**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 |
| KEM | Key encapsulation mechanism |
| 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

Zkopíroaný článek: https://www.cryptoquantique.com/blog/post-quantum-cryptography/

Úvod do postkvantové kryptografie

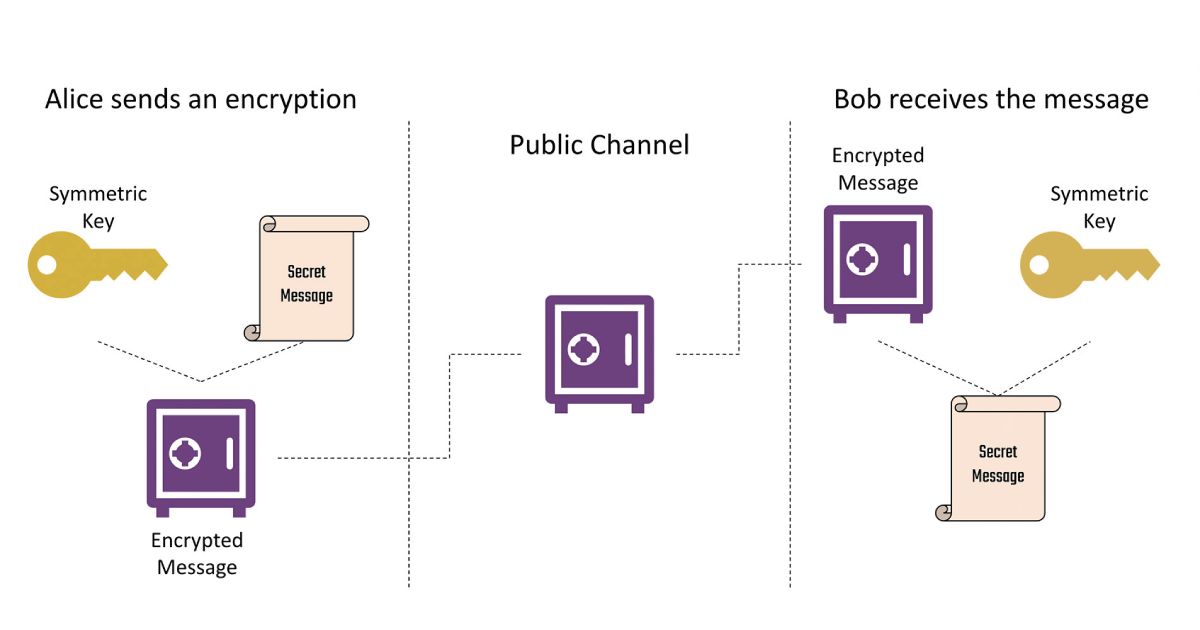
Pohled na kryptografii v budoucím postkvantovém bezpečnostním prostředí IoT

Je to vzrušující čas pracovat na kvantových technologiích; konečně, kvantové počítače jsou téměř tady. Jeho zastánci již nějakou dobu s jistotou tvrdili, že první komerčně dostupné kvantové počítače budou vyrobeny během příštích dvaceti let, ale úspěchy, jako jsou úspěchy týmů IBM, DWave a naposledy Google, naznačují, že tentokrát tato víra může být připisován více než jen optimismu a prodejním názorům. Ačkoli je příliš brzy na pochopení důsledků zařízení skutečně schopného využívat kvantové efekty, je nepochybné, že uvidíme pokroky v oblastech, kde lze správně využít sílu kvantového počítače k ​​pochopení vzorců ve velkých svazcích dat, jako je např. jako datová věda.

Bohužel ne každý může sledovat příchod kvantových počítačů jen s nadšením. Kryptografové, praktici zodpovědní za ochranu dat a soukromí lidí na internetu i mimo něj, se připravují na to, že desítky let dobře vyladěných technik skrývání dat budou zrušeny první vlnou sofistikovaných kvantových strojů. Naštěstí je zde rozhodující slovo příprava. Algoritmus vyvinutý Peterem Shorem v polovině 90. let ukázal, že kvantové počítače budou schopny vyřešit slavný problém „faktorizace celého čísla“, bezpečnostní páteř slavného kryptosystému RSA, dramaticky rychleji, než to dokážou naše moderní počítače. Ještě horší je, že hlavní stavební blok Shorova algoritmu, kvantová Fourierova transformace, lze upravit tak, aby prolomila většinu ostatních běžně implementovaných kryptografií, jako je výměna klíčů Diffie-Helman a digitální podpisy elliptické křivky. Otázka tedy zní: co dělali kryptografové za posledních dvacet let, aby se na tuto eventualitu připravili? Stručně řečeno, vyvinuli předmět Post-Quantum Cryptography (PQC), studium kryptografie, kterou lze provozovat na klasických počítačích a přitom si zachovat odolnost vůči kvantovým útokům.​

Kryptografie 101

Než se dostaneme k postkvantové kryptografii, měli bychom nejprve porozumět některým základům kryptografie a tomu, co pro kvantový počítač znamená, že ji rozbije. Tradičně se kryptografové snažili vyřešit varianty následujícího problému: Alice chce poslat zprávu – „Schoval jsem poklad v knihovně“ – svému příteli Bobovi, ale nemá žádný soukromý kanál, přes který by tuto zprávu poslala. Navíc se obává, že by se Eva mohla pokusit zachytit její tajnou zprávu a ukrást poklad. Co tedy Alice umí? Tady přichází na scénu šifrování. Řekněme, že když byli Alice a Bob spolu naposledy, věděli, že by si mohli chtít posílat zprávy poté, co Alice ukryla poklad. Dohodli se tedy na klíči, jak to udělat tajně. Nyní, když chce Alice poslat zprávu Bobovi, použije tento klíč k zašifrování své zprávy (představte si, že zprávu vloží do zamčené schránky), a jakmile dorazí s Bobem, on pouze dešifruje pomocí stejného klíče (odemknutí krabice). Pokud se Evě podaří zachytit jejich komunikaci, nebude schopna odemknout schránku, protože nemá klíč. Protože Alice a Bob sdílejí stejný klíč, ten, na kterém se shodli dříve, nazýváme to symetrická kryptografie.

*Symetrická kryptografie v akci*

Matematicky šifrování zprávy představuje použití klíče ke zkomolení Aliciny zprávy do řetězce nesmyslů – „vrbayebcjaelaiyurbjeaby!“ – to lze rozbalit do Aliciny zprávy pouze pomocí jejího a Bobova klíče. Evě, která zprávu úspěšně zachytila, ale postrádala klíč, zjistí, že nedokáže pochopit, co Alice poslala .

