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

Звіт з виконання комп'ютерного практикуму

# ДОСЛІДЖЕННЯ СУЧАСНИХ АЛГЕБРАЇЧНИХ КРИПТОСИСТЕМ

Виконали студентки
групи ФІ-32мн
Зацаренко А.Ю.
Футурська О.В.

Перевірив:
Фесенко А.В.

Київ — 2023

# ЗВІТ

## 1.1 Мета проведення комп'ютерного практикуму

Дослідження особливостей реалізації сучасних алгебраїчних криптосистем на прикладі учасника першого раунду процесу стандартизації постквантової криптографії (NIST PQC) – NTRU-HRSS-KEM.

## 1.2 Постановка задачі

Провести детальний огляд теоретичних відомостей про обраний криптографічний алгоритм та реалізувати усі можливі його варіації. На основі цього виконати порівняльний аналіз алгоритму щодо постквантової стійкості.

## 1.3 Хід виконання роботи, опис труднощів, що виникали, та шляхи їх подолання

Хід роботи:

1) Розглянути основні теоретичні відомості щодо відповідного криптографічного алгоритму, які необхідні для подальшої реалізації;

2) Розробити програмну реалізацію криптосистеми згідно з обраним варіантом;

3) Перевірити код на коректність за допомогою відповідних тестів;

4) Дослідити схожі алгоритми та модифікації і провести порівняльний аналіз на швидкодію;

5) Розглянути можливість перенесення відомих атак на обраний алгоритм.

Опис труднощів та шлях вирішення:

– Спочатку було прийнято рішення реалізовувати алгоритм на мові програмування *C#*. Однак, у нас виникли проблеми з реалізацією геш-функції Keccak, а саме з реалізацією раундових обчислень у контексті таких довгих значень. Було прийнято наступне: змінити мову і спробувати

почати писати реалізацію спочатку. Наш вибір впав на Python, оскільки там не має такої проблеми визначення типів довгих даних, хоча нам відомо, що цю мову не радять використовувати у реалізації криптографічних елементів. Використовуючи його, нам вдалося описати весь алгоритм, однак усе одно виникли питання з гешуванням;

– Обернення елементів у $S_q\_inverse$ працює дуже довго, покращити час допомогла оптимізація функції.

## 1.4 Детальний опис алгоритму NTRU-HRSS-KEM та його складових частин

У даному розділі буде наведено детальне теоретичне підґрунття, необхідне для виконання комп'ютерного практимуму.

### 1.4.1 NTRU

NTRU був вперше опублікований Гофштейном, Пайфером, Сільверманом у 1998 році. У цій оригінальній роботі поліноміальна алгебра і модульна арифметика використовуються для отримання криптосистеми, безпека якої ґрунтується на тому, що важко знайти надзвичайно короткі вектори в так званій ґратці NTRU. Ця задача безпосередньо пов'язана з більш загальною задачею про найкоротший вектор (SVP) на решітках, яка вважається стійкою до квантових атак.

У цій криптосистемі всі об'єкти, такі як ключі, повідомлення та зашифровані тексти, представлені поліномами максимального степеня. Зокрема, разом з поліномами визначено дві операції, які надають їй кільцеву структуру.

Параметри:

1) просте $n$:

2) $p, q$ – не обов'язково прості, НСД$(p, q) = 1, q > p$;

3) $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_r, \mathcal{L}_m$ – чотири набори поліномів степені $n - 1$ з цілими коефіцієнтами;

Кільце має вигляд $R = \mathbb{Z}[x]/(x^n - 1)$. Сам поліном записується в наступному вигляді:

$$f = \sum_{i=0}^{n-1} f_i x_i = \{f_0, f_1, \dots f_{n-1}\}$$

Крім цього, введемо операцію множення на $R$, яка позначається наступним знаком – $\circledast$ та яку задано як добуток циклічної згортки (конволюція). Отже, маємо, що

$$f \circledast g = h, \text{ де } h_k = \sum_{i=0}^{k} f_i g_{k-i} + \sum_{i=k+1}^{n-1} f_i g_{n+k-i} = \sum_{i+j \equiv k \ (mod \ n)} f_i g_j$$

Варто зазначити, що під мультиплікацією з модулем, мається на увазі зменшення коефіцієнтів за заданим модулем.

Нехай $r$ – випадковий многочлен. Позначимо $F_p$ та $F_q$ – відповідні обернені елементи. Після генерації отримуємо приватний ключ – пара $(f, F_p)$:

$$f \circledast F_q \ mod \ q \equiv 1 \text{ і } f \circledast F_p \ mod \ p \equiv 1$$

і публічний ключ $h$:

$$h \equiv f \circledast F_q \ (mod \ p).$$

Шифрування відбувається наступним чином:

$$c \equiv (p * r \circledast h + m) \ mod \ q$$

Розшифрування:

$$F_p \circledast a' \ (mod \ p), \text{ де } a \equiv f \circledast e \ (mod \ q),$$

де $a'$ – обернений згортковий многочлен до $a$.

