НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Звіт з переддипломної практики

ПЕРЕВІРКА IND-CCA ТА IND-CPA СТІЙКОСТІ BABY KYBER

Виконала студентка
групи ФІ-13
Балацька Вікторія Віталіївна

Київ — 2025

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

KEM — механізм інкапсуляції ключів (key encaplusation mechanism)

IND-CCA — нерозрізнення за допомогою адаптивно вибраного шифрованого тексту (indistinguishability under adaptive chosen ciphertext attack)

CCA — атака за допомогою вибраного шифротексту (chosen ciphertext cttack)

IND-CPA — нерозрізнення за допомогою вибраного відкритого тексту (indistinguishability under chosen plaintext attack)

ML-KEM — механізм інкапсуляції ключів на основі решітки (module-lattice-based key-encapsulation mechanism)

MLWE — модульне навчання на помилках (module learning with errors)

PKE — схема шифрування з відкритим ключем (public-key encryption)

LWE — задача навчання на помилках (learning with errors)

FO — перетворення Фудзісакі-Окамото (Fujisaki-Okamoto transform)

ВСТУП

Сучасні криптографічні протоколи встановлення ключів стикаються з викликами, зумовленими розвитком квантових обчислень. Традиційні асиметричні схеми, що забезпечують конфіденційність переданих даних, можуть стати вразливими в умовах появи достатньо потужних квантових комп'ютерів, здатних розв'язувати складні криптографічні задачі. У зв'язку з цим активно розвиваються гібридні схеми і протоколи встановлення ключів, що поєднують класичні та постквантові механізми задля досягнення довготривалої безпеки.

Одна із таких схем представлена у статті Joppe Bos, Leo Ducas та інших «CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM» [1]. Вона пропонує гібридне поєднання класичних та постквантових компонентів для обміну ключами. Однак для практичного впровадження необхідно оцінити безпеку такої конструкції відносно стандартних моделей стійкості, зокрема IND-CCA та IND-CPA.

Метою даної переддипломної практики є реалізація спрощеної версії даної схеми Kyber та її формальна перевірка на відповідність вимогам IND-CCA та IND-CPA. Дослідження спрямоване на виявлення можливих вразливостей та підтвердження криптографічної стійкості отриманої конструкції.

Об'єктом дослідження є процеси оцінки криптографічної стійкості гібридних схем.

Предметом дослідження є методи аналізу стійкості до IND-CPA та IND-CCA атак.

Реалізація алгоритму та експериментальні атаки доступні у відкритому репозиторії за посиланням:

https://github.com/tOrryV/Practice_DL.git.

1 РЕАЛІЗАЦІЯ СПРОЩЕНОЇ ВЕРСІЇ ГІБРИДНОЇ КРИПТОГРАФІЧНОЇ СХЕМИ KYBER

У цьому розділі розглянуто реалізацію спрощеної версії гібридної криптографічної схеми Kyber [2]. Для цього спочатку буде проведено короткий огляд загальної схеми KEM Kyber, яка є постквантовою криптосистемою стійкість якої ґрунтується на складності розв'язання задачі навчання на помилках (LWE) над модульними решітками. Далі буде розглянуто Baby Kyber – спрощену версію схеми, яка дозволяє проаналізувати основні принципи роботи Kyber у спрощеному вигляді [3]. Теоретичне обґрунтування Baby Kyber допоможе зрозуміти його основні компоненти та алгоритми. Нарешті, буде представлена практична реалізація Baby Kyber, включаючи опис використаних криптографічних примітивів, алгоритми обміну ключами та результати тестування.

1.1 Огляд криптосистеми Kyber

Kyber – це постквантова криптографічна схема для обміну ключами, що ґрунтується на задачах решіткової криптографії. Вона була розроблена для забезпечення стійкості до атак квантових комп'ютерів і стала одним із фіналістів конкурсу NIST з постквантової криптографії.

У цьому підрозділі буде розглянуто загальну схему роботи Kyber, включаючи основні алгоритми генерації ключів, шифрування та розшифрування. Також буде проаналізовано властивості та рівень безпеки Kyber, зокрема її стійкість до криптографічних атак і ефективність у практичному застосуванні.

Загальна схема роботи Kyber

Kyber є пост-квантовим KEM, що використовує відкритий ключ. Ця схема є першою схемою KEM, який стандартизовано. Механізм інкапсуляції ключів Kyber використовує задачу модульного навчання на помилках (MLWE) [7] та складність її розв'язання. Тобто Kyber - це захищений IND-CCA механізм інкапсуляції ключів (KEM), стійкість якого ґрунтується на складності розв'язання задачі навчання на помилках (LWE) над модульними решітками [2]. Звичайно, існують аналогічні схеми, які ґрунтуються на неструктурованому MLWE, проте ця конструкція має значно більші переваги в ефективності [5].

Для побудови Kyber спершу вводиться IND-CPA-безпечна схема шифрування з відкритим ключем, яка використовується для шифрування повідомлення фіксованої довжини, після цього, використовуючи FO-перетворення, будується IND-CCA2-безпечний KEM [2].

Структура Kyber ґрунтується на модульній версії схеми шифрування Ring-LWE LPR [6, 7] із відкиданням бітів [8]. Цей KEM також посилений багатьма поліпшеними реалізаціями схем шифрування на основі решітки, таких як NewHope [9]. У схемах шифрування, що використовуються (NewHope та схемах Ring-LWE) операції мали вигляд $As + e$, де всі ці змінні є поліномами у кільці [10]. Проте, на відміну від них, Kyber використовує матрицю A над кільцем поліномів, а s, e - це вектори над тим самим кільцем, що і матриця [10]. Для Kyber A є квадратною матрицею розміру $k \times k$, а вектори s, e є k-вимірними векторами.

Особливістю Kyber є використання так званого теоретико-числового перетворення (NTT) [11]. NTT перетворює поліном $f \in R_q$ в представлення у вигляді вектора f лінійних поліномів [11]. Саме це перетворення дозволяє пришвидшити операцію множення поліномів.

Опис роботи KEM та алгоритм поданий нижче, ґрунтується на роботі [12]. За допомогою алгоритму KeyGen створюються відкритий та

особистий ключі. Спочатку, випадково створюється матриця A із кільця поліномів $R_{3329}^{k\times k, X^{256}+1}$ та вектори s, e, що використовуються для додавання шуму. У результаті обчислюється значення t, та повертаються необхідні ключі: $pk = (\mathbf{A}, t)$ - відкритий ключ, $sk = s$ - особистий ключ. За допомогою алгоритму Encrypt, шифрується повідомлення m, використовуючи відкритий ключ pk. Спочатку створюються вектори r, e_1, e_2, далі обчислюються значення u^T та v. Після виконання алгоритму отримується шифротекст (u, v). За допомогою алгоритму розшифрування відновлюється повідомлення m використовуючи отриманий шифротекст та особистий ключ sk. Спершу обчислюється значення u' де шифротекст переводиться назад у простір за модулем q та обчислюється значення v' де аналогічно обробляється друга частина шифротексту v. У результаті обчислення m' отримується повідомлення m.

Algorithm 1.1 Kyber Encryption Scheme

KeyGen:

Generate $\mathbf{A} \leftarrow \mathcal{R}_{3329}^{k\times k, X^{256}+1}$

Sample $(s, e) \leftarrow \psi_{\eta_1}^k \times \psi_{\eta_1}^k$

Compute $t := \mathbf{A}s + e$

Output $pk = (\mathbf{A}, t)$ and $sk = s$

Encrypt $(pk = (\mathbf{A}, t), m)$**:**

Sample $(r, e_1, e_2) \leftarrow \psi_{\eta_1}^k \times \psi_{\eta_1}^k \times \psi_{\eta_2}$

Compute $\mathbf{u}^T := \left[\mathbf{r}^T\mathbf{A} + e_1^T\right]_{q\to 2^{d_u}}$

Compute $v := \left[\mathbf{r}^T t + e_2 + \frac{q-1}{2}m\right]_{q\to 2^{d_v}}$

Output ciphertext $= (\mathbf{u}, v)$

Decrypt $(sk = s, \mathbf{ciphertext} = (\mathbf{u}, v))$**:**

Recover $\mathbf{u}' := [\mathbf{u}]_{2^{d_u}\to q}$

Recover $v' := [v]_{2^{d_v}\to q}$

Compute $m' := \left[v' - \mathbf{u}'^T s\right]_{q\to 2}$

Output m'

Властивості Kyber

1. Параметри та рівні безпеки Kyber:

Kyber має три основні варіанти параметрів, кожен з яких відповідає певному рівню криптографічної стійкості [3]. Рівень безпеки визначається відповідно до стійкості класичних криптографічних алгоритмів, таких як AES та RSA, до атак квантових комп'ютерів.

1) Kyber512: орієнтований на рівень безпеки, еквівалентний AES-128.

2) Kyber768: відповідає рівню безпеки AES-192.

3) Kyber1024: забезпечує рівень безпеки, подібний до AES-256.

Kyber512

Kyber512 є найменшою конфігурацією у сімействі Kyber і забезпечує рівень безпеки, еквівалентний AES-128. Він був розроблений для ефективного постквантового обміну ключами з використанням решіткової криптографії [2].