S tímto nastavením je samozřejmě zřejmý problém. Co když Alice a Bob potřebují vést tajný rozhovor, aniž by se mohli sejít a předem se dohodnout na klíči? Zde se setkáváme s více zapojenými technikami kryptografie s veřejným klíčem. Pro nastavení výměny si Bob vytvoří pár klíčů: veřejný klíč a soukromý klíč. Veřejný klíč, který posílá do světa, a lze si ho představit jako šifrovací klíč pro posílání zpráv Bobovi. Když chce Alice poslat zprávu Bobovi, vyhledá jeho veřejný klíč a použije jej k zašifrování své zprávy. Vzhledem k tomu, že Eve má přístup pouze k veřejnému šifrovacímu klíči, pokud se jí podaří zachytit Alicinu zprávu, stejně uvidí jen nesmysly. Jakmile Bob zprávu přijme, nyní použije svůj soukromý klíč, v podstatě dešifrovací klíč, k rozbalení a přečtení Aliciny komunikace .

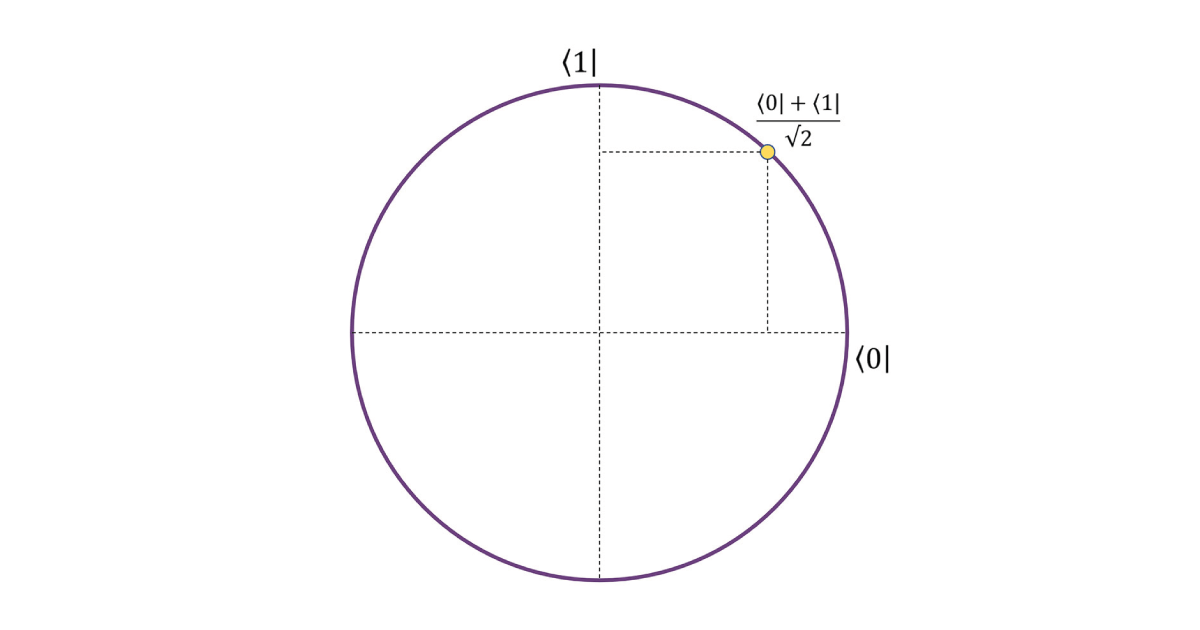
V praxi je nepravděpodobné, že by se strany dokázaly dohodnout na klíči předem – zamyslete se nad tím, jak často musíte zadávat své bankovní údaje, abyste mohli nakupovat na webové stránce, kterou jste nikdy předtím nenavštívili. V důsledku toho jsou bezpečné techniky pro kryptografii veřejného klíče pro bezpečnost internetu nanejvýš důležité. Bohužel se ukazuje, že vytváření schémat s veřejnými klíči je docela obtížné a obvykle to skončí velmi pomalu, když chcete poslat spoustu informací. Naštěstí to jde snadno opravit. Alice a Bob začínají pomocí schématu veřejného klíče nebo podobné techniky známé jako protokol výměny klíčů, aby tajně sdělili jednu malou zprávu, která je pouze sdíleným klíčem. Poté použijí tento sdílený klíč s dobře zavedeným symetrickým schématem, typicky schématem nazývaným  [AES](https://translate.google.com/website?sl=auto&tl=cs&hl=cs&client=webapp&u=https://en.wikipedia.org/wiki/Advanced_Encryption_Standard) , aby pokračovali ve své tajné komunikaci bez režie, kterou vyžadují techniky veřejného klíče. Takto běžné internetové standardy, jako je  [TLS,](https://translate.google.com/website?sl=auto&tl=cs&hl=cs&client=webapp&u=https://en.wikipedia.org/wiki/Transport_Layer_Security)  navrhují, aby byla v praxi zabezpečena komunikace.

Robustní symetrická kryptografie ve formě AES existuje již od 90. let a při správné implementaci je považována za extrémně bezpečnou. Potenciální slabé místo v tomto nastavení tedy spočívá ve fázi veřejného klíče. Jak bezpečné je schéma veřejného klíče? Je známé, že bezpečnost kryptografie s veřejným klíčem, obtížnost prolomení schématu, je možné založit na výpočetních matematických problémech. Nejznámějším příkladem je samozřejmě schéma RSA. V RSA je Bobův veřejný klíč nějaké velké celé číslo n a jeho tajný klíč je nějaká další informace o n, konkrétně to, že jej lze faktorizovat jako součin dvou prvočísel p a q, takže pq = n. Aby poslala zprávu Bobovi, může ji Alice zašifrovat pomocí n takovým způsobem, že k dešifrování zprávy potřebujeme p a q. Pro Boba je to snadné, protože tato prvočísla zná, ale zachycující Eva je musí být schopna získat zpět faktorizací veřejného klíče n. Úkolem Evy je tedy právě problém „rozkladu celého čísla“ řešený Shorovým algoritmem, který nelze na dnešních počítačích efektivně vyřešit za předpokladu, že p a q jsou dostatečně velké. Jiné běžné techniky výměny veřejných klíčů a klíčů odvozují svou bezpečnost z různých těžkých matematických problémů, například problém diskrétního logaritmu, ve stejném smyslu – schéma se ukázalo být stejně těžké prolomit, jako je těžké problém vyřešit .

