Možnosti implementace kvantově odolných algoritmů

Filip Hajduch

Bakalářská práce
2025

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Filip Hajduch**
Osobní číslo:	**A22008**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Možnosti implementace kvantově odolných algoritmů**
Téma práce anglicky:	**Possibilities of Implementing Quantum-Resistant Algorithms**

Zásady pro vypracování

1. Vypracujte literární rešerši na téma práce.
2. Popište algoritmy kvantově odolné kryptografie.
3. Rozeberte možnosti implementace kvantově odolných algoritmů v kontextu doporučení NÚKIB a NIST.
4. Popište implementaci X25519Kyber768.
5. Proveďte ukázkovou implementaci.
6. Vhodně prezentujte výsledky práce a rozeberte další alternativy.

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

Seznam doporučené literatury:

1. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. ČSN ISO 690:2022, *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. Federal Information Processing Standards Publication (FIPS) NIST FIPS 203. Department of Commerce, Washington, D.C., 2024. Dostupné také z: https://doi.org/10.6028/NIST.FIPS.203
2. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. ČSN ISO 690:2022, *Module-Lattice-Based Digital Signature Standard*. Federal Information Pro-cessing Standards Publication (FIPS) NIST FIPS 204. Department of Commerce, Washington, D.C., 2024. Dostupné také z: https://doi.org/10.6028/NIST.FIPS.204.
3. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. ČSN ISO 690:2022, *Stateless Hash-Based Digital Signature Standard*. Federal Information Processing Standards Publication (FIPS) NIST FIPS 205. Department of Commerce, Washington, D.C., 2024. Dostupné také z: https: //doi.org/10.6028/NIST.FIPS.205.
4. BERNSTEIN, Daniel J. a DAHMEN, Erik (ed.). *Post-Quantum Cryptography*. Online. Berlin, Heidelberg: Springer-Verlag, 2009. ISBN 978-3-540-88702-7. Dostupné z: https://doi.org/https://doi.org/10.1007/978-3-540-88702-7. [cit. 2024-10-18].
5. STALLINGS, William. *Cryptography and Network Security – Principles and Practice*. 7th Edition. Pearson Education India, 2017. ISBN 978-1-292-15858-7.
6. SINGH, Simon a KOUBSKÝ, Petr. *Kniha kódů a šifer. Tajná komunikace od starého Egypta po kvantovou kryptografii*. 4. vydání. Argo, 2022. ISBN 9788076750845.

Vedoucí bakalářské práce: **Ing. Petr Žáček, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **27. října 2024**
Termín odevzdání bakalářské práce: **2. června 2025**

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

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

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 17. listopadu 2024

PROHLÁŠENÍ AUTORA BAKALÁŘSKÉ PRÁCE

Beru na vědomí, že

- odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení bakalářské práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- 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;
- 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;
- podle § 60 odst. 2 a 3 mohu užít své dílo – bakalářskou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- 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(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.
- při tvorbě této práce jsem použil nástroj generativního modelu AI ChatGPT; https://chatgpt.com za účelem zpřesnění odborné terminologie a jazykové korektury textu. Po použití tohoto nástroje jsem provedl kontrolu obsahu a přebírám za něj plnou zodpovědnost.

Ve Zlíně, dne ....

podpis autora

ABSTRAKT

Tato bakalářská práce se zabývá problematikou postkvantové kryptografie v souvislosti s rozvojem kvantových počítačů a rostoucí potřebou zabezpečení digitální komunikace. Práce shrnuje aktuální stav, principy a standardizační snahy v oblasti kvantově odolných algoritmů dle doporučení organizací NIST a NÚKIB. Hlavní část je zaměřena na hybridní algoritmus X25519MLKEM768, jeho implementaci a testování v prostředí Python. Výsledky jsou porovnávány s dalšími přístupy z hlediska výkonnosti, bezpečnosti i datové režie. Práce poskytuje ucelený pohled na možnosti nasazení moderních kryptografických algoritmů a poukazuje na klíčové výhody i limity jejich využití v praxi.

Klíčová slova: postkvantová kryptografie, kvantově odolné algoritmy, X25519MLKEM768, NIST, NÚKIB, hybridní šifrování

ABSTRACT

This bachelor thesis focuses on post-quantum cryptography in connection with the development of quantum computers and the growing need for secure digital communication. The thesis summarizes the current state, principles, and standardization efforts related to quantum-resistant algorithms according to NIST and NÚKIB recommendations. The main part is devoted to the hybrid algorithm X25519MLKEM768, its implementation and testing in a Python environment. The results are compared with other approaches in terms of performance, security, and data overhead. The thesis provides a comprehensive overview of the deployment possibilities of modern cryptographic algorithms and highlights the main advantages and limitations of their practical use.

Keywords: post-quantum cryptography, quantum-resistant algorithms, X25519MLKEM768, NIST, NÚKIB, hybrid encryption

Rád bych touto cestou poděkoval panu Ing. Petru Žáčkovi, Ph.D., za odborné vedení, cenné rady, vstřícnost a podporu během zpracování této bakalářské práce. Jeho připomínky a podněty významně přispěly ke zkvalitnění výsledků i celkové úrovně práce.

Poděkování patří také mé rodině za trpělivost, podporu a povzbuzení v průběhu celého studia.

OBSAH

ÚVOD

S rozvojem kvantových počítačů čelí současné kryptografické systémy zásadní hrozbě. Kvantové počítače mají potenciál prolomit algoritmy, které jsou dnes považovány za bezpečné, a tím zásadně ovlivnit digitální bezpečnost napříč všemi obory. Výběr tohoto tématu byl motivován rostoucí aktuálností problematiky a potřebou reagovat na rychlý vývoj v oblasti postkvantové kryptografie.

V současnosti probíhá intenzivní výzkum i standardizace nových kryptografických algoritmů, zejména v rámci organizace NIST, a doporučení k jejich nasazení vydávají i národní autority jako NÚKIB. Přestože jsou některé nové algoritmy již doporučovány, otázka jejich efektivity, praktického nasazení a možných kompromisů zůstává stále otevřená.

Cílem této práce bylo analyzovat aktuální stav v oblasti postkvantové kryptografie, implementovat a otestovat hybridní algoritmus X25519MLKEM768 a posoudit jeho výhody a nevýhody v porovnání s dalšími přístupy. Pracovní hypotéza vycházela z předpokladu, že hybridní postkvantové algoritmy přinášejí zvýšenou výpočetní i datovou náročnost, což může mít zásadní dopad na jejich praktické nasazení.

Přínosem této práce je detailní srovnání teoretických možností a praktických výsledků při implementaci moderních kvantově odolných algoritmů, což může pomoci odborné komunitě i praxi lépe se zorientovat v problematice a zvolit vhodné řešení při přechodu na postkvantovou kryptografii.

I TEORETICKÁ ČÁST

1 POSTKVANTOVÁ KRYPTOGRAFIE

Experti odhadují, že kvantové počítače schopné prolomit současné asymetrické kryptografické systémy, jako je Rivest–Shamir–Adleman (RSA), mohou být dostupné během příštích 10 až 20 let. Tento časový rámec závisí na rychlosti technologického pokroku a investicích do vývoje kvantových počítačů. Ačkoli přímé prolomení šifrovacích algoritmů zatím nepředstavuje bezprostřední hrozbu, nebezpečí spočívá v současném sběru šifrovaných dat. Tato data mohou být v budoucnu zpětně dešifrována pomocí kvantových počítačů, což má zásadní dopad na ochranu citlivých informací, které mají dlouhou dobu životnosti, například v oblasti zdravotnictví, financí nebo státní bezpečnosti[1]

Postkvantová kryptografie (PQC) se zaměřuje na vývoj kryptografických algoritmů, které zajišťují ochranu dat i v prostředí kvantových počítačů. Tyto algoritmy využívají matematické problémy, které jsou považovány za obtížně řešitelné i s využitím kvantových výpočetních technologií. Význam PQC spočívá nejen v ochraně současných systémů, ale i v jejich přípravě na budoucí výzvy spojené s masivním rozšířením kvantových počítačů.

1.1 Hrozba kvantových počítačů

Moderní kryptografie je založena na matematických problémech, které jsou pro klasické počítače považovány za neřešitelné v rozumném čase. Například algoritmus RSA využívá obtížnosti faktorizace velkých čísel, zatímco algoritmy Diffie-Hellman a kryptografie eliptických křivek (ECC) staví na složitosti nalezení diskrétního logaritmu. U symetrických šifer, jako je například Advanced Encryption Standard (AES), je bezpečnost založena na nemožnosti efektivního prohledání všech možných klíčů. Vývoj kvantových počítačů však tuto rovnováhu zásadně narušuje. Kvantové algoritmy, jako jsou Shorův a Groverův, umožňují efektivně řešit úlohy, na nichž stojí bezpečnost těchto systémů, a tím výrazně ohrožují jejich odolnost vůči útokům.[2]

1.1.1 Shorův algoritmus

Shorův algoritmus, představený autorem Peterem Shorem v roce 1994, je kvantový algoritmus, který efektivně řeší dva klíčové matematické problémy: faktorizaci velkých čísel a nalezení diskrétního logaritmu. Shorův algoritmus využívá schopnosti kvantových počítačů, jako je superpozice a kvantová Fourierova transformace, k dosažení výsledků, které jsou pro klasické algoritmy výpočetně neřešitelné.

Princip Shorova algoritmu spočívá ve dvou hlavních krocích. Nejprve je problém převeden na zjištění periodicity specifické funkce, což je úkol zvládnutelný pomocí kvantových výpočtů. Poté je pomocí kvantové Fourierovy transformace určena perioda, která vede k efektivní faktorizaci čísla nebo nalezení diskrétního logaritmu.[3]

1.1.2 Groverův algoritmus

Groverův algoritmus, navržený Lovem K. Groverem v roce 1996, přináší kvadratické zrychlení při vyhledávání v nestrukturovaných databázích. Oproti klasickým algoritmům, které vyžadují v průměru N/2 pokusů pro N položek, zvládne Groverův algoritmus najít hledaný prvek za přibližně √N iterací. Algoritmus opakovaně zesiluje pravděpodobnost správného řešení pomocí orákula a amplifikační operace.[4]

Hlavní dopad na kryptografii je u symetrických šifer, jako je AES – efektivní délka klíče je kvůli Groverovu algoritmu snížena na polovinu (např. 128bitový klíč má efektivní bezpečnost pouze 64 bitů), což vede k doporučení používat dvojnásobnou délku klíče. Groverův algoritmus může mít vliv i na složitější problémy jako je Shortest Vector Problem (SVP) v oblasti postkvantové kryptografie, kde může být využit v hybridních kvantově-klasických útocích na některé mřížkové systémy.[5]

1.2 Přístupy postkvantové kryptografie

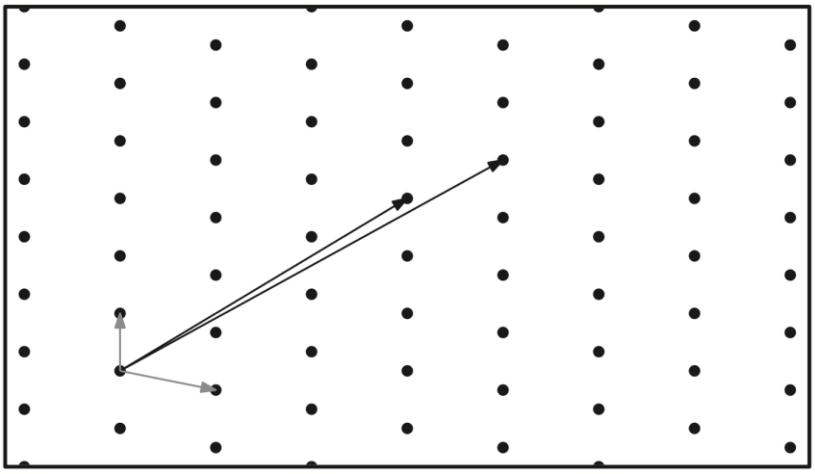
Postkvantová kryptografie se snaží zajistit bezpečnost šifrovacích systémů i v době, kdy budou dostupné výkonné kvantové počítače. Na rozdíl od současné kryptografie jsou nové algoritmy navrhovány tak, aby jejich bezpečnost byla založena na matematických problémech, pro které zatím nejsou známy efektivní kvantové algoritmy.

V této kapitole budou popsány a vzájemně srovnány přístupy, které jsou v současnosti považovány za kvantově odolné, pro něž neexistují známé efektivní kvantové útoky. Zároveň budou zmíněny i přístupy, které byly v minulosti dlouho považovány za bezpečné, ale s rozvojem kvantových algoritmů byly prolomeny a dnes se již nedoporučují pro praktické nasazení.

1.2.1 Kryptografie založená na mřížkách

Kryptografie založená na mřížkách představuje jeden z hlavních směrů postkvantové kryptografie. Tento přístup staví na pravidelných geometrických strukturách, nazývaných

mřížky – množinách bodů v n-rozměrném prostoru, které vznikají jako lineární kombinace několika bázových vektorů s celočíselnými koeficienty. [6]



Obrázek 1 Dvourozměrná mřížka a dvě možné báze

(zdroj: [6])

Matematickým základem jsou obtížné výpočetní problémy, jako je SVP a Closest Vector Problem (CVP). Ty spočívají v hledání nejkratšího nenulového vektoru v mřížce, respektive nejbližšího bodu k danému vektoru mimo mřížku. Vysoká výpočetní složitost těchto problémů, zejména ve vyšších dimenzích, zajišťuje jejich odolnost vůči jak klasickým, tak kvantovým počítačům.[7]

Na těchto problémech jsou založeny moderní kryptografické konstrukce, jako jsou Learning With Errors (LWE) a jeho rozšíření Ring-Learning With Errors (RLWE). V případě LWE se do výpočtů přidává šum, což znesnadňuje zpětné odvození původních dat. RLWE tuto myšlenku rozšiřuje do algebraických struktur kruhů (např. polynomiálních okruhů), čímž umožňuje efektivnější výpočty a menší velikosti klíčů při zachování stejné úrovně bezpečnosti.[8]

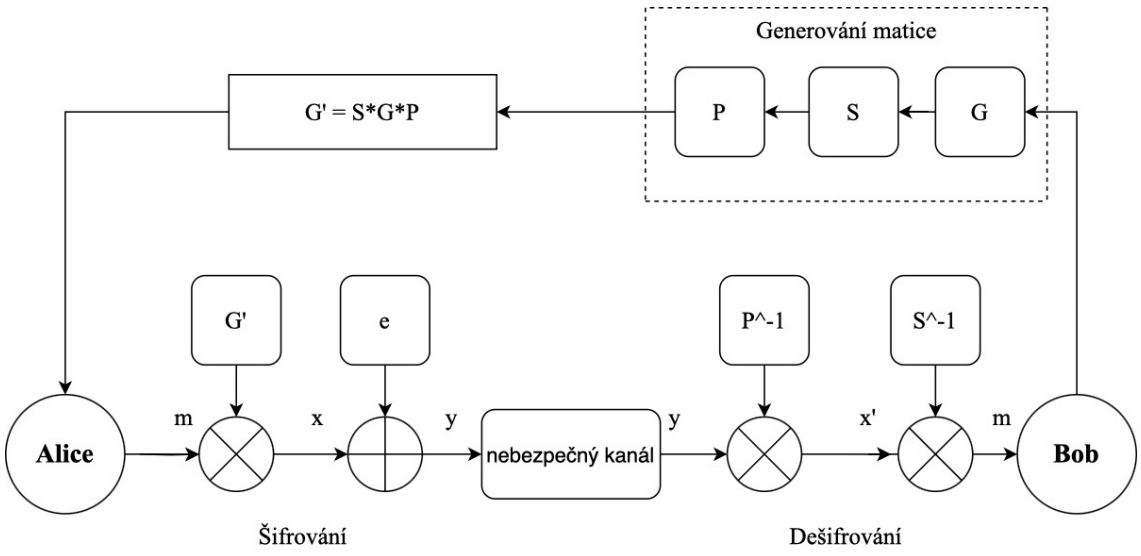
Mřížková kryptografie nachází široké uplatnění v moderních šifrovacích algoritmech navržených jako odolné vůči kvantovým útokům. Mezi klíčové příklady patří algoritmus Kyber, určený pro výměnu klíčů, a Dilithium, algoritmus pro digitální podpisy. Oba tyto algoritmy, byly v roce 2024 standardizovány jako federální standardy pro zpracování informací (FIPS) 203 a 204.[9; 10]

1.2.2 Kryptografie založená na teorii kódování

Kryptografie založená na kódech využívá principy z oblasti kódování s chybovou korekcí. Základní myšlenkou je zakódovat zprávu pomocí určitého kódu a poté ji úmyslně „narušit“ přidáním chyb. Dekódování je možné pouze se znalostí tajného kódu, který umožňuje chyby odstranit. Bez této znalosti je zpětné dekódování výpočetně velmi náročné, a to i s pomocí kvantového počítače.

Nejznámějším zástupcem kódové kryptografie je McElieceův kryptosystém, navržený již v roce 1978. Využívá tzv. Goppovy kódy, které umožňují efektivní opravu chyb, ale z pohledu útočníka působí jako náhodné matice. Zpráva se zašifruje pomocí veřejného klíče ve formě generující matice a přidáním náhodného vektoru chyb. Dešifrování je možné pouze se znalostí tajného dekódovacího algoritmu. Bez něj je nalezení původní zprávy výpočetně neproveditelné. Princip fungování McElieceova je znázorněn na Obrázku 2. Existuje i Niederreiterova varianta, která je s původním systémem ekvivalentní, liší se však způsobem zakódování zprávy.[6]

V rámci standardizačního procesu Národního institutu pro standardy a technologie (NIST) byl McElieceův kryptosystém zařazen mezi finalisty v kategorii mechanismů pro výměnu klíčů (KEM – Key Encapsulation Mechanism). I přes svou vysokou odolnost vůči kvantovým útokům nebyl nakonec doporučen ke standardizaci, a to především kvůli velké velikosti veřejného klíče, která omezuje jeho praktické nasazení.



Obrázek 2 Schéma McElieceova kryptosystému

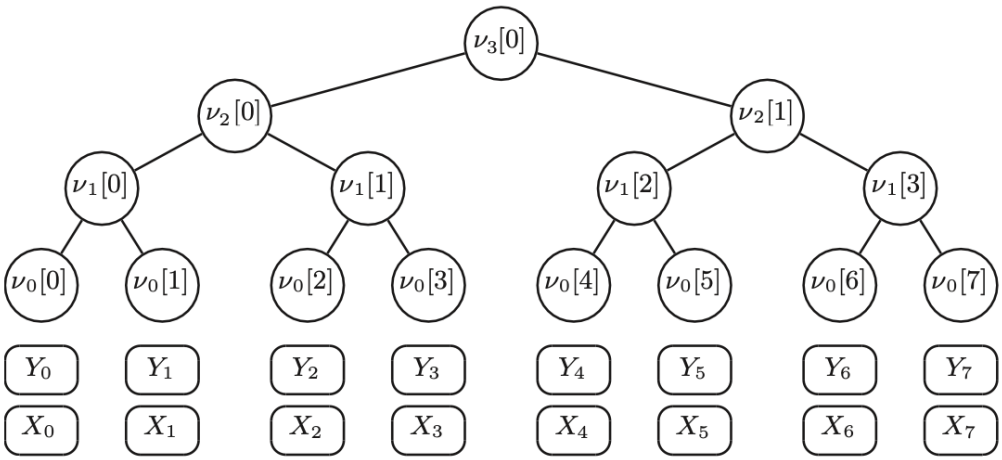
(zdroj: [11])

Ve čtvrtém kole tohoto procesu byly ponechány čtyři alternativní kandidáty, z nichž tři vycházejí z kódové kryptografie. Jedná se o algoritmy Classic McEliece, Bit Flipping Key Encapsulation (BIKE) a Hamming Quasi-Cyclic (HQC). Tyto algoritmy zůstávají jako záložní řešení pro případ, že by u hlavních finalistů došlo v budoucnu k objevení bezpečnostních slabin.[12]

1.2.3 Kryptografie založená na hashovacích funkcích

Hashovací kryptografie představuje jednoduchý a robustní přístup k postkvantové kryptografii. Využívá výhradně vlastností kryptografických hashovacích funkcí, jako je odolnost vůči kolizím a jednoznačnost výstupu. Bezpečnost těchto systémů nezávisí na složitých algebraických problémech, ale pouze na existenci jednosměrné funkce, což je považováno za jeden z nejzákladnějších kryptografických předpokladů.[6]

Hashovací schémata se dělí na stavová a bezstavová. Stavová, jako je Merkleho podpisový systém, využívají hashovací stromy, kde každý podpis odpovídá jedinečné větvi, a proto vyžadují sledování stavu, aby nedošlo k opětovnému použití podpisového klíče – to sice zajišťuje efektivitu, ale zároveň zvyšuje riziko chyb při správě klíčů. Naproti tomu bezstavová schémata, jako Stateless Practical Hash-based Incredibly Nice Cryptographic Signature Plus (SPHINCS+), tento problém odstraňují tím, že nevyžadují uchovávání stavu, což usnadňuje nasazení v praxi, avšak za cenu větší výpočetní náročnosti a rozměrnějších podpisů.[13]



Obrázek 3 Merkeleho strom s výškou 3

(zdroj: [6])

Na obrázku (Obrázek 3) lze vidět schéma Merkleho strom s výškou 3. Spodní uzly v_0 představují listy stromu odpovídající jednorázovým podpisovým klíčům. Pod každým listem je znázorněna dvojice X_i, Y_i, kde X_i je jednorázový tajný klíč a Y_i odpovídající veřejný klíč, který bývá typicky vytvořen jako hash hodnoty X_i. Každý vnitřní uzel vzniká hashováním svých dvou potomků, přičemž kořen stromu v_3 slouží jako veřejný klíč celého schématu. Při ověřování podpisu se využívá tzv. autentizační cesta od konkrétního listu až ke kořeni, což umožňuje efektivní ověření bez nutnosti znalosti celého stromu.[13]

Bezstavový algoritmus SPHINCS+, který staví výhradně na hashovacích funkcích, byl v roce 2024 oficiálně standardizován pod označením FIPS 205, čímž se stal prvním schváleným postkvantovým podpisovým algoritmem nezávislým na algebraických strukturách.[14]

1.2.4 Prolomené přístupy

I když se některé přístupy k postkvantové kryptografii původně jevily jako slibné a byly zařazeny do standardizačních procesů, pozdější analýzy a praktické útoky odhalily jejich zásadní zranitelnosti. Mezi nejvýznamnější příklady patří kryptografie založená na isogeniích nad supersingulárními eliptickými křivkami a multivariační kryptografie.

Kryptografie založená na isogeniích využívá obtížnosti nalezení isogenie mezi dvěma supersingulárními eliptickými křivkami. Tato metoda byla zpočátku považována za slibnou postkvantovou alternativu, protože problémy spojené s isogeniemi nejsou přímo řešitelné pomocí kvantových algoritmů jako Shorův algoritmus. [15]

Nejznámějšími zástupci této kategorie byly protokoly Supersingular Isogeny Diffie-Hellman (SIDH) a jeho rozšíření Supersingular Isogeny Key Encapsulation (SIKE), který kromě základní výměny klíčů implementuje také mechanismus KEM, čímž rozšiřuje možnosti praktického nasazení. SIKE byl zařazen jako alternativní kandidát do třetího kola soutěže NIST PQC. Oba protokoly nabízely malé velikosti klíčů, ale jejich výpočetní náročnost byla vysoká. [16]

V srpnu 2022 však Castryck a Decru publikovali útok, který umožnil efektivní získání soukromého klíče, čímž byla bezpečnost SIDH a následně i SIKE zcela narušena. Následkem toho byl SIKE vyřazen ze standardizačního procesu NIST a přestal být nadále považován za bezpečný postkvantový kandidát. [17]

Druhým již prolomeným přístupem je multivariační kryptografie (MPKC) založena na obtížnosti řešení systémů multivariabilních kvadratických rovnic nad konečnými tělesy. Její bezpečnost byla dlouho považována za velmi slibnou, protože řešení těchto rovnic je nedeterministicky polynomiálně těžký (NP) problém, vůči kterému jsou kvantové algoritmy, jako je Shorův, neúčinné. [7]

Jedním z nejznámějších zástupců tohoto přístupu byl algoritmus Rainbow, který byl finalistou třetího kola soutěže NIST PQC. Využíval rozšíření schématu Oil-Vinegar a sliboval vysokou efektivitu při podepisování a ověřování.[12]

Nicméně v roce 2022 byl Rainbow prakticky prolomen – útok umožnil efektivní rekonstruování soukromého klíče na běžném hardwaru. Tato událost zásadně otřásla důvěrou v MPKC a vedla k vyřazení Rainbow ze standardizačního procesu. Od té doby nebyl žádný multivariační algoritmus zařazen mezi finální NIST standardy.[18]

1.2.5 Porovnání jednotlivých přístupů

Postkvantové kryptografické algoritmy se liší v několika klíčových aspektech, jako je výpočetní náročnost, velikost klíčů, bezpečnostní status a praktická použitelnost. Tyto faktory jsou rozhodující při výběru algoritmů pro reálné nasazení a jejich efektivitu v různých prostředích.

Mřížková kryptografie patří mezi nejperspektivnější oblasti postkvantové kryptografie. Mezi její hlavní výhody patří vysoká efektivita operací, relativně malé velikosti klíčů a flexibilita nasazení v různých prostředích. Díky těmto vlastnostem jsou dva ze tří dosud standardizovaných algoritmů v rámci NIST PQC založené právě na této technologii. Hlavní nevýhodou může být větší velikost šifrovaného textu ve srovnání s klasickými algoritmy, která je však v praxi stále dobře zvládnutelná.

Kryptografie založená na kódech, představuje jeden z nejdéle známých a zároveň nejrobustnějších přístupů postkvantové kryptografie. Mezi hlavní výhody patří vysoká odolnost vůči známým typům útoků a stabilní teoretický základ. Hlavní nevýhodou této technologie je však velká velikost veřejného klíče, což výrazně komplikuje nasazení v prostředích s omezenou pamětí nebo šířkou pásma. Přesto zůstává kódová kryptografie vhodná zejména pro scénáře, kde velikost klíče není zásadní překážkou.

Kryptografie založená na hashovacích funkcích nabízí jednoduchý a konzervativní přístup k postkvantové bezpečnosti. Její hlavní výhodou je nezávislost na algebraických

strukturách, což zajišťuje robustnost vůči široké škále útoků, včetně těch kvantových. Nevýhodou jsou však větší velikosti podpisů a vyšší výpočetní náročnost ve srovnání s jinými přístupy, což může být limitující v prostředích s omezenými zdroji.

Isogenní kryptografie byla po dlouhou dobu považována za perspektivní přístup, a to především díky velmi malé velikosti klíčů. V roce 2022 však došlo k jejímu prolomení prostřednictvím efektivního útoku, který umožnil zrekonstruovat soukromý klíč. Následkem toho byla vyřazena z procesu standardizace NIST PQC a její další použití již není doporučováno.

Multivariační kryptografie, nabízela vysokou rychlost operací a nízkou výpočetní náročnost, což z ní činilo atraktivního kandidáta pro nasazení v reálném světě. V roce 2022 však byl algoritmus Rainbow prolomen praktickým útokem. Tento průlom zásadně narušil důvěru v multivariační přístupy, které již nejsou považovány za bezpečné a byly vyřazeny ze standardizačního procesu NIST PQC. [19; 20]

Tabulka 1 Porovnání postkvantových přístupů

Typ	**Výhody**	**Nevýhod**	**Bezpečnostní status**
Mřížky	Rychlost operací	Obtížné nastavení parametrů	Bezpečné
Kódy	Malá velikost podpisů, rychlé operace	Velká velikost klíčů	Bezpečné
Hashe	Prokázán důkaz bezpečnosti	Velká velikost podpisů	Bezpečné
Isogenní	Malá velikost klíčů	Pomalejší rychlost operací	Prolomeno
Multivariační	Rychlost operací	Velká velikost klíčů	Prolomeno

(zdroj: [7])

2 STANDARDIZAČNÍ PROCES NIST PQC

NIST zahájil proces standardizace post-kvantové kryptografie jako reakci na hrozbu, kterou představují kvantové počítače pro současné kryptografické systémy. Tento proces staví na dosavadních úspěšných standardizačních projektech, jako byly výběry algoritmů AES a Secure Hash Algorithm 3 (SHA-3), a klade důraz na transparentnost a zapojení odborné komunity.

