НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичних методів захисту інформації

Звіт до лабораторної роботи за темою: Дослідження сучасних алгебраїчних криптосистем на прикладі постквантових криптографічних алгоритмів. Алгоритм TiGER

Оформлення звіту:
Юрчук Олексій, ФІ-52мн
Дигас Богдан, ФІ-52мн

6 листопада 2025 р.
м. Київ

ЗМІСТ

Розділ 1

Вступне слово

Мета роботи (власне, для чого ми тут зібралися):

Дослідити особливостей реалізації сучасних алгебраїчних криптосистем на прикладі учасників першого раунду національного конкурсу з постквантової криптографії в Кореї (KpqC).

Наші задачі на комп'ютерний практикум та порядок їх виконання:

1) Роздітися на бригади. Визначили хто за що відповідатиме. Богдан – займається реалізацією алгоритму TiGER, Олексій – теоретичною частиною і звітом загалом.

2) Провести теоретичне дослідження теми, надавши вичерпний та повний опис теоретичної сторони алгоритму з усіма деталями та відомими результатами досліджень; провести аналіз вже існуючих атак на алгоритм TiGER, а також загалом можливих атак; виконати порівняльний аналіз нашого алгоритму зі схожими та дослідити можливість перенесення та застосування відомих атак на нього.

3) Реалізувати алгоритм програмно та всі(нє, ну ми постараємося) можливі варіанти цього алгоритму;

4) Перевірити коректність – підтвердити правильність реалізації за допомогою тестів, використавши тестові дані з офіційної реалізації;

5) Зробити аналіз продуктивності алгоритму та, знову ж таки, провести порівняння та аналіз швидкодії за різних умов, дослідити вплив модифікацій окремих його складових частин на ефективність.

Розділ 2

Загальне теоретичне дослідження

2.1 Постквантова криптографія

Сучасна криптографія з відкритим ключем, зокрема RSA та криптографія на еліптичних кривих – Elliptic Curve Cryptography (ECC), базується на обчислювальній складності задач факторизації великих чисел та дискретного логарифмування. Однак у 1994 році Пітер Шор [1] продемонстрував квантові алгоритми, здатні розв'язувати ці задачі за поліноміальний час на достатньо потужному квантовому комп'ютері. Це створює критичну загрозу для існуючої криптографічної інфраструктури.

Постквантова криптографія (Post-Quantum Cryptography, PQC) – це галузь криптографії, що розробляє алгоритми, стійкі як до класичних, так і до квантових атак. Серед основних напрямків PQC виділяють криптографію на решітках, криптографію на кодах виправлення помилок, багатовимірну поліноміальну(квадратичну) криптографію та криптографію на основі геш-функцій [2].

2.2 Передумови створення TiGER

Механізм інкапсуляції ключа (Key Encapsulation Mechanism, KEM) є одним з найважливіших криптографічних примітивів для захищеного обміну ключами. У контексті заміни класичних протоколів, таких як Diffie-Hellman (DH) або Elliptic Curve Diffie-Hellman ECDH, постквантові KEM повинні забезпечувати не лише високий рівень безпеки, але й бути ефективними за розміром даних та залишатися обчислювано складними для зламу зловмисником.

Криптографія на решітках, зокрема алгоритми на основі задач Learning With Errors (LWE) [3] та Ring Learning With Errors (RLWE) [4], продемонструвала перспективність у створенні ефективних постквантових схем. Розвиток цього напрямку призвів до появи сімейства алгоритмів, що використовують детермінований варіант – Learning With Rounding (LWR) [5], який замінює випадкову помилку округленням, що покращує як продуктивність, так і довжину шифротексту.

Серед попередніх розробок слід відзначити алгоритми Lizard [6] та RLizard [7], які комбінували RLWE для генерації ключів з RLWR для шифрування, досягаючи балансу між безпекою та ефективністю. Однак ці схеми мали певні обмеження щодо розміру відкритого ключа та шифротексту, що ускладнювало їх інтеграцію в існуючі протоколи.

TiGER (Tiny bandwidth key encapsulation mechanism for easy miGration based on RLWE(R)) [8] був розроблений командою дослідників з метою створення компактного та ефективного KEM, придатного для легкої інтеграції в існуючі системи безпеки. Основні задачі, які ставили перед собою науковці це:

- **Мінімізація розміру шифротексту та відкритого ключа**
- **Висока обчислювальна ефективність** — використання в якості модуля число, яке є степенем двійки ($q = 2^k$) (для оптимізації операцій округлення через побітові зсуви);
- **Відмова від NTT** — алгоритм не використовує Number Theoretic Transform, що спрощує реалізацію;
- **Використання розріджених секретів (з малою вагою Гемінга)** — зменшення розміру секретного ключа та прискорення множення многочленів;
- **Корекція помилок** — застосування кодів XEf та D2 для зниження ймовірності помилки дешифрування.

Конструкція TiGER базується на комбінації RLWR для генерації відкритого ключа та RLWE для шифрування, з подальшим застосуванням перетворення Fujisaki-Okamoto [9, 10] для досягнення IND-CCA безпеки.

2.3 Участь у KpqC та злиття з SMAUG

У 2022 році Національна служба розвідки Республіки Корея ініціювала Korean Post-Quantum Cryptography Competition скорочено – KpqC [11]. Це національний конкурс для стандартизації постквантових криптографічних алгоритмів.

Обраний нами для аналізу алгоритм TiGER був поданий на перший раунд конкурсу KpqC у категорії механізмів інкапсуляції ключа (KEM) і був одним з чотирьох алгоритмів, які пройшли до другого раунду.

2.3.1 Злиття TiGER та SMAUG

Команди TiGER та SMAUG об'єдналися для створення спільного алгоритму SMAUG-T [12]. Метою злиття було поєднання переваг обох підходів:

- Від **TiGER**: Компактність шифротексту, використання RLWE/RLWR на кільцевому рівні, корекція помилок через D2 кодування (для параметра TiMER);
- Від **SMAUG**: Модульна структура (MLWE/MLWR), розріджені секрети через використання гаусівського шуму, покращена безпека за рахунок збільшення розмірності.

