

Středoškolská Odborná Činnost
Obor 18: Informatika

Post-kvantová kryptografie: Implementaci a
způsoby rozbití dnešní kryptografie

Středoškolská Odborná Činnost
Obor 18: Informatika

Post-kvantová kryptografie: Implementace a
způsoby rozbití dnešní kryptografie

Post-quantum cryptography: Implementation
and ways to break today's cryptography

Autor: Hlib Arseniuk

Škola: SPŠE a VOŠ v Plzni, Koterovská 85

Kraj: Plzeňský kraj

Konzultant práce: Mgr. Jan Plzák

Plzeň 2025

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Plzni dne _____ Hlib Arseniuk

Anotace

Tahle práce je výzkum post-kvantových algoritmu které jsou ověřeny NIST a způsobům jejich prolomení pomocí vyhledávací nebo faktorizačních algoritmu na kvantových počítačů. Implementace post-kvantových algoritmu do různých systému. Integrace jejich s VPN, webem. Srovnání s jinými algoritmy a jejich matematickými modely. Jejich popis, vlastnosti a problémy.

Annotation

This article is a research of post-quantum algorithms which are verified by NIST, ways of breaking them using search or factorization algorithm on quantum computers.

Implementation of post-quantum algorithms in different system. Integrating them with VPN, web. Comparison with other algorithms and their mathematical models. Their description, properties and problems.

Klíčová slova a zkratky

Zkratka	Vysvětlení
CC	Classical cryptography
PQ	Post-quantum
PQC	Post-quantum cryptography
QC	Quantum computing
QP	Quantum programming
CFT	Classical fourier transformation
QFT	Quantum fourier transformation
UI	User interface
UX	User experience
VPN	Virtual proxy network
NIST	National Institute of Standards and Technology
NSA	National Security Agency
FIPS	Federal Information Processing Standards
UBÚ	Národní Bezpečnostní Úřad

Pojmy

Classical cryptography – je komunikující strany sdílejí tajnou posloupnost náhodných čísel, která se nazývá „klíč“.

Post-quantum cryptography – je vývoj kryptografických algoritmů (obvykle algoritmů s veřejným klíčem), které jsou v současnosti považovány za bezpečné proti kryptoanalytickému útoku kvantového počítače.

Quantum computing – je rychle se rozvíjející technologie, která využívá zákony kvantové mechaniky k řešení problémů příliš složitých pro klasickou mechaniku.

Quantum programming – je proces navrhování nebo sestavování sekvencí instrukcí, tzv. kvantových obvodů, pomocí hradel, přepínačů a operátorů.

Classical fourier transformation – je integrální transformace, která bere na vstup funkci a na výstup dává jinou funkci, která popisuje, do jaké míry jsou v původní funkci přítomny různé frekvence.

Quantum fourier transformation – je lineární transformace na kvantových bitech a je kvantovou obdobou diskrétní Fourierovy transformace.

User interface – je design aplikace pro usera.

User experience – je to uživatelský zážitek z používání aplikace.

Virtual proxy network – je síťová architektura pro virtuální rozšíření privátní sítě přes jednu nebo více dalších sítí, které jsou buď nedůvěryhodné, nebo je třeba jejich izolovat.

National Institute of Standards and Technologies – je agentura spadající pod ministerstvo obchodu USA. Hlavním posláním NIST je shromažďovat a organizovat vědecké, technické, inženýrské a obchodní informace, které vytvářejí USA.

National Security Agency – je zpravodajská agentura Ministerstva obrany Spojených států amerických, která spadá pod ředitele národního zpravodajství.

Federal Information Processing Standard – jsou standardy pro federální počítačové systémy, které vypracoval Národní institut pro standardy a technologie (NIST) a schválil ministr obchodu.

Národní Bezpečnostní Úřad – je vládní agentura České republiky odpovědná za udržování bezpečnostních prověrek, ochranu utajovaných informací a kybernetickou bezpečnost České republiky.

Hash-based, Lattice-based, Code-based – jsou to různé druhy problém PQC algoritmu které pomáhají chránit proti Quantum útoku.

Qubit – logická jednotka v Kvantových výpočtech.

Superpozice – Qubit může být současně ve stavech 0 i 1, což umožňuje paralelní výpočty.

Provázání – Dva qubity sdílejí stav, změna jednoho okamžitě ovlivní druhý, i na dálku.

Kvantová interference – Kombinuje pravděpodobnostní amplitudy stavů qubitů, čímž ovlivňuje výsledek výpočtu.

Měření – Při měření qubitu se jeho superpozice zhroutí do jednoho konkrétního stavu.

Tunneling – Částice překonává energetickou bariéru, i když by to klasicky nebylo možné.

Dekoherence – proces, při kterém kvantový systém ztrácí svou kvantovou superpozici a stává se "klasickým".

Topologické stavy hmoty – jsou kvantové fáze, které vykazují odolnost vůči poruchám a lokalizovaným výkyvům.

Brute-force – je metoda, která zahrnuje systematické zkoušení všech možných kombinací k nalezení správného řešení, například při lámání šifry nebo hledání hesla.

HSM – je zařízení pro bezpečné generování a uchovávání kryptografických klíčů.

Knihovný

qiskit – Python IBM opensource knihovna.

qiskit_aer – Python IBM knihovna pro hledání QPU backendu.

oqs – Python knihovna z implementací PQC algoritmu.

numpy – Python knihovna pro práce s maticemi.

math – Python knihovna pro práce s matematikou.

matplotlib – Python knihovna pro zpracování grafů.

liboqs – C++ knihovna z implementací PQC algoritmu.

pillow – Python knihovna pro práce s obrázky.

hashlib – Python knihovna pro hashování.

flask – Python knihovna pro práce s web aplikacemi.

Poděkování

Rád bych poděkoval všem, kdo pomohl s vytvořením této práce a osobně:

Mgr. Jan Plzák

Ing. Vladyslav Arseniuk

Děkuji firmě IBM, a konkrétně pobočce na Praha, Chodov za rychlejší přístup k Quantum Processing Unit, **Heron** na 127 Qubitu. A takže k **HSM** kartám.

Obsah

1. Úvod	10
1.1. Kryptografie	10
1.1.1 Symetrická kryptografie	10
1.1.1 Asymetrická kryptografie	10
1.2. Kvantové výpočty	11
1.2.1 Kvantové jevy	11
1.2.2 Qubity	11
1.2.3 Brány	12
1.2.4 Kvantové programování	13
1.3. Post-quantová kryptografie.....	14
2. CC Algoritmy	15
2.1. NP-problemy	15
2.1.1 Eliptická křivka	16
2.1.2 FaktORIZACE čísel	16
2.1.3 Diskrétní logaritmus	16
2.2 Příklad RSA.....	17
2.3 Příklad SHA	17
3. Způsoby prolomení CC	19
3.1. Video karty a matematika.....	19
3.2. QC algoritmy	21
3.2.1 Oracle	22
3.2.2 QFT.....	23
3.2.3 Shor algoritmus	24
3.2.4 Grover algoritmus.....	26
4. Použití PQC	27
4.1. NP-problemy	27
4.1.1 Lattice-based.....	27
4.1.2 Hash-based	27
4.1.3 Code-based	28
4.2. Algoritmy	28
4.2.1 KYBER.....	29
4.2.2 DILITHIUM	30
4.2.3 FALCON	31
4.2.4 SPHINCS.....	32
5. Implementaci PQC.....	34
5.1. Web.....	34

5.2. VPN	34
Závěr	35
Seznamy	36
Literatury.....	36
Obrázků	37
Tabulek.....	38
Grafů	38
Příloh	38

1. Úvod

Post-quantová kryptografie je oblast výzkumu zaměřená na vývoj **PQC** algoritmu odolných vůči kvantovým počítačům. Tyto počítače by mohly díky algoritmům, jako je Shorův, efektivně lámat asymetrickou kryptografii používanou v současných systémech, například **RSA** nebo **ECC**. Také symetrická kryptografie by byla oslabena Groverovým algoritmem, který zrychluje hrubou sílu. Cílem postkvantových metod je proto navrhnout nové algoritmy, například založené na **lattice-based**, **code-based** a nebo **hash-based** funkcích, které budou bezpečné i v éře kvantových výpočtů.

V této práci chci ukázat, jak lze vytvořit aplikaci využívající **PQC** algoritmy. Součástí bude také demonstrace prolomení klasických kryptografických algoritmů, aby bylo vidět, jak jsou zranitelné vůči kvantovým.

1.1. Kryptografie

Kryptografie je věda o zabezpečení komunikace a dat pomocí šifrovacích technik. Zajišťuje důvěrnost, integritu a autenticitu informací. Dělí se na symetrickou a asymetrickou kryptografii, přičemž moderní systémy, jako **RSA** nebo **AES**, chrání digitální komunikaci a transakce.