První významný krok NIST zahrnoval vytvoření hodnotících kritérií, která reflektují požadavky na bezpečnost, výkon a implementační vlastnosti algoritmů. Tato kritéria byla zveřejněna k veřejné konzultaci v roce 2016 a následně finalizována. Na základě těchto kritérií byla v roce 2017 vyhlášena výzva k předkládání návrhů algoritmů pro šifrování, digitální podpisy a výměnu klíčů. Cílem NIST je vybrat algoritmy, které budou schopny odolat kvantovým i klasickým útokům, přičemž proces výběru je nastaven tak, aby zajistil dostatečný prostor pro odborné hodnocení a zpětnou vazbu.

NIST zdůrazňuje, že tento proces není koncipován jako soutěž, ale jako otevřená platforma, která má vést ke konsensu o nejvhodnějších standardech. Návrhy jsou hodnoceny na základě jejich odolnosti vůči různým typům útoků, jejich výkonnosti a také flexibility implementace na široké škále zařízení a platforem. Po přijetí návrhů byla stanovena tří až pětiletá lhůta pro veřejné hodnocení, během níž budou návrhy analyzovány a testovány. Tento přístup má zajistit, že standardizované algoritmy budou odpovídat nejen současným, ale i budoucím potřebám.

Proces zahrnuje také důraz na tzv. "crypto agility", tedy schopnost rychle a efektivně přejít na nové standardy bez zásadních narušení stávajících systémů. To je zvláště důležité v kontextu kvantových počítačů, jejichž plný potenciál by mohl být realizován během příštích 10 až 20 let.

Jedním z klíčových cílů NIST je zajistit, aby nové standardy byly přijaty nejen na národní, ale i na globální úrovni. Proto NIST aktivně spolupracuje s odborníky z akademické sféry, průmyslu a dalších standardizačních organizací. Transparentnost celého procesu má zajistit důvěru v nové standardy a jejich hladkou implementaci do současné infrastruktury.[21]

2.1 První kolo standardizačního procesu

První kolo standardizačního procesu NIST PQC [22], které probíhalo od prosince 2017 do ledna 2019, představovalo klíčovou fázi v identifikaci algoritmů odolných vůči kvantovým útokům. Tento proces umožnil identifikovat slibné návrhy a zároveň poukázat na slabiny některých algoritmů. Výsledkem byla užší skupina kandidátů, kteří postoupili do druhého kola, a zajistilo se, že vybrané algoritmy splňují nejen technická kritéria, ale také očekávání odborné kryptografické komunity.

Do procesu bylo přihlášeno 82 návrhů algoritmů, z nichž 69 splnilo minimální požadavky na přijetí a bylo zařazeno mezi kandidáty prvního kola. Tyto návrhy zahrnovaly 20 schémat pro digitální podpisy a 49 algoritmů pro šifrování veřejného klíče nebo výměnu klíčů. Mezi základní podmínky přijetí patřilo poskytnutí referenční implementace v jazyce C, testovacích případů a písemné specifikace. Algoritmy musely být rovněž implementovatelné na široké škále hardwarových a softwarových platforem.

2.1.1 Hodnotící kritéria

Hodnocení kandidátů v prvním kole standardizačního procesu NIST PQC se řídilo třemi hlavními kritérii: bezpečností, výkonem a náklady a charakteristikami algoritmů.

Bezpečnost – Bezpečnost byla nejdůležitějším kritériem. Algoritmy musely poskytovat ochranu proti kvantovým i klasickým útokům a splňovat požadavky na odolnost vůči adaptivním útokům na šifrované texty (IND-CCA2 – indistinguishability under chosen ciphertext attack) a podpisy (EUF-CMA – existential unforgeability under chosen message attack). Hodnoceny byly také odolnost vůči útokům postranními kanály a dopředná bezpečnost. NIST definoval pět kategorií bezpečnosti, aby mohl porovnat odolnost kandidátů.

Výkon a náklady – Důraz byl kladen na efektivitu algoritmů, včetně velikosti klíčů, šifrovaných textů a podpisů, výpočetní náročnosti a paměťových požadavků. Algoritmy musely být implementovatelné na široké škále hardwarových a softwarových platforem, přičemž NIST prováděl předběžné testy na referenční platformě.

Charakteristiky algoritmů – Upřednostňovány byly algoritmy s jednoduchým a flexibilním designem, které podporují široké nasazení a snadnou analýzu bezpečnosti. Hodnoceny byly také jejich licenční podmínky a dostupnost implementací.

2.1.2 Výsledky prvního kola

Na základě hodnocení podle kritérií definovaných v předešlé kapitole bylo z 69 kandidátů prvního kola vybráno 26 algoritmů, které postoupily do druhého kola. Z těchto 26 algoritmů bylo 17 určeno pro šifrování veřejného klíče a výměnu klíčů a 9 pro digitální podpisy.

Tabulka 2 Seznam algoritmů vybraných do druhého kola standardizace

Digitální Podpisy	**Šifrování a výměna klíčů**
CRYSTALS-DILITHIUM	BIKE
FALCON	Classic McEliece
GeMSS	CRYSTALS-KYBER
LUOV	FrodoKEM
MQDSS	HQC
Picnic	LAC
qTesla	LEDAcrypt
Rainbow	NewHope
SPHINCS+	NTRU
	NTRU Prime
	NTS-KEM
	ROLLO
	Round5
	RQC
	SABER
	SIKE
	Three Bears

(zdroj: [22])

2.2 Druhé kolo standardizačního procesu

V druhém kole procesu [23], které probíhalo od ledna roku 2019 do července 2020, bylo vybíráno ze 26 kandidátů na standardizaci. Druhé kolo bylo klíčovou fází, ve které byla kandidátská schémata podrobena detailnější analýze bezpečnosti, výkonu

a implementačních vlastností. Hlavním cílem bylo identifikovat algoritmy s největším potenciálem pro široké nasazení v éře post-kvantové kryptografie.

Během druhého kola prošly některé algoritmy významnými změnami, které zlepšily jejich bezpečnost a výkon. Například u CRYSTALS-KYBER bylo nahrazeno odvození klíče SHA3-256 za Secure Hash Algorithm Keccak Extendable-Output Function 256 (SHAKE256) a odstraněna komprese veřejného klíče, což zvýšilo efektivitu a snížilo riziko útoků. Nth Degree Truncated Polynomial Ring Units (NTRU) byl po spojení s dalším návrhem optimalizován, aby splňoval požadavky na bezpečnost, a byl rozšířen o transformaci Fujisaki-Okamoto. Algoritmus Strongly-Attackable Block Encryption with Rounding (SABER) získal pevnější bezpečnostní základ díky úpravám parametrů a lepší formální analýze. U FrodoKEM byla přidána vyšší bezpečnostní kategorie, i když jeho výkon zůstává nižší ve srovnání s jinými mřížkovými schématy. Multivariantní algoritmy jako Rainbow a Great Multivariate Short Signature (GeMSS) byly zjednodušeny a posíleny proti novým typům útoků.

Druhé kolo zahrnovalo také významné kryptoanalytické výsledky, které ukázaly slabiny některých návrhů. Například algoritmy Lightweight Authenticated Cipher (LAC), Low Error Decoding Algorithm Cryptosystem (LEDAcrypt) a Round5 byly eliminovány kvůli novým útokům nebo nejasnostem v jejich konstrukcích.

Na konci druhého kola bylo vybráno 7 finalistů a 8 alternativních kandidátů pro třetí kolo.

Tabulka 3 Seznam finalistů do třetího kola

Digitální Podpisy	**Šifrování a výměna klíčů**
CRYSTALS-DILITHIUM	Classic McEliece
FALCON	CRYSTALS-KYBER
Rainbow	NTRU
	SABER

(zdroj: [23])

Tabulka 4 Seznam alternativních kandidátů do třetího kola

Digitální Podpisy	**Šifrování a výměna klíčů**
GeMSS	BIKE
Picnic	FrodoKEM
SPHINCS+	NTRU Prime
	HQC
	SIKE

(zdroj: [23])

2.3 Třetí kolo standardizačního procesu

Hlavním úkolem ve třetím kole standardizačního procesu [12], které probíhalo od července 2020 do července 2022, bylo dokončit analýzu finalistů a určit první standardizované algoritmy.

Toto kolo se soustředilo na detailní hodnocení finalistů a alternativních kandidátů z druhého kola. Zaměření bylo na bezpečnostní analýzu, zahrnující odolnost vůči nově objeveným útokům, a na hodnocení výkonu, které zahrnovalo testování na různých platformách. Významnou roli hrála zpětná vazba odborné kryptografické komunity, která poskytla nové poznatky a analýzy.

Na základě výsledků třetího kola byly vybrány první algoritmy, které budou standardizovány, a současně byly definovány další kroky, včetně pokračování čtvrtého kola pro některé kategorie algoritmů. Tento proces stanovil základní standardy pro post kvantovou kryptografii, které budou implementovány do širokého spektra aplikací.

2.3.1 Hodnotící kritéria

Ve třetím kole standardizačního procesu NIST PQC byla bezpečnost hlavním kritériem hodnocení, protože algoritmy musely prokázat odolnost proti klasickým i kvantovým útokům a splnit požadavky na bezpečnost v pěti definovaných kategoriích. Mřížkové algoritmy, jako například CRYSTALS-KYBER a CRYSTALS-DILITHIUM, dosáhly vynikajících výsledků díky své robustnosti vůči kryptoanalytickým metodám, jako je redukce mřížky prostřednictvím algoritmu Block Korkine–Zolotarev (BKZ). Tyto testy potvrdily jejich stabilitu parametrů i při aplikaci optimalizovaných útoků.

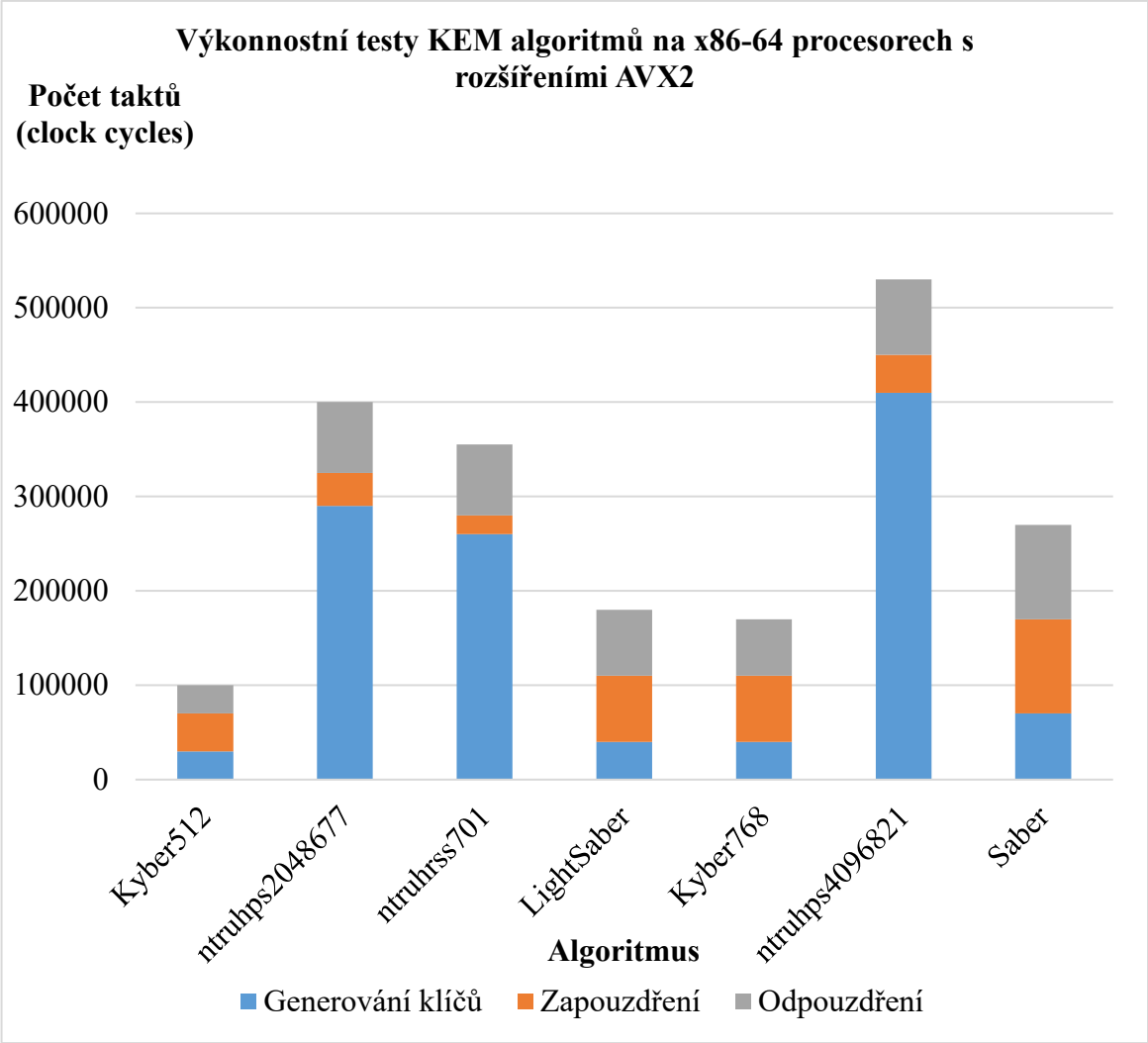
Naopak multivariantní algoritmy, například Rainbow, selhaly při testech zaměřených na algebraické redukce, které umožnily efektivní prolomení jejich bezpečnostních základů. Podobně algoritmus BIKE čelil problémům s odolností vůči postranním kanálům, zejména s chybami při dešifrování, což vedlo ke snížení důvěry v jeho praktickou implementaci.

Důkladné kryptoanalytické analýzy zahrnovaly také ověřování deklarovaných bezpečnostních kategorií, což v některých případech odhalilo, že parametry některých návrhů byly nastaveny příliš optimisticky. Tyto nálezy vedly buď k úpravám parametrů, nebo k eliminaci algoritmů, které nedokázaly splnit požadované standardy. Výsledky těchto testů zásadně ovlivnily výběr finalistů a ukázaly, že robustnost vůči široké škále útoků je klíčovým faktorem pro budoucí standardizaci.

Druhým hlavním kritériem při hodnocení byl výkon a náklady při reálném nasazení jednotlivých algoritmů. Posuzovaly se faktory, jako jsou velikost klíčů, šifrovaných textů a podpisů, rychlost operací (např. šifrování, dešifrování, generování klíčů) a nároky na paměť. Testy byly prováděny na různých platformách, aby se ukázala flexibilita algoritmů při různých scénářích použití, od výkonných serverů až po zařízení s velmi omezeným výkonem, jako jsou zařízení internetu věcí (IoT).

První graf (Obrázek 4) zobrazuje výkonovou náročnost několika algoritmů pro šifrování a výměnu klíčů na x86-64 procesorech s Advanced Vector Extensions 2 (AVX2) rozšířeními. Zobrazené algoritmy, jako jsou KYBER, SABER a varianty NTRU, byly hodnoceny na základě výpočetní náročnosti tří hlavních operací: generování klíčů, zapouzdření a odpouzdření.

Z výsledků je zřejmé, že algoritmy rodiny CRYSTALS-KYBER dosahují nejlepších výkonů jak v bezpečnostní úrovni 1 (KYBER512), tak v úrovni 3 (KYBER768). Algoritmy z rodiny SABER (LightSaber, Saber) si rovněž vedou velmi dobře a ve většině operací se přibližují výkonu CRYSTALS-KYBER. Naopak algoritmy NTRU (ntruhps2048677, ntruhrss701, ntruhps4096821) dosahují nejpomalejších výsledků, přičemž největší zátěž představuje operace generování klíčů, zatímco operace zapouzdření a odpouzdření dosahují srovnatelných časů s ostatními algoritmy.

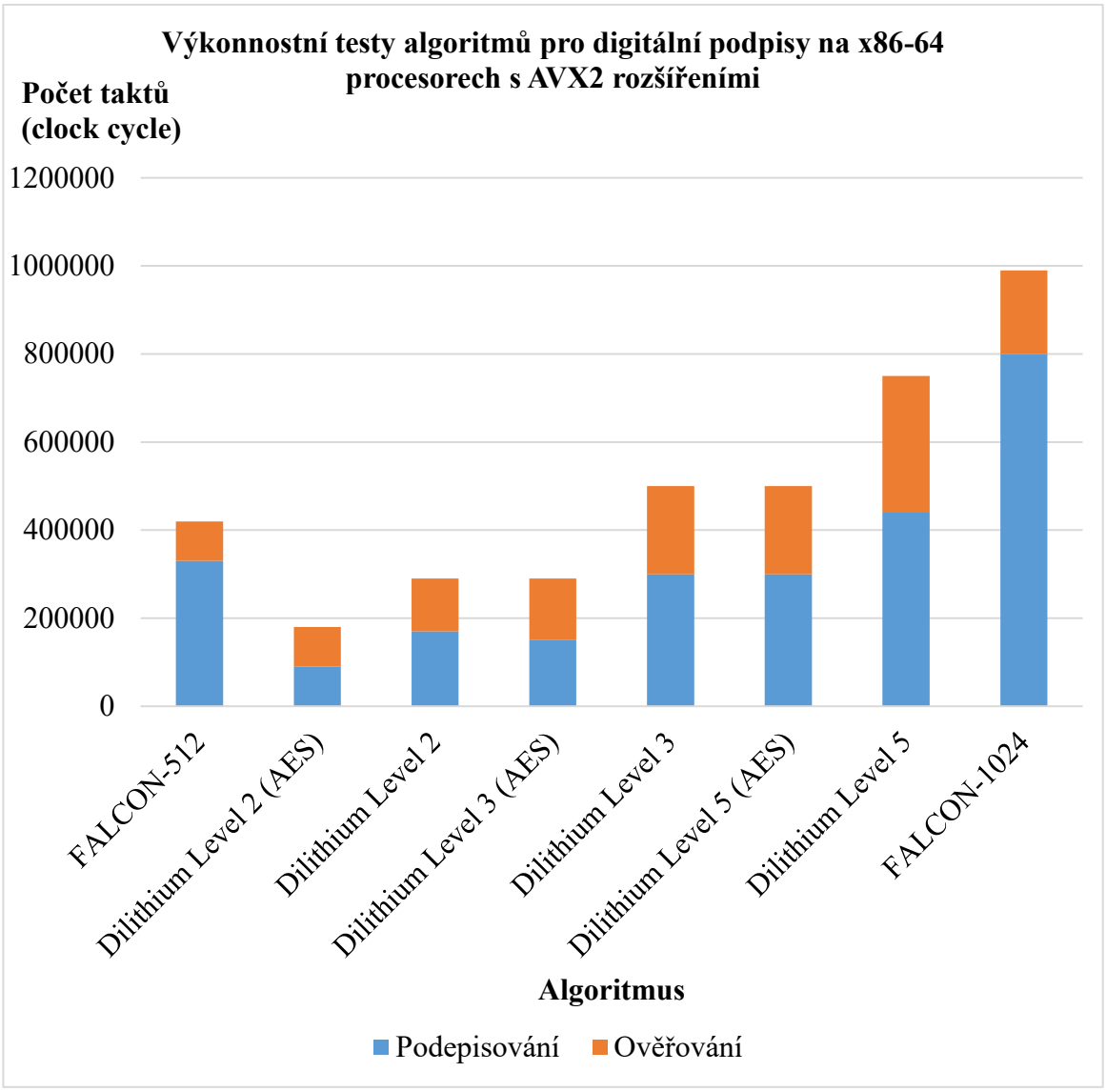


Obrázek 4 Graf výkonnostní testy KEM algoritmů na x86-64 procesorech s rozšířeními AVX2

(zdroj: [12])

V druhém grafu (Obrázek 5) jsou zobrazeny výkonnostní testy algoritmů pro digitální podpisy na x86-64 procesorech s AVX2 rozšířeními. Graf zobrazuje počet hodinových cyklů pro klíčové operace vytvoření podpisu a jeho ověření u různých variant algoritmů FALCON a CRYSTALS-DILITHIUM.

Z grafu je patrné, že algoritmy rodiny CRYSTALS-DILITHIUM jsou obecně efektivnější z hlediska klíčových operací. Například při porovnání algoritmů na úrovni zabezpečení Level 5 vykazuje CRYSTALS-DILITHIUM nižší výpočetní náročnost ve srovnání s algoritmem FALCON-1024, což z něj činí rychlejší volbu pro aplikace, kde je důležitý výkon.



Obrázek 5 Graf výkonnostní testy algoritmů pro digitální podpisy na x86-64 procesorech s AVX2 rozšířeními

(zdroj: [12])

V testech na pomalejších procesorech, jako je Advanced RISC Machine (ARM) Cortex-M4, si z KEM algoritmů vedl nejlépe CRYSTALS-KYBER, který prokázal nízké paměťové nároky a rychlé operace i na platformách s omezeným výkonem. Algoritmy SABER dosáhl podobně dobrých výsledků a ukázal se jako efektivní alternativa. Naopak algoritmus NTRU, jak již bylo možné odhadovat z výsledků na výkonnějších procesorech, vykazoval na pomalejších zařízeních výrazně vyšší výpočetní náročnost, zejména při generování klíčů, což jej činí méně vhodným pro zařízení s omezenými zdroji.

U algoritmů pro digitální podpisy si lépe vedly algoritmy rodiny CRYSTALS-DILITHIUM, především na bezpečnostních úrovních 2 a 3, kde prokázaly dobrou rovnováhu

mezi bezpečností a výkonem. Naproti tomu algoritmus FALCON-512, který odpovídá bezpečnostní úrovni 1, vyžadoval na pomalejším procesoru více času na vykonání operací, zejména při podepisování, což může omezit jeho využití v prostředích s omezeným výkonem.

Třetím klíčovým kritériem hodnocení byly charakteristiky algoritmů a jejich implementace, včetně jednoduchosti designu, flexibility nasazení a odolnosti vůči útokům postranními kanály. Mezi KEM algoritmy si CRYSTALS-KYBER vedl výborně díky své přehledné konstrukci a nízkým nárokům na implementaci, což jej činí vhodným pro širokou škálu zařízení. SABER dosáhl obdobně dobrých výsledků, přičemž jeho stabilní výkon a nízké paměťové nároky jej činí univerzální volbou.

U digitálních podpisů vynikl CRYSTALS-DILITHIUM, který díky modulárnímu designu a dobré rovnováze mezi výkonem a jednoduchostí implementace získal vysoké hodnocení. SPHINCS+ byl vyzdvihován za svou inherentní odolnost vůči útokům postranními kanály díky hashovacímu základu. Naopak algoritmus Classic McEliece u KEM a FALCON u digitálních podpisů čelily problémům s implementací, zejména kvůli velké velikosti klíčů nebo vyšším paměťovým nárokům.

2.3.2 Výsledky třetího kola

Ve třetím kole standardizačního procesu byly vybrány celkem čtyři algoritmy ke standardizaci. Z algoritmů určených pro šifrování a výměnu klíčů byl vybrán CRYSTALS-KYBER, který vynikl svou vysokou bezpečností, vynikajícím výkonem a flexibilitou implementace, což jej činí ideální volbou pro široké nasazení.

Mezi algoritmy pro digitální podpisy byly vybrány tři kandidáty. CRYSTALS-DILITHIUM byl označen za primární volbu díky vynikající rovnováze mezi bezpečností, výkonem a jednoduchostí implementace. FALCON se ukázal jako vhodná volba pro aplikace vyžadující menší velikost podpisů, čímž snižuje náklady na přenos dat. Naopak SPHINCS+ byl vybrán díky své unikátní odolnosti vůči postranním kanálům, kterou zajišťuje jeho hashovací základ, což z něj činí robustní volbu pro specifické aplikace. NIST plánuje vydat pro tyto algoritmy drafty.

Tabulka 5 Seznam algoritmů určených ke standardizaci

Digitální Podpisy	**Šifrování a výměna klíčů**
CRYSTALS-DILITHIUM	CRYSTALS-KYBER
FALCON	
SPHINCS+	

(zdroj: [12])

Některé algoritmy byly ve třetím kole z procesu standardizace zcela vyřazeny, protože neprokázaly dostatečnou bezpečnost, výkon nebo implementační vlastnosti. Mezi vyřazené patří například Rainbow, který byl eliminován kvůli zásadním kryptoanalytickým útokům zpochybňujícím jeho bezpečnostní základy, a GeMSS, jenž selhal při analýze odolnosti vůči kryptoanalýze. Dále byl vyřazen NTRUEncrypt, jehož výkon a složitost implementace neobstály v porovnání s ostatními kandidáty.

Jiné algoritmy, přestože nebyly přímo vybrány ke standardizaci, postoupily do čtvrtého kola pro další hodnocení. Mezi ně patří například BIKE, Classic McEliece, HQC a **S**IKE, které nabízejí různé přístupy, jako jsou kódově založené systémy nebo isogenie. Tyto algoritmy budou dále analyzovány, aby bylo možné určit jejich případnou budoucí standardizaci.

Tabulka 6 Seznam kandidátů do čtvrtého kola

Šifrování a výměna klíčů
BIKE
Classic McEliece
SIKE
HQC

(zdroj: [12])

3 NIST STANDARDY

Ve standardizačním procesu postkvantových kryptografických algoritmů, který vedl NIST, byly vybrány čtyři algoritmy, které jsou určeny ke standardizaci. Tento proces byl zahájen v roce 2016 jako odpověď na rostoucí hrozbu kvantových počítačů, které by mohly prolomit současné asymetrické kryptografické systémy.

V srpnu 2024 NIST zveřejnil první tři finální standardy pro šifrování a výměnu klíčů, přičemž čtvrtý algoritmus, určený pro digitální podpisy, stále prochází dokončovací fází standardizace. Tyto algoritmy budou tvořit základ budoucích kryptografických systémů, které mají zajistit ochranu citlivých dat i v éře kvantových počítačů.

NIST zároveň důrazně doporučuje organizacím a správcům systémů, aby co nejdříve začali implementovat nové standardy a připravili se na postkvantovou éru.[24]

3.1 FIPS 203

Standard FIPS 203 definuje Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM) [9], který byl vybrán jako postkvantový standard pro bezpečnou výměnu klíčů. ML-KEM vychází z algoritmu CRYSTALS-KYBER, jednoho z finalistů třetího kola standardizačního procesu NIST PQC. Tento mechanismus je navržen tak, aby poskytoval odolnost vůči kvantovým útokům, které by mohly prolomit současné asymetrické kryptografické systémy. ML-KEM je založen na mřížkové kryptografii, konkrétně na problému Module Learning with Errors (MLWE), což je jedna z nejvýznamnějších tříd postkvantových kryptografických problémů, u kterých se předpokládá odolnost vůči kvantovým algoritmům.

3.1.1 Varianty

Standard FIPS 203 definuje tři varianty ML-KEM: ML-KEM-512, ML-KEM-768, ML-KEM-1024, které se liší úrovní bezpečnosti, velikostí klíčů a šifrovaných textů. Každá varianta je navržena tak, aby poskytovala různou rovnováhu mezi bezpečností a výkonem, přičemž vyšší číslo v názvu odpovídá vyšší úrovni bezpečnosti.

Pro všechny varianty jsou společné pouze dvě konstanty, a to konstanta $n = 256$, která udává dimenzionalitu polynomů v mřížkovém prostoru, a konstanta $q = 3329$, což je modulární prvočíslo, které umožňuje rychlé výpočty pomocí Number-Theoretic Transform (NTT).

Jednotlivé varianty algoritmu ML-KEM se liší hodnotami několika klíčových parametrů, které ovlivňují jejich bezpečnostní úroveň, výpočetní náročnost a velikost výsledných dat. Parametr k určuje dimenzionalitu matic používaných při generování klíčů a šifrování – vyšší hodnota zvyšuje bezpečnost, ale zároveň zvětšuje klíče a šifrovaný text. Parametr η_1 určuje rozsah pro výběr náhodných vektorů při generování klíče, zatímco η_2 definuje rozsah šumových vektorů používaných během šifrování. Oba parametry ovlivňují náhodnost a tím i odolnost proti útokům. Parametry d_u a d_v slouží k bitové kompresi šifrovaného textu – určují, jak velká část výsledku bude při šifrování a dešifrování zachována. Vyšší hodnoty těchto parametrů vedou k větší přesnosti a bezpečnosti, ale mohou negativně ovlivnit výkon. Přehled konkrétních hodnot pro jednotlivé varianty je uveden v následující tabulce (Tabulka 7).