Основні параметри [2]:

– Рівень безпеки: аналог AES-128

– Розмір відкритого ключа: 800 байтів

– Розмір особистого ключа: 1632 байти

– Розмір шифротексту: 768 байтів

Kyber768

Kyber768 – це середній рівень безпеки в сімействі CRYSTALS-Kyber, призначений для відповідності AES-192 у класичній криптографії. Він має покращені параметри порівняно з Kyber512, що підвищує його стійкість, але водночас збільшує розмір ключів і шифротекстів [2].

Основні параметри [2]:

– Рівень безпеки: аналог AES-192

– Розмір відкритого ключа: 1184 байтів

– Розмір особистого ключа: 2400 байти

– Розмір шифротексту: 1088 байтів

Kyber1024

Kyber1024 – це найвищий рівень безпеки в сімействі CRYSTALS-Kyber, розроблений для відповідності AES-256 у класичній криптографії. Він має ще більші параметри, що забезпечує максимальну стійкість серед усіх варіантів Kyber, але водночас збільшує розмір ключів та шифротекстів. [2].

Основні параметри [2]:

– Рівень безпеки: аналог AES-256

– Розмір відкритого ключа: 1568 байтів

– Розмір особистого ключа: 3168 байти

– Розмір шифротексту: 1568 байтів

Безпека Kyber

Алгоритми сімейства Kyber (512, 768, 1024) використовують схему гібридного шифрування на основі Module-LWE (MLWE). Всі три варіанти забезпечують IND-CPA стійкість на базовому рівні, а IND-CCA стійкість завдяки трансформації Fujisaki-Okamoto (FO) [14].

1. IND-CPA стійкість [15]

а) IND-CPA гарантується випадковими шумовими компонентами в обчисленнях.

б) Використання MLWE-складності унеможливлює ефективний аналіз шифротекстів навіть для квантових алгоритмів.

в) Всі варіанти Kyber забезпечують IND-CPA стійкість без додаткових трансформацій.

2. IND-CCA стійкість [1]

а) IND-CCA досягається за допомогою перетворення Фудзісакі Окамотто.

б) Це перетворення включає повторну інкапсуляцію та перевірку правильності шифротексту, що запобігає обманним запитам на дешифрування.

в) Усі три версії Kyber використовують FO-перетворення для

забезпечення IND-CCA стійкості.

1.2 Теоретична реалізація Baby Kyber

У цьому підрозділі буде розглянуто спрощену версію криптосистеми Kyber, відому як Baby Kyber. Ця модель зберігає основні ідеї оригінальної схеми, але використовує зменшені параметри, що робить її зручнішою для аналізу та розуміння.

Спочатку буде описано принципи спрощення схеми Kyber, які дозволяють отримати Baby Kyber. Далі розглянемо основні математичні операції, що використовуються у цій криптосистемі, зокрема операції в кільцях та роботу з многочленами. Потім буде детально описано алгоритми генерації ключів, шифрування та розшифрування, які є основою роботи Baby Kyber.

Спрощення схеми Kyber

Для полегшення аналізу криптосистеми Kyber створюється її спрощена версія — Baby Kyber. Вона зберігає основні криптографічні принципи Kyber, але використовує зменшені параметри та спрощені математичні операції [3]. Спрощення полягає у:

1) Зменшення розміру параметрів: у стандартній версії Kyber використовується кільце поліномів $R_n^{k\times k, X^n+1}$, де $n = 512/768/1024$, q - велике просте число. У версії Baby Kyber використовують зменшені значення параметрів (наприклад n до 4 чи 8) та використовують менші значення модуля q (наприклад 97, замість 3329).

2) Використання малих поліномів в особистому ключі: секретний ключ складається з поліномів із малими коефіцієнтами, що полегшує їх генерацію та обробку.

3) Спрощена матриця A: у стандартному Kyber використовується

випадкова матриця A розміру $k \times k$. У Baby Kyber використовується фіксована одномірна матриця A, що значно спрощує генерацію публічного ключа.

4) Відсутність перетворення Фудзісакі Окамотто: У стандартному Kyber використовується FO-перетворення, яке підсилює схему, роблячи її стійкою до атаки з вибором шифротексту. У Baby Kyber відсутнє FO-перетворення, тому ця схема не має IND-CCA стійкості та ближча до базової форми шифрування на решітках.

5) Відсутність NTT перетворення (Number Theoretic Transform): У стандартному Kyber для ефективного множення поліномів використовується NTT-перетворення, яке суттєво прискорює обчислення у великому полі. У Baby Kyber NTT не використовується, а всі обчислення виконуються напряму в кільці, що робить реалізацію простішою, але менш ефективною.

Основні математичні операції у Baby Kyber

У Baby Kyber використовуються основні математичні операції, які є спрощеною версією тих, що застосовуються у повноцінному Kyber. Основні операції включають:

1) Додавання поліномів: операція додавання в Baby Kyber здійснюється через покомпонентне додавання коефіцієнтів двох поліномів за модулем q, де q є малим простим числом. Після цього результат доцільно взяти за модулем $x_n + 1$, щоб зменшити результати на рівні поліномів.

$$p_1(x) + p_2(x) = \big(\sum_{i=0}^{n-1}(p_{1,i} + p_{2,i}) \mod q\big) \mod (x^n + 1)$$

2) Множення поліномів: множення поліномів також виконується за допомогою покомпонентного множення коефіцієнтів двох поліномів, з наступним зведенням результату за модулем q. Після цього результат

беремо за модулем $x_n + 1$, щоб зменшити степінь полінома.

$$p_1(x) \cdot p_2(x) = \sum_{i=0}^{n-1} \big(\sum_{j=0}^{n-1} (p_{1,i} \cdot p_{2,j}) \mod q \big) \mod (x^n + 1)$$

3) Редукція полінома за модулем x_n+1: редукція поліномів виконується за допомогою операції зменшення степеня полінома в кільці. Це означає, що при наявності степеня полінома, який перевищує $n - 1$, відбувається заміна вищих степенів $x^n, x^{n+1}, ...$ на відповідні вирази, які зберігають коректність у межах кільця.

$$p(x) \mod (x^n + 1) = \sum_{i=0}^{n-1} (p_i \mod q)$$

4) Генерація випадкових поліномів: У Baby Kyber випадкові поліноми генеруються з коефіцієнтами, які мають певний набір значень. Ці поліноми використовуються для генерації секретних і публічних ключів.

Алгоритми генерування ключів, шифрування та розшифрування

1) Генерування ключів

Відкритий ключ: відкритий ключ складається із двох компонентів: випадкова матриця A та випадковий вектор t.

а) Матриця A: це матриця розміру $k \times k$, де кожен елемент є випадковим поліномом з кільця $\mathbb{Z}_q[X]/(X^n + 1)$. Кожен коефіцієнт полінома вибирається випадково з проміжку $[0, q - 1]$

б) Вектор t: це вектор розміру k, де кожен компонент обчислюється як $A \cdot s + e$, де s - особистий ключ, e - вектор шуму. Вектори s та e також складаються з поліномів, згенерованих через центральний біноміальний розподіл (CBD).

Особистий ключ: особистий ключ складається з випадкового вектора поліномів s, що з генерується малими значеннями та шуму e, що додається

для забезпечення стійкості до атак.

2) Шифрування

Шифрування виконується за алгоритмом з використанням відкритого ключа (A, t):

а) Вибір випадковий поліномів: генеруються два набори випадкових поліномів r та e_1 для шуму, вибраних за допомогою центрального біноміального розподілу (CBD).Також генерується випадковий поліно e_2 для додаткового шуму в шифротексті.

б) Компонентами шифротексту є вектори u та v. Для створення кожного елемента вектора u, проводиться множення кожного стовпця транспонованої матриці A^T на відповідний поліном вектора r, після чого додається шум e_1 Компоненту v обчислюють за допомогою додавання поліномів e_2 та результату множення кожного елемента вектора t на відповідний елемент вектора r. Після цього додається масштабоване повідомлення m, яке є поліномом, що представляє бітове повідомлення. Кожен біт повідомлення множиться на $\lceil q/2 \rceil$.

в) В результаті шифротекст складається із двох компонент (u, v), які разом містять зашифроване повідомлення.

2) Розшифрування

Розшифрування за допомогою секретного ключа s відбувається наступним чином:

а) Обчислення скалярного добутку $s^T u$. Тут кожен елемент вектора s множиться на відповідний елемент вектора u, а потім ці добутки додаються.

б) Для отримання розшифрованого результату спочатку розшифроване значення v зменшується на $s^T u$. Для отримання бітів повідомлення, кожен коефіцієнт результату розшифровки зменшується за модулем q, а потім порівнюється з межами для визначення, чи є це 1 або 0 в оригінальному повідомленні.

в) Після перевірки кожного коефіцієнта результату розшифрування формується вектор бітів m, який представляє оригінальне повідомлення.

