-hash-algorithms-after-quantum-computing/

[77] MATEEN, Abdul. What is post-quantum cryptography?. *Educative* [online]. Educative, ©2023 [cit. 2023-04-20]. Dostupné z: https://www.educative.io/answers/what-is-post-quantum-cryptography

[78] PATHAK, Amrita. Lattice-Based Cryptography Explained in 5 Minutes or Less. *Geekflare* [online]. Geekflare, © 2023, 3 May 2023 [cit. 2023-04-20]. Dostupné z: https://geekflare.com/lattice-based-cryptography/

[79] Lattice-Based Cryptography. *ISARA* [online]. ISARA, © 2023 [cit. 2023-04-20]. Dostupné z: https://www.isara.com/blog-posts/lattice-based-cryptography.html

[80] Math Paths to Quantum-safe Security: Hash-based Cryptography. *ISARA* [online]. ISARA, © 2023, 24 February 2020 [cit. 2023-04-20]. Dostupné z: https://www.isara.com/blog-posts/hash-based-cryptography.html

[81] What is Hash-based Cryptography?. *Utimaco* [online]. Utimaco Management [cit. 2023-04-20]. Dostupné z: https://utimaco.com/products/technologies/post-quantum-cryptography/what-hash-based-cryptography

[82] COOK, John D. Mixing error-correcting codes and cryptography. *John D. Cook: Consulting* [online]. 23 March 2019 [cit. 2023-04-20]. Dostupné z: https://www.johndcook.com/blog/2019/03/23/code-based-cryptography/

[83] Multivariate cryptography. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2023, 21 October 2022 [cit. 2023-04-20]. Dostupné z: https://en.wikipedia.org/wiki/Multivariate_cryptography

[84] Quantum Computing and Blockchain: What You Need to Know. *SupraOracles* [online]. SupraOracles, ©2023, 3 May 2023 [cit. 2023-04-21]. Dostupné z: https://supraoracles.com/academy/quantum-computing-and-blockchain-what-you-need-to-know/

[85] Quantum Computing vs. Blockchain: A Complete Guide. *Shardeum* [online]. Shardeum, © 2022, 19 October 2022 [cit. 2023-04-21]. Dostupné z: https://shardeum.org/blog/quantum-computing-vs-blockchain/

[86] Cryptocurrency vs. quantum computing: A deep dive into the future of cryptocurrencies. *Cointelegraph* [online]. Cointelegraph, © 2013 - 2023 [cit. 2023-04-21]. Dostupné z: https://cointelegraph.com/learn/cryptocurrency-vs-quantum-computing-a-deep-dive-into-the-future-of-cryptocurrencies

[87] DENYS. Hashgraph vs Blockchain: Analytical Tech Comparison. *Intellectsoft: Blockchain Lab* [online]. Intellectsoft, © 2023, 25 July 2019 [cit. 2023-04-21]. Dostupné z: https://blockchain.intellectsoft.net/blog/hashgraph-vs-blockchain/

[88] Difference Between Hashgraph and Blockchain. *Simplilearn* [online]. Simplilearn Solutions, © 2009-2023, 27 Feb 2023 [cit. 2023-04-21]. Dostupné z: https://www.simplilearn.com/hashgraph-vs-blockchain-article

[89] MAX, David. Hashgraph Vs Blockchain: A Quick Comparison with Pros & Cons. *TEMOK* [online]. TEMOK, © 2022, 26 January 2022 [cit. 2023-04-21]. Dostupné z: https://www.temok.com/blog/hashgraph-vs-blockchain/

[90] GALINDO SALGADO, Pablo. What's New In Python 3.11. *Python* [online]. Python, 2023, 11. March 2023 [cit. 2023-03-12]. Dostupné z: https://docs.python.org/3/whatsnew/3.11.html

[91] Python Developers Survey 2021 Results. *JetBrains* [online]. JetBrains, © 2000-2023, 2021 [cit. 2023-03-17]. Dostupné z: https://lp.jetbrains.com/python-developers-survey-2021/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

SVG	Scalable Vector Graphics (Škálovatelná Vektorová Grafika)
GUI	Graphical User Interface (Grafické Uživatelské Rozhraní)
ID	Identifikátor
CRUD	Create Read Update Delete (Vytvořit Číst Aktualizovat Vymazat)
PoW	Proof-of-Work (česky Důkaz o práci)
PoS	Proof-of-Stake (česky Důkaz o podílu)
DApps	Decentralized application (česky Decentralizovaná aplikace)
SHA	Secure Hash Algorithm (česky Bezpečný Hash Algoritmus)
MD	Message Digest
ECDSA	Eliptic Curve Digital Signature Algorithm (česky Algoritmus Digitálního Podpisu Eliptickou Křivkou)
RSA	Rivest-Shamir-Adleman
BLS	Bohen-Lynn-Shacham
ECB	Electronic Codebook Mode (česky Režim elektronické kódové knihy)
CBC	Cipher Block Chaining Mode (česky Režim řetězení šifrových bloků)
CFB	Ciphertext Feedback Mode (česky Režim zpětné vazby šifrového textu)
OFB	Output Feedback Mode (česky Režim výstupní zpětné vazby)
CTR	Counter Mode (česky Režim čítače)
XOR	Exkluzivní OR
IV	Inicializační vektor
HTTPS	Hypertext Transfer Protocol Secure
ECC	Eliptic Curve Cryptography (česky Kryptografie Eliptických Křivek)
XMSS	eXtended Merkle Signature Scheme (česky Rozšířené Merklovo podpisové schéma)