Tabulka 7 Porovnání parametrů jednotlivých variant FIPS 203

Varianta	k	η_1	η_2	d_u	d_v
ML-KEM-512	2	3	2	10	4
ML-KEM-768	3	2	2	10	4
ML-KEM-1024	4	2	2	11	5

(zdroj: [9])

Na základě rozdílů v konstrukčních parametrech jednotlivých variant, se liší také výsledné velikosti veřejného a soukromého klíče i velikost zapouzdřeného klíče. Vyšší bezpečnostní kategorie zpravidla vyžadují větší matice a delší šumové vektory, což vede k většímu objemu dat. Velikost zapouzdření označuje počet bajtů potřebných pro předání šifrovaného výstupu, který slouží k bezpečné distribuci sdíleného tajemství při výměně klíče. Přehled těchto hodnot je uveden v následující tabulce (Tabulka 8).

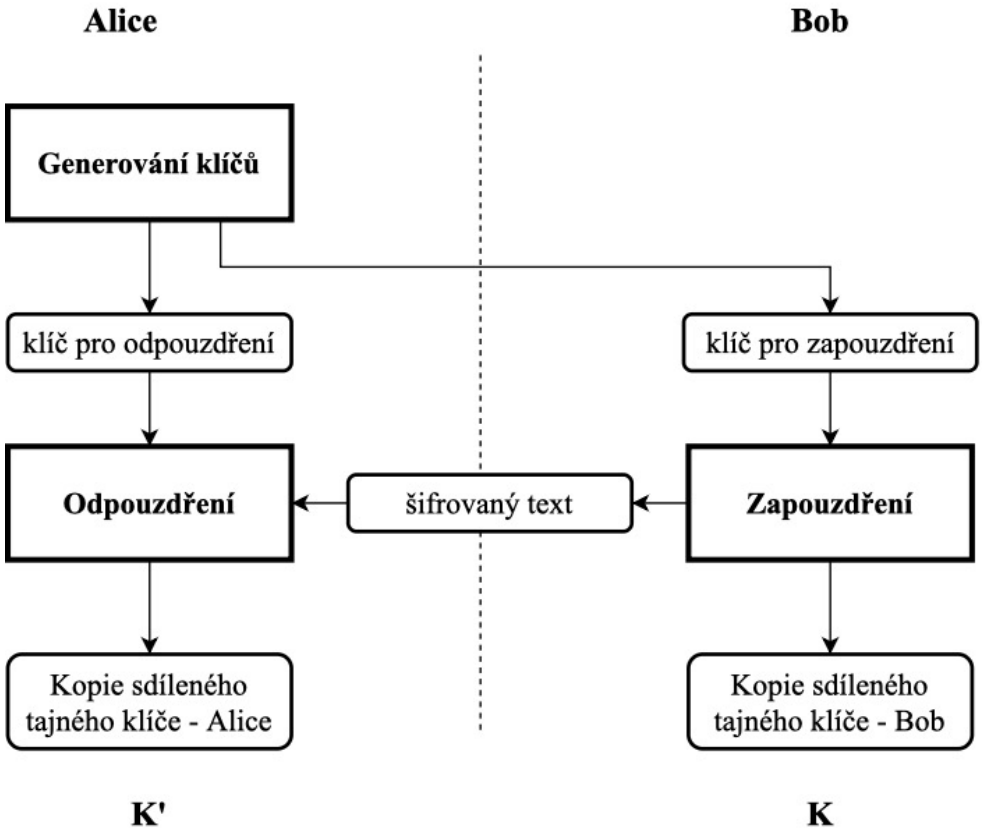
Tabulka 8 Porovnání variant FIPS 203

Varianta	**Bezpečnostní kategorie**	**Úroveň Bezpečnosti (bits)**	**Velikost klíčů (B)**	**Velikost zapouzdření (B)**
ML-KEM-512	Kategorie 1	128	2432	768
ML-KEM-768	Kategorie 3	192	3584	1088
ML-KEM-1024	Kategorie 5	256	4736	1568

(zdroj: [9])

3.1.2 KEM Mechanismus

Mechanismus KEM definovaný ve FIPS 203 poskytuje bezpečný způsob pro výměnu šifrovacích klíčů v asymetrických kryptosystémech. Proces fungování ML-KEM se skládá ze tří hlavních fází: generování klíčů, zapouzdření klíče a odpouzdření klíče.



Obrázek 6 Zjednodušené schéma KEM

(zdroj: [9])

3.1.3 Tvorba klíčů

Prvním krokem při implementaci algoritmu ML-KEM podle standardu FIPS 203 je vytvoření klíčového páru skládajícího se z veřejného klíče (ek) a soukromého klíče (dk). Tento proces zajišťuje funkce ML-KEM.KeyGen(), která nejprve vygeneruje dvě 32bajtové náhodné hodnoty *d* a *z*, a následně je předá interní funkci ML-KEM.KeyGen_internal() pro vlastní generaci klíčového páru.

Hodnota *d* slouží jako počáteční vstup pro generování klíčových komponent, zatímco hodnota z je součástí soukromého klíče a přispívá k bezpečnosti odpouzdření. Obě hodnoty jsou generovány tak, aby byly unikátní pro každou instanci, čímž je zajištěna odolnost vůči útokům založeným na opakovaném použití klíčů. Interní funkce následně deterministicky generuje veřejný a soukromý klíč.

Veřejný klíč obsahuje pouze šifrovací klíč pro schéma veřejného klíčového šifrování s externím klíčem (Keyed Public-Key Encryption, K-PKE), který je vytvořen pomocí funkce K-PKE.KeyGen(). Tento šifrovací klíč zahrnuje veřejnou počáteční hodnotu ρ, jenž slouží k deterministické regeneraci matice A, a vektor t, vypočítaný jako $t = As + e$, kde s je tajný vektor a e šumový vektor. Oba vektory jsou vzorkovány z binomického rozdělení (CBD). Díky použití počáteční hodnoty ρ není nutné ukládat celou matici A, protože ji lze kdykoli zpětně rekonstruovat.

Soukromý klíč obsahuje několik klíčových prvků nezbytných pro bezpečné odpouzdření. Zahrnuje dešifrovací klíč specifický pro schéma K-PKE (dkPKE), který obsahuje tajný vektor *s* transformovaný do NTT domény pro efektivní výpočty, dále kopii veřejného klíče *ek*, hash veřejného klíče *H(ek)* sloužící k ověřování integrity a náhodnou hodnotu *z*, která funguje jako ochranný mechanismus při odpouzdření – v případě, že odpouzdření selže, algoritmus místo skutečného tajného klíče vrátí náhodnou hodnotu, čímž zvyšuje bezpečnost celého procesu.

3.1.4 Proces zapouzdření

Po vygenerování klíčového páru následuje proces zapouzdření, jehož cílem je bezpečně přenést sdílený tajný klíč mezi dvěma stranami. Tento proces využívá veřejný klíč k vytvoření šifrované zprávy a sdíleného tajného klíče *K*, který je následně použit pro zabezpečenou komunikaci.

Zapouzdření je realizováno voláním algoritmu ML-KEM.Encaps(). Než zapouzdření proběhne, je nejprve provedena kontrola platnosti veřejného klíče. Tato kontrola zahrnuje ověření, zda veřejný klíč má správnou délku *384k + 32* bajtů podle specifikovaného parametru, a dále tzv. modulární kontrolu, která ověřuje, že zakódované hodnoty leží ve správném rozsahu *[0, q–1]*. Tento krok je důležitý pro detekci potenciálně neplatných nebo podvržených veřejných klíčů. Teprve po úspěšném dokončení těchto kontrol je možné přistoupit k samotnému zapouzdření.

Vlastní proces zapouzdření je realizován interním algoritmem ML-KEM.Encaps_internal(), kde *m* představuje nově vygenerovanou 32bajtovou náhodnou hodnotu. Nejprve se vytvoří hash veřejného klíče *H(ek)*, který je spolu s hodnotou *m* spojen a zpracován pomocí hashovací funkce G(). Výstupem tohoto kroku je sdílený tajný klíč *K* a náhodná hodnota *r* použitá pro šifrování. Pomocí algoritmu K-PKE.Encrypt() je následně hodnota *m* zašifrována a vznikne šifrovaný text *c*.

Výsledkem celého procesu je dvojice (*c, K*), kde *c* je zašifrovaný text určený k odeslání příjemci a *K* je sdílený tajný klíč, který bude následně použit pro šifrovanou komunikaci. Díky této konstrukci není tajný klíč nikdy přenášen přímo, což výrazně zvyšuje odolnost systému proti odposlechu a dalším útokům.

3.1.5 Proces odpouzdření

Po přijetí šifrovaného textu *c* následuje proces odpouzdření, jehož cílem je rekonstruovat původní sdílený tajný klíč pomocí soukromého klíče. Tento proces je realizován voláním algoritmu ML-KEM.Decaps() a zajišťuje, že komunikace zůstane důvěrná i v případě přenosu po nezabezpečeném kanálu.

Proces odpouzdření začíná extrakcí klíčových komponent ze soukromého klíče. Tyto komponenty zahrnují dešifrovací klíč K-PKE (*dkPKE*), který slouží k přímému dešifrování šifrovaného textu, dále kopii veřejného klíče *ek*, hash veřejného klíče *H(ek)* pro ověřovací operace a náhodnou hodnotu *z,* která se používá jako ochranný mechanismus v případě neúspěšného odpouzdření.

Následuje samotné dešifrování šifrovaného textu pomocí algoritmu K-PKE.Decrypt(). Tento krok rekonstruuje hodnotu *m'*, která by měla odpovídat původně zapouzdřené hodnotě *m*. Z této hodnoty *m'* je poté pomocí hashovací funkce odvozen kandidátní sdílený tajný klíč *K'*. Současně se vygeneruje nová verze šifrovaného textu *c'* na základě dešifrované hodnoty *m'* a uloženého veřejného klíče *ek*.

V dalším kroku je provedeno ověření integrity tím, že se porovná přijatý šifrovaný text *c* s nově vygenerovaným šifrovaným textem *c'*. Pokud se hodnoty *c* a *c'* shodují, je odpouzdření považováno za úspěšné a vrácen je vypočítaný sdílený tajný klíč *K'*. Pokud však dojde k nesouladu, což může být důsledkem podvrženého nebo poškozeného šifrovaného textu, je místo skutečného klíče vrácena hodnota odvozená z náhodné hodnoty *z*. Tento mechanismus zajišťuje, že odpouzdření je bezpečné i v případě aktivních útoků. Takto navržený proces odpouzdření poskytuje vysokou úroveň bezpečnosti a zároveň zachovává efektivitu, což je klíčové pro praktické nasazení v prostředích vyžadujících kvantově odolné zabezpečení.

3.1.6 Implementace a využití

Implementace ML-KEM podle standardu FIPS 203 musí splňovat přísné požadavky stanovené NIST, aby byla zajištěna kompatibilita a bezpečnost v kryptografických systémech. Implementace musí odpovídat schváleným postupům pro generování náhodných hodnot, šifrování a výměnu klíčů, přičemž podléhá federálním regulačním požadavkům. NIST plánuje vytvořit validační program, který umožní ověřit, zda konkrétní implementace splňuje všechny požadavky standardu. Tento program bude klíčový pro certifikaci kryptografických produktů, které využívají ML-KEM v reálných aplikacích.

Mezi klíčové oblasti využití ML-KEM patří federální informační systémy, kde zajišťuje zabezpečenou komunikaci a ochranu citlivých dat. V průmyslovém a komerčním sektoru se používá k šifrování databází, zabezpečení cloudu a hybridních postkvantových řešeních. Je také vhodný pro zařízení s omezenými zdroji, jako IoT a vestavěné systémy, kde nabízí varianty s nízkou výpočetní náročností. Dále se uplatňuje v protokolu Transport Layer Security (TLS), virtuálních privátních sítích (VPN – Virtual Private Network) a dalších bezpečnostních protokolech., kde nahrazuje RSA a ECC, jež jsou zranitelné vůči kvantovým útokům.

NIST doporučuje jako výchozí variantu ML-KEM-768, protože poskytuje vyvážený poměr mezi bezpečností a výkonem, a je tedy vhodná pro většinu aplikací. Varianta ML-KEM-512 je určena výhradně pro zařízení s omezenými výpočetními zdroji, kde je prioritou rychlost a nízké nároky na paměť, avšak její bezpečnostní úroveň je nižší. Naopak ML-KEM-1024, i když nabízí nejvyšší úroveň ochrany, je kvůli větší velikosti klíčů, zašifrovaných textů a vyšším výpočetním požadavkům považována za příliš náročnou pro běžné použití a vhodná spíše pro kritické aplikace s nejvyššími bezpečnostními požadavky.

3.2 FIPS 204

Standard FIPS 204 definuje algoritmus Module-Lattice-Based Digital Signature Algorithm (ML-DSA) [10], který poskytuje bezpečný způsob pro vytváření a ověřování digitálních podpisů. Tento algoritmus je navržen jako odolný vůči kvantovým útokům a je postaven na mřížkové kryptografii, konkrétně na problému MLWE – stejném problému, na kterém je založen i standard FIPS 203.

ML-DSA vychází z algoritmu CRYSTALS-Dilithium, jenž byl vybrán jako vítězný kandidát třetího kola standardizačního procesu NIST v kategorii digitálních podpisů.

Algoritmus ML-DSA umožňuje vytváření a ověřování digitálních podpisů pomocí asymetrické kryptografie. Pomocí soukromého klíče lze podepsat digitální zprávu, přičemž pravost podpisu může následně ověřit kdokoli, kdo má k dispozici odpovídající veřejný klíč. Tento mechanismus zajišťuje integritu a autenticitu podepsaných dat

3.2.1 Varianty

Standard FIPS 204 definuje tři varianty algoritmu ML-DSA: ML-DSA-44, ML-DSA-65 a ML-DSA-87. Každá z těchto variant odpovídá jiné bezpečnostní kategorii a nabízí jiný kompromis mezi úrovní ochrany, výpočetní náročností a velikostí klíčových dat. Díky tomu je možné algoritmus nasadit v široké škále scénářů – od omezených zařízení až po kritickou infrastrukturu. Všechny varianty navíc podporují dva režimy podepisování: běžný deterministický režim a tzv. zabezpečený (hedged) režim, který kombinuje interní a externí náhodnost a zvyšuje odolnost algoritmu vůči selhání generátoru náhodných čísel.

Všechny tři varianty využívají stejné základní parametry, které definují kryptografické prostředí algoritmu. Jedním z těchto parametrů je modul q, ve kterém probíhají všechny výpočty – operace se tedy provádějí v kruhu Zq, kde $q = 8380417$. Dále je to parametr ζ, což je odmocnina jednotky používaná v rámci NTT, která je klíčová pro efektivní násobení polynomů. V tomto případě je $\zeta = 1753$, což je 512. odmocnina jednotky v Zq. Třetím společným parametrem je d, což je počet bitů, které se odstraňují z hodnoty t během podpisu tento krok slouží ke kompresi podpisu a zmenšení výsledných dat. Hodnota parametru d je pro všechny varianty rovna 13.

Naopak v ostatních parametrech se jednotlivé varianty liší. Patří sem zejména rozměry matic k a l, které určují velikost mřížky a mají přímý vliv na bezpečnost a velikost klíčových

dat. Dále se liší parametr η, který určuje rozsah hodnot pro generování tajného klíče – vyšší hodnota znamená větší šum, což zvyšuje bezpečnost, ale i výpočetní náročnost. Parametr ω pak udává maximální počet nenulových koeficientů v nápovědě h, čímž ovlivňuje velikost podpisu a rychlost jeho ověřování. Dalším odlišujícím parametrem je τ, což je počet nenulových prvků (±1) ve vektoru výzvy c, který se používá při ověřování podpisu. Parametry γ_1 a γ_2 určují rozsah koeficientů pro hodnoty y a pro nízkořádové zaokrouhlování během výpočtů – ovlivňují tedy přesnost operací i odolnost vůči útokům. Konkrétní hodnoty těchto parametrů jsou uvedeny v následující tabulce (Tabulka 9).

Tabulka 9 Porovnání parametrů jednotlivých variant FIPS 204

Varianta	***(k, l)***	η	ω	τ	γ_1	γ_2
ML-DSA-44	(4, 4)	2	80	39	2^{17}	(q-1) /88
ML-DSA-65	(6, 5)	4	55	49	2^{19}	(q-1) /32
ML-DSA-87	(8, 7)	2	75	60	2^{19}	(q-1) /32

(zdroj: [10])

Na základě rozdílů v konstrukčních parametrech jednotlivých variant a jejich přiřazení k různým bezpečnostním kategoriím se liší také výsledné velikosti klíčů a podpisů. Vyšší bezpečnostní úroveň zpravidla znamená větší rozměry matic a přísnější parametry šumu, což vede k nárůstu velikosti veřejného i soukromého klíče a delšímu podpisu.

Tabulka 10 Porovnání variant FIPS 204

Varianta	**Bezpečnostní kategorie**	**Úroveň Bezpečnosti (bits)**	**Velikost klíčů (B)**	**Velikost podpisu (B)**
ML-DSA-44	Kategorie 2	128	3872	2420
ML-DSA-65	Kategorie 3	192	5984	3309
ML-DSA-87	Kategorie 5	256	7488	4627

(zdroj: [10])

3.2.2 Tvorba klíčů

První fází algoritmu ML-DSA je generování klíčového páru, které začíná voláním funkce ML-DSA.KeyGen(), která vrací veřejný a soukromý klíč. Nejprve se vygeneruje 32bajtový náhodná hodnota ξ, který slouží jako základ pro odvození všech potřebných

klíčových komponent. Tato hodnota je vstupem do interní funkce ML-DSA.KeyGen_internal(), která z něj deterministicky vytvoří trojici hodnot: 32bajtovou veřejnou počáteční hodnotu ρ, 64bajtovou soukromou počáteční hodnotu ρ' a 32bajtovou pomocnou hodnotu K určenou pro proces podepisování.

Veřejný počáteční hodnota ρ slouží k deterministickému generování matice A reprezentované v NTT doméně, která se používá při výpočtu veřejného vektoru. Ze soukromé hodnoty ρ' jsou následně vytvořeny dva tajné vektory s_1 a s_2 s krátkými koeficienty omezenými na rozsah $[-\eta, \eta]$. Následně se spočítá vektor $t = A\,s_1 + s_2$, který se dále rozdělí pomocí funkce Power2Round() na komprimovanou veřejnou část t_1 a tajnou část t_0 obsahující dolní bity koeficientů.

Veřejný klíč (public key, *pk*), který slouží pro ověřování podpisu, je reprezentován jako binární řetězec obsahující veřejnou počáteční hodnotu ρ a komprimovaný vektor t_1. Soukromý klíč (secrete key, *sk*) je tvořen následujícími komponentami: veřejnou počáteční hodnotou ρ, pomocnou počáteční hodnotou K, 64bajtovým hashem veřejného klíče *tr*, tajnými vektory s_1 a s_2 a vektorem t_0. Tato struktura zajišťuje, že všechny potřebné informace pro bezpečné a efektivní podepisování jsou obsaženy přímo v soukromém klíči, bez nutnosti dodatečných vstupů.

3.2.3 Proces podepisování

Druhou fází algoritmu ML-DSA je vytvoření podpisu zprávy pomocí soukromého klíče. Tento proces zajišťuje funkce ML-DSA.Sign(), která přijímá soukromý klíč, podepisovanou zprávu a volitelný kontextový řetězec. Pokud je délka kontextu větší než 255 bajtů, algoritmus vrací chybu. Jinak je vygenerována 32bajtová náhodná hodnota *rnd*. V základní zabezpečené variantě je získána z kryptografického generátoru, zatímco v deterministické variantě je nahrazena nulovým řetězcem.

Před vlastním podepisováním je původní zpráva rozšířena o informace o délce a obsahu kontextu a následně zakódována do bitového řetězce. Takto připravená zpráva *M'*, spolu s náhodnou hodnotou *rnd* a soukromým klíčem *sk*, je následně předána do interní funkce ML-DSA.Sign_internal().

Tato funkce provádí samotný výpočet podpisu nad zprávou a soukromým klíčem. Jde o hlavní kryptografickou část podepisování, která probíhá buď v náhodné nebo deterministické variantě. Obě verze využívají stejné postupy, liší se pouze v tom, zda je při každém podpisu použita čerstvá náhodná hodnota.

Nejprve se ze soukromého klíče získají všechny potřebné komponenty: veřejný počáteční hodnota ρ, tajné hodnoty K, s_1, s_2, t_0 a hash veřejného klíče. Z těchto hodnot se zkonstruuje interní reprezentace zprávy, která se spolu s náhodností použije k vytvoření privátního náhodné počáteční hodnoty pro daný podpis.

Podpis se generuje v takzvané smyčce s opakováním vzorkování (rejection sampling), která opakovaně vytváří návrhy podpisů a kontroluje jejich platnost. Nejprve je náhodně vygenerován vektor y, z něj se spočítá takzvaný závazek w_1, který slouží jako základ pro vytvoření výzvy c, ta je odvozena z kombinace w_1 a zprávy. Poté se vypočítá podpisová hodnota $z = y + cs_1$. Pokud z nebo další odvozené hodnoty překračují definované limity, návrh je zamítnut a celý postup se opakuje.

Jakmile jsou všechny podmínky splněny, algoritmus dopočítá takzvanou nápovědu h, která slouží k ověření podpisu bez nutnosti znát tajné části klíče. Výsledný podpis je složen z trojice hodnot: výzva c, podpisová hodnota z a nápověda h. Tyto části jsou zakódovány do bajtového řetězce a vráceny jako finální podpis.

3.2.4 Proces ověřování

Závěrečným krokem algoritmu ML-DSA je ověření platnosti podpisu pomocí veřejného klíče. Tuto operaci zajišťuje funkce ML-DSA.Verify(), která jako vstup přijímá veřejný klíč, podepsanou zprávu, podpis a volitelný kontext. Pokud je délka kontextu příliš velká, algoritmus ihned vrací chybu. V opačném případě je zpráva zformátována do standardizované podoby a spolu s veřejným klíčem a podpisem předána do funkce ML-DSA.Verify_internal().

Interní ověřovací algoritmus provede veškeré potřebné kontroly. Nejprve rozkóduje veřejný klíč a podpis a z nich získá potřebné komponenty – zejména závazek, podpisovou hodnotu a tzv. nápovědu h, která slouží k rekonstruování závazku během ověřování bez znalosti tajných hodnot. Dále rekonstruuje závazek podepisující strany a znovu vygeneruje výzvu, která by měla odpovídat té původní, obsažené v podpisu. Zároveň provádí kontrolu velikosti koeficientů v podpisové hodnotě a ověřuje, že nápověda obsahuje přípustný počet nenulových prvků.

Pokud všechny ověřovací podmínky proběhnou úspěšně a znovu vypočtená výzva odpovídá té původní, funkce vrací hodnotu „TRUE“, tedy že podpis je platný. V opačném případě vrací „FALSE“.

3.2.5 Implementace a využití

Implementace ML-DSA podle standardu FIPS 204 musí splňovat přísné požadavky stanovené NIST s cílem zajistit kompatibilitu, bezpečnost a odolnost vůči kvantovým útokům. Každá implementace musí přesně odpovídat definovaným postupům pro generování klíčů, podepisování a ověřování podpisů.

NIST plánuje vytvořit validační program, který umožní ověřit, zda konkrétní implementace splňuje všechny požadavky standardu. Tento program bude klíčový pro certifikaci kryptografických produktů využívajících ML-DSA v reálných aplikacích, včetně ochrany citlivých dat a zabezpečení komunikačních kanálů.

NIST doporučuje jako výchozí variantu ML-DSA-65, protože poskytuje vyvážený poměr mezi bezpečností a výkonem. Varianta ML-DSA-44 je určena pro aplikace s omezenými výpočetními zdroji, kde je prioritou rychlost a nízké nároky na paměť. Naopak ML-DSA-87 je navržena pro kritické aplikace s nejvyššími bezpečnostními požadavky, ale kvůli větší velikosti podpisů a vyšší výpočetní náročnosti je méně vhodná pro běžné použití.

3.3 FIPS 205

Standard FIPS 205 definuje Stateless Hash-Based Digital Signature Algorithm (SLH-DSA) [14], který byl vybrán jako alternativní postkvantový standard pro bezpečné digitální podpisy. Tento mechanismus je navržen tak, aby poskytoval odolnost vůči kvantovým útokům, které by mohly prolomit současné asymetrické kryptografické systémy. SLH-DSA je založen na hashovacích funkcích, což zajišťuje jeho bezpečnost.

SLH-DSA je odvozen od algoritmu SPHINCS+, který byl jedním z finalistů třetího kola standardizačního procesu NIST PQC. Hlavní výhodou SLH-DSA je jeho bezstavová povaha – na rozdíl od jiných hash-based podpisových schémat, jako je například eXtended Merkle Signature Scheme (XMSS), nepotřebuje algoritmus vést záznam o vnitřním stavu mezi jednotlivými podpisy. Tím je odstraněno riziko opětovného použití klíčů, což zvyšuje bezpečnost a zároveň zjednodušuje implementaci.

3.3.1 Varianty

Standard FIPS 205 definuje několik variant SLH-DSA, které se liší v použitých parametrech, bezpečnostní úrovni a efektivitě. Standard nabízí ve svých variantách dvě různé hashovací funkce, a to Secure Hash Algorithm 2 (SHA-2) a SHAKE256. Hashovací funkce SHA-2 představuje klasický přístup používaný v mnoha současných systémech, zatímco

SHAKE256, založený na konstrukci SHA-3, nabízí větší flexibilitu díky možnosti variabilní délky výstupu.

Kromě volby hashovací funkce se jednotlivé varianty liší také bezpečnostní úrovní. FIPS 205 specifikuje tři bezpečnostní úrovně. 128bitová bezpečnost, která odpovídá bezpečnostní kategorii 1, je vhodná pro méně náročné aplikace díky menší velikosti klíčů a rychlejším výpočtům. 192bitová bezpečnost, spadající do bezpečnostní kategorie 3, představuje vyvážený kompromis mezi výkonem a úrovní ochrany. Nabízí vyšší odolnost vůči útokům než 128bitová varianta, přičemž si zachovává přijatelnou velikost podpisu i výpočetní náročnost. 256bitová bezpečnost, zařazená do bezpečnostní kategorie 5, poskytuje nejvyšší úroveň ochrany a je určena pro kritické systémy s požadavky na dlouhodobou bezpečnost i v postkvantovém prostředí. Tato varianta je náročnější na výpočetní výkon a generuje větší podpisy, ale nabízí maximální kryptografickou odolnost.

Další odlišností mezi variantami je režim provozu algoritmu. Označení „s“ (small) představuje režim optimalizovaný pro co nejmenší velikost podpisu, zatímco „f“ (fast) označuje režim zaměřený na co nejvyšší rychlost zpracování. Kombinací těchto tří bezpečnostních úrovní, dvou typů hashovacích funkcí a dvou režimů provozu vzniká celkem 12 různých variant SLH-DSA, které pokrývají široké spektrum požadavků na bezpečnost a výkon.

Tabulka 11 Porovnání variant s režimem provozu „s“ FIPS 205

Varianta	**Bezpečnostní kategorie**	**Úroveň Bezpečnosti (bits)**	**Velikost klíčů (B)**	**Velikost podpisu (B)**
SLH-DSA-SHA2-128s SLH-DSA-SHAKE-128s	Kategorie 1	128	32	7856
SLH-DSA-SHA2-192s SLH-DSA-SHAKE-192s	Kategorie 3	192	48	16224
SLH-DSA-SHA2-256s SLH-DSA-SHAKE-256s	Kategorie 5	256	64	29792

(zdroj: [13])

Tabulka 12 Porovnání variant s režimem provozu „f“ FIPS 205

Varianta	**Bezpečnostní kategorie**	**Úroveň Bezpečnosti (bits)**	**Velikost klíčů (B)**	**Velikost podpisu (B)**
SLH-DSA-SHA2-128f SLH-DSA-SHAKE-128f	Kategorie 1	128	32	17088
SLH-DSA-SHA2-192f SLH-DSA-SHAKE-192f	Kategorie 3	192	48	35664
SLH-DSA-SHA2-256f SLH-DSA-SHAKE-256f	Kategorie 5	256	64	49856

(zdroj: [13])

3.3.2 Tvorba klíčů

Prvním krokem při implementaci algoritmu SLH-DSA podle standardu FIPS 205 je vytvoření klíčového páru, který se skládá ze soukromého a veřejného klíče. Soukromý klíč slouží k podepisování zpráv a musí zůstat důvěrný, zatímco veřejný klíč je určen k ověřování podpisů a může být volně distribuován.

