**UNIVERZITA OBRANY**

**FAKULTA VOJENSKÝCH TECHNOLOGIÍ**

**Studijní program: Vojenské technologie**

Studijní obor: Informační technologie

Ev. číslo: xxxx/22



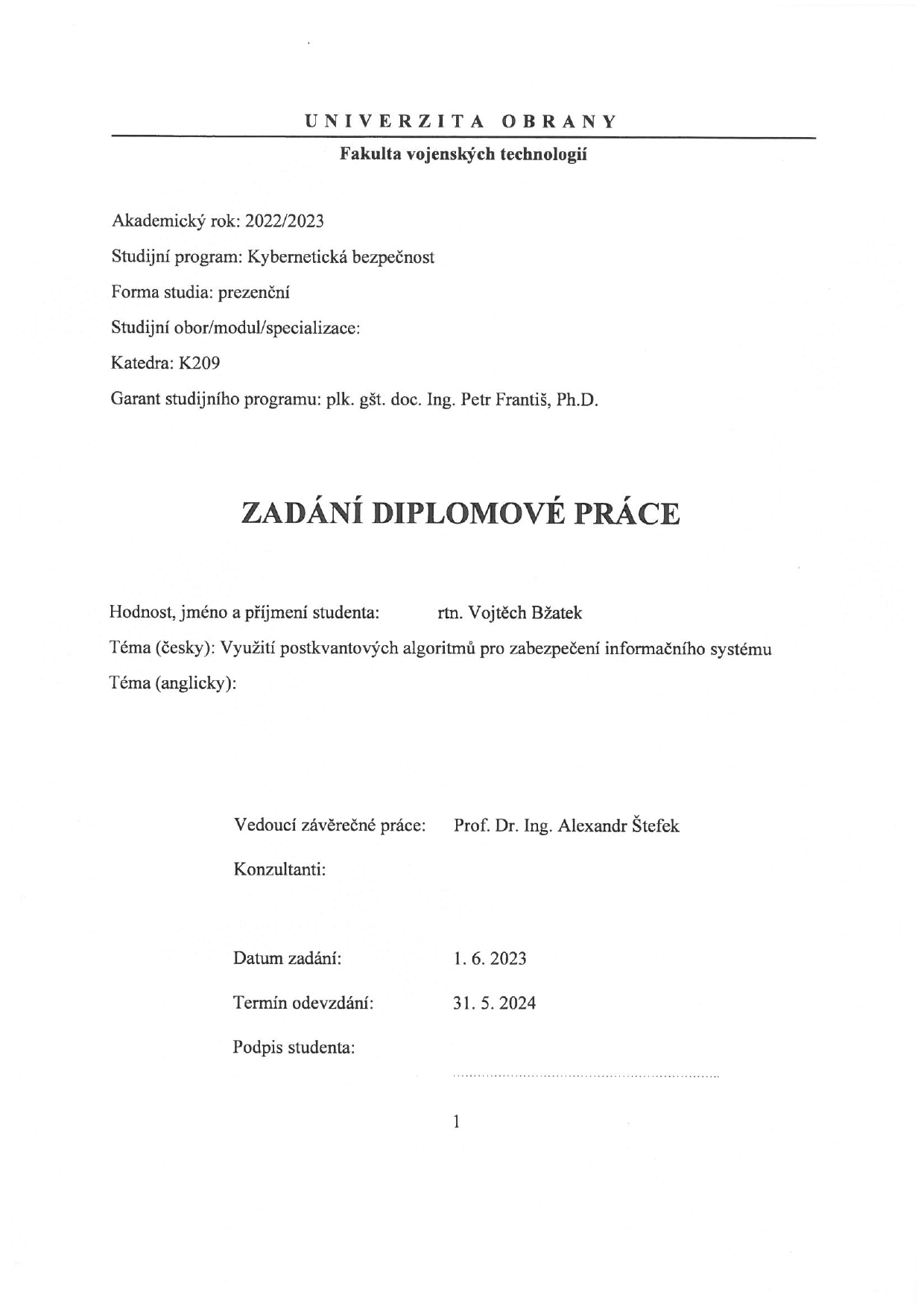
**DIPLOMOVÁ PRÁCE**

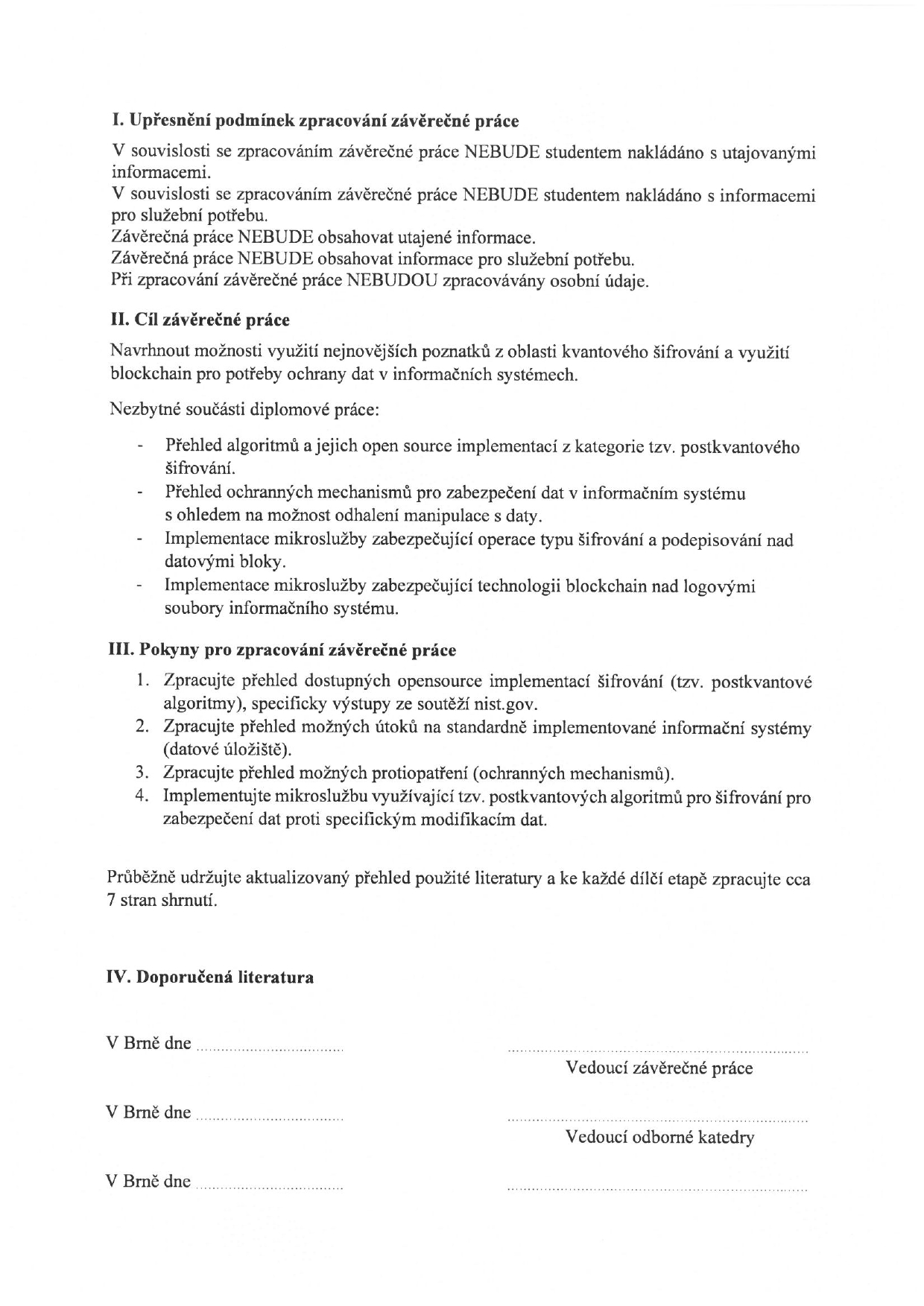
Název: Využití postkvantových algoritmů pro zabezpečení informačního systému

Zpracoval: Vojtěch Bžatek

Vedoucí práce: Prof. Dr. Ing. Alexandr Štefek

BRNO 2022





**PODĚKOVÁNÍ (nepovinné)**

**ČESTNÉ PROHLÁŠENÍ**

Prohlašuji, že jsem zadanou diplomovou práci na téma „xxx“ vypracoval samostatně, pod odborným vedením vedoucího xxx a použil jsem pouze literární zdroje uvedené v práci.

Dále prohlašuji, že jsem seznámen s tím, že se na moji diplomovou práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména skutečnosti, že Univerzita obrany má právo na uzavření licenční smlouvy o užití této bakalářské (diplomové) práce jako školního díla podle § 60 odst. 1 výše uvedeného zákona, a s tím, že pokud dojde k užití této diplomové práce mnou nebo bude poskytnuta licence o užití díla třetímu subjektu, je Univerzita obrany oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladu, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše. Souhlasím se zpřístupněním své diplomové práce pro prezenční studium v prostorách knihovny Univerzity obrany.

V Brně, dne xx. května. 2024

..........................................

Jméno studenta

**ABSTRAKT**

Tato diplomová práce řeší problematiku využití postkvantových kryptografických algoritmů a blockchainové sítě v informačních systémech. V první části se věnuji popisu postkvantové kryptografie, uvádím zavedené algoritmy a popisuji možnosti jejich implementace. Ve druhé části představuji mikroslužbu pro kvantově bezpečnou komunikaci. Ve třetí části popisuji blockchain a základní parametry blockchainové sítě. Na základě těchto informací jsem vyvinul vlastní blockchainovou síť pro posílání a uchovávání logových záznamů z koncových zařízení. Parametry a funkcionality této sítě uvádím ve čtvrté části včetně postupu pro spuštění na vlastních zařízení. V poslední části uvádím sjednocení postkvantové kryptografie a blockchainové sítě do celistvé aplikace, včetně návodu pro práci s ní. Tato diplomová práce může sloužit jako základ pro využití moderních bezpečnostních mechanismů pro ochranu dat v informačních systémech. Jednotlivé mikroslužby jsou popsány tak, aby na práci bylo možno navázat a rozšířit je až do zcela nasaditelných řešení.

**Klíčová slova:**

**Kryptografie, Postkvantové algoritmy, blockchain, blockchainová síť**

**ABSTRACT**

This diploma thesis deals with the issue of the use of post-quantum cryptographic algorithms and the blockchain network in information systems. In the first part, I describe post-quantum cryptography, introduce established algorithms and describe the possibilities of their implementation. In the second part, I present a microservice for quantum secure communication. In the third part, I describe the blockchain and the basic parameters of the blockchain network. Based on this information, I developed my own blockchain network for sending and storing log records from end devices. I present the parameters and functionalities of this network in the fourth part, including the procedure for starting it on your own devices. In the last part, I present the unification of post-quantum cryptography and the blockchain network into a complete application, including instructions for working with it. This thesis can serve as a basis for the use of modern security mechanisms for data protection in information systems. Individual microservices are described in such a way that the work can be followed up and expanded to fully deployable solutions.

**Key words:**

**Cryptography, Post quantum algorithms, blockchain, blockchain network**

Obsah

[SEZNAM POUŽITÝCH ZKRATEK 11](#_Toc165156049)

[SEZNAM OBRÁZKŮ 12](#_Toc165156050)

[SEZNAM TABULEK 13](#_Toc165156051)

[ÚVOD 14](#_Toc165156052)

[1.1 Cíle práce a její omezení 15](#_Toc165156053)

[2 Postkvantová kryptografie 16](#_Toc165156054)

[2.1 Shrnutí klasické kryptografie 16](#_Toc165156055)

[2.2 Shorův algoritmus 20](#_Toc165156056)

[2.3 Motivace postkvantové kryptografie 20](#_Toc165156057)

[2.4 NIST a jeho zapojení do pokroku v postkvantové kryptografii 20](#_Toc165156058)

[2.5 Vybrané post-kvantové algoritmy pro standardizaci 22](#_Toc165156059)

[2.5.1 Kryptografie založená na mřížkách 23](#_Toc165156060)

[2.5.2 Kryptografie založená na hashovacích funkcích 24](#_Toc165156061)

[2.5.3 CRYSTALS-Kyber 25](#_Toc165156062)

[2.5.4 CRYSTALS-DILITHIUM 25](#_Toc165156063)

[2.5.5 Falcon 25](#_Toc165156064)

[2.5.6 SPHINCS+ 25](#_Toc165156065)

[3 Mikroslužba zabezpečující postkvantovou kryptografii 27](#_Toc165156066)

[3.1 Implementace postkvantových algoritmů 27](#_Toc165156067)

[3.1.1 Společné knihovny 27](#_Toc165156068)

[3.1.2 Pqcrypto 29](#_Toc165156069)

[3.1.3 KYBER\_PY 30](#_Toc165156070)

[3.1.4 FALCON\_official 30](#_Toc165156071)

[3.1.5 SPHINCSPlus\_Tottifi 30](#_Toc165156072)

[3.2 PQC-service 30](#_Toc165156073)

[3.2.1 Nastavení symetrického klíče 31](#_Toc165156074)

[3.2.2 Získání podpisových klíčů a práce s nimi 34](#_Toc165156075)

[3.2.3 Průběh komunikace při odeslání zprávy 37](#_Toc165156076)

[3.2.4 Průvodce pro spuštění mikroslužby 38](#_Toc165156077)

[4 Přehled ochranných mechanismů pro zabezpečení dat v informačním systému s ohledem na možnost odhalení manipulace s daty 40](#_Toc165156078)

[5 Blockchain 41](#_Toc165156079)

[5.1 Řetěz 41](#_Toc165156080)

[5.2 Blockchainová síť 42](#_Toc165156081)

[5.2.1 Veřejná síť 42](#_Toc165156082)

[5.3 Mechanismy konsensu 42](#_Toc165156083)

[5.3.1 PoW (Proof of work) 43](#_Toc165156084)

[5.3.2 PoS (Proof of stake) 43](#_Toc165156085)

[6 Mikroslužba zabezpečující technologii blockchain nad logovými záznamy informačního systému. 45](#_Toc165156086)

[6.1 Popis jednotlivých částí 45](#_Toc165156087)

[6.1.1 Uzel 45](#_Toc165156088)

[6.1.2 Klient/http dotazy 45](#_Toc165156089)

[6.2 Funkcionality uzlu 46](#_Toc165156090)

[6.2.1 Třída Blockchain 46](#_Toc165156091)

[6.2.2 Registrace v síti 47](#_Toc165156092)

[6.2.3 Žádost o tabulku uzlů 48](#_Toc165156093)

[6.2.4 Přijetí zprávy 48](#_Toc165156094)

[6.2.5 Těžba 50](#_Toc165156095)

[6.2.6 Ověření řetězu 54](#_Toc165156096)

[6.2.7 Poskytnutí řetezu 54](#_Toc165156097)

[6.2.8 Průvodce pro spuštění mikroslužby 54](#_Toc165156098)

[7 Postkvantová blockchainová síť 56](#_Toc165156099)

[7.1 Výměna komunikační funkce 56](#_Toc165156100)

[7.1.1 Doplnění koncový bodů o KEM algoritmus 57](#_Toc165156101)

[7.1.2 Certifikační autorita – nezbytná součást kryptograficky zabezpečené sítě 57](#_Toc165156102)

[7.1.3 Ukázka fungování postkvantové blockchainové sítě 58](#_Toc165156103)

[7.1.4 Průvodce pro spuštění Postkvantové blockchainové sítě 58](#_Toc165156104)

[7.1.5 Ukazka fungovani 59](#_Toc165156105)

[8 Možnosti dalšího rozvoje 60](#_Toc165156106)

[8.1 Zavedení plné certifikační autority 60](#_Toc165156107)

[8.2 Implementace autorských řešení algoritmů CRYSTALS Kyber a CRYSTALS DILITHIUM 60](#_Toc165156108)

[8.3 Automatizace těžby a porovnání řetězů 60](#_Toc165156109)

[8.4 Vstupní řetěz do sítě 61](#_Toc165156110)

[8.5 Autentizace entit 61](#_Toc165156111)

[8.6 Proof of work 61](#_Toc165156112)

[8.7 Otestovat hranice schopností sítě 62](#_Toc165156113)

[ZÁVĚR 63](#_Toc165156114)

[SEZNAM POUŽITÉ LITERATURY 64](#_Toc165156115)

[SEZNAM PŘÍLOH 66](#_Toc165156116)

# SEZNAM POUŽITÝCH ZKRATEK

|  |  |
| --- | --- |
| AES | Advanced Encryption Standard |
| CA | Certifikační autorita |
| CVP | Closest Vector Problem Postkvantová kryptografie |
| DES | Data Encryption Standard |
| DH | Diffie-Hellman |
| DSA | Digital Signature Algorithm |
| ECDH | Elliptic-curve Diffie-Hellman |
| ECDSA | The Elliptic Curve Digital Signature Algorithm |
| HTTP | Hypertext Transfer Protocol |
| IP | Internet Protocol |
| JSON | JavaScript Object Notation |
| JWS | JSON Web Signature |
| KEM | Key Encapsulation Mechanism |
| NIST | National Institute of Standards and Technology |
| PoS | Proof of Stake |
| PoW | Proof of Work |
| PQC | Post-quantum cryptography |
| RFC | Request for Comments |
| RSA | Rivest–Shamir–Adleman |
| SHA | Secure Hash Algorithm |
| SVP | Shortest Vector Problem |
| TLS | Transport Layer Security |

# SEZNAM OBRÁZKŮ

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Strana |
| Obrázek 1 | − | Symetrická kryptografie |  |
| Obrázek 2 | − | Asymetrická kryptografie |  |
| Obrázek 3 | − | KEM vs Diffie-Hellman (Metody výměny klíčů) |  |
| Obrázek 4 | − | Digitální podpisy |  |
| Obrázek 5 | − | Mříž se dvěma různými základnami |  |
| Obrázek 6 | − | Třída pro KEM algoritmus |  |
| Obrázek 7 | − | Třída pro podpisový algoritmus |  |
| Obrázek 8 | − | Funkce pro výměnu symetrického klíče |  |
| Obrázek 9 | − | Popis výměny symetrického klíče pro první spojení mezi Alicí a CA |  |
| Obrázek 10 | − | Popis výměny symetrického klíče mezi Alicí a Bobem |  |
| Obrázek 11 | − | Popis žádosti o soukromý podpisový klíč |  |
| Obrázek 12 | − | Funkce pro získání veřejného podpisové klíče |  |
| Obrázek 13 | − | Popis žádosti o veřejný podpisový klíč |  |
| Obrázek 14 | − | Funkce pro odeslání zprávy |  |
| Obrázek 15 | − | Schéma odeslání zprávy |  |
| Obrázek 16 | − | Struktura blockchainové databáze |  |
| Obrázek 17 | − | http požadavek (registrace v síti) |  |
| Obrázek 18 | − | http požadavek (nastavení tabulky uzlů) |  |
| Obrázek 19 | − | http požadavek (žádost o řetěz) |  |
| Obrázek 20 | − | Zaregistrování uzlu v síti |  |
| Obrázek 21 | − | http požadavek (žádost o tabulku uzlů) |  |
| Obrázek 22 | − | http požadavek (žádost o tabulku uzlů) |  |
| Obrázek 23 | − | Schéma přijetí zprávy |  |
| Obrázek 24 | − | http požadavek (zahájení těžby) |  |
| Obrázek 25 | − | http požadavek (příkaz k zahájení těžby) |  |
| Obrázek 26 | − | Schéma zahájení těžby |  |
| Obrázek 27 | − | Schéma těžení bloku a distribuce výsledku |  |
| Obrázek 28 | − | http požadavek (příkaz k porovnání řetězů) |  |
| Obrázek 29 | − | http požadavek (žádost o řetěz) |  |
| Obrázek 30 | − | Nešifrovaná komunikace |  |
| Obrázek 31 | − | Šifrovaná komunikace |  |
| Obrázek 32 | − | Koncový bod pro KEM výměnu klíčů |  |
|  |  |  |  |

# SEZNAM TABULEK

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Strana |
| Tabulka 1 | − | Bezpečnostní kategorie PQC - NIST | … XX |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# ÚVOD

„Svět je nebezpečné místo k životu, ne kvůli lidem, kteří jsou zlí, ale kvůli lidem, kteří s tím nic neudělají.“ Je tomu téměř 70 let, kdy nás opustil geniální fyzik Albert Einstein – autor tohoto citátu. Vybral jsem ho pro úvod své diplomové práce schválně. Pochopil jsem citát tak, že pokud člověk může udělat něco pro bezpečnost nejenom nás, lidí, ale v celkovém důsledku i celé planety, tak by to udělat měl. Tento citát byla moje základní myšlenka při rozhodování se o diplomové práci.