SEZNAM OBRÁZKŮ

Obrázek 1 – Propojení bloků v blockchainu [7] 10
Obrázek 2 – Proces přidávání bloku [7] 13
Obrázek 3 – Rozdělení typů blockchainů [9] 15
Obrázek 4 – Obousměrné kryptografické algoritmy [26] 20
Obrázek 5 – Hash funkce [27] 21
Obrázek 6 – Hashovací funkce SHA a MD (vlastní zdroj) 22
Obrázek 7 – Digitální podpis [28] 22
Obrázek 8 – Eliptická křivka [31] 23
Obrázek 9 – ECDSA vs. Schnorr algoritmus [32] 24
Obrázek 10 – Symetrické šifrování [34] 25
Obrázek 11 – ECB bloková šifra [36] 25
Obrázek 12 – CBC bloková šifra [36] 26
Obrázek 13 – CFB bloková šifra [36] 26
Obrázek 14 – OFB bloková šifra [36] 27
Obrázek 15 – CRT bloková šifra [36] 27
Obrázek 16 – Proudová šifra [35] 28
Obrázek 17 – Asymetrické šifrování [34] 29
Obrázek 18 – Konstrukce Merkle Tree [40] 30
Obrázek 19 – Ověření souboru v Merkle Tree [40] 30
Obrázek 20 – Používané algoritmy [44] [45] [46] [50] [51] [53] [54] [55] 32
Obrázek 21 – Kryptoměny [59] 34
Obrázek 22 – Propojení zdravotnictví pomocí blockchainu [62] 35
Obrázek 23 – Bit a Qubit [71] 37
Obrázek 24 – Nejlepší klasický vs. Shorův algoritmus [73] 38
Obrázek 25 – Dopad kvantových počítačů na šifrování [74] 39
Obrázek 26 – Typy post-kvantové kryptografie [77] 40
Obrázek 27 – Příklad mřížky u kryptografie založené na mřížce [79] 40
Obrázek 28 – Hashgraph [87] 43
Obrázek 29 – Porovnání Hashgraph a Blockchain [87] 43
Obrázek 30 – Průzkum oblíbenosti vývojových prostředí pro Python [91] 47
Obrázek 31 – Kostra bloku v blockchainu (vlastní zdroj) 48
Obrázek 32 – Struktura projektu (vlastní zdroj) 49

Obrázek 33 – Kontrola, jestli skript je spuštěn jako hlavní program (vlastní zdroj) ..50
Obrázek 34 – Úvodní inicializace třídy StartWindow (vlastní zdroj) 50
Obrázek 35 – Grafické rozhraní úvodního okna (vlastní zdroj) 50
Obrázek 36 – Kontrola existence výstupních složek (vlastní zdroj) 51
Obrázek 37 – Otevření blockchainu z úvodní obrazovky (vlastní zdroj) 51
Obrázek 38 – Vytvoření nového blockchainu z úvodní obrazovky (vlastní zdroj) 52
Obrázek 39 – Grafické rozhraní při vytváření nového blockchainu (vlastní zdroj) ...52
Obrázek 40 – Zdrojový kód pro GUI vytvoření nového blockchainu (vlastní zdroj) 53
Obrázek 41 – Hlavní GUI pro přidávání bloků do blockchainu (vlastní zdroj) 54
Obrázek 42 – Úvodní inicializace třídy BlockChainGUI (vlastní zdroj) 55
Obrázek 43 – Vytvoření nového blockchainu z hlavního okna (vlastní zdroj) 55
Obrázek 44 – Funkce pro vizuální úpravu hlavního okna (vlastní zdroj) 56
Obrázek 45 – Otevření již existujícího blockchainu z hlavního okna (vlastní zdroj).56
Obrázek 46 – Potvrzení přidání bloku do blockchainu (vlastní zdroj) 57
Obrázek 47 – Podpůrné funkce (vlastní zdroj) 58
Obrázek 48 – Vytvoření bloku v blockchainu (vlastní zdroj) 59
Obrázek 49 – Vytvoření úvodního bloku v blockchainu (vlastní zdroj) 60
Obrázek 50 – Obrázek úvodního bloku diagramu (vlastní zdroj) 61
Obrázek 51 – Přidání dalšího bloku do blockchainu (vlastní zdroj).....61
Obrázek 52 – Obrázek dalšího přidaného bloku do diagramu (vlastní zdroj).....62
Obrázek 53 – Vytváření bloků a přidávání je do blockchainu (vlastní zdroj).....63
Obrázek 54 – Vizuální úprava zobrazení Hash hodnoty (vlastní zdroj).....65
Obrázek 55 – Rozdělení textu a dlouhých slov (vlastní zdroj).....66
Obrázek 56 – Spojení slov a rozdělení textu na řádky (vlastní zdroj).....66
Obrázek 57 – Úprava dat před vytvořením prvního bloku (vlastní zdroj).....67
Obrázek 58 – Zdrojový kód vytvoření prvního bloku diagramu (vlastní zdroj) 68
Obrázek 59 – Přidání dalšího bloku do diagramu (vlastní zdroj).....69
Obrázek 60 – Funkce pro generování diagramu a obrázku (vlastní zdroj).....70

SEZNAM PŘÍLOH

PŘÍLOHA P I: OBSAH CD

PŘÍLOHA P I: OBSAH CD

Struktura obsahu přiloženého CD:

- Adresář **Text bakalářské práce** – obsahuje text diplomové práce ve formátu PDF
- Adresář **Zdrojové kódy** – obsahuje všechny zdrojové kódy a soubory grafických rozhraní potřebné ke spuštění aplikace