Generování klíčového páru v algoritmu SLH-DSA je realizováno funkcí slh_keygen(), která nevyžaduje žádný vstup a vrací dvojici soukromého a veřejného klíče. Prvním krokem této funkce je náhodné vygenerování tří hodnot: *SK.seed, SK.prf* a *PK.seed*. Tyto hodnoty musí být generovány pomocí schváleného generátoru náhodných bitů, přičemž bezpečnostní síla tohoto generátoru musí být alespoň $8n$ bitů, kde n odpovídá velikosti bezpečnostního parametru (například 16, 24 nebo 32 bajtů podle zvolené varianty).

Hodnota *SK.seed* slouží k deterministickému generování tajných klíčových komponent uvnitř dvou vnitřních struktur algoritmu – Forest of Random Subsets (FORS) a XMSS. Hodnota *SK.prf* je určena pro deterministické generování náhodné hodnoty, která se používá při vytváření podpisu. *PK.seed* představuje veřejnou verzi počáteční hodnoty a používá se při výpočtu všech veřejných hodnot v rámci schématu. Zajišťuje například oddělení domén při využívání hashovacích funkcí, což zvyšuje bezpečnost a jednoznačnost výpočtů.

Po úspěšném vytvoření těchto hodnot funkce slh_keygen() volá vnitřní proceduru slh_keygen_internal(SK.seedm SK.prf, PK.seed), která vypočítá kořen *PK.root* vrchní vrstvy XMSS hypertree.

Výsledkem generování je klíčový pár, který se skládá ze soukromého a veřejného klíče. Soukromý klíč (*SK*) obsahuje hodnoty *SK.seed*, *SK.prf*, *PK.seed* a *PK.root*, zatímco veřejný klíč (*PK*) tvoří dvojice *PK.seed* a *PK.root*. Obě složky jsou úzce propojené a navržené tak, aby umožňovaly efektivní ověřování podpisů bez nutnosti uchovávat rozsáhlé datové struktury. Tato deterministická konstrukce zaručuje, že celý podpisový systém lze rekonstruovat pouze na základě několika počátečních hodnot, bez potřeby uchovávat celý hashový strom. Tento přístup významně usnadňuje implementaci, zejména v prostředích s omezenými výpočetními nebo paměťovými zdroji.

3.3.3 Proces podepisování

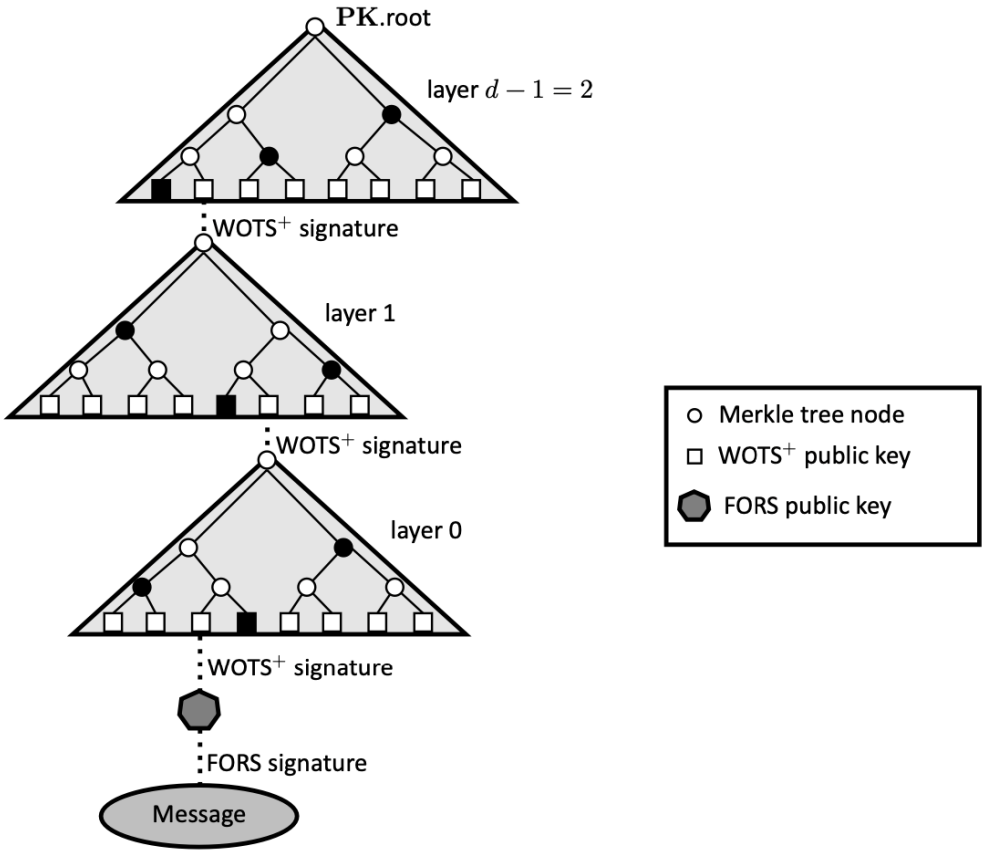
Proces podepisování v algoritmu SLH-DSA je realizován funkcí slh_sign() nebo její variantou hash_slh_sign(), přičemž v obou případech je hlavní výpočet delegován na interní funkci slh_sign_internal(). Rozdíl mezi těmito dvěma variantami spočívá v tom, že funkce slh_sign() očekává jako vstup celou zprávu, zatímco hash_slh_sign() pracuje s již předzpracovanou hashovanou zprávou. Výběr rozhraní závisí na kontextu použití a platí, že daný klíčový pár by měl být použit pouze s jednou z těchto funkcí. Podepisování probíhá deterministicky a skládá se z několika navazujících kroků, které propojují struktury FORS, Winternitz One-Time Signature Plus (WOTS+) a XMSS do hierarchické struktury hypertree.

Prvním krokem je vygenerování tzv. randomizéru – náhodného řetězce *R* pomocí funkce $PRF_{msg}()$, která jako vstup využívá *SK.prf* a náhodný vstup. Následně je vypočítán hash zprávy *M* v kombinaci s hodnotami *R*, *PK.seed* a *PK.root*. Výsledný hash se rozdělí na tři části: první část vstupuje do FORS schématu, druhá určuje, který XMSS strom v hyperstromu bude použit, a třetí část specifikuje konkrétní WOTS+ klíč.

Na základě těchto údajů se vytvoří podpis FORS pomocí funkce fors_sign(), přičemž současně je z podpisových údajů odvozen veřejný klíč FORS. Tento klíč pak vstupuje jako vstup do vyšší vrstvy – konkrétní XMSS větve. Pro podepsání FORS klíče se v XMSS použije podpisové schéma WOTS+, které následně využije autentizační cestu k dosažení kořene stromu. Tento proces se může rekurzivně opakovat v rámci vrstev hyperstromu.

Výsledný podpis se skládá ze tří hlavních částí. První z nich je náhodná hodnota *R* - randomizér, která zajišťuje jedinečnost každého podpisu a přispívá k jeho bezpečnosti. Druhou složkou je FORS podpis, jenž reprezentuje první vrstvu hashového podpisového schématu a vzniká na základě části hashované zprávy. Třetí částí je takzvaný hyperstrom

podpis, který se skládá z podpisů schématu WOTS+ a příslušných autentizačních cest v XMSS stromech, jež dohromady tvoří spojení mezi podpisem a kořenem celé hashové struktury. Celý podpis je zkonstruován jako jeden binární celek a jeho velikost závisí na zvolených parametrech algoritmu, konkrétně na hloubce hyperstromu, počtu vrstev a typu použité hashovací funkce.



Obrázek 7 Struktura podpisu SLH-DSA v hierarchii hyperstrom

(zdroj: [13])

3.3.4 Proces ověřování

Proces ověřování podpisu v algoritmu SLH-DSA je realizován funkcí slh_verify() nebo její variantou hash_slh_verify(), přičemž v obou případech se ověřovací operace předává interní funkci slh_verify_internal(). Stejně jako u podepisování funkce slh_verify() pracuje s původní zprávou, zatímco hash_slh_verify() očekává již vytvořený hash zprávy. Výběr závisí na kontextu aplikace a použití odpovídající varianty musí být v souladu s tím, jak byl vytvořen podpis.

Ověřování začíná výpočtem hash hodnoty zprávy pomocí funkce $H_{msg}()$, která kombinuje náhodnou hodnotu z podpisu, veřejnou počáteční hodnotu, kořen klíče a původní

zprávu. Výsledný hash se rozdělí na tři části – ty určují vstup do FORS, pozici v XMSS stromě a výběr WOTS+ klíče.

Z první části digestu a podpisu se zrekonstruuje veřejný klíč FORS. Ten je dále ověřován přes vrstvy XMSS: v každé vrstvě se pomocí WOTS+ ověří podpis a pomocí autentizační cesty se vypočítá uzel Merkleova stromu. Tento proces pokračuje přes všechny vrstvy až k vrcholu hyperstromu.

Na závěr se vypočtený kořen hyperstromu porovná s hodnotou *PK.root* z veřejného klíče. Pokud se shodují, podpis je platný „TRUE“, jinak je neplatný „FALSE“. Ověřování je plně deterministické a nevyžaduje uchovávání stavu.

3.3.5 Implementace a využití

Standard FIPS 205 představuje alternativu ke standardu digitálního podpisu FIPS 204, zejména v prostředích, kde je prioritou jednodušší a robustní implementace a kde není kladen důraz na minimalizaci velikosti podpisu.

Implementace algoritmu může být realizována v softwaru, firmwaru, hardwaru nebo jejich kombinaci. Standard taktéž umožňuje nahradit jakýkoli výpočetní krok alternativním matematicky ekvivalentním postupem, pokud pro všechny vstupy produkuje správný výstup. Díky tomu mohou vývojáři optimalizovat implementace například pro rychlost, paměť nebo energetickou náročnost. NIST plánuje zavedení validačního programu, který bude testovat implementace digitálních podpisových algoritmů na shodu s požadavky standardu.

Na rozdíl od předchozích FIPS publikací, standard FIPS 205 neurčuje žádnou konkrétní výchozí variantu algoritmu. Místo toho ponechává výběr konkrétní varianty na rozhodnutí implementátorů nebo bezpečnostních politik konkrétního systému.

Nakonec je třeba mít na paměti, že i když implementace odpovídá tomuto standardu, negarantuje to automaticky bezpečnost celého systému. Implementátor je odpovědný za to, že výsledný systém bude bezpečný jako celek – včetně ochrany před útoky postranními kanály, správného generování náhodných hodnot a bezpečného mazání citlivých dat.

3.4 Alternativy ke standardům

Za alternativy ke standardizovaným postkvantovým algoritmům NIST lze považovat ty algoritmy, které nebyly přímo vybrány k okamžité standardizaci, ale postoupily do čtvrtého

kola výběru. Tyto kandidáty NIST nadále vyhodnocuje z hlediska bezpečnosti, výkonu a implementačních vlastností, a v budoucnu mohou rozšířit portfolio schválených postkvantových algoritmů. Výběr těchto alternativ přispívá k větší rozmanitosti a odolnosti kryptografických standardů vůči novým typům útoků. Mezi aktuální alternativy ke standardizovaným postkvantovým algoritmům NIST patří pouze KEM algoritmy, tedy schémata pro výměnu klíčů a šifrování. V této fázi nejsou žádné alternativní kandidáty v kategorii digitálních podpisů.

3.4.1 BIKE

BIKE je postkvantový algoritmus pro výměnu klíčů založený na teorii kódování. Jeho bezpečnost stojí na obtížnosti dekódování binárních kódů, což je problém, který by měl být bezpečný i proti kvantovým útokům. BIKE používá tzv. bit-flipping dekódování, což znamená, že při dešifrování opakovaně upravuje jednotlivé bity, dokud nenajde správné řešení.

Hlavní výhodou BIKE je, že je rychlý a jednoduchý na implementaci ve srovnání s ostatními ne-mřížkovými KEM algoritmy. Díky odlišnému matematickému principu je také vhodný jako alternativa ke stávajícím standardům a zvyšuje celkovou bezpečnostní rozmanitost.

Nevýhodou BIKE je, že existuje nenulová pravděpodobnost selhání dešifrování, protože bit-flipping dekódování není vždy stoprocentně spolehlivé. Zatím také není přesně známa horní hranice této pravděpodobnosti a nebyly vyloučeny všechny možné slabé klíče. Z těchto důvodů je BIKE stále podrobován dalšímu zkoumání v rámci NIST standardizace.

3.4.2 Classic McEliece

Classic McEliece je postkvantový algoritmus pro výměnu klíčů založený na teorii kódování. Jeho bezpečnost je postavena na obtížnosti dekódování kódů Goppa, což je problém, který odolává známým útokům i v případě použití kvantových počítačů. Algoritmus má za sebou dlouhou historii výzkumu a prakticky nebyl prolomen ani klasickými, ani kvantovými metodami.

Mezi hlavní výhody Classic McEliece patří vysoká úroveň bezpečnosti ověřená desetiletími analýz a extrémně rychlé operace šifrování a dešifrování. Algoritmus generuje velmi malý šifrovaný text, což je praktické při přenosu dat, a jeho bezpečnost není závislá na mřížkových problémech, takže přináší diverzitu mezi postkvantovými algoritmy.

Nevýhodou je hlavně velikost veřejného klíče, která je oproti jiným algoritmům velmi velká a může komplikovat nasazení v prostředích s omezenou pamětí nebo šířkou pásma. Také není vhodný pro všechny scénáře použití právě kvůli těmto rozměrům klíče

3.4.3 HQC

HQC je postkvantový KEM algoritmus, který rovněž patří do oblasti kryptografie založené na teorii kódování. Bezpečnost HQC vychází z obtížnosti dekódování kvazicyklických kódů s danou Hammingovou vahou, což je úloha odolná i vůči kvantovým útokům. HQC tím rozšiřuje nabídku alternativ k mřížkovým kryptosystémům.

Mezi hlavní výhody HQC patří rovnováha mezi velikostí klíčů, rychlostí a bezpečností. Algoritmus nabízí matematickou rozmanitost v rámci portfolia postkvantových algoritmů, což je důležité pro zvýšení celkové odolnosti kryptografických systémů. HQC není závislý na stejných bezpečnostních předpokladech jako například Kyber, což snižuje riziko plošného prolomení.

Nevýhodou HQC jsou větší velikosti klíčů v porovnání s některými mřížkovými algoritmy a potřeba dalších bezpečnostních analýz, protože dosud nebyla všechna rizika plně prozkoumána. Zároveň není tak dlouho prověřen jako některé jiné kódové kryptosystémy.

3.4.4 SIKE

SIKE (Supersingular Isogeny Key Encapsulation) je speciální postkvantový algoritmus, jehož bezpečnost je založena na obtížnosti nalezení isogenie mezi supersingulárními eliptickými křivkami. Na rozdíl od předchozích algoritmů SIKE nevyužívá mříže ani kódování, ale zcela jiný, unikátní matematický princip.

Významnou výhodou SIKE je velmi malá velikost veřejných klíčů i zašifrovaného textu, což je praktické zejména pro použití v prostředích s omezenými zdroji, jako jsou mobilní zařízení nebo internet věcí. SIKE přináší do postkvantové kryptografie další diverzitu a umožňuje zkoumat nové možnosti ochrany proti kvantovým útokům.

Nevýhodou SIKE je výrazně nižší rychlost v porovnání s ostatními kandidáty, což může být problém při nasazení v aplikacích s požadavkem na vysoký výkon. Navíc v posledních letech prošel SIKE několika významnými kryptoanalytickými útoky, což vyvolalo otázky ohledně jeho skutečné bezpečnosti. Z těchto důvodů zůstává SIKE nadále předmětem dalšího výzkumu a diskusí v rámci standardizačního procesu

4 PŘECHOD NA POSTKVANTOVOU KRYPTOGRAFII

S příchodem kvantových počítačů, které mohou ohrozit současné kryptografické standardy, se stává klíčovým úkolem přechod na postkvantovou kryptografii. Tento proces zahrnuje nejen vývoj nových algoritmů, ale také jejich implementaci do existujících systémů. Kvůli komplexitě této transformace je důležité zajistit jednotný a koordinovaný přístup na mezinárodní úrovni. Proto organizace, jako je NIST ve Spojených státech a Národní úřad pro kybernetickou bezpečnost (NÚKIB) v České republice, poskytují doporučení a standardy pro efektivní a bezpečný přechod.

4.1 Doporučení NÚKIB

V únoru roku 2025 NÚKIB aktualizoval dokument „*Minimální požadavky na kryptografické algoritmy*“, kde se intenzivně zabývá přechodem na postkvantovou kryptografii a popisuje jednotlivé přístupy k řešení kvantové hrozby.

NÚKIB doporučuje, aby v úvodní fázi přechodu na postkvantovou kryptografii byla tato technologie nasazována pouze ve formě hybridního přístupu, tedy ve spojení s tradiční asymetrickou kryptografií. „*Na tomto přístupu stále trvá i většina evropských bezpečnostních autorit jako například německá BSI a francouzská ANSSI.*“

Hybridní přístup poskytuje vyšší míru bezpečnosti, protože algoritmy postkvantové kryptografie jsou sice navrženy a testovány proti útokům vedeným pomocí kvantových počítačů, ale nemusí být nutně odolné proti útokům klasických počítačů. Spojením s klasickou kryptografií tak hybridní řešení minimalizuje rizika vyplývající z možné existence zatím neznámých zranitelností postkvantových algoritmů.

NÚKIB udává, že existují různé úrovně kvantové zranitelnosti jednotlivých typů kryptografických algoritmů. Zatímco asymetrické algoritmy (RSA, Digital Signature Algorithm (DSA), Elliptic Curve Digital Signature Algorithm (ECDSA)) jsou vysoce zranitelné vůči Shorovu algoritmu a jejich nahrazení postkvantovou kryptografií je považováno za naléhavé, v případě symetrických algoritmů a hashovacích funkcí lze zranitelnost výrazně omezit použitím delších klíčů nebo výstupů. U hashů je doporučená délka zvýšena z 256 na 384 bitů, zatímco u symetrické šifry je minimální délka klíčů 256 bitů.

NÚKIB uvádí, že existují konkrétní scénáře, které si vyžadují naléhavější přechod k používání postkvantové kryptografie. Prvním je scénář označovaný jako „Harvest now,

decrypt later“, kdy útočník v současnosti zachycuje a dlouhodobě ukládá zašifrovaná data. Útočník následně vyčkává, až bude mít k dispozici kryptoanalyticky relevantní kvantový počítač, aby tato data zpětně prolomil. Druhým kritickým scénářem je oblast digitálních podpisů používaných k ochraně integrity firmware nebo softwaru, kde může být integrita dat ohrožena i mnoho let po vytvoření podpisu. V těchto případech NÚKIB doporučuje urychlený přechod na hybridní řešení s využitím důvěryhodných postkvantových algoritmů.

Právě za důvěryhodné algoritmy udává NÚKIB standardizované algoritmy instituce NIST, zejména ML-KEM a ML-DSA, které jsou primárními kandidáty pro nasazení díky kombinaci jejich výkonnosti a silných bezpečnostních záruk. V rámci hybridních řešení doporučuje NÚKIB použití bezpečnostních úrovní 3 a 5.

V souvislosti s přechodem na postkvantovou kryptografii doporučuje NÚKIB také zavedení kryptografické agility, tedy schopnosti systémů a infrastruktur rychle přecházet mezi různými kryptografickými algoritmy. Tato agilita zajistí, že organizace budou moci pružně reagovat na budoucí změny ve vývoji kryptografických technologií či případně nově objevené zranitelnosti.[15]

NÚKIB také implementoval podporu hybridního algoritmu X25519Kyber768 na svém portálu, čímž umožnil organizacím testovat kvantově odolnou kryptografii v reálném prostředí. Tímto krokem NÚKIB demonstroval, že hybridní přístup je nejen praktický, ale také vhodný pro plynulý přechod na postkvantové šifrovací standardy.[25]

4.2 Doporučení ostatních organizací

Americké organizace zabývající se kybernetickou bezpečností National Security Agency (NSA), Cybersecurity and Infrastructure Security Agency (CISA) a NIST vydaly společné doporučení k přípravě přechodu na postkvantovou kryptografii. Organizace by měly sestavit plán připravenosti na kvantové hrozby, provést analýzu aktuálních kryptografických systémů, spolupracovat s dodavateli technologií a začít prioritně migrovat kritické systémy na algoritmy, které budou odolné proti kvantovým počítačům.[26]

V roce 2024 vydala organizace NSA dokument „*CNSA Suite 2.0 and Quantum Computing FAQ*“, který poskytuje podrobné informace o přechodu na PQC a algoritmech CNSA 2.0. Tyto algoritmy, označované jako Commercial National Security Algorithms (CNSA), jsou určeny k použití ve všech veřejných standardech. Pro symetrické blokové šifry je definován algoritmus AES-256. Pro ustanovení klíčů je doporučen standard FIPS 203

(ML-KEM-1024) a pro digitální podpisy, včetně podpisů firmwaru a softwaru, standard FIPS 204 (ML-DSA-87).

NSA v dokumentu uvádí, že všechny systémy národní bezpečnosti (NSS) by měly být kvantově odolné do roku 2035. Nové systémy musí být kompatibilní s CNSA 2.0 od 1. ledna 2027. Veškeré zařízení, které CNSA 2.0 nepodporuje, musí být nahrazeno do 31. prosince 2030. Od 31. prosince 2031 budou algoritmy CNSA 2.0 povinné. Přechod bude postupný, starší algoritmy CNSA 1.0 budou dočasně fungovat souběžně s CNSA 2.0. NSA očekává, že u některých zařízení bude nutná i výměna hardwaru.

Podle dokumentu NSA důvěřuje algoritmům CNSA 2.0 a nevyžaduje hybridní přístup, ale uznává jeho možné řešení. Upozorňuje však na vyšší komplexitu a případné chyby při implementaci hybridních řešeních.[27]

V rámci Evropské unie (EU) vydalo 18 partnerů společné prohlášení o přechodu na postkvantovou kryptografii, ve kterém doporučují koordinovaný přístup v rámci Evropské unie. Stejně jako NÚKIB, EU zastává stejný přístup a rovněž zdůrazňuje význam hybridních řešení, která kombinují klasické a postkvantové algoritmy, a potřebu zajistit kryptografickou agilitu pro snadné přizpůsobení budoucím algoritmům. Dokument dále upozorňuje na riziko strategie „store now, decrypt later“ a doporučuje, aby členské státy vypracovaly plán migrace na PQC v souladu s mezinárodními standardy. Pro zajištění kvantové odolnosti infrastruktury EU byla na základě doporučení Evropské komise vytvořena pracovní skupina pro přechod na PQC, kterou společně vedou Francie, Nizozemsko a Německo. Evropská komise zároveň vyzývá všechny členské státy EU k aktivní účasti na přípravě plánu migrace na PQC.[28]

Čína přijala aktivní přístup k postkvantové kryptografii prostřednictvím programu Next-generation Commercial Cryptographic Algorithms (NGCC), který oznámil Institut pro standardy komerční kryptografie (ICCS). Program zahrnuje návrhy na nové algoritmy pro asymetrické šifrování (NGCC-PK – Public-Key Algorithms), hashování (NGCC-CH - Cryptographic Hash Algorithms) a blokové šifrování (NGCC-BC – Block Ciphers). Algoritmy budou hodnoceny podle bezpečnosti, výkonu a dalších kritérií, přičemž vítězné návrhy mohou být standardizovány. Tento přístup ukazuje snahu Číny o technologickou nezávislost a zvýšení bezpečnosti vůči kvantovým hrozbám.[29; 30]

Jižní Korea založila Výbor pro kvantovou strategii, který má koordinovat aktivity v oblasti kvantových technologií a zvýšit konkurenceschopnost země v této oblasti. Výbor

spravuje fond na podporu startupových projektů ve výši 15 milionů dolarů ročně. Plán zahrnuje vývoj kvantových počítačů, rozšíření kvantové infrastruktury a zajištění kvantově odolné kryptografie, která se plánuje integrovat do národních bezpečnostních systémů. Přestože vláda oznámila nárůst rozpočtu na tyto technologie o 51,4 % na 136 milionů dolarů, někteří odborníci se obávají, že to nebude dostatečné v konkurenci s masivními investicemi ze strany USA a Číny.[31]

Japonsko také aktivně implementuje postkvantovou kryptografii prostřednictvím spolupráce mezi vládními agenturami a soukromým sektorem. V lednu 2025 společnost PQShield, specializující se na PQC, oznámila své zapojení do programu financovaného organizací New Energy and Industrial Technology Development Organization (NEDO). Společnost hraje klíčovou roli při návrhu nových algoritmů a protokolů. Projekt poběží od roku 2024 do roku 2026 a má za cíl vytvořit robustní standardy PQC v souladu s doporučeními NIST.[32]

5 HYBRIDNÍ ALGORITMUS X25519MLKEM768

Hybridní algoritmus X25519MLKEM768 představuje kombinaci klasické kryptografie na bázi eliptických křivek a postkvantové kryptografie založené na mřížkách. Algoritmus X25519, který využívá eliptickou křivku pro efektivní výměnu klíčů, je spojen s postkvantovým algoritmem ML-KEM768, standardizovaným NISTem jako FIPS203, který je založen na mřížkových problémech a poskytuje odolnost vůči kvantovým útokům. Tato kombinace vytváří robustní a vysoce bezpečné řešení pro výměnu klíčů, které je odolné vůči útokům jak klasických, tak kvantových počítačů. Hybridní přístup je obzvláště vhodný pro zabezpečení Hypertext Transfer Protocol Secure (HTTPS) komunikace a je doporučován jako přechodné řešení před plným nasazením postkvantových algoritmů.[33]

5.1 X25519 – Algoritmus eliptických křivek

Algoritmus X25519 je založen na eliptických křivkách a využívá se pro bezpečnou výměnu klíčů v rámci protokolu Elliptic Curve Diffie-Hellman (ECDH). Je založen na křivce Curve25519, kterou navrhl Daniel J. Bernstein pro dosažení vysoké úrovně bezpečnosti a zároveň efektivity při šifrování.

Každý uživatel má 32 bajtový privátní a 32 bajtový veřejný klíč. Při navazování spojení si dvě strany vymění své veřejné klíče. Pomocí svého privátního klíče a veřejného klíče druhé strany pak spočítají sdílené tajemství pomocí funkce Curve25519(). Výsledný klíč se použije pro šifrování komunikace.

Algoritmus nabízí 128bitovou bezpečnost, je odolný vůči útokům postranními kanály a poskytuje ochranu i proti útokům na úrovni mezipaměti. Mezi hlavní výhody patří vysoká rychlost, krátké klíče (32 bajtů) s vysokou úrovní bezpečnosti a snadná implementace v různých kryptografických knihovnách (OpenSSL, BoringSSL, LibreSSL). Algoritmus je široce využíván v protokolech jako TLS, Secure Shell (SSH) a Internet Protocol Security (IPsec), kde zajišťuje bezpečnou komunikaci [34]

5.2 Princip fungování

Algoritmus je navržen pro zabezpečení v rámci protokolu TLS 1.3 a funguje následovně. Začíná se s generováním klíčových párů na klientově straně. Klient vygeneruje veřejný a privátní klíč pro ML-KEM a taktéž klíčový pár pro X25519. Klient také vygeneruje samostatný pár X25519 veřejného a privátního klíče jako záložní možnost pro případ, že

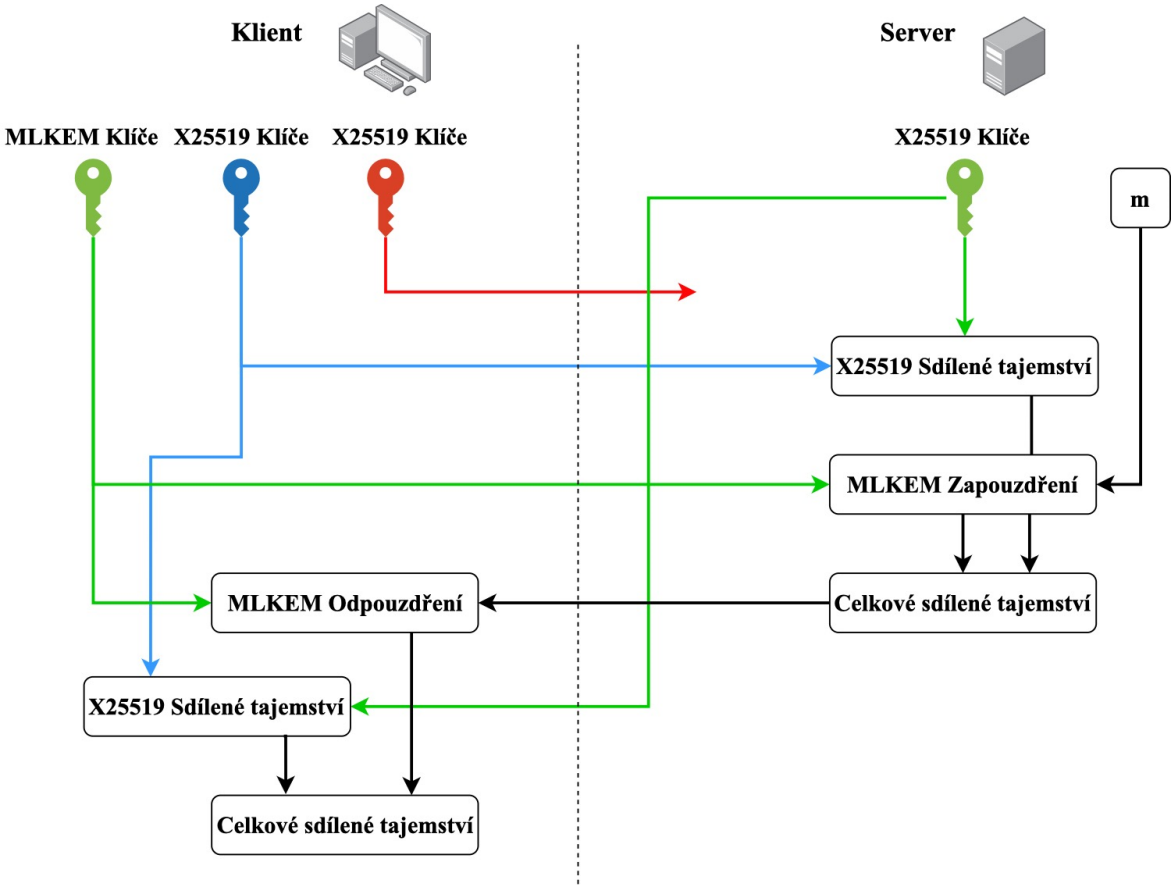
server nepodporuje X25519MLKEM 768. Pokud server tento protokol podporuje, záložní klíč se vůbec nepoužije.