1.1.1 Symetrická kryptografie

Symetrická kryptografie používá stejný tajný klíč pro šifrování i dešifrování zpráv. Je rychlá a efektivní, proto se často využívá pro šifrování velkých objemů dat. Mezi nejznámější algoritmy patří **AES** (Advanced Encryption Standard), který je bezpečný a široce používaný, a **DES** (Data Encryption Standard), který je dnes považován za zastaralý. Existují také proudové šifry, jako ChaCha20 nebo RC4. Hlavní nevýhodou symetrické kryptografie je nutnost bezpečné distribuce klíče mezi komunikujícími stranami. Příklady algoritmu:

- AES
- DES
- ChaCha20
- RC4

1.1.1 Asymetrická kryptografie

Asymetrická kryptografie, využívá dvojici klíčů – veřejný klíč pro šifrování a soukromý klíč pro dešifrování. Tento systém řeší problém bezpečné distribuce klíčů, který je hlavní nevýhodou symetrické kryptografie. Mezi nejznámější asymetrické algoritmy patří **RSA**, založený na faktorizaci velkých čísel, a **ECC** (Elliptic Curve Cryptography), který poskytuje stejnou úroveň zabezpečení s kratšími klíči. Kromě šifrování se asymetrická kryptografie používá také pro digitální podpisy (např. ECDSA) a výměnu klíčů (např. Diffie-Hellman). Příklady algoritmu:

- RSA
- ElGamal
- Paillier
- DSA
- DH

1.2. Kvantové výpočty

Kvantové výpočty jsou revolučním přístupem k řešení složitých problémů pomocí principů kvantové mechaniky. Na rozdíl od klasických počítačů, které pracují s **bity** (0 nebo 1), kvantové počítače využívají **qubity**, které mohou být současně v superpozici stavů 0 i 1. Díky jevům, jako je **provázání** (entanglement) a **kvantová interference**, mohou kvantové algoritmy exponenciálně zrychlit některé výpočty. Například **Shor's** algoritmus dokáže efektivně faktorizovat velká čísla, což ohrožuje **CC**, zatímco **Grover's** algoritmus zrychluje vyhledávání. PQC je proto klíčovým oblastem bezpečnosti budoucnosti.

1.2.1 Kvantové jevy

Superpozice – Qubit může být současně ve stavech 0 i 1, což umožňuje paralelní výpočty.

Provázání – Dva qubity sdílejí stav, změna jednoho okamžitě ovlivní druhý, i na dálku.

Kvantová interference – Kombinuje pravděpodobnostní amplitudy stavů qubitů, čímž ovlivňuje výsledek výpočtu.

Měření – Při měření qubitů se jeho superpozice zhroutí do jednoho konkrétního stavu.

Tunneling – Částice překonává energetickou bariéru, i když by to klasicky nebylo možné.

Dekoherence – proces, při kterém kvantový systém ztrácí svou kvantovou superpozici a stává se "klasickým".














1.2.2 Qubity

Qubit je základní jednotka informace v kvantových počítačích, která se liší od klasického bitu. Na rozdíl od bitu, který může být pouze ve stavu 0 nebo 1, může být qubit současně v **superpozici** těchto dvou stavů. To umožňuje kvantovým počítačům provádět paralelní výpočty. Kromě toho qubity mohou být **provázané** (entanglement), což znamená, že stav jednoho qubitů může okamžitě ovlivnit stav druhého, i když jsou daleko od sebe. Druhy qubitů:

- **Superconducting qubit**, Využívá supravodivé obvody, které umožňují velmi nízké odporové ztráty.
- **Trapped ion qubit**, Používá jednotlivé ionty, které jsou udržovány v elektromagnetickém poli a manipulovány laserovými paprsky.
- **Topological qubit**, Tento typ qubitů využívá **topologické stavy hmoty**, které jsou velmi odolné vůči dekoherenci a vnějším vlivům.
- **Photonic qubit**, Fotonové qubity využívají světelné částice (fotony) k reprezentaci kvantových informací.
- **Silicon qubit**, Tyto qubity využívají elektrony nebo jádra atomů v křemíkových materiálech.

1.2.3 Brány

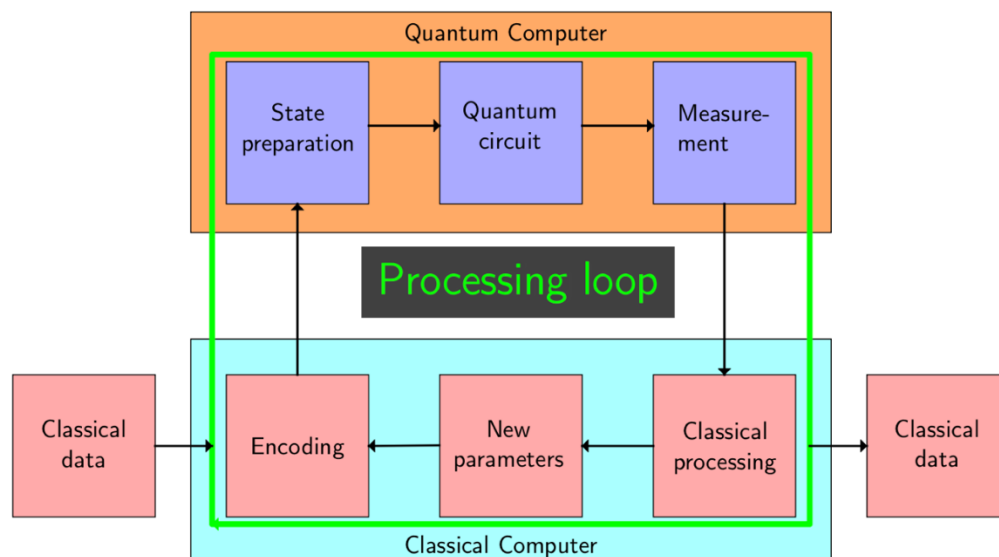
Kvantové brány jsou základními operacemi v kvantových výpočtech, které manipulují s qubity. Na rozdíl od klasických logických hradel, které mění hodnotu bitu, kvantové brány mění kvantový stav qubitů pomocí **kvantových jevů**, jako je superpozice a interference. Každá kvantová brána provádí určitou operaci, například **Hadamardova brána**, která uvádí qubit do superpozice, nebo **CNOT brána**, která vytváří kvantové provázání mezi dvěma qubity. Kvantové brány jsou základními stavebními kameny pro tvorbu kvantových algoritmů, a jejich kombinací lze provádět složité kvantové výpočty.

Operator	Gate(s)	Matrix
Pauli-X (X)	 	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)	 	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP	 	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

Obrázek 1: Kvantové brány

1.2.4 Kvantové programování

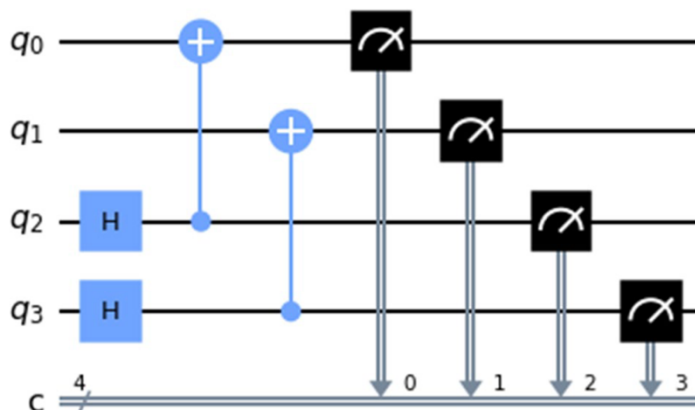
Kvantové programování je proces vytváření algoritmů a aplikací pro kvantové počítače. Místo tradičního zápisu na bázi bitů využívá kvantového zápisu, který pracuje s qubity. Kvantový programátor musí chápat kvantové jevy, jako je superpozice, provázání a interference, aby efektivně využil potenciál kvantových počítačů. Kvantové programovací jazyky, jako **Qiskit** nebo **Cirq**, umožňují psát a spouštět kvantové algoritmy. Kvantové programování vyžaduje novou paradigmatu pro vývoj, která se liší od tradičního programování na klasických počítačích. Kvantové programování zahrnuje i optimalizaci algoritmů pro kvantové architektury, kde se využívají specifické kvantové brány a operace. Cílem je **efektivní využití kvantových počítačů** pro řešení komplexních problémů. Nejčastěji se používají, pro řešení problémů, hybridní přístup:



Obrázek 2: Hybrid quantum loop

Quantum circuit je struktura, která popisuje postup operací na qubitech v kvantovém výpočtu. Skládá se z kvantových bran, které manipulují s **qubity**, a měření, které zajišťuje výstupní stav. Kvantový obvod je základem pro implementaci kvantových algoritmů.

Oni umožňují provádět složité kvantové operace, jako je superpozice a provázání. Jejich správná konstrukce je klíčová pro efektivní kvantové výpočty, které využívají paralelismus a kvantové interference pro rychlé řešení problémů. Obrazek můžete vidět na další stránce.



Obrázek 3: Preparation state for four-qubit entangled state

1.3. Post-quantová kryptografie

PQC je oblast kryptografie, která se zaměřuje na vývoj bezpečných kryptografických algoritmů, jež budou odolné vůči útokům kvantových počítačů. Kvantové počítače, díky svým schopnostem řešit problémy, které jsou pro klasické počítače extrémně složité, mohou efektivně prolomit tradiční kryptografické metody, jako je **RSA** nebo **ECC**. **PQC** se tedy soustředí na vytváření algoritmů, které by byly bezpečné i v éře kvantových výpočtů, a zajišťují tak dlouhodobou bezpečnost digitálních informací a komunikace. **PQC** se nezabývá kvantovými výpočty, ale vytváří šifrovací metody odolné vůči nim. Takže používá více složitější problémy pro vytvoření kryptografických algoritmu.