1.3 Практична реалізація Baby Kyber

У цьому розділі буде розглянуто практичну реалізацію Baby Kyber. Спочатку буде обґрунтовано вибір середовища розробки та необхідних інструментів, які забезпечують зручність роботи з многочленами та криптографічними операціями. Далі буде подано опис основних функцій реалізації, включаючи операції над поліномами, генерацію ключів, шифрування та розшифрування. На завершальному етапі буде проведено перевірку коректності реалізованих алгоритмів та тестування для оцінки їхньої відповідності теоретичним очікуванням.

Вибір середовища розробки та інструментів

Для реалізації алгоритму Baby Kyber було обрано мову програмування Python. Основними причинами такого вибору є:

1. Зручність написання та читабельність коду: Python має лаконічний синтаксис, який полегшує розуміння та підтримку коду. Це особливо важливо в криптографії, де складність алгоритмів висока, і зрозумілий код спрощує налагодження та аналіз безпеки.

2. Вбудовані бібліотеки та підтримка великих чисел: Python підтримує довільну точність цілих чисел, що є критично важливим для криптографічних алгоритмів, які працюють з великими просторами модулів. Також стандартна бібліотека **math** забезпечує всі необхідні операції для роботи з модулярною арифметикою.

3. Можливість швидкої розробки та прототипування: у криптографії часто необхідно тестувати різні підходи та параметри. Python дозволяє швидко реалізовувати нові ідеї та перевіряти їхню ефективність без необхідності компіляції та складного керування пам'яттю.

4. Наявність криптографічних бібліотек: хоча Baby Kyber реалізується вручну, Python має низку бібліотек для криптографії (PyCryptodome,

cryptography, hashlib), які можуть допомогти у створенні допоміжних інструментів, тестуванні та порівнянні з іншими алгоритмами.

5. Гнучкість та переносність: код на Python можна запускати на різних операційних системах без значних змін, що спрощує тестування та використання алгоритму в різних середовищах.

Для реалізації обрано середовище розробки PyCharm, оскільки воно надає низку переваг, які спрощують процес розробки, тестування та налагодження коду. Воно пропонує:

1. Інтелектуальне автодоповнення коду: PyCharm має потужний механізм автодоповнення коду, який допомагає швидше писати та редагувати код, зменшуючи кількість помилок і підвищуючи продуктивність.

2. Інтегрований налагоджувач (debugger): криптографічні алгоритми можуть бути складними у відлагодженні через багатоетапні обчислення. Вбудований debugger дозволяє покроково аналізувати виконання коду, перевіряти значення змінних та знаходити помилки без потреби у вставленні додаткових ‘print()‘.

3. Зручна інтеграція з Git: у процесі розробки важливо використовувати систему контролю версій Git. PyCharm надає графічний інтерфейс для роботи з Git, що спрощує управління комітами, створення гілок та аналіз історії змін.

4. Інтегрована консоль та термінал: PyCharm містить вбудований термінал, що дозволяє запускати скрипти, тестувати код і виконувати команди без потреби перемикатися між вікнами.

5. Кросплатформеність: PyCharm доступний для Windows, macOS і Linux, що робить його зручним для роботи на будь-якій операційній системі.

Для реалізації алгоритму використовуються наступні бібліотеки Python:

1. math: ця стандартна бібліотека Python надає математичні функції, які використовуються для операцій над числами, округлення, піднесення до степеня та обчислення модулів. У криптографічних алгоритмах часто

потрібні обчислення залишків, роботи з великими числами та логарифми.

2. secrets: бібліотека secrets використовується для генерації криптографічно стійких випадкових чисел. Це критично важливо для криптографічних алгоритмів, оскільки використання звичайної random може зробити систему вразливою для атак.

Опис основних функцій реалізації

Клас *RingPolynomOperations*

Цей клас *RingPolynomOperations* реалізує операції з поліномами в кільці $\mathbb{Z}_q[X]/(X^n + 1)$, що є важливим для криптосистеми Baby Kyber. Він включає методи для додавання та множення поліномів, а також для їх редукції.

1) Метод *init* ініціалізує об'єкт класу з параметрами:

а) *mod*: модуль для виконання операцій.

б) n: ступінь полінома.

2) Метод *add* додає два поліноми покомпонентно за модулем *mod*, після чого виконує редукцію за $X^n + 1$.

3) Метод *multiply* обчислює добуток двох поліномів з подальшою редукцією за $X^n + 1$ та модулем *mod*.

4) Метод *module* виконує редукцію полінома за $X^n + 1$, коригуючи коефіцієнти та приводячи їх до значень за модулем *mod*.

Цей клас забезпечує базові операції для маніпулювання поліномами у криптосистемах, де поліноми мають бути оброблені за модулем конкретного числа, а також з обмеженням на їх ступінь.

Клас *BabyKyber*

Клас *BabyKyber* реалізує криптосистему Baby Kyber, що є спрощеною версією алгоритму Kyber. Цей клас включає методи для генерації ключів, шифрування та дешифрування повідомлень. Основною особливістю є використання поліномів і центрованого біноміального розподілу (CBD) для генерації шуму.

1) Метод *init* ініціалізує об'єкт класу з параметрами:

а) n: ступінь полінома.

б) eta: параметр для контролю рівня шуму, що генерується.

в) k: розмірність матриці.

г) q: модуль для виконання операцій.

д) $ring$: об'єкт класу $RingPolynomOperations$, який надає функції для операцій над поліномами.

2) Метод $sample\_uniform\_polynomial$ генерує випадковий поліном з коефіцієнтами, вибраними рівномірно з діапазону $[0, q-1]$. Ступінь полінома не перевищує значення $degree$.

3) Метод $sample\_cbd\_polynomial$ генерує поліном, коефіцієнти якого вибрані за допомогою центрованого біноміального розподілу (CBD). Кожен коефіцієнт обчислюється як різниця двох випадкових значень з рівномірного розподілу на $[0, eta]$. Поліном редукується за модулем $X^n + 1$.

4) Метод key_gen генерує відкритий і особистий ключі для криптосистеми Baby Kyber.

5) Метод *encrypt* шифрує повідомлення, використовуючи відкритий ключ (A, t).

6) Метод *decrypt* дешифрує шифротекст (u, v) за допомогою особистого ключа s.

Main файл

Main файл є основним компонентом для реалізації криптосистеми Baby Kyber.

1) Функція $text\_to\_bits$ перетворює текст у список бітів.

2) Функція $bits\_to\_text$ відновлює текст із списку бітів.

3) Функція $encrypt\_text$ виконує шифрування. Вона ділить вхідні біти на блоки довжини n. Далі додає нулі до блоку, якщо його довжина менша за n і використовує алгоритм BabyKyber для шифрування кожного блоку.

4) Функція $decrypt\_text$ виконує розшифрування. Вона використовує алгоритм BabyKyber для розшифрування кожного блоку. Після чого об'єднує всі розшифровані біти у список та перетворює його у текст.

5) Функція *main* організовує виконання програми:

– Визначає параметри криптосистеми (q, n, k, η).

– Ініціалізує кільцеві операції та алгоритм BabyKyber.

– Генерує ключі: відкриту матрицю A, відкритий вектор t та секретний вектор s.

– Перетворює повідомлення у біти.

– Виконує шифрування та виводить шифротекст.

– Виконує розшифрування та виводить отримане повідомлення.

– Перевіряє правильність розшифрування.

