МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Дослідження алгоритмів інкапсуляції ключів. CRYSTALS Kyber

Виконала:

студентка 2 курсу групи ІПЗм-21-2

Циганок Д.А.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення

Тип програми Освітньо-наукова

Керівник проф. каф. ПІ Качко О. Г.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. Кафедри _____ З.В. Дудар
(прізвище, ініціали)

2023 р

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121– Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«___»_____2023 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентці Циганок Дар'ї Андріївні
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження алгоритмів інкапсуляції ключів. CRYSTALS Kyber» затверджена наказом університету від «29» березня 2023р. № 302Ст

2. Термін подання студентом роботи до екзаменаційної комісії «19» травня 2023 р.

3. Вихідні дані до роботи *аналіз постквантового алгоритму, поліноми, криптографія, пояснювальна записка.*

4. Перелік питань, що потрібно опрацювати в роботі *мета роботи, аналіз предметної галузі і постановка задачі, дослідження квантових комп'ютерів, криптографії та постквантової криптографії, аналіз алгоритму* CRYSTALS Kyber, *вивчення можливості використання при створенні квантових комп'ютерів.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
	Аналіз предметної галузі, огляд існуючих рішень	10.03.2023	
	Дослідження постквантових криптоалгоритмів	15.03.2023	
	Створення специфікації ПЗ	25.03.2023	
	Проектування та розробка ПЗ	1.04.2023	
	Тестування та дослідна експлуатація ПЗ	10.04.2023	
	Написання пояснювальної записки	20.04.2023	
	Перевірка пояснювальної записки керівником, підготовка роботи для проходження перевірки на антиплагіат та проходження нормоконтролю	12.05.2023	
	Оцінка роботи рецензентом, отримання відзиву від керівника кваліфікаційної роботи, попередній захист роботи	18.05.2023	
	Здача роботи у електронний архів, допуск роботи до захисту завідувачем кафедри та передача готової роботи секретарю ЕК	18.05.2023	
	Захист кваліфікаційної роботи	19.05.2023	

Дата видачі завдання 10 березня 2023 р.

Студентка Циганок Дар’я Андріївна
(підпис)

Керівник роботи _____ проф. каф. ПІ Качко О.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до роботи містить: 69 с., 11 рис., 5 табл.,5 додатків, 10 джерел.

КВАНТОВИЙ КОМП'ЮТЕР, ПОСТ-КВАНТОВИЙ АЛГОРИТМ, ПОСТКВАНТОВА КРИПТОГРАФІЯ, АНАЛІЗ, ПОРІВНЯЛЬНИЙ АНАЛІЗ, CRYSTALS KYBER.

Об'єктом дослідження є сучасні алгоритми інкапсуляції ключів, вразливі та стійкі до атак квантових компьютерів.

Метою роботи є проведення порівняльного аналізу сучасних алгоритмів інкапсуляції ключів, що будуть стійкими у постквантовий період, аналіз алгоритму CRYSTALS Kyber, та його оптимізація.

У результаті роботи проаналізовано проблему сучасних поширених алгоритмів у постквантовий період, порівняно посквантові алгоритми, проаналізовано алгоритм CRYSTALS Kyber та оптимізовано його.

QUANTUM COMPUTER, POST-QUANTUM ALGORITHM, POST-QUANTUM CRYPTOGRAPHY, ANALYSIS, COMPARATIVE ANALYSIS, CRYSTALS KYBER.

The object of research is modern key encapsulation algorithms, which are vulnerable and resistant to quantum computer attacks.

The purpose of the work is to conduct a comparative analysis of modern key encapsulation algorithms that will be stable in the post-quantum period, an analysis of the CRYSTALS Kyber algorithm, and its optimization.

As a result of the work, the problem of modern common algorithms in the post-quantum period was analyzed, post-quantum algorithms were compared, the CRYSTALS Kyber algorithm was analyzed and optimized.

Умови публікації пояснювальної записки

Я,

Циганок Дар’я Андріївна
(прізвище, ім’я, по батькові)

студентка групи ІПЗм-21-2 здобувач вищої освіти на другому (магістерському) рівні

кафедра програмної інженерії,
(повна назва кафедри)

заявляю: моя кваліфікаційна робота на тему
Дослідження алгоритмів інкапсуляції ключів. CRYSTALS Kyber,
(назва роботи)

що буде представлена до ЕК для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений (а) з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

PQC(Post Quantum Cryptography) – постквантова криптографія;

KEM(Key Encapsulation Mechanism) – механізм для передачі випадкового ключового матеріалу одержувачу за допомогою відкритого ключа одержувача;

CCA(Chosen-Ciphertext Attack) – це модель атаки для криптоаналізу, де криптоаналітик може збирати інформацію, отримуючи розшифровки вибраних зашифрованих текстів;

IND-CPA/ IND-CCA2 – рівень захищенності від атак основі підібраного відкритого тексту/ від адаптивних атак на основі підібраного шифротексту;

PKE(Public Key Encryption) – це процес використання пар відкритих/приватних ключів для захисту даних;

SVP(Shortest Vector Problem) – проблема найкоротшого вектора вимагає знайти ненульовий вектор у решітці. Задачу можна визначити щодо будь-якої норми, але евклідова норма є найпоширенішою;

G – функція гешування;

H – функція гешування;

KDF – потокове шифрування (для генерації псевдо-випадкових послідовностей);

PRF – потокове шифрування (для генерації псевдо-випадкових послідовностей);

XOF – потокове шифрування (для генерації псевдо-випадкових послідовностей);

KYBER_N – порядок поліному;

KYBER_A – матриця, елементи якої поліноми порядку n;

KYBER_K – кількість рядків в матриці A;

KYBER_Q – модуль, за яким виконуються усі обчислення по модулю;

KYBER_ETA1, KYBER_ETA2 – кількість бітів, які відводяться для елементів малих поліномів;

KYBER_DU, KYBER_DV – кількість бітів для завдання коефіцієнтів поліномів u, v, в зашифрованому тексті;

KYBER_DELTA – імовірність помилки розшифрування;

KYBER_SYMBYTES – кількість байтів для випадкового рядка та гешу;

KYBER_SSBYTES – кількість байтів для загального секрету;

KYBER_POLYBYTES – кількість байтів для завдання поліному;

KYBER_POLYCOMPRESSEDBYTES – кількість байтів для завдання поліному в упакованому форматі;

KYBER_POLYVECCOMPRESSEDBYTES – кількість байтів для завдання вектору в упакованому форматі;

KYBER_POLYVECBYTES – кількість байтів для завдання вектору, який складається з KYBER_K поліномів;

KYBER_INDCPA_MSGBYTES – кількість байтів для повідомлення, яке зашифровується;

KYBER_INDCPA_PUBLICKEYBYTES – кількість байтів для відкритого ключа;

KYBER_INDCPA_SECRETKEYBYTES - кількість байтів для секретного ключа, без додаткових полів;

KYBER_INDCPA_BYTES – довжина зашифрованого повідомлення в упакованому форматі;

KYBER_PUBLICKEYBYTES – кількість байтів для відкритого ключа, співпадає з KYBER_INDCPA_PUBLICK;

KYBER_SECRETKEYBYTES – кількість байтів для секретного ключа, з урахуванням додаткових полів;

KYBER_CIPHERTEXTBYTES – довжина зашифрованого повідомлення в упакованому форматі;

KYBER_RO – випадковий компонент для генерації матриці A завдовжки KYBER_SYMBYTES.

ВСТУП

На сьогоднішній день у світі існує величезна кількість інформації. Сенс прогресу полягає в отриманні досвіду та нової інформації для покращення якості життя та пізнання світу навколо.

Інформація є дуже цінним ресурсом. Як і будь-який цінний ресурс він має бути під захистом. Ще близько 4 тисяч років тому було знайдено перші пристрої криптографії, такі як диск Енея або шифр Цезаря. З переходом на електронні пристрої завдання із шифруванням інформації дещо ускладнилося.

Прогрес не стоїть на місті, тож з розвитком криптографії розвилися і методи її взлому. Зараз практично з кожним роком обсяг інформації зростає експоненційно і сучасним комп'ютерам стає все важче обробляти таку кількість інформації.

Квантові комп'ютери мають потенціал вирішувати проблеми, які не можуть бути вирішені традиційними комп'ютерами, такі як факторизація великих чисел або пошук у великих баз даних [1].

Принцип квантових обчислень дозволить не тільки обробляти більше даних та виконувати більше операцій, а ще краще зрозуміти природні процеси. “Хаос” суперпозиції набагато краще відображає спосіб, наприклад, мутацій в ДНК, а отже, розвиток хвороб та еволюцію. Квантові обчислення вже сьогодні використовуються для створення нових лікарських препаратів.

З усіма цими величезними можливостями перед нами з’явиться нова проблема - захисту інформації від майбутніх атак квантових комп'ютерів, оскільки квантові комп'ютери зможуть легко зламати алгоритми, що використовуються в даний час.Якщо обчислювальні можливості порушника зростуть у десятки, сотні, тисячі разів, це призведе до різкої необхідності збільшення довжини ключів до критичного рівня, не придатного їх успішної експлуатації в сьогоднішніх інформаційних системах. Також, необхідно відзначити, що з появою квантового супротивника через його обчислювальні можливості він зможе простим перебором зламати існуючу криптосистему. Однак цю проблему можна вирішити шляхом використання примітивів постквантової

криптографії, порівняно нової галузі криптографії, покликаної протистояти квантовим обчисленням.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Квантові комп’ютери

Квантовий комп’ютер — фізичний обчислювальний пристрій, функціонування якого ґрунтується на принципах квантової механіки, зокрема, принципі суперпозиції та явищі квантової заплутаності. Такий пристрій відрізняється від звичайного транзисторного комп'ютера зокрема тим, що класичний комп'ютер оперує даними, закодованими у двійкових розрядах (бітах), кожен з яких завжди перебуває в одному з двох станів (0 або 1), коли квантовий комп'ютер використовує квантові біти (кубіти), які можуть знаходитися у суперпозиції станів.

Квантові методи виконання обчислювальних операцій, а також передачі та обробки інформації, вже починають втілюватися в реально функціонуючих експериментальних пристроях, що стимулює зусилля з реалізації квантових комп'ютерів. Квантовий комп'ютер складається з n кубітів і дозволяє проводити одно- і двухкубітові операції над будь-яким з них (або будь-якою парою). Ці операції виконуються під впливом імпульсів зовнішнього поля, керованого класичним комп'ютером.

При введенні інформації в квантовий комп'ютер стан вхідного регістра, за допомогою відповідних імпульсних впливів перетворюється у відповідну когерентну суперпозицію базисних ортогональних станів. У такому вигляді інформація далі піддається впливу квантового процесора, що виконує послідовність квантових логічних операцій, яка визначається унітарним перетворенням, чинним на стан всього регістру. До моменту часу t в результаті перетворень вихідний квантовий стан стає новою суперпозицією, яка і визначає результат перетворення інформації на виході комп'ютера.

Сукупність усіх можливих операцій на вході даного комп'ютера, що формують вихідні стану, а також здійснюють унітарні локальні перетворення, відповідні алгоритму обчислення, способи придушення втрати когерентності - так званої декогеренції (decoherence) квантових станів та виправлення випадкових

помилок, грають тут ту ж роль, що і "програмне забезпечення" (software) в класичному комп'ютері.