Téma mé diplomové práce a jeho cíle jsem si, po diskusi s vedoucím, vybral vlastní. Chtěl jsem se zaměřit na možnosti využití nejnovějších poznatků z oblasti kvantového šifrování a použití blockchain pro potřeby ochrany dat v informačních systémech.

Už v průběhu studia mě zaujaly různé kryptografické algoritmy, jejich odolnosti a jakým způsobem přispívají k bezpečné komunikaci přes otevřenou datovou síť. Zároveň jsem si byl já, tak jako i spousta velmi chytrých lidí, vědom hrozby v možná brzkém příchodu kvantových počítačů. Tyto počítače významně naruší odolnost běžných, denně používaných, kryptografických algoritmů [h1, str 1]. Samozřejmě svět před touto hrozbou nezavřel oči. V roce 2017 byla vyhlášena soutěž o kvantově bezpečný kryptografický algoritmus a po pěti letech, v roce 2022, byly zveřejněny výsledky [!!]. Svět si tak mohl částečně oddychnout. Řešení odolnosti proti kvantovým počítačům existuje. Zbývá je jen začít používat.

Ve své práci vycházím právě z těchto kryptografických algoritmů. Nevytvářím žádný nový, svůj vlastní, kvantově odolný algoritmus, ale popisuji, jak implementovat ty, které byly prověřeny a schváleny odborníky na kryptografii. Využívám jak řešení přímo od autorů post-kvantových kryptografických algoritmů, tak i řešení třetích stran. Tedy těch, kdo matematické rovnice, podle instrukcí autorů převedli do spustitelných kódů. Jejich práce pak upravuji tak, aby mohly být využity pro přenos dat mezi dvěma počítači.

Bezpečnost informačních systémů ovšem nekončí jen v samotné komunikaci, byť je jedna, z mého pohledu, klíčových, ale pokračuje také v samotném uchování dat. Zeptal jsem se sám sebe na triviální otázky. Je možné, aby byla data nezměnitelná i v případě, kdy mám veškerá oprávnění pro přístup k těmto datům? Kdo mi zabrání, abych neupravil data, vlastnictví informačního systému, pokud mohu beze stopy číst, mazat a upravovat vše, co se v informačním systému nachází? Ano, současné ochranné mechanismy, logování aktivity, přímý dohled a omezené oprávnění prakticky takovou situaci neumožňují. Jenže to neznamená, že se tato hrozba rovná pravděpodobnostní nule. Zamyslel jsem se tedy nad možnostmi, jak se tato problematika řeší v jiných odvětvích. I díky základním znalostem načerpaných v průběhu studia oboru Kybernetická bezpečnost jsem se rozhodl zaměřit na využití blockchainové sítě, která se běžně používá pro zabezpečení kryptoměnových systémů. Po prozkoumání dostupných blockchainových sítí jsem se kvůli náročnosti, a především nekompatibilitě s mým zámyslem, rozhodl vyvinout síť vlastní.

První část této práce, postkvantová bezpečnost, se zabývá zajištěním důvěrnosti dat. Tedy jednou z nutných podmínek logické vrstvy bezpečnosti pro ochranu dat v informačním systému [!!]. Druhá část, blockchainová síť, se zabývá integritou dat. To je druhá z nutných podmínek [!!]. Tyto podmínky jsou nutné, nikoliv dostatečné, a proto spolu s dostupností je pro zabezpečení ochrany dat v informačním systému nutné aplikovat všechny naráz.

Byť jsem si to přímo nespecifikoval ve svém zadání práce, rozhodl jsem se do své diplomové práce přidat ještě třetí část, a tou je post-kvantová blockchainová síť. Shrnuje obě problematiky do jedné celistvé aplikace, a proto ji považuji za vrchol této diplomové práce.

## Cíle práce a její omezení

Cílem této diplomové práce je předložit důkaz o využitelnosti blockchainové sítě pro uchování dat v informačním systému, důkaz o možnosti použití post-kvantových algoritmů pro běžnou komunikaci a poskytnout přehled post-kvantových algoritmů a jejich implementací.

Splnění prvního cíle, možnost využitelnosti blockchainové sítě pro uchování dat v informační systému, jsem se rozhodl demonstrovat na funkční mikroslužbě, kterou podrobně popíšu, představím popis jednotlivých částí sítě a vysvětlím nezbytné funkcionality, které zajišťují chod sítě.

V práci popíšu, které algoritmy jsou vybrány pro bezpečnou post-kvantovou komunikaci. Popíšu odkud lze vzít jejich spustitelné knihovny a jak je implementovat do svých programovacích kódů. Představím také mikroslužbu, která zajišťuje kvantově bezpečnou komunikaci. Představím její jednotlivé části a průběh nezbytné komunikace pro vytvoření kvantově bezpečného kanálu. Tím splním i druhý a třetí cíl.

Vývoj mikroslužeb budu vykonávat na operačním systému Windows 11. Rozhodl jsem se tak z důvodu jeho širokého rozšíření. Práce je testována i na starším operačním systému Windows 10.

Jako programovací jazyk jsem se rozhodl použít Python verze 3.12. Python je oblíbený programovací jazyk s velkou uživatelskou komunitou a mám s ním nejvíce zkušeností. Verze 3.12 je v době psaní práce nejaktuálnější verzí a poskytuje podporu až do října 2028 [04].

Pro bezpečnou komunikaci je potřeba využít certifikační autoritu, která poskytuje podpisové klíče k podepisování a ověřování zpráv. Vytvořit, nebo použít certifikační autoritu třetí strany není cílem této diplomové práce, a proto využívám jen její lehkou náhradu. Tato pseudocertifikační autorita, kterou používám, nesplňuje všechny potřebné standardy, včetně těch bezpečnostních [!!]. Například v sobě nezahrnuje autentizaci uživatele. Certifikační autoritu nepovažuji za součást mé diplomové práce a budu se o ni zmiňovat jen v nejnutnějších případech.

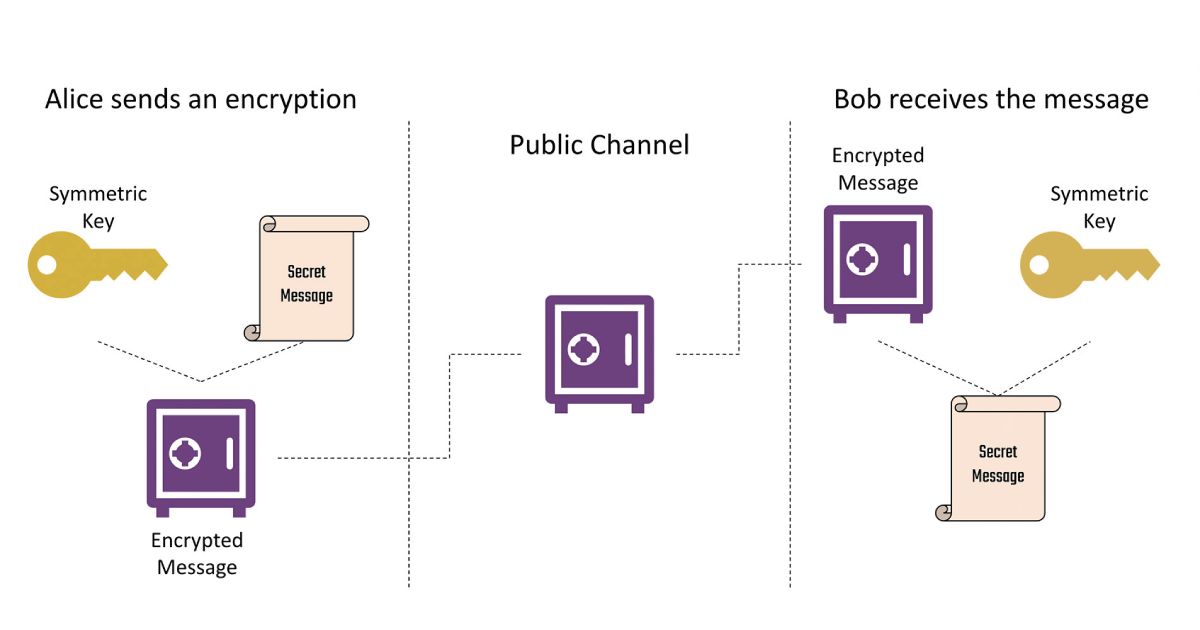
# Postkvantová kryptografie

V této kapitole představím kryptografické algoritmy, které jsou považovány za kvantově odolné. Než tak učiním, nejprve bych chtěl shrnout základní principy klasické kryptografie a objasnit proč postkvantovou kryptografii vůbec potřebujeme, proti čemu nás brání a kdo rozhodoval, jaké algoritmy budou vybrány jako ty nejlepší.

## Shrnutí klasické kryptografie

**Symetrické kryptografie**

Kryptografové historicky řešili problém, jak umožnit dvěma stranám, jako jsou Alice a Bob, bezpečně komunikovat přes veřejný kanál, který může být odposloucháván třetí stranou, například Evou. Představme si, že Alice chce poslat Bobovi zprávu: "Zítra nebudu doma. Peníze mám pod matrací." ale nemá k dispozici žádný soukromý komunikační kanál. Aby zabránila Evě v odposlechu a potenciálním odcizení peněz, Alice využije šifrování. Když byli Alice a Bob naposledy spolu, dohodli se na tajném klíči pro šifrování zpráv. Alice nyní může svou zprávu zašifrovat pomocí tohoto klíče, což je obdobné jako vložení zprávy do zamčené schránky, a Bob ji pak může dešifrovat stejným klíčem, což je jako otevření této schránky. Eva, i když zprávu zachytí, nebude moci schránku otevřít, protože nemá správný klíč. Tento způsob šifrování, kdy obě strany používají stejný klíč, se nazývá symetrická kryptografie. [h2]

Obrázek 1 - Symetrická kryptografie[s2]

**Asymetrická kryptografie**

V případě, že Alice a Bob chtějí komunikovat v tajnosti, ale nemohou se osobně setkat k výměně klíčů, mohou využít metodu kryptografie založené na veřejném a soukromém klíči. Bob si vytvoří dva klíče: jeden veřejný, který může bezpečně sdílet s celým světem a který slouží k šifrování zpráv určených pro něj, a druhý soukromý, který si nechá pro sebe a používá ho k dešifrování přijatých zpráv. Když Alice chce Bobovi poslat šifrovanou zprávu, najde jeho veřejný klíč a s jeho pomocí zprávu zašifruje. Tento proces zajišťuje, že i kdyby někdo jako Eva zprávu zachytil, bez přístupu k Bobovu soukromému klíči by viděl pouze nesrozumitelný text. Po přijetí zprávy Bob použije svůj soukromý klíč k jejímu dešifrování a může si tak přečíst Alicinu zprávu. Tento způsob šifrování založený na veřejném a soukromém klíči se nazývá asymetrická kryptografie. [h2]

Obsah obrázku text, snímek obrazovky, Písmo

Popis byl vytvořen automaticky

Obrázek 2 - Asymetrická kryptografie[s3]

V praxi je nepravděpodobné, že by se strany dokázaly dohodnout na klíči předem. Zamysleme se nad tím, jak často musíte zadávat své bankovní údaje, abyste mohli nakupovat na webové stránce, kterou jste nikdy předtím nenavštívili. V důsledku toho jsou bezpečné techniky pro kryptografii veřejného klíče pro bezpečnost internetu nanejvýš důležité. Bohužel se ukazuje, že vytváření schémat s veřejnými klíči je docela obtížné a obvykle to trvá příliš dlouho, když chcete poslat spoustu informací. Naštěstí to jde snadno opravit. Alice a Bob začínají pomocí mechanismu DH (Diffie-Hellman), nebo KEM (Key Encapsulation Mechanism), aby si tajně předali symetrický klíč. Poté použijí tento sdílený klíč s dobře zavedeným symetrickým schématem, typicky schématem nazývaným AES (Advanced Encryption Standard), aby pokračovali ve své tajné komunikaci bez režie, kterou vyžadují techniky veřejného klíče. Takto běžné internetové standardy, jako je TLS (Transport Layer Security), navrhují, aby byla v praxi zabezpečena komunikace. [h2]

Obsah obrázku text, snímek obrazovky, diagram, řada/pruh

Popis byl vytvořen automaticky

Obrázek 3 – KEM vs Diffie-Hellman (Metody výměny klíčů) [s5]

Symetrická šifrovací metoda AES, je při správném použití považována za velmi bezpečnou. Nicméně, klíčovým bodem pro zabezpečení je fáze s veřejným klíčem. Bezpečnost systémů s veřejným klíčem je často založena na složitosti určitých matematických problémů, které jsou výpočetně náročné. Například, RSA šifrovací systém využívá veřejný klíč, který je velké číslo n, a soukromý klíč, který obsahuje informace umožňující faktorizaci n na dva prvočíselné faktory p a q, kde n = pq. Alice může zašifrovat zprávu pro Boba pomocí n, ale k dešifrování je potřeba znát p a q. Pro Boba je to jednoduché, protože zná obě prvočísla, ale pro každého, kdo zachytí zprávu, jako je Eva, je výzvou získat p a q faktorizací n. Shorův algoritmus, který se zabývá problémem faktorizace, je na současných počítačích neefektivní, pokud jsou p a q dostatečně velká. Bezpečnost jiných metod výměny veřejných klíčů je odvozena z jiných složitých matematických problémů, jako je například problém diskrétního logaritmu, a jejich bezpečnost je srovnatelná s obtížností řešení těchto problémů. [h2]

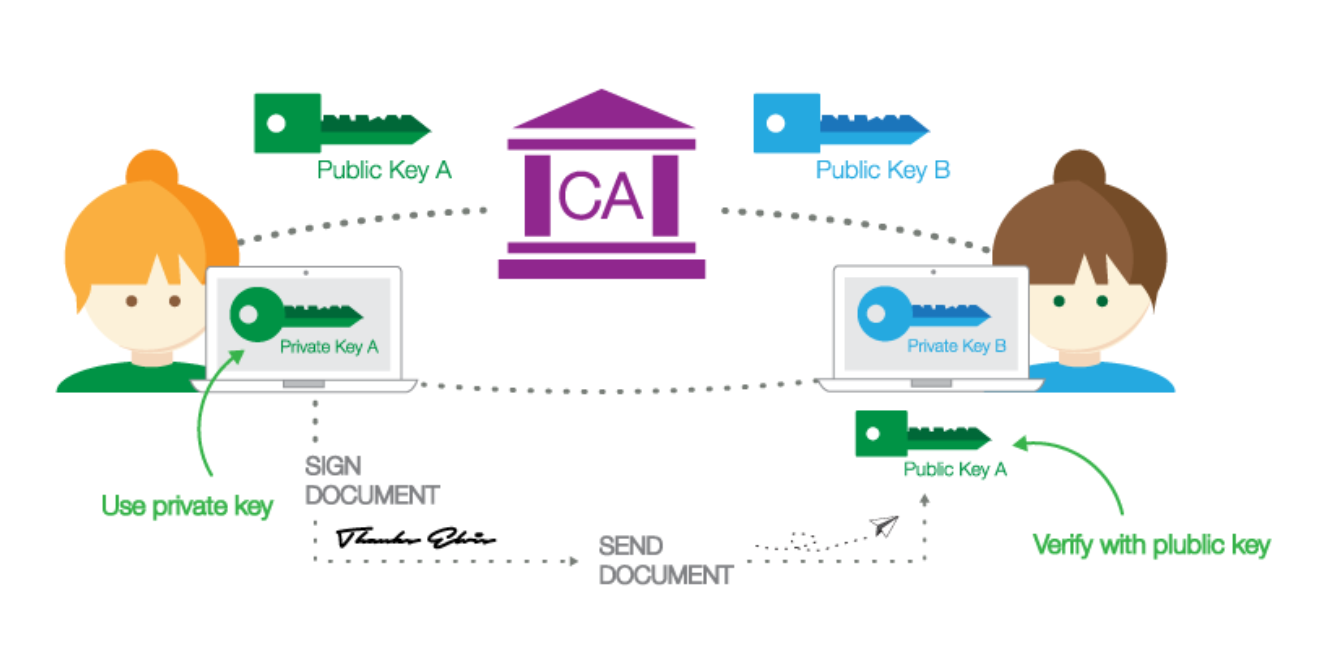
**Digitální podpis**

V oblasti digitální komunikace existuje další klíčová otázka. Jak může příjemce zprávy, jako je Bob, ověřit, že zpráva pochází opravdu od odesílatele, v tomto případě Alice, a že její obsah nebyl pozměněn? Tento problém je zvláště zásadní, když existuje riziko, že třetí strana, jako je Eva, může zprávu zachytit a potenciálně ji upravit. V digitálním světě se tento problém řeší pomocí technologie známé jako digitální podpisy.