Jedna věc, které jsme se ještě nedotkli, je problém autentizace: když Bob obdrží zprávu od Alice, jak si může být jistý, že ji skutečně poslala Alice a že to bylo skutečně to, co Alice chtěla říct? Koneckonců, když je Eva schopná zachytit zprávy, mohla by být také schopná s nimi manipulovat, než se dostanou k Bobovi? Zejména v době internetové komunikace může být těžké věřit, že každá zpráva přišla od toho, od koho se vydává. Abychom to vyřešili, spoléháme na příhodně pojmenované téma digitálních podpisů, které fungují podobně jako jejich skutečný světový jmenovec. Alice používá soukromý klíč, svou moc podepisovat zprávy, k připojení podpisu ke zprávě Bobovi, který říká Bobovi, že to byla Alice, kdo poslal tuto zprávu. Podpis navíc shrnuje zamýšlený obsah takovým způsobem, že pokud je s obsahem manipulováno, podpis by se jevil jako nesprávný. Po obdržení této zprávy se Bob může podívat na Alicin odpovídající veřejný klíč, který mu říká, jak vypadá podpis Alice, aby si ověřil, že Eva nezasáhla do zamýšlené zprávy a že pochází od Alice. Podobně jako u šifrování i zde kryptografické techniky spoléhají na komplikovanou matematiku, kde obtížnost padělání Alicina podpisu odpovídá náročnosti řešení výpočetního problému. Například k padělání podpisu v běžně používaném podpisovém schématu známém jako ECDSA musí být Eva schopna vyřešit problém diskrétního logaritmu na něčem známém jako eliptická křivka .

Kvantové počítání a těžké problémy

Přestože úplné vysvětlení mechanismů kvantového počítače by samo o sobě zaplnilo blogový příspěvek, stručné shrnutí nám pomůže pochopit myšlenky, které stojí za vytvořením kvantově odolné kryptografie. Pamatujte, že pro vytvoření postkvantových schémat veřejného klíče hledáme matematické problémy, které ani kvantový počítač nedokáže rozluštit. Kvantové počítače jsou hodně podobné klasickým počítačům, až na to, že nahrazují hradla na klasických bitech, objektu, který je buď ve stavu 0 nebo 1, operacemi na kvantové variantě zvané qubit. Qubit představuje bod na jednotkové kružnici, kružnici o poloměru 1 a kvantová brána představuje mapování z bodu na jednotkové kružnici do jiného. Záludné je, že když se pokusíte přečíst qubit, neřekne vám, kde leží na jednotkovém kruhu. Místo toho se náhodně zhroutí do stavu 0 nebo 1, s pravděpodobností 0 nebo 1 v závislosti na jeho poloze na kruhu.

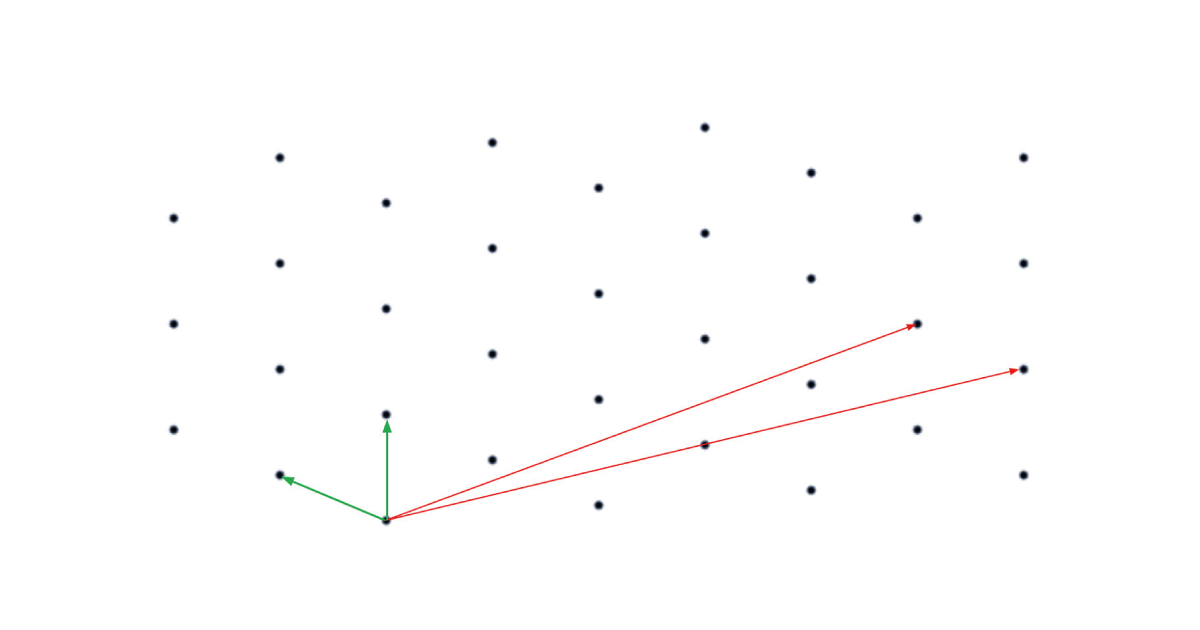
*Qubit na jednotkovém kruhu*

Díky své komplikovanější povaze jsou qubity schopny reprezentovat a uchovávat mnohem více informací než běžný bit. To neznamená, že jsou lepší ve všech úkolech a výzkumníci začínají chápat, pro jaké úkoly jsou vhodnější. Jako užitečný kus intuice si lze představit kvantový počítač, který dokáže velmi levně „paralelizovat“ úkoly. To znamená, že pokud lze úkol rozdělit na několik menších dílčích úkolů, které lze provést současně, pak je kvantové zařízení zběhlé v provádění každého z těchto menších úkolů najednou. Oproti klasickému zařízení, které musí provádět každý úkol sekvenčně, si kvantový počítač ušetří spoustu času. Vezměme si například prvočíselný faktoring: kvantový počítač by mohl pouze rozdělit možné faktory do skupin a pak paralelně kontrolovat, zda se některá čísla v každé skupině dělí , spíše než muset začínat od 1 a kontrolovat každé číslo postupně. Porovnejte to s úkoly, které vyžadují neustálou aktualizaci stejného objektu – to jsou úkoly, kde kvantové počítače nemusí fungovat o mnoho lépe než klasické počítače. Ve skutečnosti se bezpečnost běžných symetrických schémat, jako je AES, spoléhá na „smíšení“ objektů dohromady, takže k prolomení šifrování zprávy musíte ručně provést řadu kroků rozpojení. Tato myšlenka vysvětluje, proč se PQC zaměřuje především na aktualizaci kryptografie s veřejným klíčem – kvantové počítače nejsou dramaticky lepší v odstraňování symetrického šifrování než klasické počítače, takže většině symetrických konstrukcí stačí jen mírně změnit jejich parametry, aby byly stále považovány za bezpečné.

Kvantově těžké problémy

Vyzbrojeni těmito znalostmi o tom, co ztěžuje problém kvantovému počítači, se můžeme rychle podívat na hlavní kandidáty na výpočetní problémy v PQC. Existuje nějaký výzkum v modré obloze, ale většina navrhovaných schémat PQC leží v jedné ze čtyř hlavních kategorií: mřížková, kódově založená, multivariační a izogenní kryptografie.

