UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2025 Patricie Mecová

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Porovnání postkvantových šifrovacích algoritmů s klasickými šifrovacími algoritmy

Bakalářská práce

2025 Patricie Mecová

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Patricie Mecová**
Osobní číslo:	**I21190**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Porovnání postkvantových šifrovacích algoritmů s klasickými šifrovacími algoritmy**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem této bakalářské práce je porovnání postkvantových šifrovacích algoritmů s šifrovacími algoritmy, které se dnes běžně používají.
V teoretické části práce budou popsány bezpečnostní problémy současných šifrovacích algoritmů. Dále zde budou představeny
nové postkvantové algoritmy podle standardu NIST.
V praktické části práce bude porovnány výkonové parametry současných a postkvantových algoritmů.

Rozsah pracovní zprávy: **30**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

GROŠEK, Otakar a PORUBSKÝ, Štefan. Šifrovanie – algoritmy, metódy, prax. Praha: Grada, 1992. ISBN 80-85424-62-2.
OULEHLA, Milan a JAŠEK, Roman. Moderní kryptografie. [Praha]: IFP Publishing, 2017. ISBN 978-80--87383-67-4.
Bernstein Daniel J., Buchmann Johannes, Dahmen Erik. Post-Quantum Cryptography: Springer; 2009th edition. ISBN 978-3540887010

Vedoucí bakalářské práce: **Ing. Martin Pozdílek, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. prosince 2024**
Termín odevzdání bakalářské práce: **16. května 2025**

L.S.

prof. Ing. Petr Doležel, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2025

Prohlašuji:

Práci s názvem Porovnání postkvantových šifrovacích algoritmů s klasickými šifrovacími algoritmy jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 05. 03. 2025

Patricie Mecová

PODĚKOVÁNÍ

Tímto bych ráda poděkovala mému vedoucímu bakalářské práce Ing. Martinu Pozdílkovi, Ph.D. za cenné konzultace, rady a trpělivost při vypracování této bakalářské práce. Dále bych chtěla poděkovat své rodině za podporu během studia.

ANOTACE

Cílem této bakalářské práce je porovnání postkvantových šifrovacích algoritmů s šifrovacími algoritmy, které se dnes běžně používají.

V teoretické části práce budou popsány bezpečnostní problémy současných šifrovacích algoritmů. Dále zde budou představeny nové postkvantové algoritmy podle standardu NIST.

V praktické části práce bude porovnány výkonové parametry současných a postkvantových algoritmů.

KLÍČOVÁ SLOVA

Kryptografie, šifra, šifrovací algoritmy, kvantový počítač, postkvantová kryptografie, NIST.

TITLE

Comparison of post-quantum encryption algorithms with classical encryption algorithms

ANNOTATION

The aim of this bachelor's thesis is to compare post-quantum encryption algorithms with encryption algorithms that are commonly used today.

In the theoretical part of the thesis, the security problems of current encryption algorithms will be described. Furthermore, new post-quantum algorithms according to the NIST standard will be presented.

In the practical part of the thesis, the performance parameters of current and post-quantum algorithms will be compared.

KEYWORDS

Cryptography, cipher, encryption algorithms, quantum computer, post-quantum cryptography, NIST.

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

DES	Data Encryption Standard
IBM	International Business Machines Corporation
NIST	National Institute of Standards and Technology
OTS	One-Time Signature
RSA	Rivest, Shamir, Adleman

ÚVOD

S rostoucí výpočetní kapacitou moderních systémů a příchodem kvantových počítačů se klasická kryptografie dostává do bodu, kdy dnes hojně využívané algoritmy jako RSA, již nemusí být dostatečně bezpečné. To vyvolává potřebu nových kryptografických metod, nazývané postkvantové algoritmy, které by byly odolné vůči útokům s pomocí kvantového počítače.

Tato bakalářská práce si klade za cíl představit nejprve klasické šifrovací algoritmy, jejich princip fungování, výhody a případné nedostatky. Dále představí postkvantové algoritmy, jejich princip, druhy a dopad na klasické algoritmy.

V praktické části se práce zaměří na porovnání výkonových parametrů vybraných algoritmů na dvou různých zařízeních.

1 Klasické šifrovací algoritmy

Počátky kryptografie můžeme datovat do podobné doby jako vznik písma. Vzniká potřeba zajistit, aby přenášené informace byly srozumitelné pouze pro odesílatele a určeného příjemce. To vede k využívání šifrovacích metod, která mají za účel ochránit obsah zprávy před neoprávněným přístupem cizích osob. V dnešní době se kryptografie vyvinula v plnohodnotnou vědeckou disciplínu a s přechodem do internetového světa se stala důležitější než kdy předtím.

Pro zašifrování zpráv se využívají algoritmy, které převádí otevřený text na zašifrovaný text. Odesílatel i adresát zprávy vlastní klíč, který je použit k zašifrování a dešifrování zprávy. V kryptografii je klíč tajná informace, která určuje, jakým způsobem bude zpráva zašifrována a nebo dešifrována. Šifrovací klíč funguje jako vstupní parametr do šifrovacího algoritmu. Pokud se cizím osobám podaří komunikaci odposlechnout, mělo by být velmi obtížné, až nemožné zprávu bez klíče rozluštit. Šifrovací algoritmy tedy slouží k zachování soukromí a dále k ověření autenticity zprávy. Algoritmy jsou obvykle navrženy pro dva hlavní úkoly: obecné šifrování používané k ochraně informací jako jsou hesla vyměňovaná na veřejné síti a digitální podpisy používané k ověřování identity. [1]

Bezpečnostní hrozby

Jak již bylo zmíněno, důležité pro bezpečné šifrování pomocí symetrických šifer je bezpečná distribuce a uchování klíče. Postupem času může být bezpečnost klíčů oslabena, nabízí se možnost je pravidelně rotovat, aby se zachovala lepší bezpečnost dat. [2]

Další hrozbou může být útok hrubou silou zaměřující se na všechny možné kombinace klíče. Čím delší délka klíče, tím jsou algoritmy více odolné vůči tomuto typu útoku. Například DES s 54bitovým klíčem považujeme v dnešní době za již prolomený, oproti tomu AES, který používá 256bitový klíč považujeme za odolný. [3]

1.1 Symetrické šifrovací algoritmy

V symetrických šifrách je použit na obou stranách komunikace stejný klíč, a to jak pro zašifrování tak i dešifrování zprávy. Šifrovací i dešifrovací algoritmy jsou veřejně známé. Klíč je potřeba dobře zabezpečit, jelikož kdokoliv se znalostí klíče je schopen dešifrovat nebo zašifrovat komunikaci. Bezpečnostní problém tedy vzniká v tom, jak si komunikující strany vymění klíč bezpečným a způsobem. Způsobů bezpečné výměny klíčů může být více, nejčastěji jsou ale používané asymetrické algoritmy.

Napříč všemi šifrovacími algoritmy, symetrické algoritmy mají nejrychlejší implementace, to je dělá užitečné, pokud potřebujeme zašifrovat velké množství dat.

Symetrické šifry se dělí do dvou základních kategorií – blokové šifry a proudové šifry. Zásadní rozdíl mezi těmito dvěma postupy spočívá v tom, jakým způsobem je zpracován vstupní text. Zatímco blokové šifry pracují s bloky pevně definované velikosti, proudové šifry operují na jednotlivých bitech. [4]

1.1.1 Blokové šifry

Blokové šifry pracují se vstupními daty rozdělenými do bloků o pevné délce, přičemž data v bloku zpracovává současně. Tyto bloky nejčastěji mívají v moderních šifrách pevné velikosti o 64 nebo 128 bitech. Výstup blokové šifry pak má stejnou délku, jako vstupní blok.

Proces šifrování pomocí blokové šifry se dá shrnout do pár jednoduchých kroků. Nejprve se rozdělí vstupní text do bloků pevné délky, tedy např. 64 nebo 128 bitů, podle zvoleného algoritmu. Pokud délka textu není násobkem velikosti bloku, doplní redundantní informace, aby měl blok požadovanou délku. Následně je každý blok zašifrován pomocí klíče. Šifrovaný text poté vzniká spojením jednotlivých zašifrovaných bloků. Pro dešifrování šifry příjemce použije stejný klíč. [5]

Režimy provozu

Blokové šifry šifrují zprávy, které mají stejnou velikost jako délka jejich bloku. Režimy provozu definují, jak jsou tyto bloky šifrovány. Jedním z nich je například ECB (režim elektronického číselníku). V tomto režimu je každý blok zašifrován pomocí stejného klíče, stejné bloky otevřeného textu jsou tedy zašifrovány do stejných bloků šifrovaného textu. Z tohoto důvodu přestože je ECB jednoduchý na implementaci, není příliš bezpečný.

Dalším režimem je CBC (cipher block chaining). U tohoto režimu je pro každý následující blok před zašifrováním provedena operace XOR s předchozím zašifrovaným blokem. To zajistí, že identické bloky otevřeného textu nebudou identické po zašifrování. Nevýhodou je, že tento režim nelze paralelizovat a mohou se šířit chyby.

Režim CTR (čítače) generuje bloky klíčového toku šifrováním postupně narůstajících hodnot čítače. Čítačem může být jakákoliv funkce vytvářející jednoduché sekvence. Mezi výhody režimu CTR patří možnost paralelizace šifrování i dešifrování. Zároveň se možné chyby při zpracování nešíří do dalších bloků. Hodnoty čítače se ovšem nesmí opakovat, jinak hrozí narušení bezpečnosti. [6]

Schémata blokových šifer

DES byl v roce 1975 představen společností IBM. Skládal se z 64bitových bloků a 56bitového klíče. Kvůli příliš krátké velkosti klíče je v dnešní době tato šifra považována za nebezpečnou.

Double DES je varianta původního DES, přičemž používá dvě instance DES na stejném vstupním textu a dva klíče k šifrování textu. Šifra je tedy bezpečnější, než původní DES.

Další variantou DES je 3DES. Používá tři instance DES k šifrování stejného textu, jinak řečeno, nad vstupním blokem se algoritmus spustí třikrát. Pro zašifrování pomocí 3DES slouží 3 klíče. Nejprve se prvním klíčem text zašifruje, druhým klíčem dešifruje a třetí slouží pro konečné šifrování. Přestože je bezpečnější, než předchozí varianty DES, proces trojitého šifrování vyžaduje vyšší výpočetní výkon.