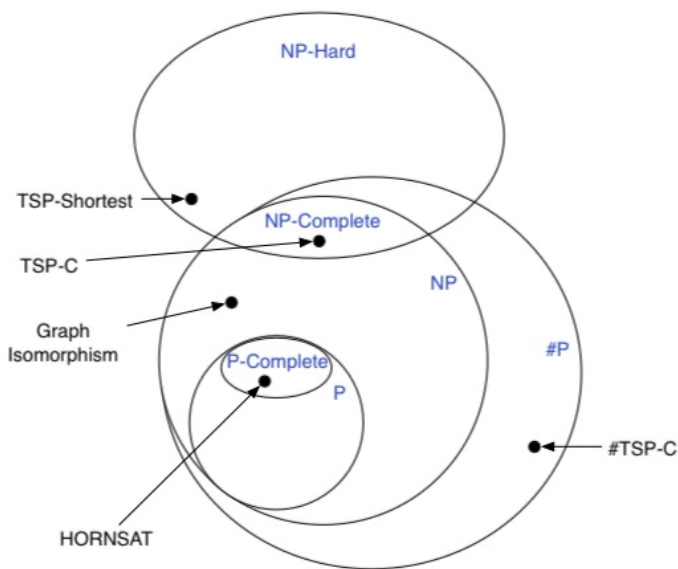
2. CC Algoritmy

Klasická kryptografie, jako je RSA, Diffie-Hellman nebo DSA, tvoří základ současné digitální bezpečnosti. **RSA** využívá faktORIZACI velkých čísel pro asymetrické šifrování a digitální podpisy. Diffie-Hellman umožňuje bezpečnou výměnu klíčů i přes nezabezpečený kanál. Tyto metody spoléhají na složitost matematických problémů (**NP** a ostatní), například faktorizace nebo výpočtu diskretního logaritmu. Přestože jsou stále široce používány, jejich bezpečnost může být ohrožena pokrokem v kvantové kryptografii. Proti tomu vznikají **PQ** algoritmy, jako je Kyber nebo Dilithium. Klasická kryptografie však zůstává klíčovou součástí zabezpečení internetu, e-mailů či bankovníctví.

CC algoritmy nás v dnešní době chrání. Nemůžeme je zlomit našimi technologiemi za vhodný čas... Nebo můžeme? Na tuhle otázku nám pomůže odpovědět 2 věci. Jak funguje algoritmus jaký se snažíme rozbit a jaké máme technologie v přístupu.

2.1. NP-problemy

NP problémy jsou úlohy, jejichž řešení lze ověřit v polynomiálním čase, ale není jisté, zda je lze také v polynomiálním čase vyřešit. Nejznámější třídou je **NP-úplné** problémy, například problém obchodního cestujícího nebo splnitelnost Booleovských formulí (SAT) pro který optimálně řešení je Davis Logemann Loveland algoritmus nebo WalkSATu.



Obrázek 4: Soubory problémů v matematice

Pokud by se našel polynomiální algoritmus pro libovolný **NP-úplný** problém, znamenalo by to **P = NP**, což je jedna z největších otevřených otázek informatiky. Tyto problémy mají praktický dopad v optimalizaci, kryptografii nebo strojovém učení. Kvantové výpočty by mohly některé NP problémy urychlit, ale úplné řešení stále chybí.

2.1.1 Eliptická křivka

Problém eliptických křivek souvisí s jejich využitím v kryptografii (**ECC** – Elliptic Curve Cryptography). Eliptické křivky jsou rovnice ve tvaru.

$$y^2 = x^3 + ax + b$$

definované nad konečnými tělesy. Jejich bezpečnost spočívá v obtížnosti problému diskrétního logaritmu na eliptických křivkách (**ECDLP**). To znamená, že pokud máme bod P a jeho násobek.

$$Q = kP$$

Je extrémně těžké určit k , i když známe P a Q . **ECC** nabízí vyšší bezpečnost než **RSA** při kratší délce klíče, což ji činí efektivnější pro šifrování, digitální podpisy i výměnu klíčů. Hrozbou pro **ECC** jsou kvantové počítače, které by pomocí Shorova algoritmu mohly prolomit **ECDLP**. Proto se zkoumají postkvantové alternativy. Další výzvou jsou chyby v implementaci, vedoucí k bočním útokům či zranitelnostem jako "invalid curve attack".

2.1.2 Faktorizace čísel

Problém faktorizace čísel spočívá v rozkladu složeného čísla na jeho prvočíselné činitele. Pro malá čísla je tento úkol snadný, ale pro velká čísla, zejména ta používaná v kryptografii, je extrémně obtížný. p a q jsou primární čísla.

$$n = pq$$

Pokud chceme velké n rozložit na 2 prvočísla, bude to časově náročné. Bez známého efektivního algoritmu pro faktorizaci velkých čísel je tento problém klíčovým základem bezpečnosti asymetrických šifrovacích systémů, jako je **RSA**.

Současné faktorizační algoritmy, například kvadratické síto nebo číslem řízené síto, jsou efektivní pro středně velká čísla, ale pro 2048bitová čísla používaná v **RSA** jsou stále příliš pomalé.

2.1.3 Diskrétní logaritmus

Problém diskrétního logaritmu (**DLP** – Discrete Logarithm Problem) spočívá v nalezení exponentu x v rovnici.

$$g^x \equiv h \pmod{p}$$

Kde g je generátor, p je velké prvočíslo a h je známý prvek. Zatímco umocňování lze snadno spočítat, inverzní operace – nalezení x – je extrémně obtížná, zejména pokud je p dostatečně velké.

Bezpečnost mnoha kryptografických algoritmů, jako je Diffie-Hellman nebo **DSA**, závisí na obtížnosti **DLP**. Existují algoritmy, jako je Pohlig-Hellman nebo index-calculus, které mohou problém urychlit, ale pro dostatečně velké moduly je stále výpočetně náročný. Kvantové počítače

představují hrozbu, protože Shorův algoritmus může **DLP** řešit v polynomiálním čase, což by zlomilo současné kryptografické systémy založené na tomto problému.

2.2 Příklad RSA

RSA je asymetrický šifrovací algoritmus využívající obtížnost faktorizace velkých čísel. Používá veřejný a soukromý klíč pro šifrování a digitální podpisy. Je základem internetové bezpečnosti. Předtím jak použít RSA, musíme z generovat náhodně dva velkých primárních čísla q a p . Používáme NP problém.

$$n = pq$$

Máme číslo n které má dvě dělitele p a q .

$$\varphi(n) = (p - 1)(q - 1)$$

Počítáme Eulerovou funkci. A musíme zjistit veřejný exponent e který není dělitel $\varphi(n)$. Často se používá 65537. Pak, určíme soukromý exponent d jako multiplikační inverzi e modulo $\varphi(n)$.

$$d \equiv e^{-1} \bmod \varphi(n)$$

A máme všechno připravené. Pubický klíč je to (n, e) , Privátní (n, d) . A můžeme uvést příklad šifrování a dešifrování zprávy.

$$c \equiv m^e \bmod n$$

Kde m je zpráva a c zašifrovaná data v binárním tvaru. A dešifrování.

$$m \equiv c^d \bmod n$$

2.3 Příklad SHA

SHA (Secure Hash Algorithm) je rodina kryptografických hašovacích funkcí používaných pro zabezpečení dat. SHA-1 byl dříve populární, ale kvůli kolizím byl nahrazen SHA-2 (např. SHA-256, SHA-512), které poskytují vyšší bezpečnost. Tyto algoritmy přeměňují vstup libovolné délky na pevně dlouhý výstup (hash), který je unikátní pro daná data. **SHA** se využívá v digitálních podpisech, ověřování hesel a blockchainových technologiích. Výpočet zahrnuje bitové operace, míchání dat a iterativní transformace. SHA-3, založená na Keccak algoritmu, je moderní varianta s odlišnou konstrukcí.

Vstupní data se nejprve doplní podle pravidel **SHA** tak, aby jejich délka byla násobkem 512 bitů. Přidá se jednička a následně nuly, dokud není délka správná. Na konec se připojí 64bitová reprezentace původní délky vstupu.

Inicializace, Použije se osm fixních 32bitových počátečních hodnot (konstant), které pocházejí z binárních reprezentací odmocnin prvních osmi prvočísel.

$$H_0 = 0x67413213, H_1 = 0x83295321 \dots$$

Pak, každý blok se rozdělí na šestnáct 32bitových slov $H_0, H_1, H_2 \dots$ a následně rozšíří na 64 slov.

$$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \ll 1, \text{ pro } t = 32 \text{ až } 63$$

Značka \oplus je **XOR** a $\ll 1$ je *bitová rotace doleva o 1 bit*. Používají se bitové operace rotace, posuny a **XOR** pro míchání dat. Každé kolo zpracování (64 iterací) využívá speciální konstanty a funkce komprese $f(x)$.

Po zpracování všech bloků je výsledkem 256bitový hash (64 hexadecimálních znaků).

$$H_0, H_1, H_2 \dots$$

Na výsledků máme hash blok které musíme jenom připojit k sobě.

$$H_{string} = H_0 || H_1 || H_2 || \dots$$

3. Způsoby prolomení CC

Moderní kryptografie je založena na složitých matematických problémech, ale existují způsoby, jak ji prolomit. **Brute-force útoky** zkoušejí všechny možné klíče, ale jsou omezené výpočetním výkonem. **Boční kanálové útoky** analyzují fyzické chování zařízení, například spotřebu energie. **Matematické útoky**, jako faktorizace čísel (RSA) nebo řešení diskretního logaritmu (ECC, Diffie-Hellman), hledají efektivnější algoritmy. **Kvantové počítače** s Shorovým algoritmem mohou prolomit RSA i ECC. **Kolizní útoky** na hašovací funkce, například SHA-1, ukazují slabiny v jejich konstrukci. Ale i když se nám nepodaří prolomit šifru hned, můžeme ji prolomit později, což je důvod, proč hackeři shromažďují data nyní a proč musíme začít přecházet na PQC.

3.1. Video karty a matematika

Výkonné **video karty (GPU)** výrazně urychlují matematické výpočty, což umožňuje efektivnější **brute-force útoky**, lámání SHA-1,2 a řešení **RSA** či **ECC**. GPU excelují v paralelním zpracování, což zvyšuje riziko prolomení zastaralých kryptografických algoritmů.

