Post-kvantová kryptografie

Ladislav Langer

Bakalářská práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav bezpečnostního inženýrství

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Ladislav Langer**
Osobní číslo:	**A20918**
Studijní program:	**B1032A020001 Bezpečnostní technologie, systémy a management**
Forma studia:	**Kombinovaná**
Téma práce:	**Post-kvantová kryptografie**
Téma práce anglicky:	**Postquantum Cryptography**

Zásady pro vypracování

1. Uveďte základní pojmy kvantových výpočtů.
2. Popište matematický aparát kvantových algoritmů.
3. Shrňte problematiku dnes používaných šifer v průmyslu komerční bezpečnosti a uveďte, které šifrovací algoritmy jsou zranitelné a které jsou naopak odolné útokům kvantových počítačů.
4. Vysvětlete a ilustrativně implementujte některé kvantové algoritmy využitelné pro odkrytí chráněné informace.

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

Seznam doporučené literatury:

1. GRUSKA, Jozef. Quantum Computing. London: McGraw-Hill, 1999. ISBN 978-0077095031.
2. JAŠEK, Roman, David MALANÍK a Nicol DAŇKOVÁ. Bezpečnost informačních systémů [online]. Druhé vydání. Zlín: Univerzita Tomáše Bati ve Zlíně, 2022 [cit. 2022-11-09]. ISBN 978-80-7678-088-0. Dostupné z: https://digilib.k.utb.cz/handle/10563/52082
3. HIDARY, Jack D. Quantum Computing: An Applied Approach. 2nd. ed. Cham, Switzerland: Springer, 2021. ISBN 978--3-030-83273-5.
4. BEČVÁŘ, Jindřich. Lineární algebra. Vydání páté. Praha: Matfyzpress, 2019. ISBN 978-807-3783-785.
5. BERNHARDT, Chris. Quantum computing for everyone. Cambridge, MA: The MIT Press, 2020. ISBN 978-0-262-53953--1.
6. YANOFSKY, N. S. a MANNUCCI, M. A. Quantum computing for computer scientists. Cambridge: Cambridge University Press, 2008. ISBN 978-0-521-87996-5.
7. LOREDO, Robert. Learn Quantum Computing with Python and IBM Quantum Experience: A hands-on introduction to quantum computing and writing your own quantum programs with Python. Birmingham, UK: Packt Publishing, 2020. ISBN 978-1-83898-100-6.

Vedoucí bakalářské práce: **Mgr. Jan Krňávek, Ph.D.**
Ústav matematiky

Datum zadání bakalářské práce: **16. prosince 2022**
Termín odevzdání bakalářské práce: **5. června 2023**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

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

Ing. Jan Valouch, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 16. prosince 2022

Prohlašuji, že

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

Prohlašuji,

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

Ve Zlíně, dne 25. května 2023 ..... Ladislav Langer v.r.

podpis studenta

ABSTRAKT

Jedním z hlavních problémů, které se vynořují s nástupem kvantových počítačů, je zranitelnost dnešních kryptosystémů. Dnes používané šifry jsou navrženy tak, aby byly odolné proti útokům klasickými počítači, ale v éře kvantových počítačů se tyto šifry stávají zranitelné a snadno prolomitelné. Práce demonstruje zranitelnost nejpoužívanějších šifer pomocí Shorova algoritmu.

Klíčová slova: kvantové výpočty, post-kvantová kryptografie, Shorův algoritmus, RSA šifrování

ABSTRACT

One of the main problems arising with the advent of quantum computers is the vulnerability of current cryptosystems. Today's ciphers are designed to be resistant to attacks by classical computers, but in the era of quantum computers, these ciphers become vulnerable and easily breakable. The thesis demonstrates the vulnerability of the most widely used ciphers using Shor's algorithm.

Keywords: quantum computing, post-quantum cryptography, Shor's algorithm, RSA encryption

S velkým uznáním a vděčností bych rád poděkoval svému vedoucímu,
Mgr. Janu Krňávkovi, Ph.D., za jeho cenné rady, trpělivost a vedení,
díky kterým jsem byl schopen úspěšně dokončit svou bakalářskou práci.

OBSAH

ÚVOD

Ochrana dat a elektronická komunikace jsou v éře počítačů a Internetu zajištěny pomocí kryptografických systémů, jejichž odolnost stojí na obtížných a časově náročných výpočtech. Prolomení šifry je zpravidla možné, nicméně vezmeme-li v úvahu technologickou, finanční a zejména časovou náročnost takového počinu, máme pádný důvod věřit v bezpečnost doporučených kryptografických standardů.

V roce 1994 publikoval Peter Shor přelomovou studii o schopnostech kvantových počítačů řešit některé složité matematické úlohy výrazně rychleji, než by to zvládaly klasické počítače. Ve své studii Shor poukázal na znepokojivý fakt, že soudobá nejpoužívanější kryptoschémata jsou vůči kvantovým počítačům zranitelná.

Je otázkou, do jaké míry je hrozba kvantových počítačů aktuální. Stále jsme v rané fázi vývoje, byť každý prvek potřebný pro existenci kvantového počítače byl experimentálně prokázán a za zmínku stojí i událost z roku 2019, kdy Google ohlásil dosažení kvantové nadvlády (z angl. Quantum supermacy), což znamená, že kvantový počítač vyřešil úlohu, kterou by klasický počítač nezvládl v rozumném čase. Konkrétně šlo o úlohu generování náhodných čísel v rámci speciálního výpočtu, kterou dokázal vyřešit kvantový počítač s procesorem Sycamore během 200 sekund a která by podle odhadů klasickému superpočítači trvala nejméně 10 000 let[1)].

S nástupem kvantových počítačů se v ruku v ruce vyvíjí obor post-kvantová kryptografie, který reflektuje, jak hrozbu ze strany kvantového počítače, tak i jeho potenciál ve využití kvantových jevů k zabezpečení komunikace.

Cílem této bakalářské práce je představit čtenářům principy dnešní kryptografie a konfrontovat je s poznatky Petera Shora.

V teoretické části si v úvodní kapitole vysvětlíme základní pojmy v kryptografii na šifrách, které se používaly před nástupem počítačů. V kapitole o moderním šifrování shrneme problematiku kryptografického zabezpečení dat a v kapitole o post-kvantové kryptografii popíšeme šifrovací algoritmy, které jsou považovány za odolné útokům kvantových počítačů.

Kapitola s názvem Matematický aparát vysvětluje sadu nástrojů zejména z oblasti lineární algebry, potřebných pro provádění operací v kvantových algoritmech. V této kapitole se čtenář může seznámit, jak využít Hilbertův prostor k abstrakci kvantového systému.

Závěrečná kapitola teoretické části rozšiřuje matematické nástroje o specifika kvan-

[1)] Zde dlužno dodat, že tím superpočítačem, kterému měl stejný problém trvat 10 000 let, byl Summit z dílny IBM. Když Google publikoval výsledky, v IBM se rozhodli vylepšit algoritmus tak, aby Summit úlohu zvládl za 2,5 dne. Téma o sporu mezi technologickými giganty zajímavě zpracoval Michael Kan v článku pro časopis PCMag.[1]

tového počítání. Vysvětlíme si co je to qubit a jak pracuje kvantový operátor.

V úvodu praktické části bude nastíněna problematika kvantových algoritmů a následně si předvedeme implementaci Shorova kvantového algoritmu pro faktorizaci čísel, který lze využít například k prolomení kryptosystému RSA. V rámci implementace Shorova algoritmu si popíšeme významné kvantové obvody a ověříme si jejich funkčnost na reálném kvantovém počítači.

Výstupem práce bude ucelený pohled na téma kvantových výpočtů v oblasti postkvantové kryptografie, včetně doporučení pro praxi v oblasti zabezpečení dat. Práce přispěje k lepšímu porozumění této problematiky a může být užitečná pro profesionály v oblasti informačních technologií a bezpečnosti dat.

V této práci je čerpáno převážně ze zahraniční literatury a jak už to v některých oborech bývá, tak nalezení vhodného překladu odborných termínů je problematické. Z toho důvodu je v textu často uváděn v závorce i původní anglický termín a v nezbytných případech, kde by český ekvivalent byl zavádějící je termín ponechán bez překladu.

I. TEORETICKÁ ČÁST

1 ZÁKLADY KRYPTOGRAFIE

V této kapitole je popsán princip kryptoschémat před nástupem počítačů. Na klasickém šifrování si snadněji vysvětlíme základní pojmy jak z oblasti kryptografie, tak kryptoanalýzy.

Primární prameny této kapitoly zahrnují zejména učebnici *Bezpečnost informačních systémů* [2] a Hoffsteinovu knihu *An introduction to mathematical cryptography* [3]. Doplňující informace a zajímavosti jsou čerpány z *Knihy kódů a šifer: tajná komunikace od starého Egypta po kvantovou kryptografii* [4] a inspirací pro příklady jsou informace z on-line kurzu *Cesta do světa kryptografie* v režii institutu Khan Academy [5].

1.1 Definice a základní pojmy obecné kryptografie

Kryptografie je věda, která se zabývá možnostmi utajování obsahu zpráv. Informace obsažené ve zprávě převádí do podoby, která je srozumitelná jen se speciální znalostí tzv. klíčem. Opakem kryptografie je kryptoanalýza, která se snaží rozluštit zašifrované zprávy bez znalosti klíče nebo samotné šifry. Kryptografie společně s kryptoanalýzou tvoří vědní obor kryptologie.

Šifrování je proces, který převede čitelnou zprávu (otevřený text) do nečitelné zašifrované podoby (šifrovaný nebo také šifrový text). Reverzní proces, při kterém získáme srozumitelnou zprávu ze zašifrovaného textu, se nazývá dešifrování. K transformaci, ať už z otevřeného textu na zašifrovaný text, nebo ze zašifrovaného na otevřený text potřebujeme v prvé řadě metodu - šifrovací, případně dešifrovací algoritmus a vstupní parametr algoritmu tajný klíč.

Používání klíčů je velmi praktické, neboť není třeba složitě tajit samotný šifrovací algoritmus, nýbrž stačí utajit klíč, který si lze představit například jako řetězec znaků představujících heslo k elektronické aplikaci, bez jehož znalosti se nelze ke službě přihlásit. Skrývání klíče namísto algoritmu je známo jako Kerckhoffsův princip.

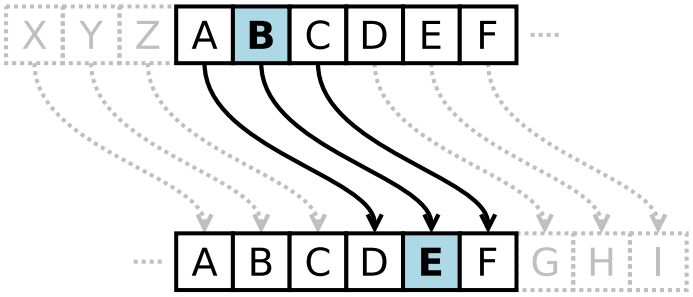
Podoba klíče má výrazně důležitější roli v šifrovacím algoritmu než složitost (počet kroků) algoritmu jako takového. Z čím větší množiny různých variant klíčů (klíčový prostor - z angl. Key space) si lze zvolit klíč, tím je časově náročnější kryptografickou ochranu prolomit.

Na příkladu posunovací a polyalfabetické šifry si vysvětlíme základní principy šifrování a výše zmíněné pojmy šifrovací algoritmus, klíč a klíčový prostor.

Poznámka: V této práci použijme zaběhlou terminologii pro popis kryptoschémat. Odesilatele zašifrované zprávy označme jako Alici, příjemcem bude Bob a Eva bude útočník, který zprávu odposlechne se záměrem zjištění obsahu zprávy.

1.2 Posunovací šifra

Posunovací šifra, též známá jako Caesarova šifra[1)], pracuje na principu záměny písmen ve zprávě a lze ji formálně definovat pomocí aritmetického operátoru modulo[2)]. Text šifruje pomocí klíče, jehož pevně daná hodnota udává počet pozic, o kolik se posune šifrované písmeno v abecedním pořadí (viz obr. 1.1).



Obrázek 1.1 Caesarova šifra - posun o tři pozice [6]

Adresát zašifrované zprávy (Bob) pomocí shodného klíče text dešifruje. V případě české abecedy (v základní sadě) může klíč nabývat celočíselných hodnot 0 až 25.

Jak probíhá šifrování:

1. Každému písmenu přiřadíme číselnou hodnotu x, které odpovídá jeho pořadí v abecedě ($a = 0, b = 1, c = 2, \ldots, z = 25$),
2. z množiny čísel $\{1, 2, \ldots 25\}$ zvolíme klíč K, který udává počet pozic o kolik se písmena posunou,
3. vypočteme hodnotu y pro každé písmeno ve zprávě, kterou chceme šifrovat pomocí $y = (x + K) \mod 26$,
4. převedeme hodnotu y zpět na písmeno v daném pořadí.

[1)] Šifra je pojmenována po Juliu Caesarovi, který ji používal pro vojenskou komunikaci a popsal ji v Zápiscích o válce galské.

[2)] Operátor modulo provádí matematickou operaci, která vrací zbytek po celočíselném dělení. Matematicky se značí jako $a \mod b$, kde a je dělenec a b je dělitel.

Například $17 \mod 5 = 2$. Když 17 dělíme 5, dostaneme 3 a zbytek 2.

Příklad 1.1 Alice a Bob si pro chráněnou komunikaci ujednají použití šifry s posunem o 15 znaků. $K = 15$. Alice zašifruje zprávu *OSEL* následovně:

	O	S	E	L		
	14	18	4	11		
+	15	15	15	15		
(	29	33	19	26	)	mod 26
	3	7	19	0		
	D	H	T	A		

Po aplikaci algoritmu posunovací šifry na slovo *OSEL* dostaneme slovo *DHTA*.

Jak probíhá dešifrování:

1. Aplikací obdobného postupů, kdy každému písmenu přiřadíme číselnou hodnotu y, které odpovídá pořadí písmena v abecedě ($a = 0, b = 1, c = 2, \ldots, z = 25$),
2. vypočteme hodnotu x pro každé písmeno v šifrované zprávě pomocí $x = (y - K) \mod 26$,
3. následně převedeme hodnotu x zpět na písmeno v daném pořadí.

Příklad 1.2 Bob dešifruje zprávu *DHTA* pomocí stejného klíče $K = 15$:

	D	H	T	A		
	3	7	19	0		
-	15	15	15	15		
(	-12	-8	4	-15	)	mod 26
	14	18	4	11		
	O	S	E	L		

Se znalostí správného klíče snadno dešifrujeme text zprávy. Pokud správný klíč neznáme, například z důvodu, že nejsme zamýšleným adresátem, pak se nabízí řešení v podobě vyzkoušení všech možných variant klíče, nebo-li hodnot, které může klíč nabývat. Tato množina, ze které klíč pochází, je právě již zmíněným klíčovým prostorem.

Pro bezpečnost každé šifry je podstatné, aby klíčový prostor byl dostatečně obsáhlý tak, aby při postupném zkoušení jednotlivých variant klíče nedošlo k prolomení šifry v rozumném čase.

Co přesně si lze přestavit pod pojmem časová odolnost kryptosystému si popíšeme dále v kapitole o moderním šifrování, nicméně na příkladu posunovací šifry je možné demonstrovat časovou náročnost prolomení šifry.

Posunovací šifra je příkladem jednoduché substituční šifry, kterou můžeme popsat pravidlem, resp. funkcí, která přiřadí každému písmenu z domény otevřeného textu jiné písmeno (případně jiný symbol) z oboru šifrového textu.

$$\{a, b, c, d, e, \ldots, x, y, z\} \rightarrow \{A, B, C, D, E, \ldots, X, Y, Z\}$$

Funkce produkuje šifrovou abecedu, která slouží jako klíč k šifrování a dešifrování zprávy (viz tab. 1.1).

Tabulka 1.1 Příklad šifrové abecedy.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
F	G	R	Y	P	W	B	Q	T	I	X	H	U	L	K	S	E	A	J	V	D	M	O	C	Z	N

Šifrový text je možné dešifrovat za podmínky, že funkce nepřiřadí žádným dvou písmenům otevřeného textu stejné písmeno v šifrovém textu. Pro funkci musí platit, že $\forall x, y \in \{a, b, c, d, e, \ldots, x, y, z\}, f(x) \neq f(y)$. O funkci s touto vlastností se říká, že je injektivní [3].

Nyní spočtěme, kolik šifrových abeced je možné pomocí této funkce vytvořit. Celkové množství získáme tak, že vyčíslíme všechny možné hodnoty šifrového textu pro každé písmeno otevřeného textu. Nejprve přiřadíme písmeno otevřeného textu a jednomu z 26 možných písmen šifrového textu $a - z$. Dále přiřadíme písmeno b jedné ze zbylých 25 možnosti, neboť nelze použít písmeno přiřazené k a. Pro písmena c až z postupujeme obdobně. Celkové množství variant je $26! = 403291461126605635584000000$.

Pokud by Eva zachytila šifrovanou zprávu určenou Bobovi a snažila by se pomocí počítače najít správný klíč v celém klíčovém prostoru substituční šifry, pak by pravděpodobně musela prozkoumat více než 10^{26} šifrových abeced. Počítač, schopný zkontrolovat jeden milion možných šifrových abeced za sekundu, by potřeboval více než 10^{13} let k prověření všech možností. V současné době se předpokládá stáří vesmíru v řádu 10^{10} let.

Výše popsaná metoda použitá k prolomení šifry je známá jako útok hrubou silou (z angl. Brute force attack). Metoda spočívá v postupném zkoušení všech možných variant klíče, případně hesla. Byť tento přístup nakonec vždy vede k úspěšnému odhalení klíče, je neefektivní a je používán pouze v případech, kdy neznáme žádné další informace o klíči nebo slabinách kryptosystému.

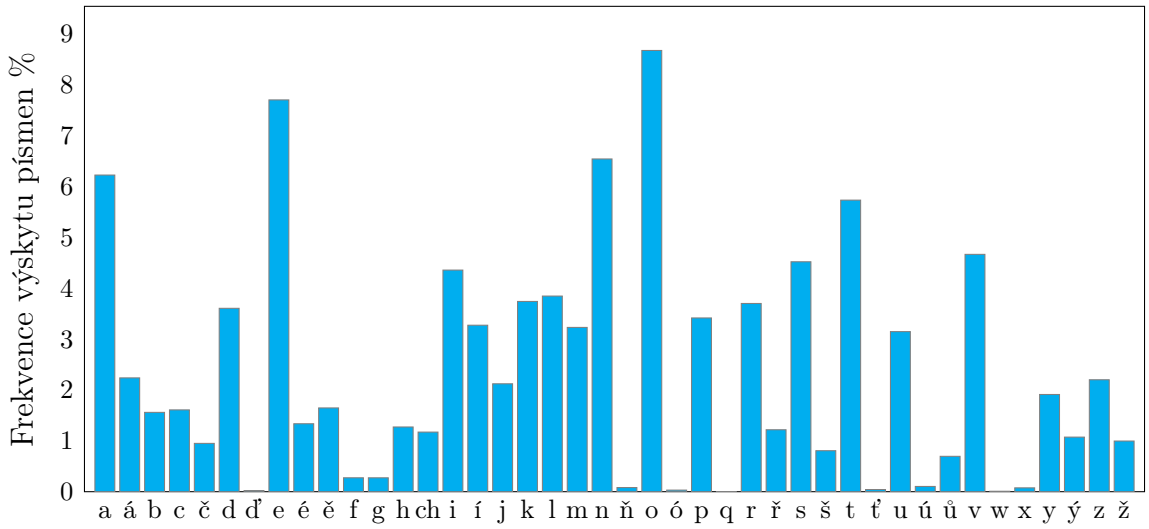
Z tohoto pohledu by se Alice a Bob mohli domnívat, že jejich tajná zpráva je v bezpečí. Nicméně v posunovací šifře je ve skutečnosti počet šifrových abeced limitován algoritmem posunutí o maximálně 25 pozic.

Efektivnější metody prolomení šifry zkoumá obor kryptoanalýza, která hledá elegantnější a účinnější cestu, jak rozluštit danou šifru. Každý kryptosystém je tak silný, jako jeho nejslabší část.

Hoffstein považuje kryptoanalýzu jako praktickou část kryptologie a dále k tomu uvádí, že Tvůj protivník vždy používá svou nejlepší strategii, aby tě porazil, ne tu strategii, kterou chceš, aby použil. Proto bezpečnost šifrovacího systému závisí na nejlepším známém způsobu, jak ho rozluštit. Jakmile jsou vyvinuty nové a vylepšené metody, jejich úroveň bezpečnosti může pouze klesat a nikdy se nebude zvyšovat [3].

Slabým místem posunovací šifry je kromě velmi malého počtu klíčů i možnost odhadu posunu písmen pomocí frekvenční analýzy jazyka zašifrovaného textu.

Porovnáme-li různé knihy psané stejnou řečí a spočítáme výskyt každého písmene v textu, tak pokaždé najdeme téměř shodný vzorec frekvence výskytu jednotlivých písmen, pomocí kterého vytvoříme histogram (viz obr. 1.2). Toto zobrazení nazýváme frekvenčním otiskem daného jazyka.



Obrázek 1.2 Frekvenční otisk češtiny. Data čerpána z [7]

K prolomení šifry stačí spočítat výskyt každého písmene v šifrovém textu a podívat se, o kolik pozic se frekvenční otisk jazyka posunul od otisku zašifrované zprávy.

Je zřejmé, že použitím statistické metody frekvenční analýzy nejen na posunovací šifru, ale i ostatní jednoduché substituční šifry se dobereme ke skrytému obsahu zašifrované zprávy poměrně rychle i bez použití výpočetní techniky.

Jakým způsobem je možné zvýšit odolnost kryptosystému, se podíváme v následující části této kapitoly.

1.3 Polyalfabetická šifra

Dalším příkladem substituční šifry je polyalfabetická šifra, též známa jako Vigenèrova šifra[3)]. Pracuje na stejném principu záměny písmen jako posunovací šifra, nicméně používá složitější formu klíče v podobě utajeného kódového slova[4)]. Kódové slovo reprezentuje sérii různých posunů, které periodicky aplikujeme na otevřený text.

Smysl polyalfabetických šifer spočívá v rovnoměrnějším výskytu písmen v šifrovém textu. Tím se výrazně oslabí možnost porovnání frekvenčních otisků jazyka otevřené zprávy a šifrového textu. Dále má polyalfabetická šifra možnost dostatečně rozšířit svůj klíčový prostor. Například použijeme-li kódové slovo (resp. několik slov) o délce 13 písmen, pak budeme mít celkem k dispozici $26^{13}(\approx 10^{18})$ klíčů. Ve své době platila Vigenèrova šifra za nerozluštitelnou šifru (z franc. *le chiffrage indéchiffrable*).

Jak probíhá šifrování:

1. Každému písmenu v otevřeném textu x a v kódovém slově K_i přiřadíme číselnou hodnotu, které odpovídá jeho pořadí v abecedě ($a = 0, b = 1, c = 2, \ldots, z = 25$),
2. číselné hodnoty kódového slova postupně a periodicky přičítáme k příslušné hodnotě otevřeného textu,
3. vypočteme hodnotu y pro každé písmeno ve zprávě, kterou chceme šifrovat pomocí $y = (x + K_i) \mod 26$,
4. převedeme hodnotu y zpět na písmeno v daném pořadí.

Příklad 1.3 Alice se předem domluví s Bobem, že k šifrování použijí kódové slovo *OLIVA*, které reprezentuje sérii posunů 14, 11, 8, 21 a 0. Zprávu *SEJDEME SE V JEDNU* zašifrujeme následovně:

	S	E	J	D	E	M	E	S	E	V	J	E	D	N	U		
	18	4	9	3	4	12	4	18	4	21	9	4	3	13	20		
+	14	11	8	21	0	14	11	8	21	0	14	11	8	21	0		
(	32	15	17	24	4	26	15	26	25	21	23	15	11	34	20	)	mod 26
	6	15	17	24	4	0	15	0	25	21	23	15	11	8	20		
	G	P	R	Y	E	A	P	A	Z	V	X	P	L	I	U		

Pro dešifrování textu *GPRYEAPAZVXPLIU* s pomocí stejného kódového slova použijeme obdobný postup jako u posunovací šifry.

[3)] Prvně šifru publikoval italský kryptograf Giovan Battista Bellaso ve svém díle *La cifra del Sig* v roce 1553, nicméně po staletí byla připisována francouzskému kryptografovi Blaise de Vigenèrovi, který podobnou šifru zveřejnil až v roce 1586. [8]

[4)] Rozdíl mezi kódováním a šifrováním spočívá v tom, že kódování se zaměřuje na snadnější přenos informací a neslouží k ochraně informací před neoprávněným přístupem.

Polyalfabetická šifra může využít všech 26 různých šifrových abeced a tím dramaticky prodloužit potřebný čas k rozluštění šifry pomocí útoku hrubou silou.

Odolnost proti prolomení polyalfabetické šifry je dána délkou kódového slova. Čím delší kódové slovo se použije pro šifrování, tím bezpečnější bude výsledná šifra. Na druhou stranu Bob a Alice si snáze zapamatují krátké kódové slovo než dlouhé. Zde vidíme začátek věčného boje v praktické (a ne pouze teoretické) kryptografii, a to boje mezi účinností (a jednoduchostí použití) a bezpečností. Toto dilema je také vlastní celé řadě moderních kryptosystémů.

Existují kryptoanalytické nástroje, které mohou být použity k prolomení polyalfabetické šifry. Například lze zmínit Kasiskiho test [9] a nebo Friedmanovu metodu [10]. Tyto nástroje vychází z frekvenční analýzy a proto je důležité si uvědomit, že účinnost těchto nástrojů závisí na složitosti šifry a na tom, zda máme dostatečné množství kódované zprávy k dispozici pro analýzu.

Prvním krokem k prolomení polyalfabetické šifry je určení délky kódového slova. Také někdy mluvíme o hledání délky bloku a nebo periody.

Kasiskiho test hledá opakující se fragmenty v šifrovém textu a zaznamenává vzdálenost s jakou se v textu opakují. Zanedbatelné množství opakujících se fragmentů je dílem náhody, nicméně pro značnou část fragmentů bude platit, že jejich vzdálenosti v textu jsou beze zbytku dělitelné délkou kódového slova.

Friedmanova metoda spočívá v analýze indexu koincidence, který měří, jak velká je pravděpodobnost, že dva náhodně vybrané znaky v šifrovém textu budou shodné. Při použití polyalfabetické šifry, kde se pro šifrování používá více než jedna šifrová abeceda, bude index koincidence větší než u monoalfabetické posunovací šifry. Pro nalezení délky klíče se Friedmanova metoda obvykle zaměřuje na frekvenci výskytu shodných dvojic písmen v šifrovém textu. Tento postup může být založen na předpokladu, že pokud jsou dvě shodné dvojice písmen odděleny konstantním počtem znaků, pak tyto dvojice byly pravděpodobně šifrovány stejným znakem z klíče.

Jakmile zjistíme délku kódového slova pak v druhém kroku stačí rozdělit šifrový text do počtu bloků odpovídajícím právě délce kódového slova. Následně text každého bloku stačí porovnat s frekvenčním otiskem jazyka pro zjištění posunu v daném bloku jako u posunovací šifry.

Síla polyalfabetické šifry spočívá v čase, který je potřebný na zjištění délky kódového slova a jak již bylo řečeno, čím delší je kódové slovo, tím silnější je šifra.

V následující kapitole se podíváme na případ, kdy bude délka kódového slova stejná jako délka otevřeného textu.

1.4 Vernamova šifra

Jednorázová tabulková šifra (OTP z angl. One-time pad), také známá jako Vernamova šifra[5)] pracuje na principu náhodného posunutí každého písmene zprávy. Jinými slovy kódové slovo používané v polyalfabetické šifře je nahrazeno číselným řetězcem náhodných hodnot posunutí o délce počtu znaků v otevřené zprávě.

Šifra je bez znalosti klíče nerozluštitelná za předpokladu dodržení těchto dvou podmínek:

1. Klíč je stejně dlouhý jako zpráva.
2. Klíč je dokonale náhodný (což implikuje další podmínku - klíč lze použít pouze jednou, jinak by už nebyl dokonale náhodný).

Zatímco první podmínka kompletně eliminuje frekvenční otisk v šifrovém textu, neboť zajistí rovnoměrný výskyt písmen, tak obě podmínky v součinnosti ochrání Vernamovu šifru před útokem hrubou silou nehledě na dostupný výpočetní výkon.

Na jednoduchém příkladu si ukážeme, proč nehraje roli výpočetní výkon při prolamování OTP šifry.

Příklad 1.4 Nejprve zašifrujme slovo *OSEL* pomocí algoritmu posunovací šifry postupně se všemi variantami klíče K.

K	O	S	E	L
1	P	T	F	M
2	Q	U	G	N
3	R	V	H	O
⋮	⋮	⋮	⋮	⋮
25	N	R	C	K

Celkový počet podob, do kterého je možné slovo *OSEL* transformovat, je roven počtu prvků v klíčovém prostoru. Pro šifrový text (i velmi krátký) je pro daný jazyk platná (mající význam v otevřeném textu) zpravidla pouze jediná varianta klíče.

Nyní zkusme stejné slovo *OSEL* šifrovat pomocí algoritmu Vernamovy šifry. K tvorbě klíče v našem případě můžeme například použít 26 stěnou hrací kostku. S každým hodem získáme náhodný posun pro jednotlivá písmena. Víme, že počet variant, do kterého lze zprávu zašifrovat je roven počtu prvků v klíčovém prostoru. Velikost této množiny pro zprávu o čtyřech znacích je rovna $\mathcal{K} = 26^4$.

[5)] Inženýr společnosti AT&T Gilbert S. Vernam v roce 1918 navrhl pro telegrafní spojení šifrování s použitím náhodné série značek a mezer (klíč) promíchané se zprávou. [8]

Ke slovu *OSEL* máme k dispozici 456776 variant klíče. S pomocí počítače útok hrubou silou ve velmi krátkém čase (pod 1ms) prověří všechny možné kombinace klíče včetně použitého tajného klíče, který šifrovanou zprávu převede na původní text *OSEL*, ale současně další podoby klíče dešifrují text na slova *OREL, VOSA, HUSA,...* a všechna ostatní čtyř písmenná slova, z čehož útočnice Eva nedovodí původní text zprávy.

Matematický důkaz o nerozluštitelnosti Vernamovy šifry publikoval Claude E. Shannon v roce 1949 v *Bell System Technical Journal* [11]. Shannon prokázal, že tabulková jednorázová šifra má vlastnost dokonalého utajení (z angl. Perfect secrecy). To znamená, že šifrovaný text neposkytuje žádnou informaci o otevřeném textu (s výjimkou jeho délky). S dokonale náhodným klíčem lze šifrový text převést na libovolný otevřený text o stejné délce a všechny varianty převodu jsou stejně pravděpodobné.

Dílčí závěr

Na příkladu posunovací a polyalfabetické šifry jsme si vysvětlili základní princip substitučních šifer, roli klíče a klíčového prostoru ve vztahu k bezpečnosti kryptosystému. Dále jsme na příkladech popsali některé statistické metody frekvenční analýzy a význam frekvenčního otisku jazyka v kryptoanalýze. U OTP šifry jsme vysvětlili princip dokonalého utajení. Tyto pojmy nám poslouží jako úvod do problematiky moderní kryptografii a posléze kvantových algoritmů, kde například obdobně hledáme periodu ve funkcích, tak jako u polyalfabetické šifry.

2 MODERNÍ ŠIFRY

Tato kapitola se zabývá současnými šiframi, které se používají k ochraně digitálních informací a komunikace. Nejprve se zaměříme na obecné aspekty moderního šifrování, jako jsou například generátory pseudonáhodných čísel, asymptotická složitost algoritmů anebo význam logické funkce XOR pro elektronické šifrování. Dále se budeme zabývat symetrickými a asymetrickými šiframi. Probereme směry útoku na používané šifry a zejména útok pomocí kvantových počítačů.

Kapitola se opírá převážné o informace, které jsou obsaženy v Hoffsteinově knize *An introduction to mathematical cryptography* [3] a v knize *Počítačová algebra* od Stanovského [12]. Další odkazované zdroje v kapitole jsou z učebnic *Bezpečnost informačních systémů* [2] a *Programování* od Králíka [13].

2.1 Aspekty moderního šifrování

Je téměř jisté, že každý z nás využívajíce Internet a různé on-line aplikace se na denní bázi setkává s kryptografickými systémy, které kdesi na pozadí chrání naše citlivá data. Spolehlivost používaných šifer je pro nás čím dál více důležitější s tím, jak se naše životy přesouvají do on-line prostoru.

Dnešní kryptografické systémy v elektronické komunikaci nezajišťují pouze šifrování přenášených dat, ale plní další nezbytné úkoly k zajištění bezpečné komunikace. Hlavní pilíře elektronické výměny informací jsou:

- **Důvěrnost** zaručuje, že přenášená data jsou tajná a mohou být čtena pouze autorizovanými osobami.
- **Autenticita** zajistí, že osoba nebo systém, který přenáší nebo přijímá data, je tím, za koho se vydává.
- **Integrita** zajišťuje, že data nebyla po cestě záměrné změněna, nebo poškozena.

K těmto účelům se zejména využívají nástroje hašovací funkce a elektronické podpisy.