Klient poté zahájí standardní TLS 1.3 handshake zprávou ClientHello, která obsahuje skupiny “X25519MLKEM768 ” a také “X25519” v sekci Supported Groups. V sekci KeyShare klient odešle dva klíče: hybridní klíč, vytvořený zřetězením veřejného KEM klíče s veřejným X25519 klíčem, a samostatný veřejný klíč X25519.

Server následně extrahuje veřejné klíče X25519 a KEM z kombinovaného klíče. Poté server vygeneruje vlastní pár veřejného a soukromého klíče pro X25519. Pomocí svého soukromého klíče a veřejného klíče od klienta spočítá sdílené tajemství pomocí algoritmu X25519.

Server dále použije funkci zapouzdření KEM s veřejným KEM klíčem klienta k výpočtu druhé části sdíleného tajemství a šifrovaného textu KEM. Obě sdílená tajemství jsou následně zřetězena do finálního sdíleného tajemství, které je použito pro výpočet klíčů v rámci TLS 1.3. Nakonec server zřetězí svůj veřejný X25519 klíč se šifrovaným textem KEM a odešle výsledný klíč zpět klientovi v rámci zprávy ServerHello.

Klient poté obdrží veřejný X25519 klíč od serveru a zkombinuje jej se svým soukromým klíčem X25519 k výpočtu sdíleného tajemství. Následně použije šifrovaný text KEM spolu se svým privátním KEM klíčem k výpočtu druhé části sdíleného tajemství pomocí funkce Decaps(). Stejně jako na straně serveru, obě tajemství zřetězí do finálního sdíleného tajemství, které použije jako vstup do funkce HKDF-Extract v rámci protokolu HKDF (HMAC-based Key Derivation Function), který slouží k bezpečné derivaci šifrovacích klíčů v protokolu TLS 1.3 (Transport Layer Security). Komunikace poté pokračuje podle standardního průběhu TLS 1.3.[35]



Obrázek 8 Zjednodušený princip fungování hybridního algoritmu X25519MLKEM768

(zdroj: [35])

5.3 Reálné využití

Tento hybridní algoritmus, jak už bylo napsáno, byl navržen primárně pro zajištění odolnosti při navazování šifrované komunikace pomocí TLS 1.3. Nyní je X25519MLKEM768 po standardizaci organizací NIST experimentálně využíván v prohlížečích Google Chrome a Mozilla Firefox, přičemž u obou je nutné aktivovat hybridní algoritmus v nastavení.

Cloudové služby jako Cloudflare a Amazon Web Services (AWS) také testují nasazení algoritmu pro šifrování komunikace mezi servery a klientem. Cloudflare již používá X25519MLKEM768 pro přenos dat mezi svými servery, a to již ve 33 % případech. Zatím však není jasné, zda je tato funkce dostupná pouze pro bezplatné účty nebo i pro všechny firemní zákazníky, a zda je šifrování použito pro úplné end-to-end spojení mezi klientem a serverem.

Společnost Meta přidala v květnu 2024 podporu pro X25519MLKEM768 a vlastní variantu X25519MLKEM512_FB do své knihovny Fizz. Zatím však není jasné, zda tuto podporu využívají na veřejně dostupných službách.[36]

Jak už bylo uvedeno dříve, web NÚKIB také nově podporuje hybridní algoritmus X25519Kyber768. Tato implementace umožňuje organizacím testovat kvantově odolnou kryptografii v reálném prostředí, čímž se potvrzuje, že hybridní přístup je vhodný pro přechod na postkvantové šifrovací standardy.[25]

5.4 Alternativní hybridní algoritmy

V této podkapitole jsou představeny alternativní hybridní algoritmy, které slouží jako možné náhrady za X25519MLKEM768 na odpovídající bezpečnostní úrovni (Level 3). Tyto algoritmy kombinují osvědčené klasické kryptografické metody s postkvantovými KEM algoritmy, které představují buď schválené standardy NIST, nebo jejich alternativy. Jejich zařazení přispívá ke zvýšení rozmanitosti a robustnosti kryptografických systémů při přechodu na postkvantové zabezpečení.

5.4.1 P-384MLKEM768

Hybridní algoritmus P-384MLKEM768 kombinuje klasickou eliptickou křivku P-384 a postkvantový algoritmus ML-KEM-768.

Výkon: P-384MLKEM768 nabízí velmi dobrý výkon, a to jak z hlediska rychlosti navazování spojení, tak efektivního využití paměti. Podle studie má pouze mírně vyšší režii oproti čistě klasickým řešením. Ve srovnání s X25519MLKEM768 jsou rozdíly v rychlosti handshake a velikosti klíčů minimální.

Kompatibilita: P-384 je standardizovaná eliptická křivka široce podporovaná v mnoha systémech. Kombinace s ML-KEM-768 je proto vhodná pro prostředí, kde je kladen důraz na standardizované algoritmy a snadnou integraci do existující infrastruktury.

P-384MLKEM768 je vhodný pro aplikace vyžadující vysokou úroveň bezpečnosti, dobrý výkon a kompatibilitu s běžně používanými kryptografickými knihovnami.[37]

5.4.2 RSA3072-MLKEM768

Hybridní algoritmus RSA3072-MLKEM768 kombinuje klasický algoritmus RSA s délkou klíče 3072 bitů a postkvantový algoritmus ML-KEM-768.

Výkon: X25519MLKEM768 má obecně nižší výpočetní nároky a menší velikost klíčů ve srovnání s RSA3072-MLKEM768. To může vést k rychlejšímu zpracování a menší zátěži na síťové přenosy.

Kompatibilita: RSA je široce podporován v existující infrastruktuře, což může usnadnit integraci RSA3072-MLKEM768 do stávajících systémů. Na druhou stranu X25519 je modernější algoritmus, který nabízí lepší výkon, ale nemusí být všude podporován.

Volba mezi RSA3072-MLKEM768 a X25519MLKEM768 závisí na konkrétních požadavcích a omezeních daného prostředí. Pokud je prioritou kompatibilita se stávající infrastrukturou, může být vhodnější RSA3072-MLKEM768[38]

5.4.3 P-384BIKEL3

Hybridní algoritmus P-384BIKEL3 kombinuje eliptickou křivku P-384 a postkvantový algoritmus BIKE na bezpečnostní úrovni L3.

Výkon: P-384BIKEL3 má vyšší latenci při navazování spojení a větší nároky na paměť ve srovnání s P-384MLKEM768 i X25519MLKEM768. Podle výsledků může být navazování spojení znatelně pomalejší.

Kompatibilita: P-384 zajišťuje kompatibilitu s běžnými kryptografickými systémy, avšak algoritmus BIKE není zatím tak rozšířený ani standardizovaný jako ML-KEM. Výhodou však je, že BIKE nabízí odlišný matematický základ, což zvyšuje diverzitu systému.

P-384BIKEL3 je vhodný tam, kde je preferována rozmanitost kryptografických schémat a diverzita bezpečnostních předpokladů před absolutním výkonem.[37]

5.4.4 P-384HQC192

Hybridní algoritmus P-384HQC192 kombinuje eliptickou křivku P-384 a postkvantový algoritmus HQC-192.

Výkon: P-384HQC192 má ve srovnání s P-384MLKEM768 i X25519MLKEM768 vyšší latenci a větší nároky na paměť. Rychlost navazování spojení je pomalejší, což může být nevýhodou v prostředích s vysokými nároky na rychlost.

Kompatibilita: P-384 je běžně podporován, ale algoritmus HQC není zatím standardizován ani natolik rozšířený jako ML-KEM. Výhodou však je matematická odlišnost, která přispívá k odolnosti systému.

P-384HQC192 lze využít v případech, kdy je upřednostňována rozmanitost postkvantových algoritmů a zvýšená odolnost systému vůči novým typům útoků.[37]

II PRAKTICKÁ ČÁST

6 IMPLEMENTACE X25519MLKEM768

V této kapitole je představena praktická implementace hybridního kryptografického algoritmu X25519MLKEM768. Pro demonstraci reálného případu užití byl zvolen kontext zabezpečené komunikace na internetu – konkrétně šifrovací handshake v rámci protokolu TLS, který tvoří základ důvěryhodného přenosu dat v protokolu HTTPS. Vzhledem k tomu, že právě v tomto prostředí se hybridní algoritmy navrhují a testují jako náhrada stávajících mechanismů ohrožených kvantovým výpočetním výkonem, představuje tento scénář ideální příklad pro demonstraci.

V rámci této kapitoly je popsán návrh zjednodušeného TLS-like handshaku využívajícího algoritmus X25519MLKEM768, jeho implementace a testování, a to bez nutnosti nasazení na reálný server. Simulace probíhá v lokálním prostředí, což umožňuje detailní sledování a pochopení jednotlivých kroků výměny klíčů a generování společného tajemství.

Celá implementace je realizována v programovacím jazyce Python, a to s důrazem na srozumitelnost, přehlednost a snadnou testovatelnost jednotlivých komponent hybridního šifrovacího mechanismu.

6.1 Implementace klasické výměny klíčů – X22519

Při implementaci algoritmu X25519, který v rámci protokolu slouží pro klasickou výměnu klíčů, bylo vycházeno z výukového dokumentu *Implementing Curve25519/X25519: A Tutorial on Elliptic Curve Cryptography* od Martina Kleppmanna. V uvedeném materiálu je podrobně popsán matematický základ a konstrukce algoritmu pomocí Montgomery ladder, včetně bezpečnostních aspektů, jako je výpočet v konstantním čase funkce – cswap() a úprava skaláru, funkce clamping().[39]

Implementace algoritmu X25519, jež se nachází v souboru *x25519.py*, byla rozdělena do několika samostatných funkcí, které odpovídají jednotlivým matematickým operacím potřebným pro výpočet skalárního násobení na křivce Curve25519.

6.1.1 Aritmetické operace

V této podkapitole jsou popsány základní matematické operace, které jsou využívány při výpočtech na eliptické křivce Curve25519. Veškeré operace probíhají v konečném tělese *Fp*, kde *p* je prvočíslo definující velikost pole.

$$p = 2^{255} - 19$$

Zvolené prvočíslo umožňuje efektivní výpočty na 64bitových procesorech, jelikož hodnoty menší než 2255 se vejdou do čtyř registrů. Jeho tvar $2^n - c$ zároveň zjednodušuje redukci modulo *p*, což zvyšuje rychlost výpočtů. Pole *Fp* zároveň poskytuje bezpečnost odpovídající přibližně 128 bitům.

Pro práci v tomto tělese byly implementovány základní aritmetické operace: sčítání, odčítání, násobení a výpočet inverzního prvku. Tyto funkce tvoří nezbytný základ pro následné výpočty v hlavní funkci x25519(), kde jsou využívány v každém kroku algoritmu Montgomery ladder.

6.1.2 Pomocné bezpečnostní funkce

Kromě základních aritmetických operací obsahuje implementace také dvě klíčové pomocné funkce, které zajišťují bezpečnost algoritmu X25519 vůči specifickým třídám útoků. Konkrétně se jedná o funkce clamp_scalar() a cswap().

Funkce clamp_scalar() upravuje vstupní privátní klíč – skalár do bezpečné podoby, jak je předepsáno ve specifikaci RFC 7748. Nejprve je vstup převeden z neměnitelného typu bytes na typ bytearray, který umožňuje manipulaci s jednotlivými bajty. Poté dochází k nastavení konkrétních bitů na požadované hodnoty: dolní tři bity jsou vynulovány, nejvyšší bit je vynulován a druhý nejvyšší bit je nastaven na 1. Tím se zaručí, že výsledný skalar je násobkem osmi, není příliš malý ani příliš blízko horní hranici tělesa, což zajišťuje bezpečné vlastnosti výsledného klíče.

Funkce cswap() provádí podmíněné prohození dvou hodnot bez použití větvení, čímž zajišťuje konstantní čas výpočtu. To zabraňuje útokům založeným na časování. Místo podmínky if používá bitové operace XOR a AND, které zaručují stejný průběh výpočtu pro všechny vstupy. Tato funkce je klíčová při práci s Montgomery ladder, kde se pravidelně rozhoduje, zda body mezi sebou prohodit.

6.1.3 Hlavní X25519 funkce

Funkce x25519() představuje hlavní výpočetní část algoritmu a slouží k provedení skalárního násobení na eliptické křivce Curve25519. Vstupem je privátní klíč ve formátu 32 bajtů a u-souřadnice vstupního bodu (standardně hodnota 9). Výstupem je u-souřadnice výsledného bodu, která slouží jako veřejný klíč nebo sdílené tajemství.

Výpočet probíhá pomocí algoritmu Montgomery ladder, který umožňuje efektivní a bezpečné násobení bodu skalárem bez použití souřadnice *y*. Funkce využívá pomocné operace jako add(), subtract(), multiply(), inverse(), cswap() a clamp_scalar() a provádí celkem 255 iterací odpovídajících jednotlivým bitům vstupního skalaru. Iterace probíhá od nejvyššího bitu směrem k nejnižšímu, což odpovídá způsobu zpracování skalárního součinu ve specifikaci RFC 7748.

Funkce x25519() tak představuje jádro celé implementace. Využívá optimalizovaný výpočet pomocí Montgomery ladder. Díky použití pouze x-souřadnic a konstantního časování operací je zajištěna vysoká výpočetní efektivita i odolnost proti běžným typům postranních útoků. Výstupem této funkce je buď veřejný klíč, nebo sdílené tajemství mezi dvěma stranami komunikace.

6.1.4 Generování klíčového páru

Funkce generate_keypair() slouží k vygenerování dvojice klíčů potřebných pro výměnu klíčů pomocí algoritmu X25519. Konkrétně se jedná o privátní klíč (skalár) a odpovídající veřejný klíč, který je získán jeho skalárním násobením se standardním výchozím bodem křivky ($u = 9$).

Nejprve je náhodně vygenerováno 32 bajtů pomocí funkce os.urandom(), která poskytuje kryptograficky bezpečné náhodné hodnoty. Tento vstup je následně v rámci funkce x25519() upraven pomocí tzv. clamping operace, čímž vznikne skutečný privátní klíč. Následně je tímto klíčem provedeno skalární násobení se základním bodem křivky za účelem výpočtu veřejného klíče. Výsledný veřejný klíč je nakonec převeden do pole bajtů ve formátu little-endian, jak vyžaduje specifikace algoritmu X25519.

6.1.5 Výpočet sdíleného tajemství

Funkce compute_shared_secret() slouží k výpočtu sdíleného tajemství mezi dvěma stranami na základě jejich klíčových párů. Tento krok odpovídá závěrečné fázi výměny klíčů dle principu Diffie-Hellmanova protokolu – každá strana využívá svůj privátní klíč a veřejný

klíč protistrany k výpočtu stejného tajného klíče, aniž by došlo k jeho přímému přenosu. Funkce jako vstup očekává vlastní privátní klíč ve formátu 32 bajtů a veřejný klíč protistrany, rovněž ve formátu 32 bajtů v little-endian reprezentaci.

Veřejný klíč je převeden na celé číslo a použit jako vstupní bod *u* pro funkci x25519(), kde se provede skalární násobení s vlastním privátním klíčem. Výsledkem je sdílené tajemství, které je opět převedeno do formátu 32 bajtů v little-endian reprezentaci. Toto sdílené tajemství je identické pro obě komunikující strany, pokud použijí odpovídající klíče.

6.2 Implementace postkvantové výměny klíčů – MLKEM768

V rámci implementace kvantově odolného algoritmu bude realizována jeho nejaktuálnější standardizovaná verze ML-KEM-768. Předloha implementace vychází přímo z oficiálního dokumentu vydaného organizací NIST, ve kterém jsou detailně popsány všechny potřebné parametry a funkce algoritmu.

6.2.1 Kryptografické primitiva

Kryptografická primitiva definovaná ve FIPS 203 v sekci 4.1 jsou v této práci implementována v souboru *crypto_primitives.py* a staví na standardních funkcích SHA-3 a SHAKE, které využívají modul *hashlib* jazyka Python.

Funkce PRF() je pseudonáhodná funkce založená na SHAKE256. Generuje deterministicky pseudonáhodná data pevně dané délky ze vstupní počáteční hodnoty *s* a čítače *b*. V ML-KEM slouží zejména k deterministickému odvození šumových vektorů.

Funkce H() představuje standardní hashovací funkci SHA3-256 vracející 32bajtový hash vstupu *s*. Používá se například pro hashování veřejného (zapouzdřujícího) klíče, jehož hash je součástí soukromého (odpouzdřujícího) klíče.

Funkce J() poskytuje 32bajtový hash pomocí SHAKE256. Využívá se při implicitním odmítnutí v procesu odpouzdření k výpočtu alternativního sdíleného klíče.

Funkce G() získává dva nezávislé 32bajtové výstupy rozdělením 64bajtového hashe SHA3-512 vstupu *c*. Slouží jako funkce pro odvození klíče například pro derivaci interních počátečních hodnot ρ a σ nebo sdíleného klíče *K* a náhodnosti *r*.

6.2.2 Implementace pomocných algoritmů

V sekci 4.2 standardu je definována sada obecných pomocných obecných algoritmů, které slouží jako základní stavební bloky pro hlavní funkce ML-KEM. Tyto algoritmy, implementované v souboru *utils.py*, zajišťují konverze datových typů, kompresi dat a pseudonáhodné vzorkování. Jejich korektní implementace, popsaná níže, je klíčová pro správnou funkci a bezpečnost celého mechanismu.

Pro převod mezi bajtovými řetězci a poli bitů byly implementovány funkce BytesToBits() a BitsToBytes(). Dle specifikace FIPS 203 je použita konvence little-endian, kdy nejméně významný bit bajtu odpovídá bitu na nejnižším indexu.

Pro ztrátovou kompresi koeficientů polynomů slouží funkce Compress() a Decompress(). Funkce Compress() převede celé číslo x na přibližnou hodnotu s menším počtem bitů d. Funkce Decompress() provádí přibližnou rekonstrukci původní hodnoty z d-bitového komprimovaného čísla y. Obě funkce využívají definovaný způsob zaokrouhlování implementovaný pomocí celočíselné aritmetiky. Jejich účelem je snížit velikost přenášených dat v šifrovaném textu.

Funkce ByteEncode() a ByteDecode() zajišťují převod mezi polem $N=256$ d-bitových celých čísel, která reprezentují koeficienty, a odpovídajícím bajtovým řetězcem. Funkce ByteEncode() prochází vstupní pole koeficientů a pro každý z nich postupně extrahuje jeho d bitů, počínaje nejméně významným. Všechny získané bity z celého pole jsou poté spojeny a převedeny na výsledný bajtový řetězec. Funkce ByteDecode() provádí inverzní operaci: nejprve převede vstupní bajtový řetězec na sekvenci bitů a následně pro každou skupinu d bitů zrekonstruuje odpovídající celé číslo.

Funkce SampleNTT(), implementována v souboru *sampling.py*, generuje pseudonáhodné koeficienty polynomu přímo v NTT doméně. Na základě vstupního 32bajtové vstupní hodnoty ρ a dvou indexů i, j využívá SHAKE-128 k produkci proudu kandidátních hodnot pro koeficienty. Tyto hodnoty jsou akceptovány pouze pokud jsou menší než modul Q. Tento proces odmítacího vzorkování se opakuje, dokud není získáno všech N koeficientů, které se typicky používají pro deterministické generování matice A.

Funkce SamplePolyCBD(), rovněž umístěná v souboru *sampling.py*, vytváří speciální šumové polynomy. Tyto polynomy se skládají z čísel, která jsou záměrně malá. Funkce k tomu používá vstupní data B a parametr η. Tento parametr η určuje, jak malá výsledná

čísla v polynomu mají být. Výsledkem je tedy seznam čísel, který se v algoritmu ML-KEM používá jako tajný nebo šumový prvek.

6.2.3 Implementace NTT a polynomiálního násobení

Klíčovou součástí algoritmu ML-KEM pro dosažení potřebné efektivity je využití NTT, jak je definováno ve standardu v sekci 4.3 a implementováno v souboru *ntt.py*. Přímé násobení polynomů by bylo výpočetně náročné proto NTT umožňuje převést polynomy do NTT domény, kde lze jejich součin spočítat výrazně rychleji pomocí násobení po jednotlivých složkách, s využitím specializovaných operací.

Funkce NTT() a InvNTT() realizují Číselně Teoretickou Transformaci a její inverzi standardu FIPS 203. Tyto transformace jsou výpočetně efektivní variantou Diskrétní Fourierovy Transformace přizpůsobenou pro operace v konečném tělese – modulo Q a tvoří základ pro rychlé násobení polynomů. Pro dosažení efektivity využívají Cooley-Tukey algoritmus, jehož klíčovou součástí jsou předpočítané faktory, jejichž hodnoty jsou uvedeny v dodatku *A* standardu a v této implementaci uloženy v konstantě *ZETAS_BITREV*, které se přímo používají v motýlkových operacích obou algoritmů. Funkce NTT() transformuje polynom ze standardní koeficientové reprezentace do NTT domény s bitově převráceným uspořádáním koeficientů. Funkce InvNTT() provádí inverzní proces a zahrnuje finální škálování předpočítaným faktorem.

Funkce MultiplyNTTs() slouží k násobení dvou polynomů, které jsou již v NTT doméně. V NTT doméně se násobení provádí efektivně po jednotlivých složkách. Tato funkce tedy prochází všech 128 komponent a pro každou komponentu volá funkci BaseCaseMultiply(), která provede samotné násobení pro danou komponentu. Výsledkem je nové pole 256 koeficientů, které představuje součin obou vstupních polynomů v NTT doméně.

Funkce BaseCaseMultiply() je podprogramem pro MultiplyNTTs() a provádí základní násobení pro jednu komponentu. Na vstupu dostává koeficienty dvou prvků a konstantu gama specifickou pro danou komponentu. Konstanta γ je jednou z hodnot předpočítaných podle Dodatku A standardu a je již uložena v konstantách implementace. Funkce pak podle definovaných algebraických pravidel vypočítá dva koeficienty výsledného součinu, přičemž všechny operace probíhají modulo Q.

6.2.4 Implementace schématu K-PKE

Tato kapitola popisuje implementaci schématu K-PKE, které tvoří základní komponentu pro mechanismus zapouzdření klíče ML-KEM, jak je definováno ve standardu FIPS 203, konkrétně v sekci 5. Implementace algoritmů K-PKE je realizována v souboru *pke.py*, přičemž všechny specifické parametry pro danou bezpečnostní úroveň jsou pro přehlednost definovány v souboru *constants.py*. Je důležité zdůraznit, že K-PKE není navrženo pro samostatné nasazení jako šifrovací schéma, ale slouží výhradně jako sada podprogramů, které využívají hlavní algoritmy ML-KEM. Tyto algoritmy K-PKE pracují s parametry, které jsou dány zvolenou bezpečnostní úrovní ML-KEM – v případě této implementace se jedná o bezpečnostní úroveň 3.

Prvním algoritmem schématu K-PKE je generování klíčů, implementované funkcí K_PKE_KeyGen(). Tato funkce má za úkol vygenerovat na základě vstupní 32bajtové náhodnosti *d* dvojici klíčů: veřejný šifrovací klíč *ekPKE* a soukromý dešifrovací klíč *dkPKE*. Veřejný klíč *ekPKE* bude později použit jako zapouzdřovací klíč v ML-KEM, zatímco *dkPKE* zůstává soukromý a slouží odpouzdření.

Funkce K_PKE_KeyGen() nejprve ze vstupní náhodnosti *d* odvodí pomocí *G* vstupní hodnoty ρ a σ. Počáteční hodnota ρ slouží k vygenerování matice *A* v NTT doméně, kde je využita pomocná funkce generate_matrix(). Z počáteční hodnoty σ se pak pomocí PRF a SamplePolyCBD() vygenerují tajný vektor *s* a šumový vektor *e*, které jsou obratem převedeny funkcí NTT() do NTT reprezentace. Následně se vypočítá vektor $\hat{t}$ pomocí rovnice

$$\hat{t} = \hat{A}\hat{s} + \hat{e}$$

Kde násobení a sčítání probíhá efektivně v NTT doméně. Nakonec se pomocí funkce ByteEncode() zakóduje $\hat{t}$ a připojí ρ pro vytvoření veřejného klíče ekPKE, a zakóduje s pro vytvoření soukromého klíče *dkPKE*.

Funkce K_PKE_Encrypt() implementuje K-PKE šifrování. Přijme veřejný klíč *ekPKE*, zprávu *m* a náhodnost *r* a vytvoří zašifrovaný text *c*. Nejprve z klíče *ekPKE* získá potřebný vektor $\hat{t}$ a počáteční hodnot ρ. Hodnotu ρ použije pomocná funkce generate_matrix() k regeneraci matic *A* v NTT doméně. Z náhodnosti *r* se pak pomocí PRF() a SamplePolyCBD() odvodí vektory *y*, e_1 a polynom e_2. Vektor *y* je převeden do NTT domény. Poté se pomocí *y*, matice *A*, vektoru $\hat{t}$ a šumových složek e_1, e_2 vypočítají vektory *u* a *v*. Do výpočtu *v* je také zakomponována zpráva *m*. Tyto výpočty zahrnují operace v NTT

doméně a zpětnou transformaci InvNTT. Výsledné vektory *u* a *v* jsou pak komprimovány pomoci funkce Compress() a spojeny do zašifrovaného textu *c*.

Funkce K_PKE_Decrypt() implementuje K-PKE dešifrování. Přijme soukromý klíč *dkPKE* a šifrovaný text *c* a vrátí původní zprávu *m*. Nejprve z šifrovaného textu *c* extrahuje a dekomprimuje pomocí funkcí ByteDecode() a Decompress() vektory *u′* a *v′*. Z klíče *dkPKE* dekóduje tajný klíč *s*. Následně vypočítá rozdílový polynom *w*: zkombinuje tajný klíč *s* s vektorem *u′* a výsledek odečte od *v′*. Tento výpočet využívá operace v NTT doméně a odečtení díky pomocné funkci _poly_sub(). Z výsledného polynomu *w* je nakonec pomocí Compress() a ByteEncode() získána původní zpráva *m*.

6.2.5 Implementace interních algoritmů

Tato kapitola se zaměřuje na implementaci interních algoritmů ML-KEM, specifikovaných v sekci 6 standardu FIPS 203 které jsou v této práci realizovány v souboru *kem_internal.py*. Tyto funkce představují základní stavební bloky hlavních ML-KEM operací a jejich klíčovou charakteristikou je deterministická povaha: negenerují žádnou vlastní náhodnost a jejich výstup je plně určen vstupy. Popisovaná implementace těchto algoritmů staví na dříve definovaných kryptografických primitivech, pomocných funkcích a schématu K-PKE a její korektnost je zásadní pro správnou funkci celého ML-KEM.