Video karty taky se využívají pro počítání těchto algoritmů, ale jsou lepší přístroje na to jako **HSM**, který nejsou udělaný pro prolomení šifry.

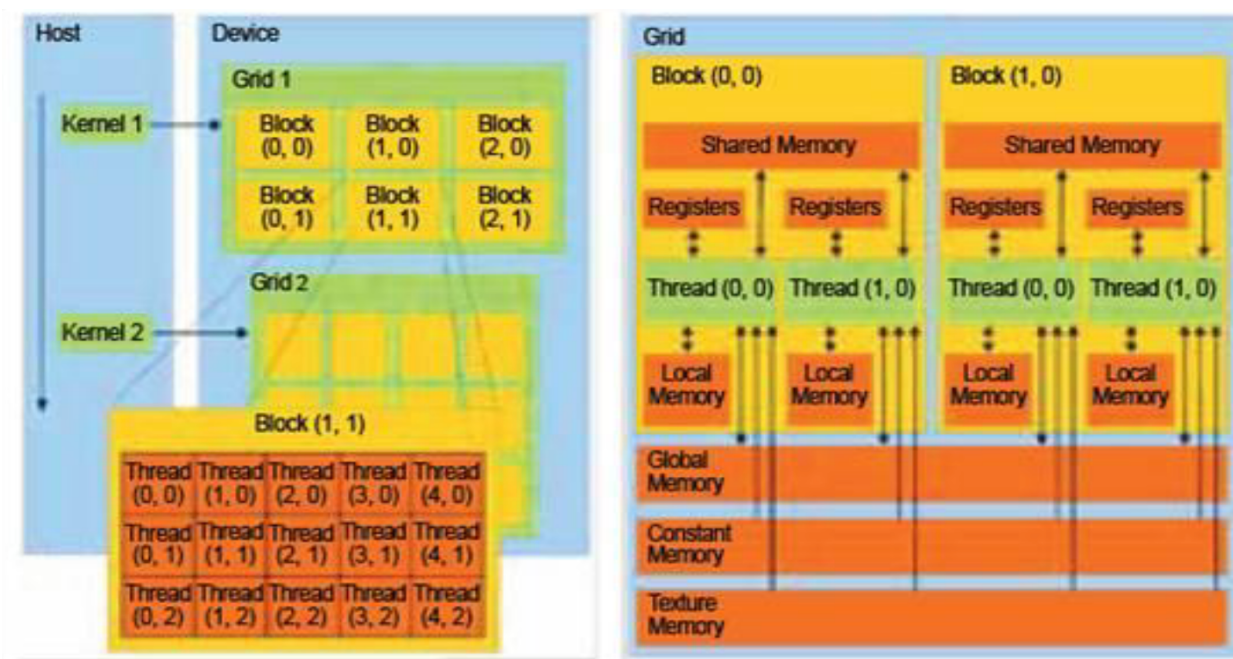
Pokud chceme rozbit RSA tak musíme zjistit q a p , pro malý n je to lehký. Můžeme použít cyklus for.

```
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True

def find_prime_factors(N):
    for q in range(2, N):
        if is_prime(q) and N % q == 0:
            p = N // q
            if is_prime(p):
                return q, p
    return None
```

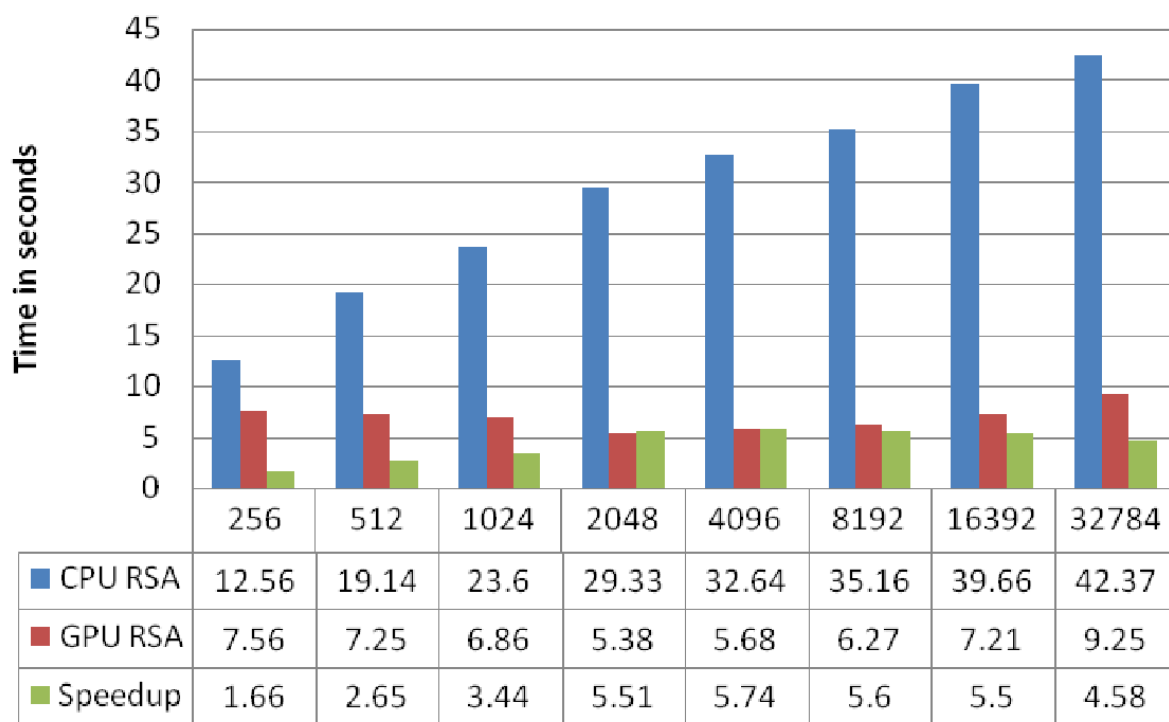
Obrázek 5: Code for determining of p and q

Kde res je první primární číslo a n/res je druhé primární číslo. Na **GPU** můžeme zrychlit a rozdělit na bloky \sqrt{n} a počítat stejný cyklus jenom v více jádrech, a **GPU** umožňují to udělat rychleji protože mají více jader.

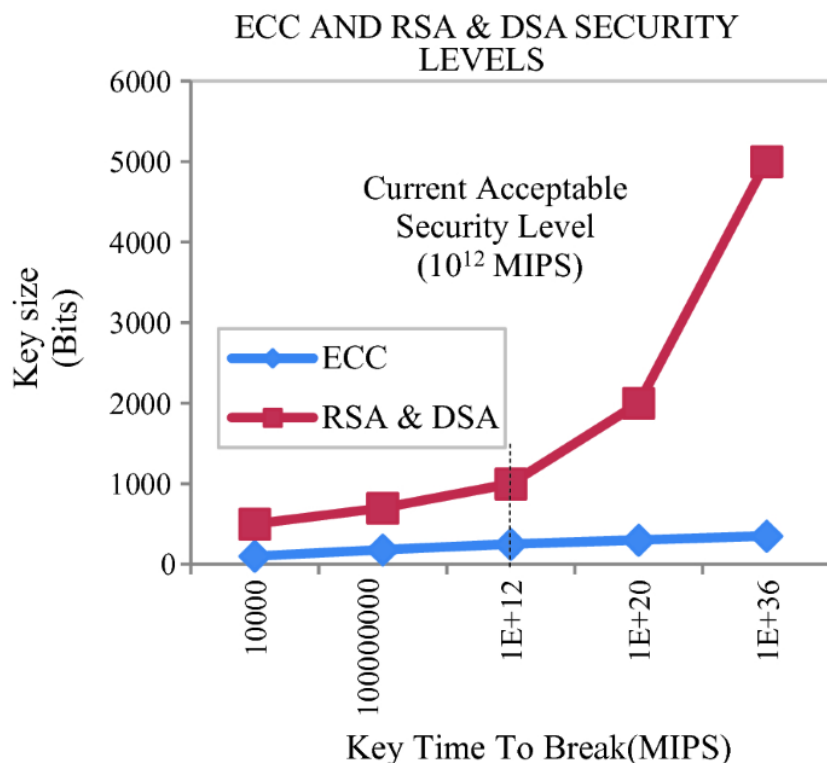


Obrázek 6: CUDA Architecture

Když vytváříme klíč tak **GPU** urychlí to v **několik** krát.



Graf 1: CPU a GPU porovnání

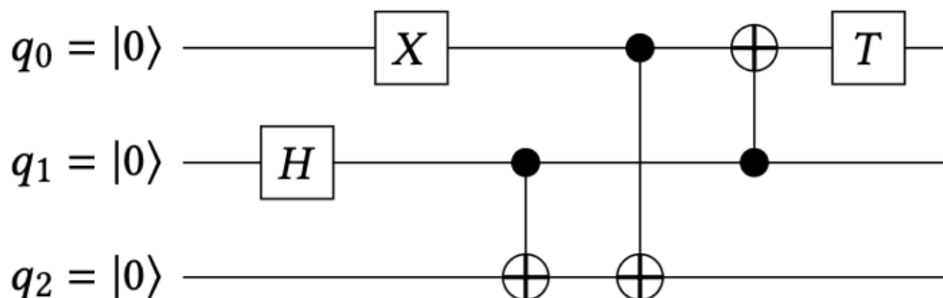


Graf 2: Not Optimized by GPU breaking of RSA and ECC with dependence on time

Jak můžeme vidět s grafů. Trvá to docela dlouho. ECC a RSA jsou dobré algoritmy ale da se jich prolomit. Otázka jenom čas. A naše úloha je jeho zmenšení.