Na prvním místě je nejoblíbenější kategorie, mřížková kryptografie. Mřížku lze chápat jako pravidelnou mřížku bodů v prostoru, kde body na mřížce jsou vybírány systematicky z objektu zvaného její základna, který popisuje mřížku tím, že vysvětluje, jak se pohybujete mezi mřížkovými body.

*Mříž se dvěma různými základnami*

Mříže jsou klasické matematické objekty a výpočetně náročné problémy na nich byly studovány již dávno předtím, než existoval kryptosystém RSA, což dává jistou důvěru představě, že tyto problémy by měly zůstat těžké i tváří v tvář kvantovým útokům. Archetypální problém tvrdé mřížky je nejkratší vektorový problém: po obdržení základu pro mřížku je řešitel požádán, aby našel bod mřížky nejblíže k počátku. Ačkoli je to snadné pro malé mřížky, jako je ta znázorněná výše, když je mříž objektem ve stovkách rozměrů, tento problém se stává velmi náročným. Páteř budování kryptografie z problémů s mřížkou pochází ze zajímavého pozorování o definování mřížky ze základu. Základ použitý ke konstrukci mřížky není jedinečný a je možné mít „dobrý“ a „špatný“ základ stejné mřížky. Například ve výše uvedeném diagramu lze všech bodů mřížky dosáhnout tak, že začneme na 0 a pak půjdeme po kombinacích buď červených nebo zelených čar, nazývaných vektory. Pokud máte zelený základ, je snadné rychle porozumět tvaru mřížky, protože vaše vektory jsou krátké a navzájem téměř 90 stupňů. Jakmile dobře porozumíte mřížce, můžete jen odečíst nejkratší vektor. A naopak, pomocí červených čar musíte hodně kroutit, abyste našli body blízké 0, a i když už nějaký najdete, jak byste ověřili, že tam není nic kratšího? Kryptosystémy založené na mřížkách využívají tuto myšlenku k budování kryptografie – na vysoké úrovni je myšlenkou použít špatný základ pro mřížku jako veřejný klíč a dobrý základ pro stejnou mřížku jako soukromý klíč. Špatný základ popisuje mřížku dostatečně dobře, aby skryl zprávu jako těžký problém na mřížce. Pak může tento problém snadno vyřešit pouze zamýšlený příjemce, protože to vyžaduje přístup k dobrému základu. Mřížková kryptografie má tendenci být nejflexibilnější kategorií PQC, protože existuje mnoho prostoru pro návrh, se kterým lze pracovat, pokud jde o to, jak si vybrat mřížky a skrýt body. Tato flexibilita vedla k úspěšné aplikaci mřížek na problémy šifrování i digitálních podpisů .

Kryptografie založená na kódu je nejstarší ze čtyř hlavních rodin, datuje se od vývoje kryptosystému  [McEliece](https://translate.google.com/website?sl=auto&tl=cs&hl=cs&client=webapp&u=https://en.wikipedia.org/wiki/McEliece_cryptosystem)  v roce 1978. Je založena na oblasti teorie kódování, která je studiem toho, jak úspěšně přenášet informace přes nespolehlivý kanál. K odesílání informací tímto způsobem používáte něco, čemu se říká kód pro opravu chyb, kde je zpráva zakódována způsobem, který toleruje určité množství chybných dat pocházejících z nespolehlivého kanálu, přičemž příjemce je stále schopen obnovit zamýšlenou zprávu. . Soukromý klíč pro kryptosystém založený na tomto je jen dobře zvolený kód pro opravu chyb a odpovídající veřejný klíč je recept na zakódování zprávy, která má být dekódována. Šifrování je jednoduché – odesílatel provede obvyklý proces kódování zprávy a poté simuluje nespolehlivý kanál ručním zadáním některých chyb. Dokud existuje dostatek chyb, protivník bez přístupu k popisu kódu pro opravu chyb nemůže zprávu obnovit, zatímco zamýšlený příjemce je schopen ji dekódovat pomocí svého kódu pro opravu chyb. Pěkné na těchto schématech je, že McEliece existuje již dlouhou dobu, a proto existuje velká míra důvěry v jejich bezpečnost a schémata jemu podobná. Na druhou stranu, McEliece existuje už dlouho a nikdo ho nepoužívá – má velmi velké veřejné klíče, což může trochu zpomalit, a obecně se špatně srovnává s RSA v předkvantové éře .

Vícerozměrná kryptografie je založena na problému řešení systémů mnohorozměrných polynomiálních rovnic, typicky vícerozměrných kvadratických. Tedy soustava rovnic, která vypadá takto:

Ačkoli mnoho existujících forem kryptografie eliptických křivek, jako je ECDSA, je citlivých na kvantové počítače, je to důsledek specifické povahy algoritmů a nikoli slabina samotných eliptických křivek. Abychom kryptografii založili na izogeniích, použijeme následující pozorování: vzhledem ke dvěma eliptickým křivkám E1 a E2 je těžké najít izogenii φ, která posílá E1 na E2. Veřejný klíč by tedy mohl být pár eliptických křivek, se soukromým klíčem odpovídající izogeneze φ: E1 -> E2. Nyní může vlastník soukromého klíče použít izogenii φ k výpočtu operací zahrnujících E1 a E2, které jsou neovladatelné pro někoho, kdo nemá přístup k φ, a jako obvykle lze k nastavení těchto problémů použít veřejný pár E1 a E2. Přestože je oblast kryptografie na bázi izogeny stále v relativních plenkách, v kryptografické komunitě je o ní značné vzrušení, zejména po  [úspěšných experimentech](https://translate.google.com/website?sl=auto&tl=cs&hl=cs&client=webapp&u=https://www.imperialviolet.org/2019/10/30/pqsivssl.html)  s využitím schémat založených na izogenii v internetových protokolech. Navíc působivě efektivní schéma podpisů nazvané SQISign, založené na novém problému v kryptografii založené na izogenii, získalo na letošní konferenci Asiacrypt cenu za nejlepší knihu .