Результатом злиття став алгоритм SMAUG-T версії 3.0 (лютий 2024), який включає в себе:

- Три основні набори параметрів: **SMAUG-T128**, **SMAUG-T192**, **SMAUG-T256** (відповідають рівням безпеки NIST 1, 3, 5);
- Додатковий набір параметрів **TiMER** (Tiny SMAUG using Error Reconciliation) – оптимізований для IoT(Internet of Thing)-пристроїв з мінімальним шифротекстом завдяки використанню D2 кодування з TiGER.

2.3.2 Результати KpqC 2023

У січні 2025 року було оголошено фінальні результаті конкурсу KpqC. Переможцями стали:

- У категорії KEM: **SMAUG-T** та **NTRU+**;
- У категорії цифрового підпису: **HAETAE** (до речі, також від команди SMAUG).

Таким чином, ідеї та технології TiGER увійшли до складу національного стандарту постквантової криптографії Кореї через алгоритм SMAUG-T.

Розділ 3

Теоретична база алгоритму TiGER

3.1 Алгебраїчні структури

Критографія на решітках (Lattice-based cryptography) використовує алгебраїчні структури для забезпечення ефективності обчислень та компактності представлення даних. У цьому розділі я розпишу основні алгебраїчні об'єкти, що застосовуються в алгоритмі TiGER.

3.1.1 Кільця та многочлени

Означення 3.1.1 (Кільце)**.**

Кільце $(R, +, \cdot)$ *– це множина з двома операціями: додавання* $(+)$ *та множення* $(\cdot)$*, що задовольняє наступні властивості:*

1. $(R, +)$ *є абелевою групою за додаванням. Нейтральний елемент* 0*;*
2. *Множення є асоціативним:* $(a \cdot b) \cdot c = a \cdot (b \cdot c)$*,* $\forall\, a, b, c \in R$*;*
3. *Дистрибутивність:* $a \cdot (b + c) = a \cdot b + a \cdot c$ *та* $(b + c) \cdot a = b \cdot a + c \cdot a$*,* $\forall\, a, b, c \in R$*;*
4. *Існує нейтральний елемент за множенням:* $1 \in R$ *такий, що* $1 \cdot a = a \cdot 1 = a$*,* $\forall\, a \in R$*.*

Якщо ще $a \cdot b = b \cdot a$*,* $\forall\, a, b \in R$*, то таке кільце називається комутативним.*

Означення 3.1.2 (Кільце многочленів)**.**

Нехай R *– комутативне кільце з одиницею. Кільце многочленів* $R[x]$ *складається з усіх виразів виду*

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0,$$

де $\forall\, i : a_i \in \mathbb{R}$*,* $a_n \neq 0$ *та* $n \in \mathbb{Z}$*.*

Число n *називається степенем многочлена* $f(x)$*, позначається* $\deg(f)$*.*

Операції додавання та множення многочленів наступним чином:

- $(f + g)(x) = \sum\limits_{i=0}^{\max(\deg(f),\deg(g))} (a_i + b_i) x^i$, де a_i, b_i – коефіцієнти f та g відповідно;
- $(f \cdot g)(x) = \sum\limits_{k=0}^{\deg(f)+\deg(g)} c_k x^k$, де $c_k = \sum\limits_{i=0}^{k} a_i b_{k-i}$.

3.1.2 Факторкільця многочленів

Означення 3.1.3 (Факторкільце)**.**

Нехай R – кільце та I – його ідеал. Факторкільце R/I складається з класів еквівалентності $a + I = \{a + r : r \in I\}$ для $a \in R$, з операціями:

$$(a + I) + (b + I) = (a + b) + I, \quad (a + I) \cdot (b + I) = (a \cdot b) + I.$$

У Lattice-based cryptography найчастіше використовується факторкільце многочленів за ідеалом, що породжений циклотомічним многочленом.

Означення 3.1.4 (Циклотомічний многочлен)**.**

n-ий циклотомічний многочлен (Cyclotomic polynomial) $\Phi_n(x)$ визначається як мінімальний многочлен над $\mathbb{Q}$, коренями якого є примітивні корені n-го степеня з одиниці:

$$\Phi_n(x) = \prod_{\substack{1 \le k \le n \\ \gcd(k,n)=1}} \left(x - e^{2\pi i k/n}\right).$$

Лема 3.1.1. *Для $n = 2^k$, де $k \in \mathbb{N}$, циклотомічний многочлен має вигляд:*

$$\Phi_n(x) = x^{n/2} + 1.$$

Доведення: При $n = 2^k$ примітивними коренями n-го степеня з одиниці є $e^{2\pi i m/n}$ для непарних m. З $(x^{n/2} + 1) = (x^n - 1)/(x^{n/2} - 1)$ випливає твердження леми. □

3.1.3 Кільце многочленів у TiGER

В алгоритмі TiGER використовується кільце многочленів виду:

$$R_q = \frac{\mathbb{Z}_q[x]}{(x^n + 1)},$$

де n – степінь двійки (зазвичай $n = 512$ або $n = 1024$), а q – модуль, що також є степенем двійки ($q = 256$ у всіх варіаціях алгоритму TiGER).

Елементами R_q є многочлени степеня не вище $n - 1$ з коефіцієнтами з $\mathbb{Z}_q$:

$$f(x) = \sum_{i=0}^{n-1} a_i x^i, \quad a_i \in \mathbb{Z}_q.$$

Вибір саме такого n та многочлена $x^n + 1$ забезпечує:

- Ефективність множення многочленів (без потреби в NTT);
- Редукцію за модулем $x^n + 1$, що спрощує обчислення;
- Зв'язок з циклотомічними многочленами та решітковими задачами.

3.1.4 Операції в кільці R_q

Для $f(x), g(x) \in R_q$ операції в кільці визначаються наступним чином:

Додавання: покоефіцієнтне за модулем q:

$$(f + g)(x) = \sum_{i=0}^{n-1} ((f_i + g_i) \bmod q) x^i.$$

Множення: спочатку виконується звичайне множення многочленів, потім взяття за модулем $(x^n + 1)$ та q. Оскільки $x^n \equiv -1 \pmod{x^n + 1}$, то маємо:

$$h_i = \left(\sum_{j=0}^{i} f_j g_{i-j} - \sum_{j=i+1}^{n-1} f_j g_{n+i-j} \right) \bmod q,$$

де $h(x) = (f \cdot g)(x) = \sum_{i=0}^{n-1} h_i x^i$.