Funkce MLKEM768_KeyGen_internal() implementuje deterministické jádro generování klíčů pro ML-KEM. Na vstupu přijímá dvě 32bajtové náhodnosti, označované jako *d* a *z*, a vrací výsledný pár klíčů: zapouzdřující klíč *ek* a odpouzdřující klíč *dk*. Hlavním krokem funkce je zavolání již popsané funkce K_PKE_KeyGen s náhodností *d*, čímž se získá základní dvojice veřejného *ekPKE* a soukromého *dkPKE* klíče pro schéma K-PKE. Finální zapouzdřující klíč ML-KEM – *ek* je pak přímo roven veřejnému klíči K-PKE. Odpouzdřující klíč ML-KEM – *dk* je následně sestaven zřetězením čtyř částí: soukromého klíče K-PKE, veřejného klíče K-PKE, hashe veřejného klíče a druhé vstupní náhodnosti *z*. Funkce také obsahuje kontroly správné délky vstupních počáteční hodnoty *d* a *z*.

Funkce MLKEM768_Encaps_internal() implementuje deterministické jádro zapouzdření klíče pro ML-KEM. Na vstupu přijímá zapouzdřující klíč *ek* a 32bajtovou náhodnost *m* a vrací výsledný pár: 32bajtový sdílený tajný klíč *K* šifrovaný text *c*. Nejprve pomocí kryptografické funkce G() odvodí jak výsledný sdílený klíč *K*, tak interní 32bajtovou náhodnost *r*, která bude použita pro K-PKE šifrování. Vstupem pro funkci G() je přitom zřetězení vstupní náhodnosti *m* a hashe zapouzdřujícího klíče *ek*, který je vypočítán pomocí

funkce H(). Následně funkce zavolá již popsanou funkci K_PKE_Encrypt() se zapouzdřujícím klíčem *ek*, zprávou *m* a odvozenou náhodností *r*, čímž získá šifrovaný text *c*. Nakonec vrátí dvojici *K* a *c*. Implementace obsahuje také kontrolu správné délky vstupní náhodnosti *m*.

Funkce MLKEM768_Decaps_internal() implementuje deterministické jádro odpouzdření klíče. Na vstupu přijímá odpouzdřující klíč *dk* a šifrovaný text *c* a vrací výsledný 32bajtový sdílený klíč *K*. Funkce nejprve rozparsuje odpouzdřující klíč *dk* na jeho jednotlivé součásti: soukromý klíč *dkPKE*, veřejný klíč *ekPKE*, hash veřejného klíče *h* a hodnotu *z* pro implicitní zamítnutí. Poté pomocí K_PKE_Decrypt() dešifruje šifrovaný text *c*, čímž získá kandidáta na původní zprávu m'. Následně z m' a hashe *h* odvodí pomocí funkce G() kandidáta na sdílený klíč K' a náhodnost r' pro opětovné zapouzdření. Také vypočítá alternativní klíč $\bar{K}$ pro případ implicitního zamítnutí pomocí funkce J(). Poté provede kontrolní opětovné zapouzdření zprávy m' pomocí K_PKE_Encrypt() a získá rekonstruovaný šifrovaný text c'. Nakonec porovná pomocí pomocné funkce constant_time_compare() v konstantním čase původní šifrovaný text *c* s rekonstruovaným c'. Pokud se neshodují, nahradí kandidátní klíč K' alternativním klíčem $\bar{K}$. Vrácenou hodnotou je finální sdílený klíč K'.

6.2.6 Hlavní algoritmy ML-KEM

Tato podkapitola popisuje tři hlavní algoritmy mechanismu ML-KEM, jejichž implementace je realizována v souboru *mlkem768.py*. Tyto funkce představují finální vnější rozhraní celého schématu, které je určeno pro koncové použití. Jak bylo naznačeno dříve, tyto hlavní algoritmy volají příslušné interní funkce pro provedení samotných kryptografických operací. Navíc oproti interním funkcím zajišťují generování potřebné náhodnosti a provádějí nezbytné kontroly vstupních parametrů, jak vyžaduje standard FIPS 203. Protože jádro výpočtů již bylo rozebráno v předchozích částech, následující popisy se zaměří především na tyto dodatečné kroky a celkové propojení.

Funkce MLKEM768_KeyGen() představuje hlavní rozhraní pro generování klíčů. Tato funkce nepřijímá žádné vstupní argumenty, ale sama interně generuje potřebnou náhodnost pomocí kryptograficky bezpečného generátoru náhodných čísel. Konkrétně vygeneruje dvě 32bajtové náhodné hodnoty *d* a *z*. Tyto hodnoty poté předá jako vstup interní funkci MLKEM768_KeyGen_internal(), která provede veškeré kryptografické výpočty a vrátí dvojici klíčů. Funkce MLKEM768_KeyGen() pak tuto dvojici klíčů *ek* a *dk* pouze vrátí jako

svůj výsledek. Implementace zahrnuje také základní ošetření případných chyb při generování náhodnosti.

Funkce MLKEM768_Encaps() představuje hlavní rozhraní pro zapouzdření klíče ML-KEM. Na vstupu přijímá pouze veřejný zapouzdřující klíč *ek* a jejím výstupem je dvojice: 32bajtový sdílený tajný klíč *K* a šifrovaný text *c*. Než dojde k samotnému zapouzdření, funkce nejprve provede validaci vstupního klíče *ek* pomocí pomocné funkce _validate_encapsulation_key(). Tato validace ověřuje správnou délku klíče a také kontroluje, zda je klíč správně zakódován podle pravidel standardu. Pokud klíč validací neprojde, funkce selže. V opačném případě interně vygeneruje 32bajtovou kryptograficky bezpečnou náhodnost *m*. Tuto náhodnost *m* spolu s validovaným klíčem *ek* následně předá interní funkci MLKEM768_Encaps_internal(), která provede jádro kryptografických výpočtů. Funkce MLKEM768_Encaps pak pouze vrátí výsledný pár *K* – sdílené tajemství, *c* – zašifrovaný text.

Funkce MLKEM768_Decaps() představuje hlavní rozhraní pro odpouzdření klíče. Na vstupu přijímá soukromý odpouzdřující klíč *dk* a šifrovaný text *c* a jejím výstupem je výsledný 32bajtový sdílený tajný klíč *K*. Než dojde k samotnému odpouzdření, funkce nejprve provede validaci vstupů *dk* a *c* pomocí pomocné funkce _validate_decaps_inputs(). Tato validace ověřuje, zda mají *dk* a *c* správné očekávané délky pro danou bezpečnostní úroveň a také kontroluje vnitřní konzistenci klíče *dk* pomocí uloženého hashe. Pokud vstupy validací neprojdou, funkce selže. V opačném případě předá validované vstupy *dk* a *c* interní funkci MLKEM768_Decaps_internal(), která provede jádro opouzdřujícího výpočtu včetně mechanismu implicitního zamítnutí. Funkce MLKEM768_Decaps() pak pouze vrátí výsledný sdílené tajemství *K* získaný z interní funkce.

6.3 Návrh TLS-like protokolu

Pro ukázku reálné implementace vlastní verze hybridního algoritmu X25519MLKEM768 byl navržen a realizován protokol inspirovaný standardem TLS, který se běžně používá pro zabezpečenou komunikaci na internetu, například v rámci protokolu HTTPS. V souboru *tls_core.py* je kombinován klasický přístup založený na eliptických křivkách X25519 s postkvantovým algoritmem ML-KEM768, a to za účelem výpočtu sdíleného tajemství mezi klientem a serverem. Princip fungování této výměny odpovídá popisu uvedenému v kapitole 5.2.

Pomocí funkce client_create_hello() klient inicializuje navázání spojení se serverem. Během této operace jsou vytvořeny tři páry klíčů – dvojice pro hybridní výměnu a jeden záložní pár pro klasickou výměnu pomocí X25519. Výsledná zpráva ClientHello obsahuje náhodně generované ID relace, seznam podporovaných skupin pro výměnu klíčů – klasický nebo hybridní přístup a dvojici sdílených klíčových hodnot. Prvním z nich je zřetězený veřejný klíč X25519 a ML-KEM768 určený pro hybridní výměnu, druhým samostatný veřejný klíč X25519 pro případ, že server hybridní režim nepodporuje.

Funkce server_process_hello() zajišťuje zpracování zprávy ClientHello, kterou server obdrží od klienta. Nejprve proběhne výběr jedné ze skupin pro výměnu klíčů, které obě strany podporují. Pokud je zvolen hybridní přístup X25519MLKEM768, server extrahuje z kombinovaného sdíleného klíče veřejný klíč ML-KEM768 a veřejný klíč X25519. Na základě těchto údajů vygeneruje svůj vlastní pár klíčů X25519 a následně provede výpočet dvou dílčích tajemství – jedno pomocí X25519 a druhé pomocí zapouzdření KEM. Oba výsledky jsou zřetězeny a následně zpracovány funkcí HKDF za účelem získání finálního sdíleného tajemství. Server poté vytvoří odpověď ServerHello, která obsahuje vybranou skupinu, výsledné sdílené tajemství a zřetězený sdílený klíč tvořený jeho veřejným klíčem X25519 a šifrovaným textem vzniklým při zapouzdření KEM. Pokud je místo hybridní skupiny vybrána pouze klasická skupina X25519, server provede standardní výměnu klíčů pomocí ECDH a odpověď obsahuje pouze veřejný klíč X25519 a výsledné tajemství.

Funkce client_process_server_hello() zpracovává odpověď serveru a na straně klienta dopočítává finální sdílené tajemství. V případě hybridní výměny klient rozdělí přijatý zřetězený sdílený klíč na dvě části: veřejný klíč serveru X25519 a šifrovaný text získaný zapouzdřením postkvantového klíče ML-KEM768. Klient nejprve spočítá sdílené tajemství pomocí algoritmu X25519, následně pomocí funkce decapsulate() získá druhou část tajemství z šifrovaného textu a svého soukromého ML-KEM768 klíče. Obě hodnoty spojí a vstupem do funkce HKDF() získá finální sdílený klíč. V případě klasického režimu X25519 proběhne pouze výpočet sdíleného tajemství metodou ECDH() s použitím klientova soukromého a server veřejného klíče.

6.4 Implementace protokolu

V této kapitole je popsána praktická implementace navrženého TLS-like protokolu, který využívá hybridní algoritmus nebo klasický algoritmus pro bezpečnou výměnu klíčů mezi klientem a serverem. Základní funkcionalita výměny klíčů je soustředěna v modulu

tls_core.py, zatímco samotné navázání síťového spojení, výměna zpráv a šifrování komunikace probíhá ve skriptech *client.py* a *server.py*. Obě strany navazují spojení prostřednictvím TCP socketů a využívají algoritmus AES-GCM (Galois/Counter Mode) pro šifrování a autentizaci zpráv.

Soubor *client.py* tvoří klientskou část TLS-like protokolu a slouží k navázání šifrovaného spojení se serverem. Implementace využívá standardní knihovnu jazyka Python pro práci se síťovými sockety (*socket*) a pro výměnu data ve formátu JavaScript Object Notation (JSON). Díky těmto knihovnám je možné přenášet strukturované zprávy mezi klientem a serverem přes TCP spojení.

Na začátku skriptu je definována Internet Protocol (IP) adresa a port serveru, ke kterému se klient připojuje. V rámci lokálního testování je IP adresa nastavena na 127.0.0.1 (localhost) a port na hodnotu 4443. Tato konfigurace zajišťuje, že komunikace probíhá výhradně lokálně na daném zařízení a je určena pro účely testování.

Klient po spuštění naváže síťové spojení se serverem a inicializuje výměnu klíčů pomocí funkce client_create_hello() z modulu *tls_core.py*. Tato funkce vytvoří potřebné klíčové páry a připraví zprávu ClientHello, která obsahuje informace o podporovaných kryptografických skupinách a veřejných klíčích klienta. Tato zpráva je serializována do formátu JSON a odeslána serveru.

Následně klient přijme odpověď ServerHello, kterou zpracuje pomocí funkce client_process_server_hello() ze stejného modulu. Tato funkce na základě zvolené kryptografické skupiny vypočítá sdílené tajemství, které je následně použito jako klíč pro šifrování zpráv.

Po úspěšném navázání spojení klient nejprve přijme šifrovanou zprávu od serveru a dešifruje ji pomocí algoritmu AES-GCM. Tento úvodní příjem zprávy slouží jako testovací krok pro ověření, že sdílené tajemství bylo správně vypočítáno a že následná symetrická šifrovací komunikace funguje. Pro šifrování a dešifrování zpráv je využita knihovna *cryptography*, která v jazyce Python poskytuje rozhraní pro bezpečné symetrické šifrování. Následuje interaktivní režim, ve kterém uživatel klienta může prostřednictvím konzole zadávat vlastní textové zprávy. Tyto zprávy jsou šifrovány, odeslány serveru, a následně je očekávána odpověď. Komunikace probíhá v režimu simplexního střídání – vždy je odeslána jedna zpráva a klient musí nejprve obdržet odpověď od serveru, než může pokračovat v dalším odesílání.

Soubor *server.py* tvoří druhou polovinu protokolu a zajišťuje serverovou stranu šifrované komunikace. Stejně jako klientská část využívá standardní knihovnu *socket* pro práci s TCP sockety a knihovnu *json* pro serializaci dat. Server naslouchá na zvoleném portu, ve výchozím nastavení 4443, a čeká na navázání spojení klientem.

Po přijetí spojení server obdrží zprávu ClientHello, kterou zpracuje pomocí funkce server_process_hello() z modulu *tls_core.py*. Tato funkce zvolí odpovídající kryptografickou skupinu, extrahuje veřejné klíče z přijatého sdíleného klíče a následně provede výpočet sdíleného tajemství. Při použití hybridního algoritmu je sdílený klíč rozdělen na veřejný klíč ML-KEM768 a veřejný klíč X25519, přičemž server provede zapouzdření KEM a výpočet X25519. Obě hodnoty jsou spojeny a zpracovány pomocí HKDF.

Po vytvoření odpovědi ServerHello je tato zpráva serializována a odeslána klientovi. Server následně odešle testovací šifrovanou zprávu, která slouží k ověření, že klient je schopen správně dešifrovat zprávu pomocí sdíleného tajemství. Pro šifrování zpráv server rovněž využívá algoritmus AES-GCM prostřednictvím knihovny *cryptography*.

Po odeslání úvodní zprávy vstupuje server do interaktivního režimu. V tomto režimu přijímá od klienta šifrované zprávy, které dešifruje a vypisuje na konzoli. Uživatel na straně serveru může následně zadat odpověď, která je zašifrována a odeslána zpět klientovi. Komunikace probíhá střídavě, jak již bylo uvedeno v části popisující chování klienta.

6.5 Uživatelské rozhraní

Pro účely lepší demonstrace a ověření výsledků byl vytvořen grafické uživatelský rozhraní (GUI) samostatně pro serverovou (*gui_server.py*) i klientskou část (*gui_client.py*) aplikace. Obě rozhraní slouží jako jednoduché demo, které umožňuje snadnější ovládání a vizualizaci základních funkcí navrženého systému. K realizaci GUI byla použita knihovna *customtkinter*, která oproti klasické verzi knihovny *tkinter* umožňuje snadnější úpravu vzhledu, lepší možnosti stylování a modernější grafické prvky. Princip ovládání a celková logika aplikace zůstávají obdobné jako v původní konzolové verzi.

Při spuštění aplikace si uživatel nejprve zvolí požadovaný režim komunikace. K dispozici je buď hybridní režim, kdy jsou zprávy šifrovány kombinací algoritmů X25519 a ML-KEM768, nebo klasický režim využívající pouze X25519 (Obrázek 9 a Obrázek 10).

Zvolený režim ovlivňuje průběh navázání spojení i způsob šifrování následné komunikace mezi klientem a serverem.

Obrázek 9 Výběr režimu komunikace v klientské aplikaci

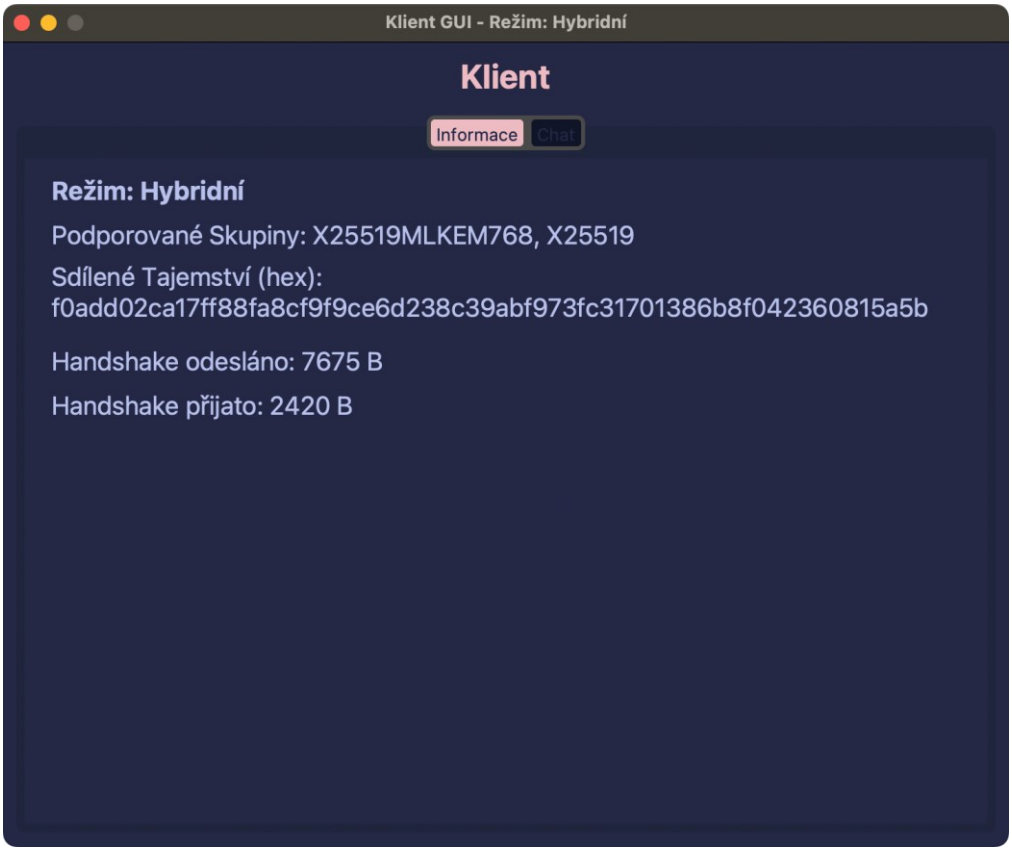
(zdroj: vlastní)

Obrázek 10 Výběr režimu komunikace v serverové aplikaci

(zdroj: vlastní)

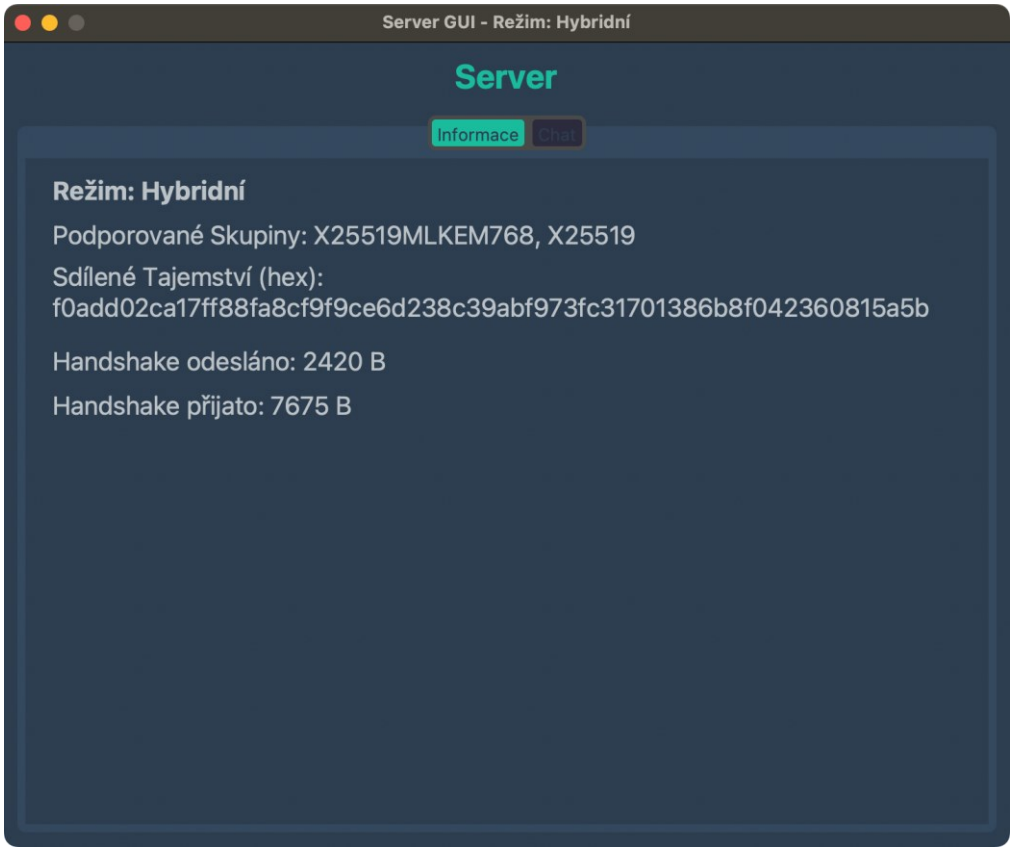
Po výběru režimu komunikace je automaticky navázáno spojení mezi serverovou a klientskou aplikací. Uživatelské rozhraní je rozděleno do dvou hlavních sekcí – Chat a Informace. V záložce Informace (Obrázek 11 a Obrázek 12) je zobrazen zvolený režim

(hybridní nebo klasický), včetně konkrétního algoritmu, který aplikace podporuje, buď hybridní X25519MLKEM768, nebo klasický X25519. V případě zvoleného hybridního režimu aplikace zároveň podporuje i čistě klasický režim X25519 pro situace, kdy druhá strana hybridní výměnu klíčů nepodporuje. Po úspěšném navázání spojení je zde také vypsáno vypočítané sdílené tajemství, což umožňuje ověřit, že server i klient disponují totožným klíčem. Pod touto hodnotou jsou zobrazeny údaje o objemu odeslaných a přijatých dat během fáze navazování spojení. Podrobnější analýza rozdílů v datové režii při navazování spojení je uvedena v kapitole č.7.3 Testování protokolu.



Obrázek 11 Záložka Informace v klientské aplikaci

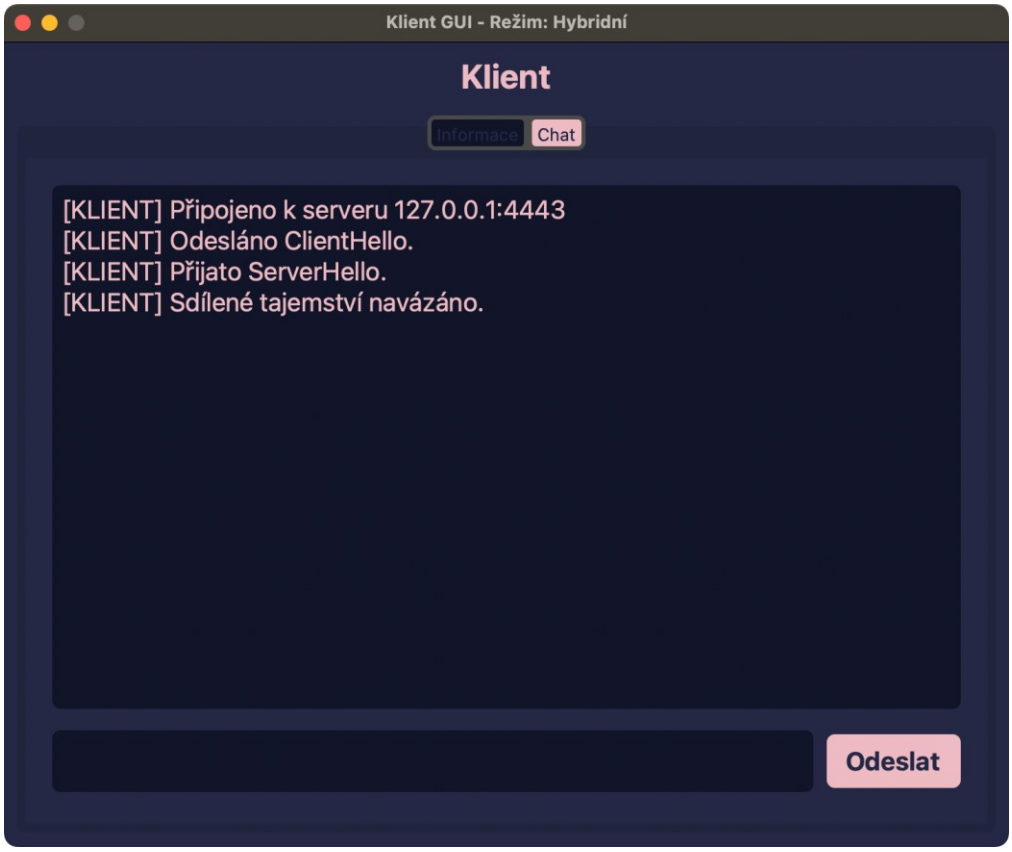
(zdroj: vlastní)



Obrázek 12 Záložka Informace v serverové aplikaci

(zdroj: vlastní)

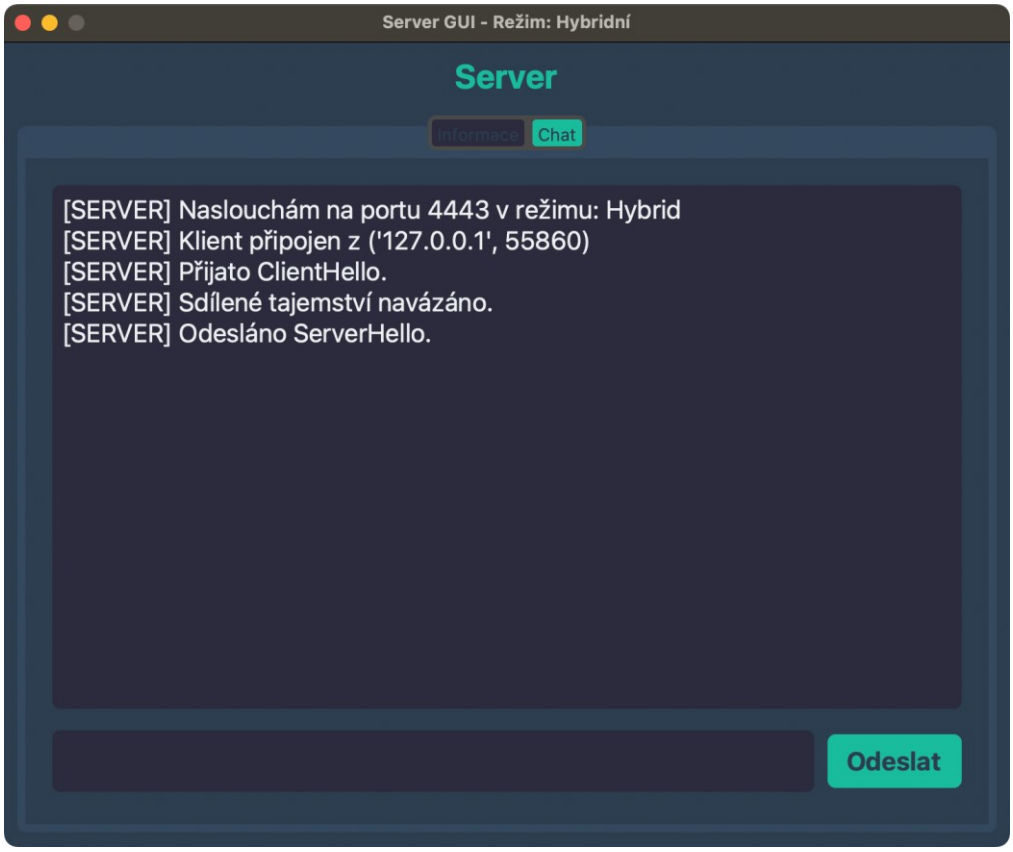
V záložce Chat klientské aplikace (Obrázek 13) jsou zobrazovány všechny důležité informace týkající se navazování spojení a průběhu komunikace se serverem. Uživatel zde vidí IP adresu a port serveru, ke kterému se připojuje, zprávu o odeslání *ClientHello*, následné přijetí *ServerHello* a potvrzení, že sdílené tajemství bylo úspěšně vypočteno. Tyto záznamy umožňují jednoduše ověřit správný průběh celého kryptografického procesu i samotného spojení. Po úspěšném navázání spojení tato záložka nejen zobrazuje všechny odesílané i přijímané zprávy v rámci zašifrované komunikace, ale zároveň umožňuje uživateli přímo odesílat nové zprávy a aktivně tak komunikovat se serverem.