Směrem ke standardizaci: Proces NIST

Vzhledem ke všem těmto konkurenčním nápadům může být těžké pochopit, kde leží budoucnost kryptografie. Jedním z míst, kde se můžete obrátit s žádostí o radu, je proces postkvantové standardizace, který provozuje Národní institut pro standardy a technologie (NIST) ve Spojených státech. NIST již dříve spouštěl procesy pro standardizaci symetrické kryptografie a hašovacích funkcí, což vedlo k široce používaným primitivům AES a SHA3. Ačkoli se nejedná o oficiální globální standard, je pravděpodobné, že jakýkoli algoritmus podporovaný NIST bude mít úrovně absorpce podobné těm u AES a SHA3, takže zkoumání tohoto procesu by mělo poskytnout nejmodernější pochopení pravděpodobné budoucnosti PQC. . NIST se snaží objasnit, že by to mělo být považováno za proces a ne za soutěž s „vítězem“, takže se mohou rozhodnout standardizovat nebo navrhnout více algoritmů pro různé případy použití, pokud neexistuje jasná nejlepší technika. Dvě hlavní vlastnosti, které NIST hledá, jsou bezpečnost a efektivita: pokud protokol není bezpečný, nemá smysl jej používat k ochraně dat, a pokud není efektivní, zpomaluje internetovou komunikaci. Obecně má PQC tendenci pracovat pomaleji než jeho předkvantový protějšek, takže algoritmus lze považovat za účinný, pokud nedojde k žádnému nebo malému poklesu výkonu při jeho použití namísto stávajícího předkvantového schématu .

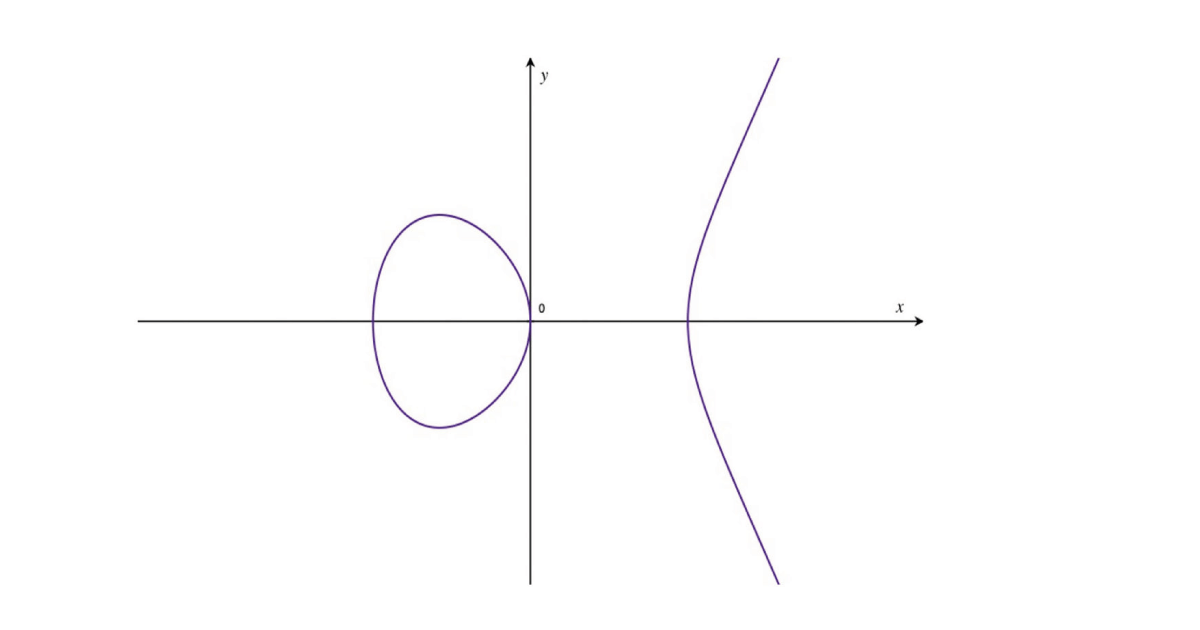
Proces byl zahájen NIST v roce 2016 ve formě otevřené výzvy kryptografické komunitě, která žádala o návrhy mechanismů postkvantového zapouzdření klíčů (v podstatě asymetrická šifrovací schémata určená k odeslání symetrického klíče) a podpisová schémata s nastavením parametrů. vhodné pro různé praktické případy použití. Tato otevřenost je hlavním rysem procesu NIST; existuje  [veřejná google skupina](https://translate.google.com/website?sl=auto&tl=cs&hl=cs&client=webapp&u=https://groups.google.com/a/list.nist.gov/forum/%23!forum/pqc-forum)  obsahující podrobné diskuse o konkrétních kandidátských schématech a také o obecnějších rysech procesu. 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 programů, 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 zapouzdření klíčů a 3 schémata podpisů, se záměrem vybrat na konci procesu alespoň jeden z každého typu. Přestože byla všechna tato schémata důkladně prověřena, každý z nich má jiné silné a slabé stránky, kterým je užitečné porozumět.

Ačkoli mnoho existujících forem kryptografie eliptických křivek, jako je ECDSA, je citlivých na kvantové počítače, je to důsledek specifické povahy algoritmů a nikoli slabina samotných eliptických křivek. Abychom kryptografii založili na izogeniích, použijeme následující pozorování: vzhledem ke dvěma eliptickým křivkám E1 a E2 je těžké najít izogenii φ, která posílá E1 na E2. Veřejný klíč by tedy mohl být pár eliptických křivek, se soukromým klíčem odpovídající izogeneze φ: E1 -> E2. Nyní může vlastník soukromého klíče použít izogenii φ k výpočtu operací zahrnujících E1 a E2, které jsou neovladatelné pro někoho, kdo nemá přístup k φ, a jako obvykle lze k nastavení těchto problémů použít veřejný pár E1 a E2. Přestože je oblast kryptografie na bázi izogeny stále v relativních plenkách, v kryptografické komunitě je o ní značné vzrušení, zejména po  [úspěšných experimentech](https://translate.google.com/website?sl=auto&tl=cs&hl=cs&client=webapp&u=https://www.imperialviolet.org/2019/10/30/pqsivssl.html)  s využitím schémat založených na izogenii v internetových protokolech. Navíc působivě efektivní schéma podpisů nazvané SQISign, založené na novém problému v kryptografii založené na izogenii, získalo na letošní konferenci Asiacrypt cenu za nejlepší knihu .

Směrem ke standardizaci: Proces NIST

Vzhledem ke všem těmto konkurenčním nápadům může být těžké pochopit, kde leží budoucnost kryptografie. Jedním z míst, kde se můžete obrátit s žádostí o radu, je proces postkvantové standardizace, který provozuje Národní institut pro standardy a technologie (NIST) ve Spojených státech. NIST již dříve spouštěl procesy pro standardizaci symetrické kryptografie a hašovacích funkcí, což vedlo k široce používaným primitivům AES a SHA3. Ačkoli se nejedná o oficiální globální standard, je pravděpodobné, že jakýkoli algoritmus podporovaný NIST bude mít úrovně absorpce podobné těm u AES a SHA3, takže zkoumání tohoto procesu by mělo poskytnout nejmodernější pochopení pravděpodobné budoucnosti PQC. . NIST se snaží objasnit, že by to mělo být považováno za proces a ne za soutěž s „vítězem“, takže se mohou rozhodnout standardizovat nebo navrhnout více algoritmů pro různé případy použití, pokud neexistuje jasná nejlepší technika. Dvě hlavní vlastnosti, které NIST hledá, jsou bezpečnost a efektivita: pokud protokol není bezpečný, nemá smysl jej používat k ochraně dat, a pokud není efektivní, zpomaluje internetovou komunikaci. Obecně má PQC tendenci pracovat pomaleji než jeho předkvantový protějšek, takže algoritmus lze považovat za účinný, pokud nedojde k žádnému nebo malému poklesu výkonu při jeho použití namísto stávajícího předkvantového schématu .