3.2 Задачі на решітках

Безпека TiGER базується на обчислювальній складності певних задач на решітках. У цьому підпункті зазначимо основні решіткові задачі, що лежать в основі постквантової криптографії.

3.2.1 Решітки та базові задачі

Означення 3.2.1 (Решітка)**.**
Нехай $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_m \in \mathbb{R}^n$ – лінійно незалежні вектори. Решітка Λ, породжена цими векторами, визначається як:

$$\Lambda = \Lambda(\mathbf{b}_1, \dots, \mathbf{b}_m) = \left\{ \sum_{i=1}^{m} a_i \mathbf{b}_i : a_i \in \mathbb{Z} \right\}.$$

Вектори $\mathbf{b}_1, \dots, \mathbf{b}_m$ називаються базисом решітки, а m – розмірністю решітки.

Означення 3.2.2 (Мінімальна відстань решітки(shortest vector))**.**
Мінімальна відстань решітки Λ визначається як:

$$\lambda_1(\Lambda) = \min_{\mathbf{v} \in \Lambda \setminus \{\mathbf{0}\}} \|\mathbf{v}\|,$$

де $\|\cdot\|$ – евклідова норма.

Означення 3.2.3 (SVP)**.**
Shortest Vector Problem (SVP): Для заданого базису решітки Λ знайти ненульовий вектор $\mathbf{v} \in \Lambda$ такий, що $\|\mathbf{v}\| = \lambda_1(\Lambda)$.

SVP є NP-складною задачею [13]. Для криптографічних цілей часто використовується наступна версія:

Означення 3.2.4 (Апроксимаційна задача SVP)**.**
γ-approximate SVP (γ-SVP): Для заданого базису решітки Λ та параметра наближення $\gamma \geq 1$, знайти ненульовий вектор $\mathbf{v} \in \Lambda$ такий, що $\|\mathbf{v}\| \leq \gamma \cdot \lambda_1(\Lambda)$.

3.2.2 Задача Learning With Errors (LWE)

Задача Learning With Errors була введена Одедом Регєвим у 2005 році [3] і стала основою для багатьох постквантових криптосистем.

Означення 3.2.5 (LWE задача (search version))**.**

Нехай $n, q \geq 1$ – цілі числа, χ – розподіл ймовірностей на $\mathbb{Z}_q$. Пошукова задача $LWE_{n,q,\chi}$ визначається наступним чином:

Для невідомого секрету $\mathbf{s} \in \mathbb{Z}_q^n$ та заданої послідовності пар $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, де

$$b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \pmod{q},$$

з випадково обраними $\mathbf{a}_i \in \mathbb{Z}_q^n$, $\langle \cdot, \cdot \rangle$ – операція векторного добутку та $e_i \xleftarrow{p} \chi$, знайти секрет $\mathbf{s}$.

Означення 3.2.6 (LWE задача (detection version))**.**

Розпізнавальна задача Decision-$LWE_{n,q,\chi}$ полягає у розрізненні наступних двох розподілів:

- *$(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e \bmod q)$, де $\mathbf{a} \leftarrow \mathbb{Z}_q^n$, $\mathbf{s} \in \mathbb{Z}_q^n$ фіксоване, $e \xleftarrow{p} \chi$;*
- *$(\mathbf{a}, u)$, де $\mathbf{a}, u$ – рівноймовірно розподілені на $\mathbb{Z}_q^n$ та $\mathbb{Z}_q$ відповідно.*

Теорема 3.2.1 (Регєва)**.**

Для певних параметрів n, q, χ, розв'язання задачі Decision-LWE за поліноміальний час у середньому випадку еквівалентно розв'язанню наближеної задачі γ-SVP за квантовий поліноміальний час у найгіршому випадку для деякого $\gamma = \tilde{O}(n/\sigma)$.

3.2.3 Задача Ring Learning With Errors (RLWE)

Ring-LWE є алгебраїчною версією LWE, що вже використовує кільця многочленів для більшої ефективності [4].

Означення 3.2.7 (RLWE задача)**.**

Нехай $R = \mathbb{Z}[x]/(x^n + 1)$, $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, та χ – розподіл ймовірностей на R_q. Розпізнавальна задача Decision-$RLWE_{n,q,\chi}$ полягає у розрізненні наступних розподілів:

- *$(a, a \cdot s + e)$, де обрано $a \in R_q$ (рівномірний розподіл), $s \in R_q$ фіксоване, $e \xleftarrow{p} \chi$;*
- *(a, u), де a, u обрані рівномірно з R_q.*

Важливим є те, що RLWE дозволяє представляти n секретів (LWE-зразків) у вигляді одного многочлена в R_q, що значно зменшує розмір ключів та шифротекстів. Для TiGER використовується $n \in \{512, 1024\}$ та $q = 256$.

3.2.4 Задача Learning With Rounding (LWR)

Learning With Rounding є детермінованим варіантом LWE, де замість додавання випадкової помилки використовується округлення [5].

Означення 3.2.8 (LWR задача)**.**

Нехай n, q, p – цілі числа, $p < q$. Визначимо функцію округлення $\lfloor \cdot \rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$ як

$$\lfloor x \rceil_p = \left\lfloor \frac{p}{q} \cdot x \right\rceil \bmod p,$$

де $\lfloor \cdot \rceil$ позначатиме округлення до найближчого цілого.

Розпізнавальна задача Decision-$LWR_{n,q,p}$ полягає у розрізненні наступних розподілів:

- $(\mathbf{a}, \lfloor \langle \mathbf{a}, \mathbf{s} \rangle \rceil_p)$, *де* $\mathbf{a} \leftarrow \mathbb{Z}_q^n$, $\mathbf{s} \in \mathbb{Z}_q^n$ *– фіксоване;*
- $(\mathbf{a}, u)$, *де* $\mathbf{a} \leftarrow \mathbb{Z}_q^n$, $u \leftarrow \mathbb{Z}_p$.

