**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

Zadání

I. Upřesnění podmínek zpracování závěrečné práce

V souvislosti se zpracováním závěrečné práce NEBUDE studentem nakládáno s utajovanými informacemi.

V souvislosti se zpracováním závěrečné práce NEBUDE studentem nakládáno s informacemi pro služební potřebu.

Závěrečná práce NEBUDE obsahovat utajené informace.

Závěrečná práce NEBUDE obsahovat informace pro služební potřebu.

Při zpracování závěrečné práce NEBUDOU zpracovávány osobní údaje.

II. Cíl závěrečné práce

Navrhnout možnosti využití nejnovějších poznatků z oblasti kvantového šifrování a využití blockchain pro potřeby ochrany dat v informačních systémech.

Nezbytné součásti diplomové práce:

Přehled algoritmů a jejich open source implementací z kategorie tzv. postkvantového šifrování.

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

Implementace mikroslužby zabezpečující operace typu šifrování a podepisování nad datovými bloky.

Implementace mikroslužby zabezpečující technologii blockchain nad logovými soubory informačního systému.

III. Pokyny pro zpracování závěrečné práce

Zpracujte přehled dostupných opensource implementací šifrování (tzv. postkvantové algoritmy), specificky výstupy ze soutěží nist.gov.

Zpracujte přehled možných útoků na standardně implementované informační systémy (datové úložiště).

Zpracujte přehled možných protiopatření (ochranných mechanismů).

Implementujte mikroslužbu využívající tzv. postkvantových algoritmů pro šifrování pro zabezpečení dat proti specifickým modifikacím dat.

Průběžně udržujte aktualizovaný přehled použité literatury a ke každé dílčí etapě zpracujte cca 7 stran shrnutí.

**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**

**Klíčová slova:**

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

**ABSTRACT**

**Key words:**

Obsah

[SEZNAM POUŽITÝCH ZKRATEK 9](#_Toc162977411)

[SEZNAM OBRÁZKŮ 10](#_Toc162977412)

[SEZNAM TABULEK 11](#_Toc162977413)

[ÚVOD 12](#_Toc162977414)

[2 Postkvantová kryptografie 13](#_Toc162977415)

[2.1 Současný stav 13](#_Toc162977416)

[2.2 NIST 13](#_Toc162977417)

[2.3 Seznam algoritmů 13](#_Toc162977418)

[2.3.1 Kyber 13](#_Toc162977419)

[2.3.2 DILITHIUM 13](#_Toc162977420)

[2.3.3 Falcon 13](#_Toc162977421)

[2.3.4 Sphincsplus 13](#_Toc162977422)

[3 Přehled ochraných mechanismů pro zabezpečení dat v informačním systému s ohledem na možnost odhalení manipulace s daty 14](#_Toc162977423)

[4 Blockchain 15](#_Toc162977424)

[4.1 Blockchainová síť 15](#_Toc162977425)

[5 Praktická část 16](#_Toc162977426)

[5.1 Cíl praktické části a její omezení 16](#_Toc162977427)

[6 Mikroslužba zabezpečující postkvantovou kryptografii 16](#_Toc162977428)

[6.1 Popis jednotlivých částí 17](#_Toc162977429)

[6.2 Nastavení symetrického klíče 17](#_Toc162977430)

[6.3 Získání podpisových klíčů a práce s nimi 20](#_Toc162977431)

[6.3.1 JWT 20](#_Toc162977432)

[6.3.2 Získání soukromého podepisovacího klíče 21](#_Toc162977433)

[6.3.3 Získání veřejného podpisového klíče 22](#_Toc162977434)

[6.4 Průběh komunikace při odeslání zprávy 24](#_Toc162977435)

[6.5 Průvodce pro spuštění mikroslužby 25](#_Toc162977436)

[7 Mikroslužba zabezpečující technologii blockchain nad logovými soubory informačního systému. 26](#_Toc162977437)

[7.1.1 Popis jednotlivých částí 26](#_Toc162977438)

[7.1.2 Popis funkcionalit nodu 26](#_Toc162977439)

[7.1.3 Průvodce pro spuštění mikroslužby 26](#_Toc162977440)

[8 Postkvantová blockchainová síť 26](#_Toc162977441)

[8.1.1 Popis jednotlivých částí 26](#_Toc162977442)