Digitální podpis je pokročilá kryptografická metoda, která ověřuje pravost a nedotčenost digitálních dokumentů, zpráv nebo softwaru. Jedná se o digitální obdobu ručně psaného podpisu nebo razítka, ale poskytuje mnohem vyšší úroveň zabezpečení. Cílem digitálního podpisu je zabránit padělání a zosobnění dat v digitální komunikaci. Digitální podpisy slouží jako důkaz původu, identity a stavu elektronických dokumentů, transakcí nebo digitálních zpráv. [h3]

Digitální podpisy jsou založeny na šifrování pomocí asymetrické kryptografie. Pro šifrování dat, souvisejících s podpisem, používá osoba vytvářející digitální podpis (Alice) soukromý klíč. Jediný způsob, jak tato data dešifrovat, je použití veřejného klíče signatáře. Pokud příjemce (Bob) nemůže dokument otevřít veřejným klíčem signatáře, naznačuje to problém s dokumentem nebo podpisem. Takto jsou digitální podpisy ověřovány. [h3] Oproti šifrování dat, se používá opačná posloupnost klíčů. Při klasickém šifrování Alice šifruje zprávu veřejným klíčem, ale při digitálním podpisu zprávu podepisuje soukromým klíčem. Při podpisu nejde o utajení zprávy, ale o ověření její pravosti.

K ověření, že veřejný klíč patří vydavateli (Alici) slouží digitální certifikáty, známé také jako certifikáty veřejného klíče. Digitální certifikáty obsahují veřejný klíč, informace o jeho majiteli, data expirace a digitální podpis vydavatele certifikátu. Digitální certifikáty vydávají důvěryhodné třetí strany, certifikační autority (CA). [h3]



Obrázek 4 – Digitální podpisy[s4]

[x2] <https://www.cryptoquantique.com/blog/post-quantum-cryptography/>

[s2] <https://medium.com/@qcgiitr/post-quantum-cryptography-214b4e4b269a>

[s3]<https://cdn.venafi.com/994513b8-133f-0003-9fb3-9cbe4b61ffeb/41af4733-4c90-44a9-ac83-d3d1c9dedac3/asymmetric-encryption.png?fm=webp&q=85>

[s4] <https://www.gdpicture.com/blog/safer-signing-process/>

[s5]https://miro.medium.com/v2/resize:fit:1400/format:webp/0\*-q0OB65GCekDIm-6.png

## Shorův algoritmus

Shorův kvantový faktorizační algoritmus je revoluční metoda pro rozklad velkých čísel na prvočísla pomocí kvantového počítání. Vytvořený v roce 1994 Peterem Shorem, algoritmus využívá kvantové superpozice a provázání k odhalení periody funkce, což je zásadní pro faktorizaci. Základem je kvantová Fourierova transformace, která umožňuje kvantovému počítači zpracovat všechny možné vstupy najednou. Algoritmus začíná výběrem náhodného čísla a jeho zpracováním pomocí modulární aritmetiky k nalezení periody. Tato perioda je pak klíčem k faktorizaci, což má důležité implikace pro klasickou asymetrickou kryptografii, jako je RSA, DSA, ECDH a ECDSA. Shorův algoritmus, v případě existence funkční kvantových počítačů, silně ovlivní bezpečnost současné digitální komunikace. [h6] [h4]

## Motivace postkvantové kryptografie

Postkvantová kryptografie je oblastí, která se věnuje vývoji šifrovacích systémů odolných proti útokům pomocí kvantových počítačů. Tyto počítače jsou schopné provádět výpočty s využitím kvantových algoritmů, jako je Groverův a Shorův algoritmus, které mohou ohrozit současné šifrovací metody. Například Groverův algoritmus může zrychlit luštění symetrických šifer, jako jsou AES a DES, což by mohlo být vyřešeno zvětšením délky klíčů. Vzhledem k tomu, že se předpokládá, že kvantové počítače budou běžně dostupné v horizontu 10 až 20 let, je důležité již nyní přemýšlet nad využitím nových postkvantových algoritmů, které jsou schopné odolat potenciálním kvantovým útokům. To je zásadní zejména pro data, která potřebujeme chránit dlouhodobě, neboť i když se zdá, že 10 až 20 let je dlouhá doba, některá data, obzvlášť ve vojenském prostředí, musí být zabezpečena na mnohem delší období. [h4]

## NIST a jeho zapojení do pokroku v postkvantové kryptografii

Národní institut pro standardy a technologie, známý pod zkratkou NIST, je prestižní federální agentura, která byla založena na počátku 20. století a od té doby hraje klíčovou roli ve vědeckém a průmyslovém pokroku Spojených států. Jeho založení bylo reakcí na potřebu zlepšit měřicí infrastrukturu v USA, která byla v té době považována za nedostatečnou ve srovnání s jinými průmyslovými mocnostmi, jako bylo Spojené království nebo Německo. Díky své práci v oblasti vývoje a normalizace měřicích metod a technologií se NIST stal základním kamenem pro inovace a udržitelný růst v mnoha odvětvích. [h5]

NIST se podílí na široké škále výzkumných a vývojových projektů, od základních fyzikálních studií až po pokročilé aplikace v nanotechnologiích a kybernetické bezpečnosti. Jeho práce zahrnuje vývoj atomových hodin, které jsou základem pro přesné měření času a synchronizaci globálních komunikačních sítí, a také vývoj nových materiálů s výjimečnými vlastnostmi, které nacházejí uplatnění v nejnovějších technologických zařízeních. NIST také hraje klíčovou roli v oblasti kybernetické bezpečnosti, kde vyvíjí a propaguje rámce a standardy pro ochranu kritické infrastruktury a soukromých dat. [h5]

V současné době NIST nadále rozšiřuje své působení a poskytuje podporu pro nejnovější technologické inovace. Jeho měření a standardy jsou nezbytné pro vývoj a provoz zařízení v nanoměřítku, která jsou tak malá, že desítky tisíc z nich mohou být umístěny na konci jediného lidského vlasu. NIST také přispívá k bezpečnosti velkých staveb, jako jsou mrakodrapy navržené tak, aby odolaly zemětřesením, a k rozvoji globálních komunikačních sítí, které jsou základem moderní společnosti. [h5]

NIST je tedy nejen svědkem, ale i tvůrcem historie v oblasti vědy a technologie. Jeho odkaz a pokračující práce jsou zásadní pro pokrok a bezpečnost po celém světě. S jeho pomocí se podařilo překonat mnohé výzvy a otevřít dveře k novým možnostem v různých průmyslových a vědeckých oborech. [h5]

V roce 2017 NIST zahájil proces výběru, hodnocení a standardizace jednoho nebo více algoritmů pro kvantově odolné veřejné klíčové kryptografie. Tyto algoritmy mají nahradit stávající, které jsou zranitelné vůči útokům kvantových počítačů. Nové standardy měli zahrnovat algoritmy pro digitální podpisy, šifrování a vytváření klíčů, které budou dostupné celosvětově a ochrání citlivé informace vlády i po vzniku kvantových počítačů. Jako první krok NIST vyžádal veřejné komentáře k návrhům minimálních požadavků na přijatelnost, požadavků na předložení a kritérií pro hodnocení kandidátních algoritmů. [h5] Na základě toho zveřejnil 5 bezpečnostních kategorií podle odpovídající náročnosti útoku hrubou silou [h8]. Například pro algoritmy zařazené do 3. kategorie je odpovídající náročnost na útok hrubou silou odpovídající útok hrubou silou na AES-192.

|  |  |
| --- | --- |
| Úroveň | Náročnost |
| 1 | AES-128 |
| 2 | SHA256/ SHA3-256 |
| 3 | AES-192 |
| 4 | SHA384/ SHA3-384 |
| 5 | AES-256 |

Tabulka 1 - Bezpečnostní kategorie PQC - NIST

<https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf>

Nominace na post-kvantové kandidátní algoritmy mohly být podány až do konečného termínu 30. listopadu 2017 [h5]. Původně bylo předloženo 69 kandidátů, i když hrstka se rychle stáhla poté, co byly objeveny chyby, a v lednu 2019 NIST ukončil první kolo procesu a oznámil 26 algoritmů, které se dostaly do druhého kola. V červenci 2020, po téměř 18 měsících soustředění komunity na tato schémata, NIST oznámil své kandidáty do posledního kola: 4 mechanismy výměny klíčů a 3 schémata podpisů, se záměrem vybrat na konci procesu alespoň jeden z každého typu. [x2]

Na začátku července 2022 byly vybrány 4 algoritmy, které komunita uznala za kvantově bezpečné. Jedná se o CRYSTALS-Kyber pro výměnu klíčů, CRYSTALS-DILITHIUM, Falcon a SPHINCS+ pro digitální podpisy. [x6]

NIST se sice rozhodl začlenit do standardizace čtyři algoritmy, ale algoritmy CRYSTALS se vyznačují výjimečným postavením. Pro kategorii šifrování a klíčové výměny byl zvolen pro standardizaci pouze algoritmus CRYSTALS-Kyber. V oblasti post-kvantových digitálních podpisů byly vybrány tři algoritmy, avšak CRYSTALS-Dilithium je NISTem preferován jako hlavní volba [26]. NIST hodnotí oba tyto algoritmy jako vědecky dobře podložené, relativně jednoduché na implementaci a umožňující efektivní provádění kryptografických operací. [nukib]

[26] D. Moody: NIST PQC: LOOKING IN THE FUTURE, Selected presentations of the Fourth PQC Standardization Conference, NIST PQC: LOOKING INTO THE FUTURE https://csrc.nist.gov/csrc/media/Presentations/2022/nist-pqc-looking-into-the-future/images-media/session-1-moody-looking-into-future-pqc2022.pdf

[x2] https://www.cryptoquantique.com/blog/post-quantum-cryptography/

[x5] <https://groups.google.com/a/list.nist.gov/g/pqc-forum?_x_tr_sl&_x_tr_tl&_x_tr_hl&pli=1>

[x6] <https://csrc.nist.gov/csrc/media/Projects/post-quantum-cryptography/documents/selected-algos-2022/history-pqc-selected-algorithm-updates.pdf>

## Vybrané post-kvantové algoritmy pro standardizaci

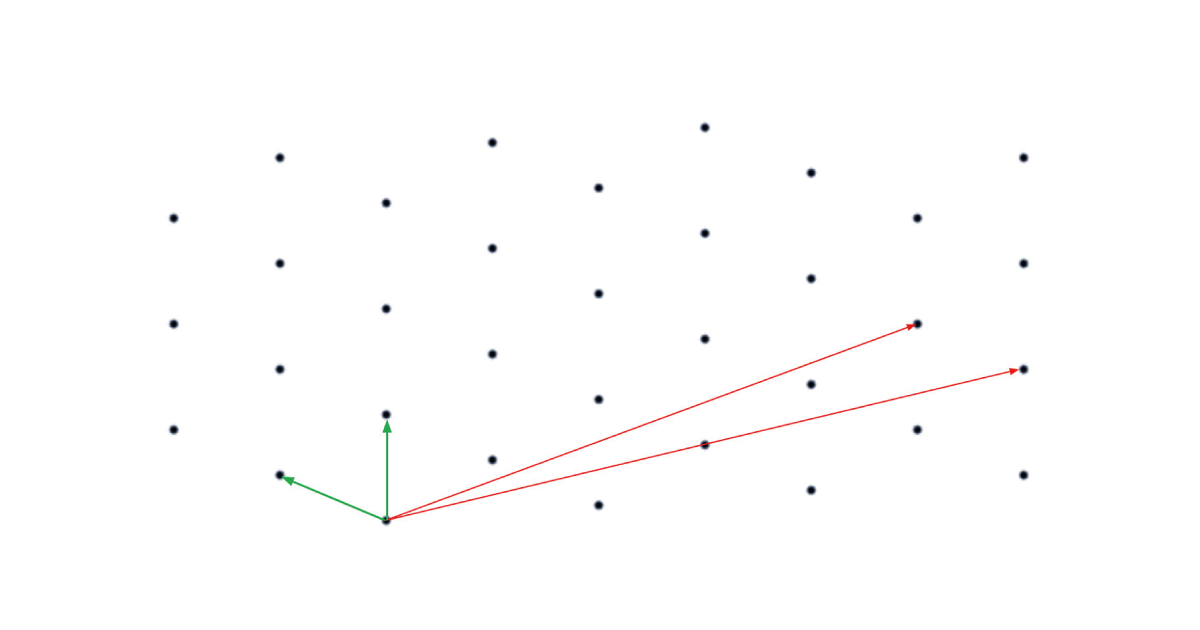
CRYSTALS Kyber, CRYSTALS Dilithium a Falcon patří do stejné rodiny kryptografických algoritmů založených na algebraických strukturách zvané mřížky, oproti tomu rodina algoritmů SPHINCS+ spoléhá na robustnost hashovací funkce. V této kapitole nahlédneme do základní podstaty zmíněných kryptografických oborů a poté popíšu vybrané post-kvantové algoritmy. V podkapitole 2.6 popíšu jakým způsobem jsem začlenil Python projekty vybraných algoritmů třetích stran do vlastních knihoven.

Kryptografie založená na mřížkách

Mřížky využívají dvou matematických problémů, které je dělají imunní vůči kvantovým počítačům. Jedná se o problém Shortest Vector Problem (SVP) a Closest Vector Problem (CVP).

Problematika SVP je spojena s hledáním nejkratšího vektoru báze. Lze si to představit na příkladu, kdy máme zadanou mřížku 𝐿, libovolnou bázi a cílem je najít nejkratší nenulový vektor, který do mřížky patří. Je třeba si uvědomit, že se hledá bod mřížky, který je nejblíže nule, ale není nulový.

Druhou problematikou je CVP, která je spojena s nalezením nejbližšího vektoru vůči libovolnému vektoru mřížky. Problematika se opírá o dvě báze vektorového prostoru. Jednoduchou a složitou. Jednoduchá obsahuje krátké a pokud možno na sebe kolmé vektory, které snadno popisují celou mřížku. Naopak složitá báze obsahuje velké vektory s podobným směrem. Popsat jimi celou mřížku, z definice báze samozřejmě jde, ale je to o dost náročnější. Ovšem nalézt nejbližší vektor mřížky vůči zadanému vektoru je podstatně náročnější než s využitím jednoduché báze. [x7] [x8] Problém postavený na SVP a CVP není možné efektivně překonat pomocí nám známým kvantových algoritmů, a proto lze na nich stavět postkvantové kryptografické algoritmy jako jsou CRYSTALS-Kyber, CRYSTALS-DILITHIUM a Falcon.



Obrázek 5 - Mříž se dvěma různými základnami [x2]

Obrázek 4 popisuje mřížku v nejjednodušším dvojrozměrném vektorovém prostoru, ovšem ta slouží pouze pro ilustraci problému, ale v realitě se využívají mřížky ve značně vyšším řádu rozměrů vektorového prostoru. [x2]

[x2] https://www.cryptoquantique.com/blog/post-quantum-cryptography/

[x7] POPELOVÁ, Lucie. Metody postkvantové kryptografie. Brno, 2018, 55 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Lukáš Malina, Ph.D.

[x8] https://www.youtube.com/watch?v=QDdOoYdb748

Kryptografie založená na hashovacích funkcích

Kryptografie založená na hashovacích funkcích používá jednorázová podpisová schémata jako základní stavební blok. Tyto schémata se liší od většiny jiných podpisových metod tím, že potřebují pouze bezpečnou kryptografickou hashovací funkci a nevyžadují jiné předpoklady o složitosti problémů z teorie čísel. Navíc nejsou náchylné k Shorově algoritmu. Zabezpečení těchto schémat se týká buď odolnosti vůči kolizím nebo pouze odolnosti proti druhému obrazu, což záleží na konkrétním použitém jednorázovém podpisu. [h7]

Jednorázová podpisová schémata zahrnují například Lamportovo schéma, Winternitzovo schéma a jeho varianta W-OTS+. V těchto schématech je soukromý klíč obvykle generován náhodně a veřejný klíč je funkcí soukromého klíče, zahrnující základní hashovací funkci. Pokročilá jednorázová podpisová schémata zahrnují parametry umožňující obchodování mezi časem a pamětí, jako je například Winternitzův parametr. Nicméně samotná jednorázová podpisová schémata nejsou praktická, protože každý soukromý klíč lze použít pouze k podpisu jediné zprávy. [h7]

Stavová schémata N-time podpisů, představená Merklem a dále zdokonalovaná, využívají jednorázová podpisová schémata a umožňují opakované použití jednorázových podpisů. Tato schémata kombinují N jednorázových podpisů do jedné struktury, obvykle úplného binárního stromu. Schéma N-time podpisu se skládá z algoritmů pro generování klíče, podpis a ověření podpisu. Soukromý klíč pro toto schéma se skládá z N soukromých klíčů základního jednorázového schématu. Každým generováním podpisu je použit další jednorázový soukromý klíč. Pro snížení velikosti soukromého klíče lze použít kryptograficky bezpečnou pseudonáhodnou funkci. [h7]

V kontextu kryptografie založené na haších je důležité zdůraznit, že schémata N-time podpisů představují významný pokrok ve snaze o zabezpečení digitálních komunikací. Tato schémata umožňují použití jednorázového podpisu vícekrát, aniž by došlo ke kompromitaci bezpečnosti, což je klíčové pro praktické aplikace. Využitím pseudonáhodné funkce pro generování soukromých klíčů z jednoho základního klíče se podstatně snižuje potřeba ukládat velké množství dat, což je zásadní pro zařízení s omezenou pamětí. [h7]

CRYSTALS-Kyber

https://medium.com/@hwupathum/using-crystals-kyber-kem-for-hybrid-encryption-with-java-0ab6c70d41fc

„Je KEM kvantově odolné schéma na bázi strukturovaných mřížek. Ke standardizaci byl vybrán pro svou bezpečnost a výkonnost. Jeho výkonnost na různých platformách je institucí NIST hodnocena jako excelentní. NIST předpokládá, že bude standardizovat varianty Kyber-768 (úroveň 3) a Kyber-1024 (úroveň 5). O standardizaci varianty Kyber-512 (úroveň 1) ještě není rozhodnuto, ale zatím se k ní NIST spíše přiklání. Pro jednotlivé bezpečnostní úrovně 1, 3 a 5 mají jeho veřejné klíče po řadě délky 800, 1184 a 1568 bajtů a jeho šifrové texty mají po řadě délky 768, 1088 a 1568 bajtů.“ [nukib]

Autoři Kyber pro 2. kolo projektu NIST navrhli variantu Kyberu (Kyber-90s) přizpůsobenou pro případ harwarové podpory symetrických primitiv [h9].