V roce 2001 byl společností NIST publikován standard AES. Byl navržen jako náhrada DES, používá 128bitovou velikost bloku a 128bitovou až 256bitovou velikost klíče. Je mnohem bezpečnější, než všechny varianty DES a v dnešní době je široce používán pro zabezpečení dat, databází, virtuálních privátních sítí nebo prohlížečů. [5]

1.1.2 Proudové šifry

Proudové šifry představují alternativní přístup k symetrickému šifrování, na rozdíl od blokových šifer, které aplikují algoritmus na jednotlivé bloky, při požití proudových šifer jsou data šifrována po jednotlivých bitech. To umožňuje rychlý a relativně jednoduchý proces šifrování.

Mezi největší výhody proudových šifer patří právě rychlost. Jakmile je vytvořen klíč, samotný proces šifrování a dešifrování je téměř okamžitý. Zároveň pro jejich použití není nutný složitý nebo drahý hardware.

Další výhodnou těchto šifer je možnost dešifrovat pouze vybrané úseky šifrovaného textu. Jelikož každý bit dat v šifrovaném textu odpovídá bitu dat původního textu na stejné pozici, je možné dešifrovat pouze část dokumentu a není potřeba čekat na dešifrování celého souboru. Zároveň kvůli schopnosti šifrovat data po jednotlivých bitech, lze informace postupně zašifrovat a odesílat, jakmile jsou data k dispozici. Kvůli této výhodě jsou proudové šifry stále používány například real time aplikacemi jako streamování nebo online hraní.

Mezi hlavní nevýhody proudových šifer patří nepříliš vysoká bezpečnost,. Dalším problémem je šíření chyb. Pokud je nějaký bit přijat chybně nebo dojde k chybě během přenosu, mohou být ovlivněny následující bity a dojít k chybě během dešifrování. [7]

1.1.3 Generování klíčů

Pro generování klíčů se nejčastěji používají generátory náhodných nebo pseudonáhodných klíčů. Jejich úkolem je generování posloupností, které se neoprávněným osobám jeví jako náhodné. Pro tyto generátory je důležité, že jednotlivé hodnoty mají stejnou pravděpodobnosti výskytu a jeví se vzájemně nezávislá. Můžeme je rozdělit na náhodné generátory, které generují na základě nějakého náhodného fyzikálního děje a pseudonáhodné, které generují pomocí stavového automatu. Generátor pseudonáhodných čísel je deterministický, pokud je nám znám počáteční seed, přechodová a konverzní funkce, je možné dosáhnout stejného vygenerovaného čísla. [8]

1.2 Asymetrické šifrovací algoritmy

Na rozdíl od symetrických šifrovacích algoritmech, u kterých slouží k šifrování i dešifrování jeden a ten samý klíč, u asymetrických šifer je klíč rozložen na veřejný a soukromý klíč. Veřejný klíč je k dispozici všem a slouží k zašifrování, naopak soukromý klíč by měl být střežen a znám pouze majiteli a slouží k dešifrování zprávy.

Jedním z účelů asymetrických šifer je poskytnutí bezpečného kanálu pro výměnu symetrického klíče mezi stranami. Dalším důležitým účelem jsou digitální podpisy, které slouží jako náhrada klasických vlastnoručních podpisů. Digitální podpis závisí na podepsané zprávě a jakákoliv nezávislá osoba by měla být schopná podpis ověřit. [4]

Asymetrické šifrovací algoritmy jsou často výpočetně drahé, oproti většinou používaným symetrickým schématům. Pro lepší efektivitu a ušetření zdrojů se zavádí hybridní šifrování. Asymetrické šifry slouží k zašifrování a bezpečné výměně klíčů symetrické šifry a poté se pokračuje v šifrování pomocí symetrické šifry. [1]

RSA

Mezi nejznámější algoritmy patří RSA, který je založen na obtížnosti faktorizovat velká čísla. Pro generování klíčů se vybírají dvě prvočísla, jejichž součin tvoří základ soukromého i veřejného klíče. Pro prolomení šifry teoreticky stačí vypočítat, o které prvočísla se jedná. Protože jsou zvolena velmi velká čísla, výpočet pomocí klasického počítače je prakticky téměř nemožný, příchod kvantových počítačů by ale mohl bezpečnost RSA ohrozit.

V současné době je algoritmus považován za bezpečný a je široce používán jak pro výměnu klíčů, tak i pro digitální podpisy a své využití najde například v online bankovnictví. Mezi jeho hlavní nevýhody patří velké velikosti klíčů, které vyžadují více zdrojů a úložného prostoru a pomalá rychlost zpracování oproti jiným šifrovacím algoritmům. Není vhodné ho tedy používat v situacích, které vyžadují neustálé šifrování a dešifrování velkého množství dat. [9]

Bezpečnost RSA šifry souvisí s délkou klíče. V dnešní době, konkrétněji do roku 2030 by podle NIST ještě mohla být přijatelná délka klíče o 2048 bitech. S ohledem na co nejnižší riziko kybernetických hrozeb je doporučeno použití delších klíčů, minimálně RSA-3072, případně i RSA-7680 nebo RSA-15380. S vyšší bezpečností ovšem přichází i větší nároky na výkon. [10]

1.3 Hashovací funkce

Hashovací funkce jsou důležitou součástí kryptografie. Jedná se o matematické funkce, které mají na vstupu text libovolné délky a mapují jej na otisk s pevnou délkou. Tento výstup můžeme také nazývat jako hash.

Od kryptografických hashovacích funkcí se očekávají určité vlastnosti. Prvním je, že se jedná o jednosměrnou funkci. Hash lze vypočítat jednoduše a rychle, ale opačný proces, nalezení původního vstupu je velmi složitý.

Další důležitou vlastností je odolnost vůči kolizím. Vzhledem k libovolné délce vstupu a pevné velikosti výsledného hashe se kolize očekávají, jejich záměrné nalezení by mělo být výpočetně nemožné.

Hashovací funkce jsou deterministické, tedy při stejném vstupu vrátí stejný výsledek. Poslední vlastností je takzvaný „lavinový efekt“. Stačí, aby byl změněn pouze jeden bit vstupu a následkem je velká změna na výstupu.

Jejich nevýhodou je nižší kryptografická bezpečnost. Jsou zranitelné vůči útokům hrubou silou a nejrůznějším kryptografickým útokům. Během let bylo nespočet z nich prolomeno. [11]

Přesto mají důležité uplatnění. Jedním z nich je kontrola integrity dat. Hashovací funkce lze použít k vytvoření kontrolního součtu, který lze použít k ověření změny dat. Změna může být důsledkem nejen útoku cizí osoby, ale i prosté chyby, například během stahování softwaru.

Hashovací funkce jsou důležité i pro fungování digitálních podpisů a určení jejich pravosti nebo při vytváření integrity transakcí v kryptoměnách.

Mezi nejznámější zástupce patří například MD5, přestože je z dnešního hlediska zastaralá, stále se vyskytuje v méně citlivých aplikací. Dalšími důležitými zástupci je SHA-256 uplatněna například v kryptoměnách a digitálních certifikátech nebo SHA-3, sloužící jako náhrada za některá zastaralé funkce. [12]

2 Kvantové algoritmy

2.1 Kvantový počítač

Kvantový počítač je výpočetní zařízení, které využívá principy kvantové mechaniky k provádění výpočtů. Na rozdíl od klasických počítačů, které pracují s bity (0 nebo 1), kvantové počítače používají qubity (kvantové bity), které mohou být ve stavu 0, 1 nebo superpozici obou současně. Díky této vlastnosti jsou kvantové počítače schopny paralelního zpracování obrovského množství výpočtů a teoreticky mohou prolomit některé současné šifrovací algoritmy (např. RSA). [13]

Koncept kvantových počítačů začíná nabírat na síle počátkem 80. letech, kdy vizionáři jako Richard Feynman, Paul Benioff a David Deutsch položili základy pro kvantové výpočetní modely a koncepci univerzálních kvantových počítačů. Během 80. a 90. let byl jejich popis formalizován, ukázalo se, že jsou na mnohé specializované problémy výkonnější než klasické počítače. To v roce 1994 demonstroval Peter Shor prvním kvantovým algoritmem, nyní známým jako Shorův algoritmus, kdy kvantový počítač efektivně vyřešil známý problém celočíselné faktorizace, na který neexistuje známý efektivní algoritmus využívající klasické počítače. Dalším významným algoritmem se stal v roce 1996 Groverův algoritmus, který prohledává netříděné databáze rychleji než klasické algoritmy. Groverův algoritmus dokáže prohledat databázi N prvků zhruba v $\sqrt{N}$ krocích, ve srovnání s klasickými algoritmy vyžadující přibližně N kroků se jedná o kvadratické zrychlení. Praktické příklady obou algoritmů vzbudily značný zájem dalších vědců a investorů o zkoumání nových kvantových algoritmů. Počátek 21. století je známý intenzivním závodem mezi technologickými giganty i nejrůznějšími startupy. [13]

Mezi nejdůležitější milníky patří v roce 2019 dosažení Quantum Supremacy neboli „kvantové nadvlády" společností Google. Jejich kvantový počítač se svým 53-qubitovým procesorem vyřešil konkrétní problém za 200 sekund, přičemž jeho vyřešení by klasickému počítači trval přibližně 10 000 let. Výzkumníci vedou výzkumy nových kvantových algoritmů pro zlepšení odvětví jako je optimalizace nebo strojové učení. [13]

Kvantové výpočty představují i významnou hrozbu, například Shorův algoritmus má potenciál prolomit šifrování RSA. To podnítilo výzkum kvantové kryptografie k zabezpečení komunikace a postkvantové kryptografie, jejímž cílem je vyvinutí nových kryptografických schémat odolných vůči útokům klasických i kvantových počítačů. [13]

2.2 Postkvantová kryptografie

Šifrovací algoritmy chrání důvěrné elektronické informace před neoprávněnými uživateli. Po desetiletí se většina algoritmů ukázala, jako dostatečně silná na to, aby zabránili útokům pomocí klasických počítačů. Nový typ zařízení, nazývaný kvantový počítač, by mohl tyto algoritmy prolomit. Je tedy potřeba vyvinout nové šifrovací metody, které dokážou odvrátit kybernetické útoky jak klasických počítačů, tak i kvantových počítačů. Tyto nové metody se nazývají postkvantové šifrovací algoritmy. [14]

Kvantové počítače mají potenciál řešit problémy, které klasické počítače vyřešit nedokáží. Mezi ně patří například problémy zahrnující souhru komplexních proměnných nebo řešení klasického problému „cestujícího obchodníka" - tedy nalezení nejefektivnější cesty přes řadu instancí. Řešení podobných problémů lze uplatnit například při návrhu léků či simulace složitých molekul. [14]