Hašovací funkce jsou matematické algoritmy, které přijímají vstupní data libovolné délky a generují z nich výstupní řetězec s pevně danou délkou, který se nazývá hash. Tyto algoritmy mají specifické vlastnosti. Jsou jednosměrné a jednoznačné. To znamená, že vytvoření hashe je rychlé, ale zpětné získání původní informace je extrémně těžké a zároveň je velice nepravděpodobné, že by dva různé dokumenty vytvořily shodný hash.

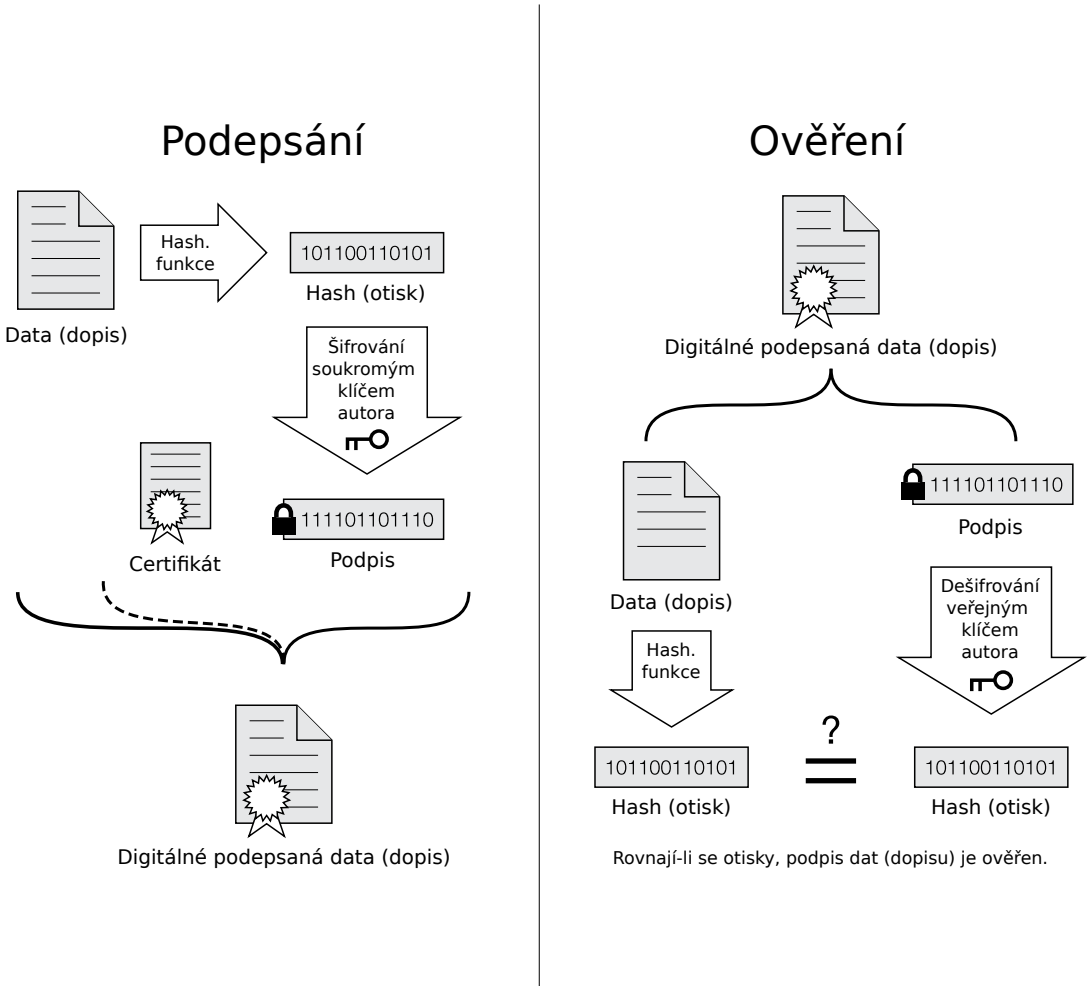
Elektronické podpisy využívají zajímavý kryptografický mechanismus, tzv. asymetrické šifrování, kde uživatelé mají soukromý klíč (private key), který je tajný, a veřejný klíč (public key).

Proces podepsání dokumentu zahrnuje kroky:

1. Generování hash hodnoty zprávy nebo dokumentu pomocí hašovací funkce.
2. Zašifrování hash hodnoty pomocí soukromého klíče podepisující strany.

Proces ověřování dokumentu zahrnuje kroky:

1. Dešifrování zašifrovaného hash pomocí veřejného klíče podepisující strany.
2. Porovnání obou hash hodnot. Pokud se shodují, je podpis platný a zpráva nebo dokument nebyl pozměněn.



Obrázek 2.1 Diagram elektronického podpisu [14]

2.1.1 Algoritmická složitost

Algoritmus[1)] je definován jako omezená sekvence kroků, která řeší daný problém. Algoritmy nejsou vlastní pouze strojům a neomezují se pouze na matematiku. Christian v knize *Algoritmy pro život* [15] poukazuje na fakt, že algoritmem je i běžný výrobní návod, například kuchyňský recept a datuje používání algoritmů již do doby kamenné.

V informatice klademe na kvalitu algoritmu určité nároky. Obecně každému počítačovému algoritmu je vlastní:

- **Obecnost** předpokládá, že algoritmus neřeší pouze jeden konkrétní problém, ale celou třídu daného problému (např. součin dvou čísel $a \cdot b$, nikoliv pouze $3 \cdot 4$).
- **Jednoznačnost** přiřazuje stejnému vstupu algoritmu za stejných podmínek odpovídající stejný výstup.[2)]
- **Konečnost** znamená, že algoritmus musí skončit v určitém počtu kroků.

Kritériem kvalitního algoritmu určitě není nejnižší možný počet kroků, ale poskytnutí výstupu v rozumném čase.

Dobu potřebnou pro běh algoritmu určuje hardwarový výkon (rychlost zpracování instrukcí na procesoru) a složitost algoritmu, která se uvádí v počtu jednotkových operací N v závislosti na velikosti vstupu n. Jednotkovou operaci a velikost vstupu je třeba specifikovat pro daný problém. Například jednotková operace pro celočíselné výpočty je zpravidla operace, která se provádí na jednom čísle o jedné cifře. Při výpočtech se může použít libovolná jednoduchá operace, která bude prováděna na každé jedné cifře vstupu. Velikost vstupu se určuje podle počtu cifer, které jsou v čísle.

Pro vyčíslení složitosti algoritmu se používá asymptotický zápis funkcí s $\mathcal{O}$-notací[3)] (viz obr.2.2).

[1)] Slovo Algoritmus je odvozeno od jména perského matematika Muhammada ibn Musa al-Khwarizmiho (lat. Al-Gorizmí), který žil v 9. století. Al-Khwarizmi napsal knihu, ve které popisoval postupy pro výpočty polynomických rovnic. Kniha nese název *al-Kitāb al-Mukhtasar fī Hisāb al-Jabr wal-Muqābalah* (lat. Liber Algebræ et Almucabola), kde ze slova *al-Jabr* byla odvozena algebra. I přesto, že jméno al-Khwarizmiho je spojováno s vynálezem algoritmu, existují důkazy, že matematické algoritmy byly známy již dávno před tímto perským matematikem. Dokazuje to například nález hliněné tabulky popisující schéma pro nezkrácené dělení z období Sumerské říše.

[2)] Říká se, že každé pravidlo má nějakou výjimku. V tomto případě pro šifrování velmi důležitou. U algoritmů pro generování náhodných čísel se naopak snažíme dosáhnout co nejméně předvídatelných výsledků.

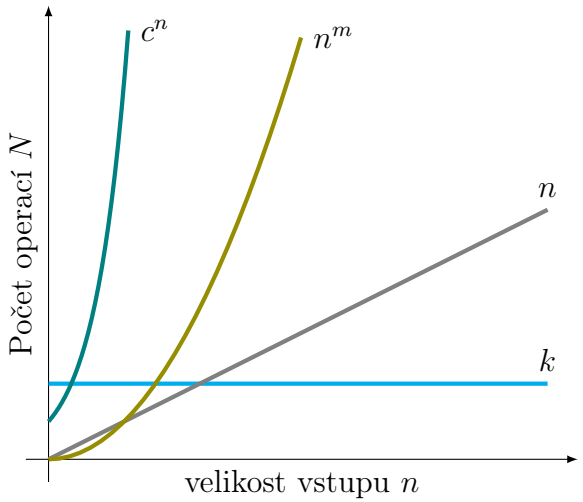
[3)] Je důležité poznamenat, že $\mathcal{O}$ (zvaná big-O notace) se používá pro horní mez složitosti algoritmu. Jinými slovy jde o nejhorší možný scénář, který algoritmus může v daném případě řešit.

Stanovský definuje asymptotickou složitost následovně:

Definice 2.1 [12] Nechť $f, g : \mathbb{N}^k \to \mathbb{R}^+$ jsou k-ární funkce.

- Píšeme $f = \mathcal{O}(g)$ (a říkáme, že f je *asymptoticky menší nebo se rovná* g), pokud existuje konstanta C taková, že všechna $x_1, \ldots, x_k \in \mathbb{N}$
$$f(x_1, \ldots, x_k) \leq C \cdot g(x_1, \ldots, x_k) \tag{2.1}$$

- Píšeme $f = \Theta(g)$, pokud $f = \mathcal{O}(g)$ a $g = \mathcal{O}(f)$. Jde o relaci ekvivalence, tedy potom také platí, že $g = \Theta(f)$.



Obrázek 2.2 Graf časové složitosti.

Z asymptotického chování algoritmu je možné odhadnout, jak se změní jeho náročnost (resp. časová složitost[4)]) v závislosti na velikosti vstupu. Například pokud vstup zvětšíme desetkrát, očekáváme, že algoritmus s lineární složitostí poběží desetkrát déle, zatímco algoritmus s kvadratickou složitostí poběží stokrát déle a u algoritmu s exponenciální složitostí se výsledku pravděpodobně nikdy nedočkáme.

Na grafu (obr.2.2) je zobrazena náročnost algoritmu pro velikost vstupu konstantní $\mathcal{O}(k)$, lineární $\mathcal{O}(n)$, polynomiální $\mathcal{O}(n^m)$ pro nějaké pevné $m > 1$ a exponenciální $\mathcal{O}(c^n)$ pro nějaké pevné $c > 1$.

[4)] Informatika také pracuje s pojmem časová složitost algoritmu, která popisuje, kolik času trvá algoritmu, aby vyřešil určitý problém pro danou velikost vstupu. Časová složitost se obvykle měří v počtu kroků, operací nebo v sekundách, které jsou potřebné k vyřešení problému.

Zjednodušeně řečeno, časová složitost popisuje skutečný čas, který algoritmus potřebuje k vykonání pro daný vstup, zatímco asymptotická složitost popisuje, jak rychle se zvětšuje časová složitost s rostoucí velikostí vstupu.

2.1.2 Generátor pseudonáhodných čísel

Faktor náhody hraje významnou roli v bezpečnosti většiny moderních kryptoschémat. Dokonce i populární deterministický kryptosystém RSA (podrobněji se mu budeme věnovat v kapitole 2.3.2) využívá náhodná čísla k posílení bezpečnosti.

Ideálně bychom potřebovali zařízení, nebo algoritmy, který by generovaly náhodný výstup jedniček a nul. Byť kvantová teorie existenci takového zařízení, které by generovalo řetězce skutečně náhodných čísel připouští (a v nedávné době již některá zařízení spatřila světlo světa), tak v praxi si budeme muset vystačit se softwarovým řešením, které nám poskytne zdánlivě náhodný výstup číslic, neboť získávání dostatečně náhodných čísel z obtížně předvídatelných dějů (např. odečítáním rádiového, nebo termálního šumu) je pomalé a nepraktické.

Tyto algoritmy resp. funkce nazýváme generátory pseudonáhodných čísel PRNG (z angl. pseudo random number generator). Hoffstein [3] poukazuje na kontradikci v názvu, neboť výstup funkce PRNG není vůbec náhodný, naopak je zcela determinován vstupem.

Vstupní datový řetězec pro inicializaci generátoru, který slouží jako základ pro generování sekvence pseudonáhodných čísel se nazývá *seed*. Kvalitní seed reprezentuje řetězec náhodných čísel. Některé PRNG algoritmy zvyšují kvalitu výstupu, tím že zahrnují do výpočtů externí zdroj entropie, kterým může být například systémový čas z CPU.

U PRNG funkcí velikost seedu (počet číslic v řetězci) určuje po jaké době se sekvence výstupu pseudonáhodných čísel začne opakovat, nebo-li dosáhne své periody.

Ukažme si použití seedu na příkladu s využitím von Neumannovy metody Middle square [16] pro generování pseudonáhodných čísel.

Middle Square metoda je jednoduchý PRNG, který generuje nová čísla tím, že vezme střední číslice z druhé mocniny předchozího čísla.

Příklad 2.1 Zde je příklad s použitím seedu o $n = 4$, kde n je sudé číslo a představuje počet číslic v seedu:

1. Zvolíme seed (počáteční hodnotu): $x_0 = 0123$
2. Spočítáme kvadrát seedu: $123^2 = 15129$ a doplníme před výsledek počet nul, tak aby výsledné číslo mělo $2n$ číslic.
3. Vybereme střední číslice výsledku: 00015129
4. Nové číslo x_1 je: 0151
5. Opakujeme kroky 2 až 4 pro další čísla v sekvenci.

Po 50 opakováních dostaneme sekvenci pseudonáhodných čísel 0151 0228 0519 2693 2522 3604 9888 7725 6756 6435 4092 7444 4131 0651 4238 9606 2752 5735 8902 2456 0319 1017 0342 1169 3665 4322 6796 1856 4447 7758 1865 4782 8675 2556 5331 4195 5980 7604 8208 3712 7789 6685 6892 4996 9600 1600 5600 3600 9600 1600.

Všimnete si, že sekvence po 49 kroku začíná opakovat vygenerované hodnoty a zjevně přestává plnit svoji roli statistické nerozlišitelnosti od skutečně náhodných čísel.

Kryptografické systémy využívají funkce CSPRNG (z angl. cryptographically secure PRNG), které splňují podmínky:

- **Nepředvídatelnost** znamená, že nelze předpovědět generovaná čísla ani při znalosti předchozích čísel v sekvenci, pokud neznáme seed. Přesněji řečeno, neměl by existovat rychlý (např. polynomiální časový) algoritmus, který by dokázal předpovědět další bit s více než 50% šancí na úspěch.
- **Odolnost proti zpětnému inženýrství** zajišťuje, že CSPRNG jsou odolné vůči pokusům o zjištění interního stavu generátoru nebo seedu na základě znalosti generovaných čísel.

V kryptografii jsou CSPRNG využívány zejména pro generovaní klíčů, inicializačních vektorů, nonce a dalších kryptografických prvků.

Inicializační vektor (IV) je unikátní hodnota, která se používá při šifrování dat pomocí blokových šifer, kterým se věnuje kapitola o blokových a proudových šifrách. Role inicializačních vektorů je zabránit opakujícím se šifrovým zprávám při šifrování stejného textu. Příkladem mohou být hlavičky různých protokolů, které se neustále v datové komunikaci opakují.

Nonce (zkratka z angl. number used once) je jednou použité číslo v kryptografických protokolech, které zahrnují autentizaci, digitální podpisy nebo jednorázová hesla a slouží pro zajištění integrity a ochrany proti tzv. replay útokům[5)], tím že zaručí jedinečnost každé zprávy. Na rozdíl od IV nemusí být nonce vždy náhodné a v některých kryptosystémech je pevně dané.

2.1.3 XOR

V každém počítačovém kryptografickém systému pracujeme s daty a symboly jenž jsou vyjádřeny pomocí binárních čísel, proto je vhodné se v krátkosti podívat na operace s jednotlivými bity.

[5)] Replay attack je druh kybernetického útoku, kde útočník zachytává platnou zprávu, šifrovaný text nebo autentizační token při komunikaci mezi dvěma stranami a poté tuto zprávu nebo token opakovaně přeposílá, aby získal neoprávněný přístup nebo narušil integritu komunikace.

Pro šifrování dat je důležitá logická funkce Exkluzivní disjunkce, značená XOR (z angl. exclusive OR), nebo symbolem $\oplus$. Operace XOR patří mezi nejrychlejší, neboť zpravidla bývá hardwarově implementována přímo v procesoru.

V úvodní kapitole Základy kryptografie jsme se podrobně zabývali takřka historickými šiframi, jejichž princip substituce znaků je společný i modernímu šifrování v počítačích. Substituční šifry matematicky těží z množství znaků, se kterými pracují, přesto na Vernamově šifře, která se používala pro telegrafní zprávy složené pouze z teček a čárek byl popsán princip dokonalého utajení. Na obdobném principu funguje šifrování v počítači pomocí bitové operace XOR.

Tabulka 2.1 Pravdivostní tabulka

A	B	$A \wedge B$	$A \vee B$	$A \oplus B$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Máme-li zprávu M a klíč K, jež jsou různé řetězce binárních dat, pak šifrový text C je výsledkem aplikace XOR na páry bitů složených z jednotlivých řetězců.

Příklad 2.2 $M = 100111$, $K = 110010$

$$C = 100111 \oplus 110010 = [1 \oplus 1]\,[0 \oplus 1]\,[0 \oplus 0]\,[1 \oplus 0]\,[1 \oplus 1]\,[1 \oplus 0] = 010101 \qquad (2.2)$$

Pro bitovou operaci XOR platí:

$$e_k(M) = K \oplus M \quad \text{a} \quad d_k(C) = K \oplus C. \qquad (2.3)$$

kde šifrovací funkce e_k je shodná s dešifrovací funkcí d_k. Tato reverzibilita je podstatná vlastnost operace XOR. Pokud bychom použili operaci logickou konjunkci (symbol AND, nebo $\wedge$) namísto XOR, tak by nebylo možné šifrový text jednoznačně dešifrovat, neboť by při šifrování docházelo ke ztrátě informace. V případech, kdy $C = 0$ a $K = 0$ nelze při dešifrování dovodit původní hodnotu M. Obdobné platí i pro pro logickou disjunkci (symbol OR, nebo $\vee$).

2.1.4 Blokové a proudové šifry

Blokové a proudové šifry jsou dvě hlavní kategorie symetrických šifer používajících stejný klíč jak pro šifrování, tak dešifrování zpráv. Obě kategorie mají své výhody a nevýhody. V praxi se často zvažují protichůdné požadavky na výkon a odolnost, nebo-li efektivitu a bezpečnost kryptosystému (viz tab 2.2).

Proudové šifry šifrují data bit po bitu nebo byte po bytu. Tyto šifry pracují tak, že pomocí PRNG generují proud klíče a inicializační vektory. Proud klíče je poté kombinován s daty (často pomocí operace XOR) jak pro šifrování, tak pro dešifrování. Proudové šifry se zpravidla zaměřují na rychlost a snadnou implementaci.

Blokové šifry šifrují data ve formě bloků pevné délky. Vstupní data jsou rozdělena do bloků, které jsou zašifrovány nezávisle na sobě pomocí klíče s pevnou délkou. Blokové šifry většinou používají iterativní proces, kde se provádí několik kol šifrování na každém bloku.

Tabulka 2.2 Porovnání blokových a proudových šifer

Blokové šifry	
Výhody	Nevýhody
Poskytují vysokou úroveň bezpečnosti při správném použití.	Vyžadují více systémových zdrojů (čas na CPU a paměť) než proudové šifry.
Lze použít různé operační módy pro zvýšení odolnosti proti replay útokům a celkové bezpečnosti.	V některých operačních režimech je nutné požít navíc inicializační vektory.
Jsou vhodné pro šifrování velkých objemů dat.	
Příklady blokových šifer: AES, DES, 3DES, Blowfish, IDEA	
Proudové šifry	
Výhody	Nevýhody
Jsou obvykle rychlejší a méně náročné na zdroje než blokové šifry.	Jsou citlivé na chyby v přenosu, protože chyba v jednom bitu šifrovaných dat může vést k chybě v dešifrovaném výstupu.
Jsou vhodné pro šifrování dat s proměnlivou délkou. Například pro streamování dat.	Neumožňují použití bezpečnějších operačních módů.
Příklady proudových šifer: RC4, Salsa20, ChaCha20, A5/1, A5/2	

V rámci porovnání blokových a proudových šifer můžeme zmínit další kryptografický mechanismus. Doposavad jsme v práci probírali kryptoschémata, která pracují se substitucí, kdy se podle určitého pravidla nahradí prvek (například písmeno, nebo bit) jiným prvkem. Kromě substituce využívá kryptografie také permutace, kdy se mění pořadí prvků ve zprávě.

Proudové šifry využívají pouze substituci, neboť zpravidla pracují s dynamickým rozsahem dat, kde například není dopředu známo jestli se bude šifrovat 50 znaků, nebo 50 stránek textu. Na druhou stranu blokové šifry využívají v rámci bloku šifrování pomocí substituce, která může být doplněna o permutaci, a tím se může výrazně zvýšit bezpečnost kryptosystému.

V tabulce (tab. 2.2) je mezi výčtem výhod blokových šifer uvedena možnost použití

různých operačních módů. Operační mód u blokových a proudových šifer určuje jakým způsobem se šifra aplikuje na data otevřené zprávy.

V praxi se používá několik operačních režimů pro blokové a proudové šifry, které jsou navrženy tak, aby splňovaly různé bezpečnostní a provozní požadavky. Burda [17] řadí mezi nejběžněji používané operační režimy:

- Režim kódové knihy ECB (z angl. Electronic Codebook) šifruje každý blok nezávisle na ostatních, což vede k jeho zranitelnosti vůči replay útokům s opakujícími se vzory. ECB režim se v praxi shoduje s proudovým šifrováním.

- Řetězení šifrových bloků CBC (z angl. Cipher Block Chaining) následující blok dat před šifrováním XORuje s předchozím zašifrovaným blokem. Tím se zajišťuje, že zašifrované bloky jsou závislé na všech předchozích blocích, což na jedné straně implikuje vyšší bezpečnost a na druhé nevýhodu, že v tomto režimu nelze šifrování paralelizovat. První blok se XORuje s inicializačním vektorem (IV).

  Matematicky lze operační režim pro šifrování popsat rovnicí:

  $$\begin{aligned} C_i &= e_k(M_i \oplus C_{i-1}), \\ C_0 &= \mathrm{IV}, \end{aligned} \tag{2.4}$$

  kde index i vyjadřuje pořadí bloku.

- Čítačový režim CTR (z angl. Counter) šifruje jednotlivé bloky dat pomocí nonce a inkrementálního čítače. Unikátní číslo složené z nonce a čítače se šifruje do bloku, který následně XORuje s daty.

  $$\begin{aligned} C_i &= M_i \oplus e_k(\mathrm{CTR}_i), \\ \mathrm{CTR}_i &= \mathrm{CTR}_{i-1} + 1, \\ \mathrm{CTR}_0 &= \mathrm{nonce}. \end{aligned} \tag{2.5}$$

  Tento mód umožňuje paralelní zpracování a šifrování dat s proměnlivou délkou.

2.2 Symetrické šifry a jejich zranitelnost

Kryptografický systém označujeme za symetrický pokud uživatelé Alice a Bob používají stejný klíč k šifrování i dešifrování zpráv, tedy Alice a Bob mají stejnou, nebo-li symetrickou znalost a platí, že $d_k(e_k(M)) = M$.

Odolnost symetrických šifer proti útoku spočívá ve velikosti klíče, kterým se zprávy šifrují. V úvodní kapitole jsem si na příkladech jednoduché posunovací a polyalfabetické šifry ukázali jak velikost klíče ovlivňuje velikost klíčového prostoru[6)] a jelikož soudobé

[6)] U šifrovacích protokolů se velikost klíčového prostoru uvádí v bitové notaci 2^n, kde n značí počet bitu v řetězci klíče.

symetrické šifrování je vcelku rezistentní vůči krypto analytickým metodám zejména založeným na statistice, nezbývá útočníkům než se popasovat s velikostí klíčového prostoru pokud chtějí prolomit kryptografickou ochranu.

Velikost klíče neovlivňuje pouze bezpečnost kryptosystému, ale i jeho efektivitu. Větší klíče spotřebovávají více systémových zdrojů a času k šifrování a dešifrování zpráv.

Vyvstává tedy otázka: jaký velký klíčový prostor zvolit, resp. kolik bitů by mělo tvořit řetězec klíče, tak aby s velkou mírou pravděpodobnosti nedošlo k prolomení šifry pomocí útoku hrubou silou v rozumném čase.

Hoffstein [3] předpokládá, že pokud útočník zná šifrovací mechanismus d_k a snadno rozezná platný otevřený text od neplatného otevřeného textu (zná i kódování), pak s přihlédnutím k hardwarovým možnostem je pro symetrické šifry minimální délka bezpečného klíče 80 bitů.

Ačkoliv délka klíče může zvyšovat výkonové nároky, zvýšení bezpečnosti je často důležitější než malé snížení výkonu. Je důležité zvolit délku klíče, která poskytuje dostatečnou úroveň zabezpečení vzhledem k očekávané životnosti dat, jejich citlivosti a požadavkům na výkon. V praxi se často používají klíče o délce 128 bitů nebo 256 bitů, které poskytují dobrý kompromis mezi bezpečností a výkonem.

2.2.1 Příklad symetrické šifry: AES

Na příkladu šifry AES (z angl. Advanced Encryption Standard) si ukážeme některé postupy v symetrickém šifrování v počítačové kryptografii. Vysvětlíme si například co jsou S-boxy, nebo jak lze v rámci algoritmu vytvořit ze základního klíče další tzv. rundovní klíče.

AES patří k nejpoužívanějším a nejdůvěryhodnějším kryptosystémům[7)]. Jedná se o blokovou iterativní šifru s délkou bloku 128 bitů a s volitelnou délkou klíče o velikosti 128, 192 a 256 bitů. AES může pracovat v různých operačních módech. V praxi se často využívá vylepšený CTR mód tzv. Galoisovský čítačový režim, který je standardizován např. pro 802.11ad, IPsec, nebo Secure Shell. Díky své bezpečnosti a efektivitě je AES široce používán v mnoha aplikacích, jako je ochrana dat, bezdrátová komunikace, finanční transakce a další.

[7)] Původní název AES byl Rijndael, což byla přesmyčka složená ze jmen autorů Joana Daemena a Vincenta Rijmena. Tvůrci šifru v roce 1997 přihlásili do soutěže, kterou vypsal americký úřad pro standardizaci (NIST), který tuto vítěznou šifru v roce 2002 prohlásil za standard pro šifrování tajných federálních dokumentů.

Protokol AES [18] se skládá z řady transformací (jak ze substitucí, tak i permutací), které se provádějí v několika kolech. Počet kol závisí na délce klíče:

- 10 kol pro 128bitový klíč
- 12 kol pro 192bitový klíč
- 14 kol pro 256bitový klíč

Před zahájením samotného šifrování dat dojde k expanzi šifrovacího klíče, ze kterého se vygenerují podklíče (z angl. round keys). Tyto rundovní podklíče mají stejnou délku jako zdrojový klíč a jsou pak použity v každém kole šifrovacího procesu. Jeden podklíč složen z N 32 bitových úseků (words), tedy pro AES s 128 bitovým klíčem je $N = 4$.

Funkce expanze klíče w funguje tak, že nejprve se rozdělí zdrojový (původní) klíč na prvních N úseků a následně další úseky jsou generovány ve smyčce, kde každý nový úsek je vypočten jako XOR dvou předchozích úseků. Existuje však speciální případ pro úseky, jejichž index je násobek N. V těchto případech je jeden z úseků nejdříve posunut o jedno místo doleva (tzv. rotWord operace) a poté transformován pomocí S-boxu (tzv. subWord operace) než je použit v XOR operaci. K tomuto úseku je také přidána konstanta závislá na kole (tzv. Rcon operace). Tento proces pokračuje, dokud není vygenerován dostatečný počet úseků pro všechna kola AES.

SubWord operace je nelineární transformace, která nahrazuje každý bajt[8)] úseku odpovídajícím bajtem z S-boxu.

Matematická reprezentace operací v expanzi klíče v AES může být zapsána následovně:

$$w[i] = \begin{cases} w[i-N] \oplus \text{subWord}(\text{rotWord}(w[i-1])) \oplus \text{Rcon}[i/N] & \text{pro } i \mod N \equiv 0 \\ w[i-N] \oplus w[i-1] & \text{pro ostatní } i, \end{cases} \tag{2.6}$$

kde index $[i]$ označuje pozici úseku v řetězci.

Konkrétní implementace funkce expanze klíče může záviset na specifických požadavcích a omezeních dané aplikace, jako je dostupná paměť a výpočetní výkon.

S-box (substituční box) je tabulka obsahující 256 jedinečných hodnot (od 0x00 do 0xFF), které jsou použity pro substituční kroky subWord v expanzní funkci a subBytes operaci v hlavní části algoritmu. S-box se využívá nejen v AES, ale i v mnoha dalších symetrických šifrovacích algoritmech.

[8)] Bajt (z angl. byte) je složen z 8 bitů a může nabývat 256 různých hodnot.

Standard AES popisuje S-Box jako inverzní funkci[9] v konečném tělese $GF(2^8)$, následovanou afinní transformací[10]. Tato konstrukce je navržena tak, aby poskytla dodatečný rozptyl dat, a tím razantně zvýšila odolnost proti lineární a diferenciální kryptoanalýze[11].

Matematická definice S-boxu v AES je následující:

Afinní transformace $GF(2)$: $b'' = b' \oplus c \oplus \mathrm{rot}(b', 1) \oplus \mathrm{rot}(b', 2) \oplus \mathrm{rot}(b', 3) \oplus \mathrm{rot}(b', 4)$, kde konstanta (hexadecimální číslo) c =0x63, $\mathrm{rot}(x, n)$ značí rotaci bajtu x o n pozic doleva, a kde ke každému bajtu $b \neq 0$ náleží jeho inverze b' v $GF(2^8)$. Tato funkce je aplikována na všechny hodnoty bajtu (0x00 až 0xFF), aby se vytvořila S-box.

Hlavní část algoritmu AES probíhá následovně:

Otevřený text je rozdělený do bloků o velikosti 128 bitů. Každý blok tvoří sloupcově uspořádanou stavovou matici S typu 4×4, kde každý prvek v matici představuje 1 bajt. Algoritmus vykoná postupně řadu operací na každé matici ve 3 fázích:

1. Inicializace: Operace AddRoundKey XORuje stavovou matici s prvním podklíčem.

   Předpokládejme stavový blok $S = [S_{ij}]$ a podklíč $K = [K_{ij}]$, kde i označuje řádek a j označuje sloupec. Potom operace AddRoundKey je definována jako $\forall i, j : S_{ij} := S_{ij} \oplus K_{ij}$[12]

2. Iterace (9, 11, nebo 13 kol) opakuje operace:

   (a) SubBytes operace podobně jako subWord nahrazuje prvky v matici S odpovídajícími prvky v S-Boxu B a je definována jako $\forall i, j : S_{ij} := B(S_{ij})$.

   (b) ShiftRows je jednoduchá permutační operace. Každý řádek stavového bloku se posune o určitý počet pozic doleva. První řádek zůstává nezměněn, druhý řádek se posune o jednu pozici, třetí řádek o dvě pozice a čtvrtý řádek o tři pozice. ShiftRows je definována jako $\forall i, j : S_{ij} := S_{i,(j+i) \mod 4}$.

[9] Inverzní funkce pro daný prvek a tělesa najde takový prvek b, že jejich násobení v tělese dá jednotkový prvek. Například, pokud máme prvky a a b takové, že $a \times b = 1$, pak b je multiplikativní inverzí prvku a, a naopak.

[10] Afinní transformace f prvku x v obecné formě může být zapsána jako $f(x) = Ax + b$, kde A je matice, která reprezentuje lineární transformaci (škálování a rotaci) a b je vektor, který reprezentuje translaci (posun).

[11] Lineární kryptoanalýza se snaží vytvořit lineární aproximaci šifrovací funkce. Tato aproximace se poté snaží využít statistické vztahy mezi vstupem a výstupem šifrovací funkce a klíčem použitým pro šifrování. Pokud je možné vytvořit dostatečně přesnou lineární aproximaci, pak by mohl být odhalen šifrovací klíč, nebo jeho části.

Diferenciální kryptoanalýza se naopak zaměřuje na změny v šifrovací funkci způsobené malými změnami vstupu. Pokud se malá změna vstupu projeví jako velká změna ve výstupu (nebo naopak), může to naznačovat určité vlastnosti šifrovací funkce, které mohou být využity k odhalení šifrovacího klíče.

[12] Symbol := značí aktualizaci hodnoty v proměnné v sekvenci výpočtů.

(c) MixColumns provádí lineární transformaci na každém sloupci stavového bloku. Tato operace je implementována jako násobení polynomů v konečném tělese $GF(2^8)$.

Každý sloupec c stavového bloku je považován za polynom[13)] nad $GF(2^8)$, který je vynásoben jiným pevně daným polynomem[14)]

$$3x^3 + x^2 + x + 2 \mod (x^4 + 1). \tag{2.7}$$

Tato operace se provádí v $GF(2^8)$, kde sčítání je implementováno jako operace XOR a násobení je prováděno podle pravidel pro násobení v konečných tělesech.