Теорема 3.2.2 (Редукція (взяття за модулем) LWR до LWE [5])**.** *Для відповідних параметрів n, q, p та достатньо малого розподілу помилок χ, задача Decision-LWR$_{n,q,p}$ зводиться до задачі Decision-LWE$_{n,q,\chi}$.*

Зауваження. (Ідея теореми) При достатньо великому відношенні q/p, округлення $\lfloor \langle \mathbf{a}, \mathbf{s} \rangle \rceil_p$ стає еквівалентно до додавання малої помилки округлення, яка розподілена майже рівномірно на інтервалі $(-q/(2p), q/(2p)]$.

3.2.5 Задача Ring Learning With Rounding (RLWR)

RLWR поєднує переваги RLWE (компактність – за рахунок алгебраїчної структури) та LWR (детермінованість, та відсутність потреби у відборі помилок(sampling)).

Означення 3.2.9 (RLWR задача)**.**
Нехай $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, $R_p = \mathbb{Z}_p[x]/(x^n + 1)$, де $p < q$. Функція округлення застосовуються для кожного коефіцієнта окремо:

$$\lfloor f \rceil_p = \sum_{i=0}^{n-1} \left\lfloor \frac{p}{q} \cdot f_i \right\rceil x^i \bmod p.$$

Розпізнавальна задача Decision-RLWR$_{n,q,p}$ полягає у розрізненні:

- $(a, \lfloor a \cdot s \rceil_p)$, *де* $a \in R_q$ *обрано рівномірно,* $s \in R_q$ *фіксоване;*
- (a, u), *де* $a \in R_q$, $u \in R_p$ *обрані рівномірно.*

Твердження 3.2.1.
При певних параметрах n, q, p, задача RLWR$_{n,q,p}$ є не легшою за задачу RLWE$_{n,q,\chi}$ з розподілом помилок χ, що відповідає помилці округлення.

У TiGER використовується комбінація: RLWR – для генерації відкритого ключа (компактність) та RLWE – для шифрування (гнучкості у контролі помилок). Модулі обираються як степені двійки: $q = 256$, $p \in \{64, 128\}$, що дозволяє реалізувати округлення через побітові зсуви. Про це поговоримо детальніше наступному, 4 розділі.

3.3 ”Сімейство” алгоритмів на базі RLWE/RLWR

TiGER належить до сімейства алгоритмів на решітках, які базуються на задачах RLWE та RLWR. Опишемо основні алгоритми цього сімейства, які вплинули на дизайн TiGER (далі). Далі їх буде порівняно у розділі 6 та попередньо у таблиці 3.1.

3.3.1 Lizard

Lizard [6] був одним з перших алгоритмів, що комбінував у собі LWE та LWR для досягнення балансу між безпекою та ефективністю.

Основні характеристики:

- **Структура:** LWE для генерації відкритого ключа, LWR для шифрування;

- **Простір:** Цілочисельні решітки над $\mathbb{Z}_q^n$ без використання кільцевої структури;
- **Модуль:** Малий модуль q для покращення коректності;
- **Розміри:** Великі ключі (через відсутність алгебраїчної структури).

Переваги:

- Консервативна безпека (базується на стандартній LWE);
- Низька ймовірність помилки дешифрування.

Недоліки:

- Великі розміри ключів та шифротекстів;
- Повільніше множення векторів (порівняно з кільцевими варіантами).

3.3.2 RLizard

RLizard [7] є кільцевою версією попереднього алгоритму, оптимізованою для IoT-пристроїв.

Основні характеристики:

- **Структура:** RLWE для генерації ключів, RLWR для шифрування;
- **Простір:** $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, $n = 512$ або $n = 1024$;
- **Модуль:** Малий модуль q (наприклад, $q = 1024$);
- **Помилки:** Дискретний гаусівський розподіл помилок з високою точністю (CDT-sampling).

Переваги:

- Компактні ключі та шифротексти завдяки кільцевій структурі;
- Швидке шифрування/дешифрування;
- Висока коректність;
- Підходить для пристроїв з обмеженим ресурсом.

Недоліки:

- Відносно велика пропускна здатність порівняно з найкомпактнішими схемами;
- Залежність від якісного(справді випадкового) семплювання гаусівських помилок.

3.3.3 CRYSTALS-Kyber

Kyber [14] є одним з фіналістів конкурсу NIST PQC і базується на Module-LWE (MLWE).

Основні характеристики:

- **Структура:** MLWE для обох операцій – генерації ключів та шифрування повідомлення;
- **Простір:** Модульні решітки R_q^k, де $k \in \{2, 3, 4\}$ (залежно від рівня безпеки);

- **Модуль:** $q = 3329$ (просте число для ефективного NTT);
- **Оптимізація:** Number Theoretic Transform (NTT) для швидкого множення многочленів (хах, тут порівнюючи з чим саме і що вважати швидко);
- **Компресія:** Агресивне ”стиснення” шифротексту.

Переваги:

- Стандартизовано під NIST (FIPS 203);
- Непоганий баланс між розмірами, швидкістю та безпекою;
- Ефективна реалізація з NTT.

Недоліки:

- Використання простого модуля ускладнює деякі оптимізації;
- Більший шифротекст порівняно з деякими MLWR-схемами.

3.3.4 Saber

Saber [15] є Module-LWR схемою, фіналістом NIST PQC Round 3.

Основні характеристики:

- **Структура:** MLWR для обох операцій;
- **Простір:** Модульні решітки R_q^k, $k \in \{2, 3, 4\}$;
- **Модуль:** $q = 8192 = 2^{13}$;
- **Оптимізація:** Відсутність NTT (округлення завдяки побітовим операціям);
- **Помилки:** Детерміністичні (через застосування округлення).

Переваги:

- Компактний шифротекст;
- Простота реалізації (без потреби в NTT чи гаусівському семплюванні);
- Ефективні побітові операції;
- Хороша стійкість до side-channel атак.