Obrázek 13 Záložka Chat v klientské aplikaci

(zdroj: vlastní)

V serverové části aplikace jsou v záložce Chat (Obrázek 14) postupně zaznamenávány události od naslouchání na zvoleném portu, přes informaci o připojení klienta a jeho IP adrese, přijetí zprávy *ClientHello*, výpočet sdíleného tajemství až po odeslání *ServerHello*. Záložka umožňuje také přehledně sledovat celou historii komunikace a po úspěšném navázání spojení nejen zobrazuje všechny zprávy v rámci zašifrované komunikace mezi klientem a serverem, ale zároveň umožňuje uživateli přímo odesílat nové zprávy a aktivně tak komunikovat s protistranou.



Obrázek 14 Záložka Chat v serverové aplikaci

(zdroj: vlastní)

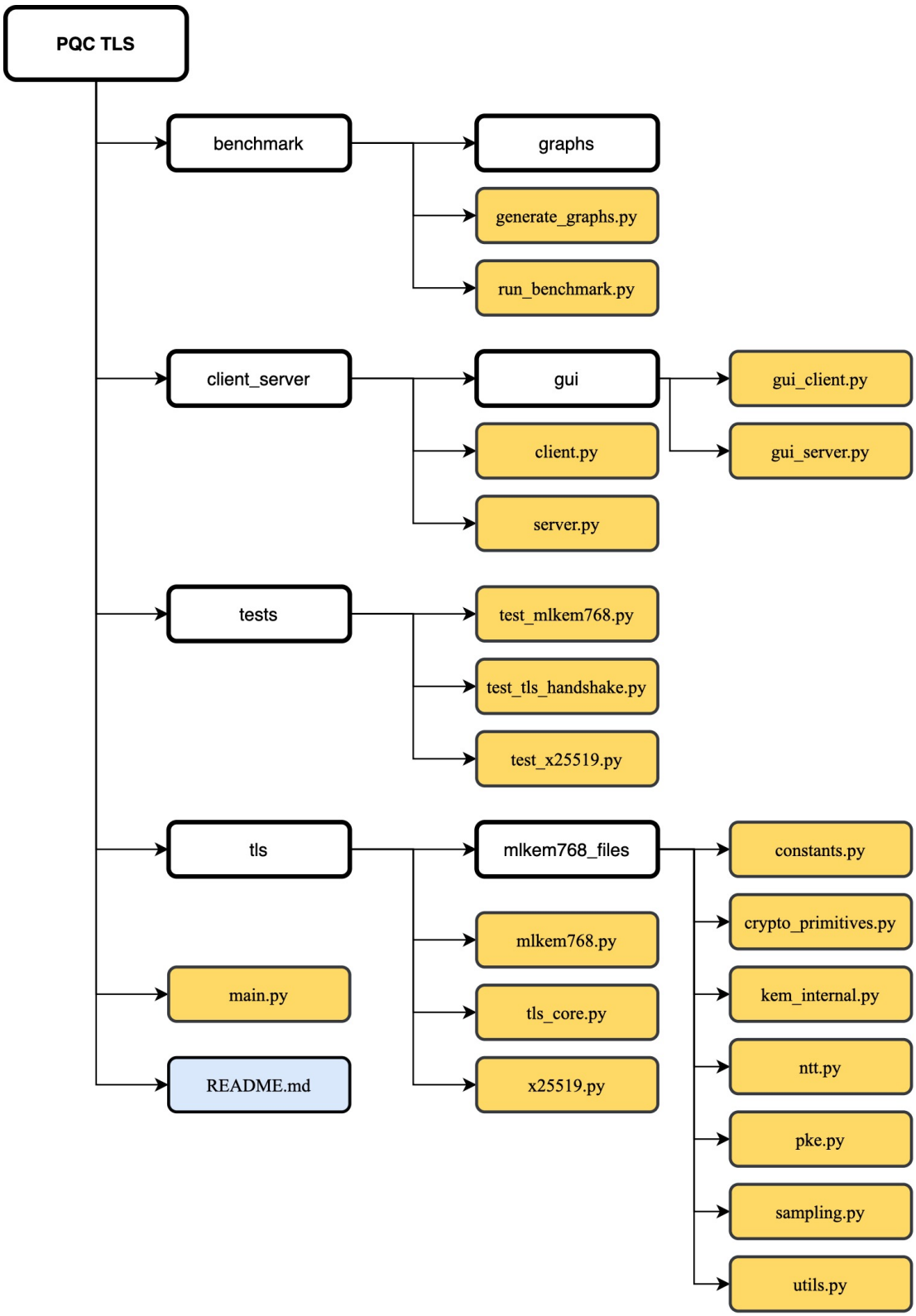
Grafické uživatelské rozhraní významně usnadnilo ověření správné funkce implementovaných algoritmů a umožnilo názorně demonstrovat šifrovanou komunikaci mezi klientem a serverem. Díky rozdělení na část Chat a Informace bylo možné jednoduše sledovat klíčové operace i přenosy dat během spojení. GUI tak efektivně podpořilo testování a prezentaci výsledků práce.

6.6 Struktura a spuštění projektu

Při spuštění projektu prostřednictvím hlavního souboru *main.py* umístěného v kořenové složce projektu je spuštěna jak serverová, tak klientská část aplikace. Pro zajištění paralelního běhu obou částí je využívána knihovna *multiprocessing*, což umožňuje více vláknové spouštění.

Při spuštění aplikace je nejprve nutné zvolit režim serverové aplikace a až následně režim klientské. Tím je zajištěno, že serverová část bude připravena pro navázání spojení s klientem. Po zvolení těchto režimů je spojení mezi klientskou a serverovou částí navazováno automaticky.

Podrobný popis postupu spuštění a seznam požadovaných balíčků včetně návodu na jejich instalaci je uveden v souboru README.md, který je součástí projektu.



Obrázek 15 Diagram struktury projektu

(zdroj: vlastní)

7 TESTOVÁNÍ IMPLEMENTACE

V této kapitole jsou prezentovány výsledky testování implementovaných algoritmů a navrženého TLS-like protokolu. Provedené testy zahrnují ověření správnosti implementace pomocí srovnání výsledků s oficiálními testovacími vektory a referenčními implementacemi, a dále výkonnostní testování zaměřené na měření rychlosti generování klíčových párů, výpočtu sdíleného tajemství, operací zapouzdření a odpouzdření, i celkového trvání handshake. Součástí testování bylo také vyhodnocení velikosti přenášených dat při komunikaci.

Každý výkonnostní test byl proveden opakovaně (100 měření) a výsledky jsou uváděny jako průměrné hodnoty pro minimalizaci vlivu náhodných odchylek. Pro automatizaci testování byl použit vlastní skript, který všechna měření zajišťuje a ukládá výsledky ke zpracování. Výsledky jsou prezentovány v tabulkách a grafech umožňujících přehledné srovnání jednotlivých implementací.

Testování probíhalo na zařízení MacBook Pro s procesorem Apple M1 Pro, operačním systému macOS Sequoia (verze 15.4.1), v prostředí Python 3.12.

7.1 Testování X25519

Tato část se věnuje testování implementace algoritmu X25519. Testování zahrnuje ověření správnosti výměny klíčů a porovnání výsledků s oficiální knihovní implementací. Následně je provedeno také výkonnostní srovnání obou variant.

7.1.1 Test správnosti

Pro ověření správnosti a funkčnosti implementace algoritmu X25519 byl vytvořen testovací skript *test_x25519.py*, který obsahuje dvě samostatné testovací funkce.

První funkce testu simuluje výměnu klíčů mezi dvěma stranami. Každá strana si vygeneruje vlastní klíčový pár a následně pomocí své privátní hodnoty a veřejné hodnoty druhé strany spočítá sdílené tajemství. Test ověřuje, zda jsou obě vypočtená sdílená tajemství identická, jak předpokládá Diffie-Hellmanův princip. Tento test potvrzuje, že implementace dokáže správně provádět výměnu klíčů.

Druhá funkce v testu porovnává výsledky naší implementace s výsledky oficiální knihovní implementace X25519 z Python balíčku *cryptography*. Pro náhodně vygenerovaný vstupní skalár a veřejný bod se provede výpočet sdíleného tajemství jak pomocí vlastní

funkce x25519(), tak pomocí knihovny. Shoda výsledků potvrzuje korektnost implementace algoritmu.

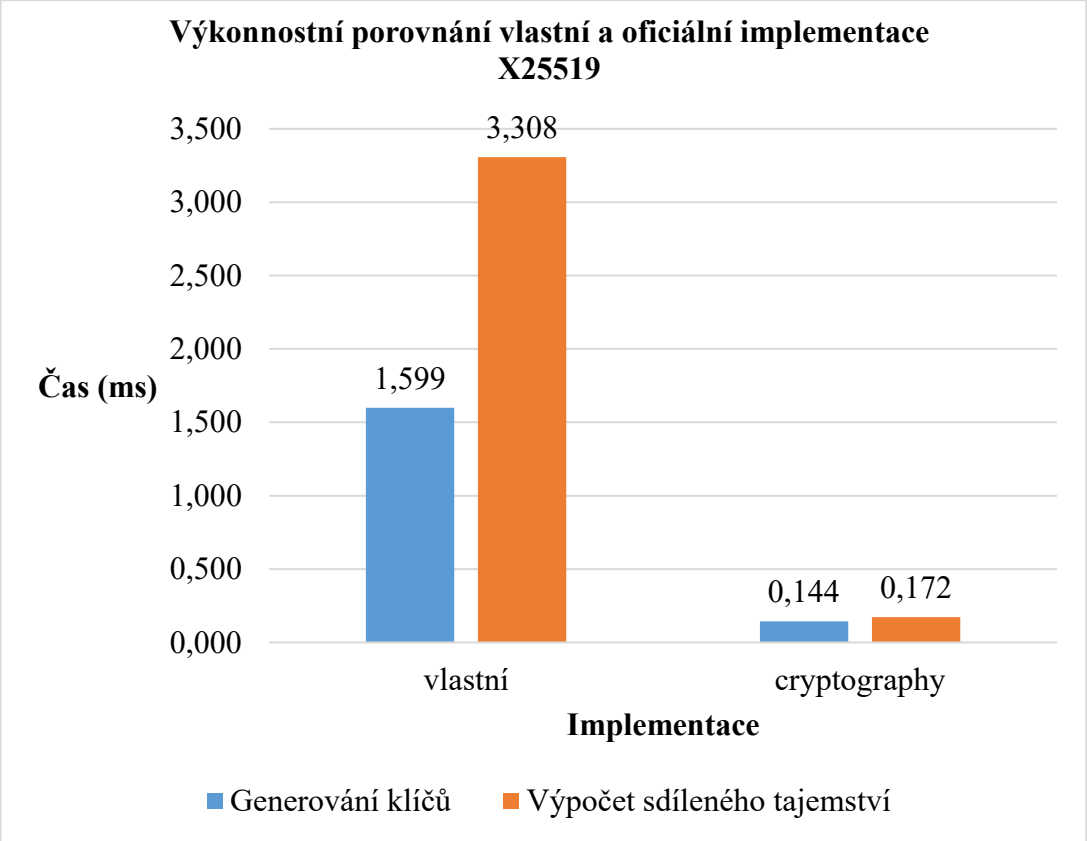
7.1.2 Výkonnostní testování

V této části je provedeno výkonnostní porovnání vlastní implementace algoritmu X25519 a oficiální implementace dostupné v knihovně *cryptography*. Měření zahrnuje průměrné časy generování klíčových párů a výpočtu sdíleného tajemství.

Tabulka 13 Výkonnostní porovnání vlastní a oficiální implementace X25519

Implementace	Generování klíčů (ms)	Výpočet sdíleného tajemství (ms)
vlastní	1,599	3,308
cryptography	0,144	0,172

(zdroj: vlastní)



Obrázek 16 Graf výkonnostní porovnání vlastní a oficiální implementace X25519

(zdroj: vlastní)

Jak lze vidět v tabulce (Tabulka 13) i v grafu (Obrázek 16), knihovna *cryptography* dosahuje výrazně vyšší výkonnosti než vlastní implementace algoritmu X25519. Podle dat uvedených v tabulce je generování klíčového páru v knihovně *cryptography* přibližně 11krát rychlejší než ve vlastní implementaci a výpočet sdíleného tajemství je dokonce více než 19krát rychlejší. Tyto rozdíly jsou způsobeny tím, že *cryptography* využívá optimalizované nativní implementace v jazyce C, zatímco vlastní verze byla implementována výhradně v jazyce Python bez dalších optimalizací.

7.2 Testování MLKEM768

Tato část se zabývá popisem testování vlastní implementace algoritmu ML-KEM768. V rámci testů byla posuzována jak správnost implementace, tak výkonnost jednotlivých operací – konkrétně generování klíčů, zapouzdření a odpouzdření. Výsledky byly porovnávány s referenční implementací v jazyce Python i s oficiální implementací v jazyce C, aby bylo možné objektivně zhodnotit funkčnost i efektivitu navrženého řešení.

7.2.1 Test správnosti

Pro ověření správnosti a funkčnosti implementace algoritmu ML-KEM768 byl vytvořen testovací skript *test_mlkem768.py*. Tento skript obsahuje sadu testovacích funkcí, které samostatně ověřují jednotlivé moduly a jejich implementované funkce – například převody bajtů, hashovací a kryptografické primitivy, vzorkování, NTT, šifrování a dešifrování, interní funkce KEM i kompletní API rozhraní.

Následně byly porovnávány výstupy hlavních deterministických funkcí vlastní implementace s výstupy referenční knihovny *kyber-py*, avšak nebyla zjištěna shoda. Proto byly dále hledány oficiální KAT testy, které se však nepodařilo dohledat, a dostupné testovací vektory rovněž neodpovídaly hodnotám generovaným implementací. Navíc v průběhu ladění byly zjištěny dílčí nesrovnalosti v knihovně *kyber-py* (například odlišnost v hashovací funkci H(), kde je výsledku přičítán navíc jeden bajt), což komplikovalo přímé porovnání. Porovnání s oficiální implementací v jazyce C bylo omezené kvůli uzavřenosti interních funkcí a využití interního generátoru náhodných čísel, což znemožnilo detailní kontrolu mezivýsledků.

V průběhu analýzy bylo zjištěno, že v implementaci dochází k nesprávnému generování K-PKE klíčů, což se odráží v odlišných výsledcích při porovnávání s testovacími vektory i s výstupy knihovny *kyber-py*. Tato chyba se projevuje zejména ve fázi generování

klíčového páru, kde některé hodnoty neodpovídají očekávaným výsledkům dle specifikace algoritmu ML-KEM768.

Navzdory těmto obtížím jsou klíče i zašifrované texty generovány ve správné délce, je umožněno korektní šifrování a dešifrování zpráv a všechny základní operace fungují v souladu se specifikací.

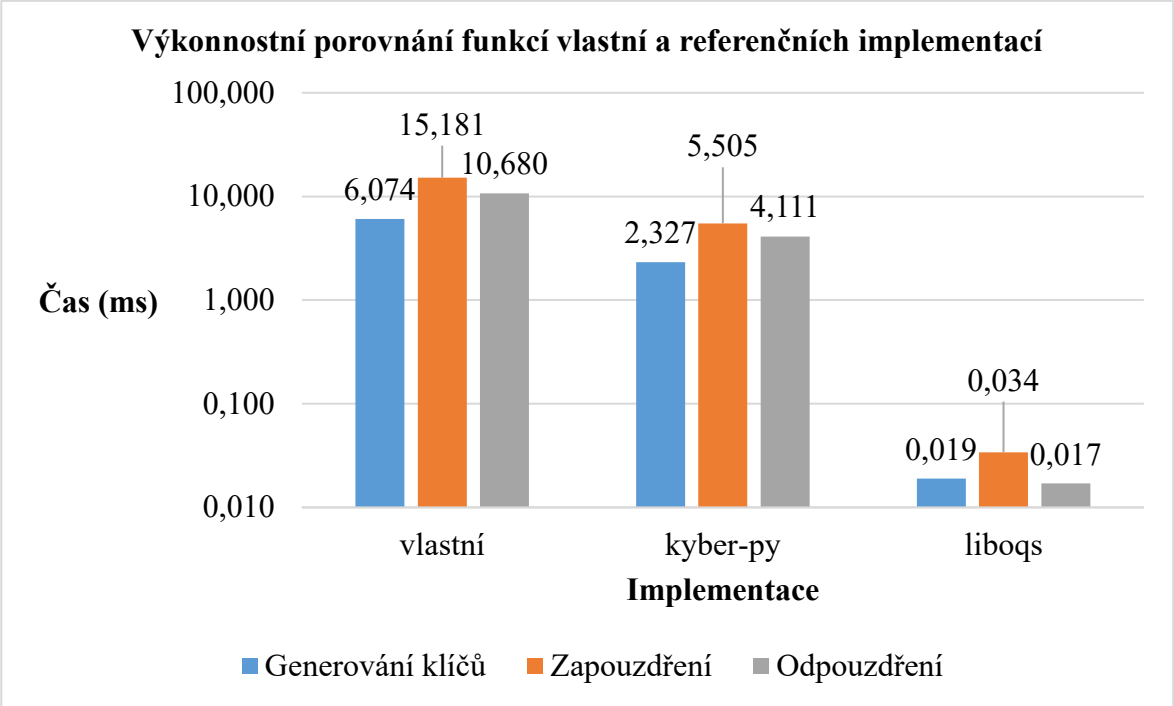
7.2.2 Výkonnostní testování

V této části je analyzována výkonnost algoritmu ML-KEM768. Testování zahrnuje tři základní operace algoritmu – generování klíčového páru, zapouzdření sdíleného tajemství a jeho následné odpouzdření. Výkon vlastní implementace byl porovnáván s další čistě Pythonovou implementací využívající knihovnu *kyber-py*, která poskytuje funkční, ale neoptimalizovanou verzi algoritmu. Dále byl výkon porovnán s oficiální referenční implementací využívající knihovnu *liboqs*, která poskytuje optimalizované implementace v jazyce C zpřístupněné prostřednictvím Python wrapperu *oqs-python*. Tímto způsobem bylo možné srovnat tři úrovně implementací – vlastní studijní řešení, existující Pythonovou knihovnu a vysoce optimalizovanou C implementaci.

Tabulka 14 Výkonnostní porovnání funkcí vlastní a referenčních implementací ML-KEM768

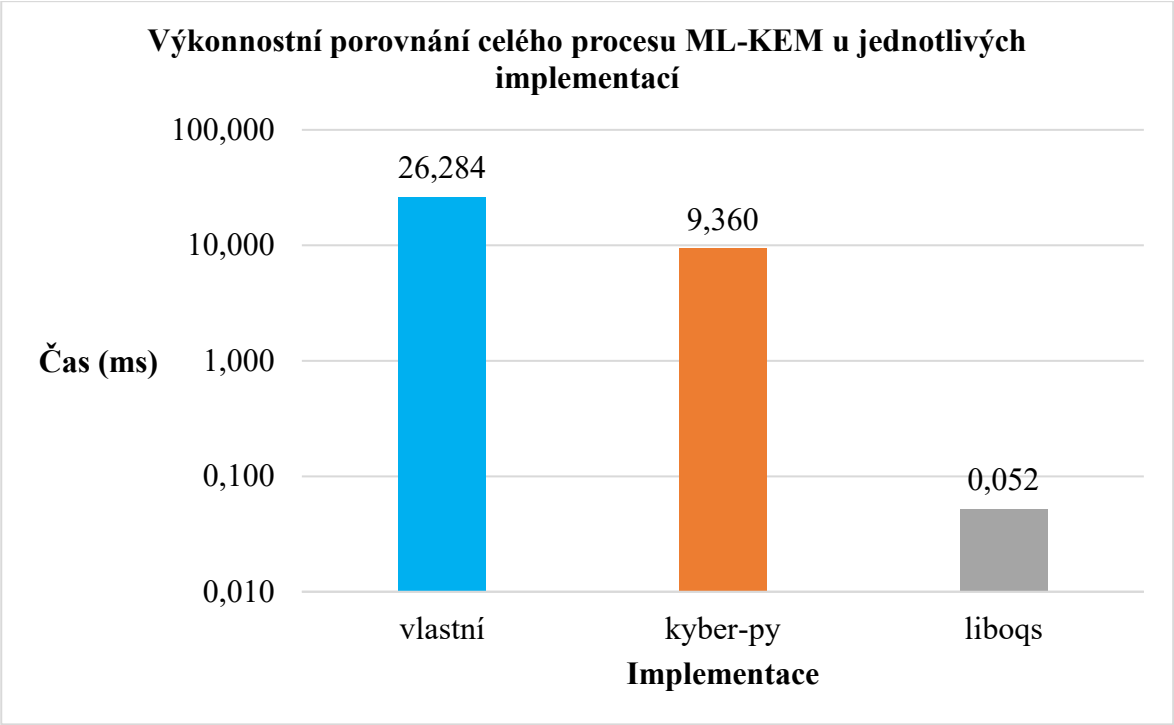
Implementace	**Generování klíčů (ms)**	**Zapouzdření (ms)**	**Odpouzdření (ms)**	**Celý KEM proces (ms)**
vlastní	6,074	15,181	10,680	26,284
kyber-py	2,327	5,505	4,111	9,360
liboqs	0,019	0,034	0,017	0,052

(zdroj: vlastní)



Obrázek 17 Graf Výkonnostní porovnání funkcí ML-KEM u jednotlivých implementací

(zdroj: vlastní)



Obrázek 18 Graf Výkonnostní porovnání celého procesu ML-KEM u jednotlivých implementací

(zdroj: vlastní)

Výsledky uvedené v tabulce (Tabulka 14) a grafech (Obrázek 17 a 18) ukazují výrazné rozdíly ve výkonnosti jednotlivých implementací algoritmu ML-KEM768. Nejpomalejší je vlastní implementace v jazyce Python, u které celý proces trvá přibližně trojnásobek času oproti čistě Pythonové knihovně *kyber-py*. Výrazný nárůst výkonu je vidět při použití knihovny *liboqs*, která díky optimalizované C implementaci dosahuje více než 500násobného zrychlení oproti vlastní implementaci. Použití vysoce optimalizovaných knihoven je tak klíčové pro praktické nasazení kvantově odolných algoritmů v reálných systémech, kde je vysoká výkonnost nezbytná.

7.3 Testování protokolu

Tato část se zaměřuje na ověření správnosti a výkonnosti navrženého TLS-like protokolu v různých režimech výměny klíčů. Testování zahrnuje simulaci navazování zabezpečeného spojení mezi klientem a serverem, kde je posuzována shoda vypočítaných sdílených tajemství i porovnání výkonnostních parametrů, jako je délka handshaku a objem přenesených dat. Výsledky umožňují posoudit dopad použití postkvantových technik na efektivitu a datovou náročnost zabezpečené komunikace.

7.3.1 Testování správnosti

Pro účely ověření před samotným použitím v síťové komunikaci byl vytvořen testovací skript *test_tls_handshake.py*. Tento soubor simuluje proces navázání zabezpečeného spojení mezi klientem a serverem pro obě varianty – jak hybridní výměnu klíčů, tak klasickou výměnu. V každé simulaci je ověřeno, že obě strany nezávisle vypočítají totožné sdílené tajemství, čímž je potvrzena správnost celé výměny. Testování probíhá výhradně lokálně v rámci jednoho skriptu, bez použití skutečné síťové komunikace.

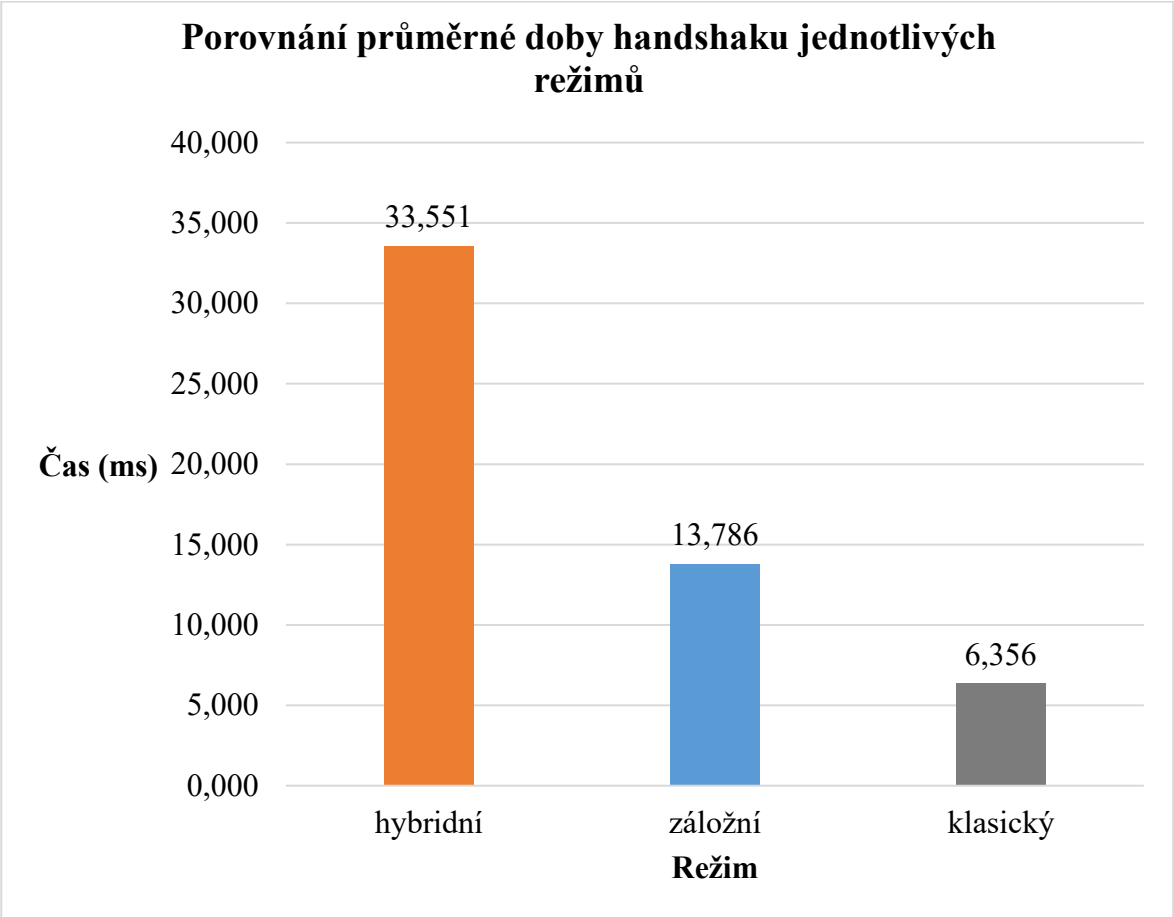
7.3.2 Výkonnostní testování

V této části jsou prezentovány výsledky výkonnostního testování tří režimů TLS-like protokolu. Měřeny byly všechny klíčové varianty: hybridní handshake využívající kombinaci algoritmů X25519 a ML-KEM768, záložní handshake, kdy klient podporuje hybridní výměnu, ale server podporuje pouze klasický X25519, a nakonec čistý handshake realizovaný pouze pomocí algoritmu X25519 na obou stranách. Cílem testování bylo porovnat průměrné časy trvání jednotlivých handshake procesů a analyzovat vliv použití postkvantových technik na celkovou rychlost navázání šifrovaného spojení.