[8.1.2 Ukázka funkcionalit mikroslužeb 26](#_Toc162977443)

[8.1.3 Průvodce pro spuštění Postkvantové blockchainové sítě 26](#_Toc162977444)

[9 Možnosti dalšího rozvoje 26](#_Toc162977445)

[ZÁVĚR 26](#_Toc162977446)

[SEZNAM POUŽITÉ LITERATURY (vzor) 27](#_Toc162977447)

[SEZNAM PŘÍLOH 28](#_Toc162977448)

# SEZNAM POUŽITÝCH ZKRATEK

|  |  |
| --- | --- |
| AČR | Armáda České republiky (→ Styl seznam zkratek) |
| CA | Certifikační autorita |
| PQC | Post quantum cryptography  Postkvantová kryptografie |
| ISO | International Organization for Standardization  Mezinárodní organizace pro standardizaci |

# SEZNAM OBRÁZKŮ

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Strana |
| Obrázek 1 | − | Popis obrázku (→ styl Odstavec) | … XX |
| Obrázek 2 | − | Popis obrázku | … XX |
| Obrázek 10 | − | Popis výměny symetrického klíče mezi Alicí a CA |  |
| Obrázek 11 | − | Popis výměny symetrického klíče mezi Alicí a Bobem |  |
| Obrázek 12 | − | Popis žádosti o soukromý podpisový klíč |  |
| Obrázek 13 | − | Popis žádosti o veřejný podpisový klíč |  |
|  | − |  |  |
|  | − |  |  |
|  | − |  |  |
|  | − |  |  |
|  | − |  |  |

# SEZNAM TABULEK

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | Strana |
| Tabulka 1 | − | Popis tabulky | … XX |
| Tabulka 2 | − | Popis tabulky | … XX |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# ÚVOD

# Postkvantová kryptografie

## Současný stav

## NIST

## Seznam algoritmů

Kyber

DILITHIUM

Falcon

Sphincsplus

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

# Blockchain

## Blockchainová síť

Decentralizovana sit pro bezpecne ukladani komunikace. Jednotlive zpravy se vkladaji do bloku a ty se uzamikaji hashem a nasledne se pripojuji k retezu. Plati jednoduche pravidlo: co bylo jednou ulozeno do retezu uz z nej nikdy nemuze byt vzato. Zaroven zpravy v bloku musi potvrdit alespon 51% autorit, jinak zprava nebude prijata.

# Praktická část

## Cíl praktické části a její omezení

Cílem praktické části je představit dvě služby zabezpečující postkvantovou kryptografii a technologii blockchain. U každé služby nejprve popíšu její jednotlivé části, poté se věnuji jejím funkcionalitám a následně předvedu postup, jak danou službu spustit.

Obě služby jsou na sobě zcela nezávislé, ale součástí praktické části této diplomové práce je i jejich kombinace do Postkvantové blockchainové sítě. Také tuto síť v podkapitole 5.4 podrobně popíšu a předvedu její použití.

Ani jednu ze dvou služeb a jejich kombinace v praktické části mé diplomové práce jsem nepsal podle určitého zadání. Nikdo mi přesně nespecifikoval, co od služeb očekává, a proto jsem postupoval buď podle zavedených postupů vyčtených z internetu anebo podle mého vlastního rozhodnutí. Cílem práce tedy nebylo poskytnout někomu řešení na míru, ale představit funkční koncept, který vyžaduje pouze triviální změny k jeho použití v praxi.

Pro podepisování a následné ověření zpráv je potřeba přítomnost certifikační autority, která poskytuje jednotlivým uživatelům klíče k ověření podpisu. Vytvořit, nebo použít cizí certifikační autoritu není součástí této diplomové práce, a proto využívám jen její lehkou náhradu. Tato pseudocertifikační autorita nesplňuje všechny potřebné standardy, včetně těch bezpečnostních. Například v sobě nezahrnuje autentizaci uživatele, a proto ji nepovažuji za součást mé diplomové práce a budu se o ni zmiňovat jen v nejnutnějších případech.