Operace MixColumns je definována jako: $\forall j : c'_j = a(x) \cdot c_j \mod m(x)$, kde c_j je j-tý sloupec stavového bloku a c'_j je j-tý sloupec stavového bloku po operaci MixColumns.

(d) Výsledek je pomocí operace AddRoundKey opět XORován s následujícím podklíčem.

3. Finalizace v závěru šifrovacího procesu provede na stavovém bloku tyto operace:

(a) SubBytes

(b) ShiftRows

(c) AddRoundKey

V poslední fázi se MixColumns operace již neprovádí.

Algoritmus provádí řadu transformací. SubBytes má stejnou úlohu jako subWord, tedy znesnadnit prolomení šifry pomocí lineární a diferenciální kryptoanalýzy. Operace ShiftRows a MixColumns poskytují difuzi mezi jednotlivými řádky, potažmo sloupci.

Difuze je jedním ze dvou klíčových aspektů bezpečného šifrovacího systému, který navrhl Claude Shannon [11], otec moderní kryptografie. Difuze znamená, že pokud změníte jediný bit vstupu (otevřeného textu), měl by se změnit každý bit výstupu (šifrovaného textu) s pravděpodobností přibližně 50%. Jinými slovy, malé změny na vstupu by měly vést k rozsáhlým a nepředvídatelným změnám na výstupu.

[13)] V kontextu operace MixColumns v algoritmu AES se každý sloupec čtyřbajtového stavového bloku považuje za polynom s koeficienty v konečném tělese $GF(2^8)$. Například sloupec s byty $(b0, b1, b2, b3)$, tento sloupec lze reprezentovat jako polynom $b0 \cdot x^3 + b1 \cdot x^2 + b2 \cdot x + b3$.

[14)] Polynom $x^4 + 1$ je redukční polynom použitý pro operaci modulo v rámci násobení polynomů. To znamená, že pokud výsledek násobení polynomů překročí stupeň 3 (to by odpovídalo pěti bytům, což by nebylo kompatibilní se čtyřbytovým sloupcem), je výsledek dělen modulárně tímto polynomem, aby se vrátil zpět na stupeň 3 nebo nižší.

V kontextu operace MixColumns v algoritmu AES difuze mezi jednotlivými sloupci znamená, že každý bajt výstupního sloupce je funkcí všech čtyř bajtů vstupního sloupce. To znamená, že změna jakéhokoli bajtu ve vstupním sloupci způsobí změny všech bajtů ve výstupním sloupci.

Tímto způsobem se šifrovací vliv jednoho vstupního bajtu rozprostře na více bajtů výstupu, což ztěžuje útočníkovi analýzu šifrovaných dat.

Dalším aspektem, který Shannon považuje za zásadní pro bezpečnost šifer je zmatení (z angl. confusion). Cílem zmatení je, aby každý bit šifrového textu závisel na několika bitech šifrovacího klíče a aby tato závislost byla co nejkomplikovanější. Pokud je tento vztah jednoduchý, útočník může zkusit odvodit klíč z šifrovaného textu, což je samozřejmě nežádoucí. V AES mají zmatení na starosti operace SubBytes a AddRoundKey.

2.2.2 Analýza zranitelnosti symetrických šifer proti kvantovým útokům: Groverův algoritmus

Poznámka: Princip Groverova algoritmu je detailněji popsán v praktické části této práce. Co je stav superpozice a co je qubit vysvětluje kapitola 5. Kvantové výpočty.

Na popisu protokolu AES jsme si představili některé matematické operace, jejichž cílem je skrýt veškeré informace o vnitřním stavu šifrovacího procesu, které by se daly využít k prolomení šifry.

Za prolomení šifry považujeme situaci, kdy jsme schopni odhalit šifrovací klíč dříve, než pomocí útoku hrubou silou.

S vyloučením zastaralých kryptoschémat (např. DES, nebo FEAL[15)]), lze symetrické šifry považovat za úspěšné v minimalizaci účinnosti kryptoanalytických metod založených na statistice, lineární a diferenciální analýze.

Z těchto důvodů se uplatňují alternativní směry útoků. Příkladem mohou být tzv. side-channel útoky[16)], nebo útoky pomocí kvantových počítačů, kterým se věnuje tato kapitola.

Symetrické šifry, jako je AES, jsou zranitelné vůči útokům využívajícím kvantové výpočty, neboť umožňují paralelní prohledávání celého klíčového prostoru. K tomuto

[15)] Ačkoli DES jako takový nebyl plně prolomen pomocí lineární nebo diferenciální kryptoanalýzy, tyto techniky ukázaly, že DES má menší bezpečnostní rezervu, než se původně myslelo. Diferenciální kryptoanalýza může prolomit šifru DES se zredukovaným počtem kol, což ukazuje na potenciální slabosti v plné šifře.

[16)] Side-channel attacks představují kategorii kryptoanalytických technik, které se zaměřují na informace získané z fyzické implementace kryptografického systému, nikoli z přímé analýzy samotných kryptografických operací. Tyto útoky mohou využívat různé typy informací o výpočetním systému, jako je spotřeba energie, čas zpracování, elektromagnetické záření, zvuky a další fyzikální jevy. Například, doba potřebná pro provedení šifrovací operace může někdy odhalit informace o použitém klíči.

účelu lze využít Groverův algoritmus [19] pro prohledávání neuspořádaných databází.

Groverův algoritmus je jedním z důležitých kvantových algoritmů, který může být použit k významnému snížení složitosti hledání klíče v symetrických šifrách. Tento algoritmus vytvoří superpozici všech možných stavů (klíčů) a poté postupně zvyšuje pravděpodobnost správného stavu (správného klíče). Kvantový počítač, který by byl schopný efektivně implementovat Groverův algoritmus, by mohl provést útok hrubou silou na symetrickou šifru v čase proporcionálním k druhé odmocnině velikosti klíčového prostoru, nikoli lineárním čase jako u klasických počítačů. Dodejme, že Groverův algoritmus je deterministický algoritmus, který vždy najde hledaný prvek (pokud existuje) v daném čase a není tedy možné jej využít pro opakované postupné zúžení množiny možných klíčů.

V kontextu AES, to znamená, že 128-bitový klíč, který by byl považován za bezpečný proti útoku hrubou silou na klasickém počítači, by mohl být prolomen na kvantovém počítači v čase proporcionálním k 2^{64}. Tato hodnota je již pod Hoffsteinem definovanou hodnotou bezpečného klíče (80 bitů). Podobně, 256-bitový klíč by mohl být prolomen v čase proporcionálním k 2^{128}, což je stále mimo dosah současných kvantových i klasických počítačů.

Groverův algoritmus na kvantovém počítači vyžaduje počet qubitů[17)] rovný velikosti klíče. Pro 128-bitový klíč je tedy nutné mít minimálně 128 qubitů. Nicméně kvůli kvantové chybové korekci a technickým nárokům kvantového počítání je v praxi skutečný počet qubitů mnohem vyšší. Odhady naznačují, že kvantová chybová korekce může zvýšit počet potřebných qubitů až tisíckrát.

Je důležité poznamenat, že ačkoli kvantové počítače představují hypotetickou hrozbu pro současné symetrické šifry, stále jsou to jen teoretické úvahy. Kvantové počítače schopné efektivně implementovat Groverův algoritmus na velkých šifrovacích klíčích zatím neexistují. V současné době (k roku 2023) kvantové počítače dosahují řádově několik stovek qubitů a stále mají významné technické výzvy, které brání praktické realizaci takového útoku.

Přesto je důležité se na možnost kvantových útoků připravit. Jedním z možných přístupů je zvýšení délky klíče symetrických šifer. Například přechod z AES-128 na AES-256 by měl poskytnout dostatečnou bezpečnostní rezervu i v případě rozvoje kvantových počítačů.

2.3 Asymetrické šifry a jejich zranitelnost

Asymetrické šifry, známé také jako šifry s veřejným klíčem (PKE z angl. Public Key Encryption), jsou založeny na matematických problémech, které jsou snadno řešitelné

[17)] Qubit (kvantový bit) je základní jednotka kvantové informace.

v jednom směru, ale velmi obtížné v opačném směru. Tyto problémy jsou známé jako jednosměrné funkce s tajemstvím (z angl. trapdoor function).

Příkladem takového matematického problému, na němž jsou založeny asymetrické šifry, je faktorizace velkých prvočíselných součinů (jako v případě RSA) nebo problém diskrétního logaritmu (jako v případě Diffie-Hellmanovo nebo ElGamalovo šifrování).

Tyto složité matematické operace jsou náročnější na hardwarové zdroje a jejich zpracování trvá déle, než je tomu v případě symetrických šifer.

V praxi se často používá kombinace symetrického a asymetrického šifrování, kde asymetrické šifrování se používá pro bezpečný přenos symetrického klíče a poté se pro samotné šifrování dat používá rychlejší symetrické šifrování. Tento přístup (KEM z angl. Key encapsulation mechanism) kombinuje praktičnost asymetrického šifrování s efektivitou symetrického šifrování.

Takto funguje asymetrické šifrování:

1. **Generování klíčů** : Každý účastník generuje pár klíčů - veřejný a soukromý. Veřejný klíč je určen k šifrování zpráv nebo k ověření podpisu, zatímco soukromý klíč je použit k dešifrování zpráv, nebo k vytvoření podpisu. Důležité je, že soukromý klíč nemůže být vypočítán z veřejného klíče v rozumném čase pomocí dostupných výpočetních zdrojů.

   Proces generování klíčů se liší v závislosti na konkrétním algoritmu, ale obecně zahrnuje následující kroky:

   (a) Vygenerování soukromého klíče: Soukromý klíč je generován náhodně. Měl by být dostatečně dlouhý, aby nebylo možné ho uhodnout hrubou silou a měl by být generován pomocí CSPRNG.

   (b) Výpočet veřejného klíče: Veřejný klíč je vypočítán ze soukromého klíče pomocí specifické funkce, která je závislá na konkrétním algoritmu. Tato funkce je jednosměrná, což znamená, že je snadné vypočítat veřejný klíč ze soukromého klíče, ale téměř nemožné vypočítat soukromý klíč z veřejného klíče.

2. **Šifrování** : Odesílatel šifruje zprávu pomocí veřejného klíče příjemce.

3. **Dešifrování** : Příjemce dešifruje zprávu pomocí svého soukromého klíče .To je možné díky matematické vlastnosti páru klíčů - to, co bylo zašifrováno veřejným klíčem, může být dešifrováno pouze odpovídajícím soukromým klíčem.

Asymetrické šifrování čelí dvěma výzvám: zajištění odolnosti veřejného klíče proti odvození soukromého klíče a problému s distribucí klíčů.

Významným faktorem, který ovlivňuje odolnost veřejného klíče proti útokům, je jeho velikost, neboli počet bitů, které klíč obsahuje. Obecně platí, že čím je klíč delší, tím je těžší ho prolomit pomocí hrubé síly nebo jiných útoků. Na druhé straně zvýšení velikosti klíče také zvyšuje náročnost šifrování a dešifrování pro legitimní uživatele.

Problém s distribucí veřejných klíčů spočívá v jejich autenticitě. Jinými slovy, když Alice chce poslat Bobovi zašifrovanou zprávu, musí si být absolutně jistá, že veřejný klíč, který používá, skutečně patří Bobovi a ne nějakému útočníkovi.

Existuje několik metod, jak řešit tento problém:

- Infrastruktura pro správu a distribuci veřejných klíčů (PKI z angl. Public Key Infrastructure): Tato infrastruktura zahrnuje třetí stranu, známou jako certifikační autorita (CA). CA vydává digitální certifikáty, které spojují veřejný klíč s jeho vlastníkem. Certifikát je elektronicky podepsán CA, takže jakákoli strana může ověřit jeho pravost. Například v kontextu webových serverů, tyto certifikáty jsou instalovány na server. Když klient (webový prohlížeč) naváže spojení se serverem, server předá svůj certifikát klientovi, který ho může u CA ověřit. PKI je pilířem bezpečné komunikace na internetu a je základem pro technologie jako je HTTPS, SSL/TLS, a mnoho dalších systémů zabezpečení.

- Internet Key Exchange Protocols: Protokoly jako Diffie-Hellman umožňují dvěma stranám vytvořit společný tajný klíč přes nezabezpečený kanál. Tento klíč může být pak použit pro symetrické šifrování. Avšak, bez dodatečných opatření je Diffie-Hellman náchylný k útokům Man-in-the-middle [18)].

- Web of Trust: V tomto modelu, uživatelé podepisují veřejné klíče ostatních uživatelů, kterým důvěřují. Tímto způsobem se vytváří "síť důvěry", která umožňuje uživatelům ověřovat veřejné klíče ostatních uživatelů.

[18)] Man-in-the-Middle attack, je typ útoku, kdy útočník přeruší komunikaci mezi dvěma stranami a sám se stane prostředníkem v jejich komunikaci. Tímto způsobem může útočník sledovat, manipulovat a přeposílat zprávy mezi oběma stranami, aniž by si toho strany byly vědomy. Útok může probíhat v kontextu asymetrického šifrování následovně:

1. Alice chce poslat Bobovi zašifrovanou zprávu, takže požádá Boba o jeho veřejný klíč.
2. Eva, která sleduje komunikaci mezi Alicí a Bobem, zachytí Alicin požadavek.
3. Eva pošle Alici svůj vlastní veřejný klíč místo Bobova.
4. Alice, která si není vědoma útoku, zašifruje zprávu obdrženým klíčem a pošle ji Bobovi.
5. Eva zachytí Alicinu zašifrovanou zprávu, dešifruje ji svým soukromým klíčem, zaznamená její obsah a poté zprávu znovu zašifruje Bobovým veřejným klíčem a pošle ji Bobovi.
6. Bob dostane zprávu, o které se domnívá, že ji poslala Alice.

Tento druh útoku je důvodem, proč je kladen takový důraz na autenticitu v asymetrickém šifrování.

2.3.1 Příklad asymetrické šifry: RSA

Šifrovací algoritmus RSA [20] byl poprvé popsán v roce 1977 třemi výzkumníky na MIT, Ronem Rivestem, Adim Shamirem a Leonardem Adlemanem, po nichž je také šifra pojmenována.

RSA je založen na matematické složitosti rozkladu velkého čísla N na součin prvočísel p a q. Velké číslo N (modul) tvoří společně s malým číslem e (veřejný exponent) veřejný klíč. Zatímco soukromý klíč je tvořen stejným modulem N a soukromým exponentem d, který je vypočítán tak, aby splňoval určité matematické vlastnosti ve vztahu k veřejnému exponentu.

Základní RSA algoritmus pro generování klíčů je následující:

1. Vyberou se dvě prvočísla p a q. V praxi tyto čísla mají velikost několika stovek bitů a jsou generovány pomocí CSPRNG. Přestože jsou p a q vybrány náhodně, musí být vybrány tak, aby splňovaly některá specifická kritéria. Například p a q by měly mít přibližně stejnou délku, ale zároveň by měly být dostatečně odlišná[19)].

2. Vypočítá se modul $N = p \cdot q$ a hodnota Eulerovy funkce $\varphi(N) = (p-1)(q-1)$.

3. Vybere se celé číslo e takové, že $1 < e < \varphi(N)$ a $\text{NSD}(e, \varphi(N)) = 1$, kde funkce NSD označuje největšího společného dělitele[20)]. Nejčastěji se používají hodnoty $e = 3$ nebo $e = 65537$ bez ohledu na velikost modulu N.

4. Určí se d, pro které platí $d \equiv e^{-1} \mod \varphi(N)$, nebo-li d je modulární multiplikativní inverzí e modulo $\varphi(N)$. K určení d se v RSA využívá Rozšířený Eukleidův algoritmus[21)].

5. Veřejný klíč je dvojice (N, e) a soukromý klíč je (N, d).

Před šifrováním se nejprve převede text otevřené zprávy na číslo m podle předem určeného kódování, které je implementováno v konkrétním protokolu RSA. Při šifrování zprávy m se vypočítá šifrovaná zpráva c jako $c = m^e \mod N$. Při dešifrování se pak vypočítá původní zpráva jako $m = c^d \mod N$.

[19)] Tyto pravidla zvyšují zabezpečení RSA proti různým typům útoků.Pokud by jedno z prvočísel bylo mnohem menší než druhé, mohlo by to zjednodušit útok na RSA pomocí algoritmů pro faktorizaci. Algoritmy pro faktorizaci, jako je například algoritmus kvadratického síta, mají vyšší efektivitu nalezení faktorů, pokud je jedno z prvočísel mnohem menší než druhé.

[20)] Funkce $\text{NSD}(a, b)$ označuje největšího společného dělitele dvou celých čísel a a b. Největší společný dělitel je největší celé číslo, které dělí obě čísla beze zbytku.

[21)] Rozšířený Eukleidův algoritmus se používá pro nalezení největšího společného dělitele dvou čísel a a b, a současně pro nalezení koeficientů x a y v tzv. Bézoutově rovnosti ve výrazu: $ax + by = \text{NSD}(a, b)$, kde $a, b \in \mathbb{N}$ a $x, y \in \mathbb{Z}$. [12]

Vztah veřejného exponentu e a soukromého exponentu d vychází z teorému Malé Fermatovy věty[22], díky čemuž platí, že $(m^e)^d = m^{e \cdot d} = m^1 = m \mod N$, neboť $e \cdot d \equiv 1 \mod \varphi(N)$.

Pro správné fungování RSA musí být zpráva m menší než modul N. Pokud je zpráva větší než N, musí být rozdělena na menší bloky.

Výsledek šifrování RSA je konzistentní a předvídatelný. Stejná zpráva zašifrovaná stejným klíčem vždy vede ke stejnému šifrovanému textu. Tato deterministická povaha RSA může být problematická z hlediska bezpečnosti. Pokud útočník odhalí, že určitá zpráva vede ke konkrétnímu šifrovanému textu, může tuto informaci využít k odhalení dalších informací o klíči nebo šifrovaných datech. Tato technika, kde útočník má schopnost získat zašifrované zprávy odpovídající otevřeným zprávám, které si sám zvolil je známá jako chosen plaintext attack (CPA).

Odolnost vůči CPA je často považována za minimální požadavek na bezpečnost šifrovacího systému.

V kontextu asymetrického šifrování se využívá tzv. padding[23], který zajišťuje, že i když jsou zašifrovány dvě identické zprávy, výsledné šifrované zprávy budou odlišné. Padding je proces obsažený ve fázi kódování, kdy se k otevřené zprávě přidávají další data (často náhodný řetězec) před jejím šifrováním.

2.3.2 Analýza zranitelnosti asymetrických šifer proti kvantovým útokům: Shorův algoritmus

Poznámka: Princip Shorova algoritmu a problematika faktorizačního problému jsou popsány v praktické části této práce.

U RSA šifry, s rostoucí velikostí veřejného klíče (konkrétně části N) roste složitost faktorizace N exponenciálně. Pro velké N je tato složitost velmi vysoká a faktorizace takových čísel je prakticky nerealizovatelná na současných klasických počítačích, proto asymetrické šifrování pracuje s veřejným šifrovacím klíčem, který je násobně větší, než je tomu u symetrického šifrování. V případě RSA se považuje za bezpečný klíč o velikosti 2048 bitů.

Je důležité poznamenat, že zvětšení klíče sice zvyšuje bezpečnost, ale také zvyšuje náročnost na výpočetní zdroje potřebné pro šifrování a dešifrování. Jašek doslova označuje RSA algoritmus za neúnosně pomalý [2]. Proto je důležité najít správnou rovnováhu mezi bezpečností a efektivitou.

Shorův algoritmus [21] může efektivně řešit v polynomiálním čase dva matematické

[22] Malá Fermatova věta tvrdí, že pro každé prvočíslo p a pro každé celé číslo a, které není násobkem p, platí, že $a^{p-1} \equiv 1 \mod p$.

[23] Jeden z nejčastěji používaných padding schémat pro RSA je OAEP (z angl. Optimal Asymmetric Encryption Padding), který je součástí aktuálního (květen 2023) RSA standardu PKCS #1 v2.2. [20]

problémy, které jsou základem populárních kryptosystémů:

1. Faktorizace velkých čísel: Algoritmus rychle rozloží zadané číslo na součin prvočísel. To ovlivňuje kryptosystémy, které se spoléhají na matematické problémy, které lze redukovat na faktorizační problém. Mezi tyto kryptosystémy patří zmíněný algoritmus RSA a dále méně známé Rabin, nebo Paillier.

2. Výpočet diskrétních logaritmů: Řešení problému diskrétního logaritmu, který spočívá v nalezení x ve výrazu $g^x = h \mod p$ pro daná g, h a p. Kryptosystémy, které se spoléhají na obtížnost tohoto problému jsou Diffie-Hellman, ElGamal a eliptické křivky.

V současné době nemáme kvantový počítač, který by mohl efektivně provést Shorův algoritmus na klíčích s běžně používanou velikostí, nicméně v praktické části je demonstrován Shorův faktorizační algoritmus, který je schopný provést rozklad N o velikosti jednotek bitů.

Pro faktorizaci 2048-bitového čísla by bylo potřeba v ideálním případě 4096 qubitů pro uchování čísla a provedení Shorova algoritmu, nicméně v praxi kvantové počítače trpí různými druhy chyb, které mohou narušit výpočet. Tak jako v případě Groverova algoritmu je potřeba implementovat techniky kvantové korekce chyb.

Existují odhady, které naznačují, kolik qubitů by bylo potřeba k provedení Shorova algoritmu pro faktorizaci 2048-bitového RSA klíče. Jedna z těchto studií, *Factoring integers with sublinear resources on a superconducting quantum processor* od Yana et al. z roku 2022 [22], odhaduje, že by bylo potřeba přibližně 20 milionů qubitů.

Dílčí závěr

V této kapitole jsme si představili základy moderní kryptografie a ukázali jsem si, jak mohou být jednoduché matematické operace transformovány na mocné nástroje pro zajištění bezpečnosti dat. Na příkladu symetrické šifry AES byl popsán koncept využití kombinace substituce a permutace, který posouvá bezpečnost šifer na úroveň, kdy používání klasických kryptoanalytických metod už nedává smysl. U asymetrického šifrování byl prezentován geniální způsob využití jednosměrných funkcí pro distribuci šifrovacích klíčů mezi mnoha účastníky. Na druhé straně jsme si také představili výzvu pro moderní šifrování v podobě kvantových počítačů a byť jsou kvantové algoritmy využitelné pro odkrytí chráněné informace prozatím zásadně limitovány přesností fyzických qubitů provádět složitější výpočty a těžko lze odhadnou kdy (nebo zda) se dostaneme do bodu skutečné kvantové nadvlády, tak je nezbytné abychom zůstali o krok napřed. Proto se odborníci na kryptografii již delší dobu po vzoru praemonitus praemunitus zabývají vývojem kvantově odolných kryptoschémat, na které se podíváme v další kapitole.

3 POST-KVANTOVÁ KRYPTOGRAFIE

V závěru předchozí kapitoly byly popsány dva kvantové algoritmy. V případě Groverova algoritmu, pomocí kterého lze redukovat šifrovací klíče symetrických šifer a hašovacích funkcí na polovinu jejich délky, se sice jeví využití jako užitečné, ale prakticky bezvýznamné pro algoritmy pracující s klíči o 256 bitech. Druhý algoritmus Petera Shora schopný nalézt prvočíselný rozklad v polynomiálním čase má naopak zásadní dopad na současnou úroveň kryptografické ochrany a to navzdory, že nás dělí 10 či více let od existence potřebného hardwaru.

Bezpečnostní komunita upozorňuje na skutečnost, že nejen kriminální živly, ale zejména některé státy po delší dobu schraňují odcizená zašifrovaná data s cílem je dešifrovat později s pomocí kvantových počítačů. Tato forma útoku se označuje prozaickým názvem Steel now, decrypt later attack.

Ohrožení kryptografické ochrany informací je v řadě zemí anticipováno. V lednu 2022 Bidenova administrativa nařídila všem vládním agenturám vypracovat reakční plány během následujících 6 měsíců a již v dubnu téhož roku byl představen návrh zákona o připravenosti na kybernetickou bezpečnost v oblasti kvantové výpočetní techniky[1)], který požaduje, aby americké federální agentury zahájily proces stanovení priorit pro migraci na kvantově odolné šifrování do jednoho roku od vyhlášení kvantově odolného kryptografického standardu. Také v Evropě Evropská komise v roce 2019 navrhla iniciativu The European Quantum Communications Infrastructure (EuroQCI), v níž jedním z cílů je vybudovat v Evropě kvantově bezpečný internetový systém využívající kvantovou distribuci klíčů (QKD z angl. Quantum Key Distribution).

3.1 Definice post-kvantových šifer a jejich matematických základů

Post-kvantová kryptografie (PQC z angl. Post-quantum Cryptography) jako obor ověřuje vlastnosti moderních kryptografických systémů, zda jsou odolné vůči útokům kvantových počítačů a navrhuje nové způsoby šifrování.

U symetrického šifrování stačí k zachování bezpečnosti využít větší šifrovací klíč, zatímco u asymetrického šifrování je třeba nalézt matematické problémy a s nimi spojené jednosměrné funkce, které mají v opačném směru obtížné řešení jak na klasických, tak i kvantových počítačích.

Uvažované kvantově odolné šifry se nejčastěji zakládají na jednom ze třech matematických problémech:

- **Problém mřížek** (Lattice-based cryptography): Tyto kryptosystémy jsou založeny na obtížnosti některých problémů spojených s mřížkami. Mřížka L v n-

[1)] Nyní zákon: H.R.7535 - Quantum Computing Cybersecurity Preparedness Act [23]

rozměrném prostoru je definována jako množina všech celočíselných lineárních kombinací dané sady vektorů, které tvoří bázi mřížky[2)].

Můžeme to zapsat matematicky jako: [3]

$$L = \left\{ \sum a_i v_i \mid a_i \in \mathbb{Z}, v_i \in \mathbb{R}^n \text{ a tvoří bázi mřížky} \right\}, \tag{3.1}$$

kde $\mathbb{Z}$ je množina všech celých čísel a $\mathbb{R}^n$ je n-rozměrný reálný prostor. Každý bod na mřížce L lze vyjádřit jako celočíselnou lineární kombinaci vektorů tvořících bázi mřížky.

Dva nejznámější mřížkové problémy, u kterých není znám žádný efektivní algoritmus pro jejich řešení, jsou:

1. Nalezení nejkratšího vektoru (SVP z angl. Shortest Vector Problem): V dané mřížce se hledá nejkratší nenulový vektor.
2. Nalezení nejbližšího vektoru (CVP z angl. Closest Vector Problem): Je dán bod t mimo mřížku a úkolem je najít nejbližší vektor v mřížce k danému bodu t.

V kontextu mřížkových kryptosystémů se tyto problémy využívají pro vytvoření bezpečných kryptografických schémat. Například, v mřížkovém šifrovacím systému by mohl být veřejný klíč reprezentován mřížkou a soukromý klíč by mohl být reprezentován nejkratším vektorem této mřížky. Pouze držitel soukromého klíče (nejkratšího vektoru) by potom mohl efektivně dešifrovat zprávy zašifrované pomocí veřejného klíče (mřížky).

- **Problém kódování** (Code-based cryptography): Jde o třídu kryptografických problémů, které jsou založeny na teorii kódování[3)]. Jeden z nejznámějších kódových problémů je problém dekódování, který spočívá v nalezení nejbližšího kódu ke danému slovu. V kontextu lineárního kódování, kde kód je množina vektorů v n-rozměrném prostoru, je tento problém ekvivalentní hledání nejbližšího vektoru (CVP) v mřížkové kryptografii.

[2)] Vektor je matematický objekt, který má jak velikost, tak směr. V lineární algebře se vektor definuje jako prvek vektorového prostoru, množiny, kterou lze sčítat a násobit skaláry (čísly).

Báze je v matematice sada vektorů v daném vektorovém prostoru, které jsou lineárně nezávislé, a které pokrývají celý prostor, tj. jakýkoli vektor v prostoru lze vyjádřit jako lineární kombinaci vektorů báze. Každý vektorový prostor má bázi a všechny báze daného vektorového prostoru mají stejný počet prvků, což se nazývá dimenze prostoru.

[3)] Teorie kódování je obor matematiky, který se zabývá návrhem a analýzou kódů, které se používají k opravě chyb v přenášených datech. Problémy založené na teorii kódování vznikají, když se pokusíme dekódovat zprávu bez znalosti použitého kódu, a jsou známy svou výpočetní náročností.

- **MQ problém** (Multivariate cryptography): MQ (z angl. Multivariate Quadratic) problém je úloha řešení systému kvadratických rovnic. Máme-li danou posloupnost kvadratických polynomů P a daný vektor hodnot v z konečného tělesa, MQ problém spočívá v nalezení takového vektoru u, že když tento vektor dosadíme do posloupnosti polynomů P, dostaneme vektor hodnot v, nebo-li $\mathcal{P}(u) = v$, kde $\mathcal{P}$ je kvadratické zobrazení definované P.

  Jinak řečeno, hledáme vektor vstupů, který, když je dosazen do systému kvadratických rovnic, vyprodukuje daný vektor výstupů a jelikož systém má více neznámých než rovnic existuje mnoho možných řešení. Bez znalosti nějaké specifické struktury systému rovnic je nalezení konkrétního řešení (například toho, které odpovídá soukromému klíči v asymetrickém kryptosystému) výpočetně obtížné.

Mezi méně časté mechanizmy, které mají potenciál ve využití v kvantově odolném šifrování patří například problém hledání isogenie (speciálního druhu homomorfismu) u supersingulárních eliptických křivek, nebo šifrovací protokoly založené na hašovacích funkcích a Zero-Knowledge důkazech.

3.2 Standardizace kvantově odolných šifer

Vliv amerických technologických firem a potažmo legislativní rámec, ve kterém společnosti operují má nesporný dopad na globální úrovni v oblasti komunikačních protokolů a jejich zabezpečení. To je jedním z důvodů, proč se zraky upírají na americký národní úřad pro standardizaci (NIST z angl. National Institute of Standards and Technology) a na proces schvalování standardů v PQC.

Dalším důvodem je renomé NIST a jeho důkladný výběr šifer. Již koncem roku 2016 vyhlásil NIST soutěž [24] na standardizaci PQC, jejímž cílem bylo najít nové vhodné šifrovací algoritmy s veřejným klíčem. V prvním kole bylo přihlášeno 64 kandidátských algoritmů s různými přístupy k PQC. Řada kandidátů byla posouzena jako nevyhovující, proti 14 kandidátům byly provedeny úspěšné útoky a některé kandidátské šifry byly sloučeny pro podobný přístup dohromady. Do druhého kola postoupilo celkem 26 algoritmů z nichž 17 řešilo PKE-KEM a 9 pokrývalo oblast elektronických podpisů.

V červenci 2022 NIST oznámil čtyři kandidátské šifry, které podstoupí proces standardizace (viz. tab. 3.1) a vhledem k jedinému vybranému algoritmu pro PKE-KEM bylo vyhlášeno další kolo do kterého byly vybrány algoritmy s odlišným způsobem šifrování (viz tab. 3.2).

Podle NIST jsou šifry CRYSTALS-Kyber a CRYSTALS-Dilithium zvoleny především na základě jejich robustní bezpečnosti a uspokojivého výkonu, což je činí vhodnými pro většinu aplikací. Falcon byl identifikován jako užitečný pro situace, kdy by

Tabulka 3.1 Standardizace PQC. Data čerpána z [24]

Typ algoritmu	PKE-KEM	Elektronické podpisy
Šifrování na mřížce	CRYSTALS-Kyber	CRYSTALS-Dilithium Falcon
Hash-based šifrování		SPHINCS+

Tabulka 3.2 Kandidátské algoritmy PQC. Data čerpána z [24]

Typ algoritmu	PKE-KEM
Code-based šifrování	BIKE Classic McEliece HQC
Supersingular elliptic curve isogeny	SIKE

elektronický podpis generovaný pomocí CRYSTALS-Dilithium byl příliš velký a jako alternativa k šifrám založeným na mřížce byl vybrán SPHINCS+.

Ve čtvrtém kole jsou hodnoceny algoritmy BIKE, HQC, SIKE a Classic McEliece. Algoritmus McEliece kvůli velkosti svého veřejného klíče zřejmě nepostoupí do procesu standardizace a v Castryckově studii [25] ze srpna 2022 byl popsán úspěšný útok na algoritmus SIKE. Je tedy pravděpodobné, že NIST vybere alespoň jednoho z dvojice BIKE a HQC jako dalšího finalistu.

Cílem NIST není zvolit pouze jeden algoritmus, ale sestavit skupinu tzv. rodin algoritmů, které fungují na různých základech. Tento krok je klíčový v případě, že by v budoucnosti byl objeven nový způsob útoku, ať už pro tradiční nebo kvantový počítač, schopný porazit jeden ze zvolených standardů šifrování. V takové situaci bychom mohli snadno přejít na použití šifrovacího algoritmu z jiné rodiny, který by pravděpodobně nebyl tímto novým útokem ohrožen. Realita je taková, že u žádných šifrovacích algoritmů nejsme schopni s jistotou prohlásit, že jsou absolutně neprolomitelné v rámci nějakého přijatelného časového období.

Implementace kvantově odolného šifrování bude dlouhodobý a poměrně náročný proces. NIST předpokládá, že plná implementace PQC bude trvat 5-15 let po ukončení standardizačního procesu.