NTRU вписується в загальні рамки ймовірнісної криптосистеми. Це означає, що шифрування включає в себе випадковий елемент, тому кожне повідомлення має багато можливих варіантів шифрування. Для відповідних значень параметрів існує надзвичайно висока ймовірність того, що процедура розшифрування відновить оригінальне повідомлення. Однак, деякі значення параметрів можуть призвести до випадкових невдач при розшифруванні, тому слід додавати декілька додаткових контрольних бітів

у кожний блок повідомлення. Звичайною причиною невдалого розшифрування є неправильне центрування повідомлення, тобто коли значення $a \notin [-\frac{q}{2}, \frac{q}{2}]$, і чим далі від даної множини, тим більше значення ймовірності провалу.

Як зазначається в оригінальній роботі, шифрування та дешифрування з NTRU надзвичайно швидкі, а створення ключів є простим процесом.

Крім NTRU-HRSS-KEM, у конкурсі також був представлений алгоритм NTRU Prime. Це варіант NTRU, який використовує скінченне поле виду $(Z/q)[x]/(x^n - x - 1)$, де $n$ –просте. Алгоритми NTRU-HRSS і NTRU Prime - це дві криптографічні схеми, засновані на ґратці NTRU. Вони є частиною пост-квантової криптографії та учасниками першого раунду процесу стандартизації постквантової криптографії (NIST PQC). На сьогодні NTRU-HRSS-KEM не є частиною процесу стандартизації пост-квантової криптографії NIST (був об'єднаний з іншим учасником), в той час як NTRU Prime є учасником третього раунду процесу стандартизації пост-квантової криптографії NIST.

Безпека NTRU ґрунтується на складності певних проблем ґратки (складність пошуку коротких векторів).

Приватний ключ $(f, g)$ і публічний $h = g * f^{-1}(mod\ q)$, $p, q$ – певні параметри.

В оригінальній версії $f(1) = 1, g(1) = r(1) = 0(mod\ q)$. Нехай є два відкритих текста: $m_0, m_1$

Оригінальна версія NTRU є семантично незахищеною криптосистемою щодо CPA. Оцінюючи шифротекст в 0 дає $c(1) = m(1)(mod\ q)$, так як $r(1) = 0(mod\ q)$. Це дозволяє зловмиснику провести успішну атаку, оскільки $m_1(x)$ і $m_2(x)$ мають різні значення при першому біті.

CCA атака також можлива: можна подати на вхід оракула $\tilde{c} = c + 1$. Тоді дешифратор поверне $\tilde{m} = m + 1$, звідки отримуємо $m$.

Щоб протистояти цьому, застосовують паддінг. До самого повідомлення на початок дописується рандомний паддінг, а $r(x)$ - геш від повідомлення і публічного ключа. Тоді в першому випадку перший біт не видаватиме ніякої

інформації про відкритий текст, а в другому – $\tilde{c}$ буде відхилятися оракулом.

### 1.4.2 OW CPA-безпечна схема шифрування NTRU HRSS

NTRUEncrypt High-Res Security Standard (NTRU HRSS) – це криптографічний алгоритм на основі ґратки, який належить до пост-квантової криптографії, який був вперше опублікований у 2017 році. Він є вдосконаленням оригінального алгоритму NTRU і призначений для забезпечення більш високого рівня захисту від атак, в тому числі з боку квантових комп'ютерів.

NTRU-HRSS - це схема шифрування з відкритим ключем, яка захищена OW CPA. Це поняття безпеки, яка гарантує, що схема шифрування захищена від атак на обраний відкритий текст (CPA), і додатково забезпечує складність отримання додаткової інформації про відкритий текст із зашифрованого, навіть за наявності потужних обчислювальних ресурсів.

Криптосистема є прямою параметризацією NTRU. Її конструктивна новизна полягає у виборі просторів вибірок для повідомлень, сліпих поліномів і закритих ключів.

Ці простори були обрані таким чином, щоб

а) NTRU-HRSS була коректною (дешифрування ніколи не дає збою);

б) допускала просту і ефективну реалізацію з постійним часом;

в) уникала зайвих параметрів, характерних для інших реалізацій NTRU.

**Основні зміни порівняно з NTRU:**

1) операції виконуються безпосередньо з $S_n$, щоб уникнути поширених проблем безпеки, пов'язаних з підкільцем $S_1$. Хоча можна реалізувати NTRU безпосередньо в $S_n$ і не використовувати $R_n$ взагалі, все ж елементи $S_n$ підносяться в $R_n$, щоб скористатися зручними обчислювальними і геометричними особливостями $R_n$;

2) параметри підбираються таким чином, щоб повністю виключити

збої при розшифруванні, і робиться це без обмеження простору ключа і повідомлення;

3) усувається будь-яка необхідність у розподілах з фіксованою вагою. Всі процедури вибірки обрано таким чином, щоб вони допускали прості та ефективні реалізації з постійним часом.

**Параметри:**

1) просте $n$, для якого:

а) порядок 2 в $(\mathbb{Z}/n)^\times$ дорівнює $(n-1)$, тобто 2 – генератор даної мультиплікативної групи;

б) порядок 3 в $(\mathbb{Z}/n)^\times$ дорівнює $(n-1)$, тобто 3 – генератор даної мультиплікативної групи.

У даній роботі ми використовуватимемо $n = 701$, щоб забезпечити 128-бітову безпеку для постквантів.