V současné době se některé ze šifrovacích algoritmů spoléhají na potíže klasických počítačů s faktorováním velkých čísel. Dostatečně výkonný kvantový počítač by však byl schopen prosévat všechny potenciální prvočinitele současně, než jeden po druhém, a tedy dospět k odpovědi exponenciálně rychleji. Odborníci takový počítač nazývají „kryptograficky relevantním". Namísto miliard let by takové zařízení mohlo prolomit šifrování několika dnů či hodin, to představuje zásadní bezpečnostní hrozbu. [14]

Existuje i riziko, kdy útočník může úspěšně zachytit šifrovaná data v naději, že někdy později je bude moct pomocí kvantového počítače zachytit. Jedná se o jeden z důvodů, proč je vhodné začít šifrovat citlivá data pomocí postkvantových technik co nejdříve. [14]

2.2.1 Kryptografie založená na hashovacích funkcích

Kryptografie založená na hashovacích funkcích používá hashovací funkce pro generování veřejného klíče několikanásobným hashováním hodnoty x0, například až do x3. Podepisující může použít x1 jako soukromý klíč a podepsat s ním zprávu. Příjemce se znalostí hodnoty x3 (veřejného klíče), může ověřit, zda je soukromý klíč platný. Tento podpis lze použít pouze

jednou, jedná se tedy o jednorázové podpisové schéma (OTS). Pokud by byl pár klíčů OTS využit k podepsání dvou různých zpráv, bylo by možné snadno zfalšovat podpisy.

Aby bylo možné podepisovat více zpráv bez nutnosti generovat pro každou z nich nový veřejný klíč, lze využít strukturu nazývanou jako Merkleho strom. Tato struktura umožňuje kombinovat více klíčů pod jedním společným veřejným klíčem.

Pro vytvoření veřejného klíče se nejprve vygeneruje několik párů soukromých a veřejných klíčů metodou jednorázového podpisu. Každý veřejný klíč se zahashuje, tyto hodnoty tvoří listy Merkleho stromu. Hashe listů se párují a opakovaně hashují, dokud nevznikne Merkle root, jediný kořenový hash, který bude sloužit jako hlavní veřejný klíč.

Výsledná struktura je binárním stromem a počet jeho listů, tedy počet zpráv, které je možné podepsat musí být mocninou dvou. Výška stromu odpovídá logaritmu tohoto počtu, to umožňuje škálovat počet podpisů bez exponenciálního nárůstu klíče. Od původního Merkleho návrhu byly vytvořeny různé vylepšené varianty podpisových schémat, například XMSS, LMS nebo SPHINCS.

Mezi výhody hashovacích schémat patří vysoká úroveň zabezpečení a relativně malá velikost klíčů která je činí rychlými a efektivními. Mezi nevýhody patří nutnost sledování použitých klíčů a riziko vyčerpání klíčů. Počet potřebných podpisů by měl být odhadnut předem kvůli vyšší efektivitě a náhodné znovupoužití klíče OTS by mohlo vést k padělání. Kvůli těmto nevýhodnám NIST nepovažuje hashovací schémata za vhodné pro všeobecné použití. [15][16]

2.2.2 Kryptografie založená na mřížkách

Kryptografie založená na mřížkách je jednou ze nejlépe analyzovaných oblastní postkvantové kryptografie. Jak název napovídá, jedná se o schéma založené na matematických problémech týkajících se mřížek. Mřížku si v tomto kontextu můžeme představit jako mřížku milimetrového papírů, používá množinu bodů umístěných v průsečících mřížky přímek. Mřížka popisuje vzor, který pokračuje donekonečna.

Vycházíme z množiny bodů, známých jako vektory. Mezi nimi lze provádět sčítání a odčítání v libovolných celočíselných násobcích. Výpočetně obtížný problém pak spočívá v nalezení bodů v mřížce, které leží blízko nuly nebo jsou blízké jinému konkrétnímu bodu. Přestože ve dvou rozměrech jsou tyto problémy poměrně jednoduché, například ve 400 rozměrech se jeho složitost stává extrémně náročnou.

K odvození soukromého klíče z veřejného by bylo nutné hrubou silou prohledat všechny možnosti. I přesto, že kvantové počítače by dokázaly toto hledání urychlit, předpokládá se, že ani ty by nebyly schopné vyřešit složité problémy založené na mřížkách v rozumném časovém horizontu. Mezi nejznámější algoritmy založených na mřížce patří CRYSTALS-KYBER, algoritmus pro šifrování a výměnu klíčů a CRYSTALS-Dilithium, algoritmus pro digitální podpisy. [17]

2.2.3 Kryptografie založená na kódování

Jedná se o oblast výzkumu, která se zaměřuje na studium kryptografických systémů založených na kódech pro opravu chyb. V digitální komunikaci může mít změna jediného bitu katastrofální důsledky, pokud neexistuje způsob, jak chyby detekovat a napravit. Jedním z takových mechanismů jsou kontrolní součty, které představují základní formu detekce chyb. Cílem těchto kódů, je zajistit maximální omezení chybovosti při minimálním zvýšení datového objemu.

S myšlenkou kódů pro opravu chyb poprvé přišel Robert McEliese v roce 1978. Použil konkrétní typ opravného kódu, binární Goppův kód, na který použil lineární transformaci. Na základní úrovni se McEliesův přístup rovná tajné faktorizaci podobná RSA.

Veřejný klíč je tvořen náhodně generovanou maticí – permutovanou verzí soukromého klíče, kterým je náhodný binární Goppův kód. Pouze vlastník soukromého klíče (Goppova kódu) může opravit zavedené chyby v šifrovaném textu a obnovit původní zprávu.

Princip fungování tohoto typu kryptografie tedy spočívá v tom, že odesílatel zprávy záměrně do šifrovacího procesu zavádí chyby, aby znemožnil jejich dekódování bez znalosti tajného klíče. Příjemce zprávy, který vlastní potřebné tajné informace (například často týkající se struktury kódu) je schopen tyto chyby opravit a zprávu správně dešifrovat, narozdíl od případného útočníka.

V minulosti byla McElieceova technika dlouho přehlížena, zejména kvůli tomu, že jeho přístup vyžadoval výrazně větší veřejné klíče oproti jiným metodám jako RSA. Zároveň se ale jeho metoda jeví jako kvantově odolná, konkrétně se zdá odolná vůči útokům využívající Shorovu metodu. S blížící se érou kvantových počítačů se jí tedy věnuje další pozornost. [18]

2.3 Postkvantové algoritmy dle standardu NIST

NIST (The National Institute of Standards and Technology) je jednou z nejstarších fyzikálních laboratoří v USA. Cílem organizace je podpora inovace a konkurenceschopnosti pokrokem ve

vědě, standardech a technologii, které zvýší ekonomickou bezpečnost a zlepší kvalitu života. [19]

2.3.1 Mezinárodní soutěž NIST

Vzhledem k hrozbám, které přinášejí kvantové počítače, organizace NIST provedla mezinárodní proces výběru postkvantových kryptografických algoritmů. V roce 2022 oznámila čtyři vybrané algoritmy. [20]

Postkvantové algoritmy jsou navrženy tak, aby odolaly útokům kvantových počítačů, a většinou se zakládají na složitosti jiných matematických problémů než současné asymetrické algoritmy. Vybrané algoritmy NIST zahrnují:

CRYSTALS-Kyber

Jedná se o algoritmus pro šifrování a výměnu klíčů. Je založený na mřížkové kryptografii, a díky tomu poskytuje silné teoretické zabezpečení. Kyber nabízí rychlou generaci klíčů, zapouzdření i dekapsulaci v softwaru, hardwaru i hybridních nastavení. [20]

CRYSTALS-Dilithium

Dilithium, využívající mřížkovou kryptografii, je podpisové schéma s vysokou účinností, relativně jednoduchou implementací a silným teoretickým základem. Jedná se o vynikající volbu pro širokou škálu kryptografických aplikací. Proto byl NIST vybrán jako primární podpisový algoritmus pro standardizaci. [20]

FALCON

Za předpokladu správné implementace, má NIST důvěru v bezpečnost algoritmu. Přestože podepisování a generování klíčů je poněkud pomalejší oproti například algoritmu Dilithium, vzhledem k rychlosti ověřování a velikosti klíčů může být lepší volbou v některých scénářích. [20]

SPHINCS+

Algoritmus založený na hashovacích funkcích, využívá hashovací stromy pro konstrukci digitálního podpisu. Jeho komplexnost může být potenciálním problémem pro bezpečnosti v implementaci. SPHINCS+ byl vybrán pro standardizaci, protože poskytuje funkční, i když poměrně velké a pomalé podpisové schéma. Jeho bezpečnost se zdá poměrně solidní a je založen na zcela odlišném souboru předpokladů než ostatní standardizované podpisové schémata.

Přestože bylo vybráno více algoritmů, z výše uvedených algoritmů je doporučeno používat algoritmus CRYSTALS-Dilithium. Pro budoucí standardizaci organizace zvažuje i další algoritmy, mezi které patří například Classic McEliece nebo BIKE. [20]

2.3.2 První standardy postkvantového šifrování

V roce 2024 vydala organizace NIST první 3 finalizované standardy postkvantového šifrování. Jedná se o algoritmy označené jako FIPS (Federal Information Processing Standard). [21]

FIPS 203

Jedná se o standard určený pro výměnu klíčů. Jeho základem je algoritmus CRYSTALS-Kyber, přejmenován na ML-KEM (Module-Lattice-Based Key-Encapsulation Mechanism). Mezi jeho výhody patří rychlost, malé klíče a snadná výměn klíčů mezi stranami. [21]

FIPS 204

Určen jako hlavní standard pro ochranu digitálních podpisů. Standard používá algoritmus CRYSTALS-Dilithium, který byl přejmenován na ML-DSA, (Module-Lattice-Based Digital Signature Algorithm). [21]

FIPS 205

Také zaměřený na digitální podpisy. Standard využívá algoritmus Sphincs+, přejmenovaný na SLH-DSA, (Stateless Hash-Based Digital Signature Algorithm). Standard je založen na jiném matematickém přístupu než ML-DSA, nepoužívá mřížkové struktury, ale hašovací stromy. Je zamýšlen jako záložní metoda pro případ, že se ML-DSA ukáže jako zranitelný. [21]

3 Experimentální část

V této kapitole se budeme zabývat praktickým porovnáním jednotlivých šifrovacích algoritmů, a to z hlediska časové náročnosti.