3.2. QC algoritmy

QC algoritmy využívají principy kvantové mechaniky, jako superpozici a provázanost, k řešení složitých problémů efektivněji než klasické algoritmy. Dohromady, strukturu QC algoritmy mají stejnou. Předběžné zpracování klasických dat (data normalization), převod do kvantového stavu (**Oracle**), řešení problému nebo použití optimalizačních algoritmů, měření výsledku a opakování algoritmu.



Obrázek 7: QC algorithm example

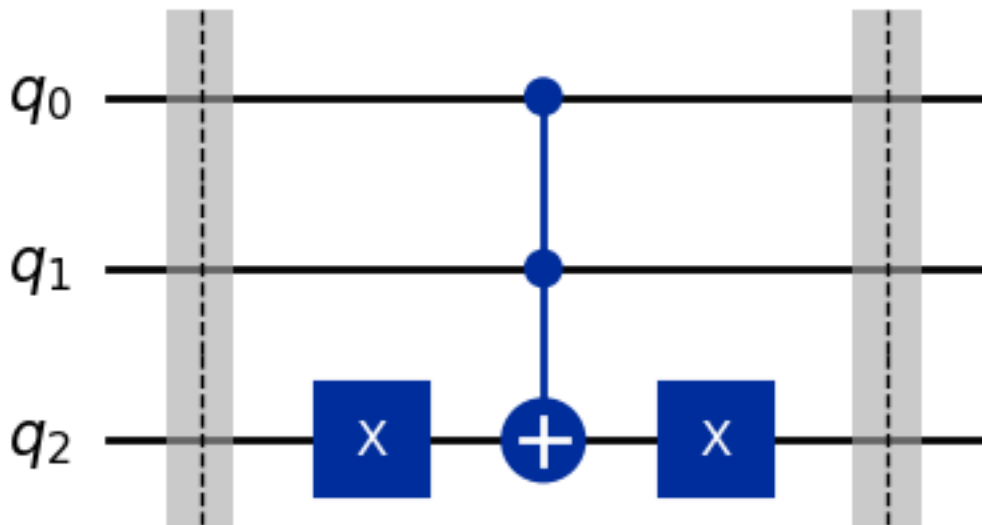
3.2.1 Oracle

V kvantových algoritmech **oracle** je speciální kvantová podfunkce, která **zakóduje informaci o problému** a umožňuje efektivní výpočet pomocí kvantových operací. Orákulem se obvykle implementuje jako kvantová černá skříňka, která při aplikaci na vstupní stav **označí správná řešení**, aniž by bylo nutné je explicitně vyhledávat.

Například v **Grover's algoritmu** orákulum invertuje amplitudu hledaného prvku, čímž zvyšuje pravděpodobnost jeho nalezení. V **Deutsch's algoritmu** testuje, zda je funkce konstantní nebo vyvážená.

Orákulum je zásadní pro efektivitu kvantových algoritmů, protože umožňuje exponenciální nebo kvadratickou akceleraci oproti klasickým metodám při řešení specifických problémů, jako je vyhledávání, faktorizace a optimalizace. Každý algoritmus má jiný oracle, ale jejich princip stejny. Takže často jeden qubit se přidává navíc pro collapse oraclu kvůli přesnějším výsledku.

Na obrázku 7 a 8 můžete vidět implementace oraclu pro grov's algoritmus na 3 qubity v qiskitu z podrobnou schématu toho jak vypadá.



Obrázek 9: Grover's algorithm oracle circuit for finding state $|111\rangle$

```
from qiskit import QuantumCircuit

# Oracle
def grover_oracle():
    oracle = QuantumCircuit(3)
    oracle.barrier() # for visual separation
    oracle.x(2)
    # Apply a controlled-Z gate with the first qubit as control and the third qubit as target
    oracle.ccx(0, 1, 2)
    oracle.x(2)
    oracle.barrier()

    return oracle
```

Obrázek 8: Code for oracle of grover's algorithm

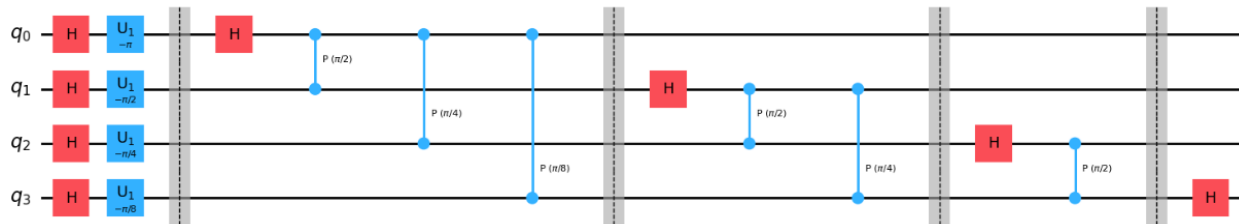
3.2.2 QFT

Kvantová Fourierova transformace (QFT) je kvantová verze klasické **diskrétní Fourierovy transformace (DFT)** a hraje klíčovou roli v kvantových algoritmech, například **Shor's algoritmu** pro faktORIZACI čísel.

Matematicky QFT transformuje vstupní kvantový stav $|x\rangle$ do superpozice všech možných stavů s fázovými posuny.

$$QFT|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i xy/N} |y\rangle$$

Kde $N = 2^n$ pro n -qubitový registr. QFT lze implementovat kvantovým obvodem složeným z Hadamardových hradel a řízených fázových posunů, přičemž její složitost je $O(N^2)$ oproti klasické $O(N^2)$, což umožňuje exponenciální urychlení. Používá se v kvantových algoritmech pro periodické funkce, řešení lineárních rovnic nebo fázovou odhadní technikou kvantové informatice.



Obrázek 11: QFT circuit

```
import qiskit
import math as m
from qiskit.circuit.library import U1Gate
def QTF(n):
    qtf_circuit = qiskit.QuantumCircuit(n)
    qtf_circuit.h([i for i in range(n)])

    for i in range(n):
        qtf_circuit.append(U1Gate(-m.pi/pow(2, i)), [i])

    for i in range(n):
        qtf_circuit.barrier()
        qtf_circuit.h(i)
        for j in range(i+1, n):
            qtf_circuit.cp(m.pi/pow(2, j-i), i, j)

    return qtf_circuit
```

Obrázek 10: QFT implementation in qiskit

QTF v kódu se implementuje pomocí Hadamardových hradel a řízených fázových posunů.

První část kódu aplikuje Hadamardova hradla na všechny qubity, čímž vytvoří superpozici všech stavů. Poté se přidává fázový posun pomocí $U1$ gate s úhlem $-\pi/2^i$, což zajistí správnou transformaci.

Ve druhé části se znovu aplikují Hadamardova hradla, tentokrát jednotlivě, a mezi qubity se vkládají řízené fázové posuny $CP(-\frac{\pi}{2^i})$, které odpovídají Fourierově transformaci v kvantovém obvodu.

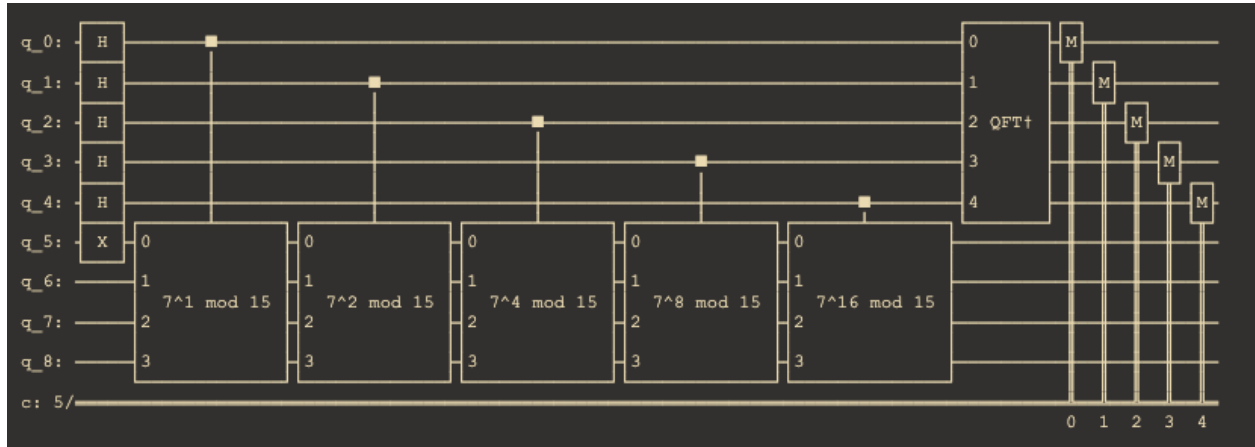
$$QTF|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i xy/N} |y\rangle$$

$$= \frac{1}{\sqrt{2^N}} (|0\rangle + e^{2\pi i 0 \cdot j_n} |1\rangle) * (|0\rangle + e^{2\pi i 0 \cdot j_{n-1}} |1\rangle) \dots (|0\rangle + e^{2\pi i 0 \cdot j_1 j_2 \dots j_n} |1\rangle)$$

Pokud rozložíme a porovnáme s kvantovým stavem qubitu, oni budou stejní. Použité fáze vycházejí z definice Fourierovy transformace a zajišťují správné interference amplitud, které jsou klíčové pro kvantové algoritmy jako Shorův faktorizační algoritmus nebo kvantový odhad fáze.