2) $p = 3$;

3) $q = 2^{3.5+log\ n}$, у нас це $q = 8192$;

Визначимо

$$\mathcal{T} = \{v \in \{-1, 0, 1\}^n : v_{n-1} = 0\}, \text{ та}$$
$$\mathcal{T}_+ = \{v \in \mathcal{T} : <x \circledast v, v> \geqslant 0\}$$

Тоді простори:

$$\mathcal{L}_f = \mathcal{L}_g = \mathcal{T}_+ \text{ і } \mathcal{L}_r = \mathcal{L}_m = \mathcal{T}$$

Крім цього, введемо наступні поняття:

– $\Phi_n$ – поліном вигляду $\dfrac{(x^n-1)}{(x-1)} = x^{n-1} + x^{n-2} + \ldots + 1$, має бути незвідним за модулем як $p$ та $q$. Це позбавляє від перевірки на обернений під час генерації ключів і робить процес більш придатним для реалізації з постійним часом. Цікаво, що для NTRU Prime умова незвідності вимагається лише для $q$, в той час як для NTRU ця вимога взагалі була не обов'язковою, хоча і рекомендованою;

– $S$ – фактор-кільце $\mathbb{Z}[x]/(\Phi_n)$;

– $S/p$ – фактор-кільце $\mathbb{Z}[x]/(p, \Phi_n)$.

Тоді канонічний $S/q$-представник полінома $a$ – це єдиний поліном $b$ степеня не більше $n-1$ з коефіцієнтами в $\{1, 2, \ldots q-1\}$ такий, що $a \sim b$ як елементи $S/q$.

| | |
|---|---|
| n | 701 |
| k | 2 |
| seed_bits | 256 |
| coin_bits | 256 |
| shared_key_bits | 256 |
| logq | 13 |
| q | 8192 |
| s3_packed_bits | 1120 |
| owcpa_public_key_bits | 9100 |
| owcpa_private_key_bits | 2240 |
| owcpa_ciphertext_bits | 9100 |
| cca_public_key_bits | 9100 |
| cca_private_key_bits | 10220 |
| cca_ciphertext_bits | 10220 |

**Рисунок 1.1** – Таблиця рекомендованих параметрів

**Генерація ключа:**

1) Обираємо $f$ та $g$ з $\mathcal{L}_f$ і $\mathcal{L}_g$;

2) Знаходимо таке $F_q$, що $(f \circledast F_q)\ mod\ q \equiv 1$ в $S$;

3) Знаходимо таке $F_p$, що $(f \circledast F_p)\ mod\ p \equiv 1$ в $S$;

4) $h = (p \circledast (x-1) \circledast g \circledast F_q)\ mod\ q$.

**Вихід:** приватний ключ $(f, F_p)$ і публічний ключ $h$.

У попередніх варіаціях NTRU $f$ вважалося коротким елементом $R$ з оберненими як в $R/p$, так і в $R/q$. З параметрами попереднього розділу, кожен ненульовий елемент $\mathcal{T}$ є оберненим як елемент $S/p$ та $S/q$. Оберненість в $S/p$ та $S/q$ є достатньою для процедури дешифрування, тому можна відмовитись від перевірки на наявність оберненого у $R/p$ та $R/q$. Все одно потрібно обчислювати обернені, але цей процес ніколи не дає збоїв.

**Шифрування:**

**Вхід:** повідомлення $m \in \mathcal{L}_m$.

1) Обираємо $r$ з $\mathcal{L}_r$;

2) $c = (r \circledast h + LiftP(m)) \ mod \ q$, де $LiftP(m) = (x-1) \circledast m_0$,

при $m_0 \in \mathcal{T}$ і $LiftP(m) = m$, у випадку $S/p$;

**Вихід:** шифротекст $c$.

У попередніх інстанціях NTRU $r$ та $m$ були обрані так, щоб мати коефіцієнти з $\{-1, 0, 1\}$ з визначеною кількістю коефіцієнтів, що приймають кожне значення. У даній схемі $r$ та $m$ приймають довільні значення на $\mathcal{T}$.

**Дешифрування:**

**Вхід:** шифроекст $c$

1) $v = (c \circledast f) \ mod \ q$;

2) $u = (u \circledast F_p) \ mod \ p$;

3) $m^{'} = (u - u_{n-1} \cdot \Phi_n) \ mod \ p$.

**Вихід:** $m^{'}$.

**Коректність:**

Алгоритм шифрування NTRU-HRSS з параметрами $p = 3$ та $q > 8\sqrt{2}n$ буде працювати коректно, якщо виконуватиметься наступна вимога:

$$c \circledast f = |r \circledast h + LiftP(m) \circledast f| < \frac{q}{2}$$

Розписавши $h$ та $LiftP(r)$, отримаємо наступне:

$$c \circledast f = |r \circledast p \circledast (x-1) \circledast g + LiftP(m) \circledast f| < \frac{q}{2}$$
$$c \circledast f = |p \circledast LiftP(r) \circledast g + LiftP(m) \circledast f| < \frac{q}{2}$$

Згідно з лемою, описаною та доведеною в офіційному документі[1], маємо:

$$LiftP(r) \circledast g \leqslant \sqrt{2}n$$

Тоді отримаємо, що

$$c \circledast f = |p \circledast LiftP(r) \circledast g + LiftP(m) \circledast f| =$$
$$= |3 \circledast LiftP(r) \circledast g + LiftP(m) \circledast f| < 4\sqrt{2}n < \frac{q}{2}$$

[1]Andreas Hülsing, Joost Rijneveld, John Schanck, and Peter Schwabe: High-speed key encapsulation from NTRU

Варто зазначити, що точний розподіл, з якого беруться $f$ і $g$, впливає на безпеку. Генерація ключів використовує загальну функцію $Sample\mathcal{T}_+$, яку можна розглядати як вибірку з рівномірного розподілу на $Sample\mathcal{T}_+$. Однак це складно і довго, тому варто розглянути інші варіанти. Спочатку зазначимо, що будь-яку процедуру вибірки з $\mathcal{T}$ можна перетворити на процедуру вибірки з $\mathcal{T}_+$ з втратою не більше одного біта в ентропії її вихідного розподілу.