Недоліки:

- Більший(за розміром) відкритий ключ порівняно з Kyber;

3.3.5 Порівняння алгоритмів, і що з них взяв TiGER

Характеристика	**Lizard**	**RLizard**	**Kyber**	**Saber**
Структура	LWE/LWR	RLWE/RLWR	MLWE	MLWR
Розмірність	n	n	$n \times k$	$n \times k$
Модуль q	малий	малий	3329	8192
NTT	Ні	Ні	Так	Ні
Семплювання	Гаусс	Гаусс/CDT	Centered binomial	Округлення
Компресія	Помірна	Помірна	Агресивна	Помірна

Таблиця 3.1: Порівняння підходів у сімействі RLWE/RLWR алгоритмів

TiGER об'єднує в собі найкращі ідеї з попередніх доробок:

Позиція TiGER:

- Базується на **RLWE/RLWR** (як RLizard);
- Використовує **степені двійки** для q, p (як Saber);
- **Розріджені секрети** для ефективності;
- **Корекція помилок** (XEf, D2) для мінімізації DFP/R;
- **Без NTT** для простоти;
- Фокусування на **мінімальному шифротексті** для легшого впровадження в існуючі протоколи.

3.4 Криптографічні примітиви

У цій частині розділу розглянемо базові криптографічні примітиви, що використовуються для безпосередньо при побудові TiGER: схеми шифрування з відкритим ключем (PKE) та механізми інкапсуляції ключа (KEM).

3.4.1 Схема шифрування з відкритим ключем (PKE)

Означення 3.4.1 (PKE схема)**.**

Схема шифрування з відкритим ключем (Public Key Encryption, PKE) складається з трьох алгоритмів:

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$ — *ймовірнісний алгоритм генерації ключів, що на вході приймає параметр безпеки λ і повертає пару: відкритий ключ* pk *та секретний ключ* sk*;*
- $\mathsf{Enc}(\mathsf{pk}, m; r) \to c$ — *ймовірнісний алгоритм шифрування, що приймає відкритий ключ* pk*, повідомлення m з простору повідомлень $\mathcal{M}$ та випадкове число r, і повертає шифротекст c;*

- $\mathsf{Dec}(\mathsf{sk}, c) \to m'$ — *детермінінований алгоритм дешифрування, що приймає секретний ключ* sk *і шифротекст* c*, та повертає повідомлення* m' *або помилку –* $\perp$*.*

Означення 3.4.2 (Коректність PKE)**.**

PKE схема є $(1-\delta)$*-коректною, якщо для будь-якої пари ключів* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *та будь-якого повідомлення* $m \in \mathcal{M}$ *виконується:*

$$\mathbb{P}[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) \neq m] \leq \delta,$$

де ймовірність визначається через випадковість алгоритму Enc*. Параметр* δ *називається ймовірністю помилки дешифрування (Decryption Failure Probability/Rate, DFP/R)*

У задачах на решітках, через наявність помилок у RLWE/RLWR, коректність не завжди є ідеальною. Тому для практичних застосувань необхідно, щоб δ було незначним ($\delta \leq 2^{-128}$).

3.4.2 Механізм інкапсуляції ключа (KEM)

Означення 3.4.3 (KEM схема)**.**

Механізм інкапсуляції ключа (Key Encapsulation Mechanism, KEM) складається з трьох кроків:

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$ — *алгоритм генерації ключів (аналогічно до PKE);*
- $\mathsf{Encaps}(\mathsf{pk}) \to (c, K)$ — *ймовірнісний алгоритм інкапсуляції, що приймає відкритий ключ* pk *і повертає шифротекст* c *та спільний секретний ключ* $K \in \mathcal{K}$*;*
- $\mathsf{Decaps}(\mathsf{sk}, c) \to K'$ — *детермінований алгоритм декапсуляції, що приймає секретний ключ* sk *і шифротекст* c*, та повертає прихований секретний ключ* K' *або символ помилки* $\perp$*.*

Означення 3.4.4 (Коректність KEM)**.**

KEM схема є $(1-\delta)$*-коректною, якщо для будь-якої пари ключів* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *справджується таке:*

$$\mathbb{P}[\mathsf{Decaps}(\mathsf{sk}, c) \neq K : (c, K) \leftarrow \mathsf{Encaps}(\mathsf{pk})] \leq \delta.$$

3.4.3 Побудова KEM з PKE

Стандартний спосіб побудови KEM з PKE полягає у шифруванні випадкового повідомлення та використанні геш-функції для отримання спільного ключа.

Твердження 3.4.1 (Нативна побудова KEM)**.**

Нехай $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *– PKE це ось така трійка, що містить у собі простір повідомлень* $\mathcal{M}$*, та* $H : \mathcal{M} \to \mathcal{K}$ *– деяка геш-функція. Тоді можна побудувати KEM таким способом:*

- *Спосіб генерації* KeyGen *такий же, як у PKE;*
- $\mathsf{Encaps}(\mathsf{pk})$*: обирають* $m \in \mathcal{M}$*, а далі обчислюють* $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ *та* $K \leftarrow H(m)$*. На виході отримано пару:* (c, K)*;*

- $\mathsf{Decaps}(\mathsf{sk}, c)$: *обчислюють* $m' \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$ *та на виході отримуємо* $K' \leftarrow H(m')$.

Така побудова не забезпечує IND-CCA безпеки навіть якщо базова PKE схема є IND-CPA безпечною, але для нас головне щоб було зрозуміло як цей механізм працює). А для досягнення IND-CCA безпеки вже необхідно застосувати перетворення Fujisaki-Okamoto (розглянемо його у секції 3.6).

3.4.4 Криптографічні геш-функції

Означення 3.4.5 (Криптографічна геш-функція)**.**

Функція $H : \{0,1\}^* \to \{0,1\}^n$ *називається криптографічною геш-функцією, якщо вона задовольняє наступні умови:*

1. ***Стійкість до знаходження прообразу:*** *Для випадкового* $y \in \{0,1\}^n$ *обчислювально важко знайти* x *такий, що* $H(x) = y$*;*
2. ***Стійкість до знаходження другого прообразу:*** *Для заданого* x *обчислювально важко знайти* $x' \neq x$ *такий, що* $H(x') = H(x)$*;*
3. ***Стійкість до колізій:*** *Обчислювально важко знайти дві різні величини* x, x' *такі, що* $H(x) = H(x')$.