Proces byl zahájen NIST v roce 2016 ve formě otevřené výzvy kryptografické komunitě, která žádala o návrhy mechanismů postkvantového zapouzdření klíčů (v podstatě asymetrická šifrovací schémata určená k odeslání symetrického klíče) a podpisová schémata s nastavením parametrů. vhodné pro různé praktické případy použití. Tato otevřenost je hlavním rysem procesu NIST; existuje  [veřejná google skupina](https://translate.google.com/website?sl=auto&tl=cs&hl=cs&client=webapp&u=https://groups.google.com/a/list.nist.gov/forum/%23!forum/pqc-forum)  obsahující podrobné diskuse o konkrétních kandidátských schématech a také o obecnějších rysech procesu. 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 programů, 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 zapouzdření klíčů a 3 schémata podpisů, se záměrem vybrat na konci procesu alespoň jeden z každého typu. Přestože byla všechna tato schémata důkladně prověřena, každý z nich má jiné silné a slabé stránky, kterým je užitečné porozumět.

*Eliptická křivka ve dvou částech*

Zbývající 4 schémata zapouzdření klíče jsou NTRU, klasické schéma založené na mřížce využívající techniky pocházející z 90. let, CRYSTALS-Kyber, modernější schéma mřížky založené na problému zvaném Module LWE, SABRE, ještě modernější schéma založené na mřížce. na „zakulaceném“ problému zvaném Module LWR a Classic McEliece, modernizaci schématu založeného na kódu z konce 70. let. Tato tři mřížková schémata lze považovat za výběr různých kompromisů mezi bezpečností a efektivitou: na jedné straně SABRE funguje lépe než CRYSTALS-Kyber, který zase funguje lépe než NTRU. Naopak NTRU má nejlépe zavedené bezpečnostní vlastnosti, zatímco SABRE se spoléhá na problém modulu LWR, který stále není příliš dobře pochopen, přičemž spoléhání CRYSTALS-Kyber na modul LWE jej opět umisťuje doprostřed. Classic McEliece má horší výkon než jeho konkurenti na bázi mřížky, ale mohl by být standardizován jako bezpečný pro případ selhání v případě, že by se ukázalo, že síla kvantových útoků na mřížková schémata byla značně podceněna; rozmanitost v kryptografických předpokladech může být mocným nástrojem tváří v tvář neznámému. NIST naznačil, že pravděpodobně zpočátku zvolí jeden z NTRU nebo CRYSTALS-Kyber ke standardizaci, ale důkladnější bezpečnostní analýza SABER nebo kvantové útoky na mřížky by mohly změnit jejich směr.

Když obrátíme svou pozornost na podpisová schémata, znovu vidíme, že mřížky jsou pravděpodobnou budoucností PQC. Dva ze tří zbývajících kandidátů používají mřížky: CRYSTALS-Dilithium, podpisové schéma vyvinuté stejným týmem jako CRYSTALS-Kyber s použitím podobných technik, a Falcon, podpisové schéma podobné NTRU s několika novými technickými příspěvky. Další zbývající schéma, Rainbow, je vícerozměrné schéma s horším celkovým výkonem než jeho mřížkové protějšky. Podobně jako v případě Classic McEliece se NIST domnívá, že je přínosné zahrnout kandidáty bez mřížky, a relativní mládí Rainbow jako schématu znamená, že je možné, že by mohl být v blízké budoucnosti vylepšen, nebo by mohl vynikat v konkrétních případech použití. . NIST opět navrhl, že pravděpodobně standardizují jedno ze dvou mřížkových schémat: Falcon je pravděpodobně o něco efektivnější, ale přichází s podstatnými technickými komplikacemi při implementaci, které mohou vést k tomu, že je považován za zbytečně komplikovaný ve srovnání s elegantnějším designem. CRYSTALS-Dilithium, který si stále vede velmi dobře.

NIST také vybírá 8 schémat jako „alternativní“, schémata, která by podle jejich názoru měla být dále studována, protože NIST jsou otevřeny možnosti jejich standardizace v budoucnu. Vybrané alternativy mají kombinaci žádoucích vlastností a pozoruhodných nedostatků: mohou být například vysoce bezpečné, ale dosti neefektivní, vyžadují větší racionalizaci nebo postrádají potřebnou úroveň vystavení odhodlaným útočníkům, což je typické při vytváření důvěry v kryptografické zabezpečení. Z pěti klíčových alternativ zapouzdření lze čtyři nejlépe pochopit ve srovnání s finalisty: FrodoKEM lze považovat za konzervativnější verzi CRYSTALS-Kyber, která běží pomaleji, ale je bezpečnější, a NTRU Prime je schéma podobné NTRU, které funguje předpokládaná slabost v jiných technikách mřížky a malé náklady na efektivitu. Bike a HQC jsou dvě moderní šifrovací schémata založená na kódu, která fungují efektivněji než Classic McEliece, ale zavádějí nové bezpečnostní předpoklady, a proto potřebují podstatně větší kontrolu. Páté schéma, SIKE, je jediným zbývajícím kryptosystémem založeným na izogenii v procesu. NIST se domníval, že SIKE není dostatečně vyspělý, aby mohl být standardizován, ale jeho jedinečnost znamená, že je to kandidát, který pravděpodobně podstoupí seriózní výzkumné úsilí v budoucnu. Co se týče alternativních podpisů, pouze jeden má nějakou podobnost s finalisty: schéma známé jako GeMSS, které úzce konkuruje Rainbow, ale očekává se, že bude ve většině aplikací méně žádoucí. Další dvě alternativy, SPHINCS+ a Picnic, jsou založeny na hashovacích funkcích. Oba běží dost pomalu, ale mají extrémně silné bezpečnostní záruky, takže mohou být standardizovány pro použití s ​​vysoce citlivými daty nebo v případě kvantového „Armageddonu“, kde se ukazuje, že kvantové počítače mohou prolomit kryptografii, která používá cokoli složitějšího než jednoduchá primitiva, jako je hash. funkce .