Dílčí závěr

Matematické problémy, na kterých se buduje PQC musí zajistit dostatečnou odolnost před útoky pomocí kvantových počítačů, ale také před útoky pomocí klasických počítačů. Hledání vhodných šifrovacích algoritmů je dlouhý proces, který začal mnohem dříve než před šesti lety, kdy NIST uveřejnil svoji výzvu na nalezení PQC. Tento proces začal záhy po konání 35. sympózia o základech počítačových věd (FOCS), kde Peter Shor představil svůj kvantový algoritmus.

4 MATEMATICKÝ APARÁT KVANTOVÝCH VÝPOČTŮ

Matematický model kvantového počítání vychází z kvantové teorie pole a a kvantové mechaniky. Je známo, že klasická fyzika popisuje fyzikální objekty pomocí jejich polohy a hybnosti, zatímco fyzikální objekty - atomární a subatomární částice v případě kvantové mechaniky jsou popsány vlnovou funkcí.

V této kapitole se seznámíme s matematickým konceptem Hilbertova prostoru a Braket notace, které jsou nezbytné pro pochopení a popis fungování kvantových systémů.

Kapitola rozšiřuje znalosti zejména z oboru lineární algebry. Čtenář by měl mít již povědomost, co jsou vektory, jak provádět operace s maticemi a jak pracovat s komplexními čísly.

4.1 Hilbertův prostor

Hilbertův prostor poskytuje pevný matematický základ pro analýzu a manipulaci kvantových systémů. Díky tomuto rámci je možné vytvářet a studovat kvantové algoritmy, které využívají unikátních vlastností kvantových jevů, jako je superpozice a provázanost.

Reprezentace kvantových stavů[1)] pomocí vektorů v unitárním Hilbertově prostoru usnadňuje jejich matematický popis a jejich abstrakci [27].

Unitární prostor V (také nazývaný jako prostor se skalárním součinem) je vektorový prostor nad komplexními čísly, který je vybaven skalárním (vnitřním) součinem.

Skalární součin je zobrazení, které každé dvojici vektorů přiřadí komplexní číslo, obvykle se značí $\langle \cdot | \cdot \rangle$ a splňuje následující axiomy [28]:

- Lineární v prvním argumentu: $\langle au + bv | w \rangle = a\langle u|w\rangle + b\langle v|w\rangle$ pro všechny vektory $u, v, w \in V$ a komplexní čísla a, b.
- Konjugovaně symetrický: $\langle u|v\rangle = \overline{\langle v|u\rangle}$ pro všechny vektory $u, v \in V$, kde $\overline{z}$ značí komplexně sdružené číslo z.
- Pozitivně definitní: $\langle u|u\rangle \geq 0$ a $\langle u|u\rangle = 0$ právě tehdy, když $u = 0$ pro všechny vektory $u \in V$.

Skalární součin v unitárním prostoru indukuje metriku (vzdálenostní funkci $\mathbf{d}$) a umožňuje měřit vzdálenosti a úhly mezi vektory u a v. Metrika je založena na normě vektoru [27].

Norma (délka) vektoru $u \in V$ je:

[1)] Kvantový stav můžeme považovat za matematický objekt, který popisuje fyzikální systém na kvantové úrovni atomů nebo subatomárních částic [26].

$$\|u\| = \sqrt{\langle u|u\rangle}. \tag{4.1}$$

Vzdálenost (Eukleidovská metrika) dvou vektorů $u, v \in V$ je:

$$\mathbf{d}(u, v) = \|u - v\|. \tag{4.2}$$

V unitárním prostoru s daným skalárním součinem můžeme definovat úhel θ mezi dvěma vektory u a v pomocí následujícího matematického výrazu:

$$\cos(\theta) = \frac{\langle u|v\rangle}{\|u\|\|v\|} \tag{4.3}$$

Metrika je důležitá pro měření kvantových stavů, neboť převádí abstraktní vnitřní součin na měřitelnou veličinu pro výpočet pravděpodobností a amplitud.

Hilbertův prostor $\mathcal{H}$ je zvláštním případem unitárního prostoru, který splňuje následující podmínky [27]:

- Úplnost: Každá Cauchyova posloupnost $\{u_i\}_{i=1}^{\infty} \in \mathcal{H}$ konverguje k nějakému vektoru $u \in \mathcal{H}$, což znamená, že pro
$$\lim_{i,j\to\infty} \|u_i - u_j\| = 0 \tag{4.4}$$
existuje takový vektor $u \in \mathcal{H}$, že pro
$$\lim_{i\to\infty} \|u_i - u\| = 0 \tag{4.5}$$

- Separabilita: Existuje spočetná množina vektorů $\{e_i\}$ v $\mathcal{H}$, která pokrývá celý prostor, tedy každý vektor $u \in \mathcal{H}$ lze zapsat jako lineární kombinaci prvků báze s koeficienty z komplexního číselného pole, tedy $u = \sum_{i=1}^{\infty} c_i e_i$, kde $c_i \in \mathbb{C}$.

V kvantovém počítání se často pracuje s konečně dimenzionálními Hilbertovými prostory $\mathcal{H}^N$, kde N značí počet dimenzí.

Pro systémy s konečně dimenzionálním Hilbertovým prostorem můžeme zvolit konečnou ortogonální bázi a reprezentovat kvantové stavy a operátory pomocí konečných vektorů a matic. Zároveň vlastnost úplnosti prostoru zajišťuje, že limity vektorových posloupností jsou dobře definované a díky tomu lze v rámci prostoru provádět diferenciální počet.

4.2 Bra-ket notace

Bra-ket notace, známá také jako Diracova notace, je elegantní způsob zápisu vektorů a vnitřních součinů v kvantovém počítání. Tato notace zavádí zkrácený způsob zápisu vektorů a operací s nimi, a tím usnadňuje manipulaci s kvantovými stavy a operátory. Tento styl zápisu zavedl matematik Paul Dirac [29].

Bra-ket notace se skládá ze dvou základních symbolů: $|\cdot\rangle$ a $\langle\cdot|$. Symbol $|\cdot\rangle$ označuje tzv. *ket* vektor (sloupcový vektor) v Hilbertově prostoru. Symbol $\langle\cdot|$ označuje duální vektor (řádkový vektor), tzv. *bra* vektor, který je komplexně sdruženým transponovaným vektorem původního *ket* vektoru.

Obecně platí, že pro libovolný *ket* vektor $|\psi\rangle = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$, kde $a_i \in \mathbb{C}$, je jeho příslušný *bra* vektor $\langle\psi| = \begin{pmatrix} a_1^* & a_2^* & \cdots & a_n^* \end{pmatrix}$, kde a_i^* je komplexní sdružené číslo a_i.

Skalární součin dvou vektorů $|\psi\rangle$ a $|\phi\rangle$ je zapsán v bra-ket notaci jako $\langle\psi|\phi\rangle$.

Matematicky, $\langle\psi|\phi\rangle = \langle\psi| \cdot |\phi\rangle = \sum_{i=1}^{n} a_i^* b_i$, kde $\psi = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$ a $\phi = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$.

Tensorový součin vektorů $|\psi\rangle$ a $|\phi\rangle$ definujme v bra-ket notaci jako $|\psi\phi\rangle$.

Matematicky, $|\psi\rangle \otimes |\phi\rangle = \begin{pmatrix} a_1 |\phi\rangle \\ a_2 |\phi\rangle \\ \vdots \\ a_n |\phi\rangle \end{pmatrix} = \begin{pmatrix} a_1 b_1 \\ a_1 b_2 \\ \vdots \\ a_1 b_m \\ a_2 b_1 \\ a_2 b_2 \\ \vdots \\ a_2 b_m \\ \vdots \\ a_n b_1 \\ a_n b_2 \\ \vdots \\ a_n b_m \end{pmatrix}$, kde $\psi = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$ a $\phi = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$.

Vnější součin dvou vektorů $|\psi\rangle$ a $\langle\phi|$ je zapsán jako matice, která je vytvořena násobením každého prvku $|\psi\rangle$ s každým prvkem $\langle\phi|$.

Matematicky, $|\psi\rangle \langle\phi| = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} \begin{pmatrix} b_1^* & b_2^* & \cdots & b_m^* \end{pmatrix} = \begin{pmatrix} a_1 b_1^* & a_1 b_2^* & \cdots & a_1 b_m^* \\ a_2 b_1^* & a_2 b_2^* & \cdots & a_2 b_m^* \\ \vdots & \vdots & \ddots & \vdots \\ a_n b_1^* & a_n b_2^* & \cdots & a_n b_m^* \end{pmatrix}$, kde $\psi = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$ a $\phi = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$.

Zápis v Bra-ket notaci je vhodný pro seskupování operací. Například výraz $\langle u|v\rangle |w\rangle$ značí vnitřní součin vektorů u, v které jsou následně jako skalár násobeny vektorem w, nebo výraz $|u\rangle^{\otimes n}$ značí tensorový součin vektoru sám se sebou n-krát a představuje vektor $|u_1, u_2, \cdots, u_n\rangle$.

4.3 Ortonormální báze

V kontextu Hilbertova prostoru se báze skládá z ortogonálních a ortonormálních vektorů, které tvoří kompletní množinu pro popis všech možných stavů systému. Každý vektor v Hilbertově prostoru lze vyjádřit jako lineární kombinaci těchto bázových vektorů [27].

Formálně, nechť $\mathcal{H}^N$ je N-rozměrný Hilbertův prostor a $|\psi\rangle$ je libovolný vektor v $\mathcal{H}^N$. Potom říkáme, že množina vektorů $|\psi\rangle_{i=1}^N$ tvoří bázi pro $\mathcal{H}^N$, pokud platí:

- Ortogonalita: $\langle\phi_i|\phi_j\rangle = 0$ pro $i \neq j$.
- Normalizace: $\langle\phi_i|\phi_i\rangle = 1$ pro všechna i.
- Úplnost: Každý vektor $|\psi\rangle$ v $\mathcal{H}^N$ lze vyjádřit jako lineární kombinaci vektorů báze:
$$|\psi\rangle = \sum_{i=1}^{N} c_i |\phi_i\rangle,$$
kde $c_i \in \mathbb{C}$ jsou komplexní koeficienty.

V kvantovém počítání se používají různé báze. Některé běžné báze používané v kvantovém počítání zahrnují:

- Standardní báze (také známá jako výpočetní báze): Tato báze je tvořena vektory reprezentujícími základní stavy kvantového systému. Například ve čtyřrozměrném reálném Hilbertově prostoru (tj. $\mathbb{R}^4$) je standardní báze tvořena vektory $\phi_1 = (1, 0, 0, 0)$, $\phi_2 = (0, 1, 0, 0)$, $\phi_3 = (0, 0, 1, 0)$ a $\phi_4 = (0, 0, 0, 1)$.

- Hadamardova báze: Tato báze je tvořena vektory $|+\rangle$ a $|-\rangle$, které jsou lineární kombinací základních stavů. Pro 2-stavový kvantový systém vypadá báze:

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \tag{4.6}$$

, kde $|0\rangle$ a $|1\rangle$ představují vektory $\begin{pmatrix}1\\0\end{pmatrix}$ a $\begin{pmatrix}0\\1\end{pmatrix}$.

- Fázová báze: Tato báze se získá z Hadamardovy báze pomocí fázových rotací. Fázová báze je tvořena vektory $|i+\rangle$ a $|i-\rangle$, které jsou lineární kombinací základních stavů s komplexními koeficienty. Pro 2-stavový kvantový systém vypadá báze:

$$|i+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle), \quad |i-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i\,|1\rangle). \tag{4.7}$$

Volba vhodné báze závisí na konkrétním problému, který se snažíme řešit pomocí kvantového počítače.

4.4 Lineární Operátory

Operátor představuje lineární zobrazení A v Hilbertově prostoru $\mathcal{H}$ a lze si jej představit jako maticové násobení. Operátor má tyto vlastnosti [26]:

- Aditivita: zachovává sčítání vektorů. Pro všechny vektory $|u\rangle, |v\rangle \in H$ a pro lineární zobrazení $A : \mathcal{H} \to \mathcal{H}$, platí:

$$A(|u\rangle + |v\rangle) = A(|u\rangle) + A(|v\rangle). \tag{4.8}$$

- Homogenita: zachovává násobení skalárem. Pro všechny vektory $|u\rangle \in \mathcal{H}$ a skaláry $c \in \mathbb{C}$ a pro lineární zobrazení $A : \mathcal{H} \to \mathcal{H}$, platí:

$$A(c\,|u\rangle) = cA(|u\rangle). \tag{4.9}$$

Obecně řečeno, operátor $A \in \mathcal{H}$ je funkce, která každému vektoru $|\psi\rangle$ z prostoru $\mathcal{H}$ přiřadí jiný vektor $|\phi\rangle$ z tohoto prostoru přičemž zachová jeho strukturu a lze jej zapsat jako:

$$|\phi\rangle = A(|\psi\rangle). \tag{4.10}$$

Operátory v unitárních prostorech mají vlastní vektory a vlastní hodnoty.

Vlastní vektor lineárního zobrazení (operátoru) A je nenulový vektor $|u\rangle$, který se při aplikaci zobrazení A změní pouze o násobek, tedy:

$$A\,|u\rangle = \lambda\,|u\rangle\,, \tag{4.11}$$

kde λ je komplexní číslo, které se nazývá vlastní hodnota (vlastní číslo) přidružená k vlastnímu vektoru $|u\rangle$.

Dva hlavní typy operátorů používané v kvantovém počítání jsou unitární a hermitovské operátory.

Unitární operátory jsou zvláštní třída lineárních operátorů, které zachovávají skalární součin a tedy i délku a úhly mezi vektory. Unitární operátor U splňuje podmínku, že jeho adjungovaný (komplexně sdružený a transponovaný[2)]) operátor $U^\dagger$ je také jeho inverzní operátor U^{-1}, tedy $UU^\dagger = U^\dagger U = I$, kde I je identický operátor (jednotková matice).

Hermitovské operátory H jsou takové lineární operátory, které jsou rovny svému adjungovanému operátoru, tedy $H = H^\dagger$. Vlastní hodnoty hermitovských operátorů jsou vždy reálné a odpovídají fyzikálním hodnotám kvantových stavů, které můžeme naměřit.

[2)] Nechť A je matice 2x2: $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, pak adjungovaná matice k A je rovna $A^\dagger = \begin{pmatrix} a^* & c^* \\ b^* & d^* \end{pmatrix}$, kde a^*, b^*, c^* a d^* jsou komplexní sdružené hodnoty prvků a, b, c a d původní matice A.

5 ZÁKLADNÍ POJMY KVANTOVÝCH VÝPOČTŮ

5.1 Qubit: Kvantový bit

V klasickém počítači je základní jednotkou informace bit, který může nabývat hodnot 0 nebo 1. V kvantovém počítači je tato role předána qubitu (kvantovému bitu), který představuje kvantový analog klasického bitu. Na rozdíl od klasických bitů, které mohou existovat pouze v jednom ze dvou stavů (0 nebo 1), může qubit existovat v superpozici těchto stavů [30].

Matematicky je qubit reprezentován vektorem v 2-dimenzionálním Hilbertově prostoru se dvěma bázovými stavy, které jsou označeny jako:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{5.1}$$

kde $|\cdot\rangle$ představuje ket vektor ve standardní (výpočetní) bázi. Vektor (stav) $|0\rangle$ přestavuje logickou 0 a vektor (stav) $|1\rangle$ představuje logikou 1.

5.1.1 Superpozice

Princip superpozice je definován jako lineární kombinace dvou nebo více vektorů, která je zároveň dalším vektorem ve stejném Hilbertově prostoru, nebo-li představuje další stav kvantového systému [31].

Stav qubitu odpovídá lineární kombinací těchto bázových stavů:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{5.2}$$

kde α a β jsou komplexní koeficienty, které jsou známé jako amplitudy pravděpodobnosti, a které popisují váhy, s jakými se stav qubitu skládá z bázových stavů. Aby qubit reprezentoval platný kvantový stav, musí splňovat podmínku normalizace:

$$|\alpha|^2 + |\beta|^2 = 1. \tag{5.3}$$

Qubit ve stavu superpozice může současně reprezentovat více informací, zahrnuje jak stav $|0\rangle$, tak stav $|1\rangle$, a to současně a tím umožňuje paralelní výpočty. Qubity tedy poskytují značný výpočetní potenciál.

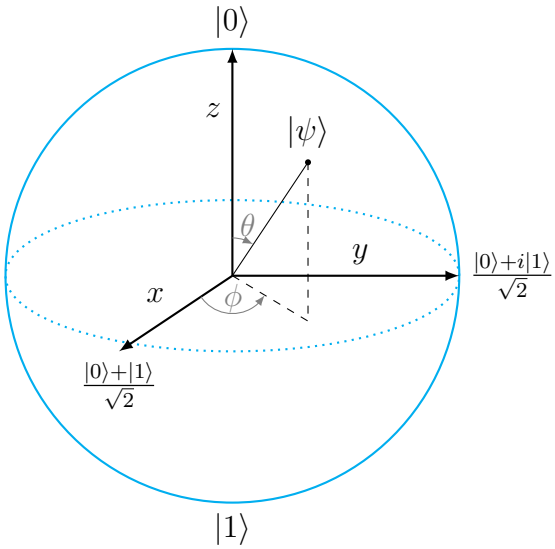
5.1.2 Zobrazení qubitu v Blochově sféře

Blochova sféra je geometrická reprezentace qubitu, která poskytuje názornou vizualizaci kvantových stavů. Stav qubitu v Blochově sféře je reprezentován jako bod na povrchu jednotkové sféry, kde poloha bodu závisí na koeficientech α a β.

Pomocí Eulerových úhlů θ a ϕ lze stav qubitu vyjádřit jako [30]:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle\,, \tag{5.4}$$

kde θ a ϕ jsou reálné úhly z intervalu $0 \leq \theta \leq \pi$ a $0 \leq \phi < 2\pi$.



Obrázek 5.1 Blochova sféra

Na Blochově sféře se stavy standardní báze (základní stav) $|0\rangle$ a $|1\rangle$ nacházejí na severním a jižním pólu sféry, zatímco smíšené stavy, například Hadamardova báze $|+\rangle$ a $|-\rangle$, se nacházejí na rovníku [31].

5.1.3 Manipulace s qubity

Operace na jednom qubitu lze zobrazit jako rotace kolem osy Blochovy sféry o určitý úhel. Tímto způsobem lze lépe ilustrovat účinky různých kvantových operátorů na qubit. Například působení Pauliho X operátoru (kvantový ekvivalent klasického NOT hradla) na qubit odpovídá otočení stavového vektoru o 180° kolem x-ové osy (úhel θ na obr. 5.1) Blochovy sféry.

Pauliho X operátor je reprezentován následující maticí:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{5.5}$$

Aplikace Pauliho X operátoru na qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ je pak popsána v Bra-ket notaci následujícím způsobem:

$$X|\psi\rangle = X(\alpha|0\rangle + \beta|1\rangle) = \alpha X|0\rangle + \beta X|1\rangle = \alpha|1\rangle + \beta|0\rangle. \tag{5.6}$$

Tento přístup umožňuje snadno popsat a provádět složitější kvantové operace a algoritmy.

5.2 Kvantový registr

Kvantový registr je soubor qubitů, který slouží jako základní jednotka pro ukládání a zpracování informací v kvantovém počítači. Kvantový registr umožňuje qubitům nejen existovat v superpozici stavů, ale i využívat kvantovou provázanost.

Matematicky kvantový registr je Hilbertův prostor $\mathcal{H}^N$, kde $N = 2^n$ je počet dimenzí prostoru a n je počet qubitů v registru [26].

Stav kvantového registru s n qubity je reprezentován jako tensorový součin jednotlivých qubitů:

$$|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle, \tag{5.7}$$

kde $|\psi_i\rangle$ jsou jednotlivé qubity ve stavu $\alpha_i|0\rangle + \beta_i|1\rangle$.

Kvantový registr o 2 qubitech má ve standardní bázi 4 stavy:

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\cdot 1\\1\cdot 0\\0\cdot 1\\0\cdot 0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}. \tag{5.8}$$

Stejné platí pro další kombinace 2-qubitového registru:

$$|01\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \quad |10\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \quad |11\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix} \tag{5.9}$$

5.2.1 Inicializace kvantového registru

Kvantový registr se obvykle inicializuje do základního stavu, který je tvořen stavem $|0\rangle$ všech qubitů. Tento stav můžeme zapsat jako:

$$|\Psi_0\rangle = |0\rangle^{\otimes n} = |0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle . \tag{5.10}$$

Po inicializaci můžeme na kvantový registr aplikovat kvantové operátory, které manipulují se stavem qubitů v registru a umožňují provádění kvantových výpočtů.

5.2.2 Měření kvantového registru

Měření stavu kvantového registru je základní proces, kterým získáváme informace o jeho stavech. Měření způsobuje změnu kvantového systému z lineární kombinace všech stavů (kolaps vlnové funkce) na jeden ze základních stavů s určitou pravděpodobností, která je dána absolutní hodnotou amplitudy daného stavu.

Pro kvantový registr s n qubity můžeme měření zapsat jako:

$$|\Psi\rangle = \sum_{i=0}^{2^n-1} c_i |i\rangle \xrightarrow{\text{měření}} |k\rangle , \tag{5.11}$$

kde $|i\rangle$ jsou základní stavy qubitů, c_i jsou amplitudy těchto stavů a $|k\rangle$ je jeden ze základních stavů, na který kolabuje vlnová funkce s pravděpodobností $P_i = |c_i|^2$.

Příklad 5.1 Představme si kvantový registr se 2 qubity, který je ve stavu:

$$|\Psi\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle . \tag{5.12}$$

Měření tohoto registru způsobí kolaps vlnové funkce na jeden ze stavů $|00\rangle$, $|01\rangle$ nebo $|10\rangle$. Pravděpodobnosti kolapsu na jednotlivé stavy jsou:

$$P(|00\rangle) = \left|\frac{1}{2}\right|^2 = \frac{1}{4}$$
$$P(|01\rangle) = \left|\frac{1}{2}\right|^2 = \frac{1}{4}$$
$$P(|10\rangle) = \left|\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2}$$

Po měření se stav kvantového registru změní na jeden z těchto stavů. Například, pokud měření vyústí v stav $|01\rangle$, registr se ocitne ve stavu $|\Psi'\rangle = |01\rangle$.

Vlnová funkce v příkladu neobsahuje nenulový koeficient pro stav $|11\rangle$, tudíž je

nepravděpodobné, že by registr zkolaboval do stavu $|11\rangle$.

Jednou změřený a zkolabovaný stav kvantového registru nelze vrátit zpět do původního stavu superpozice. Měřením získáváme pouze jednu z mnoha možných hodnot, které kvantový registr reprezentoval.

5.2.3 Kvantová paralelnost

Kvantová paralelnost je vlastnost kvantového počítání, která umožňuje kvantovým počítačům zpracovávat více informací současně. Díky superpozici qubitů může kvantový registr reprezentovat mnoho klasických stavů najednou. Výpočetní operace lze provádět na všech těchto stavech paralelně.

Zatímco v klasickém počítači přesouváme data mezi paměťmi a procesorem, tak v rámci kvantového počítače se veškeré počítání odehrává na všech qubitech zároveň. Výpočet popisuje tzv. časová evoluce kvantového systému [32].

5.2.4 Kvantová provázanost

Kvantová provázanost (z angl. entanglement) popisuje vzájemně závislé kvantové stavy dvou nebo více částic, které tvoří vázaný systém. Informace o jednom qubitu nelze získat nezávisle na informacích o provázaném qubitu, i když jsou tyto částice odděleny velkou vzdáleností. Tento jev, který Albert Einstein nazval „strašidelným působením na dálku“[1)], má zásadní význam pro kvantovou komunikaci, kvantové šifrování a mnoho dalších kvantových výpočtů [31].

Představme si, že máme dva qubity v a w ve stavu superpozice v registru Ψ. Již víme, že pravděpodobnostní amplitudy obou qubitů (komplexní koeficienty c_0, c_1 qubitu v a d_0, d_1 qubitu w) musí splnit podmínku normalizace (rov. 5.2) a registr představuje tensorový součin qubitů (rov. 5.8).

$$\begin{aligned}\Psi = |v\rangle \otimes |w\rangle &= (c_0 |v_0\rangle + c_1 |v_1\rangle) \otimes (d_0 |w_0\rangle + d_1 |w_1\rangle) \\ &= (c_0 |0\rangle + c_1 |1\rangle) \otimes (d_0 |0\rangle + d_1 |1\rangle) \\ &= c_0 d_0 |0\rangle \otimes |0\rangle + c_0 d_1 |0\rangle \otimes |1\rangle + c_1 d_0 |1\rangle \otimes |0\rangle + c_1 d_1 |1\rangle \otimes |1\rangle\end{aligned} \tag{5.13}$$

Výraz přepišme do konvenčního tvaru

$$\Psi = c_0 d_0 |00\rangle + c_0 d_1 |01\rangle + c_1 d_0 |10\rangle + c_1 d_1 |11\rangle \tag{5.14}$$

a nahraďme součin koeficientů jediným symbolem, tedy $c_0 d_0 = \alpha$, $c_0 d_1 = \beta$, $c_1 d_0 = \gamma$

[1)] V orig. „spooky action at a distance“ se citace objevila na přední stránce 4. května 1935 The New York Times v článku s titulkem *Einstein Attacks Quantum Theory* [31]

a $c_1 d_1 = \delta$

$$\Psi = \alpha \left|00\right\rangle + \beta \left|01\right\rangle + \gamma \left|10\right\rangle + \delta \left|11\right\rangle . \tag{5.15}$$

Výraz zachovává podmínku normalizace, neboť $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$. Zároveň platí, že $\alpha\delta = \beta\gamma = c_0 c_1 d_0 d_1$.

Idea kvantové provázanosti dvou qubitů vychází z jejich tensorového součinu $\alpha \left|00\right\rangle + \beta \left|01\right\rangle + \gamma \left|10\right\rangle + \delta \left|11\right\rangle$, kde stále platí podmínka normalizace $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$, ale $\alpha\delta \neq \beta\gamma$.

Názorně si uvedeme dva příklady [32].

Příklad 5.2 2-qubitový kvantový registr $\Psi = \left|vw\right\rangle$, kde jsou pravděpodobnostní amplitudy $\alpha\delta = \beta\gamma$, je popsán vlnovou funkcí:

$$\Psi = \frac{1}{2\sqrt{2}} \left|v_0 w_0\right\rangle + \frac{\sqrt{3}}{2\sqrt{2}} \left|v_0 w_1\right\rangle + \frac{1}{2\sqrt{2}} \left|v_1 w_0\right\rangle + \frac{\sqrt{3}}{2\sqrt{2}} \left|v_1 w_1\right\rangle . \tag{5.16}$$

Součin vnitřních a součin vnějších pravděpodobnostních amplitud je roven $\sqrt{3}/8$, tedy qubity nejsou provázány.

Změříme-li registr Ψ, dostaneme jeden ze stavů s danou pravděpodobností $P(\left|00\right\rangle) = 1/8$, $P(\left|01\right\rangle) = 3/8$, $P(\left|10\right\rangle) = 1/8$, nebo $P(\left|11\right\rangle) = 3/8$.

Též je možné změřit pouze jeden qubit v registru. Upravíme matematický výraz tak, že si jeden qubit obrazně řečeno vytkneme před závorku.

$$\left|v_0\right\rangle \left(\frac{1}{2\sqrt{2}} \left|w_0\right\rangle + \frac{\sqrt{3}}{2\sqrt{2}} \left|w_1\right\rangle \right) + \left|v_1\right\rangle \left(\frac{1}{2\sqrt{2}} \left|w_0\right\rangle + \frac{\sqrt{3}}{2\sqrt{2}} \left|w_1\right\rangle \right) \tag{5.17}$$

Abychom v závorkách dostali jednotkové vektory, vydělíme jejich obsah normou (vzdáleností vektorů).

$$\frac{1}{\sqrt{2}} \left|v_0\right\rangle \left(\frac{1}{2} \left|w_0\right\rangle + \frac{\sqrt{3}}{2} \left|w_1\right\rangle \right) + \frac{1}{\sqrt{2}} \left|v_1\right\rangle \left(\frac{1}{2\sqrt{2}} \left|w_0\right\rangle + \frac{\sqrt{3}}{2\sqrt{2}} \left|w_1\right\rangle \right) \tag{5.18}$$

$$\left(\frac{1}{\sqrt{2}} \left|v_0\right\rangle + \frac{1}{\sqrt{2}} \left|v_1\right\rangle \right) \left(\frac{1}{2\sqrt{2}} \left|w_0\right\rangle + \frac{\sqrt{3}}{2\sqrt{2}} \left|w_1\right\rangle \right) \tag{5.19}$$

Z upraveného výrazu je patrné, že qubity nejsou v provázaném stavu, neboť zde máme tensorový součin qubitu v a qubitu w. Provedeme-li měření pouze qubitu v dostaneme stav 0, nebo 1 se stejnou pravděpodobností aniž bychom ovlivnili pravděpodobnosti měření qubitu w.

Příklad 5.3 2-qubitový kvantový registr $\Phi = |xy\rangle$, kde jsou pravděpodobnostní amplitudy $\alpha\delta \neq \beta\gamma$, ve stavu:

$$\Phi = \frac{1}{2}|x_0y_0\rangle + \frac{1}{2}|x_0y_1\rangle + \frac{1}{2}|x_1y_0\rangle + 0\,|x_1y_1\rangle\,. \tag{5.20}$$

V tomto případě se vnitřní a vnější součin pravděpodobnostních amplitud liší, tudíž qubity v kvantovém registru jsou provázány.

Pokud změříme oba qubity x a y najednou, dostaneme pravděpodobnosti stavu registru $P(|00\rangle) = P(|01\rangle) = 1/4$, $P(|10\rangle) = 1/2$ a $P(|11\rangle) = 0$.

Nyní se podíváme na případ, kdy změříme pouze qubit x v provázaném registru

$$|x_0\rangle\left(\frac{1}{2}|y_0\rangle + \frac{1}{2}|y_1\rangle\right) + |x_1\rangle\left(\frac{1}{\sqrt{2}}|y_0\rangle + 0\,|y_1\rangle\right). \tag{5.21}$$

Výraz upravíme obdobně jako v předchozím příkladu, tak abychom dostali v závorkách jednotkové vektory

$$\frac{1}{\sqrt{2}}|x_0\rangle\left(\frac{1}{\sqrt{2}}|y_0\rangle + \frac{1}{\sqrt{2}}|y_1\rangle\right) + \frac{1}{\sqrt{2}}|x_1\rangle\left(1\,|y_0\rangle + 0\,|y_1\rangle\right). \tag{5.22}$$

Pravděpodobnost změření qubitu x je stejná, jak pro stav 0, tak pro stav 1. Změříme-li na qubitu stav $x = 0$, pak pro registr Φ bude platný stav $|x_0\rangle\left(\frac{1}{\sqrt{2}}|y_0\rangle + \frac{1}{\sqrt{2}}|y_1\rangle\right)$ a opačně pokud změříme stav $x = 1$, pak bude registr ve stavu $|x_1\rangle\left(1\,|y_0\rangle + 0\,|y_1\rangle\right)$.

V tomto případě jsou qubity x a y provázané, protože jejich společný stav nelze rozdělit na součin stavů jednotlivých qubitů. Výsledkem je, že měření jednoho z qubitů okamžitě určuje stav druhého qubitu.

Provázanost mezi qubity v kvantovém registru umožňuje koordinaci a sdílení informací mezi qubity. V praktické části je popsán kvantový algoritmus odhadu fáze, který dokonce využívá provázanosti dvou kvantových registru.

5.2.5 Chyby a dekoherence v kvantových registrech

Kvantové registry jsou náchylné k chybám a dekoherenci způsobeným vnějšími vlivy, jako jsou tepelné šumy, vibrace nebo elektromagnetické pole. Tato interakce s okolním prostředím může vést ke ztrátě kvantové informace a koherence qubitů.

Kvantová chybová korekce je důležitým nástrojem pro udržení koherence a spolehlivosti kvantových počítačů. Existuje řada technik a protokolů pro korekci chyb v kvantových registrech, jako jsou kvantové chybové korekční kódy a kvantové opakovací kódy. Tyto metody umožňují detekovat a opravit chyby, aniž by došlo ke ztrátě kvantové

informace.

5.3 Kvantové operátory a obvody

Již jsme si popsali operátory jako matematické objekty (matice), které umožňují manipulaci se stavem kvantových systémů. Na hardwarové úrovni kvantových počítačů existuje několik kvantových bran (hradel), které přestavují konkrétní kvantové operátory, a které se často používají jako stavební bloky pro konstrukci složitějších kvantových operátorů.

V rámci této práce zanedbáme hardwarový aspekt a nadále pojmy kvantový operátor a hradlo (příp. brána) budeme uvažovat jako ekvivalentní.