Квантовий комп’ютер складається з трьох основних частин: області для зберігання кубітів, методу передачі сигналів кубітам та класичного комп’ютера для запуску програми та надсилання інструкцій.[2]

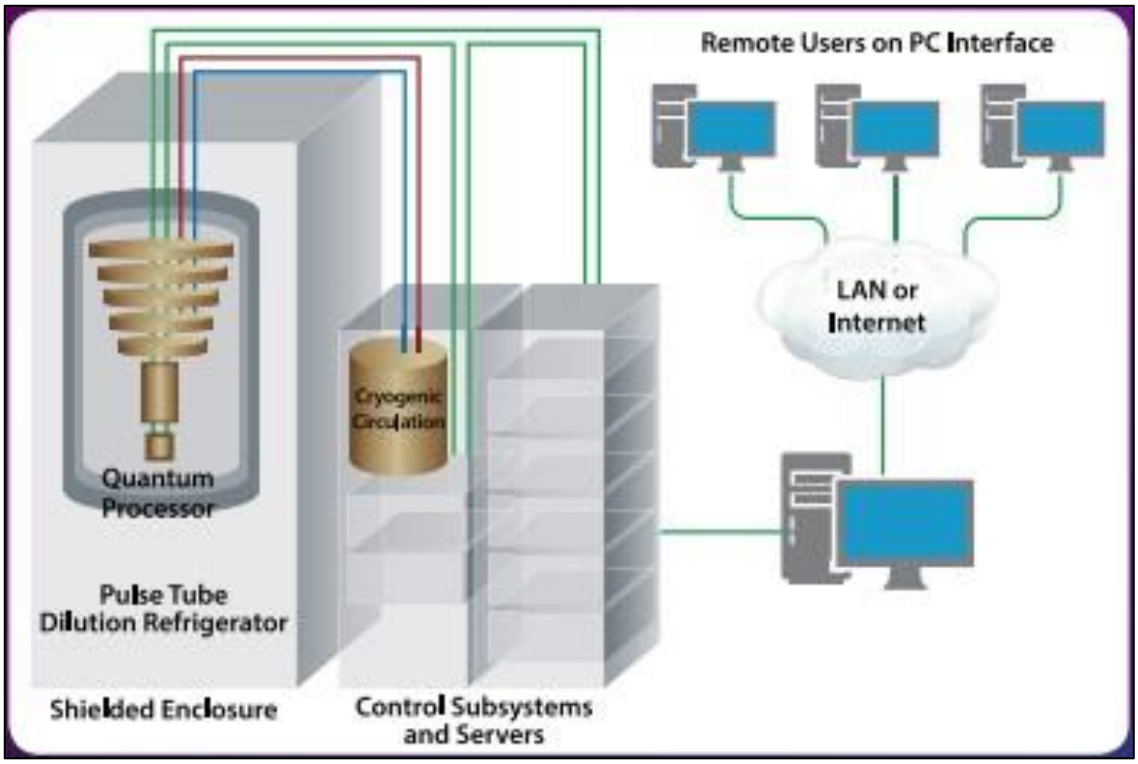

Рисунок 1.1 Схема побудови квантових комп’ютерів

Квантовий процесор складається з решіток крихітних надпровідних ланцюгів (кубітів), виготовлених з ніобію, який проявляє квантову поведінку при дуже низьких температурах. Сам процесор оточений електронікою, яка використовується для програмування і зчитування результатів.

Квантовий матеріал, який утворює кубіти, є делікатним і надзвичайно чутливим до впливів навколишнього середовища. Для деяких методів зберігання кубітів одиниця, яка вміщує кубіти, тримається при температурі, близькій до абсолютного нуля, щоб максимізувати їх узгодженість. Інші типи кубітових сховищ використовують вакуумну камеру для мінімізації вібрації та стабілізації кубітів.

Існують різні методи передачі сигналів кубітам, наприклад мікрохвилі, лазер та електрична напруга.

Щоб налагодити нормальну роботу квантових комп'ютерів, потрібно вирішити багато проблем. Суттєвою проблемою квантових комп'ютерів є виправлення помилок, а масштабування (додавання більшої кількості кубітів) ще більше збільшує частоту їх появи. Через ці обмеження квантовий персональний комп'ютер на вашому столі все ще є картиною з далекого майбутнього, але комерційні квантові комп'ютери можуть стати доступними вже найближчим часом.

1.1.1 Порівняння сучасних та квантових комп'ютерів

У класичних комп'ютерах кожна інформація зберігається як послідовність одиниць та нулів. Така інформація сприймається та інтерпретується комп'ютером, консоллю, смартфоном, розумним годинником та смарт-телевізором, подібно до операцій, які виконуються над цією інформацією. Незалежно від того, переглядаємо ми фотографії з відпустки, спілкуємося з друзями в чаті, граємо в останню гру або виконуємо розширені криптографічні розрахунки – все відбувається в двійковій системі, де є 0 або 1 і нічого іншого. Насправді, це більш схоже на класичне "так" чи "ні".

Наскільки неефективною є ця система, видно, коли ми доходимо до її меж. І незалежно від того, чи бракує нам місця на смартфоні для чергового селфі, чи вчені намагаються створити математичні моделі розвитку пандемії, проблема полягає в тому, що занадто багато нулів і одиниць, і ресурсів для їх зберігання та потужностей для їх обчислення немає.

Кубіт вирішує цю проблему. Ця частина інформації використовує властивості квантової фізики, які дозволяють їй залишатися у так званій суперпозиції. Кубіт може приймати будь-яке значення від 0 до 1. Він має властивості всього спектра і може мати значення, наприклад, на 15 відсотків нуль і на 85 відсотків один. Теоретично це дозволяє зберегти набагато більше інформації, або пришвидшити обчислення. Але при цьому також виникає маса проблем, які важко контролювати і навіть зрозуміти.

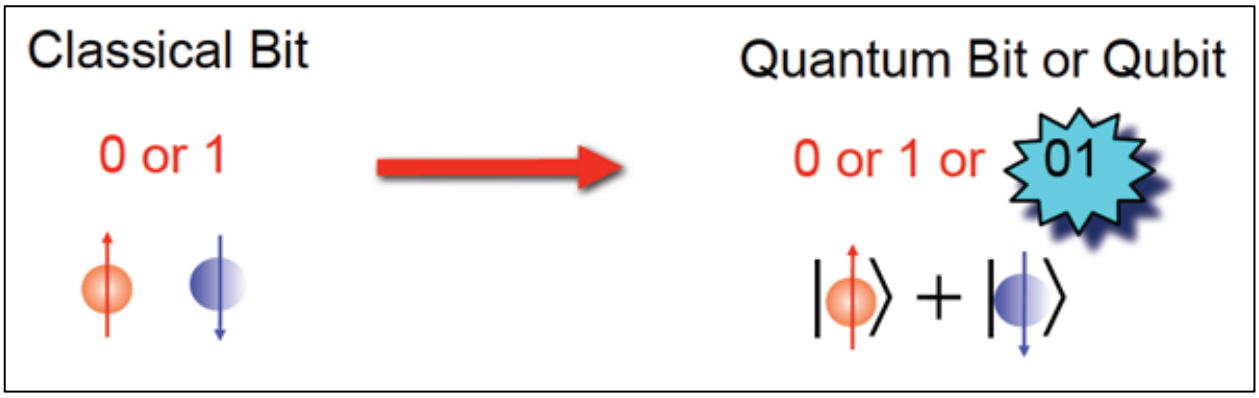

Рисунок 1.2 Схема порівняння біту та квантового біту (кубіту)

Ще однією особливістю квантових комп’ютерів, яка дозволяє додатково масштабувати обчислювальний приріст потужності, є використання квантового переплутування. Це стан, коли два кубіти з’єднані між собою, і кожного разу, коли ми спостерігаємо один з них, інший буде знаходитися у точно такому ж стані. Заплутаність дозволяє згрупувати кубіти в ще більш ефективні одиниці для запису та обробки інформації.

Оскільки кубіт може ефективно зберігати два біти даних, а не один, квантові комп'ютери мають значно збільшену обчислювальну потужність. Тим не менш, це може бути корисним тільки для певних типів проблем, і основний інтерес досліджень полягає у виявленні цих проблем та розробці алгоритмів для них.

Класичним комп'ютерам важко виконувати функції, які мають експоненційне зростання, наприклад, це Prime Factorization для великих чисел, на які класичним комп'ютерам потрібно багато років, щоб їх розкласти на множники. Навпаки, за допомогою алгоритму Шора, який ґрунтується на принципах квантової механіки, це завдання виконується за лічені секунди.

Поширена помилка про квантові обчислення полягає в тому, що вони збираються замінити класичні обчислення. Квантові комп'ютери не повністю замінять класичні. Вони запропонують принципово інший спосіб виконання обчислень і зможуть виконувати завдання, виконання яких класичним комп'ютерам знадобляться мільйони років.

Google оголосив, що його 54-кубовий процесор Sycamore зміг виконати обчислювальне завдання за 200 секунд, що найпотужніший суперкомп'ютер у світі робив би 10000 років. IBM, яка управляє суперкомп'ютером, який, за твердженнями Google, обіграв, заперечує їхні претензії. У відповідь IBM заявила, що проблема з випадковими числами, яку вирішує квантовий комп'ютер, займає на його суперкомп'ютері менше трьох днів, а не 10000 років, і тому сперечаються про законність претензії.

Квантові комп'ютери матимуть величезний вплив на безпеку даних. Навіть якщо квантові комп'ютери легко зламають багато сучасних методів шифрування, вони створять злостійкі заміни. Інші додатки, де квантові обчислення можуть вплинути - молекулярне моделювання, математична оптимізація, машинне навчання та багато іншого.

1.1.2 Стан досягнень у побудові квантових комп’ютерів

Теоретично квантовий комп’ютер буде здатний в одну мить проводити мільйон обчислень, в той час як класичний — тільки одне, тобто потенційна обчислювальна потужність квантового комп’ютера перевищує потужність класичних комп’ютерів приблизно в 10 у 80-му ступені разів. Зараз вже описані десятки алгоритмів роботи квантового комп'ютера, навіть розробляються особливі мови програмування.

На виставці CES компанія IBM анонсувала свій перший комерційний квантовий комп’ютер для використання поза лабораторією – Q System One[3]. 20-кубітна система , яка укладена у герметичний корпус у формі куба з гранню довжиною 2,75 м, та виконаний з боросилікатного скла товщиною 1,27 см поєднує в одному пакеті квантові та класичні обчислювальні частини, необхідні для використання такої машини для досліджень і бізнес-додатків. Цей пакет, система IBM Q, звичайно, все ще величезний комп’ютер, але він включає в себе все, що знадобиться, щоб розпочати експерименти з квантовими обчисленнями, включаючи всю техніку, необхідну для охолодження апаратного забезпечення квантових обчислень.

«IBM Q System One — це великий крок вперед у комерціалізації квантових обчислень, — сказав Арвінд Крішна, старший віце-президент Hybrid Cloud і директор IBM Research.

IBM Q System One була розроблена IBM Research за підтримки Map Project Office та Universal Design Studio. CERN, ExxonMobil, Fermilab, Аргонська національна лабораторія та Національна лабораторія ім. Лоуренса Берклі входять до числа клієнтів, які зареєструвалися для доступу до комп'ютера через хмару.

IBM Research також серйозно просунулися у дослідженнях із створення першого квантового комп'ютеру. Але не лише вони беруть участь у так званих «квантових перегонах». Google та Intel також розробляють власні рішення.

В лютому 2007 року канадська компанія D-Wave заявила про створення зразка квантового комп'ютера, що складається з 16 кубіт (пристрій одержав назву Orion) [4]. Інформація про цей пристрій не відповідала вимогам достовірного наукового повідомлення, тому новина не отримала наукового визнання. Більше того, подальші плани компанії - створити вже в найближчому майбутньому 1024-кубітні комп'ютер - викликали скепсис у членів експертної спільноти.