[nukib] https://nukib.gov.cz/download/uredni\_deska/Priloha%20-%20Minimalni%20pozadavky%20na%20kryptograficke%20algoritmy.pdf

CRYSTALS-DILITHIUM

„Je EUF-CMA-bezpečné post-kvantové podpisové schéma na bázi strukturovaných mřížek22 . Ke standardizaci bylo vybráno pro svou bezpečnost, vysokou výkonnost a poměrně jednoduché schéma návrhu. Institucí NIST je hodnoceno jako vysoce efektivní schéma se snadnou implementací a silnými bezpečnostními zárukami. NIST předpokládá, že bude standardizovat varianty[26], sl. 13: • Dilithium 2 (úroveň 2), • Dilithium 3 (úroveň 3), • Dilithium 5 (úroveň 5). Pro jednotlivé bezpečnostní úrovně 2, 3 a 5 mají jeho veřejné klíče po řadě délky 1312, 1952 a 2592 bajtů a jeho podpisy mají po řadě délky 2420, 3293 a 4595 bajtů.“ [nukib]

Falcon

Je EUF-CMA-bezpečné post-kvantové podpisové schéma na bázi strukturovaných mřížek23 . Jeho výhodou jsou malé délky klíčů a digitálních podpisů. Nevýhodou je jeho velmi složitý návrh, který znesnadňuje dobré porozumění detailům schématu a korektní implementaci. Právě krátkost klíčů a podpisů při dobrých zárukách bezpečnosti byla důvodem jeho výběru pro standardizaci. NIST předpokládá, že bude standardizovat varianty[26], sl. 14: • Falcon-512 (úroveň 1), • Falcon-1024 (úroveň 5). Pro bezpečnostní úrovně 1 a 5 mají jeho veřejné klíče po řadě délky 897 a 1793 bajtů a jeho podpisy mají po řadě délky 666 a 1280 bajtů. Jeho standard bude vydán až po standardu algoritmu Dilithium[26], sl. 14 . [nukib]

SPHINCS+

Je EUF-CMA-bezpečné post-kvantové podpisové schéma na bázi hašovacích funkcí. Jeho bezpečnost je založena na bezpečnosti použité hašovací funkce, v tomto případě buď SHAKE256, nebo SHA-256, nebo Haraka. Na rozdíl od schémat XMSS a LMS není v případě SPHINCS+ nutné, aby podepisující zařízení udržovalo informaci o podpisech vytvořených daným klíčem, a proto v jeho případě nevzniká omezení počtu podpisů týmž klíčem. To je ale do značné míry vyváženo enormně dlouhými digitálními podpisy. Tento podpisový algoritmus byl vybrán ke standardizaci, protože má velmi silné bezpečnostní garance a protože je zkonstruován na jiné bázi než na mřížkách. Budou standardizovány[26], sl. 15 jeho tzv. „jednoduché verze“ (nikoliv tzv. „robustní verze“) pro bezpečnostní úrovně 1, 3 a 5, a to s následujícími hašovacími funkcemi schválenými institucí NIST pro konstrukci SPHINCS+: SHAKE a SHA-2. Přitom pod SHA-2 zde NIST rozumí: • SHA-256 pro úroveň 1, • Mix funkcí SHA-256 a SHA-512 pro úrovně 3 a 5[nukib]

Rodina SPHINCS+ ma nekolik algoritmu, ktere se lisi pouzitou hashovaci funkci. haraka\_128f haraka\_192f haraka\_256f sha2\_128f sha2\_192f sha2\_256f shake\_128f shake\_192f shake\_256f haraka\_128s haraka\_192s haraka\_256s sha2\_128s sha2\_192s sha2\_256s shake\_128s shake\_192s shake\_256s

Haraka\_xxxf (xxx je 128, 192 nebo 256): Tyto verze používají hashovací funkci Haraka s různými délkami výstupu (128, 192 nebo 256 bitů). Haraka je konstrukce kryptografické hashovací funkce, která byla navržena pro rychlost a bezpečnost.

SHA2\_xxxf (xxx je 128, 192 nebo 256): Tyto verze používají standardní hashovací funkce SHA-2 (Secure Hash Algorithm 2) s různými délkami výstupu (128, 192 nebo 256 bitů).

SHAKE\_xxxf (xxx je 128, 192 nebo 256): Tyto verze používají SHA-3 variantu SHAKE (Secure Hash Algorithm KECCAK) s různými délkami výstupu (128, 192 nebo 256 bitů). SHAKE umožňuje generovat libovolně dlouhé hashovací hodnoty.

Haraka\_xxxs, SHA2\_xxxs, SHAKE\_xxxs (xxx je 128, 192 nebo 256): Tyto verze jsou stejné jako jejich odpovídající "f" varianty, ale s optimalizacemi pro snížení velikosti veřejných klíčů a podpisů. To znamená, že generované klíče a podpisy jsou kratší, což může být výhodné v prostředích s omezenými zdroji.

# Mikroslužba zabezpečující postkvantovou kryptografii

V předešlé kapitole jsem psal o postkvantové kryptografii. Vysvětlil jsem, proč je důležitá a jaké algoritmy lze použít. V této kapitole je uveden seznam dostupných implementací vybraných postkvantových algoritmů a popis dvou použitelných knihoven pro práci s nimi, které jsem vytvořil. Knihovny jsem následně použil v praxi. Vytvořil jsem mikroslužbu PQC-service (Post Quantum Cryptography service), která zabezpečuje odesílání zpráv zašifrovaných a podepsaných pomocí postkvantové kryptografie mezi dvěma entitami. Entitou, pro tento případ, rozumím počítač/software, který může existovat nezávisle a který lze jednoznačně identifikovat.

Pro podepisování je použita datová struktura JWS (JSON Web Signature), která je jedna z možností jak se zprávou a jejím podpisem nakládat. JWS je definována ve standardu RFC 7515 a skládá se ze dvou datových struktur JSON: hlavičky a těla a poté podpisu. Hlavička by měla obsahovat o jaký typ zprávy se jedná a jaký kryptografický algoritmus byl použit pro podepsání zprávy a obsah. Tělo obsahuje samotnou zprávu a případně dalších parametry, popisující vlastnosti zprávy. Například pokud je zprávou přístupový token, tak dalším parametrem může být datum expirace tohoto tokenu. Podepisuje se zakódovaná hlavička a zakódované tělo oddělené tečkou. Pro kódování se používá Base64. Samotný podpis se následně také kóduje a přikládá se ke zbytku opět tečkou. Výsledkem je tedy řetězec tří Base64 kódů oddělenými tečkami. [h10]

Sice jsem se snažil nalézt knihovnu pro práci s JWS, která by umožňovala přiložit vlastní podpisové algoritmy, ale takovou knihovnu jsem nenašel. Proto jsem si vytvořil knihovnu vlastní. Při jejím vývoji jsem bral do úvahy komptabilitu s již zavedenou a široce používanou knihovnou PyJWT [https://pyjwt.readthedocs.io/en/stable/].

## Implementace postkvantových algoritmů

V rámci své práce jsem se rozhodl využít více implementací vybraných postkvantových algoritmů z různých zdrojů. Byl jsem sice limitován zvoleným programovacím jazykem, Pythonem, který pro kryptografii, není až tak obvyklý, ale přesto se mi podařilo najít minimálně dvě různé implementace pro každý z vybraných algoritmů.

Kladl jsem si za cíl vytvořit dvě Python knihovny, které budou obsahovat všechny mnou použité implementace. Chtěl jsem, aby se uživatel, bez nutnosti úpravy kódu, mohl rozhodnout, který algoritmus, jeho verzi a také implementaci bude chtít použít.

Společné knihovny

Vytvořil jsem dva soubory „kemAlgLib.py“ a „signAlgLib.py“. Jeden pro algoritmy výměny klíčů a druhý pro algoritmy digitálních podpisů. Každý ze souborů obsahuje několik sjednocených tříd, kde jedna třída se rovná jedné implementaci jednoho algoritmu. Na obrázku 5 a 6 ukazuji příklad třídy pro algoritmus výměny klíčů a pro algoritmus digitálního podpisu. Těla funkcí jsou skryta.

Obsah obrázku text, snímek obrazovky, Písmo, řada/pruh

Popis byl vytvořen automaticky

Obrázek 6 – Třída pro KEM algoritmus

Obsah obrázku text, snímek obrazovky, Písmo, řada/pruh

Popis byl vytvořen automaticky

Obrázek 7 – Třída pro podpisový algoritmus

Ona sjednocenost tříd je zajištěna názvy vnitřních funkcí, které jsou totožné. Také jejich počet, názvy a typy parametrů jsou pro všechny algoritmy stejné. Jednotlivé třídy se od se liší v tělech vnitřních funkcí, protože jsou napsána tak, aby mohla obsluhovat svoji přidělenou implementaci postkvantového algoritmu.

Iniciační funkce načítá svoji přidělenou implementaci. Z konfiguračního souboru si bere ještě dodatečnou informaci o verzi algoritmu, kterou má použít. Pro jednotlivé třídy se možnosti liší, ale například pro třídu KYBER\_PY algoritmu CRYSTALS-Kyber se jedná o volbu mezi verzemi Kyber\_512, Kyber\_768 a Kyber\_1024 viz. kapitola 2.5.3.

Funkce „generate\_keypair“ ze svého názvu napovídá, že obsluhuje generování páru soukromého a veřejného klíče. Ten vrací ve dvou řetězcích. Každý klíč má své záhlaví a zápatí.

"-----BEGIN PUBLIC KEY-----\n" + veřejný klíč + "\n-----END PUBLIC KEY-----"

"-----BEGIN SECRET KEY-----\n" + soukromý klíč + "\n-----END SECRET KEY-----"

Ač se funkce stejně jmenují a vrací stejné hodnoty, klíče nemohou být zaměňovány. Není možné používat klíče algoritmu CRYSTALS-Kyber pro podepisování zpráv, nebo naopak klíče od například algoritmu Falcon pro výměnu klíčů.

Ve třídách algoritmů pro výměnu klíčů se funkce „encrypt“ stará o vytvoření dvojici symetrického klíče a zašifrovaného textu na základě přijatého veřejného klíče. Tu vrací ve dvou řetězcích se záhlavím a zápatím.

"-----BEGIN CIPHER TEXT-----\n" + zašifrovaný text + "\n-----END CIPHER TEXT-----"

"-----BEGIN SYMMETRIC KEY-----\n" + symetrický klíč + "\n-----END SYMMETRIC KEY-----"

Naopak funkce „decrypt“ přijímá jako parametr soukromý klíč a zašifrovaný text. Jejím výstupem je symetrický klíč totožný s klíčem z funkce „encrypt“.

"-----BEGIN SYMMETRIC KEY-----\n" + symetrický klíč + "\n-----END SYMMETRIC KEY-----"

Ve třídách podpisových algoritmů se funkce „sign“ zabývá podepisováním zpráv. Přijímá jako parametry soukromý klíč a zprávu, kterou chce uživatel podepsat. Na oplátku vrací podpis ve tvaru:

"-----BEGIN SIGNATURE-----\n" + podpis + "\n-----END SIGNATURE-----"

Funkce „verify“ ověřuje podpis na základě veřejného klíče, zprávy a podpisu. Vrací buď chybovou hlášku, nebo hodnotu pravda.

V příloze a uvádím kopii Jypyter notebooku, který ukazuje obsluhu všech algoritmů v knihovnách „kemAlgLib.py“ a „signAlgLib.py“

Pqcrypto

Balíček pqcrypto je volně dostupnou sadou post-kvantových algoritmů dostupnou na <https://github.com/kpdemetriou/pqcrypto>. Umožňuje instalaci i jako python knihovnu pomocí příkazu „pip install pqcrypto“. Nicméně instalace, i když nezobrazí žádnou chybovou hlášku, selže. V adresáři knihoven prostředí Python se vytvoří prázdná složka „pqcrypto“, která neobsahuje zdrojové kódy pro práci s post-kvantovými algoritmy. Tuto chybu lze obejít. Postačí si stáhnout git repozitář z uvedeného odkazu výše a v hlavní adresáři repozitáře spustit příkaz „python compile.py“. Ten bude několik minut kompilovat přiložené zdrojové kódy programovacího jazyka C. Po skončení vzniknou dva adresáře „\_kem“ a „\_sign“, které obsahují python knihovny pro práci s post-kvantovými algoritmy. Tyto dva adresáře, které byly zkompilovány mimo balíček Python knihoven je potřeba vzít a vložit je adresáře „pqcrypto“ vytvořeného dříve příkazem „pip install pqcrypto“.

Balíček poskytuje všechny vybrané post-kvantové algoritmy a několik dalších, které nepostoupili do posledního kola. CRYSTALS-Kyber má v tomto balíčku šest verzí (Kyber512, Kyber512\_90s, Kyber768, Kyber768\_90s, Kyber1024, Kyber1024\_90s). CRYSTALS-DILITHIUM má verze tři (dilithium2, dilithium3, dilithium4). Falcon má verze dvě (falcon512 a falcon1024). Největší počet verzí má rodina algoritmů SPHINCS+, která má verzí třicet šest (sphincs\_haraka\_128f\_robust, sphincs\_haraka\_128f\_simple, sphincs\_haraka\_128s\_robust, sphincs\_haraka\_128s\_simple, sphincs\_haraka\_192f\_robust, sphincs\_haraka\_192f\_simple, sphincs\_haraka\_192s\_robust, sphincs\_haraka\_192s\_simple, sphincs\_haraka\_256f\_robust, sphincs\_haraka\_256f\_simple, sphincs\_haraka\_256s\_robust, sphincs\_haraka\_256s\_simple, sphincs\_sha256\_128f\_robust, sphincs\_sha256\_128f\_simple, sphincs\_sha256\_128s\_robust, sphincs\_sha256\_128s\_simple, sphincs\_sha256\_192f\_robust, sphincs\_sha256\_192f\_simple, sphincs\_sha256\_192s\_robust, sphincs\_sha256\_192s\_simple, sphincs\_sha256\_256f\_robust, sphincs\_sha256\_256f\_simple, sphincs\_sha256\_256s\_robust, sphincs\_sha256\_256s\_simple, sphincs\_shake256\_128f\_robust, sphincs\_shake256\_128f\_simple, sphincs\_shake256\_128s\_robust, sphincs\_shake256\_128s\_simple, sphincs\_shake256\_192f\_robust, sphincs\_shake256\_192f\_simple, sphincs\_shake256\_192s\_robust, sphincs\_shake256\_192s\_simple, sphincs\_shake256\_256f\_robust, sphincs\_shake256\_256f\_simple, sphincs\_shake256\_256s\_robust, sphincs\_shake256\_256s\_simple).

Kromě vybraných post-kvantových algoritmů, balíček poskytuje také algoritmy FireSaber, FrodoKEM, McEliece, NTRUEncrypt, Saber a Rainbow, ale ty jsem nepoužil, protože tyto algoritmy zatím nebyly vybrány.

KYBER\_PY

Github repozitář od uživatele GiacomoPope jsem získal z <https://github.com/GiacomoPope/kyber-py>. Autor čerpal z autorské dokumentace algoritmu KRYSTALS-Kyber a podle instrukcí převedl matematické rovnice do spustitelného kódu jazyka Python. Repozitář obsahuje přístupový soubor kyber.py jenž má v sobě tři třídy: Kyber512, Kyber768 a Kyber1024. Je tedy možné si vybrat mezi všemi základními verzemi algoritmy Kyber, ale neposkytuje modifikace pro hardwarovou podporu symetrických primitiv. Jejich rozdíly jsem uvedl v kapitole 2.5.3.

FALCON\_official

Autoři podpisového algoritmu Falcon na svém webu poskytli implementaci svého algoritmu pro jazyk Python [x10]. Jejich implementace formou github repozitáře obsahuje obě verze Falconu (Falcon-512, Falcon-1024). Přístupovým souborem pro práci s algoritmem je soubor „falcon.py“. Ten v sobě, mimo jiné, má dvě třídy: PublicKey a SecretKey, které uživatel volá pro získání klíčů. PublicKey používá při své iniciaci třídu SecretKey jako parametr. Jinými slovy, v tomto případě se nejedná o klasické vygenerované klíče, ale přímo o třídy, které se používají jako klíče. Třída SecretKey v sobě obsahuje metodu pro podepisování zpráv a třída PublicKey vlastní naopak metodu pro ověření podpisu.

[x10] https://falcon-sign.info/

<https://github.com/tprest/falcon.py>

SPHINCSPlus\_Tottifi

## PQC-service

Tato mikroslužba využívá pro šifrování a podepisování postkvantové knihovny „kemAlgLib.py“ a „signAlgLib.py“, která byly specifikovány v předešlé kapitole. Mikroslužba se skládá ze tří entit:

* Alice – entita, která odesílá Bobovi podepsanou a zašifrovanou zprávu a čeká na odpověď, kterou posléze dešifruje, ověří její podpis a vytiskne na monitor.
* Bob – entita, který od Alice přijímá zprávu, dešifruje ji, ověří její podpis, zprávu vytiskne na monitor a následně odešle zpět podepsanou a zašifrovanou odpověď.
* Certifikační autorita poskytuje Alici a Bobovi klíče pro podepisování a ověření podpisů.

V následující kapitolách jsou podrobně popsány všechny funkcionality, které jsou nezbytné k tomu, aby Alice mohla odeslat postkvantově zašifrovanou a podepsanou zprávu Bobovi.