Výhled

Kvantové počítače přicházejí a každý, kdo má zájem o adekvátní zabezpečení svých dat, se musí začít dívat do budoucnosti již nyní. Pro zvědavého pozorovatele bude sledování procesu NIST stále aktuální. Od nynějška až do doby, kdy si NIST vybere své standardy, což očekávají, že to bude trvat přibližně 12–18 měsíců, bude 7 finalistů podrobeno enormní úrovni zkoušek a časem bude pravděpodobně jasné, od kterých kandidátů se očekává, že projdou. Pro aktivnější lidi tam nikdy nebyl lepší čas zvážit, jaké algoritmy nejlépe podporují vaše případy použití. Všechna schémata zbývající v procesu NIST mají jedinečné prodejní body, ale počáteční standardy se pravděpodobně zaměří na schémata, která jsou nejlépe vyvážená. Pokud například pracujete s daty, která jsou obzvláště citlivá, může být vhodné zapojit se do diskusí o konzervativnějších schématech. Naopak, pokud potřebujete zabezpečit zařízení, která nemusí mít přístup k velkému výpočetnímu výkonu, jako jsou zařízení IoT, možná budete muset zvážit, která schémata lze provozovat ve zvláště nenáročných nastaveních; možná dobrý námět na další blogový příspěvek!

## 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

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

Blockchain je technologie, která umožňuje ukládat a přenášet data v decentralizované a zabezpečené formě. Blockchain se skládá z řetězce bloků, které obsahují informace o transakcích, smlouvách nebo jiných událostech. Každý blok je spojen s předchozím pomocí kryptografického otisku, který zaručuje jeho jedinečnost a neměnnost. Blockchain je tedy jakousi digitální knihou účetnictví, kterou nelze podvrhnout ani změnit.

Blockchain má několik výhod oproti tradičním způsobům ukládání a přenosu dat. Jednou z nich je decentralizace, což znamená, že data nejsou uložena na jednom centrálním serveru, ale na mnoha počítačích rozprostřených po celém světě. Tyto počítače se nazývají uzly a tvoří síť blockchainu. Každý uzel má kopii celého blockchainu a ověřuje pravost nových bloků pomocí konsensu. Konsensus je proces, kterým se uzly dohodnou na platnosti nových bloků podle určitých pravidel. Pokud se většina uzlů shodne na tom, že nový blok je validní, je přidán do blockchainu a distribuován mezi ostatní uzly.

Další výhodou blockchainu je bezpečnost. Díky kryptografii a konsensu je téměř nemožné změnit nebo smazat data z blockchainu. Pokud by někdo chtěl podvrhnout nebo napadnout blockchain, musel by ovládnout více než polovinu uzlů sítě a změnit historii transakcí na všech z nich. To by bylo velmi obtížné a nákladné. Navíc by takový útok byl snadno odhalitelný a odmítnutý ostatními uzly.

Blockchain také nabízí transparentnost a důvěru. Každý, kdo má přístup k síti blockchainu, může vidět historii všech transakcí a ověřit jejich platnost. To eliminuje potřebu důvěryhodných třetích stran, jako jsou banky, úřady nebo soudy, které obvykle zprostředkovávají transakce mezi lidmi nebo organizacemi. Blockchain umožňuje přímou a rychlou výměnu hodnot bez poplatků nebo zpoždění.

Blockchain má mnoho potenciálních aplikací v různých oblastech, jako jsou finance, logistika, energetika, zdravotnictví nebo vzdělání. Například blockchain může sloužit jako platforma pro digitální měny, jako je bitcoin nebo ethereum, které umožňují anonymní a globální platby bez potřeby centrální autority. Blockchain také může podporovat tzv. chytré smlouvy, které jsou samovykonávací programy uložené na blockchainu a spouštěné podle předem definovaných podmínek. Chytré smlouvy mohou automatizovat a zefektivnit procesy jako např. dodavatelské řetězce, pojištění nebo hlasování.

## 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)

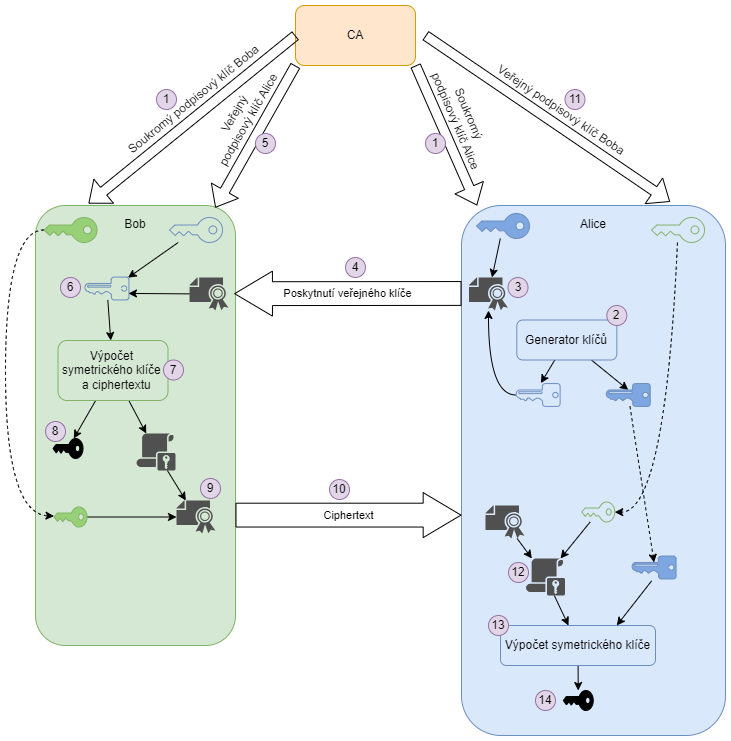
Tato funkce naváže spojení s druhou stranou a vzájemně si vymění informace takovým způsobem, aby výsledkem byl symetrický klíč, se kterým následně mohou otevřít nové, zašifrované spojení. Symetrický klíč jako takový nebyl nikdy poslán po síti, a tedy nikdo, kromě zúčastněných stran, o klíči nemůže vědět 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 KEM 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 KEM 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 získají od CA svůj soukromý podpisový klíč (bude vysvětleno v 6.3).
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 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 podpisových algoritmech a klíčích jsem uvedl v kapitole 1 této práce včetně seznamu dostupných algoritmů a jejich činnosti. V této podkapitole už tedy zobecním činnost postkvantových kryptografických podpisových 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.