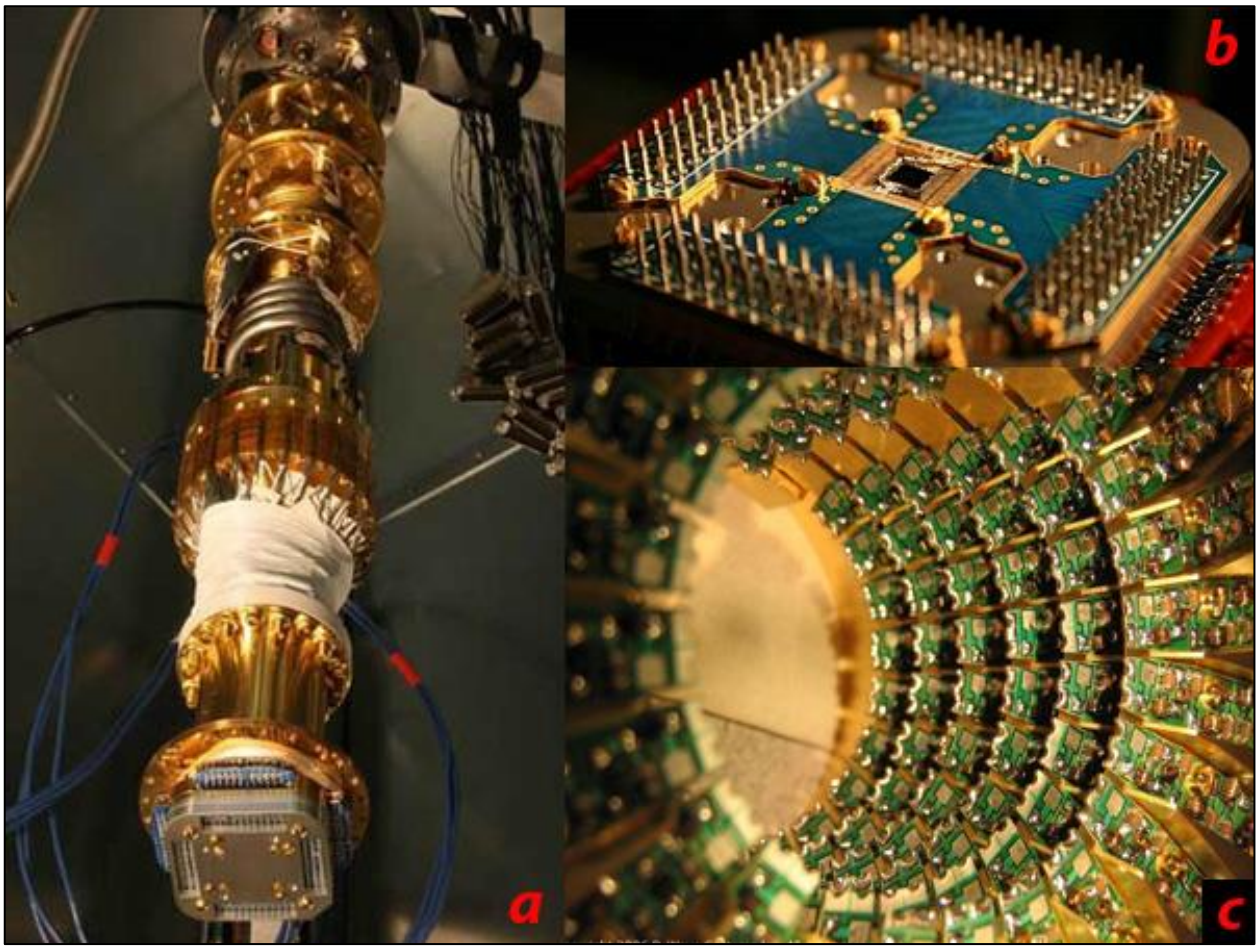

Рисунок 1.3 Елементи квантового комп'ютера D-Wave Orion: a) квантовий процесор в зборі; b) кремнієвий квантовий чіп з 16 кубітами; c) електронні модулі для зв'язку з квантовим чіпом.

11 травня 2011 D-Wave розгорнув апаратне забезпечення, здатне виконувати дедалі складніші завдання. Представлений комп'ютер D-Wave One, створений на базі 128-кубітного процесора, який коштує 11 млн. доларів. Пізніше компанія випустила D-Wave Two з 512-кубітним чіпом. На цьому D-Wave вирішила не зупинятися. У вересні 2016 році у продаж надійшов комп'ютер D-Wave 2000Q з процесором на 2000 кубіт за ціною 15 млн. доларів.

Його робоча температура становить приблизно -273 °C, комп'ютер ретельно екранований від зовнішніх електричних і магнітних полів.

D-Wave 2000Q має 2048 кубітів; суттєве збільшення порівняно з 1000-кубітами D-Wave 2X. Не менш важливо, що фірма вже знайшла клієнта для свого

нового продукту вартістю 15 мільйонів доларів. Temporal Defense Systems використовуватиме машину для захисту певних чутливих державних мереж.

Рисунок 1.4 Квантовий комп'ютер D-Wave D-Wave 2000Q з 2000-кубітним процесором

«Об’єднана потужність рішень D-Wave/D-Wave і cyber quantum зробить революцію в безпечних комунікаціях і допоможе виявити кібер-супротивників», — сказав Джеймс Баррелл, головний технічний директор TDS.

Історично обмеженням комп’ютерів D-Wave було те, що вони вирішували певні класи проблем. Архітектура D-Wave використовує квантовий відпал, і велика кількість дослідницьких проектів задавалися питанням, чи справді ми можемо назвати цю машину квантовою, чи ми просто працюємо над дуже складною емуляцією.

Квантовий відпал принципово відрізняється від класичних обчислень. Він використовує природну тенденцію квантових систем знаходити стани з низькою енергією. Наприклад, якщо задача оптимізації аналогічна пейзажу вершин і долин, кожна координата представляє можливе рішення, а її висота представляє її енергію. Найкращим рішенням є найнижча точка в найглибшій долині ландшафту.

Цікава відмінність між старими системами D-Wave і новою D-Wave 2X полягає в тому, як кубіти з’єднані разом. Ось старий метод:

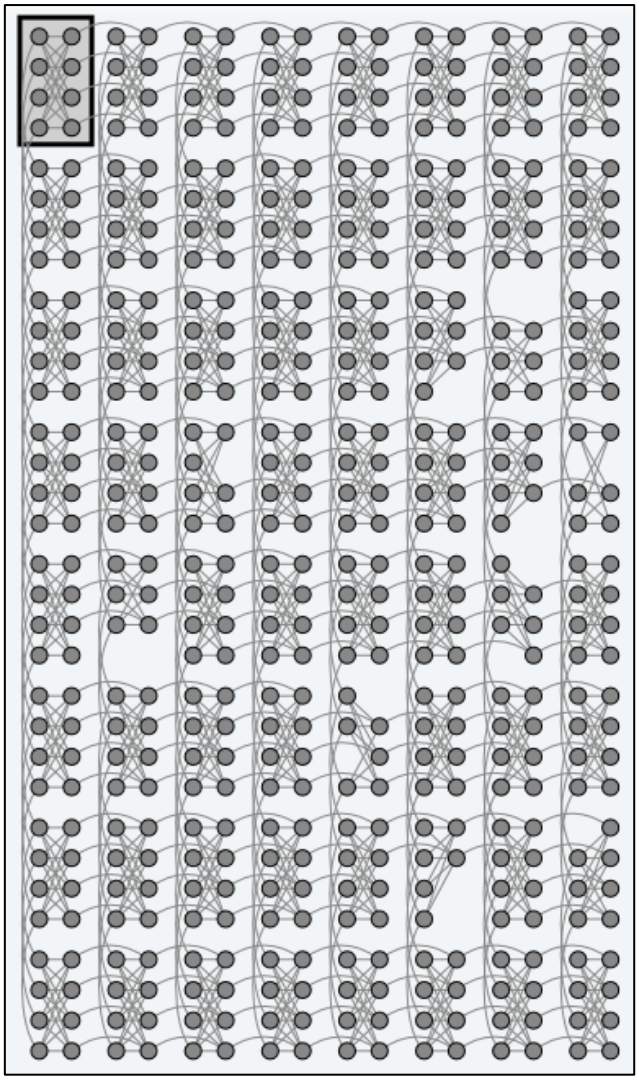

Рисунок 1.5 Система кубітів D-Wave

А ось і система D-Wave 2X.

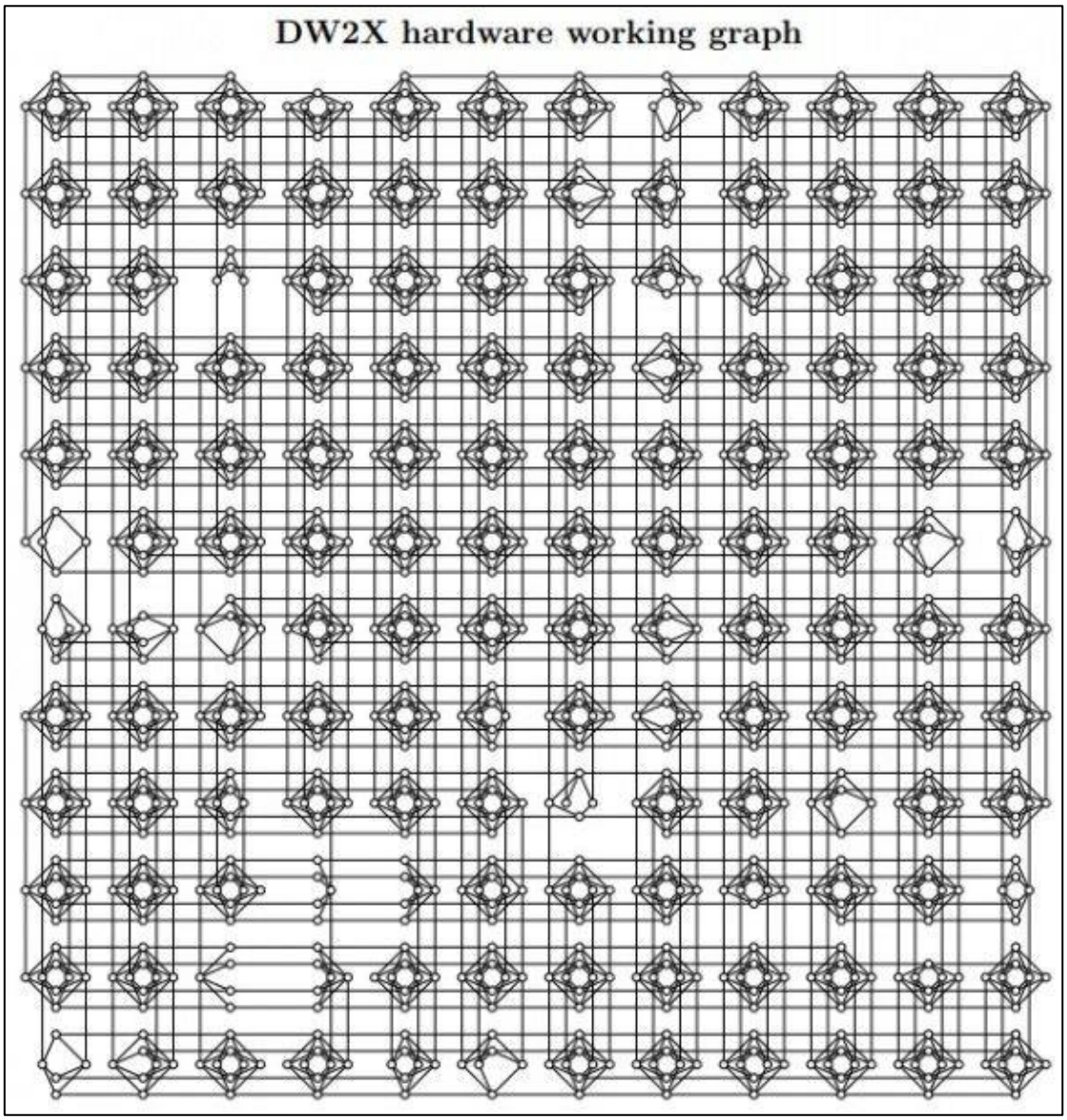

Рисунок 1.6 Система кубітів D-Wave 2X [3]

Мережа все ще слабо підключена, але, здається, міжкубітних з’єднань набагато більше, ніж у попередніх системах. Це повинно призвести до більшої простоти використання D-Wave 2000Q для вирішення різноманітних проблем і дати вченим більше корисних сценаріїв для оцінки продуктивності квантового відпалу порівняно з його класичним аналогом.

До компанії D-Wave System, в розробці і вдосконаленні квантового комп’ютера, приєдналися такі потужні гіганти як Google і NASA, зі своїми передовими технологіями, інтелектуальними і фінансовими можливостями.

У майбутньому квантові комп'ютери обіцяють стати головною обчислювальною силою у вирішенні великої низки проблем. Отже, існує

ймовірність, що в недалекому майбутньому кожен з нас зможе користуватися власним портативним квантовим суперпотужним комп’ютером.

1.2 Пошуки розв’язання проблеми, постквантова криптографія

Завдяки величезній швидкості розкладання на прості множники квантовий комп'ютер дозволить розшифровувати повідомлення, зашифровані за допомогою популярного асиметричного криптографічного алгоритму RSA. До цього часу цей алгоритм вважається порівняно надійним, оскільки ефективний спосіб розкладання чисел на прості множники для класичного комп'ютера нині невідомий. Для того, наприклад, щоб отримати доступ до кредитної картки, потрібно розкласти на два простих множники число завдовжки сотні цифр. Навіть для найшвидших сучасних комп'ютерів виконання цього завдання зайняло більше часу, ніж вік Всесвіту, в сотні разів.