Nastavení symetrického klíče

První věc, kterou jakákoliv entita musí provést, pokud chce komunikovat s někým dalším je stanovit si bezpečný (zašifrovaný) komunikační kanál. Pro tento účel mají Alice, Bob i CA v modulu communication.py asynchronní funkci:



Obrázek 8 – Funkce pro výměnu symetrického klíče

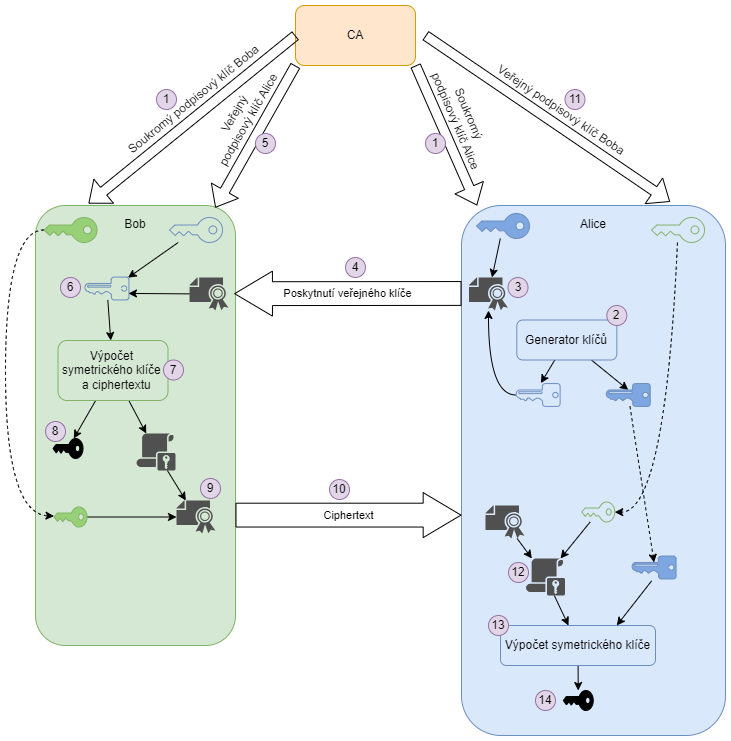
Tato funkce naváže spojení s druhou stranou a pomocí KEM algoritmu si vzájemně vymění informace takovým způsobem, aby výsledkem byl symetrický klíč, kterým následně mohou otevřít nové zašifrované spojení. Symetrický klíč jako takový nebyl nikdy poslán po síti, a tedy nikdo, kromě zúčastněných stran, o klíči nemůže vědět, a proto je komunikace bezpečná. Podrobný popis komunikace uvádím na obrázcích 10 a11. Oba ukazují výměnu symetrického klíče, ale liší se v tom, s jakými informacemi strany disponují. Na prvním obrázku Alice nevlastní svůj soukromý podpisový klíč, a tedy nemá, jak svoji zprávu podepsat. Taková situace je akceptována pouze mezi Alicí/Bobem a CA, a to pouze tehdy pokud se jedná o první navázání spojení za účelem registrace Alice/Boba u CA. Druhý obrázek popisuje standardní výměnu symetrického klíče pro už zaregistrované entity u CA.

Obsah obrázku text, diagram, snímek obrazovky

Popis byl vytvořen automaticky

Obrázek 9 – Popis výměny symetrického klíče pro první spojení mezi Alicí a CA

1. Uživatel vloží do zdrojového adresáře veřejný klíč CA ve tvaru „CA\_public\_key.pem“. To udělá ještě před spuštěním Alice. Pokud je CA umístěna ve stejném adresáři jako Alice, uživatel nemusí ručně nic vkládat.
2. Alice vygeneruje dvojici soukromého a veřejného KEM klíče.
3. Alice svůj KEM veřejný klíč, v plaintextové podobě, odešle CA.
4. CA s využitím veřejného klíče Alice vygeneruje ciphertext a symetrický klíč.
5. CA v tuto chvíli vlastní symetrický klíč, kterým bude pozdější komunikace šifrována.
6. CA vezme vygenerovaný ciphertext a podepíše ho svým soukromým podpisovým klíčem.
7. CA odesílá Alici podepsaný ciphertext.
8. Alice ověří podpis pomocí veřejného podpisového klíče CA a získá tak ciphertext.
9. Alice vygeneruje stejný symetrický klíč, jako má CA, díky znalosti ciphertextu a svého soukromého KEM klíče vygenerovaného v kroku 2.
10. Alice vlastní symetrický klíč, kterým bude šifrována pozdější komunikace.



Obrázek 10 - Popis výměny symetrického klíče mezi Alicí a Bobem

1. Alice a Bob získají od CA svůj soukromý podpisový klíč (bude vysvětleno v 3.2).
2. Alice vygeneruje dvojici soukromého a veřejného KEM klíče.
3. Alice podepíše vygenerovaný veřejný klíč svým soukromým podpisovým klíčem.
4. Alice odešle Bobovi svůj podepsaný veřejný klíč.
5. Bob získá od CA veřejný podpisový klíč Alice (bude vysvětleno v 3.2)
6. Bob ověří podpis a získá tak veřejný KEM klíč Alice.
7. Bob, s pomocí veřejného KEM klíče Alice, vygeneruje ciphertext a symetrický klíč.
8. Bob v tuto chvíli vlastní symetrický klíč, kterým bude pozdější komunikace šifrována.
9. Bob vezme vygenerovaný ciphertext a podepíše ho svým soukromým podpisovým klíčem.
10. Bob odesílá Alici podepsaný ciphertext.
11. Alice získá od CA veřejný podpisový klíč Boba (bude vysvětleno v 3.2).
12. Alice ověří podpis pomocí veřejného podpisového klíče Boba a získá tak ciphertext.
13. Alice vygeneruje stejný symetrický klíč, jako má Bob, díky znalosti ciphertextu a svého soukromého KEM klíče vygenerovaného v kroku 2.
14. Alice vlastní symetrický klíč, kterým bude šifrována pozdější komunikace.

Získání podpisových klíčů a práce s nimi

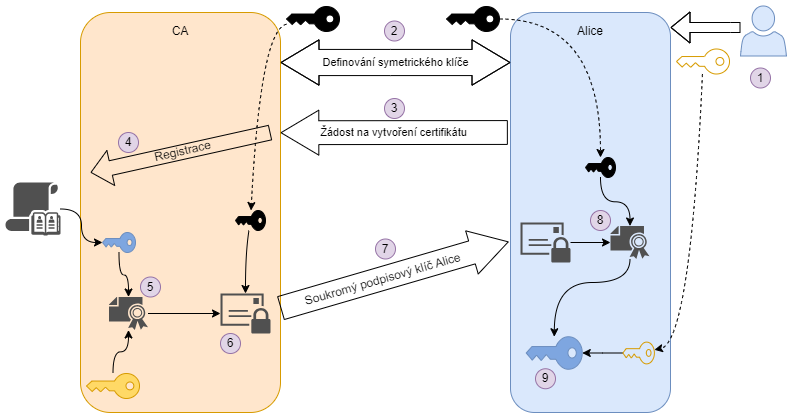
Teorii o podpisových algoritmech a klíčích jsem uvedl v kapitole 2.2 této práce a seznam dostupných algoritmů a jejich činnosti v kapitole 2.6. Pro tuto chvíli zobecním činnost post-kvantových kryptografických podpisových algoritmů na tři základní funkce: generování klíčů, výpočet symetrického klíče a šifrovaného textu a výpočet symetrického klíče.

V této podkapitole popíšu, jakým způsobem entity získávají soukromé a veřejné podpisové klíče.

**Získání soukromého podepisovacího klíče**

Každá entita, pokud chce komunikovat v síti musí vlastnit svůj soukromý podpisový klíč. Bez něj nedokáže podepsat zprávu, kterou by chtěla odeslat a druhá strana by neměla, jak ověřit, kdo je skutečným odesílatelem. V mé mikroslužbě PQC-service je získání soukromého podpisového klíče prováděno hned při startu entity (Alice/Bob). Po načtení konfiguračního souboru je zavolána asynchronní funkce „get\_sign\_private\_key“, která má tři parametry:

* my\_address – informace o adrese a portu na kterém běží entita
* ca – informace o adrese a portu certifikační autority (získáno z konf. souboru)
* algorithm – algoritmy, které bude pro komunikaci entita využívat(z konf. souboru)

Výsledkem je podpisový soukromý klíč na straně entity, která o klíč žádala a uložený certifikát entity na straně CA.

Obrázek 11 - Popis žádosti o soukromý podpisový klíč

1. Uživatel vloží do zdrojového adresáře veřejný klíč CA ve tvaru „CA\_public\_key.pem“. To udělá ještě před spuštěním Alice. Pokud je CA umístěna ve stejném adresáři jako Alice, uživatel nemusí ručně nic vkládat.
2. Alice a CA si vymění symetrické klíče. Popsáno v kapitole 6.2.
3. Alice posílá žádost CA o vytvoření certifikátu.
4. CA vydá Alici certifikát.
5. CA vezme soukromý podpisový klíč Alice a podepíše ho svým soukromým podpisovým klíčem.
6. CA vezme podepsaný klíč Alice a zašifruje ho symetrickým klíčem.
7. CA odešle zašifrovaný podepsaný klíč Alici.
8. Alice dešifruje zprávu pomocí symetrického klíče.
9. Alice ověří podpis pomocí veřejného podpisového klíče CA, získá tak svůj soukromý podpisový klíč.

**Získání veřejného podpisového klíče**

Další důležitá funkcionalita, která umožňuje všem entitám zeptat se CA na veřejný klíč jiné požadované entity. Například pokud Alice odešle podepsanou zprávu Bobovi, tak se Bob dotáže CA na veřejný klíč Alice. Tímto klíčem je následně schopný ověřit, zda zprávu skutečně odeslala Alice. Asynchronní funkce, kterou by Bob zavolal je „ask\_public\_key“.



Obrázek 12 – Funkce pro získání veřejného podpisové klíče

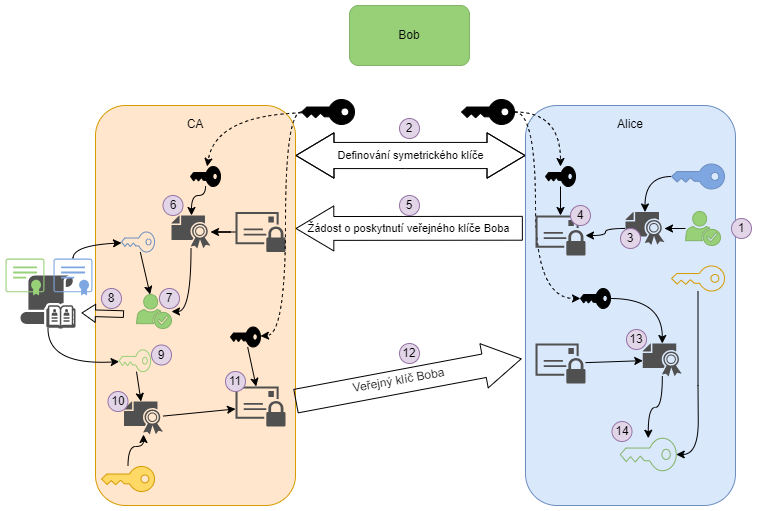
Tato funkce má 5 parametrů:

* subjekt – identifikátor, pod kterým je entita uložena u CA.

V mé práci se jedná o formát „ip adresa:port“.

* sign\_private\_key – soukromý podpisový klíč entity, která se dotazuje CA.
* my\_address – adresa entity
* ca – informace o CA
* algorithm – informace o zvolených algoritmech

Následující obrázek uvádí podrobný popis komunikace a procesů nutných k tomu, aby Alice mohla získat od CA veřejný podpisový klíč Boba.



Obrázek 13 - Popis žádosti o veřejný podpisový klíč

1. Vstupní informace pro Alici jsou: vlastní soukromý podpisový klíč a veřejný podpisový klíč CA. Alice také potřebuje znát identifikátor Boba. Ten získá například z hlavičky přijaté zprávy od Boba.
2. Alice a CA si vymění symetrické klíče pro šifrování. Popsáno v kapitole 6.2
3. Alice podepíše identifikátor Boba svým soukromým podpisovým klíčem.
4. Alice zašifruje zprávu pomocí symetrického klíče.
5. Alice odešle zašifrovanou zprávu CA
6. CA pomocí symetrického klíče dešifruje zprávu.
7. CA pomocí veřejného podpisového klíče ověří podpis. (klíč má uložen ve své databázi)
8. CA prověří, zda má Boba uloženého ve své databázi.
9. CA vezme veřejný podpisový klíč Boba.
10. CA podepíše veřejný podpisový klíč Boba svým soukromým podpisovým klíčem.
11. CA zašifruje zprávu pomocí symetrického klíče.
12. CA odešle zašifrovanou zprávu Alici.
13. Alice pomocí symetrického klíče dešifruje zprávu.
14. Alice veřejným podpisovým klíčem CA ověří podepsanou zprávu a získá tak veřejný podpisový klíč Boba.

Průběh komunikace při odeslání zprávy

Po objasnění všech dílčích kroků se můžeme konečně přesunout k té nejkomplexnější funkci, a to je samotné odeslání zprávy. Zahrnuje všechny předchozí typy komunikací a přidává k nim ještě tu samotnou zprávu. Asynchronní funkce, která odeslání zprávy umožňuje se nazývá „send\_request“ a je definovaná v tomto tvaru:



Obrázek 14 – Funkce pro odeslání zprávy

Její parametry:

* ip\_address – adresa adresáta
* port – port adresáta
* payload – obsah zprávy
* sign\_private\_key – soukromý podpisový klíč entity, která tuto funkci zavolala
* my\_address – informace o entitě, která tuto funkci zavolala
* ca – informace o CA
* algorithm – informace o zvolených algoritmech
* uri – přímá specifikace koncového bodu, kam má být zpráva odeslána

Na následujícím, prozatím nejsložitějším, obrázku 14 se pokusím objasnit dílčí kroky, které zajistí správné a bezpečné doručení zprávy od Alice k Bobovi a její odpovědi od Boba zpět Alici. Obsah obrázku diagram, mapa, text, Plán

Popis byl vytvořen automaticky

Obrázek 15 – Schéma odeslání zprávy

1. Alice a Bob se zaregistrují u CA. Získají tak své soukromé podpisové klíče.
2. Alice (uživatel) napíše zprávu.
3. Alice zprávu podepíše svým soukromým podpisovým klíčem.
4. Alice a Bob si vymění symetrické klíče pro šifrování.
5. Alice zašifruje zprávu symetrickým klíčem.
6. Alice odešle zprávu Bobovi.
7. Bob dešifruje zprávu pomocí symetrického klíče.
8. Bob si vyžádá od CA veřejný podpisový klíč Alice.
9. Bob ověří podpis na zprávě od Alice.
10. Bob vyhodnotí samotný obsah zprávy a napíše odpověď.
11. Bob podepíše odpověď svým soukromým podpisovým klíčem.
12. Bob zašifruje podepsanou odpověď symetrickým klíčem.
13. Bob odešle odpověď.
14. Alice dešifruje odpověď symetrickým klíčem.
15. Alice si vyžádá veřejný podpisový klíč Boba.
16. Alice ověří podpis na odpovědi.
17. Alice získala odpověď na svoji zprávu.

Průvodce pro spuštění mikroslužby

V níže uvedených krocích je uveden postup, jak jednoduše spustit mikroslužbu PQC-service na vlastním stroji. Předpokladem pro správné fungování je nainstalovaný Python verze 3.12.2 dostupný z <https://www.python.org/downloads>, nástroj Git dostupný z <https://git-scm.com/> a kompilátor programovacího jazyka C: Microsoft Visual C++ 14.0 nebo novější, který je dostupný na <https://visualstudio.microsoft.com/visual-cpp-build-tools/>.

Zdrojový kód mikroslužby a instalační balíček je dostupný na GitHub repozitáři. Stáhnete jej pomocí Git příkazu: „git clone -b PQC-service <https://github.com/naxit-01/DiplomovaPrace.git>“.

Po stažení přejděte do složky „DiplomovaPrace“ a spusťte instalační soubor „setup.bat“. Ve vašem adresáři se vytvoří virtuální prostředí a do něj se vám nainstalují všechny potřebné knihovny. Instalace potrvá několik minut, a to z důvodu kompilace knihovny pro programovací jazyk C do Python knihovny. Na konci instalace se v adresáři vytvoří tři spustitelné soubory; „activate\_alice.bat“, „activate\_bob.bat“ a „activate\_CA.bat“. V této chvíli by mělo být vše připraveno pro spuštění všech entit.

Před spuštěním entit se můžete rozhodnout jakým algoritmem chcete šifrovat a kterým podepisovat. Jejich výběr provedete odstraněním poznámky u odpovídajícího řádku v konfiguračním souboru „config.ini“. Pro daný algoritmus se nabízí více variant, takže byste měli některou z nich vybrat. Pokud tak neuděláte, zvolí se výchozí algoritmus a výchozí verze.

Jako první spusťte soubor „activate\_CA.bat“. Nastartuje se certifikační autorita, která se musí spustit jako první, protože ostatní entity se na ní dotazují pro svůj soukromý podpisový klíč ihned po jejich spuštění. Po CA spusťte „activate\_bob.bat“. Nastartuje se Bob, který bude přijímat zprávy od Alice. Na závěr zapněte Alici souborem „activate\_alice.bat“. Alice začne okamžitě odesílat zprávy Bobovi se vteřinovým odstupem.

Pokud jste všechny tři entity nespustili ze stejného počítače, musíte upravit konfigurační soubor „config.ini“. V něm najdete oddíl „CA“, kde nastavíte IP adresu a port pro certifikační autoritu a oddíl „NODE“ pro informace o Bobovi.

# Přehled ochranných mechanismů pro zabezpečení dat v informačním systému s ohledem na možnost odhalení manipulace s daty

V tomto článku se podíváme na některé z nejčastějších ochranných mechanismů pro zabezpečení dat v informačním systému s ohledem na možnost odhalení manipulace s daty. Data jsou cenným aktivem každé organizace a jejich ztráta, únik nebo zneužití může mít vážné následky pro její provoz, pověst a finanční výsledky. Proto je důležité mít správně nastavené procesy a nástroje pro ochranu dat před různými hrozbami.

Jedním z nejzákladnějších ochranných mechanismů je zálohování dat. Zálohování dat znamená pravidelné kopírování dat z primárního úložiště na sekundární úložiště, které je fyzicky oddělené a chráněné před neoprávněným přístupem. Zálohování dat umožňuje obnovit data v případě jejich poškození, smazání nebo ztráty způsobené například hardwarovou poruchou, lidskou chybou, malwarem nebo přírodní katastrofou. Zálohovaná data by měla být pravidelně testována a verifikována, aby byla jistota jejich konzistence a dostupnosti.

Dalším důležitým ochranným mechanismem je archivace dat. Archivace dat znamená ukládání dat, která již nejsou aktivně používána, ale jsou stále potřebná pro historické, právní nebo regulační účely, do speciálního úložiště, které je optimalizované pro dlouhodobé uchování dat. Archivovaná data jsou obvykle neměnná, což znamená, že se nedají měnit ani mazat. Tím se snižuje riziko manipulace s daty nebo jejich zničení. Archivovaná data by měla být také šifrována a podepsána digitálním podpisem, aby byla zajištěna jejich důvěrnost a integrita.