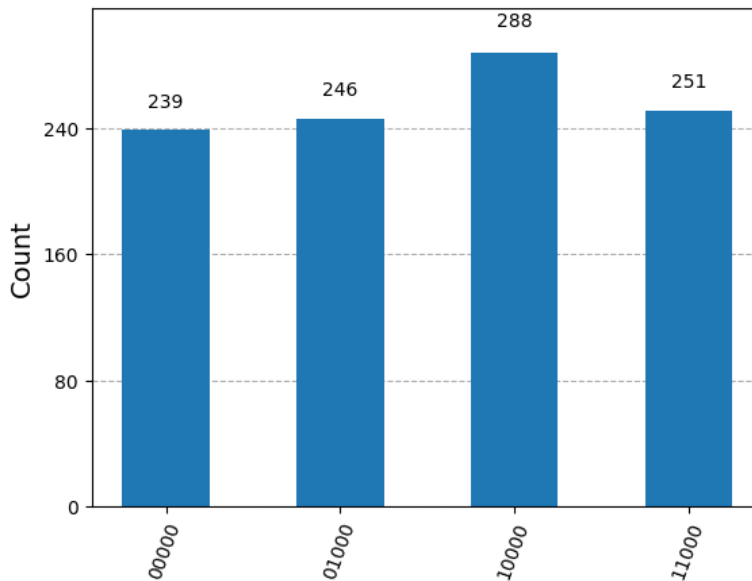
3.2.3 Shor algoritmus

Shor's algoritmus je QC algoritmus pro **faktorizaci čísel**, který ohrožuje současnou asymetrickou kryptografií, jako **RSA**. Využívá **kvantovou Fourierovu transformaci (QFT)** k nalezení **periodicity** funkce související s modulo exponentiací. Výzkum se zaměřuje na **optimalizaci implementace na kvantových počítačích a snížení chybovosti kvantových hradel**.



Obrázek 12: Shor's algorithm for 5 qubits and 3 qubits in second register

Algoritmus se používá QFT a přesně inverzní verze její.



Graf 3: Output of Quantum Computer Heron 127 qubits

Je to statistika **1024** měření **qubitu** na konce algoritmu. Výslední qubity měli jenom 4 stavy: 00000, 01000, 10000, 11000. Ted' můžeme zjistit jejich fáze s pomocí transformace do decimálně soustavy a použití vzorce.

$$\text{číslo}_{\text{decimal}}/2^N$$

Kde N – počet qubitu. Pak musíme použít quantum phase estimator algoritm z pomoci kterého můžeme odhadnout p a q v gcd() algoritmu.

```

ATTEMPT 1:
Register Reading: 01000000
Corresponding Phase: 0.25
Result: r = 4
Guessed Factors: 3 and 5
*** Non-trivial factor found: 3 ***
*** Non-trivial factor found: 5 ***
    
```

Obrázek 13: Výsledek Shor's algoritmu na 5 qubitu

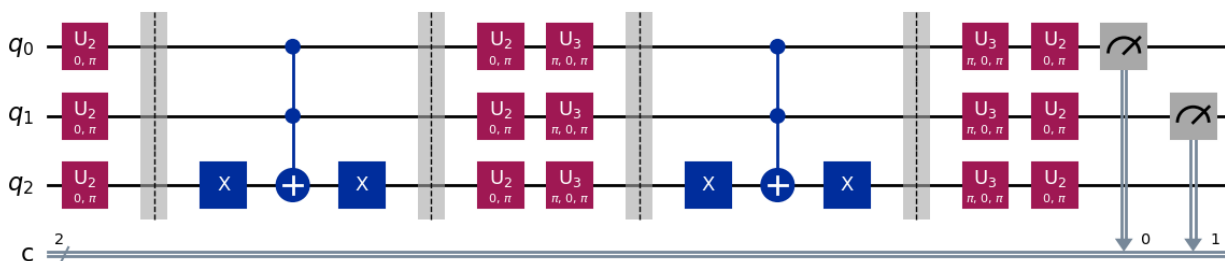
3.2.4 Grover algoritmus

Grover's algorithm je **QC** algoritmus, který slouží k nalezení konkrétního elementu v neuspořádané databázi nebo seznamu, což je úloha známá jako „problematika hledání“. Tento algoritmus představuje kvantový přístup k problému, který by klasické algoritmy řešily s časovou složitostí lineárně, zatímco **Grover's** algoritmus výrazně zkracuje časovou složitost na čtvercovou. Toho je dosaženo díky kvantovým vlastnostem, jako je superpozice a interference.

Předtím už jsme probrali oracle pro Grover. V našem příkladě my hledáme state $|111\rangle$. Mezi oběma orakly máme difuzní operátor, který zrychluje hledání a zesiluje fázovou amplitudu.

$$D = 2|\psi\rangle\langle\psi| - I$$

Je to matematický popis diffusion operátora kde ψ je stav kvantového systému, I je jednotková matice.



Obrázek 14: Grover's algorithm na 3 qubity pro $|111\rangle$ state

Na konce pro upřesnění data můžeme použít ještě jednou **D** operátor, čím více jich bude, tím více to bude náročné na výpočet a přesnější. Pokud chceme hledat jiný state (vstup). Tak potřebujeme změnit oracle. To je asi jediný problém, který má algoritmus.

4. Použití PQC

PQC algoritmy jsou algoritmy navrženy tak, aby byly odolné vůči útokům kvantových počítačů. Používají se například v šifrování (např. **Lattice-based**, **Code-based**, **Multivariate polynomial**), kde kvantové počítače mohou zcela ohrozit současné šifrovací metody, jako **RSA** nebo **ECC**. Tyto algoritmy zajišťují bezpečnost i v post-kvantové éře.

4.1. NP-problemy

Zde budou NP-problémy které dělají potíže pro **QC**.

4.1.1 Lattice-based

Lattice-based je třída problémů, které se zakládají na mřížkách (lattices) a jsou považovány za obtížné i pro kvantové počítače. Mřížky jsou matematické struktury složené z pravidelně rozmístěných bodů v n-rozměrném prostoru. Mnohé problémy v této oblasti, jako například problém najít nejbližší vektor nebo problém hledání krátkého vektoru, jsou **NP-úplné** a nelze je efektivně vyřešit klasickými ani kvantovými počítači. Tyto problémy mají široké využití v postkvantovém šifrování a kryptografii, protože jejich obtížnost zůstává zachována i po příchodu kvantových algoritmů. Matematicky, mřížku L lze definovat tak.

$$L = \left\{ \sum_{i=1}^n k_i * v_i \mid k_i \in \mathbb{Z} \right\}$$

Kde v_i jsou základní vektory mřížky a k_i jsou celá čísla. Dvě běžné problémy v lattice-based kryptografii jsou **SVP**, problém nejbližšího vektoru a zjistit nejbližší vektor k bodu. Oni jsou **NP-úplné**, a není efektivní klasicky ani kvantový přístup.

4.1.2 Hash-based

Hashed problem se týká problémů, kde se používají kryptografické **hašovací** funkce k transformaci vstupních dat na pevně stanovené délky výstupních hodnot. Tyto problémy jsou důležité v kontextu bezpečnosti, protože **hašování** zajišťuje jednosměrnost (nemůže se zpětně dešifrovat) a odolnost vůči kolizím.

$$H = \{0, 1\}^* \rightarrow \{0, 1\}^n$$

Kde $\{0, 1\}^*$ představuje množinu všech binárních řetězců libovolné délky a $\{0, 1\}^n$ je množina n-bitových výstupních hodnot. Hlavními vlastnostmi hašovacích funkcí jsou:

- **Jednosměrnost**, je těžký najít vstup x .
- **Determinističnost**, pro každý x je vždy stejný $H(x)$.
- **Odolnost vůči kolizím**, je těžký najít dvě různé výstupy.

4.1.3 Code-based

Code-based je kryptografické problém založený na teorii kódování, konkrétně na dekódování špatně přenesených nebo poškozených kódovaných zpráv. Nejznámější je problém dekódování, kde úkolem je najít původní zprávu z přijatého kódu, který obsahuje **chyby**. Tento problém je považován za těžký, protože pro jeho efektivní řešení nejsou známy polynomiální algoritmy. Code-based problémy jsou odolné vůči kvantovým počítačům, což je činí ideálními pro **PQC**.

$$y = xG$$

Kódování, x je původní zpráva $\in F_2^k$ kde F_2 je Galosovo pole, tedy binární soustava, je zakódována pomocí matice G na kódovaný vektor $y \in F_2^n$ kde n je délka kódu.

$$y' = xG + e$$

Vektor y' obsahuje chyby, a cílem je rekonstruovat původní zprávu x . Dekódování se provádí na základě dekódovací matice H , která je související s maticí G a určuje kódy které lze opravit.

Matematicky je dekódování problém **NP**-úplný a je odolné vůči kvantovým počítačům, protože na základě kódových vlastností neexistují známé efektivní algoritmy pro rozluštění původního zprávu při přítomnosti chyb.

4.2. Algoritmy

PQC algoritmy jsou kryptografické algoritmy navrženy tak, aby byly odolné vůči útokům kvantových počítačů. V tuto práci už jsme zjistili, že **RSA** šifrování je bariera ale na krátký čas. Nyní se musíme zaměřit na implementaci PQC metod, jako jsou lattice-based nebo code-based algoritmy, které poskytují vyšší odolnost vůči kvantovým útokům.

Je jejich více, ale **NIST** (National Institute of Standards and Technology) vybral několik PQC algoritmů jako kandidáty na nové standardy.