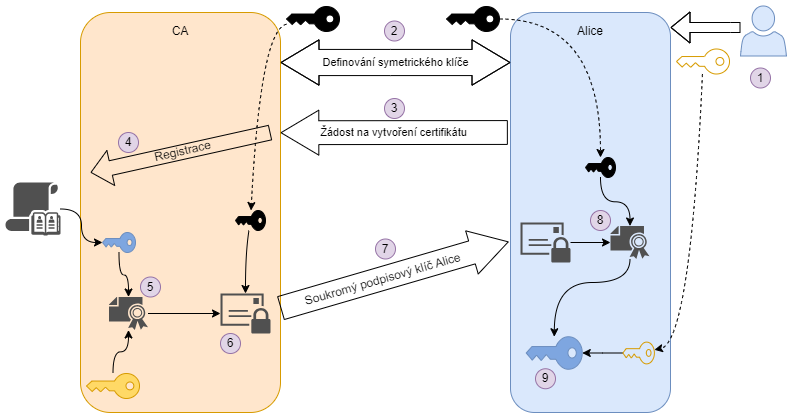
[1] <https://datatracker.ietf.org/doc/html/rfc7519>

[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.

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

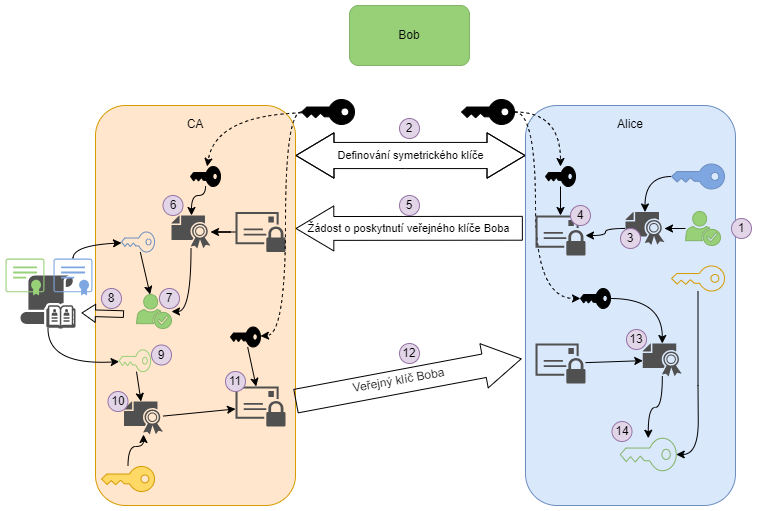
async def ask\_public\_key(subject, sign\_private\_key, my\_address, CA, ALGORITHM), která má 5 parametrů:

* 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. Obsah obrázku text, diagram, zástrčka

Popis byl vytvořen automaticky

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. Obsah obrázku diagram, text, Plán

Popis byl vytvořen automaticky

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".

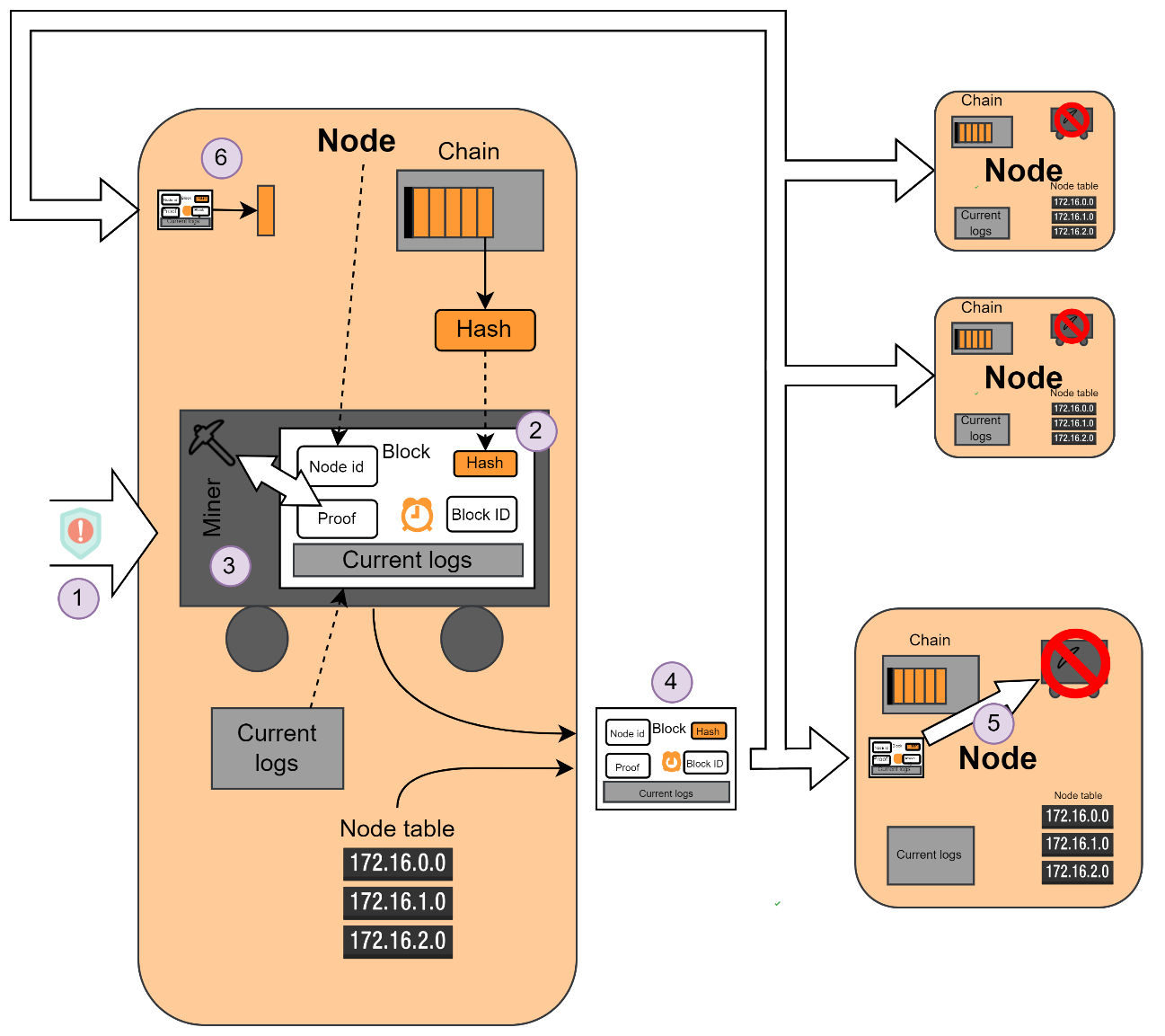
On sam zacne skladat dohromady blok zprav, ke kterym prilozi hlavicku bloku. Hlavicka obsahuje index bloku, casovou znacku zacatku tezby bloku, dukaz(prozatim nastaveny na nulovou hodnotu) a hash z predesleho bloku retezu. Tento zabaleny blok odesila do funkce pro tezbu.

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. Obsah obrázku text, snímek obrazovky, diagram, design

Popis byl vytvořen automaticky



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)

|  |
| --- |
|  |