Функція вибірки з $Sample\mathcal{T}_+$ обирає $v \in \mathcal{T}$ і потім умовно застосовує парне перевертання знаку індексу до $v$, якщо $<xv, v> < 0$.

Спрощена процедура вибірки з $Sample\mathcal{T}$ витягує $n - 1$ коефіцієнт незалежно з центрованого біноміального розподілу з параметром $t = 2$, а потім зменшує ці коефіцієнти за модулем $p$.

Центрований біноміальний розподіл з параметром $t$ визначається, як

$$\sum_{i=1}^{t} b_i - b_{t+i}, \text{ де } b_1, b_2, \ldots b_{2t} - \text{рівномірні випадкові біти}$$

Процес завжди споживає рівно $2t(n-1)$ випадкових бітів. Результуючий розподіл відноситься до типу розподілів, що демонструє симетрію відносно певної центральної точки (для будь-яких значень $p$ і $t$) і прямує до рівномірного розподілу зі збільшенням $t$.

### 1.4.3 CCA2-безпечний механізм інкапсуляції ключів (KEM)

Адаптивна атака на вибраний шифрований текст (скорочено CCA2) - це інтерактивна форма атаки на вибраний шифрований текст, в якій зловмисник спочатку надсилає декілька шифрованих текстів для розшифрування, вибраних адаптивно, а потім використовує результати для розпізнавання цільового шифротексту, не звертаючись до оракула щодо шифротексту.

В оригінальній роботі зазначається INDCCA2 безпека. Якщо криптосистема володіє властивістю нерозрізнення, то зловмисник не зможе розрізнити пари шифротекстів на основі зашифрованого ними повідомлення. Властивість нерозрізнення під час атаки з обраним

відкритим текстом вважається основною вимогою для більшості криптосистем з відкритим ключем.

Криптосистема вважається безпечною з точки зору нерозрізнення, якщо жоден зловмисник, зашифрувавши повідомлення, випадково вибране з двоелементного простору повідомлень, визначеного зазделегідь, не зможе ідентифікувати вибір повідомлення з ймовірністю, значно кращою, ніж ймовірність випадкового вгадування ($p = \frac{1}{2}$). Якщо будь-який зловмисник може розпізнати вибраний шифротекст з ймовірністю, значно більшою за $\frac{1}{2}$, то вважається, що він має «перевагу» у розпізнаванні шифротексту, і схема не вважається безпечною з точки зору нерозрізненості. Це визначення охоплює поняття того, що в безпечній схемі противник не повинен дізнатися ніякої інформації, побачивши шифротекст. Таким чином, противник не повинен мати змоги розгадати шифр краще, ніж якби він вгадував його випадковим чином.

Шифрування з відкритим ключем (PKE) - це просто використання можливості зашифрувати щось за допомогою відкритого ключа, а потім розшифрувати його іншою стороною за допомогою закритого ключа. Однак, методи інкапсуляції ключів (KEM) також використовують відкритий і закритий ключі, але природа KEM полягає в тому, що він генерує симетричний ключ як частину свого процесу.

PKE з достатньо великим технічним простором можна досить тривіально перетворити на KEM. А сам KEM можна перетворити на PKE, додавши трохи симетричного шифрування.

У квантовому розумінні, існує багато чого, що має бути передано між процесами. Тому автоматична генерація симетричного ключа має сенс. Іноді це просто через обмеження, пов'язані з тим, як працює квантовий алгоритм. Але зазвичай це пов'язано з тим, що для того, щоб відбулася комунікація, потрібно щось передати, і тому KEM просто має більше сенсу.

NTRU-HRSS-KEM - це KEM із захистом CCA2, який також був представлений у 2017 році як спроба перетворити вищезгадане безпечне шифрування OW CPA на KEM із захистом CCA2. Конструкція

використовує загальне перетворення зі схеми шифрування з відкритим ключем, що захищена OW CPA. Як пряме перетворення KEM дозволяє уникнути механізму паддінгу NAEP, який використовується в стандартному NTRU.

Принцип роботи KEM-перетворення полягає в наступному. Спочатку з простору повідомлень схеми шифрування вибирається випадкове повідомлення $m$. Цей текст шифрується за допомогою випадкового рядку бітів (coins), детерміновано отриманих з $m$ за допомогою геш-функції, яка пізніше моделюється як випадковий оракул. Ключ сеансу отримується з $m$ за допомогою іншого оракула. Нарешті, виводиться шифротекст і сеансовий ключ.