У TiGER використовуються геш-функції з сімейства SHA-3 – SHAKE256 (або SHA3-256) для генерації випадковості, обчислення спільних ключів та інших криптографічних операцій. SHAKE256 є функцією з розширеним виходом (XOF), що дозволяє генерувати вихід довільної довжини.

3.4.5 Зв'язок PKE та KEM у TiGER

TiGER складається з двох рівнів:

1. **TiGER.PKE** – являє собою базову IND-CPA безпечну схему шифрування, що базується на RLWE(R);
2. **TiGER.KEM** – KEM, отриманий застосуванням перетворення Fujisaki-Okamoto (FO) до TiGER.PKE для досягнення IND-CCA безпеки.

Це дозволяє:

- Окремо аналізувати безпеку PKE (на базі RLWE/RLWR);
- Використовувати загальні результати про перетворення FO для доведення IND-CCA безпеки KEM;

3.5 Основні поняття безпеки, що стосуються TiGER

3.5.1 IND-CPA безпека

Означення 3.5.1 (IND-CPA (Indistinguishability under Chosen-Plaintext Attack) гра для PKE)**.**

Розглянемо наступну "гру" між претендентом (challenger) $\mathcal{C}$ *та супротивником (adversary)* $\mathcal{A}$*:*

1. *$\mathcal{C}$ генерує* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *і передає* pk *супротивнику* $\mathcal{A}$*;*

2. *$\mathcal{A}$ обирає два повідомлення* $m_0, m_1 \in \mathcal{M}$ *однакової довжини і передає їх* $\mathcal{C}$*;*

3. *$\mathcal{C}$ Випадковим чином обирає біт* $b \xleftarrow{p} \{0,1\}$*, обчислює* $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$ *і передає* c *супротивнику* $\mathcal{A}$*;*

4. *$\mathcal{A}$ виводить біт* b'*.*

Перевага супротивника визначається наступним чином:

$$\mathsf{Adv}_{\mathsf{PKE}}^{\textit{IND-CPA}}(\mathcal{A}) = \left|\mathbb{P}[b' = b] - \frac{1}{2}\right| + \varepsilon(\lambda).$$

PKE схема є IND-CPA безпечною, якщо для будь-якого ефективного (PPT) супротивника $\mathcal{A}$ *перевага* $\mathsf{Adv}_{\mathsf{PKE}}^{\textit{IND-CPA}}(\mathcal{A})$ *є незначною функцією від* λ*.*

IND-CPA безпечність означає нерозрізненість шифротекстів: супротивник маючи доступ до відкритого ключа (а отже, має можливість шифрувати будь-які повідомлення), не може визначити, яке з двох повідомлень було зашифроване. Це вимагає від шифрування, щоб воно було ймовірнісним.

Означення 3.5.2 (IND-CPA (Indistinguishability under Chosen-Plaintext Attack) безпека для KEM)**.**

Для KEM "гра" IND-CPA визначається аналогічно:

1. *$\mathcal{C}$ генерує пару* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *і передає* pk *супротивнику* $\mathcal{A}$*;*

2. *$\mathcal{C}$ обирає* $b \xleftarrow{p} \{0,1\}$*. Якщо обрано* $b = 0$*, то обчислює* $(c, K_0) \leftarrow \mathsf{Encaps}(\mathsf{pk})$*; а якщо* $b = 1$*, обчислює* $(c, K_0) \leftarrow \mathsf{Encaps}(\mathsf{pk})$ *та* $K_1 \xleftarrow{p} \mathcal{K}$*. Опісля передає пару* (c, K_b) *супротивнику;*

3. *$\mathcal{A}$ виводить біт* b'*.*

Перевага супротивника $\mathsf{Adv}_{\mathsf{KEM}}^{\textit{IND-CPA}}(\mathcal{A})$ *визначається аналогічно як і в PKE.*

Безпека KEM тісно пов'язана з:

- IND-CPA безпекою базової PKE схеми;
- Ймовірністю помилки дешифрування ε;
- Параметрами випадкових оракулів (про трохи далі).

Важливим є те, що навіть за наявності помилок дешифрування (що неминуче для схем на решітках), можна довести IND-CCA безпеку за умови достатньо малого $\varepsilon(\lambda)$.

3.5.2 IND-CCA безпека

Означення 3.5.3 (IND-CCA для PKE)**.**

"Гра" IND-CCA відрізняється від IND-CPA тим, що противник $\mathcal{A}$ *має додатково доступ до оракула дешифрування (decryption oracle)* $\mathsf{Dec}(\mathsf{sk}, \cdot)$*:*

1. *$\mathcal{C}$ генерує* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *і передає* pk *супротивнику* $\mathcal{A}$*;*

2. *$\mathcal{A}$ робить запити до оракула дешифрування, подаючи на вхід довільні шифротексти c_i і отримувати $m_i = \mathsf{Dec}(\mathsf{sk}, c_i)$;*

3. *$\mathcal{A}$ обирає m_0, m_1 і отримує challenge шифротекст $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$ для випадкового b;*

4. *$\mathcal{A}$ продовжує робити запити до оракула дешифрування, але не може запитувати c^* – випадок **IND-CCA2** (а якщо запити заборонені вже після отримання c^* – це **IND-CCA1**);*

5. *$\mathcal{A}$ виводить біт b'.*

Перевага $\mathsf{Adv}^{IND\text{-}CCA}_{\mathsf{PKE}}(\mathcal{A})$ визначається аналогічно.

Означення 3.5.4 (IND-CCA безпека для KEM)**.**
*Для KEM схеми противник має доступ до оракула декапсуляції $\mathsf{Decaps}(\mathsf{sk}, \cdot)$ і **не може** запитувати challenge шифротекст c^* після його отримання.*

IND-CCA (Indistinguishability under Chosen Ciphertext Attack) безпека є значно сильнішою, ніж IND-CPA, оскільки моделює активного противника, який може маніпулювати шифротекстами та спостерігати результати дешифрування. Для практичних застосувань зазвичай потрібна IND-CCA2 безпека.