5.3.1 Základní kvantové operátory

Zde je přehled operátorů, které se často používají v kvantových obvodech [33]:

- Pauliho operátory: Pauliho matice X, Y, Z jsou jedno-qubitové (působí na jeden qubit) operátory, které rotují qubit kolem osy Blochovy sféry. Matice jsou definovány následovně:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{5.23}$$

- Rotace R jsou jedno-qubitové operátory, které mění fázi (úhel) stavu qubitu. Aplikace operátoru rotuje pouze bázový stav $|1\rangle$ o úhel φ. Matice tohoto operátoru je následující:

$$R_\varphi = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{pmatrix}. \tag{5.24}$$

  Mezi běžně používané rotace patří S a T, jejichž matice vypadají následovně:

$$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \quad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}. \tag{5.25}$$

  Operátor S rotuje stav o 90° pro $\varphi = \pi/2$ podél osy z a operátor T pro $\varphi = \pi/4$ rotuje podél osy z o 45°. Pokud $\varphi = \pi$, dostaneme Pauli Z operátor.

- Hadamardův operátor H je jedno-qubitový operátor, který transformuje základní stavy na superpozice těchto stavů. Hadamard je definován následovně:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{5.26}$$

- SWAP je dvou-qubitový operátor, který prohodí stavy dvou qubitů. SWAP je definován následovně:

$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{5.27}$$

- CNOT (řízený NOT) je dvou-qubitový operátor, který aplikuje negaci (NOT) na druhý qubit, pokud je první qubit ve stavu $|1\rangle$. CNOT operátor je definován následovně:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{5.28}$$

- Toffoliho operátor je tří-qubitový operátor, který aplikuje negaci (NOT) na třetí qubit, pokud jsou první a druhý qubit ve stavu $|1\rangle$. Toffoliho operátor je známý také jako řízený CNOT:

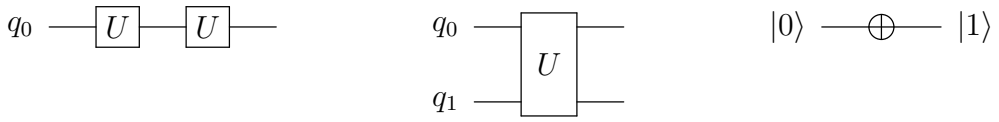
$$\text{Toffoli} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{5.29}$$

5.3.2 Kvantové obvody

Kvantové obvody se skládají z posloupnosti kvantových operátorů, které se aplikují na kvantové registry [31].

Pro vizualizaci kvantových obvodů lze použít knihovnu *Q-circuit* [34] v LaTeXu (program pro formátování textu a tiskové sazby).

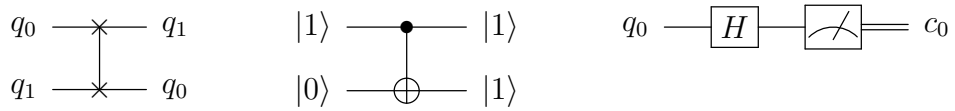
Kvantové obvody se znázorňují pomocí diagramů, ve kterých mohou být operátory aplikovány sekvenčně a nebo paralelně (obr. 5.2).



(a) Série *(b) Paralela* *(c) NOT*

Obrázek 5.2 Popis kvantových obvodů - aplikace operátorů.

Operátory se zpravidla zapisují do boxů s výjimkou negace (Pauli X operátor), který se značí $\oplus$ symbolem.



(d) SWAP *(e) CNOT* *(f) Měření*

Obrázek 5.3 Popis kvantových obvodů - binární operace a měření.

V případě CNOT (řízený NOT) je první qubit označen řídicím symbolem $\bullet$ a druhý qubit je označen cílovým symbolem $\oplus$.

Symbol měření na konci obvodu vrací binární hodnotu (0, nebo 1) do klasického registru, který je se značí dvojitou čárou.

Dílčí závěr

Díky zobrazení kvantových stavů v Hilbertově prostoru a zápisu pomocí bra-ket notace máme pevný základ pro pochopení konceptů qubitu, kvantových registrů a operátorů (hradel), které se používají v kvantových obvodech. Konkrétní kvantový obvod může být dílčí částí většího celku, který dohromady tvoří algoritmus spustitelný na kvantovém počítači.

V praktické části této práce se podrobněji podíváme na problematiku kvantových algoritmů.

II. PRAKTICKÁ ČÁST

6 ÚVOD DO PROBLEMATIKY KVANTOVÝCH ALGORITMŮ

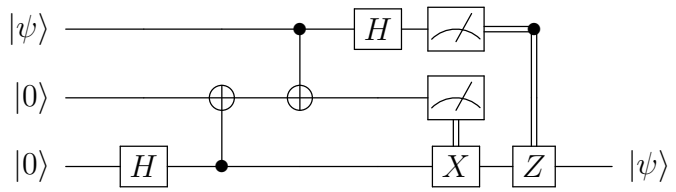
V teoretické části této práce byly popsány základní stavební kameny, ze kterých sestávají kvantové algoritmy. V praktické části navážeme na tyto poznatky z předešlých kapitol a na příkladu implementace Shorova kvantového algoritmu demonstrujeme schopnost kvantových počítačů ohrozit bezpečnost populárních krypto-schémat.

Kvantový algoritmus je matematický algoritmus určený pro výpočet na kvantovém počítači. Oproti klasickým algoritmům, které pracují s bity, kvantové algoritmy pracují s kvantovými bity (qubity), ale také s klasickými bity. Qubity jsou jednotky informace v kvantových systémech. Qubity mohou existovat ve stavu nuly, jedničky, nebo v superpozici obou stavů současně, což umožňuje kvantovým algoritmům provádět některé úlohy rychleji a efektivněji než klasické algoritmy.

Vývoj kvantových algoritmů je stále velmi aktivní oblastí výzkumu a využití kvantových počítačů se očekává v mnoha oblastech jakožto revoluční technologie s potenciálem vyřešit problémy, které by jinak byly neřešitelné s použitím klasických počítačů.

Kvantové algoritmy sestávají z kvantových obvodů, což si lze představit jako instrukce nízko úrovňového programovacího jazyka, které provádějí operace nad jednotlivými qubity.

Technicky lze popsat kvantový algoritmus jako proces efektivní dekompozice složité unitární transformace do součinů elementárních unitárních operací, které provádějí jednoduché lokální změny [27].



Obrázek 6.1 Ukázka kvantového obvodu

Kvantové obvody zapisujeme pomocí matematických rovnic a matic, které popisují chování jednotlivých kvantových operací.

Kvantové algoritmy lze programovat pomocí řady programovacích jazyků například Python, nebo Q# a frameworku, jako je například Qiskit nebo Cirq, který obsahuje potřebné knihovny a nástroje pro návrh kvantových obvodů. V našem případě pro implementaci Shorova algoritmu použijeme Qiskit.

Qubit, kvantový obvod a jednotlivé kvantové operátory jsou detailněji popsány v teoretické části v kapitole Základní pojmy kvantového počítání.

6.1 Platforma pro návrh a testovaní kvantových obvodů Qiskit

Qiskit (z angl. Quantum information software kit for quantum computation) je framework určený k programování kvantových počítačů a je vyvíjený společností IBM. Jedná se o rozsáhlý toolkit, který umožňuje vývoj kvantových algoritmů a programování kvantových obvodů pomocí jazyka Python, což usnadňuje tvorbu a testování nových kvantových algoritmů. Uživatelé mohou snadno vytvářet, vizualizovat a spouštět kvantové obvody, včetně simulace a optimalizace výkonu.

Qiskit také poskytuje rozhraní pro komunikaci s kvantovými zařízeními IBM Quantum Experience. Tato integrace umožňuje uživatelům spouštět své programy na skutečných kvantových počítačích a získávat data z experimentů.

Celkově lze říci, že Qiskit je jedním z nejvýkonnějších a nejrozšířenějších open-source vývojových prostředí pro kvantové počítače na trhu. Je snadno použitelný, modulární a má širokou podporu ze strany komunity.

Qiskit je možné používat plně on-line ve webové aplikaci IBM Quantum Lab[1)], nebo instalovat na lokální počítač. Framework Qiskit vyžaduje programovací jazyk Python ve verzi 3.7 nebo vyšší.

Qiskit je již obsažen v distribuci Anaconda[2)]. Instalace samostatného balíčku Qiskit je standardní a lze ji snadno provést pomocí příkazové řádky, tak jako u ostatních populárních knihoven pro Python.

Například pro OS Linux lze použít terminál a příkaz:

```
$ pip install -U pip && pip install qiskit
```

6.2 Kvantové algoritmy využitelné pro odkrytí chráněné informace

Podívejme se na konkrétní kvantové algoritmy, které mají dopad v oblasti kryptografie a mohou být použity k odkrytí chráněné informace. Tyto algoritmy využívají specifických vlastností kvantových systémů, jako je superpozice a kvantová paralelnost, aby byly schopny vykonávat operace, které jsou pro klasické počítače nepraktické nebo nemožné.

[1)] Aplikace IBM Quantum Lab (dostupná na webové adrese https://lab.quantum-computing.ibm.com) je online vývojové prostředí poskytované společností IBM pro vývoj kvantových programů a experimentování s kvantovými algoritmy. Webová aplikace umožňuje uživatelům vytvářet, spouštět a vizualizovat kvantové obvody bez nutnosti instalovat software na vlastním počítači.

[2)] Anaconda (dostupné na webové adrese: https://www.anaconda.com/products/distribution) je distribuce programovacího jazyka Python, která obsahuje mnoho balíčků a knihoven často používaných v datové vědě a analýze dat. Jedná se o open-source software, který umožňuje snadnou instalaci Pythonu a jeho nejčastěji používaných knihoven, jako jsou například Numpy, Scipy, Pandas, Matplotlib, a další.

6.2.1 Groverův algoritmus

Groverův algoritmus, navržený v roce 1996 Lovem Groverem [19], je kvantový algoritmus, který řeší problém hledání v neuspořádaném seznamu s kvadratickým zrychlením oproti klasickým algoritmům. Toto zrychlení může mít důsledky pro určitá kryptoschémata, která závisí na obtížnosti nalezení správného řešení mezi velkým počtem možností.

Groverův algoritmus pracuje na n-qubitovém kvantovém registru a může najít označený prvek v seznamu $N = 2^n$ položek se složitostí $\mathcal{O}(\sqrt{N})$ iterací. Algoritmus se skládá ze tří hlavních kroků [31]:

1. Inicializace kvantového registru do rovnoměrné superpozice všech stavů pomocí aplikace Hadamardova operátoru na každý qubit:

$$|\psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \,. \tag{6.1}$$

2. Iterativní aplikace Groverova operátoru $G = (2\,|\psi_0\rangle\,\langle\psi_0| - I)O$, kde O je operátor zvaný Oracle, který označí správný prvek (obrací znaménko správného řešení), a I je jednotkový operátor. Groverův operátor se aplikuje přibližně $R = \lfloor \frac{\pi}{4}\sqrt{N} \rfloor$ krát[3)].

3. Měření kvantového registru, které s vysokou pravděpodobností zkolabuje do stavu označujícího správný prvek.

Groverův algoritmus může být použit pro prolomení symetrického šifrování, jako je například Data Encryption Standard (DES). DES používá 56bitový klíč, což znamená, že existuje 2^{56} možných klíčů. Klasický brute-force útok by vyžadoval $\mathcal{O}(2^{56})$ operací pro nalezení správného klíče. S Groverovým algoritmem lze najít správný klíč s $\mathcal{O}(2^{28})$ operacemi, tím se značně zvyšuje jeho zranitelnost.

6.2.2 Simonův algoritmus

Simonův algoritmus je kvantový algoritmus navržený v roce 1994 Danielem Simonem [35]. Jeho hlavním cílem je najít skrytý periodický vzor (XOR masku) S v tzv. Black boxu (funkci f) za předpokladu, že $f(x) = f(x \oplus S)$ pro všechny $x \in 0, 1^n$ [31].

Algoritmus funguje následovně:

1. Připravíme dva kvantové registry, každý s n qubity: $|0^n\rangle\,|0^n\rangle$.

[3)] Zápis $\lfloor x \rfloor$ představuje funkci dolní celá část, pro kterou platí $\lfloor x \rfloor = n$, právě když $n \leq x < n+1$, kde $x \in \mathbb{R}$ a $n \in \mathbb{Z}$.

2. V prvním kvantovém registru aplikujeme Hadamardovy operátory na každý qubit, získáme stav $\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |0^n\rangle$.
3. Použijeme Black box na druhý registr, získáme stav $\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |f(x)\rangle$.
4. Na prvním kvantovém registru provedeme měření, získáme nějaké x_0. Druhý registr zůstane v superpozici.
5. Na druhém registru aplikujeme kvantovou Fourierovu transformaci[4)] a provedeme měření. Získáme nějaké y_0.
6. Provedeme post-processing na klasickém počítači, kde nalezneme vzor S pomocí získaných hodnot x_0 a y_0.

Problém, který řeší Simonův algoritmus, je významný, neboť souvisí s řadou kryptografických problémů, například s problémem rozšířeného eukleidovského algoritmu. Simonův algoritmus na kvantovém počítači nalezne skrytý vzor v lineárním čase, zatímco nejlepší známý klasický algoritmus potřebuje exponenciální čas.

Tento algoritmus byl prvním, který ukázal, že kvantové počítače mohou být rychlejší než klasické počítače, a byl inspirací pro pozdější Shorův algoritmus.

6.2.3 Shorův algoritmus

Shorův algoritmus je kvantový algoritmus, který byl poprvé publikován v roce 1994 Peterem Shorem [21]. Jedná se o jeden z nejdůležitějších kvantových algoritmů a je považován za první praktický kvantový algoritmus, který dokáže významně zrychlit určité výpočty oproti klasickým algoritmům.

Algoritmus se specializuje na řešení tzv. faktorizačního problému, který spočívá v rozkladu velkého celého čísla na součin prvočísel. K řešení faktorizace velkých čísel využívá kvantového paralelismu. Pomocí jevu kvantové superpozice a kvantové provázanosti dokáže Shorův algoritmus faktorovat mnohem rychleji, než to dokáže klasický algoritmus[5)]. Tato úloha je klíčová pro řadu kryptoschémat, jako je například šifrování RSA. Proto je Shorův algoritmus považován za hrozbu pro klasickou kryptografii, ale zároveň je považován za jednu z hlavních motivací pro výzkum na poli post-kvantové kryptografie [31].

[4)] Kvantová Fourierova transformace je podrobně popsána v rámci kapitoly Implementace Shorova algoritmu.

[5)] Nejrychlejší dosud známý klasický algoritmus pro rozklad velkých čísel je GNFS (z angl. General Number Field Sieve)

7 IMPLEMENTACE SHOROVA FAKTORIZAČNÍHO ALGORITMU

Vysvětlíme si podrobně princip fungování a matematický základ Shorova faktorizačního algoritmu a implementujeme jej na platformě IBM Quantum Experience, jako programovací jazyk použijeme Python 3.7+ v kombinaci s vývojářskými nástroji Qiskit (z angl. Quantum Information Software Kit for quantum computation).

Poznámka: Veškerý programový kód je čerpán z dokumentace frameworku Qiskit [36] a vlastních poznámek z webináře *Qiskit Summer School 2020* [37]. Popis programových tříd a metod čerpá z knihy *Learn Quantum Computing with Python and IBM Quantum Experience: A hands-on introduction to quantum computing and writing your own quantum programs with Python* [38].

Cílem Shorova faktorizačního algoritmu je rozložit celé číslo N na prvočísla (faktory) p a q.

U každého celého čísla lze provést prvočíselný rozklad, nicméně Shorův algoritmus předpokládá vstup N, který je poloprvočíslem, tedy součinem dvou prvočísel.

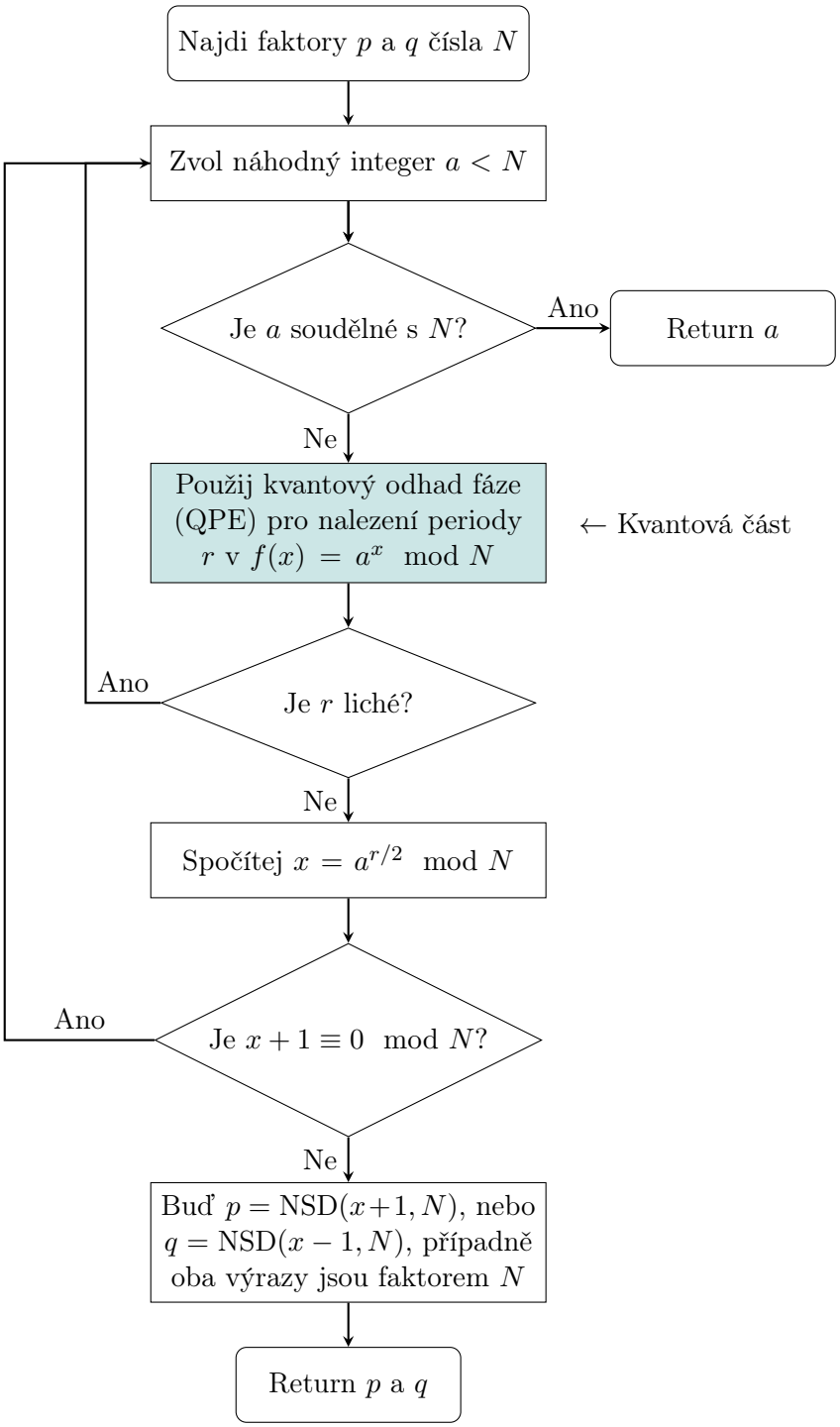
Shorův faktorizační algoritmus lze rozdělit na klasickou a kvantovou část.

Klasická část algoritmu spočívá v nalezení vhodného náhodného čísla $a < N$ a ověření, zda je toto číslo nesoudělné s rozkládaným číslem N. Pokud tomu tak není, algoritmus vrací nalezený společný dělitel. Úkolem klasické části algoritmu je na jedné straně je připravit vstup do kvantové části a na druhé straně zpracovat a verifikovat výstup z kvantové části. Časová složitost klasické části je polynomiálně závislá na počtu bitů rozkládaného čísla.

V kvantové části Shorova algoritmu se pracuje s kvantovou Fourierovou transformací, která převádí funkce v oboru čísel do frekvenčního spektra. Tato transformace se aplikuje na na qubity v superpozici stavů, ve kterých jsou uloženy všechny možné koeficienty v rovině kladného a záporného celého čísla. Inicializaci superpozice stavů zajišťuje kvantový obvod odhadu fáze obsahující tzv. modulární exponenciátor. Výstup tohoto obvodu je měřen a poté zpracován klasickým algoritmem pro nalezení faktorů rozkládaného čísla. Detailně si tento kvantový obvod popíšeme v následujících kapitolách pomocí příkladů.

S využitím kvantového paralelismu, může být náročnost Shorova algoritmu v řádu polynomiální časové složitosti, což je obrovské zlepšení oproti klasickým algoritmům, které jsou exponenciální.

Jednotlivé kroky Shorova algoritmu jsou uvedeny v diagramu (obr. 7.1).



Obrázek 7.1 Vývojový diagram Shorova foktorizačního algoritmu

Shorův algoritmus postupuje při faktorizaci čísla N následujícím způsobem:

1. Vybere náhodné celé číslo $a < N$ a ověří, zda je nesoudělné s číslem N. Pokud je náhodou číslo a soudělné s číslem N, máme nalezen faktor a algoritmus může skončit.

2. Vypočítá kvantovou Fourierovu transformaci na zadaném celém čísle N a tím vytvoří superpozici všech možných hodnot periody r funkce $f(x) = a^x \mod N$.

3. Změří výslednou superpozici, čímž získá konkrétní hodnotu periody r.

4. Pokud je možné z hodnoty r vypočítat faktor, algoritmus skončí. V opačném případě se vrátí na krok 1 a opakujeme celý postup.

Nejprve se podíváme na princip Shorova algoritmu a periodické funkce, poté si popíšeme kvantovou část, která používá kvantovou Fourierovu transformaci k určení periodických vzorců v kvantových stavech. Nakonec sestavíme kód Shorova algoritmu a provedeme kontrolní experiment.

7.1 Princip Shorova algoritmu

Princip Shorova algoritmu spočívá v převedení matematického problému faktorizace čísel na jiný snadněji řešitelný matematický problém. Algoritmus staví na poznatcích z teorie čísel[1)], zejména využívá periodické funkce, které se vyskytují v rozkladech celých čísel na prvočíselné faktory. Algoritmus dokáže najít periodu této funkce pomocí kvantových výpočtů. Perioda je poté použita k nalezení prvočíselného faktoru původního čísla.

7.1.1 Periodická funkce

Máme periodickou funkci $f(x)$ a chceme určit její periodu r.

Definice 7.1 [39] Řekneme, že funkce f je periodická s periodou $r \in (0, \infty)$, právě když pro každé $x \in D_f$ platí $x \pm r \in D_f \wedge f(x \pm r) = f(x)$.

Pro periodické funkce také obecně platí následující:

Je-li funkce f periodická s periodou r, je funkce f periodická i s periodou $k \cdot r$ pro libovolné $k \in \mathbb{N}$. Nejmenší číslo r splňující podmínky definice 2.1 nazýváme základní periodou.

Většina lidí si pod periodickou funkcí představí některou goniometrickou funkci,

[1)] Teorie čísel je matematický obor, který se věnuje vztahům mezi celými čísly.

například funkci sinus, kde je perioda patrná na první pohled. Ve skutečnosti tento zdánlivě jednoduchý problém je velmi komplikovaný. Zkusme na následujícím jednoduchém příkladu demonstrovat složitost problému.

Příklad 7.1 Najděme periodu r této funkce $f(x) := 3^x \mod 35$, kde $x \in \mathbb{N}_0$.

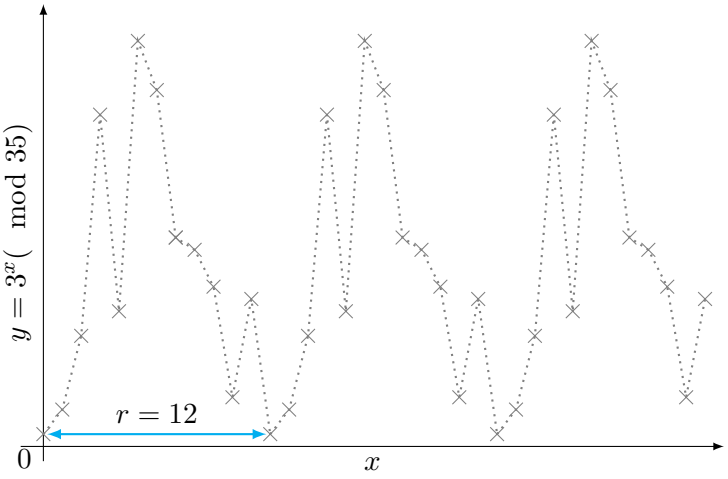
$$\begin{array}{l|l} 3^0 \mod 35 & 1 \\ 3^1 \mod 35 & 3 \\ 3^2 \mod 35 & 9 \\ 3^3 \mod 35 & 27 \\ 3^4 \mod 35 & 11 \\ 3^5 \mod 35 & 33 \\ 3^6 \mod 35 & 29 \\ 3^7 \mod 35 & 17 \\ 3^8 \mod 35 & 16 \\ \vdots & \vdots \end{array}$$

Zde již na první pohled není patrné, jaká je perioda funkce a také zda se vůbec jedná o periodickou funkci.

Skutečně jde o funkci s periodou $r = 12$ (viz obr. 7.2).

Není znám žádný efektivní algoritmus, který by vypočítal periodicitu. Periodu r je rozumné zkusit odhadnout na základě dostupných informací o dané funkci a následně předpoklad potvrdit, nebo vyvrátit pomocí definice 7.1.

Pokud bychom o funkci u níž odhadujeme periodicitu neměli žádné další informace pro kvalitní odhad periody, pak by v nejzazším případě měl algoritmus odhadu a potvrzení exponenciální časovou složitost $\mathcal{O}\left(\exp\left(cn^{\frac{1}{3}}(\log n)^{\frac{2}{3}}\right)\right)$, kde c je konstanta a n představuje počet bitů potřebných k popisu periody r, zatímco u Shorova algoritmu je časová složitost polynomiální $\mathcal{O}(n^2 \log n(\log\log n))$, kde n je počet bitů faktorizovaného čísla [31].



Obrázek 7.2 Graf periodické funkce.

7.1.2 Od faktorizace k hledání periody

Faktorizace je matematický proces, při kterém rozkládáme číslo nebo výraz na součin menších čísel nebo výrazů, které se nazývají faktory. Faktorizace se často používá v algebře pro zjednodušení výrazů nebo pro řešení rovnic. Například číslo 100 můžeme rozložit na součin 10×10. Omezíme li se pouze na prvočíselný rozklad, pak 100 rozložíme na $2 \times 2 \times 5 \times 5$.

Peter Shor staví na poznatku z teorie čísel, že nalezení periody u modulárního umocňování může být využito k urychlení řešení ještě složitějších matematických problémů, mezi které patří již zmíněná faktorizace, ale například i řešení diskrétních logaritmů.

Zobecníme-li funkci z příkladu 7.1 dostaneme následující tvar

$$f(x) := a^x \mod n, \tag{7.1}$$

kde $a \in \{1, 2, 3, \cdots, n-1\}$ je nesoudělné s N. Protože každá r-tá perioda $f(x)$ je stejná, pak platí

$$\begin{aligned} a^r &\equiv 1 \mod N \\ a^r = (a^{r/2})^2 &\equiv 1 \mod N \\ (a^{r/2})^2 - 1 &\equiv 0 \mod N, \end{aligned} \tag{7.2}$$

pokud je r sudé číslo, pak

$$(a^{r/2} + 1)(a^{r/2} - 1) \equiv 0 \mod n. \tag{7.3}$$

Z rovnice vyplývá, že výraz $(a^{r/2}+1)(a^{r/2}-1)$ je násobkem čísla N a pokud $(a^{r/2}+1)$, nebo $(a^{r/2} - 1)$ není násobkem N, pak platí, že alespoň jedna část výrazu $(a^{r/2} + 1)$, nebo $(a^{r/2} - 1)$ musí mít společný netriviální faktor[2)] s číslem N.

Faktor N obdržíme z výpočtu NSD$(a^{r/2} + 1)$, nebo NSD$(a^{r/2} - 1)$. Funkce NSD (největší společný dělitel) může být vypočítána pomocí Eukleidova algoritmu s polynomiální časovou složitostí.

7.2 Kvantová Fourierova transformace

Kvantová Fourierova transformace, známá pod označením QFT (z angl. Quantum Fourier Transform) je kvantovou verzí Diskrétní Fourierovy transformace a lze jednoduše popsat jako změnu báze kvantového registru. Operátor QFT transformuje standardní bázi na Fourierovu (duální) bázi $|x\rangle \xrightarrow{QFT} |\tilde{x}\rangle$. [36]

QFT lze popsat rovnicí[3)]

[2)] Netriviální faktor čísla n je takové celé číslo d, pro které platí $1 < d < n$ a $n \mod d = 0$.

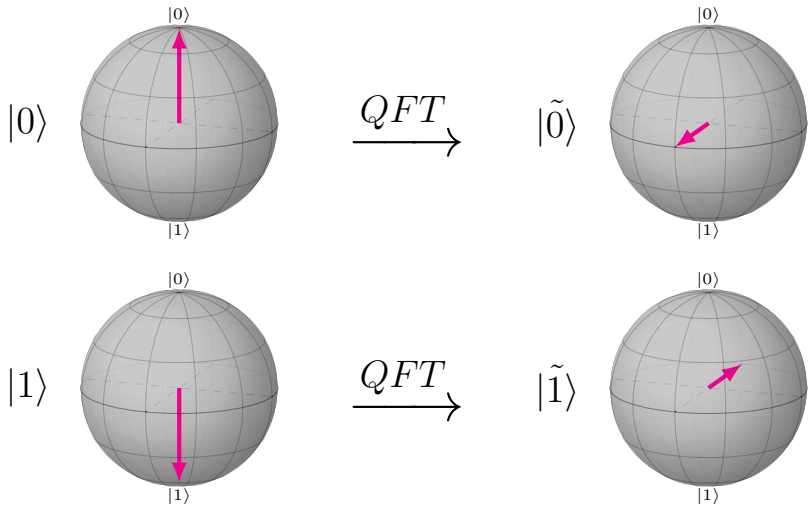
[3)] Exponenciální funkce e^x zapisujeme $\exp(x)$, pokud je exponent ve tvaru lomeného výrazu.

$$QFT\,|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \exp\left(\frac{2\pi i x y}{N}\right) |y\rangle\,, \tag{7.4}$$

kde $N = 2^n$ a n vyjadřuje počet qubitů v registru (viz tab. 7.1).[27]

Tabulka 7.1 Kvantový registr pro n-qubitů

Velikost registru	Výpočetní báze (základní stavy)
1 qubit	registr má dvě standardní báze $\{|0\rangle, |1\rangle\}$
2 qubity	registr má čtyři standardní báze $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$
⋮	⋮
n qubitů	počet standardních bází registru $N = 2^n$



Obrázek 7.3 Aplikace metody QFT na 1 qubit.

Nejprve si ukážeme, jak operátor QFT pracuje s kvantovým registrem o velikosti 1 qubitu, kde $n = 1$. Standardní báze[4)] registru je $\{|0\rangle, |1\rangle\}$, a Fourierova báze vyjadřuje plusový a minusový stav $\{|+\rangle, |-\rangle\}$. Změnu báze lze zobrazit pomocí jednotkového vektoru v Blochově sféře. Ve standardní bázi vždy vektor směřuje do pólu, zatímco po aplikaci QFT se vektor pohybuje v ekvatoreální rovině[5)] (viz obr. 7.3).

$$\begin{aligned} QFT\,|x\rangle &= \frac{1}{\sqrt{2}} \sum_{y=0}^{2-1} \exp\left(\frac{2\pi i x y}{2}\right) |y\rangle \\ &= \frac{1}{\sqrt{2}} \left[\exp\left(\frac{2\pi i x \cdot 0}{2}\right) |0\rangle + \exp\left(\frac{2\pi i x \cdot 1}{2}\right) |1\rangle\right] \\ &= \frac{1}{\sqrt{2}} \left[|0\rangle + e^{i\pi x} |1\rangle\right] \end{aligned}$$

[4)] Standardní bázové vektory reprezentují v kvantovém počítání log 0 a log 1.

[5)] Ekvatoreální rovina v Blochově sféře je rovina, která prochází středem Blochovy sféry a je kolmá na osu z (viz obr. 5.1)

Dosaďme za $x = 0$.

$$\begin{aligned} QFT\,|0\rangle &= \frac{1}{\sqrt{2}}\left[|0\rangle + e^{(i\pi x \cdot 0)}\,|1\rangle\right] \\ &= \frac{1}{\sqrt{2}}\left[|0\rangle + |1\rangle\right] \end{aligned}$$

Přesně takto je definován plusový stav $|+\rangle$.

$$QFT\,|0\rangle = \frac{1}{\sqrt{2}}\left[|0\rangle + |1\rangle\right] \equiv |+\rangle$$

Obdobně platí, že když dosadíme za $x = 1$ dostaneme minusový stav $|-\rangle$ ve Fourierově bázi, neboť dostáváme Eulerovu rovnost $(e^{i\pi} + 1) = 0$.

$$\begin{aligned} QFT\,|1\rangle &= \frac{1}{\sqrt{2}}\left[|0\rangle + e^{(i\pi x \cdot 1)}\,|1\rangle\right] \\ &= \frac{1}{\sqrt{2}}\left[|0\rangle - |1\rangle\right] \equiv |-\rangle \end{aligned}$$

Aplikace funkce QFT na jeden qubit má stejný efekt na změnu jeho stavu jako aplikace Hadamardova operátoru H (viz rov. 7.2). Zajímavá změna ve výsledku aplikace QFT funkce nastane, pokud aplikujeme QFT na registr sestávající z více qubitů (viz obr. 7.4).