Алгоритм декапсуляції розшифровує зашифрований текст, щоб отримати $m$, виводить випадкові рядки (coins) з $m$ і повторно шифрує $m$, використовуючи ці рядки (coins). Якщо отриманий шифротекст збігається з раніше надісленим, то він генерує сеансовий ключ з $m$.

Подібний KEM на основі NTRU був запропонований Мартіном Стамом (Martijn Stam) у 2005 році; основні відмінності від цієї роботи полягають у виборі параметрів для NTRU та включенні додаткового гешу, який додається до зашифрованого тексту. Додатковий геш дозволяє довести безпеку у квантово-доступній моделі випадкового оракула.

## 1.5 Результати порівняльного аналізу швидкодії обраного алгоритму зі схожими алгоритмами (або модифікаціями алгоритму за допомогою заміни складових частин)

Сфокусуємося на порівнянні з іншими системами на основі ґратки.

### 1.5.1 Порівняння зі стандартною NTRU

Деякі переваги та недоліки NTRU-HRSS у порівнянні зі стандартною NTRU є наступними:

✓ Відсутність збоїв при дешифруванні. Набори параметрів NTRU EES мають невелику, але ненульову ймовірність помилки дешифрування;

✓ Відсутність механізмів підстановки. Використовуючи пряму побудову KEM, уникається необхідності в механізмі підбиття, подібному до NAEP;

✓ Відсутність кседованих вагових розподілів. Набори параметрів НТРУ ЕЕС використовують вектори коефіцієнтних ваг щоб гарантувати, що інформація про секретні ключі (або повідомлення) не буде розкрита через $h(1)$ (або $c(1)$). Це складніше реалізувати в постійному часі, ніж комбінацію $Sample_T$ і $S_3\_to\_R$, що використовується в NTRU-HRSS;

✓ Відсутність відбраковування вибірки. NTRU EES використовує рядки з рівномірних випадкових бітів для вибірки рівномірних випадкових трійок та рівномірних значень у діапазоні $\{0, 1, ..., n-1\}$. Для того, щоб ці процеси були успішними з усією ймовірністю, окрім мізерно малої, необхідно вибрати багато бітів;

✓ Секретні ключі завжди є оберненими. Обмеження на $n$ гарантують, що $f$ завжди є оберненим за модулем 2. Параметри NTRU підібрані таким чином, що ймовірність генерації неінверсного $f$ є малою, але не обов'язково нульовою;

✗ Великий модуль. NTRU-HRSS потребує порівняно великого модуля, щоб виключити збої при дешифруванні. збоїв при дешифруванні. Це зменшує безпеку та збільшує вартість зв'язку. Набір параметрів $n = 701$ для NTRU-HRSS дасть приблизно 20 біт безпеки при використанні модуля $q = 2048$ замість $q = 8192$. Довжина зашифрованого тексту також зменшиться на 175 байт;

✗ Інвертує $mod\ p$. NTRU EES бере $f \equiv 1(modp)$ і таким чином уникає множення на $f - 1(mod\ p)$ під час розшифрування. Виконання того ж самого в NTRU-HRSS вимагало б ще більшого модуля.

### 1.5.2 Порівняння зі спрощеним NTRUPrime

Деякі переваги та недоліки NTRU-HRSS порівняно з Streamlined NTRUPrime є наступними:

✓ Відсутність зміщених вагових розподілів. Streamlined NTRUPrime використовує xed-розподіл ваг для доведення правильності;

✓ Закриті ключі завжди є інвертованими. Можна підібрати $f$, який не є оберненим за модулем 3 у Streamlined NTRUPrime;

✓ Степінь за модулем 2. Streamlined NTRUPrime вимагає простий модуль. Деякі арифметичні операції виконуються швидше, коли $q$ є степенем 2;

✗ Циклотомічне кільце. NTRUPrime було розроблено, щоб уникнути проблемної структури циклотомічних кілець;

✗ Хоча алгебраїчна структура не дає уявлення про вартість найвідоміших атак на NTRU-HRSS,цілком можливо, що існують кращі алгебраїчні атаки. Також можна припустити, що такі атаки не застосовуються до кілець, що використовуються в NTRUPrime;

✗ Імовірнісне шифрування. Спрощений NTRUPrime побудовано як детерміновану схему шифрування з відкритим ключем з детермінованим відкритим ключем;

✗ Без округлень у стилі LWR. Шифротексти оптимізованого NTRUPrime можна стискати.

### 1.5.3 Порівняння з системами LWE

✓ Відсутність збоїв при дешифруванні. Більшість практичних схем LWE вибирають малу частоту відмов при дешифруванні а не вузькому розподілу коецієнтів або великому модулю;

✗ Більша розмірність для однакової безпеки. Для того, щоб виключити збій при дешифруванні, NTRU-HRSS використовує потрійні секретні ключі та повідомлення. Це призводить до нижчого рівня безпеки, ніж можна було

б мати з тією ж розмірністю і модулем, але більшим шумом.

## 1.6 Результати порівняльного аналізу стійкості обраного алгоритму зі схожими алгоритмами з обґрунтуванням можливості застосування відомих атак

Наведемо основні види атак, описані в офіційному документі.