- Kyber
- Dilithium
- Falcon
- SPHINCS+

4.2.1 KYBER

Kyber je PQC algoritmus navržený pro zabezpečenou výměnu klíčů (tzv. Key Encapsulation Mechanism, **KEM**). Je založen na lattice-based kryptografii a využívá problem Module Learning With Errors (**MLWE**), který je považován za odolný vůči útokům kvantových počítačů.

$$R_q = \mathbb{Z}[X]/[X^n + 1]$$

Kyber je založen na matematickém problému zvaném **LWE** problemů. R_q je polynomiální okruh, p je prvočíslo, n je stupeň polynomu typicky 256.

$$(A, A * s + e) \\ (A, u \in R_q^k)$$

A je náhodná matice v $R_q^{k \times k}$, s je tajný vektor v R_q^k , e je malý chybový vektor v R_q^k . A problém **MLWE** spočívá v rozlišení mezi (A, d) a (A, u) . Problém **MLWE** je považován za obtížný jak pro klasické, tak pro kvantové počítače, což zajišťuje bezpečnost Kyberu.

Takže kyber má různé parametry od kterých zaleží jeho bezpečnost.

NIST Security Level	n	k	q	η_1, η_2	pk	sk	ct
1	256	2	3329	3, 2	800	1632	768
3	256	3	3329	2, 2	1184	2400	1088
5	256	4	3329	2, 2	1568	3168	1568

Tabulka 1: Kyber parametry

Generace klíče:

$$A \in R_q^{k \times k} \\ s \in R_q^k \\ e \in R_q^k \\ t = As + e$$

A je **náhodná** matice a s, e **náhodný** vector s R_q^k . (A, t) – public key, s – secret key.

Encrypting:

$$\begin{aligned} r &\in R_q^k \\ u &= A^T r + e_1 \\ v &= t^T r + e_2 + m \end{aligned}$$

Kde e_1 je malý **chybový** vektor a e_2 je **malá chybová hodnota**, m je **binární** zpráva.

Decrypting:

$$v' = v - s^T u$$

Pak zprávu **zaokrouhlujeme** na nejbližší hodnotu v **prostou** zpráv. A máme m jako před tím.

```
with oqs.KeyEncapsulation("Kyber512") as client:
    with oqs.KeyEncapsulation("Kyber512") as server:
        public_key_client = client.generate_keypair()
        secret_key_client = client.export_secret_key()

        ciphertext, shared_secret_server = server.encap_secret(public_key_client)
        shared_secret_client = client.decaps_secret(ciphertext)

        print(shared_secret_client == shared_secret_server)
```

Obrázek 15: Kyber implementation by oqs in python

4.2.2 DILITHIUM

Dilithium je postkvantový kryptografický algoritmus pro digitální podpisy, založený na problémech mřížek (lattice-based). Byl vybrán NIST jako jeden z nových standardů díky své vysoké bezpečnosti a efektivitě. Využívá metodu Fiat-Shamir s odporem proti útokům kvantových počítačů. Dilithium nabízí rychlé ověřování podpisů, malé klíče a je vhodný pro širokou škálu aplikací v bezpečné komunikaci.

$$\mathbb{Z}_q = \{0, 1, \dots, q - 1\}$$

Má dost podobnou strukturu jako kyber. Zvolíme parametr q , matici A , kde $A_{ij} \in \mathbb{Z}_q$. Vybereme tajné vektory s_1 a s_2 z rozdělení $D_{\hat{\sigma}}$ (diskrétní Gaussovské rozdělení). A počítáme veřejný klíč.

$$t = As_1 + s_2 \bmod q$$

Kde t je veřejný klíč. Pak verifikace nebo podpis.

$$w = Ay \bmod q$$

Zvolíme y z D_∂ a spočítáme vektor w . Použijme Fiat-Shamir transformaci, kde hashovací funkce $H(w, M)=c$, c je hash zpráva s M a w .

$$z = y + cs_1$$

Je to odpověď, pokud z nesplňuje určité normové podmínky, proces se opakuje. A teď, Ověření podpisu:

$$w' = Az - ct \bmod q$$

Ověříme, zda w' je dostatečně blízko původní hodnotě w a zda z splňuje normová omezení. Pokud ano, podpis je platný.

4.2.3 FALCON

Falcon (*Fast Fourier Lattice-Based Compact Signatures*) je **kvantově odolný** algoritmus pro **digitální podpisy**, který byl vybrán NIST jako jeden z **tří standardizovaných algoritmů** pro postkvantovou kryptografii (v kategorii digitálních podpisů). Má nejmenší digitální podpis.

Algoritmus	Typ	Bezpečnost	Velikost podpisu	Rychlost ověřování
Falcon	Mřížová kryptografie	Vysoká	666 bajtů (Falcon-512)	Velmi rychlá
Dilithium	Mřížová kryptografie	Vysoká	1312 bajtů	Rychlá
SPHINCS+	Hash-based	Extrémní	17 kB	Pomalá

Tabulka 2: Tabulka PQC algoritmu pro digitální podpisy

Falcon využívá SVP a NTRU problém. Má definovanou mřížku \mathcal{L} .

$$\mathcal{L} = \left\{ \sum_{i=1}^n \lambda_i v_i \mid \lambda_i \in \mathbb{Z} \right\}$$

Kde v_i jsou základní vektory. Generace klíčů:

$$s, t \in \mathbb{Z}^n$$

My vybereme dvě n -dimenzionální R vektory. A to bude náš secret key.

$$A \in \mathbb{Z}^{n \times n}$$

A je to. Matice publickey klíč který je odvozovan z pomoci homomorfismů mřížek

Formálně se matice A počítá jako:

$$A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}$$

Kde A_1, A_2, A_3, A_4 jsou to bloky matic, které založeny na soukromých vektorech s a t . V praxi se vygenerují dvě základní mřížky pro každý vektor, které se poté kombinují do finální matice.

Pro každý podpis je vygenerován náhodný vektor r , který je *slabý* (krátký), tj. má malou **Euklidovskou** normu. Tento vektor je vygenerován z mřížky a hraje roli při vytvoření **podpisu**.

V rámci podpisového procesu se používá hash zprávy $h=H(m)$, kde H je kryptografická hashovací funkce, která překládá zprávu m do pevné délky (obvykle 256 bitů). Tento hash slouží jako základ pro výpočet podpisu.

Na základě náhodného vektoru r a soukromého klíče (s,t) se vypočítají dočasné hodnoty, které se používají k vytvoření samotného podpisu. Tyto hodnoty jsou modifikovány pomocí mřížky a zahrnují další aritmetiku, která je závislá na mřížkové struktuře.

Vytvořený podpis $\sigma=(z,w)$ je dvojice vektorů, která je založena na **mřížkových** operacích. K tomu se používá výše zmíněný vektor r , public key A , a private key (s,t) . Podpis je vytvořen jako řešení systému mřížkových rovnic:

$$\begin{aligned} z &= r + sh \pmod{q} \\ w &= th + r \pmod{q} \end{aligned}$$

A ověření podpisu:

$$Az \equiv w \pmod{q}$$

Falcon je docela bezpečný algoritmus ale ne tak jak ten který je na řadě...

4.2.4 SPHINCS

SPHINCS+ (Stateless Practical Hash-Based Signature Scheme) je post-kvantový algoritmus pro digitální podpisy založený na hašování, který nevyžaduje žádný stav pro generování nebo ověřování podpisů. Je postaven na **hashových funkcích** a využívá **Merkle strom** pro zajištění bezpečnosti. Tento systém je **stateless**, což znamená, že každý podpis je nezávislý na předchozím. SPHINCS kombinuje různé konstrukce hašovacích funkcí, jako je **Winternitzova metoda** a **Merkle hash tree**, čímž zajišťuje vysokou odolnost vůči kvantovým útokům. Díky tomu je **SPHINCS** považován za bezpečný i pro budoucí kvantové počítače.

Pro všechny operace budeme používat H jako **SHA-3**.

Generace klíčů:

$$S \in \mathbb{Z}^n$$

Na základě s si můžeme vygenerovat $H(s)$, což bude našim pubickým klíčem. Pro podpis zprávy m je použit následující postup. Zpráva m je rozdělena do menších bloků. Každý blok je následně zpracován pomocí hashovací funkce H , která generuje **komprimované** podpisy. Používá se Merkle strom pro vytváření stromové struktury, kde kořen stromu poskytuje autentizaci a ověření podpisu.

Pro ověření podpisu se použije veřejný klíč $H(s)$, hashovaná zpráva m a kořen **Merkleho** stromu.

První věc kterou musíme ověřit je že pro každou část podpisu se spočítá hodnota pomocí hash funkce a druhá věc srovnání, zda výsledný kořen **Merkleho** stromu odpovídá hodnotě uložené v podpisu.

$$H(m) \stackrel{?}{=} H_{root}(c)$$

Kde H_{root} je kořen **Merkleho** stromu a c je podpis. **SPHINCS** je odolný vůči kvantovým útokům díky své konstrukci, která je postavena na hašovacích funkcích. Bezpečnost SPHINCS vychází z obtížnosti problémů souvisejících s hašováním a Merkle **stromy**, které jsou považovány za výpočetně neefektivní i v přítomnosti kvantových počítačů.

5. Implementaci PQC

Pro dosažení normálního výkonu v **PQC** systémech je nezbytné využít HSM (Hardware Security Modules). Tyto zařízení poskytují silné kryptografické zajištění a ochranu klíčů proti útokům, včetně těch, které mohou přijít s kvantovými počítači. HSM zaručuje bezpečný a efektivní provoz i v **PQC** prostředí, čímž umožňuje organizacím chránit citlivé informace. A k tomu používat na to knihovny jako opencryptoki.. Implementaci má dvě části, web a VPN. Každý z nich bude mít implementaci **PQC** algoritmu a oni budou moct komunikovat na mezi sebou.

5.1. Web

První půlka je web. Technologie které se používá jsou popsane níž.

- Python, flask
- oqs, liboqs-python
- Kyber, Dilithium, Falcon, Sphincs a ještě na více...