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{7.5}$$

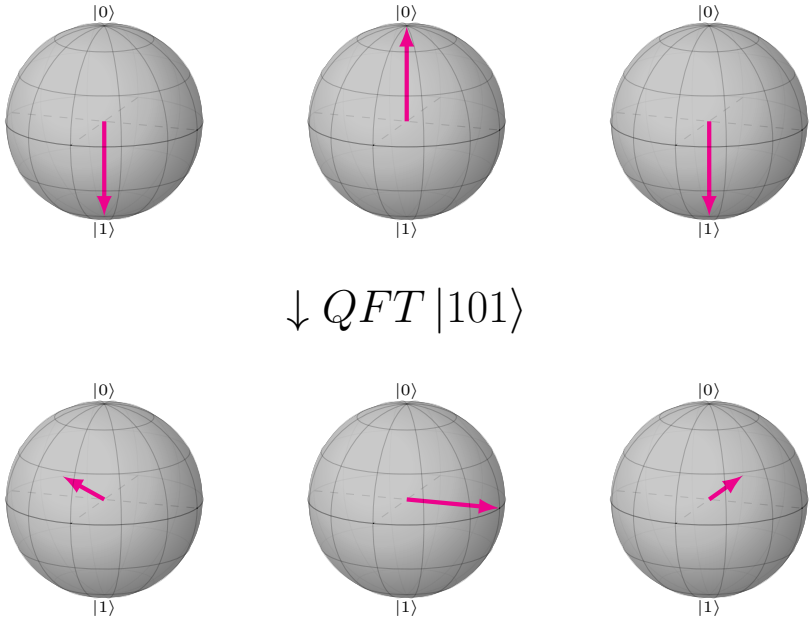
Pro větší registry, kde $N > 2$, je vhodné si rozepsat rovnici QFT do binárního tvaru $|x\rangle = |x_1, x_2, x_3, \ldots, x_n\rangle = |x_1\rangle \otimes |x_2\rangle \otimes |x_3\rangle \otimes \cdots \otimes |x_n\rangle$, kde x_1 je bit s největší vahou a značíme jej zkratkou MSB (z angl. Most Significant Bit).[36]

$$\begin{aligned} QFT_N\,|x\rangle = \frac{1}{\sqrt{N}}&\left[|0\rangle + \exp\left(\frac{2\pi i}{2}x\right)|1\rangle\right] \otimes \left[|0\rangle + \exp\left(\frac{2\pi i}{2^2}x\right)|1\rangle\right] \otimes \\ &\otimes \left[|0\rangle + \exp\left(\frac{2\pi i}{2^3}x\right)|1\rangle\right] \otimes \cdots \otimes \left[|0\rangle + \exp\left(\frac{2\pi i}{2^n}x\right)|1\rangle\right] \end{aligned} \tag{7.6}$$

Jak aplikace QFT změní báze v registru o třech qubitech se podíváme na následujícím příkladu.

Příklad 4.2 Aplikace QFT na registr o velikosti 3 qubity $|x\rangle = |101\rangle$. Lze použít i praktičtější dekadickou formu zápisu $|x\rangle = |5\rangle$.

$$QFT_8 |5\rangle = \frac{1}{\sqrt{8}} \left[|0\rangle + \exp\left(\frac{2\pi i}{2} 5\right) |1\rangle \right] \otimes \left[|0\rangle + \exp\left(\frac{2\pi i}{4} 5\right) |1\rangle \right] \otimes \left[|0\rangle + \exp\left(\frac{2\pi i}{8} 5\right) |1\rangle \right]$$



Obrázek 7.4 Aplikace metody QFT na registr $|101\rangle$.

Z obr. 7.4 je zřejmé, že vektor se v ekvatoreální rovině pohybuje různě pro každý qubit a to v závislosti na množství qubitů v registru. Qubit q_0 byl otočen o 5/8 plné otáčky, qubit q_1 o 10/8 plné otáčky (odpovídá 1/4 plné otáčky) a qubit q_2 o 20/8 plné otáčky (odpovídá 1/2 plné otáčky).

V diagramu QFT obvodu (obr. 7.5) je tato operace nad jednotlivými qubity znázorněna pomocí aplikace různých operátorů a na qubit x_n je aplikován pouze jediný Hadamardův operátor.

7.2.1 Protokol QFT obvodu

QFT obvod sestává ze dvou typů kvantových operátorů [36].

1. Hadamardův operátor H. Úkolem operátoru je změna standardní báze vektoru na maximálně smíšený stav $|+\rangle$ a $|-\rangle$.

$$H |x_k\rangle = \frac{1}{\sqrt{2}} \left[|0\rangle + \exp\left(\frac{2\pi i x_k}{2}\right) |1\rangle \right] \tag{7.7}$$

$$H |x_k\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle), (x_k = 0) \tag{7.8}$$

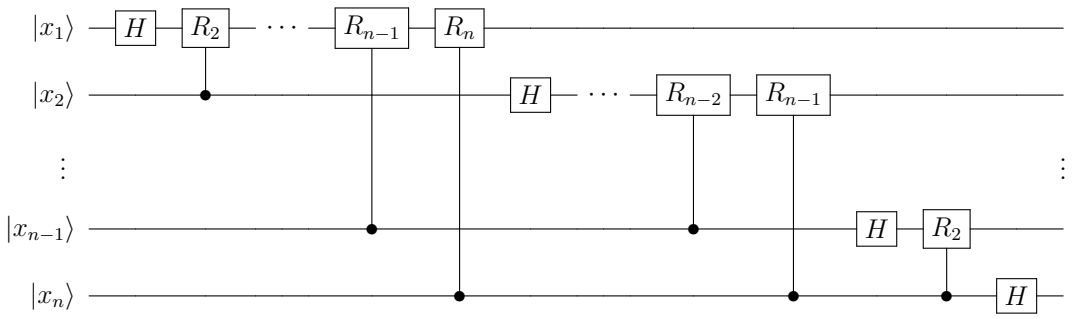
$$H\left|x_k\right\rangle = \frac{1}{\sqrt{2}}(\left|0\right\rangle - \left|1\right\rangle), (x_k = 1) \tag{7.9}$$

2. Rotace pomocí unitární matice R (z angl. Unitary Rotation) s parametrem k, který určuje velikost rotace vektoru.

$$R_k\left|x_j\right\rangle = \exp\left(\frac{2\pi i}{2^k}\right)\left|1\right\rangle \tag{7.10}$$

$$x_j = 0 \Rightarrow \exp\left(\frac{2\pi i x_j}{2^k}\right)\left|x_j\right\rangle = \left|0\right\rangle, \tag{7.11}$$

$$x_j = 1 \Rightarrow \exp\left(\frac{2\pi i x_j}{2^k}\right)\left|x_j\right\rangle = \exp\left(\frac{2\pi i}{2^k}\right)\left|1\right\rangle \tag{7.12}$$



Obrázek 7.5 Diagram QFT metody. Data čerpána z [36]

Pro schéma QFT obvodu na obr. 7.5 proveďme kontrolní výpočet. Operace QFT obvodu můžeme rozdělit do sekvence čtyř kroků [36].

Krok 0: Výchozí stav registru pro QFT obvod je $\left|x_1, x_2, x_3, \ldots, x_n\right\rangle$

Krok 1: Aplikace Hadamardova operátoru H na qubit 1

$$\frac{1}{\sqrt{2}}\left[\left|0\right\rangle + \exp\left(\frac{2\pi i}{2}x_1\right)\left|1\right\rangle\right] \otimes \left|x_2, x_3, \ldots, x_n\right\rangle \tag{7.13}$$

Krok 2: Aplikace operátoru R_2 na qubit 1, který je kontrolován qubitem 2

$$\frac{1}{\sqrt{2}}\left[\left|0\right\rangle + \exp\left(\frac{2\pi i}{2^2}x_2 + \frac{2\pi i}{2}x_1\right)\left|1\right\rangle\right] \otimes \left|x_2, x_3, \cdots, x_n\right\rangle \tag{7.14}$$

Krok 3: Aplikace operátoru R_n na qubit 1, který je kontrolován qubitem n

$$\frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2^n}x_n + \frac{2\pi i}{2^{n-1}}x_{n-1} + \cdots + \frac{2\pi i}{2^2}x_2 + \frac{2\pi i}{2}x_1\right)|1\rangle\right] \otimes |x_2, x_3, \ldots, x_n\rangle \tag{7.15}$$

Neboť $x = 2^{n-1}x_1 + 2^{n-2}x_2 + \cdots + 2^1x_{n-1} + 2^0x_n$ výraz lze přepsat na

$$\frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2^n}x\right)|1\rangle\right] \otimes |x_2, x_3, \ldots, x_n\rangle \tag{7.16}$$

Krok 4: Aplikace R_2 až R_n na qubity $2 \cdots n$ ve stejném pořadí kroků 1 až 3.

$$\begin{aligned}&\frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2^n}x\right)|1\rangle\right] \otimes \frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2^{n-1}}x\right)|1\rangle\right] \otimes \cdots \\ &\cdots \otimes \frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2^2}x\right)|1\rangle\right] \otimes \frac{1}{\sqrt{2}}\left[|0\rangle + \exp\left(\frac{2\pi i}{2^1}x\right)|1\rangle\right]\end{aligned} \tag{7.17}$$

Matematicky je funkce popsána správně, nicméně po sestavení algoritmu QFT obvodu dle schématu (obr. 7.5) dostaneme výstup qubitů v opačném pořadí a bude třeba jej následně v algoritmu otočit do stejného pořadí qubitů při vstupu.

7.2.2 Implementace QFT obvodu

Funkci QFT máme matematicky popsanou a nyní se můžeme podívat na zápis QFT algoritmu pomocí programovacího jazyka Python a frameworku Qiskit.

Nejprve inicializujeme framework Qiskit a ostatní potřebné knihovny. Zejména pro nás bude důležitá třída *QuantumCircuit* z knihovny Qiskit, která reprezentuje kvantový obvod a umožňuje aplikaci kvantových operátorů na jednotlivé qubity.

```
1 from qiskit import QuantumCircuit #knihovna pro vytváření kvantových obvodů
2 import numpy as np  # knihovna pro matematické operace s daty
3 pi = np.pi # definuje hodnotu čísla pi
```

Definujme funkci *qft_rotation*, která aplikuje rotaci fáze na kvantový obvod *circuit* se vstupem *n_qubits*, jenž udává počet qubitů v obvodu.

```
4 def qft_rotation(circuit,n_qubits):
5     """funkce převede vstupní obvod do Fourierovy báze"""
6
7     for qubit in range(n_qubits):
```

```
        circuit.h(qubit) # Hadamardův operátor
        for other_qubit in range(qubit+1, n_qubits):
            #  Controlled U operátor
            circuit.cp( pi / (2**(other_qubit - qubit)) , other_qubit, qubit)
    return circuit
```

Funkce je vytvořena pomocí smyčky, která aplikuje Hadamardův operátor a operátory $CROT_k$ postupně na každý qubit v obvodu dle zadaného postupu. $CROT$ (Controlled-R) je operátor rotace pomocí unitární matice s kontrolním qubitem. Matice operátoru $CROT_k$ je rozšířena o vstupní kontrolní qubit a vypadá následovně:

$$CROT_k = \begin{pmatrix} I & 0 \\ 1 & R_k \end{pmatrix}, \tag{7.18}$$

kde I je jednotková matice typu 2×2 a R_k je tato matice:

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & \exp\left(\frac{2\pi i}{2^k}\right) \end{pmatrix}. \tag{7.19}$$

Operátor $CROT_k$ je implementován do Qiskit pod názvem CP (z angl. Controlled Phase Gate) a lze jej zapsat pomocí následující matice:

$$CP(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{\theta i} \end{pmatrix}, \tag{7.20}$$

kde $\theta = 2\pi/2^k = \pi/2^{k-1}$.

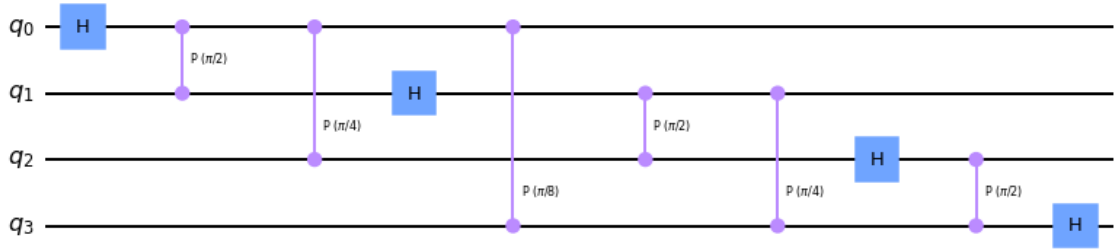
Pomocí metody *draw()* třídy *QuantumCircuit* vykreslíme kvantový obvod pro funkci *qft_rotation*. Můžeme zkontrolovat zda je obvod ve shodě s diagramem obr. 7.5.

```
circuit = QuantumCircuit(4) # kvantový obvod se 4 qubity
qft_rotation(circuit, 4).draw(output = 'mpl')
```

V našem příkladu zobrazujeme kvantový obvod *circuit* se čtyřmi qubity.

Funkci *qft_rotation* máme definovanou, nicméně nesmíme zapomenout, že obvod zpracuje qubity v opačném pořadí. Je to z důvodu snazší implementace kódu. Přidejme na výstup funkce *qft_rotation* další krátkou funkci *swap*, která nám transponuje qubity do požadovaného pořadí a výslednou funkci nazvěme *qft_swap*[6)].

[6)] Na funkci *qft_swap* jsou demonstrovány některé postupy kvantového programování a je nápsána, tak aby byla srozumitelná čtenářům bez pokročilé znalosti programování. Funkci *qft_swap* není ne-

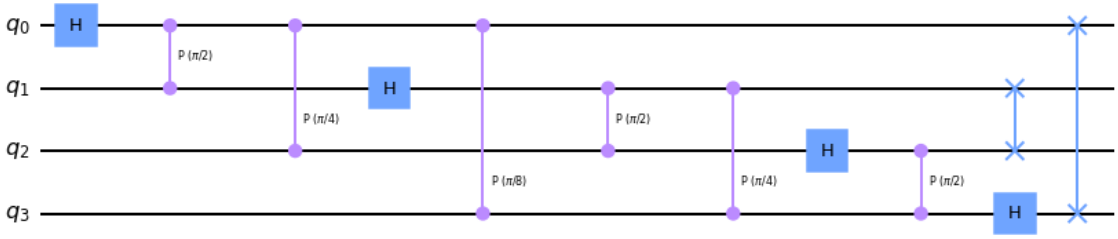


Obrázek 7.6 Výstup: obvod funkce *qft_rotation*

```
15  def swap(circuit, n_qubits):
16      for qubit in range(n_qubits//2):
17          circuit.swap(qubit, n_qubits-qubit-1)
18      return circuit
19
20  def qft_swap(circuit, n_qubits):
21      qft_rotation(circuit, n_qubits)
22      swap(circuit, n_qubits)
23      return circuit
```

Diagram kvantového obvodu *qft_swap* pro 4 qubity (viz obr. 7.7) znázorňuje *swap* operaci (modré čáry v obvodu), která mění pořadí qubitů na výstupu.

```
24  circuit = QuantumCircuit(4)
25  qft_swap(circuit, 4).draw(output = 'mpl')
```



Obrázek 7.7 Výstup: obvod funkce *qft_swap*

7.2.3 Konfigurace měření obvodu QFT

Ověřme správné fungování navrženého obvodu QFT (funkce *qft_swap*) na reálném kvantovém počítači, kterému lze zadat úlohu pomocí frameworku Qiskit.

zbytné programovat, neboť je v Qiskit již obsažena například ve třídě *QuantumCircuit*. Na rozdíl od naší funkce je metoda *qft([počet qubitů])* optimalizovaná, používá rekurzi (funkce volá samu sebe).

Nejprve náš QFT obvod musíme upravit. Kdybychom obvod spustili na skutečném kvantovém hardwaru v neupravené podobě, tak bychom na výstupu změřili pouze náhodné hodnoty, neboť všechny qubity jsou v superpozici $|0\rangle$ a $|1\rangle$. V experimentu provedeme inverzi funkce *qft_swap*, tak abychom mohli zadat vstupní stav $|\tilde{x}\rangle$. Jinými slovy inverzní obvod QFT^{-1} transformuje Fourierovu bázi na standardní bázi $|\tilde{x}\rangle \xrightarrow{QFT^{-1}} |x\rangle$.

```
26 def qft_inverse(circuit, n_qubits):
27
28     # Vytvoří QFT obvod o velikosti n qubitů:
29     qft_circuit = qft_swap(QuantumCircuit(n_qubits), n_qubits)
30     # provede inverzi obvodu
31     qft_circuit_inverse = qft_circuit.inverse()
32     # přidá inverzní QFT ke vstupnímu obvodu
33     circuit.append(qft_circuit_inverse, circuit.qubits[:n_qubits])
34     return circuit.decompose() # rozloží obvod na jednotlivé operátory
```

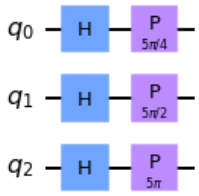
Třída *QuantumCircuit* obsahuje metodu *inverse()*, která umožňuje vytvořit inverzní obvod k zadanému kvantovému obvodu a pomocí metody *append()* jej přidá ke vstupnímu obvodu *circuit*. Pokud je obvod složitější a obsahuje například kontrolní operátory (angl. controlled gates), je třeba při vytváření inverzního obvodu použít tzv. dekompozici (rozklad) obvodu na jednodušší operátory. To lze provést v Qiskit pomocí metody *decompose()*.

Příklad 7.3 Vytvořme výchozí kvantový obvod *input_circuit* o velikosti 3 qubity a za $|\tilde{x}\rangle$ dosaďme $|\tilde{5}\rangle$, neboť pro tuto hodnotu již máme stav qubitů ve Fourierově bázi spočítán z příkladu 4.2.

```
35 n_qubits = 3 # velikost obvodu - 3 qubity
36 number = 5
37
38 # Vytvoří obvod s n qubity
39 input_circuit = QuantumCircuit(n_qubits)
40
41 for qubit in range(n_qubits):
42     input_circuit.h(qubit) # Hadamardův operátor
43
44 # nastavení fáze na qubitech
45 input_circuit.p(number*pi/4, 0) # rotace 1. qubitu
46 input_circuit.p(number*pi/2, 1) # rotace 2. qubitu
47 input_circuit.p(number*pi, 2) # rotace 3. qubitu
48
49 input_circuit.draw(output = 'mpl')
```

Na obrázku (obr. 7.8) je diagram kvantového obvodu $input\_circuit = |\tilde{5}\rangle$ se stavem qubitů ve Fourierově bázi z obrázku (obr. 7.4).

K nastavení stavu qubitu v obvodu *input_circuit* je použit parametrický fázový operátor, kde parametr θ může být libovolné reálné číslo, které určuje velikost relativní

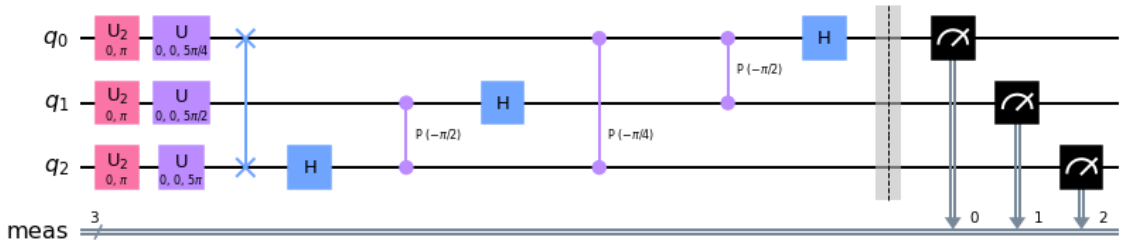


Obrázek 7.8
Výstup:
inicializace
vstupního obvodu
input_ circuit

fáze, která se má aplikovat na daný qubit. K tomu slouží v knihovně Qiskit metoda *p(θ, qubit)*.

Nyní stačí ke vstupnímu obvodu *input_circuit* přidat obvod QFT^{-1} a provést měření. Pro výsledný obvod $QFT_3^{-1}\,|\tilde{5}\rangle$ vykreslíme diagram (viz obr. 7.9).

```
50 circuit = qft_inverse(input_circuit, n_qubits)
51 circuit.measure_all() # Měří stav obvodu a zapíše hodnoty do klasického registru
52 circuit.draw(output = 'mpl')
```



Obrázek 7.9 Výstup: obvod $QFT_3^{-1}\,|\tilde{5}\rangle$

Metoda *measure_ all()* třídy *QuantumCircuit* slouží k měření stavů všech qubitů v kvantovém obvodu. Tato metoda přidává na konec kvantového obvodu instrukce pro měření všech qubitů v obvodu a zaznamená hodnoty 0 a 1 do registru v klasických bitech.

Měření qubitů může způsobit kolaps kvantového stavu do klasického stavu a také může ovlivnit stav zbývajících qubitů v obvodu. Proto je třeba měření provádět až po dokončení všech operací, které jsou potřeba provést v daném kvantovém obvodu.

7.2.4 Měření QFT na kvantovém počítači

Obvod $QFT^{-1}\,|\tilde{5}\rangle$ máme sestaven a připraven k provedení experimentu na kvantovém počítači.

K použití služby Qiskit Runtime, pomocí které spustíme vlastní kód v data centru IBM Quantum Lab, je třeba použít další knihovny Qiskit.

```
53 from qiskit import QuantumCircuit, transpile, assemble, Aer, IBMQ
54 from qiskit.providers.ibmq import least_busy
55 from qiskit.tools.monitor import job_monitor
56 from qiskit.visualization import plot_histogram
```

Pro přístup k reálnému kvantovému počítači je vyžadována registrace uživatelského účtu. K práci lze využít webovou aplikaci IBM Quantum Lab, nebo přístup přes API s uživatelským klíčem.

```
57 # Aktivuje IMBQ účet na nejméně vytížené instanci s potřebným množstvím qubitů
58 IBMQ.load_account() # API klíč
59 provider = IBMQ.get_provider(hub='ibm-q')
60 backend = least_busy(provider.backends(filters=lambda x: x.configuration().n_qubits
   ↪  >= nqubits
61                       and not x.configuration().simulator
62                       and x.status().operational==True))
63 print("least busy backend: ", backend)
```

```
Out[1]:     least busy backend:  ibmq_lima
```

Tento kód slouží k výběru nejméně zaneprázdněného fyzického kvantového zařízení, které je dostupné k výpočtům na IBMQ platformě a postupně dělá následující kroky:

1. Načte IBM Quantum Experience účet pomocí funkce *load_ account()*. Tato funkce umožňuje načtení API klíčů, které jsou potřebné k přístupu ke kvantovým počítačům v cloudu.

2. Pomocí funkce *get_ provider()* se připojí k poskytovateli, který je k dispozici v IBM Quantum Experience cloudu a předá mu identifikátor "hub", který identifikuje skupinu kvantových zařízení a služeb, které máte povolené na vašem účtu.

3. Poté se použije funkce *least_ busy()*. Tato funkce najde a vrátí nejméně zaneprázdněné kvantové zařízení (backend), které je k dispozici v rámci poskytovatele, a které splňuje několik podmínek. Tyto podmínky jsou definovány pomocí filtrů a zahrnují: minimální počet qubitů, operační stav zařízení a jeho nepoužití jako simulátor.

4. Na závěr se vypíše informace o nalezeném nejméně zaneprázdněném zařízení pomocí funkce *print()*[7].

Jaký očekáváme výstup experimentu?

Po změření kvantového obvodu dostaneme nějaké binární číslo v rozsahu 000 až 111, které bude odpovídat výpočetnímu stavu jednotlivých qubitů. Kvantové počítání je o pravděpodobnosti, s jakou obdržíme daný výsledek. U našeho obvodu s funkcí $QFT_3^{-1}\,|\tilde{5}\rangle = |5\rangle$ očekáváme výstup s nejvyšší pravděpodobností 101, což je binární zápis čísla 5.

Experiment zopakujeme vícekrát a zaznamenané výsledky vyneseme do histogramu.

```
64 shots = 2048 # počet opakování
65 transpiled_circuit = transpile(circuit, backend, optimization_level=3)
66 job = backend.run(transpiled_circuit, shots=shots)
67 job_monitor(job)
```

```
Out[1]:     Job Status: job has successfully run
```

Kód spustí výpočet na kvantovém počítači a postupně dělá následující kroky:

1. Vytvoří proměnnou *shots* s hodnotou 2048, která nastavuje počet opakování v rámci experimentálního výpočtu.

2. Pomocí funkce *transpile()* se připraví kvantový obvod *circuit* na spuštění na fyzickém kvantovém počítači (backend). Transpilace [8]umožňuje optimalizovat a připravit kvantový obvod pro konkrétní fyzické zařízení.

3. Spuštění výpočtu se provede pomocí funkce *backend.run()*. Tato funkce přijímá transpilovaný kvantový obvod *transpiled_ circuit*.

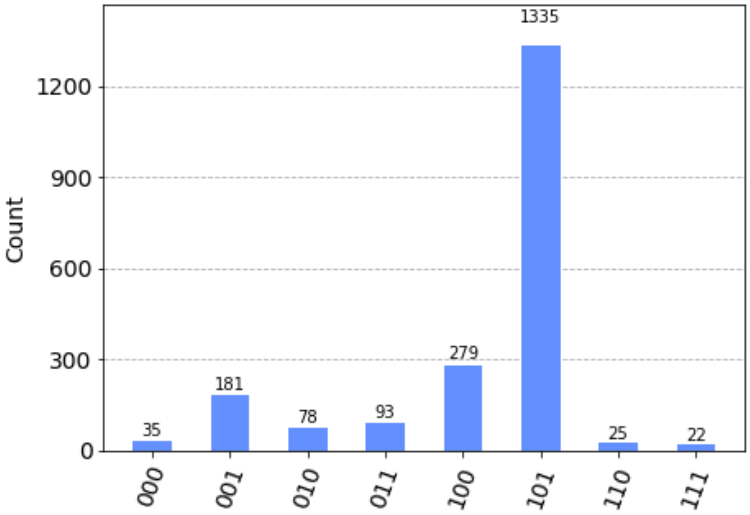
4. Funkce *job_ monitor()* z knihovny *qiskit.tools.monitor* umožňuje sledovat stav běhu výpočtu v reálném čase. Funkce ve formě textového výstupu zobrazuje počet úloh v frontě, přibližný čas dokončení úlohy a další informace.

Po dokončení výpočtu mohou být získány výsledky zpět pomocí objektu *job*. Jakmile obdržíme zprávu o dokončení výpočtu, můžeme zobrazit výsledky například pomocí histogramu.

[7]Náš experimentální výpočet proběhne na kvantovém zařízení *ibmq_ lima*. Tento kvantový počítač patří mezi nejméně výkoné počítače na platformě IBMQ, nicméně na naší úlohu s kvantovým obvodem o velikosti 3 qubity plně dostačuje, neboť počítač disponuje procesorem Falcon 4rT s 5 qubity.

[8]Rozdíl mezi tránspilátorem a kompilátorem je, že transpilátor překládá kód z jednoho programovacího jazyka na jiný jazyk s přibližně stejnou abstrakcí, zatímco kompilátor překládá kód vysokoúrovňového jazyka na nízkoúrovňový programovací jazyk.

```
counts = job.result().get_counts()
plot_histogram(counts)
```



Obrázek 7.10 Výstup: Naměřené hodnoty

Výsledek experimentu s kvantovou Fourierovou transformací je v souladu s předpokladem, neboť z naměřených hodnot vyplývá, že výstup s nejvyšší pravděpodobností je 101 (viz obr. 7.10).

7.3 Kvantový algoritmus odhadu fáze

Kvantový algoritmus odhadu fáze QPE (z angl. Quantum phase estimation) je používán k odhadu fáze (úhlu) vlastních čísel unitárních kvantových operátorů (matic).

QPE vychází z rovnice $Au = \lambda u$, kde A je $n \times n$ unitární matice, λ je vlastním číslem a u je vlastní vektor této matice [28].

Vstupy algoritmu jsou unitární operátor (matice) U a vlastní stav $|\psi\rangle$.

$$U|\psi\rangle = e^{2\pi i\theta}|\psi\rangle \tag{7.21}$$

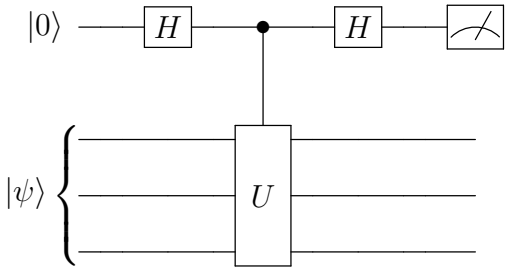
Stav $|\psi\rangle$ je vlastní vektor unitárního operátoru U s vlastním číslem $e^{2\pi i\theta}$ a θ je neznámá fáze [36].

Dalším vstupem QPE je případně počet qubitů v obvodu.

Jak extrahovat fázi θ z vlastní hodnoty operátoru U si ukážeme nejprve na algoritmu, který na odhad fáze využívá pouze jeden qubit.

7.3.1 Princip QPE obvodu

Uvažujme experimentální QPE obvod (obr. 7.11) s jedním qubitem pro odhad fáze a kvantovým registrem se vstupním stavem $|\psi\rangle$. Registr se vstupním vlastním stavem může sestávat z jednoho, nebo více qubitů v závislosti na řešeném problému, nicméně musí být dostatečně velký, aby mohl uchovávat informaci o vlastním stavu $|\psi\rangle$, který je zkoumán v rámci QPE.



Obrázek 7.11 QPE pro 1 qubit. Data čerpána z [36]

Samotný algoritmus QPE lze rozdělit do několika kroků:

Krok 0: Na vstupu algoritmu jsou dva kvantové registry, kde první (cílový) z nich obsahuje vstupní stav $|\psi\rangle$, který chceme ohodnotit. Druhý registr (kontrolní) obsahuje pomocný stav $|0\rangle$, který bude sloužit pro získání informace o fázi θ.

Celkový výchozí stav obou registrů je tedy dán jako $|\psi\rangle \otimes |0\rangle$.

Krok 1: Na kontrolní registr je aplikován Hadamardův operátor, který vytvoří superpozici ze stavů $|0\rangle$ a $|1\rangle$. Takto připravený registr je schopen převzít informaci o fázi z cílového registru a dále s informací pracovat.

Celkový stav systému je roven:

$$\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)|\psi\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle\,|\psi\rangle + |1\rangle\,|\psi\rangle\right) \tag{7.22}$$

Krok 2: Následuje aplikace unitárního operátoru U, který kontroluje kontrolní registr připravený v předchozím kroku a pomocí tzv. unitárních zdrojů fáze (angl. unitary phase kickback). Cílem je přenést informaci o fázi pomocí unitárních zdrojů fáze (angl. unitary phase kickback) z jednoho registru na druhý, a to bez toho, abychom museli explicitně znát hodnotu této fáze.

Výsledná matice U operátoru pro jeden qubit má následující tvar:

$$U = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i\theta} \end{pmatrix} \tag{7.23}$$

Na qubit ve stavu $|0\rangle$ nemá aplikace U žádný efekt a na stav $|1\rangle$ aplikuje fázi $e^{2\pi i\theta}$. Po aplikaci U se obvod nachází ve stavu:

$$\frac{1}{\sqrt{2}}\left(|0\rangle\,|\psi\rangle + |1\rangle\, e^{2\pi i\theta}\,|\psi\rangle\right) \tag{7.24}$$

Krok 3: Na kontrolní registr aplikujeme inverzní Fourierovu transformaci[9)], která obvod převede do standardní báze, a tak umožní získat informaci o fázi stavu $|\psi\rangle$.

$$\begin{aligned} &\frac{1}{\sqrt{2}}\left[\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)|\psi\rangle + e^{2\pi i\theta}\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)|\psi\rangle\right] = \\ &= \frac{1}{2}\left[|0\rangle\left(1 + e^{2\pi i\theta}\right) + |1\rangle\left(1 - e^{2\pi i\theta}\right)\right]|\psi\rangle \end{aligned} \tag{7.25}$$

Krok 4: Nakonec je provedeno měření kontrolního registru s výstupem $|0\rangle$, nebo $|1\rangle$. Z pravděpodobnosti daného výstupu můžeme pak vypočítat fázi z vlastní hodnoty matice, která je spojena se stavem $|\psi\rangle$.