Tabulka 15 Porovnání průměrné doby handshaku jednotlivých režimů

Režim	Průměrný čas handshaku (ms)
hybridní	33,551
záložní	13,786
klasický	6,356

(zdroj: vlastní)



Obrázek 19 Graf porovnání průměrné doby handshaku jednotlivých režimů

(zdroj: vlastní)

Jak ukazují výsledky v tabulce (Tabulka 15) a grafu (Obrázek 19), hybridní handshake využívající postkvantový algoritmus ML-KEM768 je výrazně pomalejší než klasický režim postavený čistě na algoritmu X25519 – konkrétně je přibližně 5krát pomalejší. Zajímavý je také mezistupeň záložní režim, kdy klient podporuje hybridní výměnu, ale server pouze klasickou. Tento režim je přibližně 2krát pomalejší než klasický handshake, ale stále výrazně

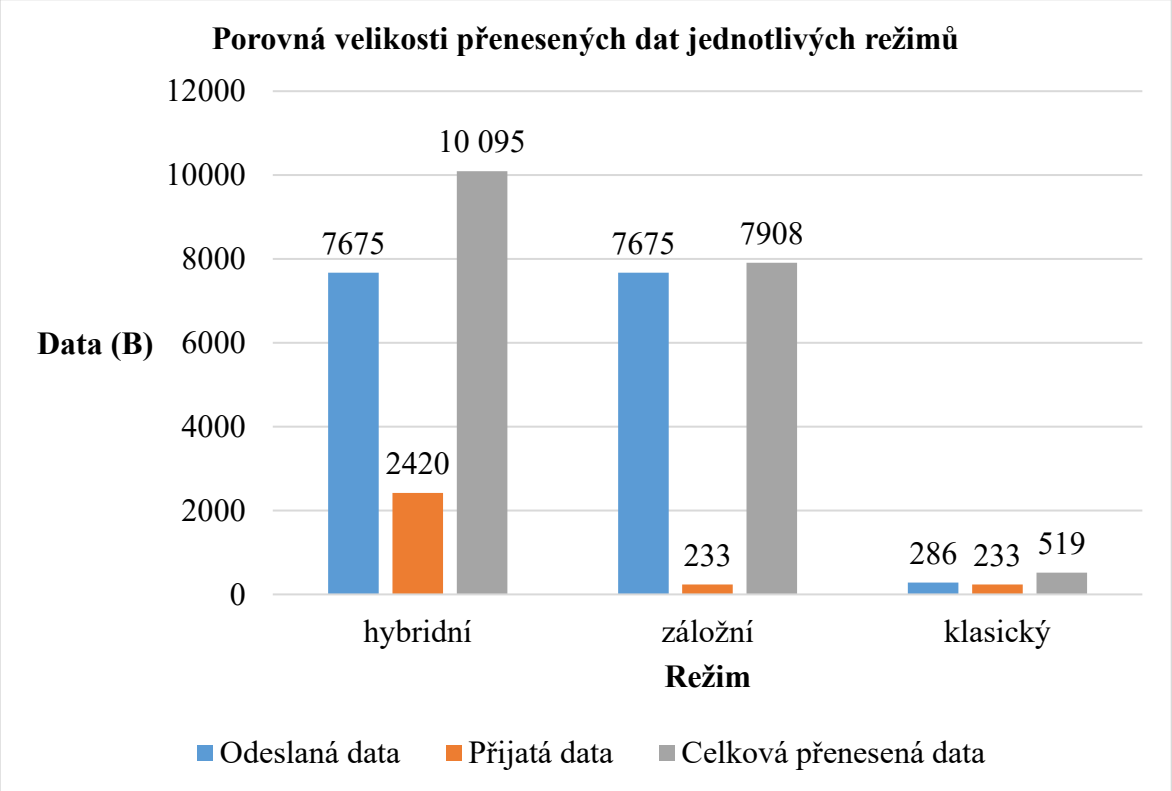
rychlejší než plně hybridní varianta. Nižší výkon záložního režimu je způsoben tím, že klient i v tomto případě generuje klíče pro ML-KEM768, které však nejsou serverem využity. Výsledky jednoznačně ukazují výkonnostní náklady spojené s nasazením postkvantových algoritmů v rámci TLS-like protokolu.

Dále byla změřena velikost přenesených dat v jednotlivých režimech handshake. Byla zaznamenána velikost odeslaných dat ze strany klienta, přijatých dat od serveru a jejich součet, který představuje celkový objem přenesených dat během fáze navazování spojení.

Tabulka 16 Porovnání velikosti přenesených dat během handshake v jednotlivých režimech

Režim	Odeslaná data (B)	Přijatá data (B)	Celková přenesená data (B)
hybridní	7675	2420	10 095
záložní	7675	233	7908
klasický	286	233	519

(zdroj: vlastní)



Obrázek 20 Graf porovnání velikosti přenesených dat jednotlivých režimů

(zdroj: vlastní)

Jak ukazuje tabulka a graf, hybridní handshake přenáší přibližně 20krát více dat než klasický handshake. Záložní režim přenáší přibližně 15krát více dat ve srovnání s klasickým režimem. Výrazné navýšení objemu přenesených dat v hybridním režimu je způsobeno přenosem rozměrného veřejného klíče ML-KEM768 a odpovídajícího šifrovaného textu. I když záložní režim eliminuje část komunikace spojenou s hybridním odpovědí serveru, klient stále posílá hybridní sdílený klíč, což způsobuje výrazně vyšší přenos dat oproti čistému X25519 handshaku. Výsledky jednoznačně potvrzují, že nasazení postkvantových kryptografických prvků přináší významné zvýšení datové režie při navazování zabezpečeného spojení.

7.4 Shrnutí testů

Provedené testy potvrdily, že všechny klíčové algoritmy i navržený TLS-like protokol fungují dle očekávání. U algoritmu X25519 byla ověřena správná realizace výměny klíčů i shoda výsledků s oficiální knihovní implementací, přestože vlastní řešení vykazuje nižší výkonnost kvůli absenci optimalizací. V případě ML-KEM768 byly sice zaznamenány nesrovnalosti při porovnávání s referenčními testovacími vektory a knihovnou *kyber-py*, základní operace jako šifrování, dešifrování i generování klíčů však probíhají v souladu se specifikací.

Výkonnostní testy jednoznačně ukázaly, že použití optimalizovaných implementací, zejména v jazyce C, přináší řádově vyšší rychlost oproti čistě Pythonovým řešením. Největší rozdíly byly zaznamenány u ML-KEM768, kde optimalizovaná knihovna *liboqs* dosahuje 180násobného zrychlení oproti knihovně kyber-py.

Testování TLS-like protokolu dále potvrdilo, že nasazení postkvantových algoritmů v hybridním režimu znamená výrazné zvýšení časových i datových nároků handshake - handshake je až pětkrát pomalejší a objem přenesených dat až dvacetkrát vyšší oproti klasickému režimu X25519.

Celkově lze shrnout, že použití postkvantových algoritmů přináší očekávané kompromisy v podobě vyšší výpočetní a datové náročnosti. Pro praktické nasazení je tak zásadní využívat optimalizované implementace, které tyto limity výrazně zmírňují.

ZÁVĚR

Tato bakalářská práce byla zaměřena na postkvantovou kryptografii se zvláštním důrazem na praktickou implementaci a testování hybridního algoritmu X25519MLKEM768. Cílem bylo analyzovat aktuální stav v této oblasti, implementovat a ověřit funkčnost hybridního protokolu a porovnat jeho efektivitu s klasickými přístupy. Pracovní hypotéza předpokládala, že zavedení postkvantových mechanismů povede ke zvýšení výpočetní i datové náročnosti při navazování šifrovaného spojení.

V teoretické části byly nejprve shrnuty hlavní důvody pro přechod k postkvantové kryptografii v souvislosti s hrozbou kvantových útoků. Byly popsány klíčové kategorie kvantově odolných algoritmů a rozebrán současný stav jejich vývoje a standardizace, především v rámci amerického institutu NIST, včetně přehledu finalistů a doporučení relevantních institucí, jako je NÚKIB. Práce také zhodnotila silné a slabé stránky jednotlivých přístupů a klíčová kritéria pro jejich výběr.

Praktická část práce byla zaměřena na vlastní implementaci algoritmů X25519 a ML-KEM768 a návrh TLS-like protokolu, který tyto algoritmy kombinuje pro bezpečnou výměnu klíčů v prostředí ohroženém kvantovými útoky. Funkčnost a správnost řešení byla ověřena prostřednictvím sady testovacích skriptů. Výsledky testování potvrdily, že nasazení hybridního postkvantového algoritmu skutečně znamená nárůst výpočetní i datové náročnosti handshake v porovnání s čistě klasickým protokolem, což potvrzuje původní pracovní hypotézu. Dále bylo zjištěno, že výkonnost implementací je významně ovlivněna volbou programovacího jazyka a mírou optimalizace kódu.

Přínosem práce je praktická demonstrace možností a omezení hybridních postkvantových řešení, která může posloužit jako základ pro další optimalizaci a výzkum. Pro další rozvoj v této oblasti doporučuji zaměřit se na zlepšení efektivity implementace, integraci nových standardů, rozšíření testování na reálné síťové prostředí a podrobnou bezpečnostní analýzu, včetně odolnosti vůči konkrétním útokům a minimalizaci datové režie při zachování vysoké úrovně bezpečnosti i v budoucích podmínkách kvantové hrozby.

SEZNAM POUŽITÉ LITERATURY

[1] MOSCA, Michele a PIANI, Marco. *Quantum Threat Timeline Report 2024: Executive Summary*. Online. 2024. Toronto: Global Risk Institute, 2024. Dostupné z: https://globalriskinstitute.org/publication/2024-quantum-threat-timeline-report/. [cit. 2025-01-22].

[2] BAVDEKAR, Ritik; JAYANT CHOPDE, Eashan; BHATIA, Ashutosh; TIWARI, Kamlesh a SANDEEP, Daniel Joshua. *Post Quantum Cryptography: Techniques, Challenges, Standardization, and Directions for Future Research*. Online. 2022. ArXiv, 2022. Dostupné z: https://doi.org/10.48550/arXiv.2202.02826. [cit. 2025-01-21].

[3] PROOS, John a ZALKA, Christof. *Shor's discrete logarithm quantum algorithm for elliptic curves*. Online. V2. Waterloo: University of Waterloo, 2004. Dostupné z: https://arxiv.org/abs/quant-ph/0301141v2. [cit. 2025-01-22].

[4] GROVER, Lov K. A fast quantum mechanical algorithm for database search. Online. *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*. 1996, s. 212–219. Dostupné z: https://doi.org/10.1145/237814.237866. [cit. 2025-01-25].

[5] PROKOP, Miloš; WALLDEN, Petros a JOSEPH, David. Grover's Oracle for the Shortest Vector Problem and Its Application in Hybrid Classical–Quantum Solvers. Online. *IEEE Transactions on Quantum Engineering*. 2025, roč. 6, s. 1–15. Dostupné z: https://doi.org/10.48550/arXiv.2402.13895. [cit. 2025-01-25].

[6] BERNSTEIN, Daniel J. a DAHMEN, Erik (ed.). *Post-Quantum Cryptography*. Online. Berlin, Heidelberg: Springer-Verlag, 2009. ISBN 978-3-540-88702-7. Dostupné z: https://doi.org/10.1007/978-3-540-88702-7. [cit. 2024-10-18].

[7] MAMATHA, G. S.; DIMRI, Namya a SINHA, Rasha. *Post-Quantum Cryptography: Securing Digital Communication in the Quantum Era*. Online. ArXiv, 2024. Dostupné z: https://arxiv.org/abs/2403.11741. [cit. 2025-01-26].

[8] MICCIANCIO, Daniele a REGEV, Oded. Lattice-based Cryptography. Online. In: *Advances in Cryptology: Proceedings of the 26th Annual Cryptology Conference (CRYPTO'06)*. 2009, s. 147–191. ISBN 978-3-540-88701-0. Dostupné z: https://cims.nyu.edu/~regev/papers/pqc.pdf. [cit. 2025-01-26].

[9] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. FIPS 203, *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. Federal Information Processing Standards Publication (FIPS) NIST FIPS 203. Department of Commerce, Washington, D.C., 2024. Dostupné také z: https://doi.org/10.6028/NIST.FIPS.203.

[10] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. FIPS 204, *Module-Lattice-Based Digital Signature Standard*. Federal Information Pro- cessing Standards Publication (FIPS) NIST FIPS 204. Department of Commerce, Washington, D.C., 2024. Dostupné také z: https://doi.org/10.6028/NIST.FIPS.204.

[11] DAVLETOVA, Alina; YATSKIV, Vasyl; IVASIEV, Stepan a KARPINSKYI, Mykola. Encryption Method Based on Codes. Online. *ADVANCES IN CYBER-PHYSICAL SYSTEMS*. 2024, roč. 9, č. 1, s. 24-31. Dostupné z: https://doi.org/10.23939/acps2024.01.024. [cit. 2025-05-24].

[12] LICHTINGER, Jacob; MILLER, Carl; MOODY, Dustin; PERALTA, Rene; PERLNER, Ray et al. *Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process*. Online. NIST IR 8413. Gaithersburg, MD: National Institute of Standards and Technology, 2022. Dostupné z: https://doi.org/10.6028/NIST.IR.8413. [cit. 2025-01-17].

[13] HÜLSING, Andreas; GAZDAG, Stefan - Lukas; BUTIN, Denis a BUCHMANN, Johannes. Hash-based Signatures: An Outline for a New Standard. Online. In: *Workshop on Cybersecurity in a Post-Quantum World*. USA: NIST, 2015. Dostupné z: https://csrc.nist.gov/csrc/media/events/workshop-on-cybersecurity-in-a-post-quantum-world/documents/papers/session5-hulsing-paper.pdf. [cit. 2025-01-30].

[14] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. FIPS 205, *Stateless Hash-Based Digital Signature Standard*. Federal Information Processing Standards Publication (FIPS) NIST FIPS 205. Department of Commerce, Washington, D.C., 2024. Dostupné také z: https: //doi.org/10.6028/NIST.FIPS.205.

[15] NÁRODNÍ ÚŘAD PRO KYBERNETICKOU A INFORMAČNÍ BEZPEČNOST. *Minimální požadavky na kryptografické algoritmy – Příloha: Kvantová hrozba a kvantově odolná kryptografie*. Online. Verze 2.0. NÚKIB, 2025. Dostupné z: https://nukib.gov.cz/download/uredni_deska/Minimalni_pozadavky_Priloha_v2_FINAL.pdf. [cit. 2025-03-17].

[16] ARPIN, Sarah; CAMACHO-NAVARRO, Catalina; LAUTER, Kristin; LIM, Joelle; NELSON, Kristina et al. Adventures in Supersingularland. Online. *Experimental Mathematics*. 2019, č. 2, s. 241-268. Dostupné z: https://doi.org/10.1080/10586458.2021.1926009. [cit. 2025-02-13].

[17] *SIKE Foreword and Postscript*. Online. SIKE team, 2022. Dostupné také z: https://csrc.nist.gov/csrc/media/Projects/post-quantum-cryptography/documents/round-4/submissions/sike-team-note-insecure.pdf.

[18] TESKE, Edlyn. *NIST PQC Finalists Update: It's Over for the Rainbow*. Online. Cryptomathic. 2022. Dostupné z: https://www.cryptomathic.com/blog/nist-pqc-finalists-update-its-over-for-the-rainbow. [cit. 2025-02-12].

[19] *Minimální požadavky na kryptografické algoritmy*. Online. 4.0. Brno: NÚKIB, 2025. Dostupné z: https://nukib.gov.cz/download/uredni_deska/Minimalni_pozadavky_v4_FINAL.pdf. [cit. 2025-01-23].

[20] KUMAR, Manish. *Post-Quantum Cryptography Algorithm's Standardization and Performance Analysis*. Online. 2022. Dostupné také z: https://arxiv.org/abs/2204.02571.

[21] CHEN, Lily; JORDAN, Stephen; LU, Yi-Kai; MOODY, Dustin; PERALTA, Rene et al. *Report on Post-Quantum Cryptography*. Online. NISTIR 8105. Gaithersburg, MD: National Institute of Standards and Technology, 2016. Dostupné z: http://dx.doi.org/10.6028/NIST.IR.8105. [cit. 2025-01-01].

[22] ALAGIC, Gorjan; ALPERIN-SHERIFF, Jacob; APON, Daniel; COOPER, David; DANG, Quynh et al. *Status Report on the First Round of the NIST Post-Quantum Cryptography Standardization Process*. Online. NISTIR 8240. Gaithersburg, MD: National Institute of Standards and Technology, 2019. Dostupné z: https://doi.org/10.6028/NIST.IR.8240. [cit. 2025-01-17].

[23] ALAGIC, Gorjan; ALPERIN-SHERIFF, Jacob; APON, Daniel; COOPER, David; DANG, Quynh et al. *Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process*. Online. NISTIR 8309. Gaithersburg, MD: National Institute of Standards and Technology, 2020. Dostupné z: https://doi.org/10.6028/NIST.IR.8309. [cit. 2025-01-17].

[24] NIST. *NIST Releases First 3 Finalized Post-Quantum Encryption Standards*. Online. 2024. Dostupné z: https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards. [cit. 2025-02-26].

[25] *Portál NÚKIB nově podporuje kvantově odolnou kryptografii*. Online. Národní úřad pro kybernetickou a informační bezpečnost. 2024. Dostupné z: https://nukib.gov.cz/cs/infoservis/aktuality/2156-portal-nukib-nove-podporuje-kvantove-odolnou-kryptografii/#jak-zjistim-ze-komunikace-s-portalem-nukib-je-zabezpecena-postkvantovym-sifrovanim. [cit. 2025-01-23].

[26] NATIONAL SECURITY AGENCY. Post-Quantum Cryptography: CISA, NIST, and NSA Recommend How to Prepare Now. Online. 2023. Dostupné z: https://www.nsa.gov/Press-Room/Press-Releases-Statements/Press-Release-View/Article/3498776/post-quantum-cryptography-cisa-nist-and-nsa-recommend-how-to-prepare-now/. [cit. 2025-03-18].

[27] NATIONAL SECURITY AGENCY. *The Commercial National Security Algorithm Suite 2.0 and Quantum Computing FAQ*. Online. Ver 2.1. 2024. Dostupné také z: https://media.defense.gov/2022/Sep/07/2003071836/-1/-1/0/CSI_CNSA_2.0_FAQ_.PDF.

[28] EUROPEAN UNION. *Securing Tomorrow, Today: Transitioning to Post- Quantum Cryptography*. Online. 2024. Dostupné také z: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Crypto/PQC-joint-statement.pdf?__blob=publicationFile&v=5.

[29] SWAYNE, Matt. *China Launches Its Own Quantum-Resistant Encryption Standards, Bypassing US Efforts*. Online. In: Quantum Insider. 2025. Dostupné z: https://thequantuminsider.com/2025/02/18/china-launches-its-own-quantum-resistant-encryption-standard-bypassing-us-efforts/. [cit. 2025-03-18].

[30] INSTITUTE OF COMMERCIAL CRYPTOGRAPHY STANDARDS. *Announcement on Launching the Next-generation Commercial Cryptographic Algorithms Program (NGCC)*. Online. In: Institute of Commercial Cryptography Standards. 2025. Dostupné z: https://www.niccs.org.cn/en/. [cit. 2025-03-18].

[31] CHOUCAIR, Cierra. *South Korea Launches Quantum Strategy Comittee, Allocates $15M Annually For Quantum Startups, But Some Warn It Falls Short*. Online. In: Quantum Insider. 2025. Dostupné z: https://thequantuminsider.com/2025/03/13/south-korea-launches-quantum-strategy-comittee-allocates-15m-annually-for-quantum-startups-but-some-warn-it-falls-short/. [cit. 2025-03-18].

[32] QUANTUM NEWS. *PQShield has announced its membership of Japanese Cyber Research Consortium*. Online. In: Quantum Zeitgeist. 2025. Dostupné z: https://quantumzeitgeist.com/pqshield-has-announced-its-membership-of-japanese-cyber-research-consortium/. [cit. 2025-03-18].

[33] PATHUM, Udara. *X25519Kyber768 Post-Quantum Key Exchange for HTTPS Communication*. Online. In: Medium. 2024. Dostupné z: https://medium.com/@hwupathum/x25519kyber768-post-quantum-key-exchange-for-https-communication-70eba681931d. [cit. 2025-03-19].

[34] BERNSTEIN, Daniel J. *Curve25519: new Diffie-Hellman speed records*. Online. 2006. Dostupné také z: https://cr.yp.to/ecdh/curve25519-20060209.pdf.

[35] SCHAUMANN, Jan. *TLS 1.3 Hybrid Key Exchange using X25519Kyber768 / ML-KEM*. Online. In: Signs of Triviality. 2024. Dostupné z: https://www.netmeister.org/blog/tls-hybrid-kex.html. [cit. 2025-03-19].

[36] SCHAUMANN, Jan. *Post-Quantum Cryptography in February 2025*. Online. In: Signs of Triviality. 2025. Dostupné z: https://www.netmeister.org/blog/pqc-2025-02.html. [cit. 2025-03-19].

[37] GIRON, Alexandre Augusto; ADAMI DO NASCIMENTO, Joao Pedro; CUSTODIO, Ricardo a PERIN, Lucas Pandolfo. *Post-Quantum Hybrid KEMTLS Performance in Simulated and Real Network Environments*. Online. Florianópolis, Brazil: Federal University of Santa Catarina (UFSC), 2022. Dostupné také z: https://eprint.iacr.org/2022/1639.pdf.

[38] OUNSWORTH, M.; GRAY, J.; PALA, M.; KLAUSSNER, J. a FLUHRER, S. *Composite ML-KEM for use in X.509 Public Key Infrastructure and CMS*. Online. 2024. Dostupné také z: https://www.ietf.org/archive/id/draft-ietf-lamps-pq-composite-kem-05.html.

[39] KLEPPMANN, MARTIN. *Implementing Curve25519/X25519: A Tutorial on Elliptic Curve Cryptography*. Online. University of Cambridge, United Kingdom, 2021. Dostupné také z: https://www.cl.cam.ac.uk/teaching/2122/Crypto/curve25519.pdf.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AES	Advanced Encryption Standard (Pokročilý šifrovací standard)
ARM	Advanced RISC Machine (architektura pokročilého RISC procesoru)
AWS	Amazon Web Services (webové služby Amazonu)
AVX2	Advanced Vector Extensions 2 (rozšíření vektorových instrukcí 2)
BIKE	Bit Flipping Key Encapsulation (zapouzdření klíče s bitovým překlápěním)
BKZ	Block Korkine-Zolotarev (blokový algoritmus Korkin-Zolotarev pro redukci mřížek)
CBD	Centered Binomial Distribution (centrální binomické rozdělení)
CNSA	Commercial National Security Algorithm Suite (sada algoritmů pro národní bezpečnost, komerční použití)
CVP	Closest Vector Problem (problém nejbližšího vektoru)
Dk	Decryption key (dešifrovací klíč)
DSA	Digital Signature Algorithm (algoritmus digitálního podpisu)
ECC	Elliptic Curve Cryptography (kryptografie na eliptických křivkách)
ECDH	Elliptic Curve Diffie-Hellman (Diffie-Hellman na eliptických křivkách)
ECDSA	Elliptic Curve Digital Signature Algorithm (digitální podpis na eliptických křivkách)
EU	European Union (Evropská unie)
EUF-CMA	Existential Unforgeability under Chosen Message Attack (existenciální nefalšovatelnost proti zvolenému útoku na zprávu)
FIPS	Federal Information Processing Standard (federální standard zpracování informací)
FORPS	Forest of Random Subsets (les náhodných podmnožin, použitý v SPHINCS+)
GCM	Galois/Counter Mode (Galoisův/čítací mód blokové šifry)
GeMSS	Great Multivariate Short Signature (velmi krátký multivariační podpis)

HKDF	HMAC-based Key Derivation Function (Funkce pro odvozování klíčů založená na HMAC)
HQC	Hamming Quasi-Cyclic (kvazicyklický kód Hammingova typu)
HTTPS	HyperText Transfer Protocol Secure (zabezpečený protokol přenosu hypertextu)
ICCS	Institute of Commercial Cryptography Standards (Institut pro standardy komerční kryptografie)
IND-CCA2	Indistinguishability under Adaptive Chosen Ciphertext Attack (nerozlišitelnost při adaptivním útoku s volbou šifrového textu)
IoT	Internet of Things (internet věcí)
IP	Internet Protocol (internetový protokol)
IPsec	Internet Protocol Security (bezpečnost internetového protokolu)
JSON	JavaScript Object Notation (formát zápisu objektů v JavaScriptu)
K-PKE	Kyber-like Public Key Encryption (Kyberu podobné šifrování s veřejným klíčem)
KAT	Known Answer Test (test se známou odpovědí)
KEM	Key Encapsulation Mechanism (mechanismus zapouzdření klíče)
LAC	Lattice-based Cryptography (mřížková kryptografie, název algoritmu)
LEDAcrypt	Low-Density Generator Matrix Cryptosystem (Kódový postkvantový algoritmus)
LWE	Learning With Errors (učení s chybami)
ML-DSA	Module-Lattice-Based Digital Signature Algorithm (modulově-mřížkový algoritmus digitálního podpisu)
ML-KEM	Module-Lattice-based Key Encapsulation Mechanism (modulově mřížkový mechanismus zapouzdření klíče)
MLWE	Module Learning With Errors (modulární učení s chybami)
MPKC	Multivariate Public Key Cryptography (multivariační kryptografie s veřejným klíčem)

NEDO	New Energy and Industrial Technology Development Organization (Japonská agentura pro výzkum a vývoj)
NGCC	Next-generation Commercial Cryptographic Algorithms (Komerční kryptografické algoritmy nové generace)
NGCC-BC	NGCC Block Cipher (Blokové šifry v rámci NGCC)
NGCC-CH	NGCC Cryptographic Hash (Hashovací funkce v rámci NGCC)
NGCC-PK	NGCC Public Key (Asymetrické algoritmy v rámci NGCC)
NIST	National Institute of Standards and Technology (Národní institut pro standardy a technologie, USA)
NP	Nondeterministic Polynomial time (nedeterministický polynomiální čas)
NSS	Network Security Services (Sada kryptografických knihoven od Mozilly)
NTRU	N-th degree Truncated Polynomial Ring Units (kryptosystém založený na polynomiálních okruzích)
NTT	Number Theoretic Transform (číselná teoretická transformace)
NÚKIB	Národní úřad pro kybernetickou a informační bezpečnost (Národní úřad pro kybernetickou a informační bezpečnost)
Pk	Public key (veřejný klíč)
PQC	Post-Quantum Cryptography (postkvantová kryptografie)
RLWE	Ring Learning With Errors (učení s chybami v okruhu)
SA	Security Association (bezpečnostní asociace)
SABER	Secure And Fast Encryption Routine (rychlá a bezpečná šifrovací procedura)
SHA-2	Secure Hash Algorithm 2 (bezpečný hashovací algoritmus 2. generace)
SHA-3	Secure Hash Algorithm 3 (bezpečný hashovací algoritmus 3. generace)
SHAKE256	Secure Hash Algorithm Keccak Extendable-Output Function 256 (Varianta SHA-3 s volitelnou délkou výstupu)
SIDH	Supersingular Isogeny Diffie-Hellman (Diffie-Hellman na supersingulárních isogeniích)

SIKE	Supersingular Isogeny Key Encapsulation (zapouzdření klíče na bázi supersingulárních isogenií)
Sk	Secret key (soukromý klíč)
SLH-DSA	Stateless Hash-based Digital Signature Algorithm (bezdotazový hashový algoritmus digitálního podpisu)
SPHINCS	Stateless Practical Hash-based Incredibly Nice Cryptographic Signature (bezdotazový praktický hashový kryptografický podpis)
SSH	Secure Shell (zabezpečený protokol vzdáleného přístupu)
SVP	Shortest Vector Problem (problém nejkratšího vektoru)
TCP	Transmission Control Protocol (protokol řízení přenosu)
TLS	Transport Layer Security (zabezpečení transportní vrstvy)
VPN	Virtual Private Network (virtuální privátní síť)
WOTS+	Winternitz One-Time Signature Plus (Winternitzův jednorázový podpis, rozšířený)
XMSS	eXtended Merkle Signature Scheme (rozšířený Merkleův podpisový schéma)

SEZNAM PŘÍLOH

PŘÍLOHA P I: ZDROJOVÉ KÓDY APLIKACE

README.md

- Obsahuje základní informace o projektu a podrobné instrukce k instalaci a spuštění.

main.py

- Hlavní spouštěcí soubor projektu.

složka client_server

- Implementace klienta a serveru včetně GUI.

  Obsahuje:

    client.py a server.py – logika klientské a serverové části.

    složka gui – soubory pro grafické uživatelské rozhraní klienta a serveru.

složka tests

- Testovací skripty pro ověření správnosti implementace.

  Obsahuje:

    test_mlkem768.py – testování algoritmu ML-KEM768.

    test_tls_handshake.py – testování správnosti navázání handshake.

    test_x25519.py – testování algoritmu X25519.

složka benchmark

- Výkonnostní testy a generování grafů.

  Obsahuje:

    run_benchmark.py – měření rychlosti jednotlivých operací.

    generate_graphs.py – skript pro generování grafu výsledků.

složka tls

- Implementace hybridního TLS-like protokolu a souvisejících algoritmů.

  Obsahuje:

    mlkem768.py, x25519.py, tls_core.py – základní logika protokolu.

    složka mlkem768_files – interní moduly pro ML-KEM768.