Na webu je možnost práce se soubory, šifrování a decrypting jejich. Podepisování data a ověření jejich na základě různých algoritmů. V planu je přidat ukázkou rozbití **RSA** klíče z pomoci **shor** algoritmů. Takže je udělaný testový protokol komunikace z pomoci **Dilithium** který ale probereme na části VPN implementaci.

5.2. VPN

Druhá půlka je VPN. Technologie které se používá jsou popsane níž.

- Python, flask
- oqs, liboqs-python
- Dilithium a Falcon
- socket, threading
- ssl, hashlib

Mimochodem, generovat certifikáty na základě **Dilithium** už se začali. VPN má tunelování, správu připojení, **PQC** algoritmy, možnost změnit použité algoritmy na jiné a udělat benchmark spojení. Není to moc nadějně jako například **OpenVPN** nebo **WireGuard** ale jak ukázka je dobrá.

Závěr

Cílem této práce je demonstrovat praktickou implementaci post-quantových kryptografických algoritmů (**PQC**) a ukázat, jak mohou být tyto metody použity k zabezpečení komunikace a dat v éře kvantových počítačů. Kvantové počítače představují významnou hrozbu pro současné kryptografické systémy, jako jsou **RSA** a **ECC**, které jsou založeny na matematických problémech, jako je faktorizace čísel a diskretní logaritmus. Tyto problémy mohou být efektivně řešeny pomocí kvantových algoritmů, jako je **Shor's** algoritmus, což vede k nutnosti přechodu na post-quantové metody.

Práce se zaměřuje na implementaci a analýzu několika klíčových PQC algoritmů, které byly vybrány NIST jako kandidáti na nové standardy. Mezi tyto algoritmy patří **Kyber** (pro výměnu klíčů), **Dilithium**, **Falcon** a **SPHINCS+** (pro digitální podpisy). Tyto algoritmy jsou založeny na různých matematických problémech, jako jsou mřížkové problémy (lattice-based), hašovací funkce (hash-based) a problémy z teorie kódování (code-based), které jsou považovány za odolné vůči kvantovým útokům.

Dále práce obsahuje demonstraci prolomení klasických kryptografických algoritmů, jako je **RSA**, pomocí kvantových algoritmů, aby bylo zřejmé, jak jsou tyto metody zranitelné vůči kvantovým útokům. Cílem je ukázat, že přechod na **PQC** není pouze teoretickou nutností, ale praktickým krokem, který je třeba urychleně realizovat.

V rámci práce je vytvořena aplikace, která demonstruje použití těchto algoritmů v reálných scénářích, jako je šifrování souborů, digitální podpisy a zabezpečená komunikace přes VPN. Aplikace je rozdělena na dvě části: webové rozhraní pro práci se soubory a šifrování a VPN pro zabezpečenou komunikaci. Webová část umožňuje uživatelům šifrovat a dešifrovat data pomocí různých PQC algoritmů, zatímco VPN část demonstruje použití **PQC** pro zabezpečení komunikace mezi klientem a serverem.

Celkově tato práce přispívá k lepšímu pochopení PQC a poskytuje praktický návod, jak tyto metody implementovat do reálných systémů, čímž přispívá k budování bezpečnějšího digitálního prostředí v éře kvantových počítačů.

Seznamy

Literatury

1. Shor, P. W. (2002). Quantum Computing and Shor's Factoring Algorithm. *arXiv preprint* quant-ph/0205095. <https://arxiv.org/abs/quant-ph/0205095>
2. Lall, D., Agarwal, A., Zhang, W., Lindoy, L., Lindström, T., Webster, S., Hall, S., Chancellor, N., Wallden, P., Garcia-Patron, R., Kashefi, E., Kendon, V., Pritchard, J., Rossi, A., Datta, A., Kapourniotis, T., Georgopoulos, K., & Rungger, I. (2025). A Review and Collection of Metrics and Benchmarks for Quantum Computers: definitions, methodologies and software. *arXiv*. Retrieved from <https://arxiv.org/abs/2502.06717>
3. Bavdekar, R., Chopde, E. J., Agrawal, A., Bhatia, A., & Tiwari, K. (2023). Post Quantum Cryptography: A Review of Techniques, Challenges and Standardizations. 2023 *International Conference on Information Networking (ICOIN)*, 978-1-6654-6268-6/23/\$31.00 ©2023 IEEE. <https://doi.org/10.1109/ICOIN56518.2023.10048976>
4. Chauvet, J.-M., & Mahé, E. (2013). Secrets from the GPU. *arXiv*. Retrieved from <https://arxiv.org/pdf/1305.3699>
5. Galvis Florez, C. A., Martínez-Cifuentes, J., & Fonseca-Romero, K. M. (2023). Partial and complete qubit estimation using a single observable: optimization and quantum simulation. *arXiv*. Retrieved from <https://arxiv.org/abs/2301.11121>
6. Anakinai, A. (2019). Quantum Fourier Transform (QFT). *Agentanakinai*. Retrieved September 21, 2019, from <https://agentanakinai.wordpress.com/2019/09/21/quantum-fourier-transform-qft/>
7. Hillmich, S., Zulehner, A., & Wille, R. (2021). Exploiting quantum teleportation in quantum circuit mapping. *Proceedings of the 26th Asia and South Pacific Design Automation Conference (ASPDAC '21)*, 1-6. <https://doi.org/10.1145/3394885.3431604>
8. Lee, Y. H., Khalil-Hani, M., & Marsono, M. N. (2016). An FPGA-Based Quantum Computing Emulation Framework Based on Serial-Parallel Architecture. *International Journal of Reconfigurable Computing*, 2016, 5718124. <https://doi.org/10.1155/2016/5718124>
9. Mahajan, S., & Singh, M. (2014). Analysis of RSA Algorithm Using GPU Programming. *ResearchGate*. Retrieved from https://www.researchgate.net/publication/263736309_Analysis_of_RSA_algorithm_using_GPU_programming
10. Mahto, D., & Yadav, D. K. (2018). Performance analysis of RSA and elliptic curve cryptography. *International Journal of Network Security*, 20(4), 625-635. [https://doi.org/10.6633/IJNS.201807.20\(4\).04](https://doi.org/10.6633/IJNS.201807.20(4).04)
11. Liu, X., Li, D., Zheng, Y., Liu, M., Yang, X., Zhou, J., Tan, Y., & Wang, R. (2022). Quantum Information Splitting of an Arbitrary Five-Qubit State Using Four-Qubit Entangled States. *International Journal of Theoretical Physics*, 61, 220. Springer Nature. https://www.researchgate.net/publication/362736726_Quantum_Information_Splitting_of_an_Arbitrary_Five-Qubit_State_Using_Four-Qubit_Entangled_States#full-text

12. Stamp, D. F. The Mathematics behind RSA. *CS265 Security Engineering*. Citováno z: https://www.cs.sjsu.edu/~stamp/CS265/SecurityEngineering/chapter5_SE/RSAmath.html
13. Lomitzki, J. (2008). Řešení problému splnitelnosti booleovské formule (SAT) [Bachelor's thesis]. Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Computer Science. Retrieved from <https://users.fit.cvut.cz/~fiserp/dip/2008/Lomitzki/Bak.pdf>
14. DJP3. What is #P (sharp-P) complexity? Codex Caelestis. Citováno z: https://djp3.westmont.edu/content/codexcaelestis/archives/2005/04/what_is_p_sharp_1.html
15. PCWorld. The future of VPNs: Decentralized and post-quantum. Retrieved from <https://www.pcworld.com/article/2547145/future-of-vpns-decentralized-and-post-quantum.html>
16. National Institute of Standards and Technology (NIST). Compliance FAQs: Federal Information Processing Standards (FIPS). Retrieved from <https://www.nist.gov/standardsgov/compliance-faqs-federal-information-processing-standards-fips>
17. Microsoft. PQCrypto-VPN: Post-quantum Cryptography VPN. GitHub. Retrieved from <https://github.com/microsoft/PQCrypto-VPN>
18. Wikipedia contributors. Quantum Fourier transform. Wikipedia. Citováno z: https://en.wikipedia.org/wiki/Quantum_Fourier_transform
19. Wikipedia contributors. Quantum logic gate. Wikipedia. Citováno z: https://en.wikipedia.org/wiki/Quantum_logic_gate
20. National Institute of Standards and Technology (NIST). Post-Quantum Cryptography Round 3 Submissions. Retrieved from <https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions>

Obrázků

Obrázek 1: Kvantové brány	12
Obrázek 2: Hybrid quantum loop.....	13
Obrázek 3: Preparation state for four-qubit entangled state.....	14
Obrázek 4: Soubory problémů v matematice.....	15
Obrázek 5: Code for determinating of p and q.....	19
Obrázek 6: CUDA Architecture	20
Obrázek 7: QC algorithm example	21
Obrázek 8: Code for oracle of grover's algorithm.....	22
Obrázek 9: Grover's algorithm oracle circuit for finding state $ 111\rangle$	22
Obrázek 10: QFT implementation in qiskit	23
Obrázek 11: QFT circuit	23
Obrázek 12: Shor's algorithm for 5 qubits and 3 qubits in second register	24
Obrázek 13: Výsledek Shor's algoritmu na 5 qubitu	25
Obrázek 14: Grover's algoritmu na 3 qubity pro $ 111\rangle$ state	26
Obrázek 15: Kyber implementation by oqs in python	30

Tabulek

Tabulka 1: Kyber parametry.....	29
Tabulka 2: Tabulka PQC algoritmu pro digitální podpisy	31

Grafů

Graf 1: CPU a GPU porovnání	20
Graf 2: Not Optimized by GPU breaking of RSA and ECC with dependence on time	21
Graf 3: Output of Quantum Computer Heron 127 qubits	25

Příloh

Github projekt který má open source kod - <https://github.com/Glebgor/PQC>