Pravděpodobnost s jakou změříme stav $|0\rangle$, nebo $|1\rangle$ je nám známa.

$$P(|0\rangle) = \left|\frac{1}{2}\left(1 + e^{2\pi i\theta}\right)\right|^2 \tag{7.26}$$

$$P(|1\rangle) = \left|\frac{1}{2}\left(1 - e^{2\pi i\theta}\right)\right|^2 \tag{7.27}$$

Například, dosadíme-li fázi:

a) $\theta = 1^\circ$ bude $P(|0\rangle), P(|1\rangle) \doteq \{0,9969; 0,0031\}$

b) $\theta = 10^\circ$ bude $P(|0\rangle), P(|1\rangle) \doteq \{0,7283; 0,2717\}$

Z dostatečného počtu opakování měření jsme schopni z naměřených výsledků odhadnout fázi. Můžeme si povšimnout malé diference mezi jednotlivými pravděpodobnostmi

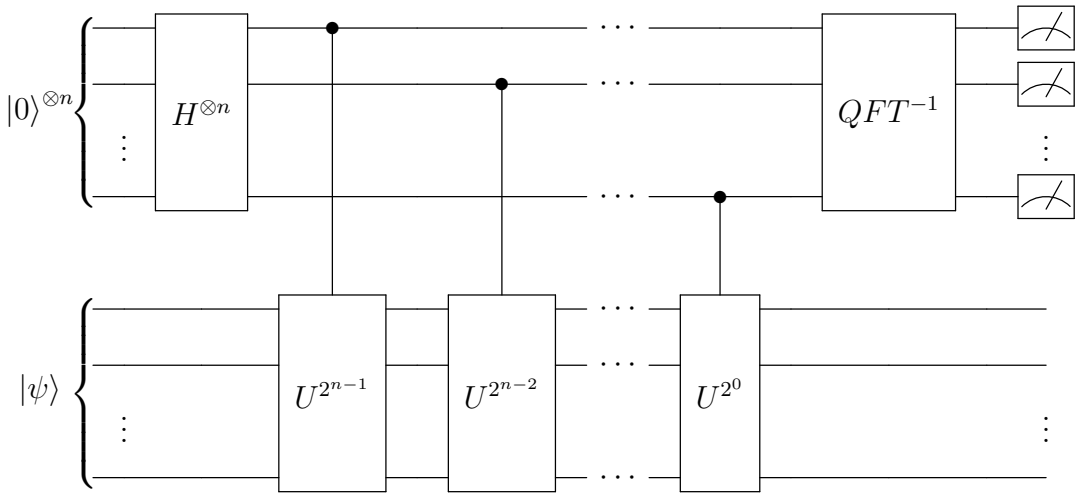
[9)] Nezapomeňme, že obvod QFT^{-1} s počtem qubitů $n = 1$ je tvořen pomocí jediného Hadamardova operátoru.

pro různé fáze θ. V tomto příkladu bychom potřebovali více než milion opakování pro zpětné dopočítání fáze.

Na tomto příkladu jsme si ukázali, že je možné odhadnout fázi unitární matice pomocí jediného qubitu, nicméně jedná se o velice náročný proces. Z toho důvodu se v obvodu QPE provádí měření na více qubitech

7.3.2 Protokol QPE obvodu

Řada kvantových algoritmů využívá obvody QPE s větším počet qubitů v kontrolním registru a v rámci Shorova algoritmu pro faktorizaci QPE obvod tvoří kvantovou část algoritmu s počtem qubitů závislým na velikosti faktorovaného čísla.



Obrázek 7.12 QPE pro n qubitů. Data čerpána z [36]

Z obvodu QPE s kontrolním registrem s více qubity (obr.7.12) bude zřejmé proč jsme si vysvětlovali princip kvantové Fourierovy transformace (QFT). S použitím více kontrolních qubitů zvětšíme diferenci v pravděpodobnosti změřeného výstupu $|0\rangle$, nebo $|1\rangle$ a tím zpřesníme odhad fáze i s mnohem menším počtem opakování měření obvodu.

Popíšeme si kroky v QPE obvodu s větším počtem kontrolních qubitů.

Příklad 7.5 QPE s kontrolním registrem $n > 1$

Krok 0: Výchozí stav obvodu je $|0\rangle^{\otimes n} |\psi\rangle$

Krok 1: Uvedení kontrolního registru do superpozice:

$$\left(\frac{1}{\sqrt{2}}\right)^n (|0\rangle + |1\rangle)^{\otimes n} |\psi\rangle \tag{7.28}$$

Krok 2: Aplikace operátoru U:

$$\begin{aligned}\left(\frac{1}{\sqrt{2}}\right)^n \left(|0\rangle + e^{2\pi i\theta 2^{n-1}} |1\rangle\right) \otimes \cdots \\ \cdots \otimes \left(|0\rangle + e^{2\pi i\theta 2^{1}} |1\rangle\right) \otimes \left(|0\rangle + e^{2\pi i\theta 2^{0}} |1\rangle\right)\end{aligned} \tag{7.29}$$

Porovnáme-li tento stav obvodu QPE se stavem obvodu QFT (rov. 7.14) zjistíme, že jsou si velmi podobné a dosadíme-li za x do QFT obvodu $2^n\theta$, pak budou výrazy shodné. Z toho vyplývá, pokud aplikujeme inverzní QFT obvod na kontrolní registr, tak dostaneme právě přibližnou hodnotu $2^n\theta$ jako výstup QPE obvodu.

Krok 3: Měření: Pravděpodobnost výstupu $|0\rangle$, nebo $|1\rangle$ je amplifikována počtem qubitů v kontrolním registru. Například bude-li počet qubit $n = 10$, pak se přesnost odhadu fáze znásobí $2^{10} = 1024$. To je velice podstatné z hlediska hardwaru kvantových počítačů, kde je třeba připočítat kvantovou chybu v měření způsobenou šumem.

7.3.3 Implementace QPE obvodu

Algoritmus QPE vytvoříme ve frameworku Qiskit a v programovacím jazyku Python. Některé části kódu a funkce Qiskit jsou již vysvětleny v předchozí kapitole, která se věnuje implementaci algoritmu QFT.

Na začátek kódu importujeme potřebné knihovny pro tvorbu kvantových obvodů.

```
1  #knihovny pro matematické operace
2  import matplotlib.pyplot as plt
3  import numpy as np
4  pi = np.pi
5
6  # Qiskit
7  from qiskit import IBMQ, Aer, transpile, assemble
8  from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister
9
10 # Vizualizace výsledků
11 from qiskit.visualization import plot_histogram
```

Využijeme stejné knihovny, jako v algoritmu QFT, nicméně přidáme navíc velice důležitou knihovnu Aer.

Aer je balíček v frameworku Qiskit, který poskytuje řadu různých simulátorů, včetně jednoduchých simulačních zařízení (např. qasm_simulator) a simulátorů umožňujících simulovat vliv šumu (např. noise_model_simulator). Díky tomu je možné vyvíjet kvantové algoritmy a ověřovat jejich funkčnost před spuštěním na skutečných kvantových

počítačích. Aer v kombinaci s knihovnou transpile umožňuje také optimalizaci kvantových obvodů pro konkrétní hardware.

Obvod pro inverzní kvantovou Fourierovu transformaci z minulé kapitoly shrňme do funkce *qft_ inverse*.

```
12 def qft_inverse(circuit, n_qubits):
13     """inverzni QFT pro n qubitu"""
14 
15     # funkce swap
16     for qubit in range(n_qubits //2):
17         circuit.swap(qubit, n_qubits - qubit -1)
18 
19     # funkce qft_inverse
20     for j in range(n_qubits):
21         for m in range(j):
22             circuit.cp( -pi / float(2**(j-m)), m, j)
23         circuit.h(j)
```

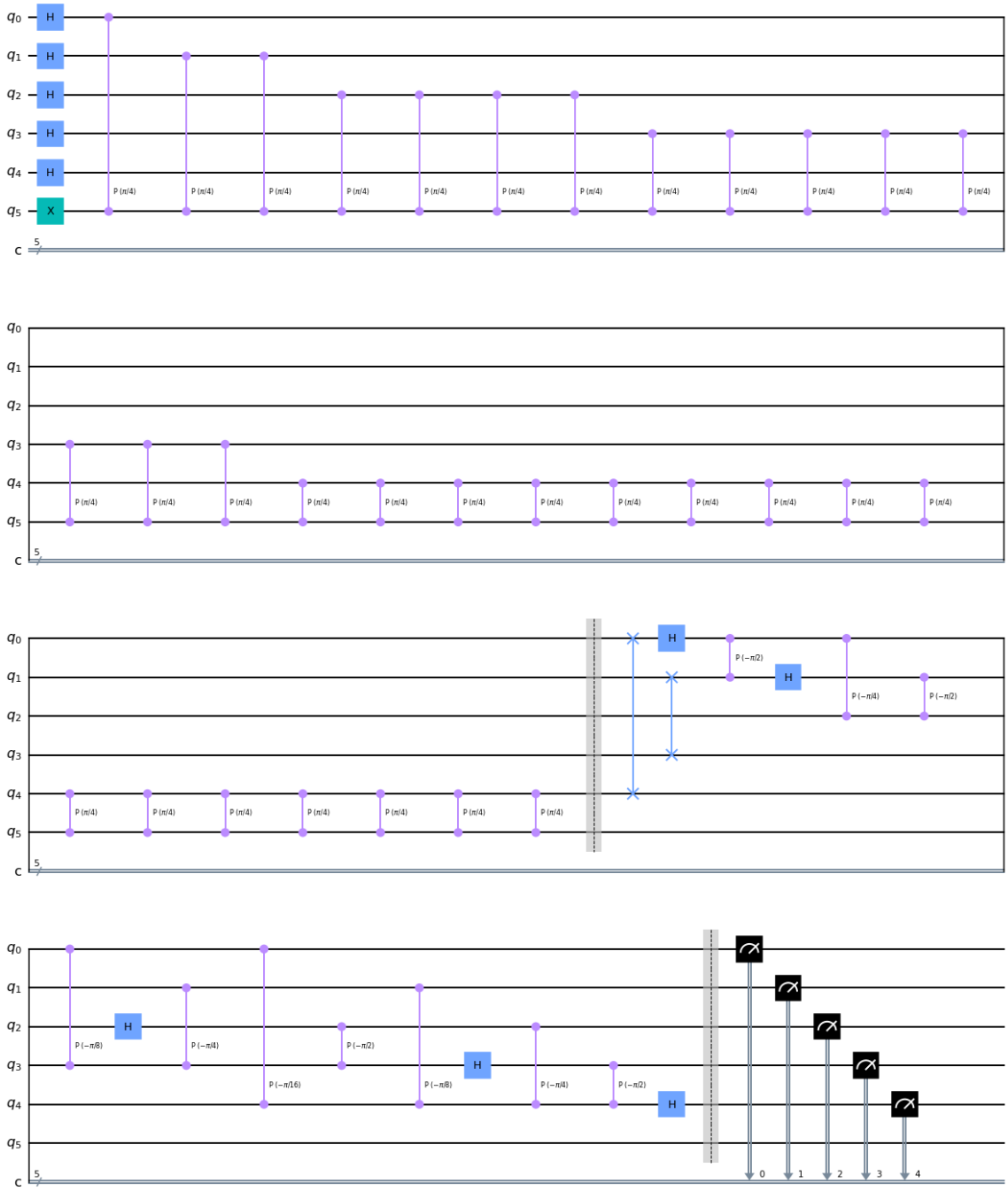
Dále zkonstruujeme obvod, ve kterém budeme chtít odhadnout fázi. Vezměme například unitární matici operátoru T, který mění fázi o $e^{i\pi/4}$ ve stavu $|1\rangle$

$$T|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = e^{i\pi/4}|1\rangle \tag{7.30}$$

Vyjádříme-li fázi θ z QPE v $T|1\rangle = e^{2\pi i\theta}|1\rangle$, pak lze očekávat výstup $\theta = \frac{1}{8}$.

```
24 # Vytvoření kvantového obvodu pro algoritmus QPE
25 qpe = QuantumCircuit(6, 5)
26 
27 # Příprava kontrolního stavu
28 for qubit in range(5):
29     qpe.h(qubit)
30 
31 # Příprava vlastního stavu |psi>
32 qpe.x(5)
33 
34 # Aplikace unitární matice a její opakované násobení
35 angle = pi/4
36 repetitions = 1
37 for counting_qubit in range(5):
38     for i in range(repetitions):
39         qpe.cp(angle, counting_qubit, 5);
40     repetitions *= 2
41 
42 qpe.barrier() # Rozdělovník diagramu
43 
44 # Inverzní QFT
45 qft_inverse(qpe, 5)
46 
```

```
47  # Měření
48  qpe.barrier()
49  for n in range(5):
50      qpe.measure(n,n)
51
52  qpe.draw() # vykreslení diagramu
```



Obrázek 7.13 Výstup: QPE diagram s 6 qubity

V kódu definujeme kvantový obvod s šesti qubity a pěti klasickými bity pro uložení výsledků měření (viz obr. 7.13). Poté se na prvních pěti kvantových bitech aplikuje

Hadamardův operátor, aby se vytvořil kontrolní stav. Operátor X na šestém qubitu připraví vlastní stav $|\psi\rangle$.

Následně se opakovaně aplikuje unitární matice U na prvních pět qubitů s exponenty, které se zvyšují s každou iterací a po aplikaci matice U následuje inverzní QFT, aby bylo možné měřit fázi θ. V posledním kroku je provedeno měření na kontrolních bitech a výsledky jsou uloženy do klasických registrů.

Nakonec se měří výsledky na prvních pěti kvantových bitech, z čehož odhadneme fázi vlastního stavu U v binární podobě, která se zapisuje pomocí pěti klasických bitů.

Pomocí funkce *Aer.get_backend('aer_simulator')* vytvoříme instanci *aer_simulator* jako backend pro Qiskit, na kterém spustíme QPE obvod s připraveným stavem $|\psi\rangle$.

```
# Spuštění simulace a získání výsledků
aer_sim = Aer.get_backend('aer_simulator')
shots = 2048 # Počet opakování
t_qpe = transpile(qpe, aer_sim)
qobj = assemble(t_qpe, shots=shots)
results = aer_sim.run(qobj).result()
answer = results.get_counts()

print(answer)
```

```
Out[1]:     {'00100': 2048}
```

Algoritmus provedl celkem 2048 opakování a ve všech případech byla změřena hodnota 00100, což je binární zápis čísla 4. Abychom získali θ musíme vydělit naměřenou hodnotu faktorem 2^n.

$$\theta = \frac{4}{2^n} = \frac{4}{2^5} = \frac{1}{8} \tag{7.31}$$

V simulovaném prostředí Aer je pravděpodobnost $P(4) = 1$. U reálného kvantového počítače je pravděpodobnost zpravidla nižší, což si ověříme na kvantovém počítači *imb_nairobi*[10)].

```
# Aktivuje IMBQ účet na nejméně vytížené instanci s potřebným množstvím qubitů
IBMQ.load_account()
from qiskit.tools.monitor import job_monitor
provider = IBMQ.get_provider(hub='ibm-q')
nairobi = provider.get_backend('ibm_nairobi')

```

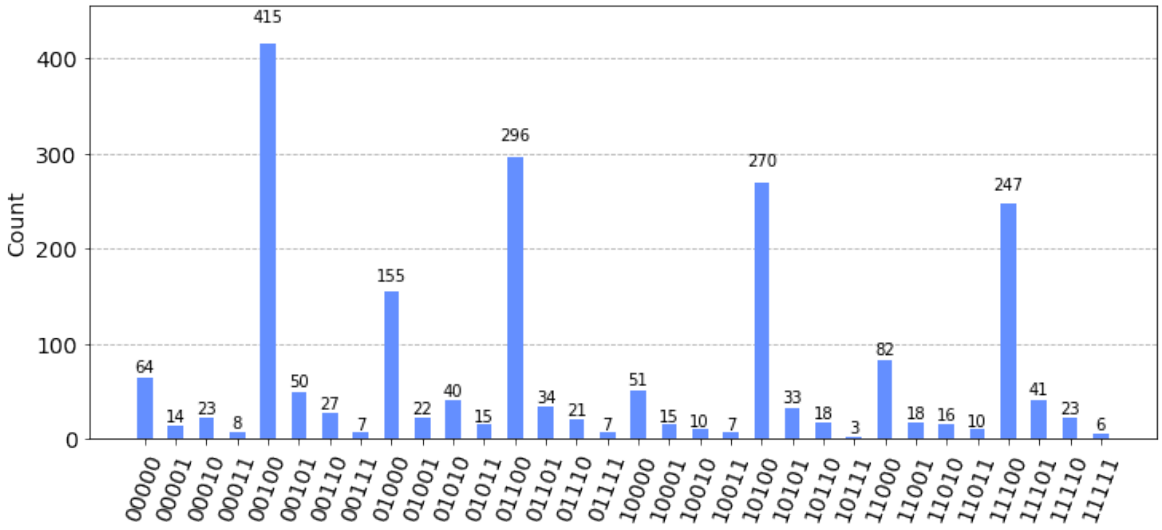
[10)] Kvantový počítač *imb_nairobi* je volně dostupný na platformě IBMQ Experience. Tento počítač je vybaven procesorem Falcon r5.11H se 7 qubity.

```
68 shots = 2048 # Počet opakování
69 t_qpe = transpile(qpe, nairobi, optimization_level=3)
70 job = nairobi.run(t_qpe, shots=shots)
71 job_monitor(job)
```

```
Out[1]:     Job Status: job has successfully run
```

Celkovou dobu zpracování úlohy na reálném kvantovém počítači lze rozdělit na část vlastního zpracování algoritmu a na část strávenou ve frontě všech úloh připravených pro dané zařízení. V době psaní této práce se pohybovala čekací doba v řádu jednotek hodin, zatímco samotné zpracování úlohy netrvalo déle než 30 sekund i s počtem opakování 4096.

```
72 # Zobrazí výsledky z QPE obvodu
73 results = job.result()
74 answer = results.get_counts(qpe)
75
76 plot_histogram(answer, figsize=(12, 5))
```



Obrázek 7.14 Výstup: Naměřené hodnoty QPE

Z výsledku měření obvodu QPE je patrné razantní snížení pravděpodobnosti $P(4) = 0,2$ oproti simulaci. Přestože se jedná výstup s nejvyšší pravděpodobností není výsledek s takto nízkou pravděpodobností uspokojivý. Pro zvýšení přesnosti odhadu fáze je možné navýšit počet kontrolních qubitů a nebo provést více opakování.

Co způsobuje chybu v odhadu fáze?

Měření v algoritmu QPE není přesné pro iracionální fáze, protože v tomto případě není možné identifikovat fázi s přesností na jediný bit. Zároveň se v kvantových systémech setkáváme s různými typy chyb, jako jsou chyby v kvantových hradlech, chyby

v inicializaci stavů, dekoherence kvantového stavu apod. Tyto chyby se mohou kumulovat a vést k nesprávným výsledkům. Proto je důležité při návrhu algoritmů zohlednit možné chyby a optimalizovat algoritmus pro co nejlepší výsledky za daných podmínek.

7.4 Protokol Shorova faktorizačního algoritmu

Ukažme si postup Shorova algoritmu na jednoduchém příkladu, kde budeme postupovat krok za krokem podle vývojového diagramu (obr. 7.1).

Příklad 7.6 Nalezení faktorů Pomocí Shorova algoritmu. Máme číslo $N = 15$ a proveďme rozklad na prvočísla p a q.

Krok 1: Zvolme číslo $a < N$, pro které zároveň platí, že a je nesoudělné s N.

Dosaďme: $a = 13$

Krok 2: Najděme periodu funkce $f(x) = a^x \mod N$.

Dosaďme: $13^x \mod 15$

Tabulka 7.2 Určení periody

r	0	1	2	3	4	5	6	7
$13^x \mod 15$	1	13	4	7	1	13	4	7

$13^4 \equiv 1 \mod 15$, perioda $r = 4$

Krok 3: Ověřme, že r je sudé číslo, pokud je r liché číslo, přejděme na krok 1 a zvolme jiné a.

Perioda $r = 4$ je sudé číslo.

Krok 4: Vypočtěme hodnotu $x = a^{r/2} \mod N$ a ověřme, že $x+1 \not\equiv 0 \mod N$. Pokud je $x + 1$ kongruentní s $0 \mod N$, pak přejděme na krok 1 a zvolme jiné a.

Dosaďme:

$$x = 13^{4/2} \equiv 4 \mod 15,$$
$$4 + 1 \not\equiv 0 \mod 15$$

V případě, že $x + 1$ není kongruentní s $0 \mod N$, pak alespoň jeden z prvočíselných faktorů N je roven $\mathrm{NSD}(x + 1, N)$ a/nebo $\mathrm{NSD}(x - 1, N)$.

Krok 5: Vypočtěme $\mathrm{NSD}(x + 1, N)$ a $\mathrm{NSD}(x - 1, N)$.

Dosaďme:

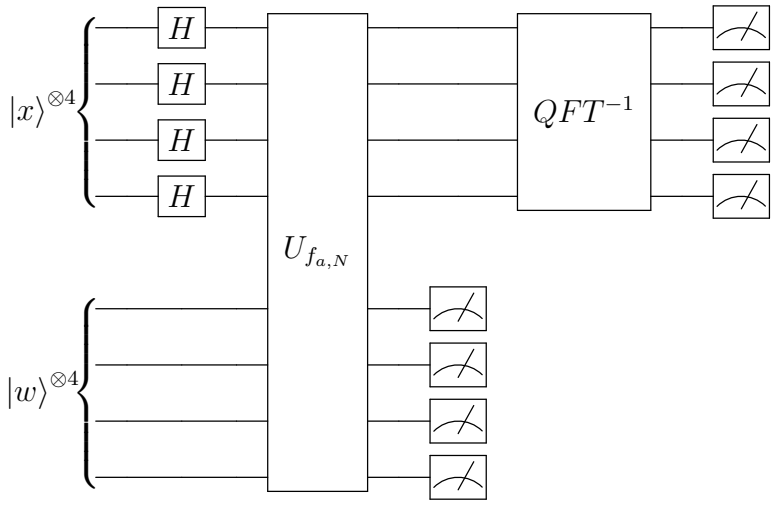
$$\left\{ \begin{matrix} \mathrm{NSD}[(4+1), 15)] \\ \mathrm{NSD}[(4-1), 15] \end{matrix} \right\} = \{5, 3\} = \{p, q\}$$

Obdrželi jsme výsledné faktory 5 a 3 čísla 15, což je v pořádku a nyní se pojďme podívat na 2. krok v algoritmu, jak zjistit periodu r. Tuto část má na starosti kvantový obvod QPE.

Příklad 7.7 Výpočet periody r pomocí obvodu QPE pro $N = 15$, $a = 13$.

Shorův algoritmus potřebuje obecně $2n$ qubitů, kde n je počet bitů potřebných k vyjádření čísla N.

Celkem budeme pro obvod QPE potřebovat 8 qubitů ($N = 15_{10} = 1111_2$) rozdělené mezi kontrolní registr se 4 qubity pro hledání periody a cílový registr se 4 qubity pro uchování čísla N ve stavu superpozice (viz obr. 7.15).



Obrázek 7.15 QPE čás Shorova algoritmu. Data čerpána z [36]

V Shorově QPE obvodu označme stav kontrolního registru $|x\rangle$ a stav cílového registru označme $|w\rangle$. Unitární operátor U aplikuje funkci $f_{a,N}(x) \equiv a^x \mod N$ na cílový registr $|x\rangle\,|w\rangle \to |x\rangle\,|w \oplus f_{a,N}(x)\rangle$.

Rozeberme si výpočet Shorova QPE obvodu po jednotlivých krocích.

Krok 0: Výchozí stav obvodu je $|0\rangle^{\otimes 4}\,|0\rangle^{\otimes 4}$ (registr $|x\rangle$ je zapsán vlevo a $|w\rangle$ vpravo).

Krok 1: Stav obvodu po aplikaci Hadamardova operátoru:

$$\left[H^{\otimes 4}\,|0\rangle^{\otimes 4}\right]|0\rangle^{\otimes 4} = \frac{1}{4}\left[|0\rangle_4 + |1\rangle_4 + |2\rangle_4 + \cdots + |15\rangle_4\right]|0\rangle^{\otimes 4} \tag{7.32}$$

Poznámka: Index 4 u ket notace zjednodušuje zápis binárních čísel a značí, že decimálně vyjádřená hodnota je číslo o čtyřech bitech. Například $|15\rangle_4 = |1111\rangle$.

Krok 2: Stav obvodu po aplikaci unitární funkce:

$$\frac{1}{4}\Big[\,|0\rangle_4\,|0\oplus 13^0 \mod 15\rangle_4 + |1\rangle_4\,|0\oplus 13^1 \mod 15\rangle_4 + |2\rangle_4\,|0\oplus 13^2 \mod 15\rangle_4 \cdots + |15\rangle_4\,|0\oplus 13^{15} \mod 15\rangle_4\Big] \tag{7.33}$$

Poznámka: Symbol $\oplus$ je *modulo 2 suma*, nebo-li XOR operace ($0 \oplus z = z$).

Výraz lze upravit:

$$\frac{1}{4}\begin{bmatrix} & |0\rangle_4\,|1\rangle_4 & + & |1\rangle_4\,|13\rangle_4 & + & |2\rangle_4\,|4\rangle_4 & + & |3\rangle_4\,|7\rangle_4 \\ + & |4\rangle_4\,|1\rangle_4 & + & |5\rangle_4\,|13\rangle_4 & + & |6\rangle_4\,|4\rangle_4 & + & |7\rangle_4\,|7\rangle_4 \\ + & |8\rangle_4\,|1\rangle_4 & + & |9\rangle_4\,|13\rangle_4 & + & |10\rangle_4\,|4\rangle_4 & + & |11\rangle_4\,|7\rangle_4 \\ + & |12\rangle_4\,|1\rangle_4 & + & |13\rangle_4\,|13\rangle_4 & + & |14\rangle_4\,|4\rangle_4 & + & |15\rangle_4\,|7\rangle_4 \end{bmatrix} \tag{7.34}$$

Krok 3: Změření kontrolního registru $|w\rangle$.

Z výrazu (rov. 7.34) je patrné, že výstupem registru $|w\rangle$ může být pouze jedna ze čtyřech hodnot se stejnou pravděpodobností $P(|1\rangle_4) = P(|13\rangle_4) = P(|4\rangle_4 = P(|7\rangle_4) = 1/4$.

Například změříme-li $|w\rangle = |7\rangle_4$ pak registr $|x\rangle = \frac{1}{2}\left[|3\rangle_4 + |7\rangle_4 + |11\rangle_4 + |15\rangle_4\right]$

Obvod bude ve stavu:

$$|x\rangle\,|w\rangle = \frac{1}{2}\left[|3\rangle_4 + |7\rangle_4 + |11\rangle_4 + |15\rangle_4\right] \otimes |7\rangle_4 \tag{7.35}$$

Poznámka: Všimněte si v změny normalizačním koeficientu z $\frac{1}{4}$ na $\frac{1}{2}$. Počet možných kombinací v obvodu klesl z 16 na 4. Také dlužno dodat, že měření v tomto kroku je pouze ilustrativní pro pochopení stavu obvodu. Ve skutečnosti proběhne pouze jedno měření kontrolního registru na konci celé operace, které si "dopočítá"informace z cílového registru.

Krok 4: Aplikace QFT^{-1} na registr $|x\rangle$.

Připomeňme si rovnice funkcí QFT a QFT^{-1} z kapitoly: Kvantová Fourierova transformace.

$$\boxed{\begin{aligned} QFT\,|x\rangle = |\tilde{x}\rangle &= \frac{1}{\sqrt{M}}\sum_{y=0}^{M-1}\exp\left(\frac{2\pi i}{M}xy\right)|y\rangle \\ QFT^{-1}\,|\tilde{x}\rangle = |x\rangle &= \frac{1}{\sqrt{M}}\sum_{y=0}^{M-1}\exp\left(\frac{-2\pi i}{M}xy\right)|y\rangle \end{aligned}} \tag{7.36}$$

Pro $|w\rangle_4 = |7\rangle_4$ odpovídají jednotlivé hodnoty v registru $|x\rangle$:

$$QFT^{-1}|3\rangle_4 = \frac{1}{16}\sum_{y=0}^{15}\exp\left(\frac{-2\pi i}{16}3y\right)|y\rangle$$

$$QFT^{-1}|7\rangle_4 = \frac{1}{16}\sum_{y=0}^{15}\exp\left(\frac{-2\pi i}{16}7y\right)|y\rangle$$

$$QFT^{-1}|11\rangle_4 = \frac{1}{16}\sum_{y=0}^{15}\exp\left(\frac{-2\pi i}{16}11y\right)|y\rangle$$

$$QFT^{-1}|15\rangle_4 = \frac{1}{16}\sum_{y=0}^{15}\exp\left(\frac{-2\pi i}{16}15y\right)|y\rangle$$

Poznámka: V QFT rovnici $M = 2^n$, kde n je počet qubitů v QFT obvodu.

Celkově je registr $|x\rangle$ ve stavu:

$$QFT^{-1}|x\rangle = \frac{1}{8}\sum_{y=0}^{15}\Bigg[\exp\left(-i\frac{3\pi}{8}y\right) + \exp\left(-i\frac{7\pi}{8}y\right) + \exp\left(-i\frac{11\pi}{8}y\right) + \exp\left(-i\frac{15\pi}{8}y\right)\Bigg]|y\rangle \tag{7.37}$$

Rozložme exponenciální části na jejich goniometrický tvar pomocí Eulerova vzorce $e^{ix} = \cos(x) + i\sin(x)$ a dosadíme do výrazu:

$$\begin{aligned} QFT^{-1}|x\rangle = \frac{1}{8}\sum_{y=0}^{15}\Bigg[&\left(\cos\left(-\frac{3\pi}{8}y\right) + i\sin\left(-\frac{3\pi}{8}y\right)\right) \\ &+ \left(\cos\left(-\frac{7\pi}{8}y\right) + i\sin\left(-\frac{7\pi}{8}y\right)\right) \\ &+ \left(\cos\left(-\frac{11\pi}{8}y\right) + i\sin\left(-\frac{11\pi}{8}y\right)\right) \\ &+ \left(\cos\left(-\frac{15\pi}{8}y\right) + i\sin\left(-\frac{15\pi}{8}y\right)\right)\Bigg]|y\rangle \end{aligned} \tag{7.38}$$

Nyní můžeme aplikovat sumaci na každý člen a získat výsledek.

Vzhledem k tomu, že tento výraz je poměrně složitý, tak pro výpočet jednotlivých hodnot raději použijeme následující skript. Výsledek bude ve tvaru superpozice kvantových stavů s komplexními amplitudami.

```
1 import numpy as np
2 pi = np.pi
```

```
3
4  for y in range(15):
5      coeff = np.exp(-1j * 3 * pi / 8 *y) + \
6              np.exp(-1j * 7 * pi / 8 *y) + \
7              np.exp(-1j * 11 * pi / 8 *y) + \
8              np.exp(-1j * 15 * pi / 8 *y)
9      # zaokrouhlení hodnot blížících se nule
10     real_part = 0 if abs(coeff.real) < 1e-10 else coeff.real
11     imag_part = 0 if abs(coeff.imag) < 1e-10 else coeff.imag
12     coeff_rounded = complex(real_part, imag_part)
13     print(f'|{y}> : {coeff_rounded}')
```

```
Out[1]:     |0> : (4+0j)
            |1> : 0j
            |2> : 0j
            |3> : 0j
            |4> : 4j
            |5> : 0j
            |6> : 0j
            |7> : 0j
            |8> : (-4+0j)
            |9> : 0j
            |10> : 0j
            |11> : 0j
            |12> : -4j
            |13> : 0j
            |14> : 0j
```

Vypočtené hodnoty dosadíme do rovnice:

$$QFT^{-1}\left|x\right\rangle = \frac{1}{8}\left[4\left|0\right\rangle_4 + 4i\left|4\right\rangle_4 - 4\left|8\right\rangle_4 - 4i\left|12\right\rangle_4\right] \tag{7.39}$$

Zde jsme se setkali s klíčovým konceptem v kvantové mechanice. V řadě výsledků získáme nulové amplitudy pro některé stavy $|y\rangle$. Tento jev je důsledkem destruktivní interference[11)] mezi složkami sumy. Významně nenulové amplitudy naznačují hodnoty y, které nám poskytují informace o periodě hledané funkce.

Krok 5: Měření na výstupu Shorova QPE obvodu:

Významně nenulové amplitudy se objevují pro stavy, které mají hodnotu $y = \{0, 4, 8, 12\}$. Změřením registru $|x\rangle$ s přibližně stejnou pravděpodobností obdržíme jednu z hodnot y. Tyto hodnoty představují vrcholy v pravděpodobnostním rozložení a poskytují informace o hledané periodě.

[11)] Interference je jev, který se vyskytuje, když se dvě nebo více vln překrývají v čase a prostoru. Při interferenci se vlastnosti vln, jako je amplituda, kombinují, což může vést k posílení (konstruktivní interference) nebo oslabení (destruktivní interference) výsledné vlny.

Po provedení měření, očekáváme, že vrcholy pravděpodobnosti nalezení periody jsou blízká hodnotám $j\frac{M}{r}$, kde $j \in \mathbb{Z}$ a je násobkem základní frekvence a $M = 2^n$,kde n je počet qubitů v QFT obvodu.

Toto umístění vrcholů je založeno na faktu, že QFT rozpoznává periodicitu v kvantovém stavu a převádí ji na diskrétní frekvence, které jsou násobky základní frekvence.

Shrnutí: Výstupem kvantové části algoritmu pro $N = 15$, $a = 13$ je hodnota 0, 4, 8, nebo 12. Zpracování změřené hodnoty provádí klasická část Shorova algoritmu.

Proveďme analýzu každého jednotlivého výstupu po změření $|x\rangle$ registru.