Основним напрямком протидії квантовим алгоритмам криптоаналізу є винайдення нових класів криптосистем, стійкість яких не ґрунтується на таких математичних проблемах як складність факторизації цілого числа чи розв’язку дискретного логарифму. Напрямок таких досліджень названо терміном постквантова криптографія, тобто криптографія, яка буде стійкою навіть після появи квантового комп’ютера, що матиме велику кількість кубітів для своєї роботи. Цей напрямок робіт популяризувався після 2006 року, саме тоді була проведена перша конференція за цією тематикою (PostQuantumCrypto 2006). Така конференція проводиться щорічно і збирає для обговорення найкращі здобутки за рік у напрямку постквантової криптографії.

Отже, враховуючи останні досягнення у напрямку постквантової криптографії, можна виділити такі класи криптосистем, що будуть стійкими до квантового криптоаналізу:

- криптографія на основі решіток;
- мультиваріативна криптографія;
- криптографія на основі геш-функцій;
- криптографія на основі кодів;

- риптографія ізогінеї суперсингулярних еліптичних кривих;
- киметрична криптографія.

Говорячи про криптографію на основі решіток, то це є новітнім класом альтернативних схем відкритого ключа і сьогодні є активною областю для досліджень. Цей тип криптографії заснований на використанні математичних об'єктів, які називаються решітками, і не залежить від складності факторизації чи пошуку дискретного логарифму. Є кілька складних проблеми, які можуть бути використані, щоб побудувати криптосистеми на базі решіток, найпопулярнішою є проблема знаходження найкоротшого вектора. Цей напрямок включає в себе такі криптографічні системи, як NTRU(криптосистема з відкритим ключем із відкритим кодом, яка використовує криптографію на основі решітки для шифрування та дешифрування даних) та GGH(асиметрична криптосистема, заснована на решітках, які можна використовувати для шифрування). Згідно з Micciancio і Регева, криптографію на основі решіток можна розділити на дві категорії: практичні криптосистеми, такі як NTRU, які не володіють доказами безпеки таких схем, і теоретичні, такі як матриці на основі навчання з помилками (Learning with Errors, LWE), які пропонують сильні докази безпеки, але використовують ключі, які є занадто великими для загального використання. З 2008 року ведуться дослідження з об’єднання цих двох категорій, проте на сьогодні ще не вдалося об’єднати ці дві категорії і створити новий клас криптографічних систем на основі решіток, які б діяли ефективно як NTRU і були б доказово стійкими як LWE.

Протягом останніх років було запропоновано низку атак на криптосистему NTRU. Найкращою відомою класичною атакою є атака «зустріч посередині», що була запропонована в Одлижко у 2003 році, з використанням ідеї цієї атаки та квантового алгоритму Гровера у 2012 році була запропонована ефективна квантова атака на NTRU «зустріч посередині», яка в подальшому була удосконалена Вангом. Аналітична часова і просторова складність таких атак наведена у табл. 1.

Аналітична складність атак на NTRU Таблиця 1

	Класична атака «зустріч посередині»	Удосконалена квантова атака «зустріч посередині»
Часова складність	$O(\frac{C_{N/2}^{d/2}}{\sqrt{N}})$	$O(C_{\lfloor N/3 \rfloor}^{\lfloor d/3 \rfloor} \log C_{\lfloor N/3 \rfloor}^{\lfloor d/3 \rfloor}) + \bar{O}(\sqrt{C_{N-\lfloor N/3 \rfloor+1}^{d-\lfloor d/3 \rfloor}})$
Просторова складність	$O(\frac{C_{N/2}^{d/2}}{\sqrt{N}})$	$O(C_{\lfloor N/3 \rfloor}^{\lfloor d/3 \rfloor})$

Результати досліджень показали, що хоча квантова атака і потребує менше операцій, проте вона вимагає все одно субекспоненційної кількості квантових гейтів для реалізації на реальні криптосистеми.

Отже криптографічні конструкції на основі решіток мають великі перспективи для свого використання в інформаційній сфері, так криптосистема NTRU вже стандартизована (стандарт IEEE 1363.1) і використовується для надання послуги конфіденційності.

Для України є важливим розвиток цього напрямку, враховуючи те, що ця криптосистема є стійкою до квантового криптоаналізу, має невеликі розміри загальносистемних параметрів і має великі швидкості зашифрування/розшифрування.

1.2.1Алгоритми Гровера, Шора та CRYSTALS-KYBER

Алгоритм Гровера - квантовий алгоритм швидкого пошуку у невпорядкованій базі даних. Для записів N пошук здійснюється за час O(N1/2) з використанням O(logN) місця[5]. Алгоритм було розроблено Л. Гровером 1996 року.

При існуючих технічних засобах одним із найшвидших алгоритмів пошуку є лінійний пошук, що вимагає O(N) часу. Алгоритм Гровера, який використовує

можливості квантових комп'ютерів, дозволяє вирішити задачу пошуку за час O(N1/2). Доведено, що він є найшвидшим квантовим алгоритмом для пошуку у невпорядкованій базі даних. Також доведено, що немає класичних алгоритмів тієї ж ефективності.

Алгоритм Гровера забезпечує квадратичний приріст швидкості, тоді як деякі інші квантові алгоритми, наприклад алгоритм факторизації Шора, дають експоненційний виграш порівняно з відповідними класичними алгоритмами. Незважаючи на це, квадратичний приріст значний при досить великих значеннях N.

Хоча основним призначенням алгоритму Гровера прийнято вважати пошук у базі даних, точніше його можна охарактеризувати як «навернення функції». Грубо кажучи, маючи функцію y=f(x), яка може бути обчислена з використанням квантового комп'ютера алгоритм Гровера дозволяє обчислити x, знаючи y. Пошук у базі даних співвідноситься зі зверненням функції, яка набуває певного значення, якщо аргумент x відповідає шуканому запису у базі даних.

Алгоритм Гровера також може бути використаний для знаходження медіани та середнього арифметичного числового ряду. Крім того, він може застосовуватися для вирішення NP-повних завдань шляхом вичерпного пошуку серед багатьох можливих рішень. Це може спричинити значний приріст швидкості порівняно з класичними алгоритмами, хоч і не надаючи «поліноміального рішення» у загальному вигляді. У базах даних, які не представлені в явному вигляді оракул викликається для оцінки елемента по його індексу, при цьому читання елементів повної бази даних один за одним і перетворення його в доступне подання може зайняти набагато більше часу, ніж пошук Ґровера.

Для врахування таких ефектів, алгоритм Ґровера можна розглядати як рішення рівняння або задовольняють обмеження. У таких використаннях оракул є способом перевірити обмеження і не пов'язаний з алгоритмом пошуку. Це поділ, як правило, запобігає алгоритмічній оптимізації, в той час як звичайні алгоритми пошуку часто покладаються на такі оптимізації і уникнути повного перебору.

Алгоритм Шора - це квантовий алгоритм факторизації (розкладання числа на прості множники), що дозволяє розкласти число N за час O((log N)3), витративши O(log N) місця.

Значимість алгоритму у тому, що з використанні досить потужного квантового комп'ютера, він уможливить зламування криптографічних систем з відкритим ключем. Наприклад, RSA використовує відкритий ключ N, що є твором двох великих простих чисел. Один із способів зламати шифр RSA - знайти множники N. При досить великому N це практично неможливо зробити, використовуючи відомі класичні алгоритми. Оскільки алгоритм Шора працює лише з квантовому комп'ютері, нині немає технічних засобів, дозволяють за поліноміальний час від довжини числа розкласти досить велике число на множники.

Алгоритм Шора, у свою чергу, використовуючи можливості квантових комп'ютерів, здатний зробити факторизацію числа за поліноміальний час. Це може поставити під загрозу надійність більшості криптосистем з відкритим ключем, що базуються на складності проблеми факторизації чисел.

Як і інші алгоритми для квантових комп'ютерів, алгоритм Шора імовірнісний: він дає правильну відповідь з високою ймовірністю. Можливість помилки може бути зменшена при повторному використанні алгоритму. Тим не менш, оскільки можлива перевірка запропонованого результату (зокрема простоти числа) в поліноміальний час, алгоритм може бути модифікований так, що відповідь, отримана в поліноміальний час, буде вірною з одиничною ймовірністю.

Алгоритм Шора був розроблений Пітером Шором у 1994 році[6]. Через сім років, у 2001 році, його працездатність була продемонстрована групою фахівців IBM. Число 15 було розкладено на множники 3 та 5 за допомогою квантового комп'ютера з 7 кубитами.

Алгоритм Шора ґрунтується на можливості швидко обчислити власні значення унітарного оператора з високою точністю, якщо можна ефективно обчислювати будь-які його ступені. Взявши як такий оператор множення на x за модулем N (цей оператор діє в 2^n мірному просторі, де $2^{n-1} < N \leq 2^n$,

перетворюючи базисний вектор, що відповідає числу a, у базисний вектор, що відповідає числу xa(modN)), ми зможемо обчислити таке n, що $x^n=1(modN)$, що дозволяє (з високою ймовірністю) розкласти N на множники на звичайному комп'ютері.

Kyber — це захищений механізм інкапсуляції ключів (KEM) IND-CCA2, безпека якого ґрунтується на складності вирішення проблеми навчання з помилками (LWE) через модульні решітки[7]. У поданні перелічено три різні набори параметрів, спрямовані на різні рівні безпеки. Зокрема, Kyber-512 націлений на безпеку, приблизно еквівалентну AES-128, Kyber-768 націлений на безпеку, приблизно еквівалентну AES-192, а Kyber-1024 націлений на безпеку, приблизно еквівалентну AES-256.

Kyber походить від методу, опублікованого в 2005 році Одедом Регевом, розробленого розробниками з Європи та Північної Америки, які працюють у різних державних університетах чи дослідницьких установах, або приватних компаніях за фінансування від Європейської Комісії, Швейцарії, Нідерландів, і Німеччина.Вони також розробили пов’язану та додаткову схему підпису Dilithium, як ще один компонент їх «Криптографічного набору для алгебраїчних решіток» (CRYSTALS).

Починаючи з оригінальної роботи Регева, практична ефективність схем шифрування LWE була покращена завдяки спостереженню, що секрет у LWE може походити від того самого розподілу, що й шум, а також помітивши, що «LWE-подібні» схеми можна побудувати за допомогою квадрата (скоріше ніж прямокутна) матриця як відкритий ключ. Іншим удосконаленням було застосування ідеї, яка спочатку використовувалася в криптосистемі NTRU для визначення проблем Ring-LWE і Module-LWE, які використовували поліноміальні кільця, а не цілі числа. CCA-захищений KEM Kyber побудований на основі CPA-захищеної криптосистеми, яка базується на надійності Module-LWE.

Конструкція Kyber сягає корінням у новаторську схему шифрування Regev на основі LWE. Методи PQC-KEM, Kyber широко використовує внутрішнє хешування. У випадку Kyber тут використовуються варіанти Keccak (SHA-

3/SHAKE), серед іншого, для генерації псевдовипадкових чисел. У 2017 році метод було подано до Національного інституту стандартів і технологій США (NIST) для публічного процесу відбору для першого стандарту для квантово-безпечних криптографічних примітивів (NISTPQC).

Це єдиний механізм інкапсуляції ключів, який був обраний для стандартизації в кінці третього раунду процесу стандартизації NIST. Згідно з приміткою до звіту, в якому оголошується рішення, воно залежить від виконання різних угод, пов'язаних з патентами, при цьому NTRU є резервним варіантом. Наразі триває четвертий раунд процесу стандартизації з метою стандартизації додаткового KEM. На другому етапі процесу відбору кілька параметрів алгоритму були налаштовані, і стиснення відкритих ключів було відмінено. Нещодавно NIST звернув особливу увагу на витрати з точки зору часу виконання та складності реалізацій, які маскують час виконання, щоб запобігти відповідним атакам на бокових каналах (SCA).

1.3 Постановка задачі

Як показав аналіз, квантові комп'ютери розвиваються дуже швидко. Наразі різні експерти прогнозують, створення квантових комп'ютерів які можуть роботи такі обчислення вже через 5 років, а інші кажуть, що це ніколи не станеться, але ми маємо бути готовими. Ми не хочемо, щоб це зламало нашу комп'ютерну систему безпеки, тож треба вже сьогодні розглядати задачі, які можуть бути вирішені швидше за допомогою цього класу обчислювальних систем, задача даної роботи – дослідити криптографічні алгоритми, які захищені від квантових атак.

Попередній аналіз показав, що найскорішими алгоритмами є алгоритми класу NTRU, ось чому в даній роботі досліджуються алгоритми саме цього класу.

Необхідно проаналізувати алгоритм KYBER , який є переможцем 3-х раундів всесвітнього конкурсу алгоритмів, та став першим алгоримтом захищений від квантових атак. У 2021 році NIST вирішив, що алгоритм KYBER гідний стандартизації.