### 1.6.1 Атаки на основі ґратки

Найбільш ефективні атаки на ґратку в літературі розглядають відновлення ключа NTRU як унікальну задачу з найкоротшим вектором. Залежно від передбачуваної неасимптотичної вартості зменшення ґратки, ці атаки використовують вгадування (або ігнорування) коефіцієнтів для зменшення розмірності ґратки. Нещодавно представлені алгоритми сита мають єдину експоненціальну «вартість», яка є достатньо малою, щоб поставити під сумнів ефективність вгадування коефіцієнтів, однак не ясно, що комбінаторні методи не мають значення в реалістичних моделях обчислень.

Атаки на ґратки часто включають алгоритми редукції ґратки, метою яких є перетворення заданої ґратки в «кращу» ґратку, в якій легше знайти найкоротший вектор. Найвідомішим алгоритмом редукції ґратки є алгоритм LLL (Lenstra-Lenstra-Lovász), але існують і більш просунуті варіанти, такі як алгоритм BKZ (Block Korkine-Zolotarev).

NTRU-HRSS був визнаний захищеним від атак на ґратки, в тому числі тих, що використовують проблему найкоротшого вектора (SVP) в криптографії на основі ґратки.

Core-SVP - це особливий випадок SVP, в якому на решітку або розв'язок можуть бути накладені додаткові обмеження або умови. Ці обмеження можуть включати обмеження на базис ґратки, форму ґратки або дозволені перетворення на ґратці. Введення таких обмежень може

зробити задачу більш спеціалізованою або пристосованою до конкретних криптографічних додатків.

У контексті ґраткової криптографії SVP, включаючи її різновиди, такі як Core-SVP, часто використовується для встановлення складності певних задач на ґратці, що є основою для безпеки ґраткових криптографічних схем. Складність розв'язання SVP має важливе значення для безпеки систем на основі ґратки, оскільки вважається, що ці проблеми є складними навіть для квантових комп'ютерів, що робить їх потенційними кандидатами для постквантової криптографії.

Дана задача поділяється на дві атаки: первісну та гібридну.

Первісна атака має два параметри: $b$ – розмір блоку, який використовується для зменшення ґратки, та $m$ – параметр зменшення розмірності. Первинна атака намагається розв'язати унікальну СЛП у підгратці рангу $d = n' + m$ та об'ємом $q^m$. Параметри $b$ та $m$ підібрані таким чином, що короткий вектор у цій спроектованій підґратці з великою ймовірністю може бути відображений у короткий вектор за допомогою алгоритму найближчої площини Бабая.

Декодування списком (англ. List Decoding Sieve) – це метод, який використовується в теорії кодування і криптографії. Він передбачає декодування отриманого повідомлення в список можливих оригінальних повідомлень, а не в унікальне повідомлення. Метод сита (решета) часто використовується для ефективного декодування списків.

Алгоритм Гровера (англ. Grover's Search Algorithm) – це квантовий алгоритм, який можна використовувати для пошуку в несортованій базі даних за $O(\sqrt{N})$ часу, що забезпечує квадратичне прискорення порівняно з класичними алгоритмами. Він має наслідки для зламу певних криптографічних примітивів, таких як шифрування з симетричним ключем.

Квантова вартість решета декодування списку залежить, головним чином, від використання моделі квантової оперативної пам'яті для обчислень. Визначення її вартості в схемотехнічній моделі є відкритою проблемою, що викликає значний інтерес. Кожна ітерація Гровера

звертається до блоку пам'яті. Якщо квантова пам'ять вимагає активної корекції помилок, то перевага над вартістю буде втрачена. Навіть якщо припустити, що на вектор припадає лише $b$ біт, вартість атаки з рекомендованим набором параметрів становитиме $2^145$ біт. Просте заповнення цієї пам'яті вже було б настільки ж дорогим як і пошук ключа в AES-128.

Сито декодування списків дозволяє великий ступінь розпаралелювання і може бути можна налаштувати так, щоб уникнути великих меж глибини у класичній моделі оперативної пам'яті. У меншій мірі це також справедливо, коли пошук Гровера використовується для виконання індивідуального пошуку найближчих сусідів, перерахувавши глибину ітерацій цих пошуків, тобто кількість ітерацій Гровера, необхідних для виконання кожного пошуку найближчого сусіда. Залежно від вартості схеми однієї ітерації Гровера, цілком ймовірно, що невелика глибина може може бути насиченою. Наприклад, оптимальна за часом параметризація розмірністю 465 має глибину ітерації $2^26$. Це призведе до насичення обмеження на глибина у $2^40$ на глибина, якби схема для ітерації Гровера мала глибину $2^14$ квантових воріт. Для порівняння, схема для одного раунду AES-128, наведена в [10], має глибину 213.4 логічних квантових воріт. Тим не менш, обмеження глибина до $2^64$, ймовірно, не вплине на оцінку безпеки Core-SVP для нашого набору параметрів $n = 701$.

Гібридна атака націлена на підґратку рангу $d = n' + m$. Нехай $w$ - короткий вектор у ґратці. Зловмисник намагається вгадати $s$ співмножників проекції $w$. Якщо здогадка то його можна підняти до короткого вектора за допомогою алгоритму найближчої площини Бабая.

У офіційному документі стверджується, що гібридна атака не конкурує з первинною атакою, коли обидві атаки використовують сито декодування списків. Однак є ще кілька цікавих компромісів, які варто розглянути. Нагадаємо, що витрати залежать від використання моделі квантової оперативної пам'яті. При цьому незрозуміло, чи є квантова оперативна пам'ять дешевшою за квантову схему загального призначення.