# Mikroslužba zabezpečující postkvantovou kryptografii

První mikroslužbou, kterou bych chtěl popsat je ta, která zabezpečuje odesílání zpráv zašifrovaných a podepsaných pomocí postkvantové kryptografie. Její primární úkol je vcelku jednoduchý. Vzít zprávu a odeslat ji na požadovanou adresu. Ovšem, aby se tak mohlo stát i v postkvantovém světě, je potřeba série několika kroků popsaných níže. Tato mikroslužba obsahuje tři entity. Alice, Bob a Certifikační autorita. Pro ukázku funkčnosti této mikroslužby Alice každou vteřinu odesílá Bobovi zprávu, kterou on přijme, vytiskne na obrazovku a odešle Alici odpověď. Pro podepisování a šifrování zpráv využívají entity postkvantové algoritmy uvedené v kapitole 2.

## Popis jednotlivých částí

* 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.

## Nastavení symetrického klíče

První věc, kterou jakákoliv entita musí provést, pokud chce komunikovat s někým další je stanovit si bezpečný (zašifrovaný) komunikační kanál. Pro tento účel mají Alice, Bob i CA v modulu communication.py asynchronní funkci: async def define\_symmetric\_key(url, ALGORITHM, my\_address, pk, sign\_sk = None)

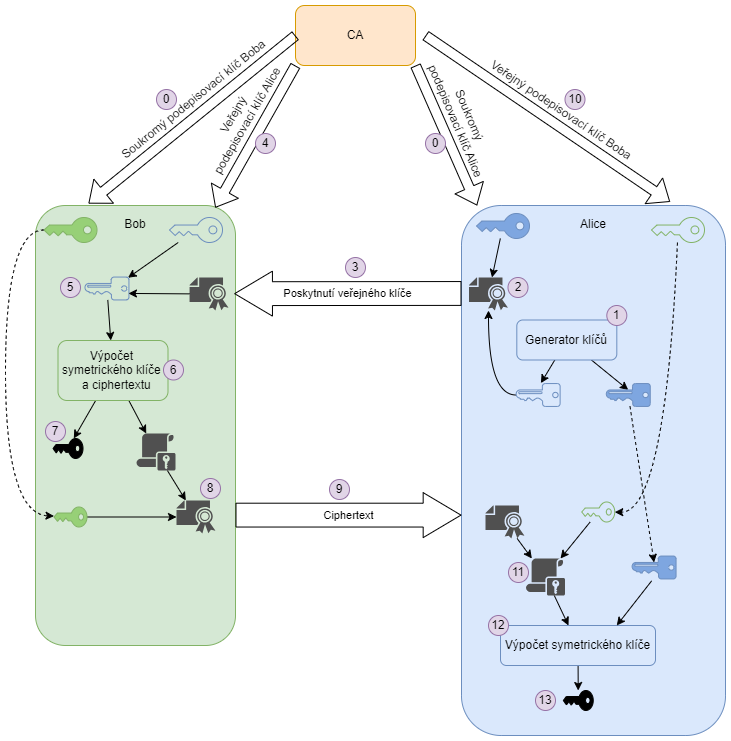
Tato funkce naváže spojení s druhou stranou a vzájemně si vymění informace takovým způsobem, aby výsledkem byl symetrický klíč, se kterým následně mohou otevřít nové, zašifrované spojení. Symetrický klíč jako takový nebyl nikdy poslán po síti, a tedy nikdo, kromě zúčastněných stran, o klíči nemůže vědět a komunikace je tedy bezpečná. Podrobný popis komunikace uvádím na dvou obrázcích níže. 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 10 – Popis výměny symetrického klíče 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 postkvantového klíče.
3. Alice svůj 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 klíče vygenerovaného v kroku 2.
10. Alice vlastní symetrický klíč, kterým bude šifrována pozdější komunikace.



Obrázek 11 - Popis výměny symetrického klíče mezi Alicí a Bobem

1. Alice a Bob získájí od CA svůj soukromý podpisový klíč (bude vysvětleno v 6.3).
2. Alice vygeneruje dvojici soukromého a veřejného postkvantového 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 6.3).
6. Bob ověří podpis a získá tak veřejný klíč Alice.
7. Bob, s pomocí veřejného 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 6.3).
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 klíče vygenerovaného v kroku 2.
14. Alice vlastní symetrický klíč, kterým bude šifrována pozdější komunikace.