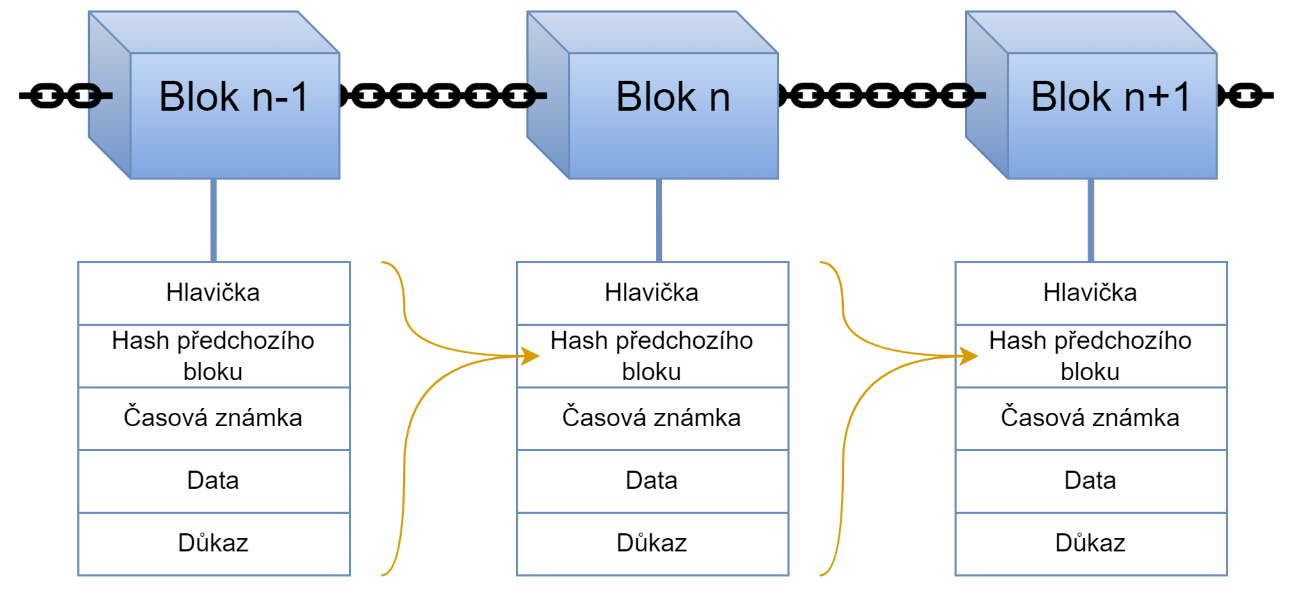
Kromě zálohování a archivace dat je také nutné zajistit bezpečnost dat v provozu, tedy v době, kdy jsou data ukládána, přenášena nebo zpracovávána v informačním systému. K tomu slouží různé techniky, jako je šifrování dat, autentizace a autorizace uživatelů a aplikací, auditování a monitorování aktivit v systému, detekce a prevence útoků, firewall a antivirový software. Tyto techniky pomáhají chránit data před neoprávněným přístupem, změnou nebo odcizením ze strany interních nebo externích aktérů.

# Blockchain

S nárustem zájmu o kryptoměny v posledních letech se do širšího povědomí dostal i pojem blockchain, databázová technologie, nad kterou je valná většina kryptoměn postavena. Její hlavní výhodou je její decentralizace. Na rozdíl od klasický centralizovaných databází neexistuje žádná autorita, která by ji spravovala nebo výhradně u sebe vlastnila její data. Nemůže se tedy stát, že by jedna osoba/instituce svévolně o databázi rozhodovala. Další její výhodou je její neměnnost. Data jsou propojena dohromady a změna i sebemenšího detailu zneplatní kompletně celou databázi. Díky decentralizaci má blockchainová síť další výhodu – transparentnost. Tím, že je blockchainová databáze decentralizovaná, všichni účastníci sítě mají přístup ke stejným datům. Každá zapsaná informace se stává součástí veřejného záznamu, který je viditelný pro všechny účastníky. Tato otevřenost a transparentnost přispívá k důvěře a odpovědnosti v síti, protože jakékoli nesrovnalosti lze okamžitě identifikovat a vyřešit. [x21]

## Řetěz

Struktura blockchainové databáze celkem dobře odpovídá jejímu názvu. Jedná se o řetěz neboli sérii za sebou jdoucích bloků, kde informace, které chceme v databázi uložit, se ukládají právě do těchto bloků. Onu propojenost ztělesňuje další informace v bloku, kterou je hash předchozího bloku, a proto pokud by byla data v bloku změněna, tak hash přestane bloku odpovídat. Samotný hash sám o sobě ještě nezajišťuje bezpečnost celého řetězu. Hashovací funkce jako je třeba SHA-256 je relativně rychlá a přepočítat hashe pro 1000 bloků by netrvalo déle než pár vteřin na většině zařízení. Na tento problém odpovídají mechanismy konsensu, které jsou vysvětleny v kapitole 5.2.



Obrázek 16 – Struktura blockchainové databáze

## Blockchainová síť

Jak je zmíněno v předchozí kapitole, blockchainová databáze je decentralizovaná, tedy funguje na více propojených zařízeních. Těm se říká uzly. Velké blockchainové sítě jako je třeba Bitcoin obsahuje přes 11000 uzlů [x22]. Satoshi Nakamoto, autor první široce používané blockchainové sítě – kryptoměny Bitcoin uvádí 5 různých činností, které uzly musí plnit:

1. „Rozeslání nových zpráv do všech uzlů.
2. Shromáždění nových zpráv do bloku.
3. Práce na hledání obtížného důkazu práce pro svůj blok.
4. Rozeslání nalezeného proof-of-work všem uzlům.
5. Přijímání bloku pouze v případě, že jsou všechny zprávy v něm platné, a ještě nebyly zapsány.“ [h11]

Veřejná síť

Veřejný blockchain umožňuje každému uzlu se volně připojit a aktivně se účastnit provozu blockchainové sítě. Každý uzel má možnost číst, zapisovat a kontrolovat probíhající aktivity na veřejné blockchainové síti, což podporuje jeho decentralizovanou povahu, která je často považována za klíčovou vlastnost blockchainu. Díky tomu, že žádná centrální autorita nekontroluje blockchain, není potřeba žádné povolení k přístupu k němu. [x11]

[x11] https://www.geeksforgeeks.org/blockchain-structure/

## Mechanismy konsensu

<https://builtin.com/blockchain/consensus-mechanism>

Mechanismus konsenzu je samoregulační zásobník softwarových protokolů zapsaných do kódu blockchainu, který synchronizuje síť do dohody o stavu digitální účetní knihy. Toho se dosahuje udržováním jediné datové sady – vzájemně odsouhlasené verze transakční historie blockchainu – spíše než využíváním každého uzlu nebo počítače v síti k individuální správě vlastní kopie databáze jako celku.

Přestože při programov

ání standardu sítě pro ověřování existuje řada mechanismů konsenzu, které je třeba vzít v úvahu, každý přístup je zaměřen na diskreditaci podvodníků v jejich pokusech odporovat záznamu.

PoW (Proof of work)

Volným překladem názvu tohoto mechanismu do češtiny by mohl být „důkaz o provedení práce“. Dost často se o něm také hovoří jako o těžbě bloků. Jeho myšlenka spočívá ve zvýšení obtížnosti pro vypočítání hashe bloku. Zavádí podmínky, jak má hash bloku vypadat, aby mohl být uznán za platný. Na příklad udá podmínku, že hash bloku musí začínat na 5 nul. Z definice hashe víme, že je konstantní k datům, ze kterých byla počítána. Proto se v bloku nachází hodnota důkazu, která je označována jako „Nonce“. Jeho hodnota není předem známa. Metodou pokus, omyl je hodnota měněna tak, aby podmínka pro náročnost hashe byla splněna. Zavedení podmínky se stane, že hash, která by byla předtím spočítána během okamžiku, trvá spočítat i několik desítek minut nebo let. Pro řetěz o 1000 blocích se doba přepočítaní násobí tisíckrát. To je problém, se kterým si ani výkonné počítače nedokážou poradit. [h11] Uzly před přidáním vytěženého bloku do řetězu ověří správnost hashe a správnost dat v něm.

PoS (Proof of stake)

Alternativa k PoW funguje na odlišném principu a nevyžaduje těžbu, a tedy i spoustu výpočetního výkonu a energie. Těžaři jsou nahrazeni tzv. validátory, kteří uzamknou určité množství peněz a tím získají právo ověřovat bloky.

Proces tvorby nových bloků pak funguje tak, že se do něj mohou zapojit všichni validátoři. Validátor, který vytvoří nový blok, je vybrán pseudonáhodně, přičemž šance na výběr se zvyšuje s množstvím uzamčených peněz. V případě, že validátor ověří nesprávné transakce, přijde o svůj vklad. Hrozba přijití o vložené peníze nutí validátory nevkládat do řetězu chyby a tím je síť zabezpečená. [h12]

Havel ve své práci porovnal PoW s PoS takto: „Každý z algoritmů má své výhody a nevýhody. Oba nabízí různá uplatnění. PoW často čelí kritice za svou neefektivitu a neekologičnost. Pálení elektřiny k dosažení konsenzu je dle zastánců PoS mechanismu zbytečné.

V PoS konceptu zamykáte měnu digitálního systému do samotného digitálního systému, abyste tento digitální systém validovali. PoW vyžaduje práci, tedy vnější prvek. Díky této provázanosti je útok na síť náročnější. PoW je však náchylný na jiné typy útoků, například 51% útok. PoS, na druhou stranu, neposkytuje neměnitelnost jako PoW.

Zpětná neměnitelnost je jednou z nejdůležitějších výhod blockchainu. Díky návaznosti bloků není možné změnit údaje několik bloků dozadu, protože by to změnilo celý řetězec bloků následujících. Zásadní rozdíl je v tom, že pokud chcete měnit minulé záznamy v řetězu s PoW, bude vás to něco stát. I kdybyste ovládali 100 % těžebních strojů v síti, stále musíte těžit, abyste přepsaly minulost. To vyžaduje ohromnou a nenávratnou investici. Pokud však souhlasí všichni akteří v PoS systému, mohou přepsat minulost bez nákladů.“ [h12]

[x22] https://kryptohodler.cz/pocet-bitcoinovych-uzlu-btc-full-node-je-na-historickem-maximu-po-nedavne-aktualizaci-bitcoin-core-zahrnujici-taproot/

[x21]https://www.techtarget.com/searchcio/definition/blockchain

# Mikroslužba zabezpečující technologii blockchain nad logovými záznamy informačního systému.

Tato mikroslužba realizuje všechny klíčové vlastnosti blockchainové sítě. Přijímá zprávy, logové záznamy, od uživatelů, které ukládá do bloků a následně bloky, těžbou zařazuje do řetězu. Těžba je realizovaná pomocí „proof of work“ mechanismu, který byl popsán v předchozí kapitole. Síť by se měla skládat z minimálně tří uzlů sítě, protože k zajištění konsenzu o pravosti řetězu je v případě odlišnosti uložených řetězů potřeba nadpoloviční většina k určení toho správného.

## Popis jednotlivých částí

Uzel

Základním stavebním prvkem sítě je uzel, anglicky „node“. Ten zajišťuje veškeré operace s řetězem. Přijímá pro něj data, poskytuje je ostatním uzlům, provádí těžbu, iniciuje těžbu v ostatním uzlech, vyhodnocuje výsledky těžby a vytěžené bloky ukládá do řetězu. Samozřejmě data dokáže nejen ukládat, ale také poskytovat na vyžádání. Konkrétně se jedná o Python tornado server. Při spuštění serveru se načte konfigurační soubor. V něm jsou pro něj informace o sousedním uzlu a náročnosti těžby. Také se vytvoří tabulka uzlů v síti, která je pro začátek prázdná a rozšiřuje se postupně, jak uzel získává informace o dalších uzlech v síti. Uzel má 11 koncových bodů. Ty používá pro komunikaci jak s klientem, tak i s dalšími uzly v síti. Jednotlivé koncové body jsou popsány níže v kapitole „Funkcionality uzlu“.

Klient/http dotazy

Klientská část může nabývat různých podob. Uživatel/klient poskytuje uzlu data pomocí http požadavku. Může tak dělat buď manuálně, nebo pomocí skriptu, který periodicky zasílá data do sítě. Tento script je přiložen ve zdrojových kódech a je možné jej pro mikroslužbu použít.

Kromě samotného odeslání dat, uživatel také může od uzlu získat řetěz, data v něm, pro další využití.

Tato mikroslužba je nastavená tak, že zahájení těžby iniciuje uživatel. Tedy zasláním specifického http požadavku jakémukoliv uzlu v síti spustí těžení na všech uzlech v síti.

Klient také má možnost poslat požadavek na vyhodnocení řetězů v síti. Jednotlivé uzly se chovají víceméně nezávisle až právě do doby, kdy přijde příkaz, aby uzly svoje řetězy sjednotili. Tím se právě eliminuje možná nežádoucí aktivita útočníka, nebo v krajních příkazech kolize při komunikaci mezi uzly.

Zahájení těžby a vyhodnocení řetězu jsou pokyny, které by teoreticky ve svém důsledku neměl dávat libovolný uživatel, ale měly by být autonomními součástmi uzlů. Pro tuto mikroslužbu jsem se rozhodl tak neučinit, protože díky manuálním příkazům dokáži lépe demonstrovat jejich činnost. Více na toto téma je uvedeno v kapitole „Možnosti dalšího rozvoje“.

## Funkcionality uzlu

Třída Blockchain

Pro uchování řetězce a práci s ním používají uzly třídu „Blockchain“ ze souboru „blockchain.py“ v adresáři „modules“. Při iniciaci třídy se vytvoří šest proměnných.

* „current\_logs“ – do této proměnné se ukládají příchozí zprávy, logové záznamy, které později budou v rámci bloku těženy a posléze přidány do řetězce.
* „chain“ – řetezec, hlavní proměnná třídy Blockchain
* „last\_block\_timestamp – časová značka posledního přidaného bloku do řetězce. Používá se pro řešení kolizí při přijetí vytěženého bloku od více uzlů.
* „ismining“ – stavová proměnná
* „isresolving“ – stavová proměnná
* „complexity“ – náročnost hashe. Číslo, které udává, jak náročná má být těžba.

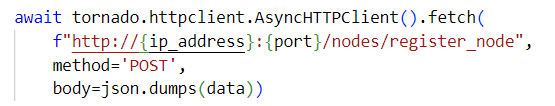
Kromě proměnných se iniciuje také osm funkcí pro práci s řetězcem.

* „valid\_chain“ – Kontroluje řetěz a rozhoduje, zda je v něm chyba.
* „resolve\_conflicts“ – Porovnává řetězy od ostatních uzlů a rozhoduje, který řetěz má největší četnost.
* „new\_block“ –Funkce, která přidá vytěžený blok do řetězu.
* „valid\_block“ – V případě přijetí více bloků od uzlů v rámci stejné těžby tato funkce rozhodne, který blok přidá do řetězu.
* „new\_log“ – Pokud zpráva ještě nabyla přidána, přidává příchozí zprávu do bloku, který bude příště těžen.
* „hash“ – Vrací hash bloku, který byl přidán jako parametr.
* „mining“ – Asynchronní funkce, která se stará o těžbu nových bloků.
* „valid\_proof“ – V parametru přijímá blok a vrací zda hash bloku odpovídá požadavku na náročnost hashe.

Jednotlivé funkce uzel volá podle potřeby a úkolům, které musí vykonat.

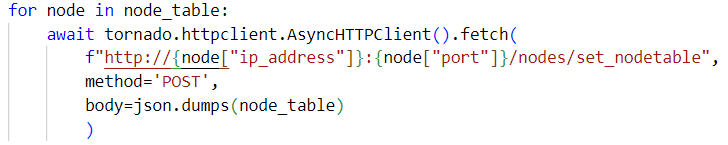
Registrace v síti

První úkolem, který uzel po svém spuštění musí učinit je zaregistrovat se do sítě. Toho docílí tím způsobem, že odešle POST http požadavek na sousední uzel, který má nastaven ve svém konfiguračním souboru. K požadavku uzel přiloží informace o své adrese, portu a jedinečném identifikátoru.



Obrázek 17 – http požadavek (registrace v síti)

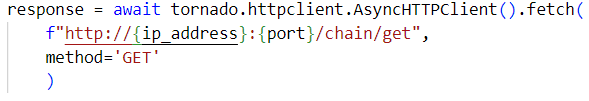
Sousední uzel, v okamžiku. kdy přijme požadavek na koncovém bodu „nodes/register\_node“ přidá informace o novém uzlu do své tabulky uzlů a všem uzlům v síti odešle požadavek na aktualizaci tabulky. V těle požadavku je tabulka uzlů, kterou ostatní uzly aktualizují tu svoji.



Obrázek 18 – http požadavek (nastavení tabulky uzlů)

Tímto způsobem nový uzel získává aktuální přehled o všech uzlech v síti. Ten potřebuje, aby ostatním uzlům mohl odesílat zprávy, přijaté od klienta nebo vytěžené bloky.

Kromě tabulky uzlů nový uzel potřebuje také aktuální uzel v síti. Ten, který si vytvořil při svém spuštění (při vytvoření třídy „Blockchain“) se neshoduje s řetězy v síti. Pro získání řetězu zavolá http požadavek na sousední uzel. Odpovědí je řetěz souseda nového uzlu, kterým přepíše svůj dosavadní uzel.



Obrázek 19 – http požadavek (žádost o řetěz)

Je to druhý a poslední okamžik, kdy nový uzel vnímá některý z již existujících uzlů v síti za svůj bližší. Po své registraci uzel informaci o svém sousedním zahazuje a nikdy více ji nevyužije. I pomocí toho je dosažena ona zmiňovaná decentralizace sítě.

Na obrázku 15 je graficky znázorněn průběh registrace nového uzlu v síti. Uzel na levé straně žádá svého souseda napravo a registraci. V horní části obrázku jsou dva další uzly v síti, kteří samotnou registraci neprovádí, ale přijímají informaci o novém uzlu v síti.

Obsah obrázku text, diagram, zástrčka

Popis byl vytvořen automaticky

Obrázek 20 – Zaregistrování uzlu v síti

Žádost o tabulku uzlů

Jak jsem popisoval v předchozí kapitole, tabulka uzlů se aktualizuje automaticky. Není tedy potřeba, aby se uzly doptávaly na tabulku. Tato možnost tady, ale je. Nemá přímý dopad na fungování sítě, ale může sloužit uživateli/administrátorovi pro lepší přehled o síti. Tabulku lze získat pomocí GET http požadavku na jakýkoliv uzel v síti.



Obrázek 21 – http požadavek (žádost o tabulku uzlů)

Přijetí zprávy

Pro přijímání zpráv od klientů uzel používá koncový bod „logs/new“. Klient na něj odešle POST http požadavek, kdy v těle funkce je zpráva, v našem případě jeden nebo více logových záznamů, které chce klient odeslat. Jako demo data pro ukázku odesílání systémových logů používám dataset „Windows\_2k.log“, který jsem získal z veřejného zdroje na stránce Github [x40].