Це призводить до того, що нас розглянути паралельні гібридні атаки, які використовують менше квантових схем, ніж було б потрібно для запуску сита декодування списків на квантовому комп'ютері.

### 1.6.2 Атаки на симетричні примітиви

Єдиним симетричним примітивом, який використовується, є SHAKE128. Зауважимо, що KEM, в принципі, може бути використаний для обміну близьким до $n * log3$ бітами ключового матеріалу. Більш того, без проблем можна застосувати будь-яку іншу геш-функцію без втрати семантичних властивостей та втручань у інші частини конструкції криптопримітиву.

### 1.6.3 Відома атака IND-CCA2 на NTRU-HRSS-KEM

NTRU-HRSS призначений для захисту від вибраних атак відкритого тексту. У роботі стверджується, що NTRU-HRSS-KEM відповідає стандартному визначенню безпеки IND-CCA2 для механізму інкапсуляції ключів. Параметри підібрані таким чином, щоб унеможливити збій при розшифруванні, а ключ можна було використовувати повторно щонайменше $2^{64}$ рази без шкоди для безпеки. У сценарії CPA, де зловмисник може вибирати відкриті тексти і спостерігати за відповідними їм шифротекстами, NTRU-HRSS забезпечить семантичну безпеку. Проте NTRU-HRSS, як і багато схем шифрування на основі ґратки, може не забезпечувати захист від атак з адаптивним підбором шифротексту CCA2. NTRU-HRSS-KEM має відносно невеликий набір параметрів, що робить його проблематичним у певних моделях атак, таких як багатоцільова атака на вибраний шифротекст з одноразовою однобітовою помилкою.

У моделі атаки IND-CCA2, доповненій одноразовим перевертанням одного біта, що зберігається у авторизованого користувача, NTRU-HRSS є катастрофічно незахищеною: існує ефективна атака, яка відновлює сеансовий ключ NTRU-HRSS. У моделі багатоцільової атаки IND-CCA2,

аналогічно доповненої одноразовою однобітовою помилкою, така ж атака ефективно відновлює всі сеансові ключі NTRU-HRSS, які були інкапсульовані до цільового відкритого ключа до помилки. Атака може пройти таким шляхом: зловмисник ініціює процес шифрування, потім вводить помилку в систему і спостерігає за її впливом на шифрований текст. Зловмисник адаптивно вибирає шифротексти для розшифрування цільовою системою, знаючи про подану помилку, і експлуатує уразливість для отримання інформації про секретний ключ. Нарешті, зловмисник успішно відновлює сеансовий ключ NTRUE-HRSS за допомогою аналізу дефектного шифротексту та інших даних.

Іншою схемою є NTRU Prime, який призначений для забезпечення семантичної безпеки щодо обраної атаки на відкритий текст (IND-CPA) і вважається більш безпечним варіантом з точки зору семантичної безпеки в порівнянні з NTRU-HRSS-KEM. В рамках проекту NTRU Prime, Streamlined NTRU Prime - це невеликий KEM на основі ґратки, спрямований на досягнення стандартної мети безпеки IND-CCA2. Існують й інші пропозиції невеликих KEM на основі ґратки, спрямовані на досягнення IND-CCA2, але Streamlined NTRU Prime систематично розробляється з метою мінімізації складності ретельного аналізу безпеки.

## 1.7 Детальний опис особливостей реалізації та приклади застосування

Програмна реалізація NTRU-HRSS-KEM складається з 5 класів:

1) Externally defined algorithms. У цьому класі реалізована функція гешування SHAKE128;

2) Arithmetic Algorithms. У цьому класі містяться алгебраїчні операції над многочленами, включно із знаходженням оберненого та функції для знаходження канонічних $R_q$, $S_q$, $S_3$, $S_2$ представників полінома. Канонічні представники дозволяють здійснювати операції над поліномами у відповідних алгебраїчних структурах, зокрема, у фактор-кільцях і полях.

Це полегшує виконання арифметичних операцій, таких як додавання, віднімання, множення тощо, і дозволяє зберігати інваріантність відносно цих операцій;

3) Sampling Algorithms. Це клас містить дві функції, які використовуються для створення ключів та генерації параметрів системи. Вони необхідні для випадкового вибору числових параметрів, таких як коефіцієнти поліномів, що застосовуються для інкапсуляції та декапсуляції даних. Ефективні алгоритми вибіркового вибору гарантують важливі криптографічні властивості, такі як стійкість до атак та властивості безпеки системи;

4) Encoding Algorithms. Алгоритми в цьому класі виконують перетворення канонічних $R_q$ та $S_3$ представників поліномів у бітовий рядок, і навпаки. Вони дозволяють працювати з поліномами у бітовому форматі, що може бути зручним для певних завдань або пристосування до конкретних вимог системи;

5) Key Encapsulation Mechanism. У цьому класі реалізований основний механізм інкапсуляцію ключів, який є CCA-безпечним у моделі квантового випадкового оракула. Він складається з трьох основних функцій, з якими і буде працювати користувач, а саме алгоритми генерації, інкапсуляції та декапсуляції ключів. Інші функції в цьому класі використовуються цими алгоритмами, але не є частиною публічного API. Наприклад, функції NTRU_OWF_Public (One-Way Function для публічного ключа) та NTRU_OWF_Private (One-Way Function для приватного ключа) є частиною схеми гешування, яка застосовується в NTRU-HRSS-KEM з метою забезпечення безпеки. Вони допомагають у створенні внутрішнього представлення ключа у вигляді гешу, що робить важким повернення початкового ключа з його геш-значення.