3.5.3 Ймовірність помилки дешифрування (DFP/R)

Означення 3.5.5 (Decryption Failure Probability/Rate)**.**
Для PKE схеми ймовірність помилки дешифрування (DFP/R) визначається як:

$$\delta = \max_{m \in \mathcal{M}} \mathbb{P}_{(\mathsf{pk},\mathsf{sk}),r}[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m; r)) \neq m],$$

де ймовірність береться за випадковістю генерації ключів та шифрування.
Для KEM схеми:

$$\delta = \mathbb{P}_{(\mathsf{pk},\mathsf{sk}),(c,K)}[\mathsf{Decaps}(\mathsf{sk}, c) \neq K].$$

У решіткових схемах помилки дешифрування виникають природнім шляхом (наявність шуму у RLWE/RLWR), тому параметри схеми (розміри модулів, розподіл помилок, коефіцієнт стиснення) мають бути підібрані так, щоб забезпечити мале δ.
(!) Висока ймовірність помилки дешифрування може призвести до наступних атак:

- **Failure boosting attacks [16]:** Супротивник може ітеративно створювати шифротексти, що мають високу ймовірність помилки, щоб витягувати поступово інформацію про секретний ключ;
- **Multi-target attacks [17]:** При наяності багатьох публічних ключів або сесій, навіть відносно мала DFP/R може стати проблемою.

Для реалізацій TiGER цільова DFP/R становить:

- TiGER128: $\delta \approx 2^{-120}$;
- TiGER192: $\delta \approx 2^{-136}$;
- TiGER256: $\delta \approx 2^{-167}$.

P.S. Ці значення вважаються достатньо малими для практичного застосування.

3.5.4 Квантова безпека a.k.a. QROM

Означення 3.5.6 (Quantum Random Oracle Model)**.**

Модель квантового випадкового оракула (скорочено QROM) – це розширення класичної моделі випадкового оракула (ROM), де противник має квантовий доступ до геш-функцій, тобто може ще робити запити у суперпозиції.

Для TiGER необхідно, щоб перетворення Fujisaki-Okamoto забезпечувало IND-CCA безпеку у QROM, це гарантуватиме стійкість проти квантових атак.

3.6 Перетворення Fujisaki-Okamoto

Перетворення Fujisaki-Okamoto (скорочено FO) [9, 10] є загальним методом перетворення IND-CPA безпечної PKE схеми у IND-CCA безпечну KEM схему. В цьому підпункті наведемо просте класичне FO перетворення та його модифікацію – варіант з неявним відхиленням, що використовується в TiGER.

3.6.1 Класичне перетворення FO

Теорема 3.6.1 (Fujisaki-Okamoto)**.**

Нехай $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *– IND-CPA безпечна PKE схема з однозначним детермінованим дешифруванням. Нехай* $G : \mathcal{M} \to \mathcal{R} \times \mathcal{K}$ *та* $H : \mathcal{M} \times \mathcal{C} \to \mathcal{K}$ *– випадкові оракули. Тоді наступна конструкція є IND-CCA безпечною KEM у моделі випадкового оракула [9]:*

Algorithm 1 KeyGen()

Generate $(pk, sk) \leftarrow \text{PKE.KeyGen}(1^\lambda)$
return (pk, sk)

Algorithm 2 Encaps(pk)

Choose $m \xleftarrow{p} \mathcal{M}$
Calculate $(r, K) \leftarrow G(m)$
Calculate $c \leftarrow \text{PKE.Enc}(pk, m; r)$
return (c, K)

Algorithm 3 Decaps(sk, c)

Calculate $m' \leftarrow \text{PKE.Dec}(sk, c)$
Calculate $(r', K') \leftarrow G(m')$
Calculate $c' \leftarrow \text{PKE.Enc}(pk, m'; r')$
if $c = c'$ **then**
    return K'
else
    return $\perp$
end if

Ключова ідея перетворення FO полягає у ***повторному шифруванні*** (re-encryption): після дешифрування повідомлення m' воно повторно шифрується з тією ж випадковістю, і результат порівнюється з отриманим шифротекстом. Це дозволяє виявити модифікації шифротексту.

3.6.2 Перетворення $\mathsf{FO}_m^{\not\perp}$ з неявним відхиленням

Для схем на решітках з ненульовою ймовірністю помилки дешифрування було розроблено модифікацію – варіант FO з *неявним відхиленням* (implicit rejection) [18].

Означення 3.6.1 (Перетворення $\mathsf{FO}_m^{\not\perp}$)**.**

Нехай PKE *– IND-CPA безпечна PKE схема. Конструкція* $\mathsf{FO}_m^{\not\perp}$ *відрізняється від класичного FO у додатковій декапсуляції:*

Algorithm 4 KeyGen()

1: Generate $(pk, sk') \leftarrow \text{PKE.KeyGen}(1^\lambda)$
2: Choose $z \xleftarrow{p} \mathcal{Z}$ ▷ Додатковий випадковий ключ
3: **return** $(pk, sk = (sk', z))$

Algorithm 5 Encaps(pk)

1: Choose $m \xleftarrow{p} \mathcal{M}$
2: Calculate $r \leftarrow G(m, pk)$
3: Calculate $c \leftarrow \text{PKE.Enc}(pk, m; r)$
4: Calculate $K \leftarrow H(m, c)$
5: **return** (c, K)

Algorithm 6 Decaps(sk, c)

1: Split $sk = (sk', z)$
2: Calculate $m' \leftarrow \text{PKE.Dec}(sk', c)$
3: Calculate $r' \leftarrow G(m', pk)$
4: Calculate $c' \leftarrow \text{PKE.Enc}(pk, m'; r')$
5: **if** $c = c'$ **then**
6: **return** $K' \leftarrow H(m', c)$
7: **else**
8: **return** $\bar{K} \leftarrow H(z, c)$ ▷ Неявне відхилення
9: **end if**

Ключова відмінність: замість повернення $\perp$ при невдалій перевірці, алгоритм повертає псевдовипадковий ключ $\bar{K} = H(z, c)$, де z – секретний випадковий ключ. Це має дві переваги:

- **Захист від side-channel атак:** Оскільки зовнішньому спостерігачу стає важче визначити, чи відбулася помилка при декапсуляції;
- **Постійний час виконання:** Обидва варіанти (успіх/невдача) тепер виконують однакові операції гешування.