Obrázek 22 – http požadavek (žádost o tabulku uzlů)

Uzel po přijetí zprávy ji nejprve zkontroluje a posléze ji přidá do zásobníku „current\_logs“ ve třídě „Blockchain“. Kontrolou se rozumí ověření, zda zpráva je ve slovníku uložená pod názvem „message“ a zda se zpráva už v zásobníku nenachází. Zprávu také odešle všem ostatním uzlům v síti. Tato zpráva zatím není přidána do řetězu. To se stane až je celý zásobník „current\_logs“ vložen do bloku a ten bude vytěžen. V této chvíli tomu tak zatím není, a proto pokud by uživatel hned po odeslání zprávy zažádal o řetěz, tak ten danou zprávu obsahovat nebude. Po zdárném vložení zprávy do zásobníku a přeposlání zprávy zbylým uzlům se klientovi vrací kladná odpověď. V případě, že zpráva už v zásobníku byla, dostane klient odpověď zápornou.

Na obrázku 16 je graficky znázorněna komunikace v síti po přijetí zprávy.

Obsah obrázku diagram, text, Plán

Popis byl vytvořen automaticky

Obrázek 23 – Schéma přijetí zprávy

1. Klient odesílá uzlu zprávu
2. Uzel přidává zprávu do zásobníku
3. Uzel přeposílá zprávu ostatním uzlům
4. Ostatní uzly přijímají zprávu a vkládají ji do svých zásobníků
5. Uzly přeposílají zprávu
6. Uzly zamítají zprávu, protože zpráva už byla přidána.

Určitě je nasnadě otázka, zda je nutné, aby uzly přeposílali zprávu znova. Dochází tak ke zbytečné komunikaci, a i zbytečnému zahlcení sítě. To je sice pravda, ale uzly nemusí rozhodovat, zda zprávu získali od uzlu, nebo od klienta, a především to umožňuje uzlům, v případě potřeby, vložit do řetězu zprávu i přímo od nich.

Těžba

Těžení je nejvíce komplexní funkcionalita, kterou uzel umí. Úkolem této funkcionality je vytěžit zprávy v zásobníku „current\_logs“ a vytěžený blok vložit na konec řetězu. V současné době se zahájení těžby ovládá manuálně, ale jediný důvod pro to je jednodušší vysvětlení toho, jak těžba funguje. V kapitole „Možnosti dalšího rozvoje“ uvádím, jakým způsobem by se zahájení těžby dalo automatizovat.

Těžba se zahájí manuálně posláním http GET požadavku na koncový bod „mine/start“ libovolného uzlu.



Obrázek 24 – http požadavek (zahájení těžby)

Uzel při přijetí požadavku vyhodnotí, zda právě neprobíhá buď těžba, nebo vyhodnocování řetězu. Pokud ano, tak požadavek na zahájení těžby odmítne. V opačném případě pošle pokyn všem uzlům v síti, včetně sebe, aby zahájili těžbu.



Obrázek 25 – http požadavek (příkaz k zahájení těžby)

Obsah obrázku text, snímek obrazovky, diagram, design

Popis byl vytvořen automaticky

Obrázek 26 – Schéma zahájení těžby

1. Pokyn od uživatele/administrátora k zahájení těžby
2. Vyhodnocení zda těžba právě neprobíhá
3. Vyhodnocení zda nejsou řetězy v síti právě porovnávány
4. Příkaz všem uzlům v síti k zahájení těžby

Každý uzel předtím, než zahájí samotnou těžbu nejprve sestaví těžený blok ve formátu JSON.

Skládá se z:

* Index – jedinečné číslo pro celý řetěz. Jedná se o hodnotu o jedna vyšší než poslední blok v řetězu.
* Začátek těžby – čas, kdy začala těžba bloku
* Logy – záznamy od klientů uložené v zásobníku „current\_logs“ s přidáním zprávy o tom, kdo daný blok vytěžil.
* Důkaz – hodnota, která bude při těžbě měněna.
* Předešlá hash – provázání do řetězu pomocí hashe posledního bloku v řetězu

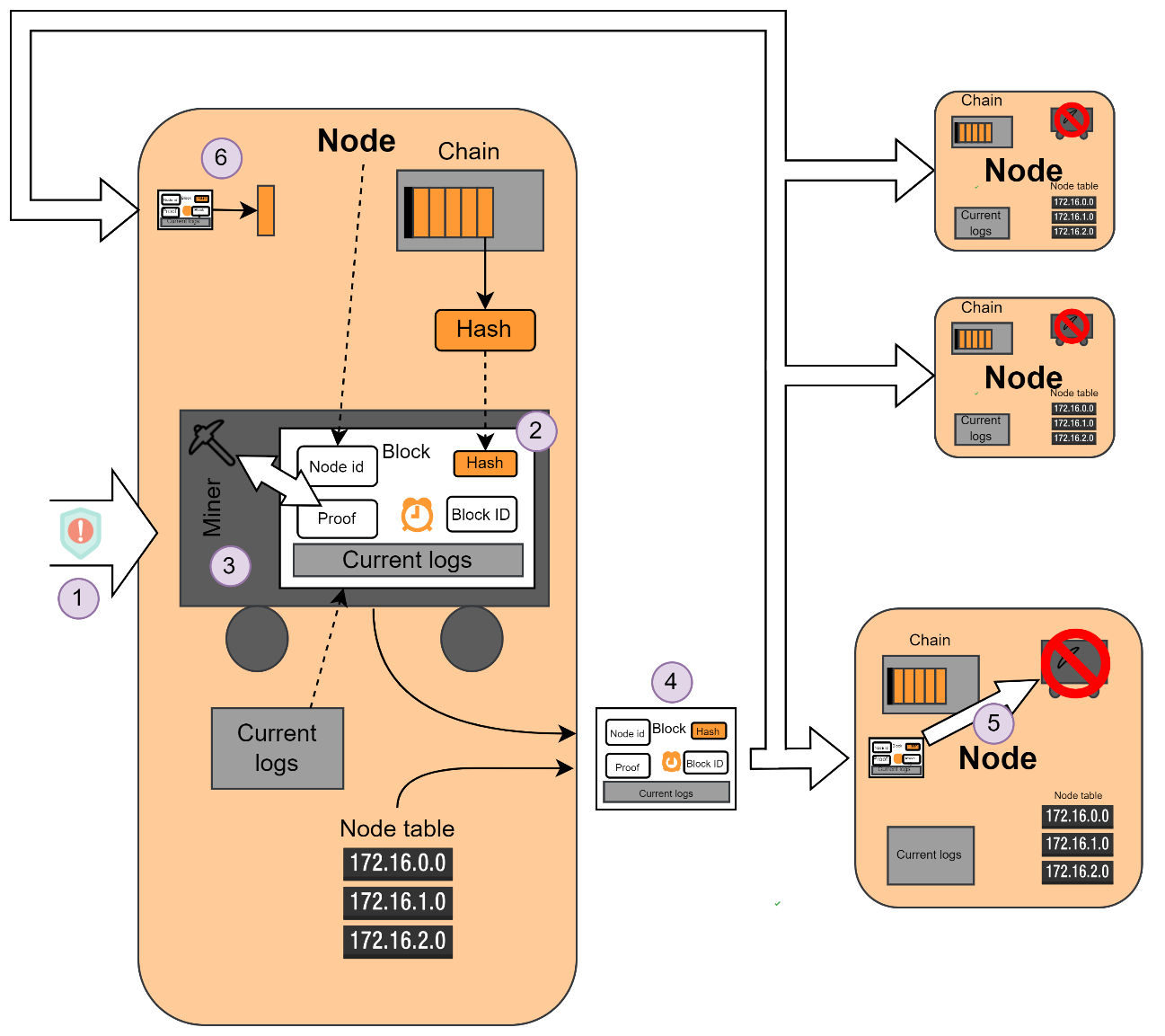
Takto sestavený blok může být poslán do těžení. Těžením, se rozumí hledáním takové hodnoty pro důkaz v bloku, aby hash celého bloku odpovídala požadavkům na náročnost hashe. Funkce, která provádí těžení je součástí třídy „Blockchain“ a jmenuje se „mining“. Jako parametry bere těžený blok, funkci, která odesílá výsledek všem uzlům v síti a identifikátor uzlu. Funkce ještě z konfiguračního souboru vezme hodnotu náročnosti pro těžbu. Základně je v něm uvedena hodnota 5. To znamená, že pokud má být blok úspěšně vytěžen, tak funkce, která těžbu provozuje musí nalézt takovou hodnotu, aby se prvních 5 znaků hashe celého bloku rovnalo 5 nulám. V průběhu testování této mikroslužby jsem došel k závěru, že v případě nízké náročnosti může docházet ke kolizím, která síť není schopna sama vyřešit, a proto doporučuji považovat náročnost 5 za minimální hodnotu. Shora náročnost omezená není. Doporučuji náročnost nastavit v závislosti na výkonu strojů, na kterých uzly běží a také na počtu uzlů v síti. V mém případě jsem testoval síť o třech uzlech, které běželi na stejném stroji. Pro hodnoty vyšších než 8 byla doba těžby, tak dlouhá, že bloky nebylo možné vytěžit v rozumné době. V mém případě to znamenalo, že ani po 30 minutách nebyla těžba zdárně ukončena.

Po nalezení vhodné hodnoty důkazu je těžba ukončena a funkce „mining“ ve třídě „Blockchain“ odešle všem uzlům v síti vytěžený blok s připnutým časovým razítkem o čase ukončení těžby. To je pokyn pro všechny uzly, aby svoji, prozatím neúspěšnou těžbu ukončili a po ověření přijatého bloku ho přidali do svých řetězů.

K vyhodnocení přijatého bloku se používá funkce „valid\_block“ ve třídě „Blockchain“. Tato funkce nejprve zkontroluje důkaz v bloku. Pokud byl vypočítán správně, tak hash by měla opět odpovídat požadavkům na náročnost. Poté se zkontroluje, zda index přijatého bloku je o jeden vyšší než poslední blok v řetězu a jako poslední se zkontroluje, zda v hodnota předešlé hashe v přijatém bloku odpovídá hashi posledního bloku v řetězu. Pokud jsou všechny podmínky splněny, tak funkce „valid\_block“ vrací hodnotu „pravda“ a blok je přidán do řetězu.

Zmínil jsem se také, že může docházet ke kolizím. K těm může dojít, pokud dva uzly vytěží své bloky současně. To znamená, že než přijde k uzlu výsledek těžby jiného bloku, on sám těžbu dokončí a rozešle všem uzlů svůj vytěžený blok. V tom případě mají uzly k dispozici dva bloky a oba jsou relevantní pro přidání do řetězu. Jenže řetězy musí zůstat na všech uzlech totožné, a proto přichází do hry časová známka, která je připojená k vytěženému bloku. Uzly přečtou časovou známku obou bloků a vyberou ten blok, který byl vytěžen dříve. Kolize se často stávají, pokud je nastavena nízká náročnost těžby. Když jsou kolize málo časté, síť si s nimi bez problému poradí. V případě, že je náročnost nastavena na nízkou úroveň a doba těžby trvá v průměru pár desítek milisekund, pak nastává neopravitelný problém. Proto doporučuji náročnost těžby ponechat minimálně na hodnotě 5, nebo ji zvýšit v závislosti na počtu uzlů a výkonu strojů, na kterých uzly běží. Ideální je, pokud doba těžby trvá déle něž 10 vteřin.

Po přidání bloku do řetězu uzel začíná další těžbu. Jedná se tak o nekonečný proces až do okamžiku, kdy uživatel/administrátor manuálně těžbu ukončí. Grafické znázornění průběhu těžby lze najít na obrázku 18.



Obrázek 27 – Schéma těžení bloku a distribuce výsledku

1. Pokyn od některého z uzlů k zahájení těžby
2. Sestavení bloku
3. Těžba bloku
4. Odeslání vytěženého bloku všem uzlům v síti
5. Ukončení těžby na ostatní uzlech
6. Vyhodnocení přijatého bloku

Ověření řetězu

Jednou za určitý čas by síť měla zkontrolovat, zda souhlasí řetězy v uzlech na síti. Zda některé uzly, potenciálně škodlivé, nemají jiné řetězy, a tedy i data než zbytek sítě. Může se tak stát, pokud někdo s přístupem k datům v řetězu tyto data změnil. Vše vyřeší právě porovnání řetězů. Na pokyn uživatele/administrátora libovolnému uzlu, celá síť, pokud probíhala, ukončí těžbu. Respektive vyčká, než se dokončí těžba posledního bloku a další už nespustí.



Obrázek 28 – http požadavek (příkaz k porovnání řetězů)

Každý uzel si získá od každého uzlu v síti hash jeho řetězu. Tyto hashe porovná a vybere tu hash, která má nejvyšší četnost. Vybranou hash porovná s hashí svého řetězu. Pokud odpovídá, ukončuje proces a uživateli odpovídá, že řetěz nebyl změněn. Pokud ovšem hash s nejvyšší četností neodpovídá, zažádá libovolný uzel, od kterého tato hash přišla, aby mu poskytl jeho řetěz. Tím pak přepíše ten svůj a získává tak řetěz, který byl odsouhlasen v síti.

To je právě jedna z výhod blockchainové sítě. Redundance dat a jejich vzájemné porovnávání tak, že alternativní, zfalšovaná, data nemají možnost se prosadit. Útočník by musel ovládnout nadpoloviční většinu uzlů, aby mohl měnit data v řetězech, a to je obtížné, obzvlášť pokud síť postavena z většího počtu uzlů.

Poskytnutí řetezu

Jedna z nejzákladnějších funkcionalit jak uzlu, tak i jakéhokoliv informačního systému, je poskytování dat. V našem případě se jedná o řetěz. Ten uživatel, nebo i uzly mohou získat pomocí http GET požadavku na libovolný uzel v síti.



Obrázek 29 – http požadavek (žádost o řetěz)

Předtím než je řetěz poskytnut je provedena kontrola na pravost řetězu. Zabrání se tak možnosti šíření falešně upraveného řetězu. Tato kontrola vezme blok po bloku a kontroluje, zda blok obsahuje správnou hash a zda obsahuje hash předchozího bloku. Nemůže se tak stát, aby uzel poskytoval data, která neprošly těžbou.

Průvodce pro spuštění mikroslužby

V níže uvedených krocích je uveden postup, jak jednoduše spustit mikroslužbu Blockchain-service na vlastním stroji. Předpokladem pro správné fungování je nainstalovaný Python verze 3.12.2 dostupný z <https://www.python.org/downloads> .

Zdrojový kód mikroslužby a instalační balíček je dostupný na GitHub repozitáři. Stáhnete jej pomocí Git příkazu: „git clone -b Blockchain-service <https://github.com/naxit-01/DiplomovaPrace.git>“.

Po stažení přejděte do složky „DiplomovaPrace“ a spusťte instalační soubor „setup.bat“. Ve vašem adresáři se vytvoří virtuální prostředí a do něj se vám nainstalují všechny potřebné knihovny. Zároveň se vám v adresáři vytvoří dva spustitelné soubory: „activate\_client.bat“ a „activate\_blockchain.bat“. V této chvíli by mělo být vše připraveno pro spuštění uzlů a klientů.

Doporučuji v síti spustit alespoň tři uzly. Toho docílíte spuštěním, alespoň třikrát za sebou souboru „activate\_blockchain.bat“. Uzly samozřejmě nemusí být všechny na stejném stroji. Můžete je rozjet na jakémkoliv stroji v dostupné síti. Důležité je, aby stroje spolu mohli komunikovat, správné uvedení sousedních uzlů v konfiguračním souboru „config.ini“.

Po spuštění uzlů můžete zprovoznit klienta spuštěním souboru „activate\_client.bat“. Klient se vás nejprve zeptá, se kterým uzlem budete chtít komunikovat. Pokud některý uzel běží na stejném počítači jako je klient, můžete ponechat pro IP adresu a port původní hodnoty. Klient poskytuje 5 různých příkazů, kterým ovládáte blockchainovou síť.

1. Získat tabulku uzlů – Dotážete se zvoleného uzlu, jaká je tabulka uzlů. Tím nijak síť neovlivňujete a je to pouze informace pro vás.
2. Odeslat log – Odešlete buď vámi napsanou zprávu, nebo náhodně vybraný systémový log z datasetu uloženém v adresáři.
3. Zahájení těžby – Odešlete příkaz celé síti aby zahájila těžbu.
4. Vyhodnotit řetězy – Odešlete příkaz síti, aby ukončila těžbu a porovnala řetězy uložené na jednotlivých uzlech v síti.
5. Získat řetěz – Dotážete se zvoleného uzlu na jeho uložený řetěz.

Pokud by měl klient sloužit pouze pro odesílání dat do sítě, tak jediný důležitý příkaz je „odeslat log“. Zbylé příkazy jsou pro administrativní účely a v budoucnu by měly být nahrazeny automatickými procesy viz. kapitola „Možnosti dalšího rozvoje“.

# Postkvantová blockchainová síť

V předchozích kapitolách byly představeny dvě na sobě nezávislé mikroslužby pro postkvantové šifrování a blockchainovou síť. V této kapitole bych chtěl představit jejich spojení do jednoho souvislého celku. Jedná se tak o vyvrcholení celé mé práce, protože do této doby její dvě nezávislé části se budou doplňovat a obě tak přispívat k bezpečnějšímu fungování informačního systému společně. V kapitole 6 komunikovala blockchainová síť na základě nešifrované http komunikace. V této kapitole bude blockchainová síť, se zachováním všech funkcionalit, komunikovat zašifrovaným provozem tak, jak bylo popsáno v kapitole 3. Změna proběhla ve využití jiné komunikační funkce, doplnění koncového bodu pro výměnu symetrických klíčů a doplnění souboru skriptů o certifikační autoritu.

## Výměna komunikační funkce

V nešifrované blockchainové síti entity využívali jednoduchou funkci „send\_request“ pro odesílání http požadavků.

Obsah obrázku text, Písmo, řada/pruh, snímek obrazovky

Popis byl vytvořen automaticky

Obrázek 30 – Nešifrovaná komunikace

Tato funkce odesílá data v nešifrované, nepodepsané podobě, a proto není bezpečná. Postkvantová blockchainová síť využívá jinou funkci, kterou je možné vidět na obrázku 10. Její podrobný popis je uveden v kapitole 3.2.3. Pro připomenutí jen uvedu, že tato funkce využívá knihovny „kemAlgLib.py“ a „signAlgLib.py“ pro postkvantové šifrování.

Obsah obrázku text, snímek obrazovky, Písmo, číslo

Popis byl vytvořen automaticky

Obrázek 31 – Šifrovaná komunikace

Doplnění koncový bodů o KEM algoritmus

K 11 už stávajícím koncovým bodům blockchainové sítě je přidán ještě jeden další. Jedná se o koncový bod, který obsluhuje výměnu symetrického klíče pomocí KEM algoritmu. Popis výměny symetrických klíčů jsem uvedl v kapitole 3.2.1 na obrázku 11. Na přiloženém obrázku je zobrazena třída, která pro uzel obstarává výměnu klíčů se stranou, která s ní chce navázat komunikaci.