K měření rychlosti jednotlivých algoritmů máme připravené dva vlastní programy napsané v programovacím jazyce C s využitím knihovny Liboqs. První program se soustředí na podpisové algoritmy, konkrétně měříme čas potřebný k podepsání a čas potřebný k ověření podpisu. V tomto testu byly vybrány algoritmy RSA s různými délkami klíče, ECDSA a dále Falcon, Dilithium a SPHINCS+.

Náš druhý připravený program se zaměřuje na měření času potřebnému k zašifrování a dešifrování. V tomto měření se budeme soustředit na algoritmy Kyber, McEliece, FrodoKEM, RSA, X25519 a ECC.

3.1 Příprava prostředí

Nejdůležitějším pro naše testy je stažení knihovny Liboqs. Jedná se o open source knihovnu psanou v jazyce C a je součástí projektu Open Quantum Safe, jehož cílem je vyvinout a testovat kvantově bezpečnou kryptografii v reálných kontextech. [22]

Korektní instalaci knihovny je možné ověřit spuštěním testů pomocí příkazu *ninja run_tests* nebo zkompilováním a spuštěním příkladů z *liboqs/tests*.

Pro kompilaci připravených programů použijeme následující příkaz:
gcc -o nazev nazev.c -I/usr/local/include /usr/local/lib/liboqs.a -lssl -lcrypto -lm .

3.2 Měření podpisových algoritmů

```
RSA-2048 (opakování 1000)
Součet podpis:   0.626642 s
Součet ověření: 0.028879 s

RSA-7680 (opakování 1000)
Součet podpis:   62.697122 s
Součet ověření: 0.398602 s

RSA-15380 (opakování 1000)
Součet podpis:   489.185911 s
Součet ověření: 2.089958 s

ECDSA-P256 (opakování 1000)
Součet podpis:   0.062827 s
Součet ověření: 0.192792 s

Falcon-512 (opakování 1000)
Součet podpis:   0.425777 s
Součet ověření: 0.086707 s

Dilithium2 (opakování 1000)
Součet podpis:   0.162257 s
Součet ověření: 0.064905 s

SPHINCS+-SHA2-128f-simple (opakování 1000)
Součet podpis:   11.294559 s
Součet ověření: 1.063333 s
```

Obrázek 1: Výpis měření podpisových algoritmů (zdroj – vlastní)

Program napsaný v C porovnává podpisové algoritmy – klasické RSA, ECDSA a postkvantové Falcon, Dilithium a SPHINCS+. U každého algoritmu měří čas potřebný k vytvoření a ověření.

K samotnému měření používáme knihovnu <time.h>.

Program se skládá z několika funkcí.

Funkce void test_rsa_with_keylen(int bits) generuje klíč, vytvoří podpis zprávy pomocí RSA_sign a ověří podpis pomocí RSA_verify.

void test_rsa_with_keylen(int bits) {

```
RSA *rsa = RSA_generate_key(bits, RSA_F4, NULL, NULL);

unsigned char message[MESSAGE_LEN] = MSG;

unsigned char signature[2048];
```

```
    unsigned int siglen;
    double total_sign = 0.0, total_verify = 0.0;
    for (int i = 0; i < REPEAT; i++) {
        double start = get_time();
        RSA_sign(NID_sha256, message, MESSAGE_LEN, signature, &siglen, rsa);
        double end = get_time();
        total_sign += (end - start);
        start = get_time();
        RSA_verify(NID_sha256, message, MESSAGE_LEN, signature, siglen, rsa);
        end = get_time();
        total_verify += (end - start);
    }
…
}
```

Funkce void test_ecdsa() generuje klíč s křivkou P-256, podepíše a ověří zprávu pomocí ECDSA_sign a ECDS_verify.

```
void test_ecdsa() {
    EC_KEY *key = EC_KEY_new_by_curve_name(NID_X9_62_prime256v1);
    EC_KEY_generate_key(key);
    unsigned char message[MESSAGE_LEN] = MSG;
    unsigned char signature[256];
    unsigned int siglen;
    double total_sign = 0.0, total_verify = 0.0;
    for (int i = 0; i < REPEAT; i++) {
        double start = get_time();
```

```
        ECDSA_sign(0, message, MESSAGE_LEN, signature, &siglen, key);
        double end = get_time();
        total_sign += (end - start);
        start = get_time();
        ECDSA_verify(0, message, MESSAGE_LEN, signature, siglen, key);
        end = get_time();
        total_verify += (end - start);
    }
…
}
```

Funkce test_pq_signature(const char *alg_name) je univerzální funkcí, která testuje libovolný postkvantový podpisový algoritmus z knihovny liboqs. Vytvoří klíčový pár pomocí OQS_SIG_keypair, podepíše zprávu pomocí OQS_SIG_sign a ověří podpis pomocí OQS_SIG_verify).

```
void test_pq_signature(const char *alg_name) {
    OQS_SIG *sig = OQS_SIG_new(alg_name);
    if (sig == NULL) {
        fprintf(stderr, "%s není dostupný.\n", alg_name);
        return;
    }
    uint8_t public_key[sig->length_public_key];
    uint8_t secret_key[sig->length_secret_key];
    uint8_t message[MESSAGE_LEN] = MSG;
    uint8_t signature[sig->length_signature];
    size_t sig_len;
```

```
    double total_sign = 0.0, total_verify = 0.0;
    OQS_SIG_keypair(sig, public_key, secret_key);
    for (int i = 0; i < REPEAT; i++) {
        double start = get_time();
        if (OQS_SIG_sign(sig, signature, &sig_len, message, MESSAGE_LEN, secret_key) !=
OQS_SUCCESS) {
            fprintf(stderr, "Chyba při podpisu %s\n", alg_name);
            OQS_SIG_free(sig);
            return;
        }
        double end = get_time();
        total_sign += (end - start);
        start = get_time();
        if (OQS_SIG_verify(sig, message, MESSAGE_LEN, signature, sig_len, public_key) !=
OQS_SUCCESS) {
            fprintf(stderr, "Chyba při ověření %s\n", alg_name);
            OQS_SIG_free(sig);
            return;
        }
        end = get_time();
        total_verify += (end - start);
    }
…
}
```

Ve funkci main zavoláme pomocné funkce a do volání funkcí pro testování postkvantových algoritmů dosadíme algoritmy falcon_512, dilithium_2 a sphincs_sha2_128f_simple.

```
int main() {
    test_rsa_with_keylen(2048);
    test_rsa_with_keylen(7680);
    test_rsa_with_keylen(15380);
    test_ecdsa();
    test_pq_signature(OQS_SIG_alg_falcon_512);
    test_pq_signature(OQS_SIG_alg_dilithium_2);
    test_pq_signature(OQS_SIG_alg_sphincs_sha2_128f_simple);
    return 0;
}
```

3.2.1 Výsledky měření časové náročnosti

Testy prvního měření probíhají ve VirtualBoxu na systému Linux Mint 64-bit, 8GB RAM, procesor Intel Core i7, povoleno 1 jádro.

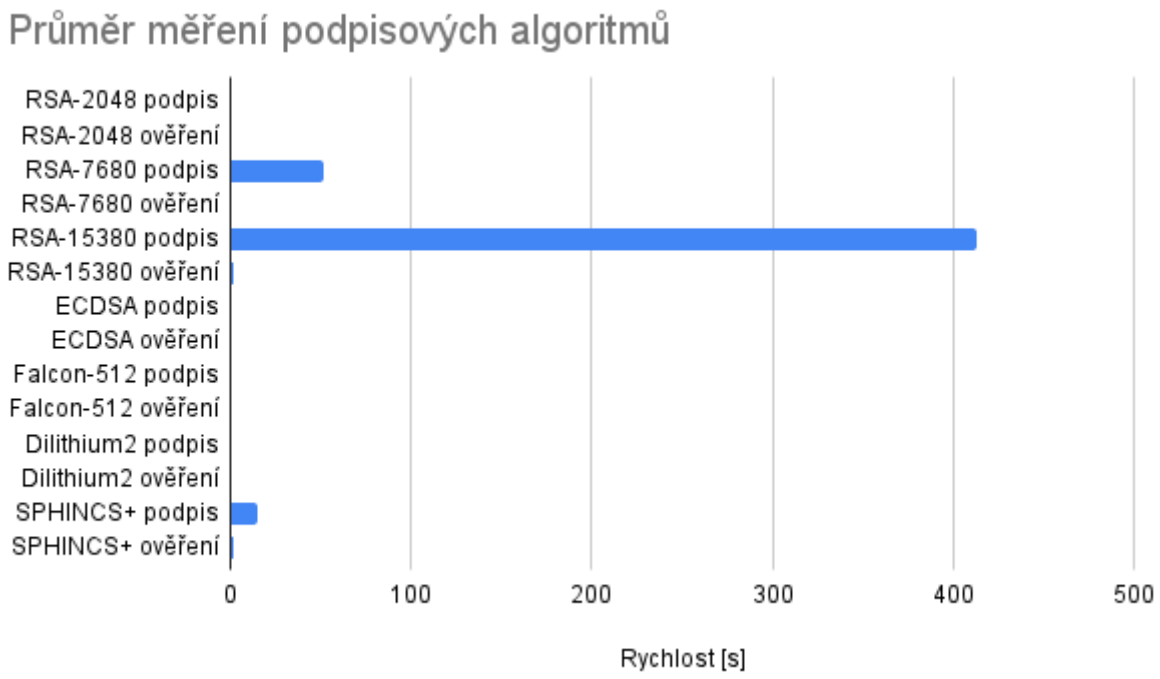
U každého algoritmu je v sekundách pomocí funkce get_time měřen čas potřebný k podepsání a čas potřebný k ověření podpisu. Každé měření obsahuje součet hodnot v cyklu o velikosti tisíc, tedy výsledná hodnota je dána součtem tisíců podpisů nebo součtem tisíců ověření. Poslední sloupce tabulky obsahují průměrné hodnoty měření a směrodatnou odchylku.