### 1.7.1 Гешування SHAKE

Як новий стандарт SHA-3 було обрано геш-функцію Keccak зі змінною довжиною виходу $224, 256, 384$ і $512$ біт. В основі Keccak лежить конструкція під назвою Sponge (губка), яка складається з двох етапів:

1) Absorbing(вбирання) – початкове повідомлення M піддається багатораундовим перестановкам $f$;

2) Squeezing(віджимання) – виведення отриманого в результаті перестановок значення $Z$.

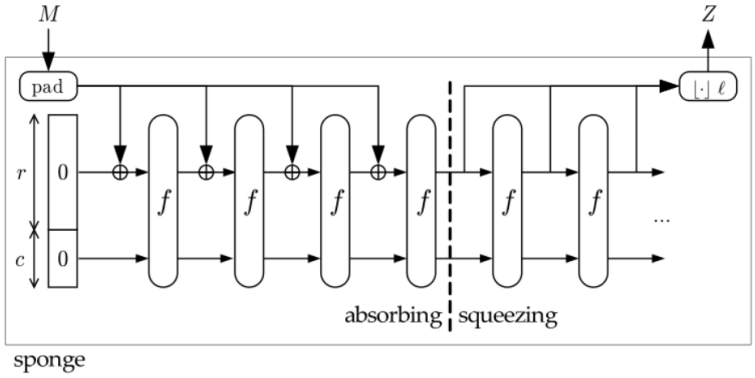


**Рисунок 1.2** – Конструкція губки

Геш-функцію Keccak реалізовано таким чином, що функцію перестановки $f$, застосовувану для кожного блоку $M_i$, користувач може обирати самостійно з набору зумовлених функції $b = \{f - 25, f - 50, f - 100, f - 200, f - 400, f - 800, f - 1600\}$. Щоб використовувати, скажімо, функцію $f - 800$, необхідно вибрати такі $r$ і $c$, щоб виконувалася рівність $r + c = 800$.

Крім того, змінюючи значення $r$ і $c$, змінюється кількість раундів геш-функції. Кількість обчислюється за формулою $n = 12 + 2l$, де $2^l = \frac{b}{25}$. Так для $b = 1600$, кількість раундів дорівнює 24.

Однак, хоча користувач має право вибирати для своєї реалізації будь-яку із запропонованих авторами функцій, слід зазначити, що як стандарт SHA-3 прийнято тільки функцію Keccak-1600, і автори всіляко рекомендують

користуватися тільки нею. Так в якості основних значень для гешів різної довжини автори вибрали такі параметри:

SHA-256: $r = 1088, c = 512$ (повернути перші 32 байт результат)

SHA-512: $r = 576, c = 1024$ (повернути перші 64 байт результат)

Крім цього, до початкового повідомлення застосовується падінг у вигляді 1111, що визначається як суфікс та використовується для більш ефективного обчислення.

## 1.8 Результати аналізу постквантової стійкості за наявними результатами аналізу

Очікується, що порушення однобічності NTRU-HRSS вимагатиме обчислювальних ресурсів більших, ніж ті, що необхідні для виконання перебору ключів в AES-128. Безпека базується в основному на тому, що вартість найвідомішої класичної атаки складає $2^{136}$ операцій і $2^{136}$ пам'яті. Ці операції маскують великі фактори, які роблять справжню вартість атаки набагато вище, ніж $2^{145}$ біт, необхідних для атаки на AES-128, навіть у моделі з оперативною пам'яттю.

Безпека також ґрунтується на тому, що вартість найвідомішої квантової атаки становить $2^{123}$ ітерацій Гровера. Ця атака відбувається в моделі квантової оперативної пам'яті і вимагає квантово доступної класичної пам'яті розміром $2^{123}$. Знову ж таки, великі значення ігноруються, і малоймовірно, що ця атака збереже свою перевагу над класичним варіантом, коли вона реалізована у квантовій моделі схеми.

# ВИСНОВКИ

У даній лабораторній роботі було розглянуто NTRU-HRSS-KEM – криптографічну схему з механізмом капсуляції, засновану на ґратці NTRU. Вона є частиною пост-квантової криптографії та учасником першого раунду процесу стандартизації постквантової криптографії (NIST PQC). На сьогодні NTRU-HRSS-KEM не є частиною процесу стандартизації пост-квантової криптографії NIST.

Однак варто зазначити, що під час 1-го раунду було кілька кандидатів, заснованих на оригінальній криптосистемі NTRU, і фактично кандидат NTRU є результатом злиття NTRUEncrypt і NTRU-HRSS-KEM перед початком Раунду 2. NTRU - це ще одна криптосистема з відкритим ключем на основі структурованої решітки, яка має рівень захисту IND-CCA2.

Причиною дискваліфікації, якщо можна так назвати, це сумнівність в стійкості проти IND-CCA2, оскільки вже існують атаки, які відновлюють сеансовий ключ, що гіпотетично може означати знаходження частково, або і повністю, відкритих текстів.