3.6.3 Застосування у алгоритмі TiGER

TiGER.KEM побудовано із застосуванням варіанту перетворення $\mathsf{FO}_m^{\not\perp}$ до TiGER.PKE:

- **TiGER.PKE:** Базується на RLWR (для відкритого ключа) та RLWE (для шифрування), забезпечує IND-CPA безпеку;
- **геш-функції:** Використовуються $G = H = \mathsf{SHAKE256}$ (функція з розширеним виходом) для генерації випадковості та спільних ключів;
- **Додаткове гешування:** Відкритий ключ pk гешується разом з повідомленням: $r \leftarrow G(m, H(\mathsf{pk}))$, що забезпечує захист від multi-target атак;
- **Неявне відхилення:** При невдалій декапсуляції повертається $\bar{K} = H(z, c)$, що захищає від витоку інформації про помилки та розкритті хоч і мізерної, та інформації, про ключ.

Розділ 4

Повний опис алгоритму TiGER

Розділ 5

Результати досліджень

Розділ 6

Аналіз атак на TiGER

Розділ 7

Порівняльний аналіз

Розділ 8

Перенесення атак та можливі покращення

Bibliography

[1] Peter W. Shor. «Algorithms for quantum computation: discrete logarithms and factoring». In: *Proceedings 35th Annual Symposium on Foundations of Computer Science* (1994), pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[2] Wikipedia. *Post-quantum cryptography — Wikipedia, The Free Encyclopedia*. [Online; accessed 7-October-2025]. URL: https://en.wikipedia.org/wiki/Post-quantum_cryptography.

[3] Oded Regev. «On lattices, learning with errors, random linear codes, and cryptography». In: *Journal of the ACM* 56.6 (2009), pp. 1–40. DOI: 10.1145/1568318.1568324.

[4] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. «On Ideal Lattices and Learning with Errors over Rings». In: *Advances in Cryptology – EUROCRYPT 2010*. 2010, pp. 1–23. DOI: 10.1007/978-3-642-13190-5_1.

[5] Abhishek Banerjee, Chris Peikert, and Alon Rosen. «Pseudorandom Functions and Lattices». In: *Advances in Cryptology – EUROCRYPT 2012*. 2012, pp. 719–737. DOI: 10.1007/978-3-642-29011-4_42.

[6] Jung Hee Cheon et al. «Lizard: Cut off the Tail! A Practical Post-Quantum Public-Key Encryption from LWE and LWR». In: *Security and Cryptography for Networks – SCN 2018*. 2018, pp. 160–177. DOI: 10.1007/978-3-319-98113-0_9.

[7] Joohee Lee et al. «RLizard: Post-quantum Key Encapsulation Mechanism for IoT Devices». In: *IEEE Access* 7 (2019), pp. 2080–2091. DOI: 10.1109/ACCESS.2018.2886964.

[8] Seunghwan Park et al. *TiGER: Tiny bandwidth key encapsulation mechanism for easy miGration based on RLWE(R)*. Cryptology ePrint Archive, Paper 2022/1651. https://eprint.iacr.org/2022/1651. 2022.

[9] Eiichiro Fujisaki and Tatsuaki Okamoto. «Secure Integration of Asymmetric and Symmetric Encryption Schemes». In: *Advances in Cryptology – CRYPTO '99*. 1999, pp. 537–554. DOI: 10.1007/3-540-48405-1_34.

[10] Eiichiro Fujisaki and Tatsuaki Okamoto. «Secure Integration of Asymmetric and Symmetric Encryption Schemes». In: *Journal of Cryptology* 26.1 (2013), pp. 80–101. DOI: 10.1007/s00145-011-9114-1.

[11] KpqC Team. *Korean Post-Quantum Cryptography Competition*. https://www.kpqc.or.kr. 2023.

[12] Jung Hee Cheon et al. *SMAUG-T: the Key Exchange Algorithm based on Module-LWE and Module-LWR*. KpqC Round 2 Submission. Version 3.0. 2024.

[13] Miklós Ajtai. «The shortest vector problem in L_2 is NP-hard for randomized reductions». In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. 1998, pp. 10–19. DOI: 10.1145/276698.276705.

[14] Joppe Bos et al. «CRYSTALS-Kyber: A CCA-Secure Module-Lattice-Based KEM». In: *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. 2018, pp. 353–367. DOI: 10.1109/EuroSP.2018.00032.

[15] Jan-Pieter D'Anvers et al. «Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM». In: *Progress in Cryptology – AFRICACRYPT 2018*. 2018, pp. 282–305. DOI: 10.1007/978-3-319-89339-6_16.

[16] Siemen Dhooghe and Svetla Nikova. «My Gadget Just Cares For Me - How NINA Can Prove Security Against Combined Attacks». In: *Progress in Cryptology – INDOCRYPT 2018*. 2018, pp. 35–55. DOI: 10.1007/978-3-030-05378-9_3.

[17] Jan-Pieter D'Anvers, Siemen Dhooghe, and Frederik Vercauteren. «On the Impact of Decryption Failures on the Security of NTRU Encryption». In: *Advances in Cryptology – ASIACRYPT 2019*. 2019, pp. 565–598. DOI: 10.1007/978-3-030-34621-8_20.

[18] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. «A Modular Analysis of the Fujisaki-Okamoto Transformation». In: *Theory of Cryptography – TCC 2017*. 2017, pp. 341–371. DOI: 10.1007/978-3-319-70500-2_12.

[19] Kathrin Hövelmanns et al. «Generic Authenticated Key Exchange in the Quantum Random Oracle Model». In: *Public-Key Cryptography – PKC 2020*. 2020, pp. 389–422. DOI: 10.1007/978-3-030-45388-6_14.