Obsah obrázku text, snímek obrazovky, Písmo, číslo

Popis byl vytvořen automaticky

Obrázek 32 – Koncový bod pro KEM výměnu klíčů

Certifikační autorita – nezbytná součást kryptograficky zabezpečené sítě

Jak už lze vidět v předchozí kapitole, pro postkvantově zabezpečenou komunikaci je nezbytná certifikační autorita. Nejenom na obrázcích 10 a 41 lze vidět, že uzel postkvantové blockchainové sítě vyžaduje pro ověření autentičnosti zpráv certifikační autoritu. Na zmíněném obrázku, na sedmém řádku, lze vidět volání funkce „ask\_public\_key“, která uzlu vrátí veřejný klíč entity, která s ním chce navázat spojení. Bez něho není uzel schopen poznat, zda je druhá strana opravdu ta, za kterou se vydává. Z toho vyplývá, že certifikační autorita je nezbytnou součástí postkvantové blockchainové sítě. Její roli a druhy služeb, které obsluhuje jsou popsány v kapitole 6.2.

Ukázka fungování postkvantové blockchainové sítě

Jak už bylo zmíněno v úvodu této kapitoly. Postkvantová blockchainová síť je výsledkem této práce. Spojuje vše, co jsem se během jejího vypracování naučil a obě mikroslužby, které jsem napsal. V této kapitole představím ukázku postkvantové blockchainové sítě při jejím provozu. Tím také dokážu funkčnost dvou předchozích mikroslužeb.

Průvodce pro spuštění Postkvantové blockchainové sítě

V níže uvedených krocích je uveden postup, jak jednoduše spustit PostChain-network na vlastním stroji. Předpokladem pro správné fungování je nainstalovaný Python verze 3.12.2 dostupný z <https://www.python.org/downloads>, nástroj Git dostupný z <https://git-scm.com/> a kompilátor programovacího jazyka C: Microsoft Visual C++ 14.0 nebo novější, který je dostupný na <https://visualstudio.microsoft.com/visual-cpp-build-tools/>.

Zdrojové kódy a instalační balíček je dostupný na GitHub repozitáři. Stáhnete jej pomocí Git příkazu: „git clone -b PostChain-service <https://github.com/naxit-01/DiplomovaPrace.git>“.

Po stažení přejděte do složky „DiplomovaPrace“ a spusťte instalační soubor „setup.bat“. Ve vašem adresáři se vytvoří virtuální prostředí a do něj se vám nainstalují všechny potřebné knihovny. Zároveň se vám v adresáři vytvoří dva spustitelné soubory: „activate\_client.bat“ a „activate\_blockchain.bat“. Instalace může trvat několik minut. V této chvíli by mělo být vše připraveno pro spuštění uzlů a klientů.

Doporučuji v síti spustit alespoň tři uzly. Toho docílíte spuštěním, alespoň třikrát za sebou souboru „activate\_node.bat“. Uzly samozřejmě nemusí být všechny na stejném stroji. Můžete je rozjet na jakémkoliv stroji v dostupné síti. Důležité je, aby stroje spolu mohli komunikovat, správné uvedení sousedních uzlů v konfiguračním souboru „config.ini“.

Po spuštění uzlů můžete zprovoznit klienta spuštěním souboru „activate\_client.bat“. Klient poskytuje 5 různých příkazů, kterými ovládáte blockchainovou síť.

1. Získat tabulku uzlů – Dotážete se zvoleného uzlu, jaká je tabulka uzlů. Tím nijak síť neovlivňujete a je to pouze informace pro vás.
2. Odeslat log – Odešlete buď vámi napsanou zprávu, nebo náhodně vybraný systémový log z datasetu uloženém v adresáři.
3. Zahájení těžby – Odešlete příkaz celé síti, aby zahájila těžbu.
4. Vyhodnotit řetězy – Odešlete příkaz síti, aby ukončila těžbu a porovnala řetězy uložené na jednotlivých uzlech v síti.
5. Získat řetěz – Dotážete se zvoleného uzlu na jeho uložený řetěz.

# Možnosti dalšího rozvoje

## Zavedení plné certifikační autority

CA kterou jsem použil, jak jsem zmínil, nesplňuje všechny požadavky na certifikační autoritu. Entity, když se k ní přihlašují a žádají o svůj soukromý podpisový klíč se žádným způsobem neautentizují. V běžném fungování, CA neposkytuje všem tazatelům certifikát entity, ale poskytuje ho jen samotné entitě, která ho přikládá ke své zprávě. Příjemce zprávy ověří veřejným podpisovým klíčem certifikační autority certifikát odesílatele a jím pak ověří podepsanou zprávu. Vytvoření CA nebylo cílem mé práce, a proto jsem její fungování značně zjednodušil. Pro plné nasazení mikroslužeb bude potřeba vyměnit CA ze jinou, běžně používanou, a upravit komunikaci mezi entitami podle požadavků nové CA. Tato úpravu by dle mého odhadu měla být triviální.

## Implementace autorských řešení algoritmů CRYSTALS Kyber a CRYSTALS DILITHIUM

Při tvorbě knihoven pro postkvantové algoritmy jsem se snažil zahrnout co nejvíce relevantních zdrojů. Velkou pozornost jsem věnoval přímo autorským řešením daných algoritmů. V případě CRYSTALS-Kyber a CRYSTALS-DILITHIUM jsem nedospěl ke zdárnému závěru. Jejich řešení jsou stavěna pro demonstraci fungování jejich algoritmů, ale ne přímo na jejich využití. Jimi dodané aplikace dokážou provést série testů na fungování jejich kryptografických mechanismů. Když jsem se snažil tyto testy rozebrat a využít jejích obsah pro moji potřebu, dokázal jsem se dostat pouze do bodu, kdy moje požadavky fungovaly v rámci jedné entity. V případě Kyber jsem byl schopný vygenerovat KEM klíče, z veřejného KEM klíče vytvořit symetrický klíč a ciphertext a ze ciphertextu a soukromého klíče dopočítat kopii symetrického klíče. V případě algoritmu DILITHIUM jsem byl schopný vygenerovat podpisové klíče, podepsat zprávu soukromým podpisovým klíčem a následně tento podpis ověřit veřejným klíčem. Jenže, jak jsem zmínil, vše fungovalo v rámci jedné entity – programu, ale když jsem jednotlivé části separoval, tak hodnoty klíčů přestaly odpovídat. S mými schopnostmi a časovými prostředky jsem nebyl schopen pochopit, v čem nastal problém, a proto autorské řešení těchto algoritmů nejsou implementovány. Vyřešení zmíněných problémů a zařazení autorských řešení považuji za důležité a rozhodně by se tímto směrem měl ubírat nejbližší rozvoj práce.

## Automatizace těžby a porovnání řetězů

Pro lepší demonstraci jsem blockchainovou síť nenapsal plně automatickou. Proces zahájení těžby a vyhodnocení řetězů iniciuje uživatel. Pro plné fungování blockchainové sítě by se tyto aktivity měli konat automaticky. Například zavést podmínku, že pro případ, kdy se uzel vyskytne v síti jako první, tak zahájí těžbu okamžitě po získání svého klíče od certifikační autority. Další uzly, už tuto podmínku plnit nemusí, protože se k těžbě připojí automaticky při zahájení těžby dalšího bloku.

Co se týče automatizace porovnání řetězů, tak dle mého názoru je potřeba provést obsáhlejší úvahu o tom, jak často je potřeba ji plnit. Zda zvolit variantu kdy se řetězy mezi uzly budou porovnávat po každé těžbě bloku, nebo při každém zažádáním o řetěz, anebo třeba jen při připojení nového uzlu do sítě. To přímo záleží na budoucím využití této blockchainové sítě. Porovnání řetězů je nákladnou operací, protože při ní dochází k přerušení těžby a také, v případě konfliktů, k odesílání celé databáze přes síť. To může trvat delší dobu a v případě velkých databází i dny.

## Vstupní řetěz do sítě

Předchozí bod o budoucím rozvoji, porovnání řetězů, nebyl vůbec triviální. Jiné jsou jednodušší. Můžeme předpokládat, že blockchainová síť nebude v provozu bez přestávky celou dobu. Data lze zálohovat pomocí zavedeného koncového bodu pro získání celého řetězu. Jenže v současné době není uzel schopen takový řetěz vzít a zahájit nad ním novou blockchainovou síť. Uzel při své iniciaci vytváří vždy nový prázdný řetěz. Jednoduchou úpravou je možné tuto iniciaci přepsat tak, aby uzel, pokud je v síti první, použil řetěz ze svého úložiště.

## Autentizace entit

Blockchainová síť, kterou jsem napsal, je plně otevřená. Neprobíhá v ní žádná autorizace a ani autentizace. Lze předpokládat, že by potenciální uživatel sítě nechtěl, aby do blockchainové databáze mohl přistupovat kdokoliv a nebo aby kdokoliv mohl provozovat uzel sítě. Takové omezení je potřeba zavést do obsluhy koncových bodů uzlů tak, aby byly vyřizovány jen ty požadavky, které uživatel sítě uzná za vhodné. Tím se ale významně sníží její decentralizovanost.

## Proof of work

Uzly při svém těžení volají funkci „mining“, která spadá pod třídu „Blockchain“. Tato funkce hledá takovou hodnotu důkazu, aby byla splněna podmínka o náročnosti hasha celého bloku. Hledaní probíhá metodou pokus, omyl. Funkce generuje náhodná čísla a ověřuje pro ně, zda je podmínka splněna. Generování čísel jsem omezil na hodnoty od 0 do x-té mocniny čísla 100, kde x se rovná hodnotě náročnosti hashe. Dolní hranice se samozřejmě nabízela, ale horní hranici jsem určil sledováním úspěšně vytěžených bloků po určitou dobu. Jednalo se spíše o kvalifikovaný odhad než od podrobný výzkum. Na základě pravděpodobnosti se musí nevyhnutelně stát, že takovýto interval nebude stačit ani jednu uzlu a blok nebude možné vytěžit. V tom případě síť zamrzne a bude potřeba ji restartovat. Považuji za vhodné se hlouběji zamyslet na tím, jakou hodnotu by měla mít horní hranice. Hashovací funkce jsou ze své podstaty nepředpokládatelné. Nelze předpokládat výstup na základě vstupu do té doby, dokud není výstup spočítán. Je tedy možné, že určení horní hranice není možné a bude nutné ji neurčovat. V důsledky to bude znamenat zpomalení těžby bloků, protože většina hodnot důkazů se nacházela pod číslem 100 miliard.

## Otestovat hranice schopností sítě

Možností rozvoje této práce je spousta. Jako poslední, z mého pohledu důležitou, uvedu otestování schopností blockchainové sítě. V rámci testů jsem do postkvantové blockchainové sítě odesílal každou vteřinu novou zprávu. Síť se skládala ze tří uzlů, CA a jednoho klienta. Síť fungovala bez jakýkoliv problémů. Protože postkvantová blockchainová síť běžela v rámci jednoho počítače, nepovažoval jsem za vypovídající provést úplnou zátěžovou zkoušku. Její výsledky by nebyly vypovídající, protože by nezohledňovala zpoždění způsobené síťovým provozem. Určitě považuji za zajímavé spustit uzly na více lokalitách a ověřit, kolik zpráv dokážou bez kolizí zpracovat.

# ZÁVĚR

V této diplomovou práci jsem popisoval, jakým způsobem mohou být implementovány moderní bezpečnostní mechanismy pro ochranu dat v informačních systémech. V kapitole 2 jsem popsal význam postkvantových kryptografických algoritmů pro zajištění odolnosti komunikaci proti nadcházejícím kvantovým počítačům. V podkapitole 2.5 jsem uvedl seznam zavedených postkvantových algoritmů včetně jejich implementací v jazyce Python. Tyto nabyté znalosti jsem rozšířil o praktické využití v mikroslužbě, která zabezpečuje kvantově odolnou komunikaci. Její podrobný popis včetně postupu k použití jsem uvedl v kapitole 4. Ve své diplomové práci jsem se také věnoval metodě ochraně dat před nežádoucími změnami. Uvedl jsem technologii blockchainových sítí jako možné řešení, kterou jsem v kapitole 5 popsal včetně jejích nezbytných součástí a funkcionalit. Vyvinul jsem vlastní blockchainoou síť pro uchování logových záznamů z koncových zařízení. Dokumentaci k mé vlastní blockchainové sítě uvádím v kapitole 6. Na závěr této práce jsem sjednotil obě předešlé mikroslužby do jednoho celistvého řešení. Všechna zadání mé diplomové práce jsem splnil.

# SEZNAM POUŽITÉ LITERATURY

[04] https://www.python.org/downloads/

— Albert Einstein německo-americký fyzik, autor teorie relativity 1879–1955

**MONOGRAFIE**

1. [h1] Kaplan, M., Leurent, G., Leverrier, A.,  Naya-Plasencia, M. (2016). Breaking Symmetric Cryptosystems Using Quantum Period Finding. In: Robshaw, M., Katz, J. (eds) Advances in Cryptology – CRYPTO 2016. CRYPTO 2016. Lecture Notes in Computer Science(), vol 9815. Springer, Berlin, Heidelberg. <https://doi.org/10.1007/978-3-662-53008-5_8>

**ELEKTRONICKÉ ZDROJE**

1. BALCAROVÁ, Pavlína. *Konkurenceschopnost ekonomiky a její měření* [online]. Brno, 2007 [cit. 2014-12-30]. Dostupné z: ˂<http://is.muni.cz/th/75962/prif_m/diplomova_prace.pdf>˃. Diplomová práce. Masarykova univerzita.
2. Česká republika: hlavní makroekonomické ukazatele. *Český statistický úřad* [online]. 2014 [cit. 2015-01-28]. Dostupné z: ˂[https://www.czso.cz/documents/10180/20555311/HLMAKRO.xls](http://www.czso.cz/csu/redakce.nsf/i/cr:_makroekonomicke_udaje/$File/HLMAKRO.xls)˃.
3. [h2] JANKO, David. Symetrická a asymetrická kryptografie. [online]. *Itnetwork*. 2022, s. 3-7. Dostupné z: <https://www.itnetwork.cz/bezpecnost/symetricka-a-asymetricka-kryptografie>. [cit. 2024-04-24].
4. [h3] GILLIS, Alexander S. a BRUNSKILL, Vicki-Lynn, LUTKEVICH, Ben (ed.). Digital signature. Online. *TechTarget*. 2023, s. 1-2. Dostupné z: <https://www.techtarget.com/searchsecurity/definition/digital-signature>. [cit. 2024-04-24].
5. [h4] BURDA, Patrik. Možnosti postkvantové kryptografie. Brno, 2019, 38 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Petr Dzurenda,
6. [h5]CHEN, Lily; MOODY, Dustin a LIU, Yi-Kai. Post-Quantum Cryptography Standardization. Online. *NIST*. 2017. Dostupné z: <https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization>. [cit. 2024-04-24].
7. [h6] Jaká je klíčová myšlenka Shorova algoritmu kvantového faktoringu a jak využívá kvantové vlastnosti k nalezení periody funkce? Online. *Evropský institut pro certifikaci informačních technologií*. 2023. Dostupné z: <https://cs.eitca.org/kvantov%C3%A1-informace/eitc-qi-qif-kvantov%C3%A9-informa%C4%8Dn%C3%AD-z%C3%A1klady/Shors%C5%AFv-algoritmus-kvantov%C3%A9ho-faktoringu/shors-factoringov%C3%BD-algoritmus/p%C5%99ezkoum%C3%A1n%C3%AD-zkou%C5%A1ky-shors-factoring-algorithm/jak%C3%A1-je-kl%C3%AD%C4%8Dov%C3%A1-my%C5%A1lenka-za-algoritmem-kvantov%C3%A9ho-faktoringu-shors-a-jak-vyu%C5%BE%C3%ADv%C3%A1-kvantov%C3%A9-vlastnosti-k-nalezen%C3%AD-periody-funkce/>. [cit. 2024-04-24].
8. [h7] MCGREW, David; KAMPANAKIS, Panos; FLUHRER, Scott; GAZDAG, Stefan-Lukas; BUTIN, Denis et al., CHEN, Lidong; MITCHELL, Chris a MCGREW, David (ed.). State Management for Hash-Based Signatures. Online. In: Security Standardisation Research. Third International Conference, SSR 2016 Gaithersburg, MD, USA, December 5–6, 2016 Proceedings, 2016, s. 244-247. Dostupné z: https://doi.org/10.1007/978-3-319-49100-4\_11. [cit. 2024-04-25].
9. [h8] Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process. Online. In: . NIST, s. 16-17. Dostupné z: <https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf>. [cit. 2024-04-25].
10. [h9] AVANZI, Roberto; BOS, Joppe; DUCAS, Léo; KILTZ, Eike; LEPOINT, Tancrède et al. *CRYSTALS-Kyber*. Online. SCHWABE, Peter. CRYSTALS. 23.12.2020. Dostupné z: <https://pq-crystals.org/kyber/index.shtml>. [cit. 2024-04-25].
11. [h10] JONES, M.; BRADLEY, J. a SAKIMURA, N. RFC 7515. Online. IETF. Květen 2015, s. 5-11. ISSN 2070-1721. Dostupné z: https://datatracker.ietf.org/doc/html/rfc7515. [cit. 2024-04-26].
12. [h11]NAKAMOTO, Satoshi, 2009. *Bitcoin: a Peer-to-Peer Electronic Cash System* [online]. Bitcoin.org. Získáno z : <https://bitcoin.org/bitcoin.pdf>
13. [h12]MÁRIO HAVEL, Alza, 2020. Proof of Work a Proof of Stake (VŠE, CO VÍME) | Alza.cz. *Alza* [online]. 13 leden 2020. Získáno z : <https://www.alza.cz/proof-of-work-a-proof-of-stake> [cit. 26 duben 2024].

* Pozn. ČSN ISO 690 Informace a dokumentace – Pravidla pro bibliografické odkazy a citace informačních zdrojů

# SEZNAM PŘÍLOH

|  |  |  |  |
| --- | --- | --- | --- |
| Příloha A | − | Příklady kódů pro práci s post-kvantovými algoritmy |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Příloha a Příklady kódů pro práci s post-kvantovými algoritmy

Obsah obrázku text, snímek obrazovky, Písmo, číslo

Popis byl vytvořen automaticky

Obsah obrázku text, snímek obrazovky, Písmo, design

Popis byl vytvořen automaticky