Tabulka 1: Průběh měření podpisových algoritmů (zdroj – vlastní)

Druh algoritmu	Součet 1. měření	Součet 2. měření	Součet 3. měření	Součet 4. měření	Součet 5. měření	Průměr měření	Směrodatná odchylka
RSA-2048 podpis	0,65631	0,64442	0,65241	1,04504	0,52275	0,70419	0,17757

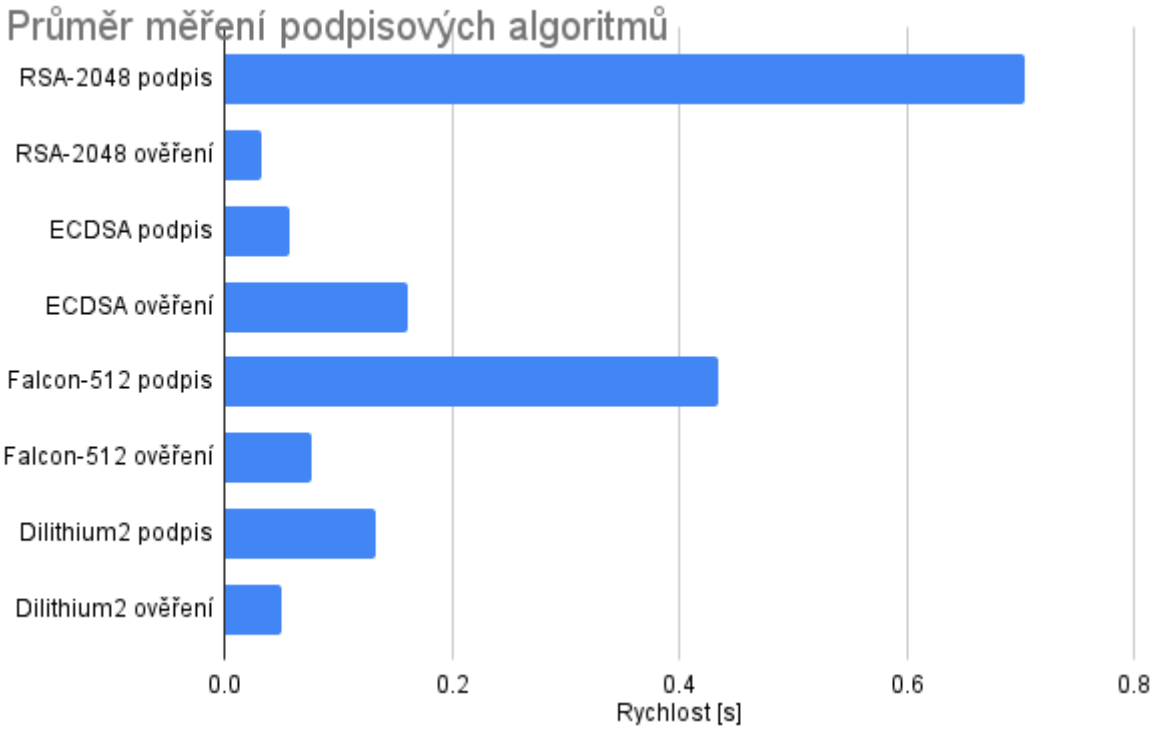
RSA-2048 ověření	0,0329	0,03033	0,03087	0,04253	0,02418	0,03216	0,00595
RSA-7680 podpis	51,05754	54,05551	49,68001	49,80048	50,82182	51,08307	1,58227
RSA-7680 ověření	0,32303	0,34643	0,31388	0,31389	0,32078	0,3236	0,01199
RSA-15380 podpis	318,6292	421,9467	454,3687	433,5972	438,8323	413,4748	48,5544
RSA-15380 ověření	1,34855	1,78396	1,91321	1,82712	1,87864	1,7503	0,20565
ECDSA podpis	0,04888	0,05413	0,05467	0,07841	0,04705	0,05663	0,01128
ECDSA ověření	0,15776	0,14871	0,15072	0,20858	0,13637	0,16043	0,02504
Falcon-512 podpis	0,40752	0,37134	0,41945	0,64195	0,32685	0,43342	0,10915
Falcon-512 ověření	0,07907	0,0737	0,08361	0,08162	0,06486	0,07657	0,00673
Dilithium2 podpis	0,13965	0,12356	0,11358	0,17917	0,10742	0,13268	0,02568
Dilithium2 ověření	0,04514	0,05065	0,04048	0,06795	0,03988	0,04882	0,01032
SPHINCS+ podpis	11,95938	17,63632	18,1889	14,03781	11,19696	14,60388	2,86248
SPHINCS+ ověření	1,12332	1,66763	1,70658	1,31683	1,0649	1,37585	0,26775

Výsledky měření v grafu:



Obrázek 2: Průběh měření podpisových algoritmů (zdroj – vlastní)

Z důvodu vysokých hodnot některých algoritmů přidáváme ještě jeden graf s totožnými hodnotami, ale již bez RSA-7680, RSA-15380 a SPHINCS+.



Obrázek 3: Průběh měření podpisových algoritmů, detail (zdroj – vlastní)

Z grafů vidíme, že u RSA-15380 a RSA-7680 ze všech testovaných algoritmů je nejdelší čas potřebný k podpisu a to mnohonásobně. Na druhém místě času potřebného k podpisu se nachází RSA. Naopak nejnižší hodnoty potřebné k podpisu byly naměřené u ECDSA.

Nejvyšší čas potřebný pro ověření opět vidíme u algoritmu RSA-15380. Zajímavé je, že přestože ECDSA měl nejkratší čas potřebný pro podpis, u měření ověření skončil na třetím místě. Nejkratší čas potřebný k ověření byl naměřen u RSA-2048, i když Dilithium2 skončil s podobnými hodnotami.

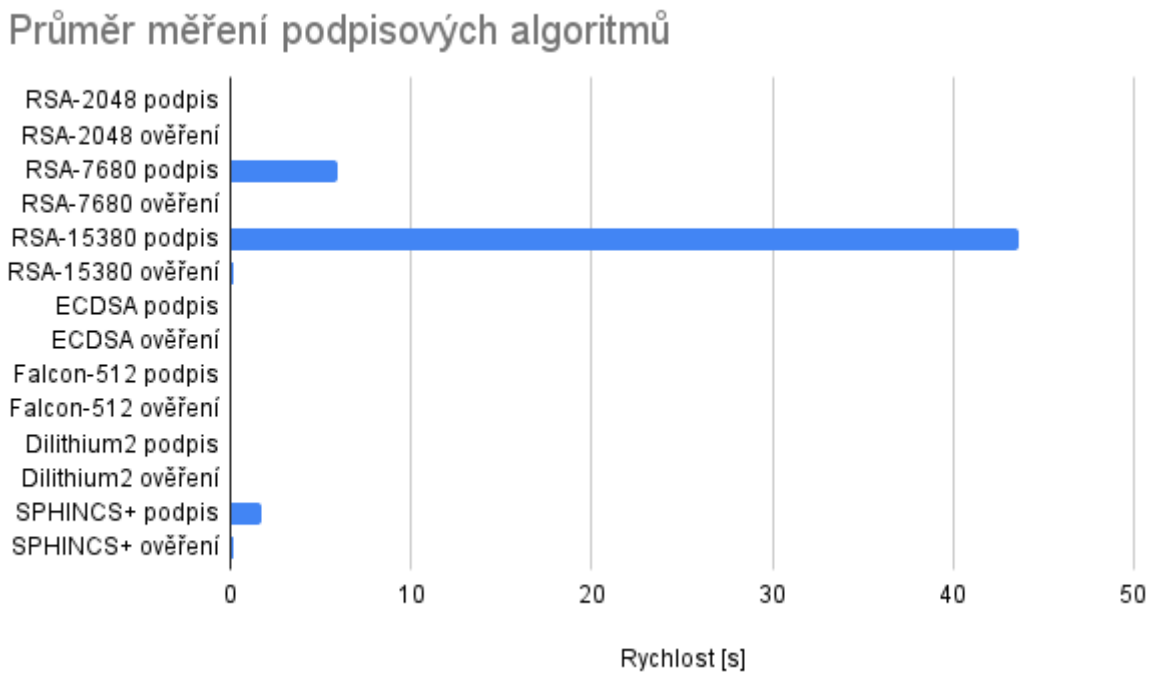
Testy druhého měření provedené na stolním počítači s operačním systémem Linux Mint 64-bit, 16GB RAM, procesor AMD FX-6300, 6 jader.

Tabulka 2: Průběh měření podpisových algoritmů (zdroj – vlastní)

Druh algoritmu	Součet 1. měření	Součet 2. měření	Součet 3. měření	Součet 4. měření	Součet 5. měření	Průměr měření	Směrodatná odchylka
RSA-2048 podpis	0,1045	0,08204	0,0716	0,07292	0,08055	0,08232	0,01182
RSA-2048 ověření	0,00425	0,00411	0,00337	0,00366	0,00379	0,00384	0,00032
RSA-7680 podpis	4,98005	6,38219	6,00617	5,67306	6,75694	5,95968	0,60961
RSA-7680 ověření	0,03139	0,04038	0,03849	0,03589	0,0433	0,03789	0,00405
RSA-15380 podpis	43,35972	39,82865	46,88297	35,40324	52,74334	43,64358	5,93021
RSA-15380 ověření	0,18271	0,16857	0,19822	0,14984	0,22299	0,18447	0,02501
ECDSA podpis	0,00784	0,00611	0,00601	0,00543	0,00677	0,00643	0,00082
ECDSA ověření	0,02086	0,01972	0,01652	0,01753	0,01859	0,01864	0,00154
Falcon-512 podpis	0,0642	0,05094	0,04126	0,04528	0,04642	0,04962	0,00791