Перевірка коректності та тестування

Результати виконання

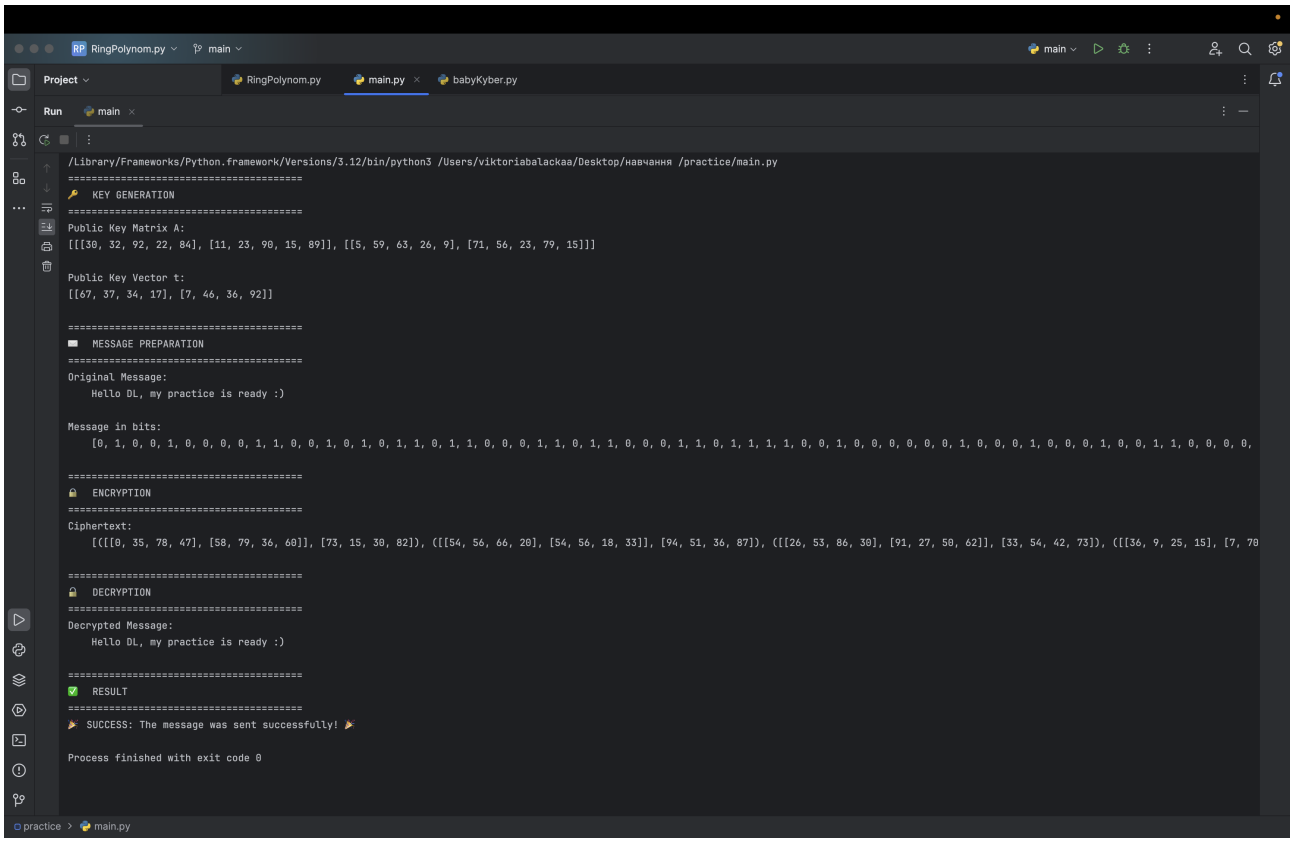

Рисунок 1.1 – Результат 1

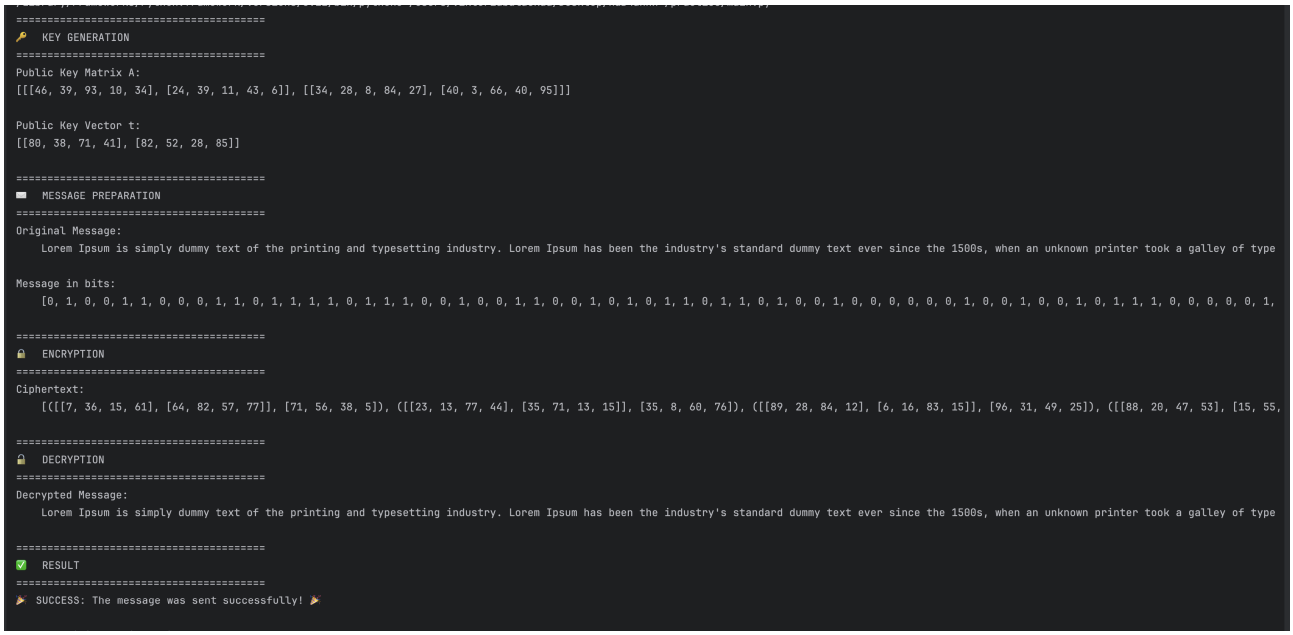

Рисунок 1.2 – Результат 2

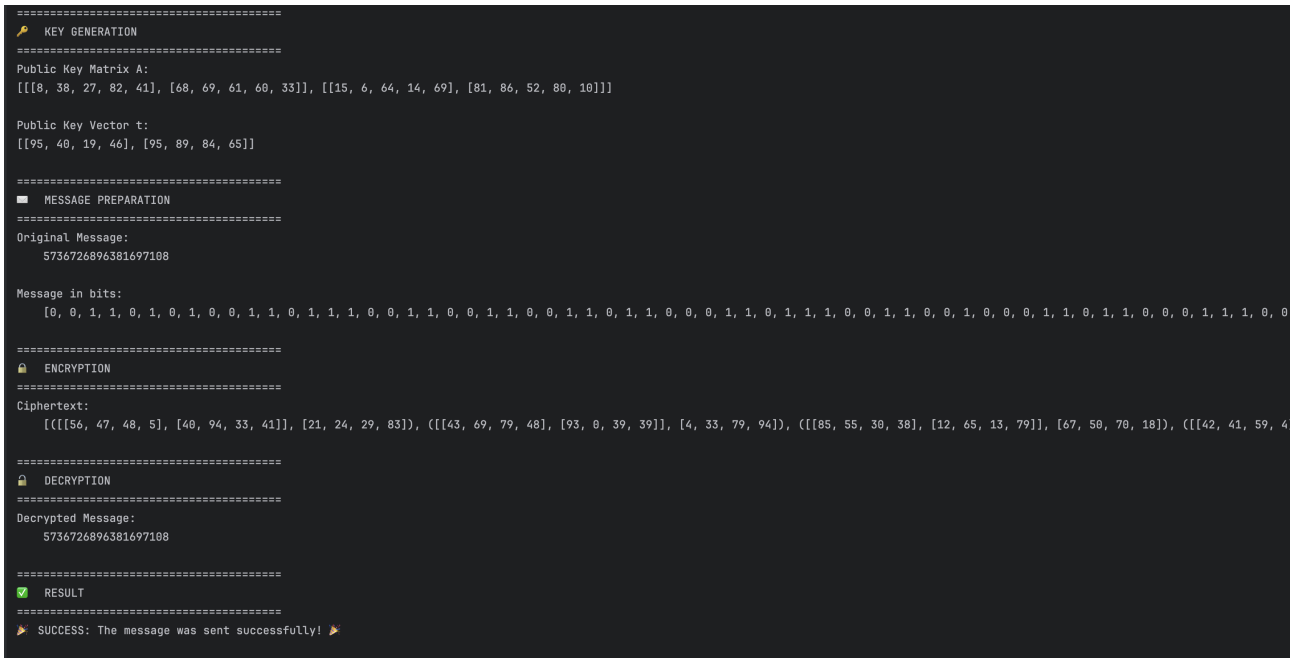

Рисунок 1.3 – Результат 3

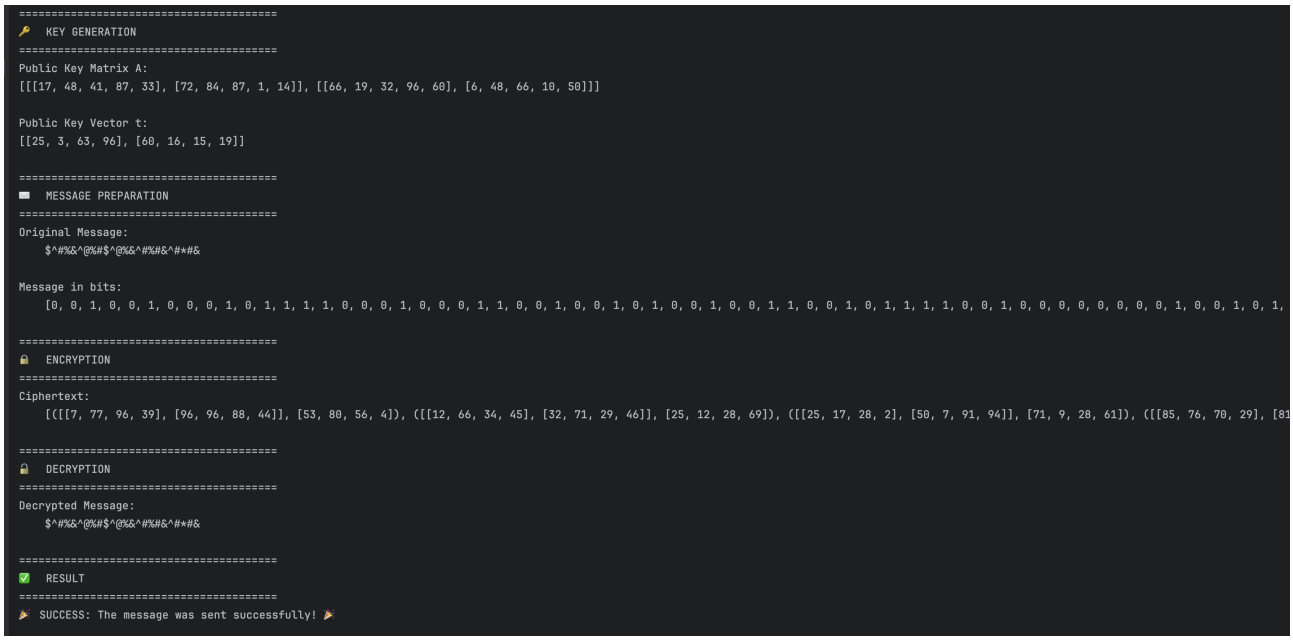

Рисунок 1.4 – Результат 4

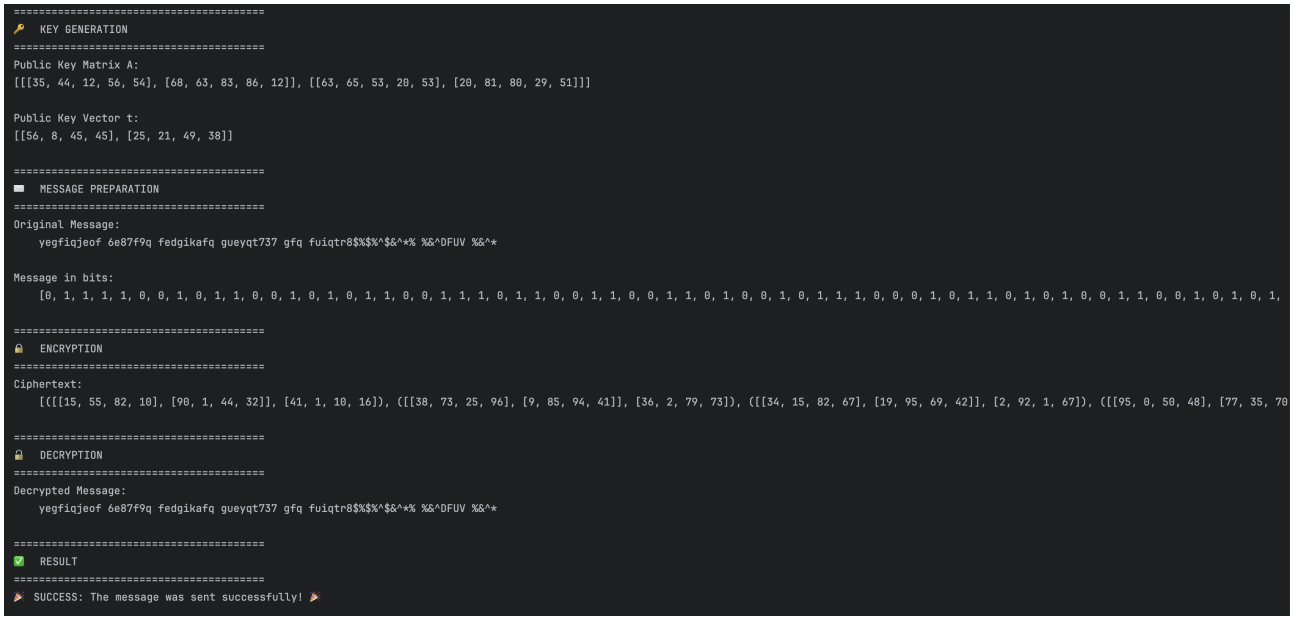

Рисунок 1.5 – Результат 5

2 ОЦІНКА СТІЙКОСТІ РЕАЛІЗОВАНОЇ КОНСТРУКЦІЇ ДО ВИМОГ IND-CPA ТА IND-CCA

У цьому розділі буде здійснено аналіз відповідності реалізованої криптографічної конструкції вимогам двох основних моделей безпеки: IND-CPA та IND-CCA. Перш за все, розглянемо теоретичні основи цих моделей безпеки, щоб зрозуміти, як вони визначають рівень стійкості криптографічних схем. Далі буде проведена оцінка стійкості конструкції до цих вимог, а також порівняння стійкості конструкції до кожної з моделей. Завершальний етап включатиме практичну реалізацію атак, які дозволять перевірити, наскільки добре конструкція Baby Kyber витримує загрози згідно з обраними моделями безпеки.

2.1 Вступ до моделей безпеки IND-CPA та IND-CCA

IND-CPA стійкість

Безпека є найважливішою складовою будь-якої схеми і її забезпечення є головною задачею дослідника. Звичайне поняття безпеки дуже схоже на концепцію безпеки шифрування з відкритим ключем. Основною ідеєю є те, що зловмисник не повинен мати змогу визначити, чи міститься конкретний ключ в зашифрованому тексті. Інакше кажучи, зловмисник не повинен відрізнити випадковий ключ від ключа, який був зашифрований [16].

Механізм інкапсуляції ключів вважається стійким до атаки обраним відкритим текстом (IND-CPA), якщо зловмисник $\mathcal{A}$ не може відрізнити спільний ключ від випадкового спільного ключа з імовірністю більше, ніж $\frac{1}{2}$. Зловмисник $\mathcal{A}$ отримує всю відкриту інформацію, яка створена KEM. Це визначення параметризується розподілом S, який описує безпечне джерело спільних секретів. Варто зазначити, що KEM не має відкритого тексту, проте це поняття використовується у визначенні через аналогію з

відповідними поняттями у шифруванні [17].

Для доведення IND-CPA стійкості будують так звані IND-CPA ігри:

Game $G0^{ind-cpa}_{KEM}(\mathcal{A})$
$(\cdot, P) \leftarrow_{\$} \mathbf{KGen}()$
$(k, R) \leftarrow_{\$} \mathbf{Enc}(P)$
if $\perp (k, R)$ **return** 0
return $\mathcal{A}(P, R, k)$

Game $G1^{ind-cpa}_{KEM,S}(\mathcal{A})$
$(\cdot, P) \leftarrow_{\$} \mathbf{KGen}()$
$(k, R) \leftarrow_{\$} \mathbf{Enc}(P), k' \leftarrow_{\$} \mathbf{S}()$
if $\perp (k, R)$ **return** 0
return $\mathcal{A}(P, R, k')$