В практичній часті роботи треба виконати реалізацію даного алгоритму з метою визначення його продуктивності. Обрати алгоритмічну мову для забезпечення найкращих швидкісних характеристик та виконати реалізацію основних етапів, а саме, Генерації ключів, інкапсуляції та декапсуляції.

2. ОПИС АЛГОРИТМУ KYBER

2.1 Опис алгоритму

Уже на початку 2000-х років криптографи все більше хвилювалися щодо потенційних досягнень квантових обчислень. Оскільки Пітер Шор опублікував свій знаменитий алгоритм Шора, ми знаємо, що достатньо великий квантовий комп'ютер зламав би всі широко використовувані системи відкритих ключів. Це включає конструкцію RSA, кінцевого поля та еліптичної кривої.

Як наслідок, у 2017 році Національний інститут стандартів і технологій закликав створити нові системи відкритих ключів, які можуть протистояти квантовим комп'ютерам. Kyber є такою запропонованою постквантовою схемою. У 2021 році NIST вирішив, що він гідний стандартизації.

Kyber — постквантова система шифрування з відкритим ключем. Його основним варіантом використання є встановлення ключів систем із симетричними ключами в протоколах вищого рівня, таких як TLS, Signal або OpenPGP. Це постквантова система, оскільки Kyber спеціально розроблений для забезпечення безпеки навіть за наявності квантових комп'ютерів. Зазвичай Kyber називають KEM, а не системою шифрування з відкритим ключем (PKE). Причина цього дещо технічна. Скажімо, Kyber використовує техніку для перетворення захищеного PKE IND-CPA на захищений KEM IND-CCA2[8]. Для розробника основна відмінність між PKE і KEM полягає в API. API KEM не дозволяє шифрувати вибрані повідомлення, повідомлення створюється для вас.

KEM пропонує три функції:

- (pk, sk) = KEM_KeyGen(), виводить відкритий і закритий ключ;
- (ct, ss) = KEM_Encrypt(pk), генерує випадкове значення ss і шифрує його в ct за допомогою відкритого ключа pk;
- (ss) = KEM_Decrypt(sk, ct), розшифровує зашифрований текст ct у відкритий текст ss за допомогою закритого ключа sk.

Не важко зрозуміти, що ми можемо перетворити систему Kyber PKE на KEM. Всередині Kyber KEM використовує Kyber PKE, як ми представили його раніше.

2.2 Принцип роботи алгоритму

Щоб отримати інтуїтивне уявлення про Kyber, ми почнемо зі зменшеної версії системи для простішого пояснення . Зменшений Kyber у цьому розділі буде еквівалентний «звичайному» Kyber, за винятком того, що параметри безпеки менші, що робить усе набагато читабельнішим. Крім того, «звичайний» Kyber використовує стиснення для зашифрованих текстів, яке ми тут опустимо, оскільки воно не має відношення до основної криптосистеми.

Kyber — система шифрування з відкритим ключем. Таким чином, він працює з відкритими ключами для шифрування та закритими ключами для дешифрування. Як і в інших системах відкритих ключів, нам потрібен модуль q. Тобто щоразу, коли ми обчислюємо з числами, ми беремо результат за модулем q. Для зменшеного Kyber ми вказуємо цей модуль як q=17. Зараз, на жаль, з Kyber ми маємо справу з поліномами (багато). Оскільки ми будемо множити і складати поліноми, нам також потрібен модуль для них. Інакше степінь стала б завеликою для нас. Модуль полінома, який ми будемо використовувати, такий $f = x^4 + 1$. Для нас не важливо чому f виглядає так, як виглядає. Важливим є той факт, що, беручи поліном за модулем f, ми гарантуємо, що його ступінь (старший показник) буде меншим за 4. Нижче всі обчислення неявно виконуються за модулем q (за коефіцієнтами) і f (на многочленах).

Ми визначили модулі, тож тепер можемо перейти до генерації ключів.

Закритий ключ пари ключів Kyber складається з поліномів з малими коефіцієнтами, тобто це вектор «малих» поліномів. У зменшеному Kyber кожен закритий ключ містить два поліноми. У нашому прикладі ми будемо використовувати закритий ключ:

$$s = (-x^3 - x^2 + x, \quad -x^3 - x)$$

Згенерувати цей закритий ключ просто. По суті, ми використовуємо випадкові малі коефіцієнти. Відкритий ключ Kyber складається з двох елементів. Матриця випадкових поліномів A та вектор поліномів t . Генерація матриці досить

проста, ми просто генеруємо випадкові коефіцієнти та беремо їх за модулем q . Для нашого прикладу ми використаємо:

$$A = \begin{pmatrix} 6x^3 + 6x^2 + 16x + 11 & 9x^3 + 4x^2 + 6x + 3 \\ 5x^3 + 3x^2 + 10x + 1 & 6x^3 + x^2 + 9x + 15 \end{pmatrix}$$

Для обчислення t нам потрібен додатковий вектор помилки e. Цей вектор помилок також складається з поліномів з малими коефіцієнтами, як і закритий ключ. У нашому прикладі ми будемо використовувати вектор помилок:

$$e = (x^2, \quad x^2 - x)$$

Тепер ми можемо обчислити t шляхом множення та додавання матриці:

$$t = As + \mathrm{e}$$

Це робить t вектором поліномів, як s і e. У нашому прикладі ми отримали

$$t = (16x^3 + 15x^2 + 7, \quad 10x^3 + 12x^2 + 11x + 6)$$

Тепер у нас є пара ключів зменшеного Kyber із:

- приватний ключ: s;
- відкритий ключ: (A,t).

Хитрість полягає в тому, що важко відновити s з (A, t). Насправді, для відновлення s вимагатиме від зловмисника вирішення проблеми модульного навчання з помилками (MLWE), на якій побудована ця система. Очікується, що проблема MLWE буде важкою навіть для квантових комп’ютерів, тому вона використовується в пост квантовій криптографії.

Як і в кожній системі шифрування з відкритим ключем, ми можемо зашифрувати повідомлення за допомогою відкритого ключа. Розшифровку можуть здійснювати лише сторони, які володіють закритим ключем. Процедура шифрування використовує помилку та поліноміальний вектор рандомізатора e_1 і r. Ці поліноміальні вектори генеруються кожного разу для кожного шифрування. Крім того, нам потрібен поліном помилки e_2. Поліноми в межах e_1, e_2 і r абсолютно випадкові та малі, як і в s.

У нашому прикладі ми будемо використовувати:

$$r = (-x^3 + x^2, x^3 + x^2 - 1)$$
$$e_1 = (x^2 + x, x^2)$$
$$e_2 = -x^3 - x^2$$

Тепер, щоб зашифрувати повідомлення, ми повинні перетворити його на поліном. Ми робимо це, використовуючи двійкове представлення повідомлення. Кожен біт повідомлення використовується як коефіцієнт. У нашому прикладі ми хочемо зашифрувати число 11. Одинадцять має двійкове представлення 1011, $(11)_{10} = (1011)_2$ Тому наше повідомлення, закодоване як двійковий поліном, таке:

$$m_b = 1x^3 + 0x^2 + 1x^1 + 1x^0 = x^3 + x + 1$$

Перед шифруванням ми повинні масштабувати цей поліном. Ми збільшуємо m_b, множачи його на $\left\lfloor \frac{q}{2} \right\rfloor$, тобто на ціле число, найближче до $\frac{q}{2}$. Це зроблено тому, що коефіцієнти полінома повинні бути великими. У частині дешифрування ми побачимо, чому це необхідно. У нашому прикладі з q = 17, $\left\lfloor \frac{q}{2} \right\rfloor$ = 9. Отже, наше остаточне готове до шифрування повідомлення таке:

$$m = \left\lfloor \frac{q}{2} \right\rfloor m_b = 9m_b = 9x^3 + 9x + 9$$

Ми шифруємо m за допомогою відкритого ключа (A, t). Процедура шифрування обчислює два значення (u, v).

$$u = A^t r + e_1$$
$$v = t^T r + e_2 + m$$

У нашому прикладі ці значення:

$$u = (11x^3 + 11x^2 + 10x + 3, 4x^3 + 4x^2 + 13x + 11)$$
$$v = 7x^3 + 6x^2 + 8x + 15$$

Зашифровані тексти Kyber складаються з цих двох значень: (u,v) Поліноміальний вектор u і поліном v.

За наявності приватного ключа s і зашифрованого тексту (u, v) розшифровка є простою. Спочатку ми обчислюємо шумовий результат m_n.

$$m_n = v - s^T u$$

Цей результат є зашумленим, оскільки обчислення насправді не дає вихідного повідомлення m. Дивлячись на рівняння, ми можемо побачити, що:

$$m_n = e^T r + e_2 + m + s^T e_1$$

Тепер стає зрозуміло, чому нам потрібно масштабувати m, роблячи його коефіцієнти великими. Якщо ви пам’ятаєте, усі інші доданки в рівнянні були обрані малими. Отже, коефіцієнти m_n або близькі до $\left\lfloor \frac{q}{2} \right\rceil$ = 9. маючи на увазі, що вихідний двійковий коефіцієнт m_b був 1 або близький до 0, тобто вихідний двійковий коефіцієнт був 0.

У нашому прикладі ми маємо $m_n = 7x^3 + 14x^2 + 7x + 5$.

Ми можемо відновити вихідне масштабоване повідомлення m, перебравши коефіцієнти m_n і перевіривши, чи вони ближчі до $\left\lfloor \frac{q}{2} \right\rceil$ = 9 чи 0 (або еквівалентно q).

Отже, давайте зробимо це для всіх коефіцієнтів:

- 7, ближче до 9, ніж 0 або p, округлення до 9;
- 14, ближче до p, ніж 9, округлити до 0;
- 7, ближче до 9, ніж 0 або p, округлення до 9;

– 5, ближче до 9, ніж 0 або p, округлити до 9.

Наш округлений поліном дорівнює $9x^3+0x^2+9x+9$, тобто масштабований поліном, який ми зашифрували! Тепер ми можемо просто відновити вихідний двійковий поліном m_b, зменшивши його з коефіцієнтом $\frac{1}{9}$.

$$m_b = \frac{1}{9}(9x^3 + 0x^2 + 9x + 9) = (1x^3 + 0x^2 + 1x + 1)$$

З m_b ми можемо просто прочитати біти вихідного повідомлення, які $(1011)_2 = (11)_{10}$. Отже, відновлений відкритий текст: 11!

2.3 Безпека Kyber

За своєю суттю Kyber — це, по суті, система шифрування Любашевського, Пейкерта, Регева (LPR) у налаштуваннях модульного навчання з помилками (MLWE). Оригінальні схеми LPR були визначені в налаштуваннях LWE та Ring LWE. Різниця між ними проста:

– LWE працює з векторами цілих чисел;
– RING LWE працює з поліномами;
– модуль LWE працює з векторами поліномів.

Хоча вектори поліномів, безумовно, неінтуїтивно зрозумілі для роботи, вони дозволяють набагато краще поєднувати швидкість/розмір/безпеку.

Безпека MLWE (і, отже, Kyber) фактично базується на проблемі решітки, а саме задачі найкоротшого вектора (SVP). Отже, відновлення повідомлення із зашифрованого тексту (або приватного з відкритого ключа) має бути таким же важким, як вирішення великого екземпляра SVP. Це має бути обчислювально нездійсненним навіть для квантового комп'ютера. Зменшення від Kyber до MLWE до SVP є нетривіальним, але добре задокументованим. Якщо вас цікавлять деталі, я рекомендую прочитати публікацію Kyber NIST і слідкувати за цитатами.

Основна перевага схем на основі решітки, таких як Kyber, полягає в тому, що вони дуже швидкі, особливо порівняно з іншими постквантовими системами. Недоліком є те, що не зовсім зрозуміло, чи можна використовувати додаткову

структуру решіток основного модуля для формулювання кращих атак на конкретний екземпляр SVP.

2.4 Порівняння безпеки та продуктивності алгоритму

Kyber доступний у трьох рівнях безпеки. Компроміси розміру та безпеки наведено в наступній таблиці з RSA як попередньо-квантове порівняння.