## Získání podpisových klíčů a práce s nimi

Teorii o podepisování a podpisových klíčích jsem uvedl v kapitole 2 této práce včetně seznamu dostupných algoritmů a jejich činnosti. V této podkapitole už tedy zobecním činnost postkvantových kryptografických algoritmů na tři základní funkce: generování klíčů, výpočet symetrického klíče a ciphertextu, výpočet symetrického klíče.

V této podkapitole popíšu, jakým způsobem entity získávají soukromé a veřejné podpisové klíče. Než se k vysvětlení dostanu, rád bych nejprve popsal, jak přesně vypadá zpráva, která je podepsána soukromým klíčem podepisovatele.

JWT

Pro podepsanou zprávu využívám formát JSON web token (dále jen JWT). Jedná se o ustálený standard pro podepisovaní tokenů, nebo zpráv [1]. Je tvořen třemi Base64 zakódovanými částmi oddělenými tečkami: header (hlavička), payload (náplň) a signature (podpis).

* Hlavička obsahuje informace o tom, jak je zpráva podepsána.
* Náplň (payload) obsahuje data.
* Podpis (signature) je vytvořen pomocí algoritmu založeného na obsahu zprávy a tajným klíčem. Slouží k ověření integrity tokenu [1].

Protože jsem nenašel veřejnou knihovnu pro práci s JWT, která by zahrnovala buď postkvantové algoritmy, nebo by umožňovala přiložit vlastní podpisový algoritmus, napsal jsem vlastní implementaci JWT. Při jejím vývoji jsem bral ohled na komptabilitu s veřejnou Python knihovnou PyJWT [2]. Zajistil jsem tak, aby projekty, které by použily moji verzi JWT, měli jednoduchý přestup na knihovnu PyJWT ve chvíli, kdy i ona začne nabízet podepisovaní pomocí postkvantových algoritmů. Zdrojový kód implementace JWT najdete v modules/jwt.py.

Součástí mé implementace jsou dvě funkce. První je def encode(payload, key, alg), která vytvoří podepsaný JWT. Parametry této funkce jsou:

* payload – zpráva, kterou chceme odeslat a další atributy)
* key – podpisový klíč, kterým chceme zprávu podepsat
* alg – název podepisovacího algoritmu.

Druhá funkce se jmenuje jwt.decode(jwt, public\_key, alg=None) a slouží k dekódování JWT formátu. Funkce nejprve ověří podpis zprávy a pokud podpis souhlasí, tak JWT dekóduje a vrátí obsah zprávy. Protože JWT hlavička obsahuje informace o použitém podpisovém algoritmu, není parametr „alg“ důležitý a funkce může být zavolána bez něj. Pro dodržení kompatibility s knihovnou PyJWT, jsem možnost volby podpisového algoritmu ve volání funkce zahrnul, tak aby byl upřednostněn parametr volané funkce, před algoritmem uloženým v hlavičce JWT.

[1] <https://datatracker.ietf.org/doc/html/rfc7519>

[2] <https://pyjwt.readthedocs.io/en/stable/>

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 funkce async def get\_sign\_private\_key(my\_address, CA, ALGORITHM), 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 o službě na straně CA.

Obsah obrázku text, diagram, snímek obrazovky, mapa

Popis byl vytvořen automaticky

Obrázek 12 - 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 vytvoří Alici certifikát.
5. CA vezme soukromý podpisový klíč Alice a podepíše ho svým soukromým podpisovým klíčem.
6. CA vezme podepsaná 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

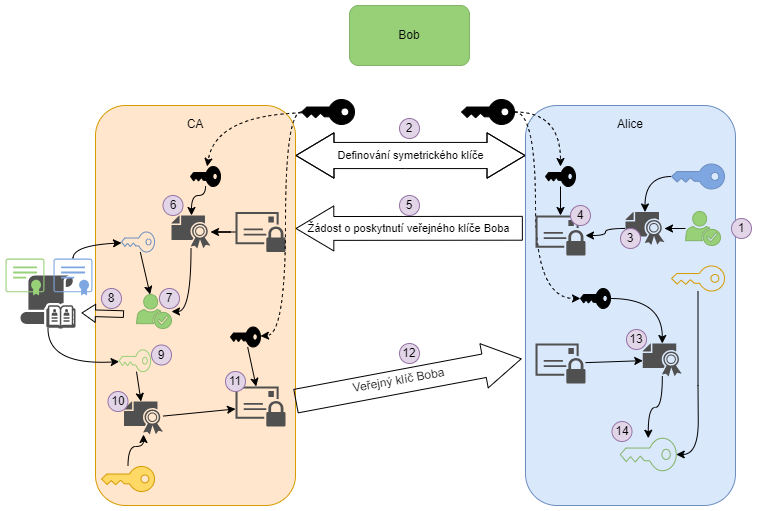
async def ask\_public\_key(subject, sign\_private\_key, my\_address, CA, ALGORITHM), která má 5 parametrů:

* 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

Dostávám se téměř do úplného závěru této kapitoly. Po objasnění všech dílčích kroků se můžeme konečně vrhnout 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:

async def send\_request(ip\_address, port, payload, sign\_private\_key, my\_address, CA, ALGORITHM, uri="").

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 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 14 – 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

Návod na spuštění

Předpoklady:

Python 3.12.2

Git

Microsoft Visual C++ 14.0 nebo novější (https://visualstudio.microsoft.com/visual-cpp-build-tools/)

stáhněte repozitář

git clone -b PQC-service https://github.com/naxit-01/DiplomovaPrace.git

spusťte setup.bat. Tento script vytvoří virtuální prostředí jazyka Python a do něj nainstaluje potřebné knihovny. Pozor, proces trvá přibližně 5 minut.

Při instalaci byly vytvořené tři nové scripty. activate\_alice.bat, activate\_ca.bat, activate\_bob.bat

Spuťte script activate\_ca.bat. Spustí se CA

Spusťte activate\_bob.bat

Na závěr spuťte activate\_alice.bat

Alice začne automaticky odesílat zašifrované zprávy Bobovi.

# Mikroslužba zabezpečující technologii blockchain nad logovými soubory informačního systému.

## Popis jednotlivých částí

Blockchain

Klient/Postman

## Popis funkcionalit nodu

Registrace v síti

Pri spusteni se vytvori objekt blockchain tridy Blockchain. Tento objekt obsahuje jak samotny retez a dalsi pomocne promenne nezbytne pro provoz, tak i radu obsluznych funkci mj. tezba, pridavani bloku, overovani retezu. Pred samotnym spustenim serveru (vyuziva se Python Tornado) se vezme z konfiguracniho souboru informace, na kterem portu a adrese ma byt server spusten a na jake adrese se nachazi nejblizsi fungujici node. Pokud zadny neni, nastavi se pro sousedni port stejna adresa, jako pro nas node. Tzn bude sam sobe sousedem. Neni to problem, protoze tuto vlastnost vyuzije jen pri svem spusteni a pote uz ji nikdy nevyuzije. Opet se tim potvrzuje pravidlo: nikdo neni nad ostatnimi. Jednu vterinu po spusteni serveru se node zaregistruje do site. To probehne tim zpusobem, ze na sousedni node odesle informace o sobe.

Soused si zapise udaje o novem nodu do sve node\_table a jeji aktualizovanou verzi odesle vsem nodum v siti. Timto zpusobem se i novy node dozvi informace o ostatnich nodech v siti. V tomto momente uz zahazuje informaci o jeho primem sousedovi, protoze nadale bude se vsemi komunikovat na stejne urovni.

Tabulka všech nodů

/nodes/get\_nodetable

GET pozadavek, ktery vrati seznam vsech nodu v blockchainu. Sit je nastavena tak, ze kazdy node ma informaci o vsech ostatnich. Se vsemi komunikuje.

Přijetí zprávy

/logs/new

{

"message":"zprava od clienta",

}

POST pozadavek, obsahuje JSON zpravy. Zprava se automaticky ulozi do zasobniku pro dalsi blok. Zaroven se rozdistribuje na vsechny nody v siti. Drzi se pravidlo, vsichni maji prehled o vsem a nikdo neni vys, nez ostatni.

Jedna z velmi typickych situaci, ktera se deje je ta, ze klient blockchainove site bude chtit do ni neco zapsat. Pozadavek odesle skrz endpoint popsany vyse na jeden z nodu v siti. Ten zkontroluje, zda zprava obsahuje vsechna potrebna data a zda ji uz nedostal drive. Pokud ne, odpovida pozitivne zpatky klientovi. Zpravu si uklada k sobe do zasobniku blockchain.current\_logs a nasledne zpravu, tak jak ji dostal, distribuje vsem nodum v siti. V pripade, ze zpravu jiz v zasobniku mel, neprovadi zadnou akci. Je zrejme, ze pokud zpravu uz predtim mel, tak ji take urcite uz predtim rozeslal. Proto nevykona nic a klientovi odpovi, ze zpravu neprijal.