Рисунок 2.1 – IND-CPA ігри

Означення 2.1. [17] (IND-CPA перевага) Нехай KEM — механізм інкапсуляції ключів, $\mathcal{A}$ - алгоритм. Тоді, перевага $\mathcal{A}$ над IND-CPA KEM джерела S дорівнює:

$$Adv^{ind-cpa}_{KEM,S}(\mathcal{A}) = |Pr[G0^{ind-cpa}_{KEM}(\mathcal{A}) = 1] - Pr[G1^{ind-cpa}_{KEM,S}(\mathcal{A}) = 1]|$$

IND-CCA стійкість

Існує і більш сильний варіант захисту. Тут зловмисник може робити запити на розшифрування вибраних ним зашифрованих текстів. Такі атаки називаються атаками на основі обраних шифротекстів (CCA). Вони моделюють ситуацію, у яких можливе знання ключів, що містяться в певних зашифрованих текстах. Хоча на практиці такі витоки зазвичай обмежені, більш загальна модель безпеки враховує ширший спектр загроз, що дозволяє побудувати схеми, які задовольняють загальну модель [16].

Насправді, концепція цієї атаки ідентична із IND-CPA за винятком того, що зловмисник також має змогу здійснити атаку за допомогою обраного шифротексту, адже має доступ до оракула декапсуляції. Він може викликати такий оракул для будь-яких шифротекстів, окрім тих які безпосередньо беруть участь у грі [17].

Для доведення IND-CCA стійкості будують так звані IND-CCA ігри [17]:

Game $G0^{ind-cca}_{KEM}(\mathcal{A})$
$(sk, pk) \leftarrow_{\$}$ **KGen**()
$(k, c) \leftarrow_{\$}$ **Enc**(pk)
if $\perp (k)$ **return** 0
return $\mathcal{A}^{\mathcal{D}^{sk,c}}(pk, c, k)$

Game $G1^{ind-cca}_{KEM,S}(\mathcal{A})$
$(sk, pk) \leftarrow_{\$}$ **KGen**()
$(k, c) \leftarrow_{\$}$ **Enc**$(pk), k' \leftarrow_{\$}$ **S**()
if $\perp (k)$ **return** 0
return $\mathcal{A}^{\mathcal{D}^{sk,c}}(pk, c, k')$

Рисунок 2.2 – IND-CCA games

Означення 2.2. [17] (IND-CCA перевага) Нехай KEM — механізм інкапсуляції ключів, $\mathcal{A}$ - алгоритм зловмисника. Тоді, перевага $\mathcal{A}$ над IND-CCA KEM джерела S дорівнює:

$$Adv^{ind-cca}_{KEM,S}(\mathcal{A}) = |Pr[G0^{ind-cca}_{KEM}(\mathcal{A}) = 1] - Pr[G1^{ind-cca}_{KEM,S}(\mathcal{A}) = 1]|$$

2.2 Оцінка стійкості Baby Kyber до вимог IND-CPA та IND - CCA

Оцінка стійкості Baby Kyber до IND-CPA

Стійкість криптографічної схеми до атаки IND-CPA є одним із важливих критеріїв її безпеки. Для оцінки цієї стійкості необхідно проаналізувати здатність зловмисника відрізнити зашифровані версії двох різних повідомлень, згенерованих за допомогою відкритого ключа, без доступу до особистого ключа. У контексті Baby Kyber, ця стійкість визначається здатністю схемі забезпечити безпечне шифрування, яке не дозволяє зловмиснику здобути корисну інформацію про повідомлення, навіть якщо він має доступ до численних зашифрованих повідомлень.