Порівняння Kyber та RSA Таблиця 2

Версія	Рівень безпеки	Розмір приватного ключа	Розмір публічного ключа	Розмір зашифрованого тексту
Kyber512	AES128	1632	800	768
Kyber768	AES192	2400	1184	1088
Kyber1024	AES256	3168	1568	1568
RSA3072	AES128	384	384	384
RSA15360	AES256	1920	1920	1920

Хоча ключі RSA все ще менші, розміри ключів Kyber залишаються такими ж. Це не дано, оскільки деякі системи PQC мають ключі в сотні кілобайт, а у випадку класичного McEliece навіть у мегабайтному діапазоні.

Тепер розглянемо гістограму розмірів відкритого ключа, закритого ключа та зашифрованого тексту еліптичної кривої (див. рис. 2.1). Для порівняння було обрано три схеми шифрування інформації : RSA який є із найбільш відомих і широко використовуваних алгоритмів шифрування в сучасній криптографії, функції еліптичної кривої Curve25519 та brainpoolIP512r1, та наш алгоритм Kyber.

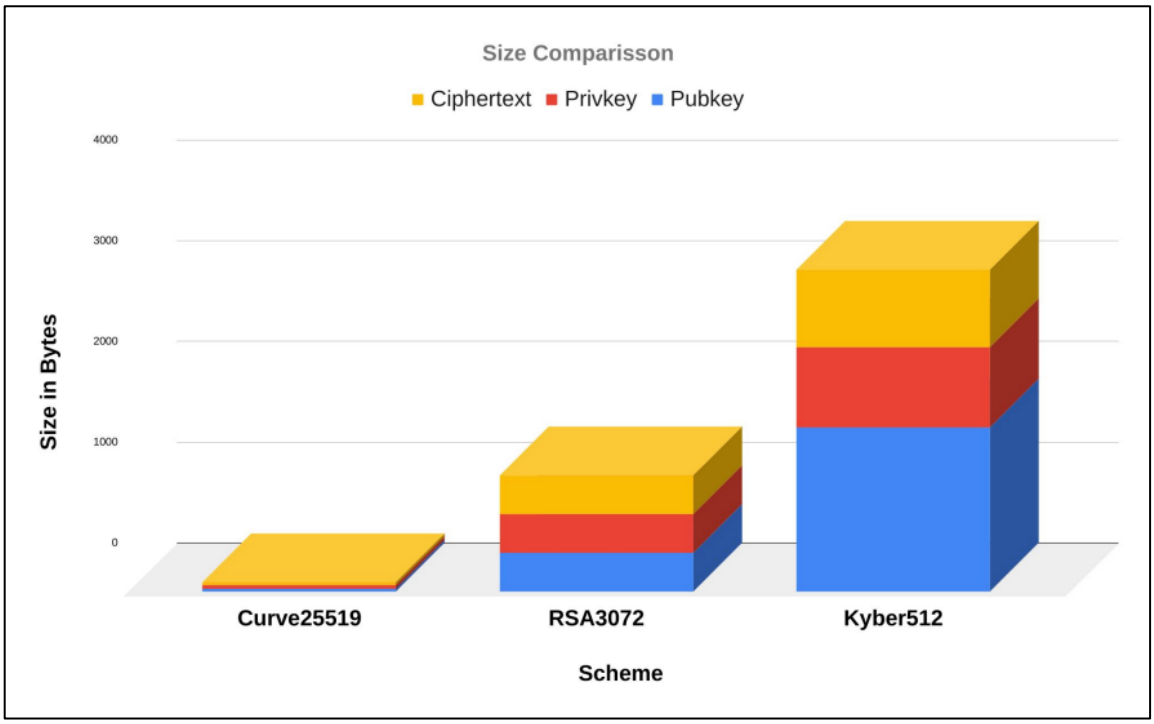

Рисунок 2.1 Гістограма розмірів: Curve25519, RSA та kyber з найменшим рівнем захисту

Тут порівнюються три схеми Curve25519, RSA та kyber із найменшим рівнем безпеки. Отже, ці три схеми мають однаковий рівень безпеки, але, як можна бачити, функція Curve25519 доволі маленька за розміром відносно інших схем, RSA дещо більше, Kyber ще більший. Виходячи з аналізу гістограми можна зробити висновок, що незважаючи на низький рівень безпеки Kyber потребує пам’яті найбільше.

Перейдемо до найвищого рівня безпеки. Дивлячись на гістограму зображну на рис. 2.2 можна одразу сказати, що при такому рівні безпеки Kyber насправді дуже схожий на RSA. Однак крипто функція з еліптичної кривою все ще залишається схемою з найменшим розміром відкритого ключа, закритого ключа та зашифрованого тексту еліптичної кривої.

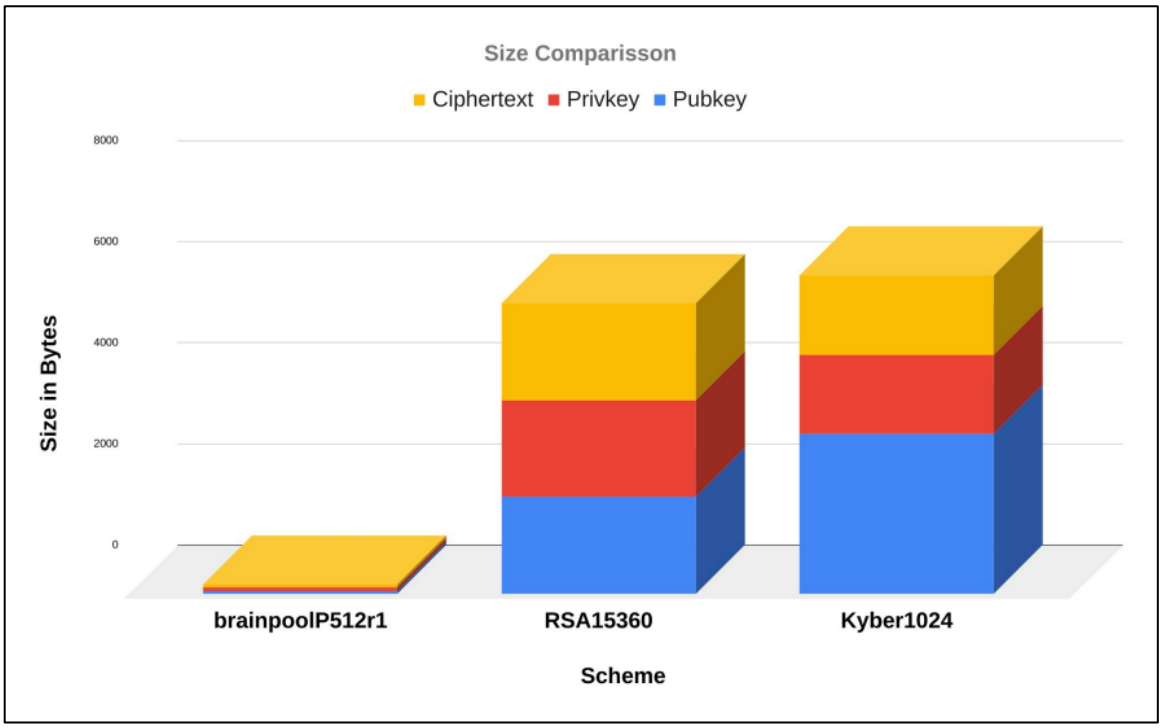

Рисунок 2.2 Гістограма розмірів brainpoolIP512r1, RSA та kyber з найвищим рівнем захисту

Об’єм пам’яті необхідних для роботи алгоритму безперечно важливий аспект, але нас цікавлять не лише розміри, а ще і швидкість. На наступному рисунку 2.3 можемо побачити гістограму порівняння схем за швидкостю шифрування та дешифрування ключів.

Ми бачимо, що однаковий рівень безпеки в Kyber, в функції крипто з еліптичною кривою - Curve25519 та в RSA, але вже можна з великою вірогідністю сказати, що швидкість алгоритму RSA сильно відстає від інших.

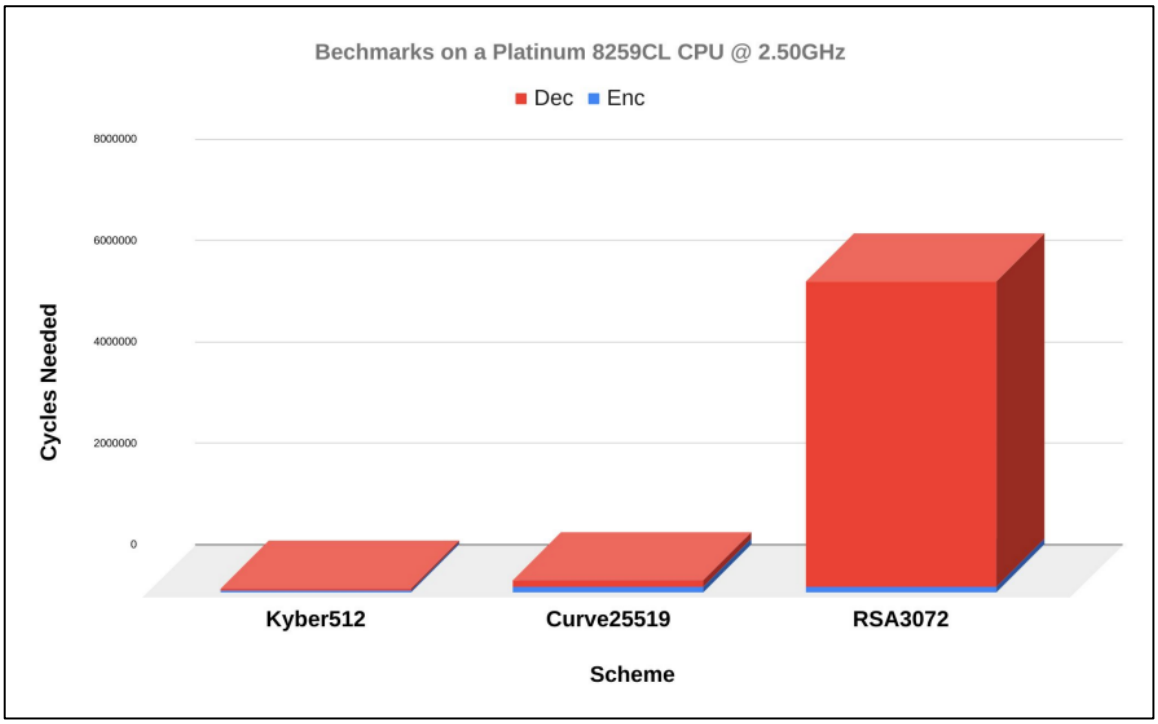

Рисунок 2.3 Гістограма порівняння продуктивності: Curve25519, RSA та kyber з найменшим рівнем захисту

На рисунку 2.4 маємо порівняння двох схем вже без RSA. У порівнянні з Curve25519 алгоритм Kyber виявився набагато швидшим.

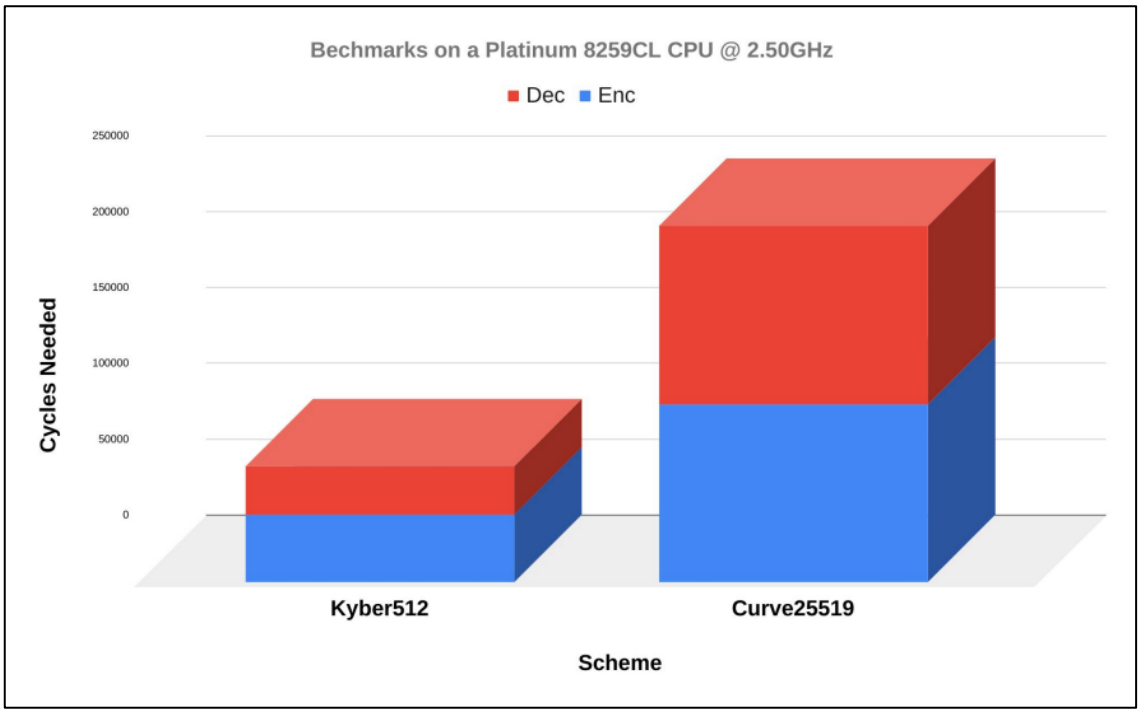

Рисунок 2.4 Гістограма порівняння продуктивності: Curve25519, RSA та kyber з найменшим рівнем захисту

На рисунку 2.5 маємо порівняння трьох схем, Curve25519, kyber512 та kyber1024. У порівнянні з Curve25519 алгоритм Kyber навіть при найвищому рівні безпеки виявився швидше за функцію Curve25519. Найшвидшою схемою вважається kyber512.