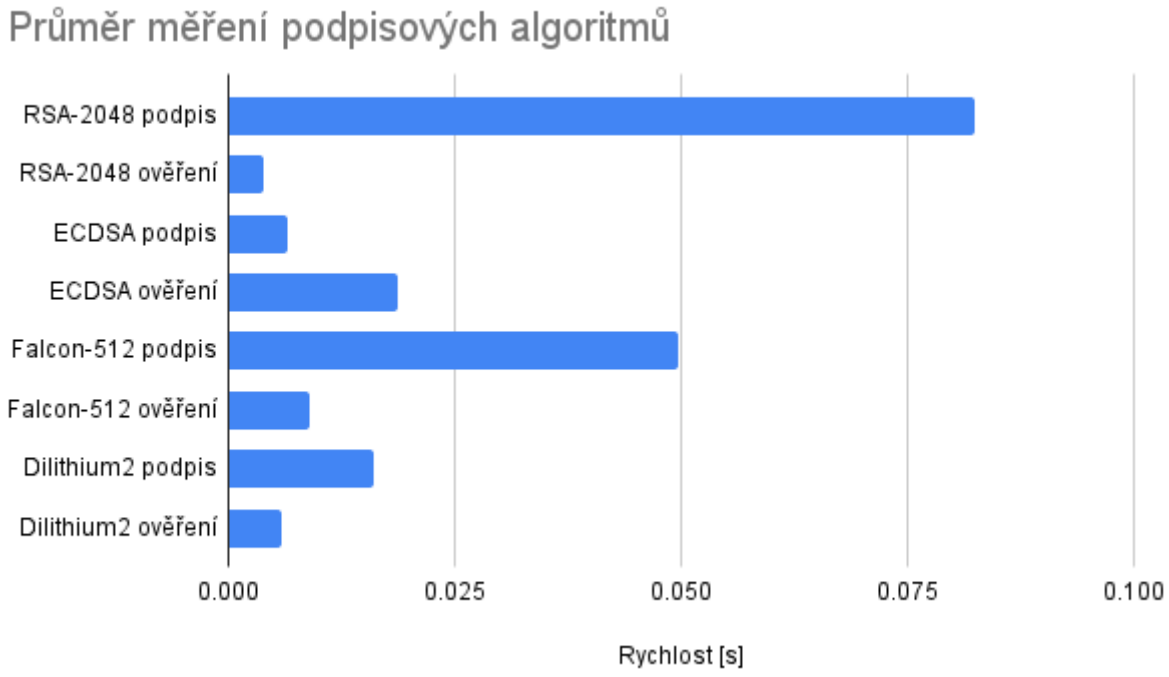
Falcon-512 ověření	0,00816	0,00988	0,00819	0,00879	0,00921	0,00885	0,00065
Dilithium2 podpis	0,01792	0,01746	0,01373	0,01552	0,01545	0,01601	0,00152
Dilithium2 ověření	0,00679	0,00564	0,00563	0,00502	0,00633	0,00588	0,00062
SPHINCS+ podpis	1,40378	1,49492	1,95959	1,32882	2,20454	1,67833	0,34269
SPHINCS+ ověření	0,13168	0,14042	0,18529	0,12481	0,20845	0,15813	0,03284

Výsledky měření v grafu:



Obrázek 4: Průběh měření podpisových algoritmů (zdroj – vlastní)

Z důvodu vysokých hodnot u některých algoritmů opět přidáváme další graf bez RSA-15380, RSA-7680 a SPHINCS+.



Obrázek 5: Průběh měření podpisových algoritmů, detail (zdroj – vlastní)

Z předchozích grafů vidíme poměrově velmi podobné hodnoty, jako u předchozího měření. U RSA-15380 a RSA-7680 ze všech testovaných algoritmů je nejdelší čas potřebný k podpisu a to mnohonásobně. Na druhém místě času potřebného k podpisu se nachází RSA. Naopak nejnižší hodnoty potřebné k podpisu byly naměřené u ECDSA.

Nejvyšší čas potřebný pro ověření opět vidíme u algoritmu RSA-15380. ECDSA u měření ověření podpisu skončil na třetím místě. Nejkratší čas potřebný k ověření byl naměřen u RSA-2048, i když Dilithium2 skončil s podobnými hodnotami.

3.3 Měření šifrovacích algoritmů

```
Kyber512 (opakování 1000)
Součet encapsulace: 0.020849 s
Součet decapsulace: 0.012471 s

Kyber768 (opakování 1000)
Součet encapsulace: 0.050913 s
Součet decapsulace: 0.032596 s

Kyber1024 (opakování 1000)
Součet encapsulace: 0.060910 s
Součet decapsulace: 0.040254 s

Classic-McEliece-348864 (opakování 1000)
Součet encapsulace: 0.029423 s
Součet decapsulace: 0.071723 s

FrodoKEM-640-AES (opakování 1000)
Součet encapsulace: 0.649182 s
Součet decapsulace: 0.611212 s

RSA-2048 (opakování 1000)
Součet šifrování: 0.052780 s
Součet dešifrování: 0.666981 s

RSA-3072 (opakování 1000)
Součet šifrování: 0.122007 s
Součet dešifrování: 3.791936 s

X25519 (opakování 1000)
Součet derivace 1: 0.094678 s
Součet derivace 2: 0.099244 s

ECC P-256 (opakování 1000)
Součet derivace 1: 0.213710 s
Součet derivace 2: 0.218367 s
```

Obrázek 6: Výpis měření šifrovacích algoritmů (zdroj – vlastní)

Program napsaný v C porovnává kryptografické algoritmy, klasické RSA a postkvantové Kyber, McEliece, FrodoKEM a dále X25519 a ECC. Měří dobu potřebnou k provedení klíčových operací.

K samotnému měření používáme knihovnu <time.h>.

Program se skládá z několika funkcí.

Funkce void test_kem(const char *alg_name) testuje postkvantové algoritmy z knihovny liboqs. Nejprve vygeneruje pár klíčů, opakovaně provede enkapsulaci (získá šifrovaný klíč) a dekapsulaci (obnoví původní sdílený klíč).

```
void test_kem(const char *alg_name) {
    OQS_KEM *kem = OQS_KEM_new(alg_name);
    if (kem == NULL) {
        fprintf(stderr, "%s není podporován.\n", alg_name);
        return;
    }
    uint8_t public_key[kem->length_public_key];
    uint8_t secret_key[kem->length_secret_key];
    uint8_t ciphertext[kem->length_ciphertext];
    uint8_t shared_secret_enc[kem->length_shared_secret];
    uint8_t shared_secret_dec[kem->length_shared_secret];
    double total_enc_time = 0.0, total_dec_time = 0.0;
    OQS_KEM_keypair(kem, public_key, secret_key);
    for (int i = 0; i < REPEAT; i++) {
        double start = get_time();
        OQS_KEM_encaps(kem, ciphertext, shared_secret_enc, public_key);
        double end = get_time();
        total_enc_time += (end - start);
        start = get_time();
        OQS_KEM_decaps(kem, shared_secret_dec, ciphertext, secret_key);
        end = get_time();
        total_dec_time += (end - start);
    }
    …
```

```
}
```

Funkce void test_rsa_2048() a void test_rsa_3072() provádějí šifrování a dešifrování s klíči o délce 2048 a 3072 bitů.

```
void test_rsa_2048() {
    RSA *rsa = RSA_new();
    BIGNUM *bn = BN_new();
    BN_set_word(bn, RSA_F4);
    RSA_generate_key_ex(rsa, 2048, bn, NULL);
    unsigned char plaintext[256] = MSG;
    unsigned char encrypted[256];
    unsigned char decrypted[256];
    double total_enc = 0.0, total_dec = 0.0;
    for (int i = 0; i < REPEAT; i++) {
        double start = get_time();
        int len = RSA_public_encrypt(strlen((char *)plaintext), plaintext, encrypted, rsa,
RSA_PKCS1_OAEP_PADDING);
        double end = get_time();
        total_enc += (end - start);
        start = get_time();
        RSA_private_decrypt(len, encrypted, decrypted, rsa, RSA_PKCS1_OAEP_PADDING);
        end = get_time();
        total_dec += (end - start);
    }
…
}
```

Funkce test_x25519() testuje derivaci společného tajemství pomocí křivky X25519, každá strana vygeneruje dva klíče, následně každá strana odvodí sdílený klíč z veřejného klíče druhé strany. Měří se časy obou operací.

```
test_x25519() {
    double total1 = 0.0, total2 = 0.0;
    for (int i = 0; i < REPEAT; i++) {
        EVP_PKEY_CTX *ctx;
        EVP_PKEY *pkey1 = NULL, *pkey2 = NULL;
        unsigned char secret1[32], secret2[32];
        size_t secret_len = 32;
        ctx = EVP_PKEY_CTX_new_id(EVP_PKEY_X25519, NULL);
        EVP_PKEY_keygen_init(ctx);
        EVP_PKEY_keygen(ctx, &pkey1);
        EVP_PKEY_keygen(ctx, &pkey2);
        EVP_PKEY_CTX_free(ctx);
        double start = get_time();
        ctx = EVP_PKEY_CTX_new(pkey1, NULL);
        EVP_PKEY_derive_init(ctx);
        EVP_PKEY_derive_set_peer(ctx, pkey2);
        EVP_PKEY_derive(ctx, secret1, &secret_len);
        double end = get_time();
        total1 += (end - start);
        EVP_PKEY_CTX_free(ctx);
        start = get_time();
        ctx = EVP_PKEY_CTX_new(pkey2, NULL);
```

```
        EVP_PKEY_derive_init(ctx);
        EVP_PKEY_derive_set_peer(ctx, pkey1);
        EVP_PKEY_derive(ctx, secret2, &secret_len);
        end = get_time();
        total2 += (end - start);
        EVP_PKEY_CTX_free(ctx);
        EVP_PKEY_free(pkey1);
        EVP_PKEY_free(pkey2);
    }
…
}
```

Ve funkci main inicializuje knihovnu liboqs, spustí testy vybraných algoritmů a následně knihovnu ukončí. .

```
OQS_init();
test_kem(OQS_KEM_alg_kyber_512);
test_kem(OQS_KEM_alg_kyber_768);
test_kem(OQS_KEM_alg_kyber_1024);
test_kem(OQS_KEM_alg_classic_mceliece_348864);
test_kem(OQS_KEM_alg_frodokem_640_aes);
test_rsa_2048();
test_rsa_3072();
test_x25519();
test_ecc_p256();
OQS_destroy();
```

3.3.1 Výsledky měření časové náročnosti

Stejně jako u měření podpisových algoritmů, i zde u je každého algoritmu v sekundách pomocí funkce get_time měřen čas potřebný k podepsání a čas potřebný k ověření podpisu. Každé měření obsahuje součet hodnot v cyklu o velikosti tisíc, tedy výsledná hodnota je dána součtem tisíců podpisů nebo součtem tisíců ověření. Poslední sloupce tabulky obsahují průměrné hodnoty měření a směrodatnou odchylku.