Для доведення стійкості побудуємо IND-CPA гру. Гра IND-CPA моделює атаку зловмисника, який має доступ до відкритого ключа та оракула шифрування. Мета — перевірити, чи зможе він відрізнити, яке з двох повідомлень було зашифроване.

1. Ініціалізація

Оракул генерує ключі для Baby Kyber:

– Випадкова матриця: $A \in \mathbb{Z}_q[X]^{k\times k}/(X^n+1)$.

– Вектори $s, e \sim$ CBD — шум, згенерований через центральний біноміальний розподіл.

– Обчислення: $t = As + e$.

– Відкритий ключ: $pk = (A, t)$, особистий ключ: $sk = s$.

– Оракул передає зловмиснику pk.

2. Шифрування

Зловмисник $\mathcal{A}$ може багаторазово шифрувати повідомлення m:

1) Надсилає m до оракула.

2) Оракул:

– Вибирає $r, e_1, e_2 \sim$ CBD.

– Обчислює шифротекст:

$$u = A^T r + e_1, \quad v = t^T r + e_2 + \left\lfloor \frac{q}{2} \right\rfloor m.$$

– Повертає $c = (u, v)$.

3. Вибір повідомлень (Challenge Phase)

– Супротивник обирає два повідомлення m_0, m_1 однакової довжини.

– Надсилає (m_0, m_1) оракулу.

– Оракул вибирає випадковий біт $b \in \{0, 1\}$ та нові випадкові r^*, e_1^*, e_2^*.

– Обчислює:

$$u^* = A^T r^* + e_1^*, \quad v^* = t^T r^* + e_2^* + \left\lfloor \frac{q}{2} \right\rfloor m_b.$$

– Повертає $c^* = (u^*, v^*)$ супротивнику.

4. Вгадування

– Супротивник аналізує c^* та видає здогад $b' \in \{0, 1\}$.

– Якщо $b' = b$, то супротивник виграв гру.

Game $G_0^{\text{ind-cpa}}(\mathcal{A})$
$(s, e) \leftarrow_{\$} \text{CBD}$
$A \leftarrow_{\$} \mathbb{Z}_q[X]^{k \times k}/(X^n + 1)$
$t = As + e$
$pk = (A, t), \quad sk = s$
$\mathcal{A}$ receives pk and access to $\mathsf{Enc}(\cdot)$
$(m_0, m_1) \leftarrow \mathcal{A}$
$b \leftarrow_{\$} \{0, 1\}$
$r, e_1, e_2 \leftarrow_{\$} \text{CBD}$
$u = A^T r + e_1$
$v = t^T r + e_2 + \left\lfloor \frac{q}{2} \right\rfloor m_b$
$c^* = (u, v)$
$b' \leftarrow \mathcal{A}(c^*)$
return $[b' = b]$

Рисунок 2.3 – IND-CPA гра для Baby Kyber

Щоб з'ясувати, чи може зловмисник визначити біт b, розглянемо, наскільки компоненти шифротексту u та v приховують інформацію про повідомлення m_b.

Розподіл u

Шифротекст u визначається як:

$$u = A^T r + e_1$$

– A — публічна випадкова матриця поліномів.

– $r \sim \text{CBD}$ — випадковий вектор поліномів, незалежний від A.

– $A^T r$ — виглядає як випадковий вектор.

– Додавання шуму e_1 (також з розподілу CBD) ще більше ускладнює структуру u.

u виглядає як повністю випадковий вектор, незалежний від m_b.

Розподіл v

Компонент v задається як:

$$v = t^T r + e_2 + \left\lfloor \frac{q}{2} \right\rfloor m_b$$

Підставимо $t = As + e$:

$$\begin{aligned} v &= (As + e)^T r + e_2 + \left\lfloor \frac{q}{2} \right\rfloor m_b \\ &= s^T (A^T r) + e^T r + e_2 + \left\lfloor \frac{q}{2} \right\rfloor m_b \end{aligned}$$

– s — секретний вектор, недоступний для супротивника.

– $A^T r$ — випадковий вектор, незалежний від s.

– Добуток $s^T (A^T r)$ має вигляд випадкового полінома.

– Шум $e^T r + e_2$ ще більше маскує значення.

v також виглядає як випадковий елемент, і m_b у ньому не видимий через шум.

IND-CPA-стійкість означає, що зловмисник не може відрізнити, яке з двох повідомлень m_0, m_1 було зашифроване, навіть маючи доступ до:

– Публічного ключа $pk = (A, t)$.

– Оракула шифрування: $\mathsf{Encrypt}(pk, \cdot)$.

– Шифротексту $c^* = (u, v)$, що містить зашифроване m_b.

Схема є IND-CPA стійкою тому що:

– Рандомізація через r гарантує, що одне й те саме повідомлення шифрується кожного разу інакше.

– Шумові доданки e_1, e_2 та e приховують структуру повідомлення.

– Без знання s неможливо відновити m_b з v.

– Побудовано на задачі Learning With Errors (LWE), яка вважається складною навіть для квантових обчислень.

Інформація про m_b не витікає із шифротексту, тому схема є стійкою до IND-CPA атак.

Оцінка стійкості Baby Kyber до IND-CCA

Стійкість криптографічної схеми до атаки IND-CCA є більш суворим критерієм безпеки, ніж стійкість до IND-CPA. У цьому випадку зловмисник може не лише отримати зашифровані повідомлення, а й зробити запити на

дешифрування обраних шифротекстів, що дає йому додаткову інформацію, яку можна використати для атаки на схему. Оцінка стійкості Baby Kyber до IND-CCA ґрунтується на здатності схеми зберігати конфіденційність повідомлення навіть при наявності такого доступу до шифротекстів.

Для доведення стійкості побудуємо IND-CCA гру. Гра IND-CCA моделює атаку зловмисника, який має доступ до відкритого ключа, оракула шифрування та оракула дешифрування. Мета — перевірити, чи зможе він відрізнити, яке з двох повідомлень було зашифроване.

1. Ініціалізація

Оракул генерує ключі для Baby Kyber:

– Випадкова матриця:

$$A \in \mathbb{Z}_q[X]^{k\times k}/(X^n+1)$$

– Вектори $s, e \sim \text{CBD}$ — шум, згенерований через центрований біноміальний розподіл.

– Обчислення:

$$t = As + e$$

– Відкритий ключ: $pk = (A, t)$, особистий ключ: $sk = s$.

– Оракул передає зловмиснику pk.

– Також зловмиснику надається доступ до оракула дешифрування: $\mathcal{D}(c) = \mathsf{Dec}_{sk}(c)$, з обмеженням, що не можна подавати $c = c^*$ у фазі виклику.

2. Шифрування

Зловмисник $\mathcal{A}$ може багаторазово шифрувати повідомлення m:

1) Надсилає m до оракула.

2) Оракул:

– Вибирає $r, e_1, e_2 \sim \text{CBD}$.

– Обчислює шифротекст:

$$u = A^T r + e_1, \quad v = t^T r + e_2 + \left\lfloor \frac{q}{2} \right\rfloor m$$

– Повертає $c = (u, v)$.

– Зловмисник може також надсилати запити на дешифрування довільних $c \neq c^*$, використовуючи оракул $\mathcal{D}$.

3. Вибір повідомлень (Challenge Phase)

– Зловмисник обирає два повідомлення m_0, m_1 однакової довжини.

– Надсилає (m_0, m_1) оракулу.

– Оракул вибирає випадковий біт $b \in \{0, 1\}$ та нові випадкові значення r^*, e_1^*, e_2^*.

– Обчислює:

$$u^* = A^T r^* + e_1^*, \quad v^* = t^T r^* + e_2^* + \left\lfloor \frac{q}{2} \right\rfloor m_b$$

– Повертає $c^* = (u^*, v^*)$ зловмиснику.

– Забороняє надсилати c^* до оракула дешифрування.

4. Вгадування

– Зловмисник аналізує c^*, продовжує робити дозволені запити до оракула дешифрування $\mathcal{D}(c \neq c^*)$.

– Видає здогад $b' \in \{0, 1\}$.

– Якщо $b' = b$, то зловмисник виграв гру.