Těžba

/mine/start

GET pozadavek. Impuls pro celou sit, ze ma zacit tezit dukaz pro overeni soucasneho bloku.

Jedna z nejvice komplexnich funkcionalit. Node v teto fazi dela nekolik veci a mnohdy najednou. 1. Provadi samotnou tezbu 2. instruuje ostatni nody, aby take zacali tezit. 3. Vyhodnocuje svoji tezbu a zaroven prijima od ostatnich nodu jejich vytezene bloky (to se deje temer jen v pripade, kdy slozitost dukazu je prilis nizka a jeho casova narocnost je mensi nez prodleni pri komunikaci po siti, realnem nasazeni se takova situace stane ojedinele). 4. Porovnava vsechny kandidaty vytezenych bloku a podle nastavenych pravidel vybira jeden jediny, ktery si uklada do retezu.

Takze postupne:

Po zavolani endpointu /mine/start node nejprve overi, zda by nedoslo ke kolizi v tezbe. Node nemuze tezit blok, pokud jeste nebyl dotezen blok predesly. Nefungovala by navaznost hashu. Pokud se tak nedeje, obesila vsechny nody v siti s prikazem "zacni tezit".

On sam zacne skladat dohromady blok zprav, ke kterym prilozi hlavicku bloku. Hlavicka obsahuje index bloku, casovou znacku zacatku tezby bloku, dukaz(prozatim nastaveny na nulovou hodnotu) a hash z predesleho bloku retezu. Tento zabaleny blok odesila do funkce pro tezbu.

Tato funkce bezi na nove vytvorenem vlaknu tak, aby neovlivnovala chod nodu. Doba tezby se odviji od narocnosti dukazu a ta byla nastavena v konfiguracnim souboru. Tezba je prirovnani pro hadani spravneho dukazu. Dukaz je cislo, ktere kdyz vlozite na stanovene misto do bloku, jeho hash bude splnovat narocnostni podminku. Napriklad ze bude zacinat ctyri nulami. Pokud bychom chteli vetsi narocnost nastavime podminku na vice pocatecnich nul. Nejde o to, ze by to nutne museli byt nuly, byt je to tak u vetsiny blockchainu, ale jde o to, ze to jsou konkretni hodnoty a to nekorespenduje s vlastnosti hashe "neodhadnutelneho vysledku". Tedy jedinym moznym zpusobem jak konkretnich hodnot dosahnout je menit vstupni data a pozorovat zda jsme ziskali pozadovany hash.

Ukonceni tezby muze mit dva duvody 1. On sam vytezil dukaz, nebo 2. Prisla mu zprava od jineho nodu, ze on byl uspesnejsi a uz nema cenu dal tezit. V takovem pripade ukoncuje vlakno pro tezbu a prijima blok od druheho nodu.

Pri nizke narocnosti dukazu a lokalni siti je rozdil mezi vytezenim vlastniho bloku a prijeti zpravy, nekolik malo desitek milisekund. Proto se stava, ze node ma k dispozici jak vlastni vytezeny blok, tak i jeden nebo vice bloku od sousedu. Vsechny jsou platne, ale vsechny jsou zaroven jine. Kazdy z nich obsahuje jinou casovou znacku zacatku tezby a v kazdem z nich je jiny udaj o autorovi uspesne tezby. To znamena ze kazdy ma take jiny hash a i kdyz by vsechny mohli byt pripojeny jako dalsi clanek k retezu, pro zachovani konsenzu musi byt vybran jen jeden. Jako kriterium, pro vyber takoveho kandidata je casova znacka ukonceni tezby. Tato znacka neni primo v samotnem bloku (tezim nad celym blokem, nemohu na konci pridat casovou znacku a tim kompletne ponicit kontrolni hash. Takovy blok by nebyl platnym), ale je pripojena vedle neho a node ji odesila po dokonceni tezby. Tato znacka dosahuje presnosti az jednotky milisekund a proto je pro takove kriterium vhodna. Po vybrani vhodneho kandidata se blok pripoji do retezu a node je pripraven na pokyn k dalsi tezbe.