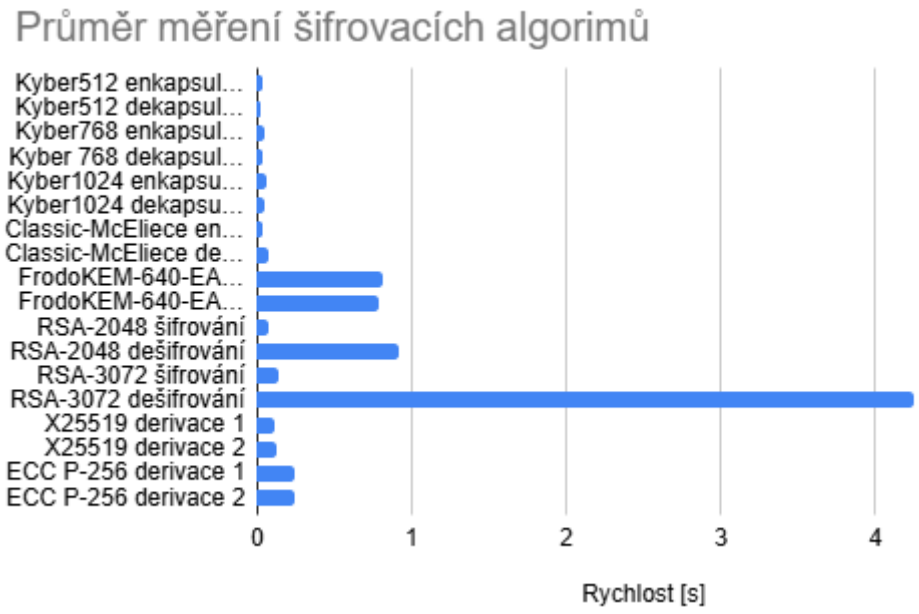
Testy prvního měření probíhají ve VirtualBoxu na systému Linux Mint 64-bit, 8GB RAM, procesor Intel Core i7, povoleno 1 jádro.

Tabulka 3: Průběh měření šifrovacích algoritmů (zdroj – vlastní)

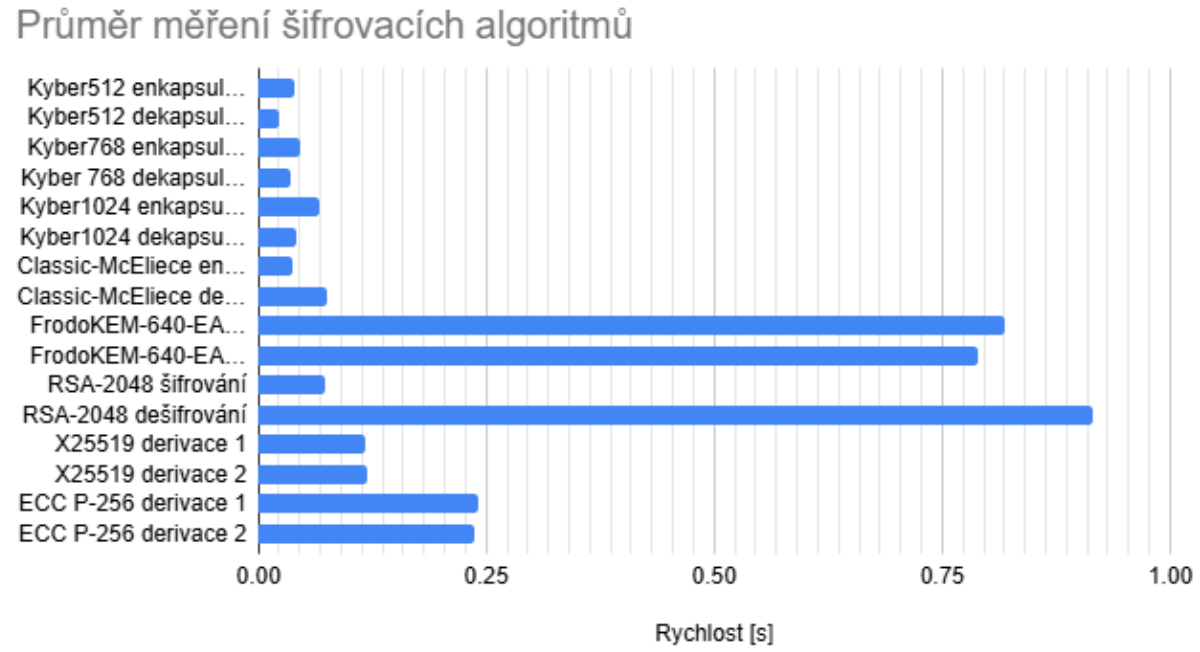
Druh algoritmu	Součet 1. měření	Součet 2. měření	Součet 3. měření	Součet 4. měření	Součet 5. měření	Průměr měření	Směrodatná odchylka
Kyber512 enkapsulace	0,03429	0,03336	0,03961	0,05095	0,03642	0,03892	0,00638
Kyber512 dekapsulace	0,0093	0,02929	0,02437	0,0228	0,02598	0,02235	0,00687
Kyber768 enkapsulace	0,03334	0,04752	0,05313	0,0513	0,04102	0,04526	0,00726
Kyber 768 dekapsulace	0,02301	0,04085	0,03342	0,03893	0,04252	0,03575	0,00707
Kyber1024 enkapsulace	0,05704	0,07204	0,0755	0,07177	0,05523	0,06631	0,00844
Kyber1024 dekapsulace	0,03286	0,04728	0,03967	0,04707	0,04658	0,04269	0,00568
Classic-McEliece enkapsulace	0,02635	0,04592	0,03256	0,05376	0,02487	0,03669	0,01132
Classic-McEliece dekapsulace	0,05927	0,08679	0,06901	0,11245	0,05275	0,07605	0,02151

FrodoKEM-640-EAS enkapsulace	0,72107	0,69403	0,77271	1,25994	0,63993	0,81753	0,22531
FrodoKEM-640-EAS dekapsulace	0,74214	0,64595	0,72312	1,22475	0,61032	0,78926	0,22307
RSA-2048 šifrování	0,09263	0,05769	0,05758	0,10592	0,05	0,07277	0,02223
RSA-2048 dešifrování	1,04067	0,71119	0,72583	1,44493	0,65596	0,91572	0,297
RSA-3072 šifrování	0,15514	0,10961	0,10273	0,21454	0,09231	0,13486	0,04525
RSA-3072 dešifrování	5,12171	3,42706	3,26969	6,49524	2,94766	4,25227	1,35169
X25519 derivace 1	0,12963	0,10788	0,09938	0,16092	0,08488	0,11654	0,02651
X25519 derivace 2	0,12536	0,10985	0,09658	0,16667	0,09835	0,11936	0,02579
ECC P-256 derivace 1	0,33835	0,23474	0,23071	0,20782	0,19732	0,24179	0,05025
ECC P-256 derivace 2	0,31079	0,22717	0,22475	0,20955	0,21485	0,23742	0,03724

U asymetrických šifrovacích algoritmů uvádíme derivaci 1 a derivaci 2. U těchto algoritmů mají oba účastníci svůj vlastní soukromý klíč a veřejný klíč druhé strany. Pomocí svého soukromého klíče a veřejného klíče druhé strany odvodí stejný sdílený tajný klíč. Pod pojmem derivace tedy rozumíme odvození tohoto klíče.



Obrázek 7: Průměr měření podpisových algoritmů (zdroj – vlastní)



Obrázek 8: Průměr měření podpisových algoritmů, detail (zdroj – vlastní)

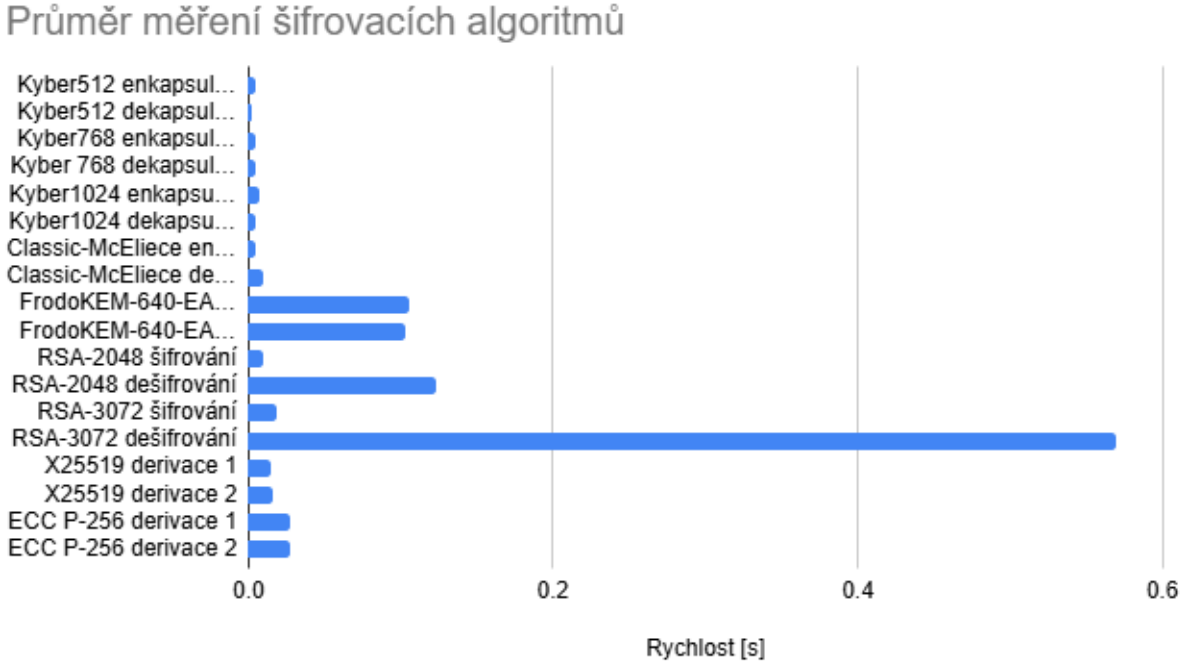
Z tabulek vidíme, že nejpomalejší v operaci zašifrování je postkvantový algoritmus FrodoKEM, který má zároveň podobně rychlou i operaci dešifrování. Ještě déle trvala operace dešifrování u klasického RSA-2048. Nejrychlejší zašifrování a zároveň i dešifrování bylo naměřeno u algoritmu Kyber512.

Testy druhého měření provedené na stolním počítači s operačním systémem Linux Mint 64-bit, 16GB RAM, procesor AMD FX-6300, 6 jader.