Game $G_0^{\text{ind-cca}}(\mathcal{A})$	
$(s, e) \leftarrow_{\$}$ CBD	$A \leftarrow_{\$} \mathbb{Z}_q[X]^{k \times k}/(X^n + 1)$
$t = As + e$	$pk = (A, t), \quad sk = s$
$\mathcal{A}$ *receives* pk*, access to* $\mathsf{Enc}(\cdot)$ *and* $\mathsf{Dec}(\cdot \neq c^*)$	
$(m_0, m_1) \leftarrow \mathcal{A}, \quad b \leftarrow_{\$} \{0, 1\}$	
$r^*, e_1^*, e_2^* \leftarrow_{\$}$ CBD	$u^* = A^T r^* + e_1^*$
$v^* = t^T r^* + e_2^* + \lfloor \frac{q}{2} \rfloor m_b$	$c^* = (u^*, v^*)$
$\mathcal{A}$ *may query* $\mathsf{Dec}(c)$ *for* $c \neq c^*$	
$b' \leftarrow \mathcal{A}(c^*)$	
return $[b' = b]$	

Рисунок 2.4 – IND-CCA гра для Baby Kyber

Щоб з'ясувати, чи може зловмисник визначити біт b, розглянемо,

наскільки компоненти шифротексту u та v приховують інформацію про повідомлення m_b, якщо зловмисник має доступ до оракула дешифрування.

Модель атаки

У моделі IND-CCA зловмисник має доступ до:

– Публічного ключа $pk = (A, t)$

– Оракула шифрування: $\mathsf{Enc}(pk, \cdot)$

– Оракула дешифрування: $\mathsf{Dec}(sk, \cdot)$, з обмеженням $c \neq c^*$

Задача зловмисника — вгадати біт b за шифротекстом $c^* = (u^*, v^*)$, який є шифруванням одного з повідомлень m_0, m_1.

Вразливість шифротексту

Challenge-шифротекст має вигляд:

$$c^* = (u^*, v^*) = (A^T r^* + e_1^*, \quad t^T r^* + e_2^* + \left\lfloor \frac{q}{2} \right\rfloor m_b)$$

Супротивник може сконструювати змінений шифротекст:

$$c' = (u^*, v^* + \delta)$$

і передати його на дешифрування. Якщо $m_b = 1$, v^* має зсув $\left\lfloor \frac{q}{2} \right\rfloor$, якщо $m_b = 0$ — ні.

Проаналізувавши результат розшифрування c', зловмисник може зробити висновок про значення m_b.

Відсутність цілісності

Схема Baby Kyber не містить механізму перевірки автентичності:

– Немає хеш-функцій або автентифікаторів у шифротексті

– Немає симетричної перевірки чи зв'язування u і v

– Шум e_1, e_2 не гарантує безпеку від модифікації

Це дозволяє зловмиснику модифікувати v^* та отримувати детерміновану зміну результату дешифрування.

Отже, зловмисник може розшифрувати змінений шифротекст c' на основі c^*. А результат дешифрування допомагає вивести біт b. Саме тому схема не є IND-CCA стійкою.

Порівняння стійкості до IND-CPA та IND-CCA

Таблиця 2.1 – Порівняння стійкості Baby Kyber до IND-CPA та IND-CCA

Критерій	**IND-CPA**	**IND-CCA**
Модель зловмисника	Має доступ до шифрування (Enc), може надсилати будь-які повідомлення	Має доступ до шифрування (Enc) і дешифрування (Dec), крім c^*
Доступ до дешифрування	Відсутній	Дозволено, але $c \neq c^*$
Тип атаки	Вибір відкритого тексту (Chosen Plaintext)	Вибір шифротексту (Chosen Ciphertext)
Стійкість Baby Kyber	Стійкий	Не стійкий
Причини стійкості / нестійкості	Випадковість r, шум e_1, e_2 та структура v приховують m_b	Можна модифікувати v^* і проаналізувати відповідь дешифрування
Приховування m_b	Ефективне через шум і LWE	Частково порушується через можливість змінювати v
Перевірка цілісності шифротексту	Відсутня, але не критично	Відсутня, що робить можливими атаки
Основна уразливість	—	Challenge можна змінити, зловмисник аналізує результат
Захист від модифікації c^*	—	Відсутній
Рівень безпеки	Семантична безпека при шифруванні	Семантична безпека + автентичність

Проведене порівняння показує, що Baby Kyber є надійною схемою шифрування з відкритим ключем у моделі IND-CPA, але не забезпечує захисту від IND-CCA атак.

У моделі IND-CPA зловмисник обмежений лише доступом до шифрування. У такому середовищі Baby Kyber демонструє високу стійкість завдяки використанню:

– випадкових векторів при шифруванні;

– шумових членів, що маскують повідомлення;

– математичної складності задачі Learning With Errors (LWE).

Однак у моделі IND-CCA, де зловмисник також має доступ до оракула дешифрування, виявляється серйозна вразливість. Через відсутність перевірки цілісності шифротексту, зловмисник може модифікувати частину challenge-шифротексту (наприклад, компонент v^*) і використовувати оракул дешифрування для витоку інформації про зашифроване повідомлення. Це робить Baby Kyber нестійким до IND-CCA атак у базовій формі.

2.3 Практична реалізація атак на відповідність моделям IND-CPA та IND-CCA

У цьому підрозділі буде розглянуто практичну реалізацію атак IND-CPA та IND-CCA на спрощену версію Baby Kyber. Буде подано опис основних компонентів атак, включаючи побудову викликів з боку супротивника, моделювання доступу до оракула шифрування та аналіз виграшу. Завершальною частиною стане перевірка коректності реалізації та оцінка ймовірності успішної атаки відповідно до теоретичних моделей IND-CPA та IND-CCA, описаних вище.

Реалізація атаки для IND-CPA

З метою оцінки стійкості реалізованої спрощеної версії Baby Kyber до IND-CPA-атак, було проведено експеримент, в якому моделюється поведінка зловмисника з доступом до оракула шифрування.

Для реалізації атаки було створено адаптивну стратегію, за якою зловмисник:

1) Ініціалізує параметри криптосистеми: розмір кільця n, модуль q, параметри шуму η тощо, та генерує відкритий ключ (A, t) і особистий ключ s.

2) Обирає два повідомлення m_0 і m_1, які представлені векторами з нулів

та одиниць відповідно.

3) Отримує шифротексти для обох повідомлень — c_0 та c_1, — та обчислює їхні оцінки, що ґрунтуються на сумі перших коефіцієнтів шифротекстів $u[0][0]$ та $v[0]$.

4) Отримує шифротекст виклику $c^\star$ на одному з повідомлень m_b, де біт b обирається випадковим чином.

5) Порівнює оцінку $c^\star$ з оцінками c_0 та c_1 та вгадує, яке з повідомлень було зашифровано. Якщо відстань до c_0 менша — зловмисник вгадує $b = 0$, інакше — $b = 1$.

6) Підраховує кількість успішних вгадувань за велику кількість запусків та оцінює статистичну значущість результатів.

Результати експерименту виводяться у вигляді:

– Кількості вдалих вгадувань правильного повідомлення;

– Ймовірності успішного вгадування;

– Виграшу зловмисника порівняно з випадковим вгадуванням;

– p-значення (p-value), що вказує на статистичну значущість результатів;

– 95% довірчого інтервалу.

Для перевірки гіпотези про те, що зловмисник діє не краще за випадкове вгадування, було використано біноміальний тест із пакету `scipy.stats`. Статистичний аналіз дозволяє зробити висновки про вразливість реалізації до IND-CPA атак.

Результати тестування: нижче детально описані результати атаки, зображені на рисунку 2.5.

– Adversary success rate = 4971/10000

– Probability of guessing correctly = 0.4971: значення 0.4971 для ймовірності успішного вгадування майже дорівнює 0.5 — як випадкове вгадування.

– Advantage over random guessing = 0.0029: перевага над випадковим вгадуванням advantage становить лише 0.0029, тобто дуже мала.

– p-value (binomial test) = 0.5687: p-value більше за 0.05, а отже ми не

можемо відхилити нульову гіпотезу про те, що результат досягнутий випадково.

– 95% confidence interval [0.4873, 0.5069] : довірчий інтервал містить значення 0.5, що підтверджує: статистично результат не є підозрілим.

Результати виконання атаки зображені на наступних рисунках :

```
==========================================
IND-CPA with adaptive attack strategy
==========================================
Adversary success rate: 4971/10000
Probability of guessing correctly: 0.4971
Advantage over random guessing: 0.0029
==========================================
p-value (binomial test): 0.5687
95% confidence interval: [0.4873, 0.5069]
==========================================
```

Рисунок 2.5 – Результат атаки 1

```
==========================================
IND-CPA with adaptive attack strategy
==========================================
Adversary success rate: 4998/10000
Probability of guessing correctly: 0.4998
Advantage over random guessing: 0.0002
==========================================
p-value (binomial test): 0.9761
95% confidence interval: [0.4900, 0.5096]
==========================================
```

Рисунок 2.6 – Результат атаки 2

На основі наведених результатів можна зробити висновок, що реалізований алгоритм Baby Kyber пройшов тест на стійкість до IND-CPA атаки. Успішність атакувальника статистично не відрізняється від випадкового вгадування, отже, шифрування є стійким у цій моделі безпеки.