Ověření řetězu

/chain/resolve

TODO: Impuls pro celou sit. Nody si porovnaji svoje retezy a ten ktery ziska vice jak 51% se stane jedinym prezivsim

Tento endpoint bude volan pokazde kdy se pripoji novy node k siti, a to proto aby nemusel spolehat na spravnost retezu sveho souseda, ale primo dostal vsemi potvrzeny vzorek.

Ve chvili kdy je potreba zkontrolovat spravnost retezu zavola se funkce valid\_chain() ve tride Blockchain. Tato funkce projede blok po bloku a kontroluje dve podminky. 1. Je v kazdem bloku spravna hodnota dukazu? 2. Odpovida predchozi hash v bloku skutecne hashi predesleho bloku? Pokud jsou obe kriteria naplnena pro kazdy blok v retezu, tak je retez povazovan za validni a muze s nim byt dal zachazeno. Napriklad vzit ho pro vytahnuti zprav pro uzivatelske pouziti, nebo pro rozhodovani o hlavnim retezu v siti.

Donutim vsechny prestat tezit. Zahodit praci (je to random je to jedno) Pokud nejaky node dostane informaci pozdeji nez nekdo jiny, nevadi. Proste posle vsem svuj block a jede se dal. Vsem poslu zpravu prestan tezit a od vsech pockam na odpoved "prestal jsem" tim si i overim ze ubehl dostatecny cas na to aby node ktery vytezi block ho poslal ostatnim.

Poskytnutí řetezu

/chain

GET pozadavek. Po zavolani node provede kontrolu spravnosti celeho retezu. Zkontroluje jestli bloky (jejich hashe) na sebe spravne navazuji a zaroven zkontroluje jestli ma kazdy blok spravne vypocitanou hash

Průvodce pro spuštění mikroslužby

Návod na spuštění

Předpoklady:

Python 3.12.2

stáhněte repozitář

git clone -b Blockchain-service https://github.com/naxit-01/DiplomovaPrace.git

spusťte setup.bat. Tento script vytvoří virtuální prostředí jazyka Python a do něj nainstaluje potřebné knihovny. Po dokončení stiskněte libovolnou klávesu.

Při instalaci byly vytvořené dva nové scripty. activate\_client.bat, activate\_blockchain.bat

Alespoň 3x spusťte activate\_blockchain.bat

Na závěr spuťte activate\_client.bat

Celou síť ovládáte pomocí příkazového okna clienta.

# Postkvantová blockchainová síť

Popis jednotlivých částí

Agent

Node

Ukázka funkcionalit mikroslužeb

Průvodce pro spuštění Postkvantové blockchainové sítě

# Možnosti dalšího rozvoje

# ZÁVĚR

Je nutné věnovat stejnou, ne-li větší pozornost jako úvodu. V závěru musíte podat shrnutí, vyzvednout nejdůležitější postřehy a také vyhodnotit, zdali jste naplnili cíle, které jste si stanovili v úvodu své práce.

# SEZNAM POUŽITÉ LITERATURY (vzor)

**MONOGRAFIE**

1. ANTUŠÁK, Emil. *Strategie a ekonomika v bezpečnostním systému České republiky*. 1. vyd. Praha: Ministerstvo obrany České republiky, 2002. s. 141 – 180. ISBN 80-7278-143-x.
2. BRČÁK, Josef. *Česká republika ve světle ekonomických teorií*. 1. vyd. Plzeň: Vydavatelství a nakladatelství Aleš Čeněk, 2012. s. 78 – 150. ISBN 978-80-7380-369-8.
3. KADEŘÁBKOVÁ, Anna. *Základy makroekonomické analýzy: růst, konkurenceschopnost, rovnováha*. 1. vyd. Praha: Linde, 2003. 175 s. ISBN 80-86131-36-x.

**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)˃.

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

# 

# SEZNAM PŘÍLOH

|  |  |  |  |
| --- | --- | --- | --- |
| Příloha č. 1 | − | Popis přílohy (→ styl Odstavec) |  |
| Příloha č. 2 | − | Popis přílohy (→ styl Odstavec) |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

(→ Styl Příloha – číslování) Příloha 1

Popis přílohy (→ styl Příloha – popis)

|  |
| --- |
|  |