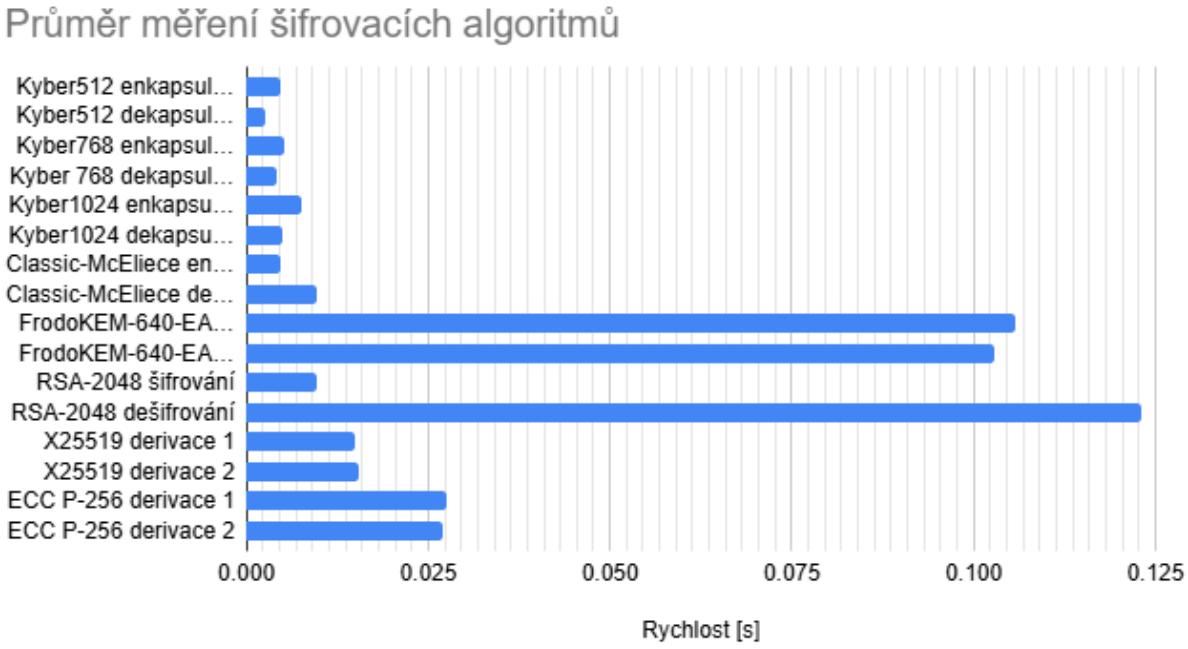
Tabulka 4: Průběh měření šifrovacích algoritmů (zdroj – vlastní)

Druh algoritmu	Součet 1. měření	Součet 2. měření	Součet 3. měření	Součet 4. měření	Součet 5. měření	Průměr měření	Směrodatná odchylka
Kyber512 enkapsulace	0,00429	0,00566	0,00371	0,00364	0,00637	0,00473	0,00122
Kyber512 dekapsulace	0,00116	0,00253	0,00325	0,0026	0,00285	0,00248	0,00079
Kyber768 enkapsulace	0,00417	0,0057	0,00528	0,0041	0,00641	0,00513	0,001
Kyber 768 dekapsulace	0,00288	0,00433	0,00454	0,00425	0,00487	0,00417	0,00076
Kyber1024 enkapsulace	0,00713	0,00797	0,008	0,00552	0,00897	0,00752	0,00129
Kyber1024 dekapsulace	0,00411	0,00523	0,00525	0,00466	0,00588	0,00503	0,00067
Classic-McEliece enkapsulace	0,00329	0,00597	0,0051	0,00249	0,00672	0,00472	0,00178
Classic-McEliece dekapsulace	0,00741	0,01249	0,00964	0,00527	0,01406	0,00978	0,00359
FrodoKEM-640-EAS enkapsulace	0,09013	0,13999	0,07711	0,06399	0,15749	0,10575	0,0408
FrodoKEM-640-EAS dekapsulace	0,09277	0,13608	0,07177	0,06103	0,15309	0,10295	0,04014
RSA-2048 šifrování	0,01158	0,01177	0,00641	0,005	0,01324	0,0096	0,00365

RSA-2048 dešifrování	0,13008	0,16055	0,07902	0,0656	0,18062	0,12317	0,05002
RSA-3072 šifrování	0,01939	0,02384	0,01218	0,00923	0,02682	0,01829	0,00749
RSA-3072 dešifrování	0,64021	0,72169	0,38078	0,29477	0,8119	0,56987	0,22249
X25519 derivace 1	0,0162	0,01788	0,01199	0,00849	0,02012	0,01493	0,00467
X25519 derivace 2	0,01567	0,01852	0,01221	0,00983	0,02083	0,01541	0,00449
ECC P-256 derivace 1	0,04229	0,02309	0,02608	0,01973	0,02598	0,02744	0,0087
ECC P-256 derivace 2	0,03885	0,02328	0,02524	0,02149	0,02619	0,02701	0,00686



Obrázek 9: Průměr měření podpisových algoritmů (zdroj – vlastní)



Obrázek 10: Průměr měření podpisových algoritmů, detail (zdroj – vlastní)

Z tabulek vidíme, že opět nastaly poměrově podobné výsledky, jako u měření na prvním zařízení. Nejpomalejší v operaci zašifrování je postkvantový algoritmus FrodoKEM, který má zároveň podobně rychlou i operaci dešifrování. Ještě déle trvala operace dešifrování u klasického RSA-2048. Nejrychlejší zašifrování a zároveň i dešifrování bylo naměřeno u algoritmu Kyber512.

ZÁVĚR

Cílem této práce bylo porovnat klasické a postkvantové šifrovací algoritmy. Nejprve byly v teoretické části představeny klasické algoritmy, jejich typy, výhody a případné nedostatky. Dále byly představeny postkvantové algoritmy a jejich dopady na klasické algoritmy.

V praktické části se práce zaměřila na měření šifrovacích schémat na dvou různých zařízeních. Ve výsledcích měření se ukázalo, že již dnes mohou být postkvantové algoritmy v některých situacích výkonnější, než ty klasické. Například při měření rychlosti podpisů se ukázal jako vítěz klasický algoritmus ECDSA, ve výsledcích měření ověření podpisů však byly rychlejší algoritmy Falcon a Dilithium. V druhém měření byl jako nejrychlejší algoritmus pro šifrování i dešifrování změřen postkvantový Kyber512.

POUŽITÁ LITERATURA

[1] Introduction to Modern Cryptography [Online]. 2005 [cit. 2025-05-12]. Dostupné z: https://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf

[2] The Dark Side of Symmetric Key Encryption: A Comprehensive Guide. Newsoftwares [online]. 2023 [cit. 2025-05-12]. Dostupné z: https://www.newsoftwares.net/blog/dark-side-of-symmetric-key-encryption/#Encryption_Methods_That_Uses_A_Single_Key_To_Encrypt_And_Decrypt_Data

[3] Data Encryption: Symmetric Cryptography and Brute Force Attack. Coreapp [online]. 2023 [cit. 2025-05-12]. Dostupné z: https://coreapp.cz/en/blog/article-broteforce-attack

[4] Introduction to Cryptography: Principles and Applications. Second Edition. Springer, 2007. ISBN 9783540492436.

[5] What is a block cipher? TechTarget [online]. 2024 [cit. 2025-04-18]. Dostupné z: https://www.techtarget.com/searchsecurity/definition/block-cipher

[6] Jaké jsou provozní režimy? EITCA [online]. 2024 [cit. 2025-05-12]. Dostupné z: https://cs.eitca.org/cybersecurity/eitc-is-ccf-classical-cryptography-fundamentals/applications-of-block-ciphers/modes-of-operation-for-block-ciphers/what-are-modes-of-operation/

[7] What is a stream cipher? TechTarget [online]. 2024 [cit. 2025-04-18]. Dostupné z: https://www.techtarget.com/searchsecurity/definition/stream-cipher

[8] Kryptografie okolo nás. CZ.NIC, 2019. ISBN 978-80-88168-52-2.

[9] RSA Algorithm in Cryptography. GeeksForGeeks [online]. 2025 [cit. 2025-04-18]. Dostupné z: https://www.geeksforgeeks.org/rsa-algorithm-cryptography/

[10] Should We Start Using 4096 bit RSA keys? [online]. [cit. 2025-05-11]. Dostupné z: https://www.jscape.com/blog/should-i-start-using-4096-bit-rsa-keys

[11] Cryptographic Hash Functions. ResearchGate [online]. 2021 [cit. 2025-04-19]. Dostupné z: https://www.researchgate.net/publication/351837904_Cryptographic_Hash_Functions

[12] Cryptographic hash functions. IBM Quantim Learning [online]. 2017 [cit. 2025-04-19]. Dostupné z: https://learning.quantum.ibm.com/course/practical-introduction-to-quantum-safe-cryptography/cryptographic-hash-functions

[13] A Brief History of Quantum Computing. Quantumpedia [online]. 2025-04-02 [cit. 2025-03-31]. Dostupné z: https://quantumpedia.uk/a-brief-history-of-quantum-computing-e0bbd05893d0

[14] What Is Post-Quantum Cryptography? Nits.gov [online]. 2024 [cit. 2025-04-12]. Dostupné z: https://www.nist.gov/cybersecurity/what-post-quantum-cryptography

[15] Stateful Hash-Based Signature Schemes – Hidden Champions of Post Quantum Cryptography. Ultimaco [online]. [2023] [cit. 2025-05-11]. Dostupné z: https://utimaco.com/news/blog-posts/stateful-hash-based-signature-schemes-hidden-champions-pqc

[16] What is Hash-based Cryptography? Ultimaco [online]. [2023] [cit. 2025-05-11]. Dostupné z: https://utimaco.com/service/knowledge-base/post-quantum-cryptography/what-hash-based-cryptography

[17] What is Lattice-based Cryptography? Ultimaco [online]. [2023] [cit. 2025-05-11]. Dostupné z: https://utimaco.com/service/knowledge-base/post-quantum-cryptography/what-lattice-based-cryptography

[18] What is Code-based Cryptography? Ultimaco [online]. [2023] [cit. 2025-05-11]. Dostupné z: https://utimaco.com/service/knowledge-base/post-quantum-cryptography/what-code-based-cryptography

[19] About NIST. Nist.gov [online]. 2009 [cit. 2025-04-13]. Dostupné z: https://www.nist.gov/about-nist

[20] Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process. Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process. 2022, **2022**(1), 4-48. Dostupné také z: https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413.pdf

[21] NIST Releases First 3 Finalized Post-Quantum Encryption Standards. NIST [online]. 2024 [cit. 2025-03-31]. Dostupné z: https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards

[22] Liboqs. GitHub [online]. 2018 [cit. 2025-03-31]. Dostupné z: https://github.com/open-quantum-safe/liboqs

SEZNAM PŘÍLOH

PŘÍLOHA A*:* Zdrojové kódy pro měření

Tato příloha obsahuje zdrojové kódy pro měření algoritmů v archivu s názvem Mereni.zip.