Реалізація атаки для IND-CCA

Для оцінки стійкості реалізованої версії Baby Kyber до IND-CCA - атаки було реалізовано експеримент, який моделює зловмисника, що має обмежений доступ до оракула розшифрування — без можливості розшифровувати виклик (challenge ciphertext) $c^\star$.

Атака ґрунтується на модифікації виклику $c^\star$ та аналізі результату розшифрування модифікованого шифротексту. Загальна стратегія виглядає так:

1) Ініціалізується схема Baby Kyber з параметрами q, n, k, η. Генеруються ключі: матриця A, вектор t та секретний ключ s.

2) Зловмисник обирає два повідомлення $m_0 = [0, 0, \ldots, 0]$ та $m_1 = [1, 1, \ldots, 1]$, та отримує шифротекст $c^\star$ для одного з них (m_b), де біт b обирається випадково.

3) Створюється оракул розшифрування $D(\cdot)$, який відмовляється обробляти запити на $c^\star$, але дозволяє всі інші.

4) Зловмисник модифікує компоненту $v^\star$ шифротексту $c^\star$, додаючи до неї невеликий відомий зсув, отримуючи змінений шифротекст c'.

5) Модифікований шифротекст c' передається до оракула $D(\cdot)$, який повертає розшифроване повідомлення.

6) Якщо сума елементів у результаті розшифрування ближча до $[1, 1, \ldots, 1]$, то зловмисник вгадує, що було зашифровано m_1, інакше — m_0.

7) Після N повторень експерименту обчислюються статистичні показники атаки.

Оцінка результатів включає:

– Загальну кількість вдалих вгадувань бітів b.

– Імовірність правильного вгадування ($\hat{p}$).

– Виграш зловмисника відносно випадкового вгадування ($|\hat{p} - 0.5|$).

– p-значення біноміального тесту, що оцінює статистичну значущість результату.

– 95% довірчий інтервал для $\hat{p}$.

Атака демонструє можливість відхилення від ідеальної IND-CCA стійкості за рахунок експлуатації навіть мінімальних змін у шифротексті.

Результати тестування: нижче детально описані результати атаки, зображені на рисунку 2.7.

– Adversary success rate = 10000/10000

– Probability of guessing correctly = 1.0000: ймовірність успішного вгадування становить 100%, що свідчить про повну успішність атаки.

– Advantage over random guessing = 0.5000: перевага над випадковим вгадуванням є максимальною — атакувальник завжди визначає правильне повідомлення.

– p-value (binomial test) = 0.0000: надзвичайно низьке значення p-value означає, що ймовірність досягнення такого результату випадково практично нульова.

– 95% confidence interval [1.0000, 1.0000]: довірчий інтервал повністю охоплює значення 1.0, що підтверджує абсолютну впевненість у результаті.

На основі наведених результатів можна зробити висновок, що реалізований алгоритм Baby Kyber не є стійким до IND-CCA атаки. Атакувальник мав повну можливість відрізнити зашифровані повідомлення, що свідчить про критичну уразливість у цій моделі безпеки. Для забезпечення повної криптографічної стійкості необхідно впровадити відповідні механізми захисту від атак із вибраним шифротекстом.

Результати виконання атаки зображені на рисунку:

```
============================================
IND-CCA experiment (basic attack strategy)
============================================
Adversary success rate: 10000/10000
Probability of guessing correctly: 1.0000
Advantage over random guessing: 0.5000
============================================
p-value (binomial test): 0.0000
95% confidence interval: [1.0000, 1.0000]
============================================
```

Рисунок 2.7 – Результат атаки

ВИСНОВКИ

У рамках переддипломної практики було реалізовано та досліджено спрощену версію постквантової криптосистеми Kyber — Baby Kyber. Було проведено теоретичний аналіз математичних основ алгоритму, визначено обмеження та особливості спрощеної моделі, зокрема фіксацію параметрів, відсутність перетворення Фур'є (NTT) та відмову від використання FO-перетворення.

Було розроблено програмну реалізацію алгоритму, яка охоплює генерацію ключів, шифрування та розшифрування повідомлень у вибраному параметричному просторі. Реалізація виконана мовою Python у середовищі PyCharm, із дотриманням модульного підходу до організації коду.

В межах експериментальної частини було проведено тестування стійкості алгоритму до атак типу IND-CPA та IND-CCA. За результатами аналізу встановлено, що реалізація демонструє статистичну стійкість до IND-CPA атак — ймовірність успішного вгадування не перевищує рівень випадковості. Водночас, алгоритм є повністю вразливим до IND-CCA атак через відсутність механізмів активного захисту, зокрема FO-перетворення.

Таким чином, реалізований алгоритм забезпечує базовий рівень конфіденційності у пасивній моделі супротивника, але не гарантує безпеки в умовах активного впливу. Робота демонструє практичне підтвердження важливості використання додаткових криптографічних трансформацій для досягнення повноцінної стійкості сучасних постквантових криптосистем.

ПЕРЕЛІК ПОСИЛАНЬ

1. Joppe Bos, Léo Ducas and others. CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM. — Cryptology ePrint Archive, Report 2017/634, 2017. Режим доступу: https://eprint.iacr.org/2017/634.pdf.

2. CRYSTALS-Kyber. Official website. — Режим доступу: https://pq-crystals.org/kyber/.

3. Kyber - How does it work?: Post-Quantum Cryptography. Cryptopedia. — Режим доступу: https://cryptopedia.dev/posts/kyber/.

4. Langlois, Adeline, Stehlé, Damien. Worst-Case to Average-Case Reductions for Module Lattices. — Cryptology ePrint Archive, Paper 2012/090, 2012. Режим доступу: https://eprint.iacr.org/2012/090/20130815:054538.

5. Gonzalez, Ruben. Kyber - How does it work? — Approachable Cryptography, 2021. Режим доступу: https://cryptopedia.dev/posts/kyber/.

6. Lyubashevsky, Vadim, Peikert, Chris, Regev, Oded. On Ideal Lattices and Learning with Errors over Rings. — Journal of the ACM, vol. 60, no. 6, pp. 1–35, 2013. Режим доступу: https://doi.org/10.1145/2535925.

7. Langlois, Adeline, Stehlé, Damien. Worst-Case to Average-Case Reductions for Module Lattices. — Cryptology ePrint Archive, Paper 2012/090, 2012. Режим доступу: https://eprint.iacr.org/2012/090/20130815:054538.

8. Pessl, Peter, Bruinderink, Leon Groot, Yarom, Yuval. To BLISS-B or not to be - Attacking strongSwan's Implementation of Post-Quantum Signatures. — Cryptology ePrint Archive, Paper 2017/490, 2017. Режим доступу: https://eprint.iacr.org/2017/490/20170828:125948.

9. Alkim, Erdem, Ducas, Léo, Pöppelmann, Thomas, Schwabe, Peter. NewHope without reconciliation. — Cryptology ePrint Archive, Paper 2016/1157, 2016. Режим доступу: https://eprint.iacr.org/2016/1157/20171109:073050.

10. Avanzi, Roberto, Bos, Joppe, Ducas, Léo, Kiltz, Eike, Lepoint, Tancrède, Lyubashevsky, Vadim, Schanck, John M., Schwabe, Peter, Seiler, Gregor, Stehlé, Damien. Algorithm Specifications and Supporting

Documentation (version 3.02). — 2021. Режим доступу: https://pqc-hqc.org/doc/hqc-specification_2024-02-23.pdf.

11. Botros, Leon, Kannwischer, Matthias J., Schwabe, Peter. Memory-Efficient High-Speed Implementation of Kyber on Cortex-M4. — Cryptology ePrint Archive, Paper 2019/489, 2019. Режим доступу: https://eprint.iacr.org/2019/489/20190520:092623.

12. Lyubashevsky, Vadim. Basic Lattice Cryptography: The concepts behind Kyber (ML-KEM) and Dilithium (ML-DSA). — Cryptology ePrint Archive, Paper 2024/1287, 2024. Режим доступу: https://eprint.iacr.org/2024/1287/20241024:103131.

13. FAQ on Kyber512. — National Institute of Standards and Technology (NIST). Режим доступу: https://csrc.nist.gov/csrc/media/Projects/post-quantum-cryptography/documents/faq/Kyber-512-FAQ.pdf.

14. CRYSTALS-KYBER Round 3 Official Comment. — National Institute of Standards and Technology (NIST). Режим доступу: https://csrc.nist.gov/csrc/media/Projects/post-quantum-cryptography/documents/round-3/official-comments/CRYSTALS-KYBER-round3-official-comment.pdf.

15. Yipei Yang, Zongyue Wang and others. Chosen ciphertext correlation power analysis on Kyber Future Generation Computer Systems, 2023. Режим доступу: https://www.sciencedirect.com/science/article/abs/pii/S0167926023000378.

16. Some KEMs and Some Proofs - Cronokirby— About - Cronokirby — 2022. Режим доступу: https://cronokirby.com/posts/2022/08/some-kems-and-some-proofs.

17. Campagna, Matthew. Security of Hybrid Key Establishment. — Springer International Publishing, 2020. Режим доступу: https://www.amazon.science/publications/security-of-hybrid-key-encapsulation.