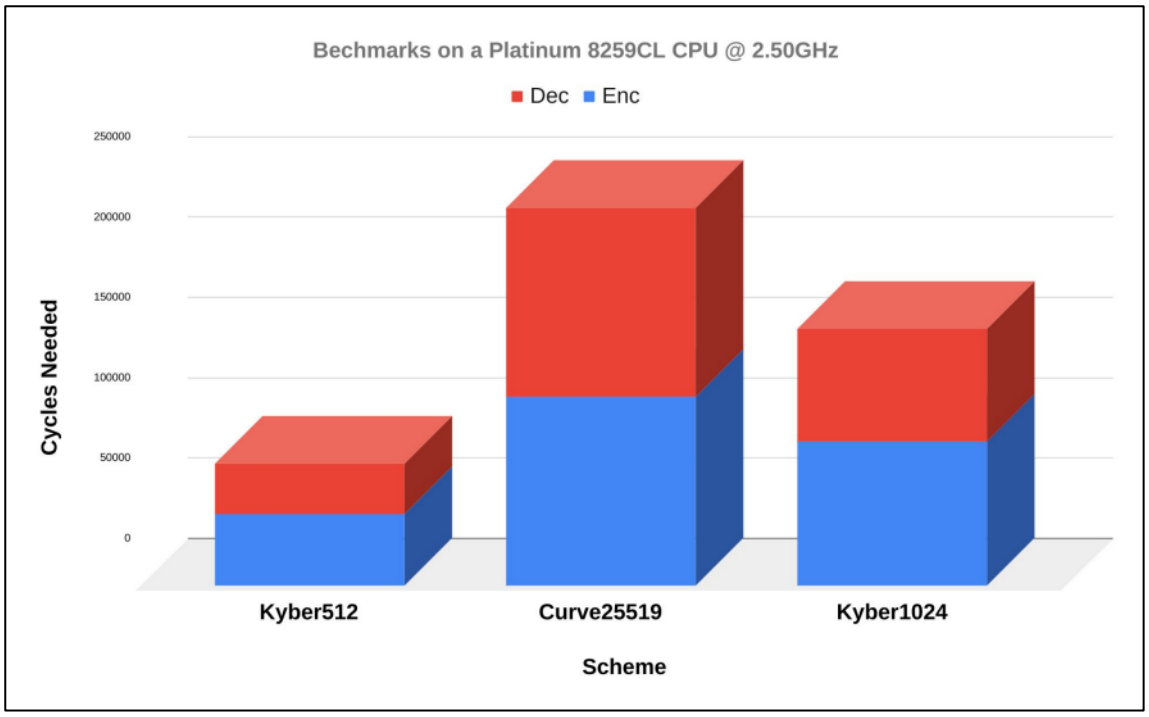

Рисунок 2.5 Гістограма порівняння продуктивності: Curve25519, RSA та kyber з найменшим та найвищим рівнем захисту

Така різниця у швидкості між RSA та Curve25519 з Kyber полягає у тому, що останні використовують шифрування з відкритим ключем на основі решіток, що забезпечує високу швидкість для шифрування та дешифрування приватного та публічного ключів. Тоді як принцип роботи RSA ґрунтується на труднощах факторизації великих простих чисел, що є по суті доволі витратною дією. Це порівняння виявило, що алгоритм Kyber через свою високу продуктивність та відносно невеликий розмір має шанси стати широко використовуваним алгоритмів криптографічного шифрування.

Щоб отримати інтуїтивне уявлення про Kyber, ми почнемо зі зменшеної версії системи, яку назвемо малий Kyber. Малий Kyber еквівалентний «звичайному» Kyber, за винятком того, що параметри безпеки менші, що робить усе набагато читабельнішим. Крім того, «звичайний» Kyber використовує стиснення для зашифрованих текстів, яке ми тут опустимо, оскільки воно не має відношення до основної криптосистеми.

3. ОПИС ПРОГРАМНОЇ СИСТЕМИ

3.1 Параметри

3.1.1 Основні параметри

Основні параметри задані в таблиці 3.1.

Основні параметри Таблиця 3.1

	KYBER_N	KYBER_K	KYBER_Q	KYBER_NJU1, KYBER_NJU2	KYBER_DU, KYBER_DV
512	256	2	3329	3,2	(10, 4)
768	256	3	3329	2,2	(10, 4)
1024	256	4	3329	2,2	(11, 5)

3.1.2 Додаткові параметри

Додаткові параметри задані в таблиці 3.2

Додаткові параметри Таблиця 3.2

Параметр	Значення або вираз		
	KYBER 512	KYBER 768	KYBER 1024
KYBER_SYMBYTES	32		
KYBER_POLYBYTES	KYBER_N * $\log_2$KYBER_Q		
KYBER_POLYVECBYTES	KYBER_K * KYBER_POLYBYTES		
KYBER_POLYCOMPRESSEDBYTES	128	128	160
KYBER_POLYVECCOMPRESSEDBYTES	320* KYBER_K	320* KYBER_K	352* KYBER_K
KYBER_INDCPA_BYTES	KYBER_POLYVECCOMPRESSEDBYTES+ KYBER_POLYCOMPRESSEDBYTES		
KYBER_INDCPA_MSGBYTES	KYBER_SYMBYTES		
KYBER_INDCPA_PUBLICKEYBYTES	KYBER_POLYVECBYTES + KYBER_SYMBYTES		

Продовження таблиці

Параметр	Значення або вираз
KYBER_INDCPA_SECRETKEYBYTS	KYBER_POLYVECBYTES
KYBER_SECRETKEYBYTES	KYBER_INDCPA_SECRETKEYBYTES+ KYBER_INDCPA_PUBLICKEYBYTES+ 2 * KYBER_SYMBYTES
KYBER_PUBLICKEYBYTES	KYBER_INDCPA_PUBLICKEYBYTES
KYBER_CIPHERTEXTBYTES	KYBER_INDCPA_BYTES

3.1.3.1 Класи алгоритмів

Для кожного набору параметрів є 2 класи алгоритмів:

- алгоритм за замовчуванням;
- алгоритм, позначений як KYBER_90S.

Різниця між цими алгоритмами представлена в табл. 1

Різниця між алгоритмом за замовченням та алгоритмом KYBER_90S

Таблиця 3.3

Тип алгоритму	Гешування	Потокове шифрування
За замовченням	H - SHA3-256 G - SHA3-512	XOF - SHAKE-128 KDF - SHAKE-256 PRF - SHAKE-256
KYBER_90S	H - SHA-256 G - SHA-512	XOF - AES-256 в режимі CTR KDF - SHAKE-256 PRF - AES-256 в режимі CTR

3.2 Генерація ключів

Функція crypto_kem_keypair призначена для генерації секретного та відкритого ключа та запису їх у вигляді рядка байтів.

Вхід – немає.

Вихід:

pk – відкритий ключ, рядок байтів завдовжки KYBER_PUBLICKEYBYTES.

sk – секретний ключ, рядок байтів завдовжки KYBER_SECRETKEYBYTES.

Спершу відбувається генерація компонентів owner_pk, owner_sk. Маємо owner_pk, owner_sk = indcpa_keypair(). Наступним кроком розраховуємо секретний кллюч sk = owner_sk + owner_pk + H(owner_pk) + KYBER_RO та відкритий ключ pk = owner_pk. Довжина секретного ключа: KYBER_INDCPA_SECRETKEYBYTES+, KYBER_INDCPA_PUBLICKEYBYTES + , KYBER_SYMBYTES +,KYBER_SYMBYTES. Для генерації компонентів ключів використовуємо функцію indcpa_keypair. Першим кроком є виділення пам’яті для buf завдовжки 2 * KYBER_SYMBYTES. Перша частина (publicseed) для випадкового рядка завдовжки KYBER_SYMBYTES, друга частина для noiseseed; Далі відбувається генерація псевдовипадкового масиву байтів завдовжки KYBER_SYMBYTES (buf) та обчислення гешу для buf завдовжки KYBER_SYMBYTES біт (hash_g, SHA3-512). Результат в buf завдовжки 2 * KYBER_SYMBYTES. Далі робится генерація квадратної матриці розміром KYBER_K * KYBER_K . На основі publicseed з функції gen_ matrix генеруємо вектори шуму (s, e) з маленькими числами (функція poly_getnoise_eta1), розмір вектору KYBER_K. Проводиться обчислення вектору помилки t = (A * s + e) mod KYBER_Q. Тож у результаті маємо sk = pack_sk (s, e) та pk = pack_pk (t).

Генерацією матриці займається функція gen_matrix.

Вхід:

seed – псевдо випадковий рядок завдовжки KYBER_SYMBYTES;

transposed – задає тип матриці, звичайна (transposed = 0), транспонована (transposed = 1).

Вихід:

A – матриця розміром KYBER_K * KYBER_K, елементи матриці – поліноми.

Для кожного елементу матриці формується стан генератору напів випадкових чисел (функція xof_absorb, параметри: state, seed, I, j). У разі транспонованої матриці параметри функції xof_absorb : state, seed, j, i.Ініціалізація виконується окремо для кожного елементу матриці. Ініціалізація виконується лише один раз.

Кількість елементів, які формуються менше в середньому на x процентів. Чи є залежність матриці від часу виконання – це фактично public елемент, який не впливає на значення компонентів секретного ключа. На основі отприманого стану формується буфер, який має фіксовану кількість блоків (розмір блоку 168 байтів), для отримання коефіцієнтів поліному (функція xof_squeezeblocks, параметри buf, state)

Далі формуються безпосередньо коефіцієнти поліному (функція rej_uniform), розмір елементу 12 бітів (q =3329, 3329/4096 = 81%).Якщо виділити 47 бітів, то буде задано 4 числа, імовірність відміни числа дорівнює 122 816 065 582 081 : 140,737,488,355,327=0.8727, але 4 ділення. Якщо кількість коефіцієнтів менше ніж KYBER_N, виконується перехід на крок 3.

Генерацією елементів матриці робіть функція rej_uniform. Для генерації наступного елементу поліному виділяється наступні 12 бітів числа. Якщо число менше q, воно приймається.

Генерація векторів з малими коефіцієнтами виконують функції poly_getnoise_eta1, poly_getnoise_eta2. Виконується для кожного компоненту вектору.Генерація на основі псевдовипадкового рядка noiseseed та значення nonce.

Послідовність генерації векторів з малими коефіцієнтами:

- генерація буферу завдовжки KYBER_ETA1*KYBER_N/4– функція prf;
- генерація поліному на основі буферу (функція cbd_eta1 для функції poly_getnoise_eta1 та функція cbd_eta2 для функції poly_getnoise_eta2).

Генерацію коефіцієнтів малих поліномів виконують функції cbd_eta1, cbd_eta2

В залежності від кількості бітів для коефіцієнту (2, 3) застосовують функції cbd2 або cbd3.

Хід роботи функції cbd3:

- виділяється 24 біт;
- обчислюється сума бітів в кожній трійці, всього отримаємо 8 значень;
- обчислюємо різницю для кожної пари, отримаємо 4 коефіцієнта.

Хід роботи функції cbd2:

- для кожних 4 байтів виділяються усі біти (0x55555555) та після зсуву на 1 біт;
- обчислюється сума;
- далі бітовий рядок з 32 бітів розглядається як масив бітових рядків по 4 біта, всього таких рядків 8;
- кожний рядок застосовується для обчислення одного коефіцієнту поліному;
- обчислюється коефіцієнт полінома c = a – b.