1. Změřená hodnota registru $|x\rangle = |4\rangle_4$:

$$j\frac{16}{r} = 4 \implies j = 1, r = 4$$

kontrola : $r = 4$ je sudé číslo

$$x \equiv a^{r/2} \mod 15 \equiv 13^{4/2} \mod 15 \equiv 4$$
$$\text{NSD}(x+1, N) = \text{NSD}(4+1, 15) = 5$$
$$\text{NSD}(x-1, N) = \text{NSD}(4-1, 15) = 3$$

Pro $|x\rangle = |4\rangle_4$ jsme obdrželi oba faktory p, q.

2. Změřená hodnota registru $|x\rangle = |8\rangle_4$:

$$j\frac{16}{r} = 8 \implies j = 1, r = 2 \text{ nebo } j = 2, r = 4 \text{ (pro } r = 4 \text{ již máme řešení)}$$

kontrola : $r = 2$ je sudé číslo

$$x \equiv 13^{2/2} \mod 15 \equiv 13$$
$$\text{NSD}(13+1, 15) = 1$$
$$\text{NSD}(13-1, 15) = 3$$

Pro $|x\rangle = |8\rangle_4$ jsme obdrželi částečný výsledek - jeden netriviální faktor.

3. Změřená hodnota registru $|x\rangle = |12\rangle_4$:

$$j\frac{16}{r} = 12 \implies j = 3, r = 4 \text{ (pro } r = 4 \text{ již máme řešení)}$$

Pro $|x\rangle = |12\rangle_4$ jsme obdrželi oba faktory p, q.

4. Změřená hodnota registru $|x\rangle = |0\rangle_4$:

$$j\frac{16}{r} = 0 \implies \text{neposkytuje žádnou informaci o } r$$

Pro $|x\rangle = |0\rangle_4$ je třeba provést další měření.

Z analýzy vyplývá, že 3 ze 4 výstupů poskytují dostatek informací pro nalezení faktorů pro vstup $N = 15, a = 13$. V obecné rovině lze říci, že pravděpodobnost vhodného výstupu QPE ($r =$ sudé číslo $\wedge\ a^{r/2} \equiv \mod N$) je přibližně 1/2.

7.5 Implementace Shorova faktorizačního algoritmu

Mnoho experimentálních demonstrací Shorova algoritmu spoléhá na významné optimalizace založené na předchozí znalosti očekávaných výsledků. Bez vysoké optimalizace se při demonstraci neobejdeme, neboť simulovat kvantové systémy o více, než několika jednotek qubitů je velice náročné, zejména z hlediska hardwarových zdrojů. Nicméně následná ukázka implementace algoritmu je poměrně snadno škálovatelná.

Algoritmus provede rozklad čísla $N = 15$.

Pro jiná rozkládaná čísla je třeba provést optimalizaci kvantové části algoritmu. Klasická část algoritmu je platná pro rozklad poloprvočísel.

K implementaci algoritmu budeme potřebovat zejména knihovnu Qiskit a také využijeme knihovny pro některé matematické operace a zpracování výstupu.

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 from qiskit import QuantumCircuit, Aer, transpile
4 from qiskit.visualization import plot_histogram
5 from math import gcd
6 import pandas as pd
```

Kvantová část Shorova algoritmu je de facto QPE protokol.

Unitární operátor funkce U_f lze rozepsat jako:

$$\begin{aligned} U_{f_{a,N}}(x) &\equiv a^x \mod N \\ &\equiv a^{2^{n-1}x_1 + 2^{n-2}x_2 + \cdots + 2^0 x_n} \mod N \\ &\equiv a^{2^{n-1}x_1} a^{2^{n-2}x_2} \ldots a^{2^0 x_n} \mod N \end{aligned} \tag{7.40}$$

V QPE diagramu (obr. 7.12) provádí unitární operátory U stejnou operaci na kontrolním registru jako operátory funkce U_f.

Nejprve si nadefinujeme vlastní operátor pro kvantový obvod podle funkce U_f.

```
7  def a_x_mod15(a, power):
8      """ operátor U = a mod 15 """
9      if a not in [2,4,7,8,11,13]:
10         raise ValueError("'a' musí být 2,4,7,8,11 nebo 13")
11     circuit = QuantumCircuit(4)
12     for iteration in range(power):
13         if a in [2,13]:
14             circuit.swap(2,3) # SWAP operace nad qubity
15             circuit.swap(1,2)
16             circuit.swap(0,1)
17         if a in [7,8]:
18             circuit.swap(0,1)
19             circuit.swap(1,2)
20             circuit.swap(2,3)
21         if a in [4, 11]:
22             circuit.swap(1,3)
23             circuit.swap(0,2)
24         if a in [7,11,13]:
25             for q in range(4):
26                 circuit.x(q) # NOT operace nad qubity
27     circuit = circuit.to_gate()
28     circuit.name = "%i^%i mod 15" % (a, power)
29     c_U = circuit.control()
30     return c_U
```

Funkce *a_x_amod15(a, power)* vytváří kontrolovanou verzi modulárního exponenciálního operátoru $U = a^x \mod 15$.

Funkce přijímá dva argumenty:

- a je základ exponenciace $a \in \{2, 4, 7, 8, 11, 13\}$.
- *power* je exponent, který určuje, kolikrát se má operátor U aplikovat. Tím dostaneme operaci U^x, kde x je počet opakování.

Funkce nejprve vytvoří nový kvantový obvod *circuit* se 4 qubity. Tento obvod reprezentuje operátor U. Funkce iteruje přes *power* a v každé iteraci upravuje obvod podle zadané hodnoty a. Zde jsou použity operátory SWAP a NOT (X) pro implementaci různých operací.

Po provedení všech iterací se obvod *circuit* převede na operátor pomocí metody *to_gate()* a přiřadí se mu jméno podle zadaných hodnot a a *power*.

Nakonec se sestavený operátor U pomocí metody *control()* změní na kontrolní unitární operátor c_U a vrátí se jako výstup funkce *a_x_amod15()*.

Dále vytvoříme funkci pro přidání operátorů c_U pomocí metody *append(operátor, cílový qubit)*, která zároveň definuje kontrolní qubit operátoru c_U.

```
31 def modular_exponentiation(circuit, n, m, a):
32     """ opakovaná aplikace kontrolního U operátoru """
```

```
33      for x in range(n):
34          power = 2**x
35          circuit.append(a_x_mod15(a, power), [x] + list(range(n, n+m)))
```

Funkce *modular_ exponentiation* je na první pohled exponenciální funkcí 2^x, nicméně její časová složitost je ve skutečnosti $\mathcal{O}(log(x))$, neboť využívá binární reprezentace čísel. Jednoduše řečeno při postupném umocňování stačí k binárnímu číslu přidat další cifru. Například: $2^2_{10} = 100_2$, $2^3_{10} = 1000_2$ a $2^4_{10} = 10000_2$.

Dále v kvantovém obvodu použijeme inverzní QFT funkci, kterou jsme si popsali v kapitole QFT protokol.

```
36  def qft_inverse(n):
37      """ inverzni QFT pro n qubitu """
38      circuit = QuantumCircuit(n)
39
40      for qubit in range(n//2):
41          circuit.swap(qubit, n-qubit-1)
42      for j in range(n):
43          for m in range(j):
44              circuit.cp(-np.pi/float(2**(j-m)), m, j)
45          circuit.h(j)
46      circuit.name = "QFT^(-1)"
47      return circuit
```

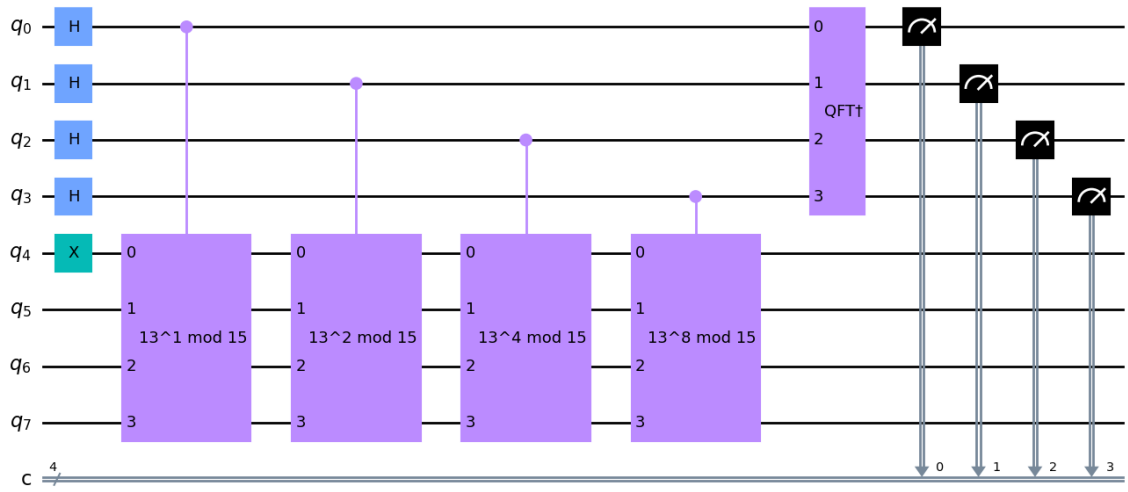
Funkce *shor_ qpe* sestaví kvantový obvod a provede měření kontrolního registru.

```
48  def shor_qpe(n, m, a):
49      """ QPE obvod """
50      # Vytvoření kvantového obvodu s 'n' qubity v kontrolním registru,
51      # se 'm' qubity v cílovém registru a 'n' bity v klasickém registru
52      circuit = QuantumCircuit(n+m, n)
53
54      # Initialize kontrolního registru |+>
55      circuit.h(range(n))
56
57      # inicializace cílového registru |1>
58      circuit.x(n+m-1)
59
60      # apply modular exponentiation
61      modular_exponentiation(circuit, n, m, a)
62
63      # aplikace qft_inverse funkce
64      circuit.append(qft_inverse(n), range(n))
65
66      # měření kontrolního registru
67      circuit.measure(range(n), range(n))
68
69      return circuit
```

Specifikujeme proměnné pro kvantový obvod *shor_ circuit*.

```
n = 4 # počet qubitů v kontrolním registru
m = 4 # počet qubitů v cílovém registru
a = 13 # 13, je vstup Shorova algoritmu

shor_circuit = shor_qpe(n, m, a)
shor_circuit.draw('mpl')
```

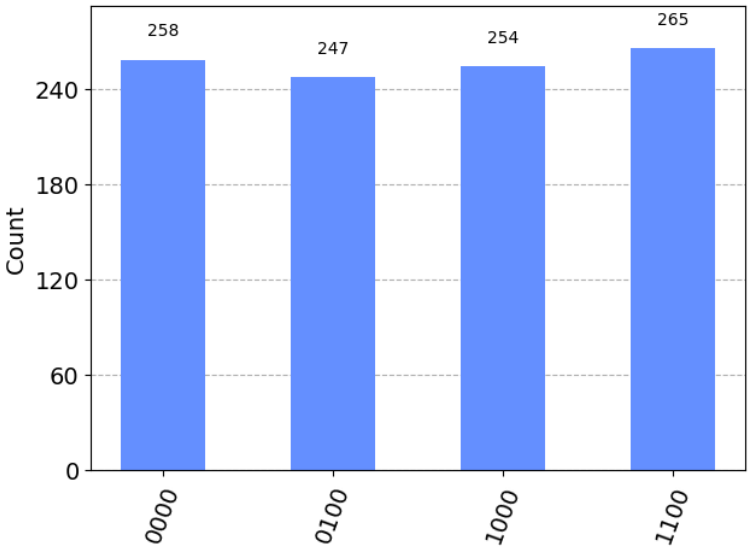


Obrázek 7.16 Výstup: Kvantová část Shorova algoritmu

Provedeme simulaci obvodu v backendu Aer simulátor a naměřené hodnoty zobrazíme v histogramu (obr. 7.17).

Platforma IBMQ neposkytuje přístup veřejnosti ke kvantovým počítačům s více než 7 qubity (náš obvod tvoří celkem 8 qubitů a tak si musíme vystačit se simulací). Nicméně v případě, že bychom chtěli navržený obvod optimalizovat na konkrétní kvantový počítač z IBMQ platformy, je možné si stáhnout od provozovatele aktuální kalibrační data z daného kvantového počítače . Aer z kalibračních dat vytvoří šumové modely, které lze použít k simulaci dekoherence, chyb kvantových hradel a chyb při inicializaci, nebo při měření obvodu. To umožňuje uživatelům zkoumat vliv těchto chyb na výkon kvantových algoritmů a obvodů a přizpůsobit je tak, aby byly robustní vůči reálným kvantovým chybám.

```
aer_sim = Aer.get_backend('aer_simulator')
t_circuit = transpile(shor_circuit, aer_sim)
results = aer_sim.run(t_circuit).result()
counts = results.get_counts()
plot_histogram(counts)
```



Obrázek 7.17 Výstup: Naměřené hodnoty obvodu *shor_ QPE*

Z kvantové části Shorova algoritmu jsme naměřili v simulaci hodnoty $\{0, 4, 8, 12\}$, které odpovídají předpokládanému výstupu kontrolního registru.

V klasické části zpracujeme výstup z kvantového obvodu *shor_ circuit*. Kód na dopočítání prvočíselného rozkladu může vypadat například takto:

```
from math import gcd # funkce NSD
print("změřená hodnota, perioda, rozklad")
for measured_value in counts:
    #změřená hodnota z binárního čísla na decimální číslo
    measured_value_decimal = int(measured_value, 2)
    if measured_value_decimal == 0:
        print(f"{measured_value}(bin), {measured_value_decimal}(dec), r=None")
        continue
    # výpočet periody 'r' z j*(M/r)
    min_r = float('inf')
    for j in range(1, measured_value_decimal * (n**2)):
        r = (j * (n**2)) / measured_value_decimal
        if r.is_integer():
            min_r = min(min_r, int(r))
            break
    # kontrola: r = sudé číslo
    if min_r % 2 != 0:
        print("Chyba. perioda r není sudé číslo")
        continue
    # výpočet x = a^(r/2) mod N
    x = int((a ** (min_r / 2)) % N)
    # kontrola: x + 1 není kongruentní s 0 mod N
    if (x + 1) % N == 0:
        print("Chyba. x + 1 = 0 mod N, kde x = a^(r/2) mod N")
        continue

    guesses = gcd(x + 1, N), gcd(x - 1, N)
    print(f"{measured_value}(bin), {measured_value_decimal}(dec),
```

```
109             r={min_r}, {guesses}")
```

```
Out[1]:     změřená hodnota, perioda, rozklad
            1000(bin), 8(dec), r=2, (1, 3)
            0000(bin), 0(dec), r=None
            0100(bin), 4(dec), r=4, (5, 3)
            1100(bin), 12(dec), r=4, (5, 3)
```

Zde uvedený kód ilustrativně počítá rozklad p a q pro všechny naměřené hodnoty v kvantové části algoritmu, tak aby si čtenář mohl snadněji prohlédnout možné varianty výstupu kvantového obvodu.

V simulaci Aer bylo provedeno přibližně 1000 opakování. Ve skutečnosti tento obvod má pravděpodobnost 3:4, že získá alespoň jeden netriviální faktor čísla 15 hned v první iteraci.

Je třeba zmínit, že Shorův algoritmus je již implementován v knihovně Qiskit jako třída *Shor(N, a, quantum_ instance)*. Vstup N je rozkládané číslo, a je náhodné číslo v rozsahu $1 < a < N$, zároveň $\mathrm{NSD}(a, N) = 1$ a *quantum_ instance* určuje typ použitého backendu.

```
1  from qiskit import Aer
2  from qiskit.algorithms import Shor
3  from qiskit.utils import QuantumInstance
4
5  N = 15
6  a = 13
7
8  quantum_instance = QuantumInstance(Aer.get_backend('aer_simulator'))
9  result = Shor(quantum_instance=quantum_instance).factor(N, a)
10
11 print(f"Výsledek: {result.factors}")
```

Poznámka: Veškerý kód obsažený v práci tvoří přílohu č. 1 *Zdrojový kód Shorova algoritmu.*

Dílčí závěr

Tento obsáhlý popis Shorova faktorizačního algoritmu měl za cíl seznámit čtenáře nejen s algoritmem samotným, ale poskytnou i návod krok za krokem, jak provést dílčí simulace algoritmů na reálném kvantovém počítači.

ZÁVĚR

Práce se zaměřila na problematiku odolnosti současného šifrování proti útokům kvantových počítačů.

Úvodní kapitola teoretické části provedla čtenáře základy kryptografie, terminologií a jednoduchými koncepty šifrování, jejichž původ nalezneme už v antickém Římě.

V kapitole o moderní (počítačové) kryptografii byly tyto koncepty rozvíjeny na příkladu symetrické šifry AES, nebo na příkladu šifry s veřejným klíčem RSA. Nicméně podstatou této kapitoly je analýza odolnosti symetrických a asymetrických šifer proti útokům kvantových počítačů.

V analýze byly popsány možnosti využití dvou kvantových algoritmů k odkrytí chráněné informace. První Groverův algoritmus je schopen redukovat délku bitového klíče na polovinu u symetrických šifer a hašovacích funkcí. Závěr analýzy sdělil, že útoky pomocí Groverova algoritmu neohrozí šifry s délkou klíče 256 bitů. Druhý Shorův algoritmus řeší matematické problémy faktorizace velkých čísel, diskrétní logaritmy a šifrování na eliptických křivkách v polynomiálním čase. Z analýzy útoku s využitím Shorova algoritmu na asymetrické kryptosystémy vyplynula nezbytnost využití jiných matematických problémů, které nebudou mít řešení v rozumném čase na kvantových počítačích. Závěr analýzy zmiňuje skutečnost, že potřebný hardware pro implementaci těchto kvantových algoritmů na provedení útoku na současná kryptoschémata vyžaduje další výzkum, a to zejména v přesnosti prováděných výpočtů na kvantových počítačích.

Kapitola s názvem post-kvantová kryptografie popisuje několik matematických problémů, které by měly zajistit nové generaci šifer odolnost před útoky kvantových počítačů. Dále kapitola poskytuje souhrnnou informaci o probíhajícím procesu standardizace kvantově odolných šifer.

Závěr teoretické části patří Kapitolám matematický aparát a základní pojmy kvantových výpočtů. Zde se čtenář mohl seznámit s matematickými nástroji kvantového počítání, které byly uplatněny v praktické části této práce.

Hlavním obsahem praktické části byla ukázka implementace Shorova faktorizačního algoritmu na kvantovém počítači. Shorův algoritmus sestává ze dvou částí. Klasická část algoritmu připraví vstup do kvantové části a poté řeší zpracování výstupu. V kvantové části algoritmu byl demonstrován paralelní výpočet na qubitech ve stavu superpozice a využití jevu kvantové provázanosti. Práce zahrnuje výsledky měření výpočtů na reálném kvantovém počítači z platformy IBMQ.

Výstupem praktické části je podrobný popis funkce Shorova algoritmu včetně komentovaného zdrojového kódu, na kterém jsou vysvětleny některé postupy programování kvantových algoritmů. Čtenář má možnost si vyzkoušet si celý algoritmus krok za

krokem a porovnat své výsledky se závěry této práce.

Post-kvantová kryptografie, jak bylo ukázáno v této práci, je komplexní a stále se vyvíjející obor. Přestože jsou dosavadní výsledky slibné, je zde stále mnoho otázek, které je třeba zodpovědět, a problémů, které je třeba vyřešit. Je zřejmé, že potřeba robustních a efektivních post-kvantových kryptosystémů bude v blízké budoucnosti stále naléhavější.

Cílem této práce bylo přispět k lepšímu pochopení problematiky a ukázat, jak se skloubí matematika, fyzika a informatika v řešení problémů, které představují hrozbu pro digitální bezpečnost.

Doufám, že tato práce dobře poslouží dalším.

SEZNAM POUŽITÉ LITERATURY

[1] KAN, Michael. Google Claims Quantum Computing Achievement, IBM Says Not So Fast. *PC Magazine* [online]. New York: Ziff-Davis Media, 2019, October 23, 2019 [cit. 2023-02-27]. ISSN 0888-8507. Dostupné z: https://www.pcmag.com/news/google-claims-quantum-computing-achievement-ibm-says-not-so-fast

[2] JAŠEK, Roman, David MALANÍK a Nicol DAŇKOVÁ. *Bezpečnost informačních systémů* [online]. Druhé vydání. Zlín: Univerzita Tomáše Bati ve Zlíně, 2022 [cit. 2022-11-09]. ISBN 978-80-7678-088-0. Dostupné z: https://digilib.k.utb.cz/handle/10563/52082

[3] HOFFSTEIN, Jeffrey, Jill PIPHER a Joseph H. SILVERMAN. *An introduction to mathematical cryptography*. New York: Springer, 2008. ISBN 978-0-387-77993-5.

[4] SINGH, Simon. *Kniha kódů a šifer: tajná komunikace od starého Egypta po kvantovou kryptografii*. 2. vyd. Přeložil Dita ECKHARDTOVÁ, přeložil Petr KOUBSKÝ. Praha: Dokořán, 2009. ISBN 978-80-7363-268-7.

[5] CRUISE, Brit. Cesta do světa kryptografie. *Khan Academy* [online]. 2014 [cit. 2023-03-07]. Dostupné z: https://cs.khanacademy.org/computing/computer-science/cryptography

[6] CEPHEUS. Caesar3.svg. In: *Wikimedia Commons* [online]. St. Petersburg (Florida): Wikimedia Foundation, 1 October 2006, pod licencí Public Domain, [cit. 2022-07-17]. Dostupné z: https://commons.wikimedia.org/wiki/File:Caesar3.svg

[7] KRÁLÍK, Jan. Statistika českých grafémů s využitím moderní výpočetní techniky. *Slovo a slovesnost* [online]. Praha: Ústav pro jazyk český Akademie věd České republiky, 1983, 44(4) [cit. 2022-11-17]. ISSN 2571-0885. Dostupné z: http://sas.ujc.cas.cz/archiv.php?art=2913

[8] Vigenère ciphers. In: *Encyclopedia Britannica* [online]. United Kingdom: Encyclopædia Britannica, 2022. [cit. 2022-07-03]. Dostupné z: https://www.britannica.com/topic/Vigenere-cipher

[9] KASISKI, Friedrich Wilhelm. *Die Geheimschriften und die Dechiffrir-Kunst: Mit bes. Berücks. der deutschen u. der franz. Sprache*. Berlin: E. S. Mittler, 1863.

[10] FRIEDMAN, Milton. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Associ-*

ation [online]. 1937, 32(200), 675-701 [cit. 2023-04-23]. ISSN 0162-1459. Dostupné z: doi:10.1080/01621459.1937.10503522

[11] SHANNON, Claude E. Communication theory of secrecy systems. *The Bell System Technical Journal* [online]. 1949, October 1949, 28(4), 656 - 715 [cit. 2022-11-17]. ISSN 0005-8580. Dostupné z: doi:10.1002/j.1538-7305.1949.tb00928.x

[12] STANOVSKÝ, David a Libor BARTO. *Počítačová algebra*. Druhé, upravené vydání. Praha: Matfyzpress, 2017. ISBN 978-80-7378-340-2.

[13] KRÁLÍK, Lukáš. *Programování* [online] [skripta pro studenty]. Zlín: Univerzita Tomáše Bati ve Zlíně, 2020 [cit. 2022-11-09]. Dostupné z: https://moodle.utb.cz/pluginfile.php/702404/mod_resource/content/1/ESF-l_kralik-programovani-seminar.pdf

[14] ToOb. Digital Signature diagram cs.svg. In: *Wikimedia Commons* [online]. St. Petersburg (Florida): Wikimedia Foundation, 29 June 2009, pod licencí Creative Commons, [cit. 2023-04-07]. Dostupné z: https://commons.wikimedia.org/wiki/File:Digital_Signature_diagram_cs.svg

[15] CHRISTIAN, Brian a Tom GRIFFITHS. *Algoritmy pro život: jak využít počítačové algoritmy při každodenním rozhodování*. Přeložil Filip DRLÍK. Brno: Jan Melvil Publishing, 2017. ISBN 978-80-7555-037-8.

[16] VON NEUMANN, John. Various techniques used in connection with random digits, Notes by G. E. Forsythe. Journal of Research of the National Bureau of Standards. 1951, (12), 36-38. ISSN 1044-677X.

[17] BURDA, Karel. *Aplikovaná kryptografie*. Brno: VUTIUM, 2013. ISBN 978-80-214-4612-0.

[18] FIPS 197: Advanced Encryption Standard (AES) [online]. *National Institute of Standards and Technology*, May 9, 2023 [cit. 2023-05-13]. Dostupné z: doi:10.6028/NIST.FIPS.197-upd1

[19] GROVER, Lov K. A fast quantum mechanical algorithm for database search. *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*. New York: ACM, 1996, s. 212-219. ISBN 0-89791-785-5.

[20] MORIARTY, Kathleen, Burt Kaliski, Jacob Jonsson a Andreas Rusch. PKCS #1: RSA Cryptography Specifications Version 2.2 [online]. *RFC 8017*, November 2016 [cit. 2023-05-13]. Dostupné z: doi:10.17487/RFC8017

[21] SHOR, Peter W. Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science* [online]. IEEE Comput. Soc. Press, 1994, s. 124-134 [cit. 2023-03-08]. ISBN 0-8186-6580-7. Dostupné z: doi:10.1109/SFCS.1994.365700

[22] YAN, Bao et al. Factoring integers with sublinear resources on a superconducting quantum processor. *ArXivLabs* [online]. December 2022 [cit. 2023-05-15]. Dostupné z: doi:10.48550/arXiv.2212.12372

[23] USA. H.R.7535 - *Quantum Computing Cybersecurity Preparedness Act*. 2022, Public Law 117-260. Dostupné také z: https://www.congress.gov/bill/117th-congress/house-bill/7535

[24] Post-Quantum Cryptography. *National Institute of Standards and Technology* [online]. Gaithersburg, MD, November 30, 2017 [cit. 2023-05-17]. Dostupné z: https://csrc.nist.gov/projects/post-quantum-cryptography

[25] CASTRYCK, Wouter a Thomas DECRU. An efficient key recovery attack on SIDH. *Cryptology ePrint Archive, Paper 2022/975* [online]. 2022 [cit. 2023-05-17]. Dostupné z: https://eprint.iacr.org/2022/975

[26] KUPČA, Vojtěch. *Teorie a perspektiva kvantových počítačů*. Praha, 2001. Diplomová práce. ČVUT, Fakulta elektrotechnická.

[27] GRUSKA, Jozef. *Quantum Computing*. London: McGraw-Hill, 1999. ISBN 978-0077095031.

[28] BEČVÁŘ, Jindřich. *Lineární algebra*. Vydání páté. Praha: Matfyzpress, 2019. ISBN 978-807-3783-785.

[29] DIRAC, Paul A. M. Bakerian Lecture - The physical interpretation of quantum mechanics. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* [online]. London: Royal Society of London, 1942, 180(980), 1-40 [cit. 2022-07-05]. ISSN 0080-4630. Dostupné z: doi:10.1098/rspa.1942.0023

[30] LUKAŠÍK, Petr a Martin SYSEL. Základní Principy Kvantového Výpočetního Systému. *Trilobit* [online]. Zlín: Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky, 2012, 2, [cit. 2022-07-03]. ISSN 1804-1795. Dostupné z: http://trilobit.fai.utb.cz/zakladni-principy-kvantoveho-vypocetniho-systemu_1e06ceed-17c9-495d-b3b7-9363bed2c098

[31] HIDARY, Jack D. *Quantum Computing: An Applied Approach*. 2nd. ed. Cham, Switzerland: Springer, 2021. ISBN 978-3-030-83273-5.

[32] BERNHARDT, Chris. *Quantum computing for everyone*. Cambridge, MA: The MIT Press, 2020. ISBN 978-0-262-53953-1.

[33] BARNCO, Adriano et al. Elementary gates for quantum computation. *Physical Review* A [online]. American Physical Society, 1995, 52(5), 3457-3467 [cit. 2022-07-05]. ISSN 2469-9934. Dostupné z: doi:10.1103/PhysRevA.52.3457

[34] EASTIN, Bryan a Steven T. FLAMMIA. *Q-circuit Tutorial* [online]. 24 Aug 2004 [cit. 2023-05-04]. Dostupné z: doi:arXiv:quant-ph/0406003

[35] SIMON, Daniel R. On the Power of Quantum Computation. *SIAM Journal on Computing* [online]. 1997, 26(5), 1474-1483 [cit. 2023-05-04]. ISSN 0097-5397. Dostupné z: doi:10.1137/S0097539796298637

[36] *Learn Quantum Computation Using Qiskit* [online]. Armonk, NY: IBM, 2020, [cit. 2022-07-03]. Dostupné z: http://community.qiskit.org/textbook

[37] ASFAW, Abraham. 2020 Qiskit Global Summer School on Quantum Computing and Quantum Hardware [online]. Qiskit, 2020 [cit. 2023-05-20]. Dostupné z: https://qiskit.org/learn/summer-school/introduction-to-quantum-computing-and-quantum-hardware-2020/

[38] LOREDO, Robert. *Learn Quantum Computing with Python and IBM Quantum Experience: A hands-on introduction to quantum computing and writing your own quantum programs with Python*. Birmingham, UK: Packt Publishing, 2020. ISBN 978-1-83898-100-6.

[39] POLÁŠEK, Vladimír, Lubomír SEDLÁČK a Lenka KOZÁKOVÁ. *Matematický seminář*. Zlín: Univerzita Tomáše Bati ve Zlíně, 2018. ISBN 978-80-7454-687-7.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

$\mathbb{Z}$	obor celých čísel
$\mathbb{N}$	obor přirozených čísel (bez nuly)
$\mathbb{R}$	obor reálných čísel
$\mathbb{C}$	obor komplexních čísel
$\mathbb{R}^+$	obor všech kladných reálných čísel
$\otimes$	tensorový součin
$a\|b$	skalární součin a, b
$\approx$	aproximace
$:=$	přiřazení
$\equiv$	kongruence
$\lfloor x \rfloor$	dolní celá část x
$\|\|x\|\|$	norma x
$\wedge$	AND
$\vee$	OR
$\oplus$	exclusive OR (XOR)
$\mathbb{R}^n$	n-dimenzionální reálný vektorový prostor
$\|x\|$	absolutní hodnota x
P	pravděpodobnostní míra
$\sum$	sumace
$\Longrightarrow$	implikace
$\forall$	kvantifikátor
mod	modulo
NSD	Největší společný dělitel
AES	Advanced Encryption Standard
API	Application Programming Interface
AT&T	American Telephone & Telegraph Company
BIKE	Bit-flipping Key Encapsulation
CA	Certification Authority
CNOT	Controlled NOT
CP	Controlled Phase
CPA	Chosen Plaintext Attack
CPU	Central Processing Unit
CROT	Controlled Rotation
CVP	Closest Vector Problem
CSPRNG	Cryptographically Secure Pseudorandom Number Generator
DES	Data Encryption Standard

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ECC	Elliptic-Curve Cryptography
FEAL	Fast Data Encipherment Algorithm
FOCS	Symposium on Foundations of Computer Science
GNFS	General Number Field Sieve
HQC	Hamming Quasi-Cyclic
HTTPS	Hypertext Transfer Protocol Secure
IBM	International Business Machines Corp.
IMBQ	IBM Quantum
IDE	Integrated Development Environment
IV	Initialization Vector
KEM	Key Encapsulation Mechanism
LSB	Least Significant Bit
MIT	Massachusetts Institute of Technology
MQ	Multivariate Quadratic
MSB	Most Significant Bit
NIST	National Institute of Standards and Technology
OAEP	Optimal Asymmetric Encryption Padding
OTP	One-Time Pad
PKCS	Public-Key Cryptography Standards
PKI	Public Key Infrastructure
PQC	Post-Quantum Cryptography
Qiskik	Quantum Information Software Kit for quantum computation
QFT	Quantum Fourier Transform
QKD	Quantum Key Distribution
QPE	Quantum Phase Estimation
RSA	Rivest-Shamir-Adleman
SIKE	Supersingular Isogeny Key Encapsulation
SSL	Secure Sockets Layer
SVP	Shortest Vector Problem
TLS	Transport Layer Security
UROT	Unitary Rotation

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I. ZDROJOVÝ KÓD SHOROVA ALGORITMU

Přiložený soubor ve formátu Jupyter notebook (priloha1.ipynb) obsahuje zdrojový kód Shorova algoritmu ze všech kapitol na jednom místě. Navíc je v kódu přidána část (simulator Aer), která umožní spouštět kvantové obvody lokálně bez nutnosti registrace na platformě IBMQ.

Doporučuji vytvořit nový environment. Můžete použít následující postup: Pokud ještě nemáte nainstalovaný Conda, stáhněte si Miniconda pro váš operační systém z oficiálního webu:

https://docs.conda.io/en/latest/miniconda.html

1. Otevřete terminál nebo příkazový řádek a spusťte následující příkaz pro vytvoření nového prostředí s názvem "myenv"

   ```
   conda create --name myenv
   ```

2. Aktivujte nově vytvořené prostředí pomocí příkazu:

   Pro Windows: `conda activate myenv`

   Pro macOS a Linux: `source activate myenv`

3. Nainstalujte balíčky potřebné pro tento kód:

   ```
   conda install -c conda-forge jupyter numpy qiskit matplotlib
   ```

4. Ze stejného terminálu nebo příkazového řádku, kde je aktivované prostředí, spusťte Jupyter Notebook:

   ```
   jupyter notebook
   ```

Po dokončení práce s prostředím jej deaktivujte pomocí příkazu:

```
conda deactivate
```