Значення 0, 1 біта в кожному рядку це a, а значення 2, 3 біта в кожному рядку це b. Функція cbd2 застосовується завжди для функції poly_getnoise_eta2.

Пакування секретного ключа виконує функція pack_sk.

Вектор s, який є нашим секретним ключем, пакується в NTT форматі. Для пакування вектору застосовується функція polyvec_tobytes. Функція для кожного компоненту вектору застосовує функцію poly_tobytes.

Функція poly_tobytes виконує пакування поліному.

Послідовність дій функції poly_tobytes:

- коефіцієнти поліному приводяться до інтервалу 0..q-1, якщо вони перевищують q;
- кожний коефіцієнт записується в 12 біт.

Пакування відкритого ключа виконує функція pack_pk.

Послідовність дій функції pack_pk:

- пакується вектор t (Функція polyvec_tobytes);

– додається значення seed завдовжки KYBER_SYMBYTES для відновлення матриці.

3.3 Зашифрування

Функція crypto_kem_enc виконує шифрування завдяки відкритому ключу.

Вхід:

Pk – відкритий ключ.

Вихід:

Ct – результат зашифрування.

Ss – shared key.

buf – буфер завдовжки 2*KYBER_SYMBYTES.

kr – буфер завдовжки 2*KYBER_SYMBYTES.

Послідовність дій функції crypto_kem_enc:

– генерація випадкового рядка завдовжки KYBER_SYMBYTES та запис їх в buf;
– обчислення гешу від випадкового рядка за допомогою функції H та запис його замість попереднього в buf
– обчислення гешу від відкритого ключа за допомогою функції H та запис його в другу половину buf
– обчислення гешу від всього буферу та запис його (функція G) та запис його в рядок kr.
– шифрування буферу.

У якості повідомлення для шифрування обирається буфер завдовжки KYBER_SYMBYTES (buf), задається відкритий ключ та випадкові дані. В якості випадкових даних обирається друга половина kr. Функція indcpa_enc. Результат шифрування в ct Обчислення гешу для ct за допомогою функції H і запис результату замість другої половини kr.Обчислення ss за допомогою функції kdf. В якості параметру функції – kr.

Шифрування отриманої послідовності робить функція indcpa_enc.

Вхід:

M – повідомлення для шифрування завдовжки 32 байта.

Pk – відкритий ключ.

Coins – випадкове дане завдовжки KYBER_SYMBYTES.

Вихід:

Cc – результат шифрування повідомлення M.

Послідовність дій функції indcpa_enc:

- виділяється пам'ять для seed завдовжки KYBER_SYMBYTES та задається початкове значення nonce = 0;
- розпаковка відкритого ключа, формування вектору та випадкового даного. Вектор t записується в pkpv, а випадкове дане в seed (функція unpack_pk);
- згідно повідомлення M формується поліном k = poly_frommsg(M); (функція poly_frommsg);
- відновлюється матриця A згідно seed в транспонованому вигляді (функція gen_matrix для транспонованої матриці);
- відновлюються вектори s = poly_getnoise_eta1 (Coins, nonce), e = poly_getnoise_eta2 (Coins, nonce); (функції poly_getnoise_eta1, poly_getnoise_eta2);
- формується поліном ep = poly_getnoise_eta2 (Coins, nonce) (функція poly_getnoise_eta2);
- обчислюємо b = A * s;
- обчислюємо v = pkpv * s;
- обчислюємо b += e;
- обчислюємо v += ep;
- обчислюємо v += k;
- формуємо загальний секрет cc по b та v (функція pack_ciphertext).

У результаті отримуємо наш шифрований текст cc = pack_ciphertext (b, v).

Функція unpack_pk займається розпаковкою відкритого ключа.

Вхід:

Packedpk – Рядок байтів завдовжки KYBER_INDCPA_PUBLICKEYBYTES;

Вихід:

Pk – Вектор завдовжки K поліномів;

Seed – Випадковий рядок для генерації матриці завдовжки KYBER_SYMBYTES.

Послідовність дій функції unpack_pk:

- формування вектору поліномів Pk з перших KYBER_POLYVECBYTES байтів Packedpk(Функція polyvec_frombytes);
- копіювання останніх KYBER_SYMBYTES байтів в Seed.

Формуванням вектору поліномів займається функція polyvec_frombytes

Вхід:

A -рядок байтів завдовжки KYBER_POLYVECBYTES.

Вихід:

R - Вектор поліномів.

Рядок ділиться на порції завдовжки KYBER_POLYBYTES. Для кожного з KYBER_K поліномів робится виконання функції poly_frombytes. Формування поліному з байтового рядка. Функція poly_frombytes. Для формування кожної пари коефіцієнтів виділяється 3 байта (24 біта), молодші 12 бітів задають молодший коефіцієнт пари, старші – старший коефіцієнт пари.

Заявдяки функції poly_frommsg формуємо формується коефіцієнт поліному.

Для наступного біту повідомлення (його довжина завжди дорівнює 32 байта або 256 бітів) формується коефіцієнт поліному, який дорівнює 0 для біта = 0 та (q+1)/2 для одиничного біту.

Формуванням інкапсульованого ключа займається функція pack_ciphertext.

Вхід:

B – вектор поліномів;

V – поліном.

Вихід:

R – рядок байтів завдовжки KYBER_INDCPA_BYTES.

Спершу робиться запис перших KYBER_POLYVECCOMPRESSEDBYTES байтів за допомогою функції polyvec_compress. Далі відбувається запис KYBER_INDCPA_BYTES - KYBER_POLYVECCOMPRESSEDBYTES байтів за допомогою функції poly_compress.

Перетворенням вектору в рядок байтів займається функція polyvec_compress.

Вхід:

A – вектор поліномів.

R – рядок байтів.

Спершу робится приведення коефіцієнтів поліномів вектору до інтервалу 0.. q-1.Якщо KYBER_POLYVECCOMPRESSEDBYTES == (KYBER_K * 352), для числа відводиться 11 біт для кожного елементу вектору для кожного елементу поліному r = (r * 2048 + q/2)/q mod 2048 запис r в наступні 11 бітів інакше для числа відводиться 10 біт для кожного елементу вектору, для кожного елементу поліному отже маємо r = (r * 1024 + q/2)/q mod 1024 запис r в наступні 10 бітів.

Перетворення поліному в рядок байтів виконує функція poly_compress.

Вхід:

A - поліном

R – рядок байтів

Спершу приводимо коефіцієнти поліному до інтервалу 0.. q-1, а якщо (KYBER_POLYCOMPRESSEDBYTES == 128) то для кожного елементу поліному r = (r * 16 + q/2)/q mod 16 та робимо запис r в наступні 4 біта. Інакше (KYBER_POLYCOMPRESSEDBYTES == 160) для кожного елементу поліному r = (r * 32 + q/2)/q mod 32 та робимо запис r в наступні 5 біта.

3.4 Розшифрування

Генерація загального секрету по зашифрованому тексту та відкритому ключу відбувається за допомогою функції crypto_kem_dec.

Вхід:

Ct – зашифрований текст.

Sk – секретний ключ.

Вихід:

Ss - Загальний секрет завдовжки CRYPTO_BYTES.

Спершу проводиться обчислення зашифрованого повідомлення M та запис його в буфер завдовжки 2*KYBER_SYMBYTES в перші 32 байти (функція indcpa_dec). Далі відбувається доповнення буферу гешем відкритого ключа.Після виконується обчислення гешу буферу, в якому повідомлення та геш відкритого ключа (функція G). Наступим кроком є зашифрування отриманої послідовності (функція indcpa_enc). Робимо перевірка коректності за допомогою функції verify.

Після перевірка коректності робимо обчислення гешу зашифрованого повідомлення. Викликаємо функція H та запис її замість випадкової послідовності на кроці зашифрування отриманої послідовності. Завдяки функції cmov робимо відновлення рядка для формування загального секрету. Формуємо загальний секрету функцією kdf.

Розшифрувати повідомлення можемо заявдяки функції indcpa_dec.

Вхід:

C – зашифроване повідомлення;

Sk – секретний ключ.

Вихід:

M - повідомлення

Відбувається розпаковка C та виділення вектору bp та поліному v (функція unpack_ciphertext). Далі відбвається розпаковка секретного ключа та обчислення вектору поліномів s (Функція unpack_sk).I нарешті можемо обчислити зашифроване повідомлення mp = v - s * bp. Останнім кроком обчислюємо повідомлення M = poly_tomsg (mp).

Розпаковкою зашифрованого повідомлення займається функція unpack_ciphertext.

Вхід.

C – зашифроване повідомлення

Вихід.

Bp – вектор поліномів;

V – поліном.

Для витягу вектору поліномів використовуємо функцію polyvec_decompress передаючи у параметри зашифроване повідомлення Bp = polyvec_decompress (C).

А для витягу поліномів використовуємо функцію poly_decompress V = poly_decompress (C + KYBER_POLYVECCOMPRESSEDBYTES).Функції polyvec_decompress та poly_decompress зворотні до функцій polyvec_compress та poly_compress віфдповідно.

Розпаковкою секретного ключа займається функція unpack_sk.

Вхід:

Pack_sk – запакований секретний ключ.

Вихід:

sk – вектор поліномів.

Для розпаковки застосовується функція polyvec_frombytes, яка є зворотньою до функції polyvec_tobytes.

Для перевірки результату використовуємо функцію verify.

Вхід:

a – заданий рядок з зашифровапними даними.

b – обчислений рядок з зашифровапними даними.

Вихід:

Success = {OK, ERROR}.

Функція порівнює отриманий результат зашифрування (a) та обчислений (b). Повертає OK в разі успішної перевірки та ERROR в разі помилки. Формуванням результату займається функція cmov. Функція записує в поле результату обчислене значення загального секрету в разі успіху та задане значення в разі помилки.

ВИСНОВКИ

Результатом цієї роботи є аналіз постквантового алгоритму Kyber. Kyber – це дуже швидкий алгоритм на основі решітки, стандартизований як KEM. Він враховує загальні характеристики, що впливають на захищеність алгоритмів сучасності та існуючи атаки, а також він є стійким до майбутніх квантових атак. Його ключі більші, ніж у доквантових схем, але досить малі, щоб їх можна було використовувати в реальних системах. Безпека Kyber заснована на складності проблеми MLWE, яка, у свою чергу, базується на складності проблеми SVP. Це робить Kyber цікавим кандидатом для багатьох програм постквантової криптографії. Алгоритм має високий рівень безпеки та високу продуктивність при відносно немаленьких ключах. При порівнянні швидкості дії алгоритму з іншими алгоритмами, враховуючи рівень безпеки, Kyber виявився найшвидшим. Kyber є перспективним кандидатом для використання в квантовій криптографії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Know How – URL: https://knowhow.pp.ua/q-system-one-germany/ (дата звернення: 20.04.2023)

2. Root Nation - URL: https://root-nation.com/ua/articles-ua/tech-ua/ua-pro-kvantovi-kompyuteri/ (дата звернення: 20.04.2023)

3. openarchive.nure.ua - URL: https://openarchive.nure.ua/server/api/core/bitstreams/b36c9482-4963-49ad-91d4-48dd907ad7c6/content (дата звернення: 05.01.2023)

4. pq-crystals - URL: https://pq-crystals.org/kyber/ (дата звернення: 30.03.2023)

5. Medium - URL: https://medium.com/nuances-of-programming/%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC-%D0%B3%D1%80%D0%BE%D0%B2%D0%B5%D1%80%D0%B0-%D0%BA%D0%B2%D0%B0%D0%BD%D1%82%D0%BE%D0%B2%D1%8B%D0%B5-%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F-c97f5af3f34f (дата звернення: 20.04.2023).

6. Modding.fr - URL: https://www.modding.fr/un-ordinateur-quantique-vendu-pour-15-millions/(дата звернення: 20.04.2023).

7. cryptopedia.dev – URL: https://cryptopedia.dev/posts/kyber/(дата звернення: 20.04.2023).

8. IBM - URL: https://www.ibm.com/docs/en/zos/2.5.0?topic=cryptography-crystals-kyber-algorithm(дата звернення: 01.05.2023).

9. NIST – URL: https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms(дата звернення: 28.04.2023).

10. The Cloudflare Blog: URL: https://blog.cloudflare.com/post-quantum-key-encapsulation/(дата звернення: 01.05.2023).