НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра математичних методів захисту інформації

«До захисту допущено»
В.о. завідувача кафедри
_____ Сергій ЯКОВЛЄВ
«__» _____ 2022 р.

Дипломна робота

на здобуття ступеня бакалавра

зі спеціальності: 113 Прикладна математика
на тему: **«Криптоаналіз алгоритму цифрового підпису «Вершина»»**

Виконав: студент 4 курсу, групи ФІ-84
Литвиненко Юлія Сергіївна

Керівник: к.ф.-м.н., ст. викладач Фесенко А.В. _____

Консультант: _ _____

Рецензент: к.т.н., доцент Стьопочкіна І.В _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Київ — 2022

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра математичних методів захисту інформації

Рівень вищої освіти — перший (бакалаврський)
Спеціальність (освітня програма) — 113 Прикладна математика,
ОПП «Математичні методи криптографічного захисту інформації»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Сергій ЯКОВЛЄВ
«__» _____ 2022 р.

ЗАВДАННЯ
на дипломну роботу

Студент: Литвиненко Юлія Сергіївна

1. Тема роботи: «*Криптоаналіз алгоритму цифрового підпису «Вершина»*»,

керівник: к.ф.-м.н., ст. викладач Фесенко А.В.,

затверджені наказом по університету №__ від «__» _____ 2022 р.

2. Термін подання студентом роботи: «__» _____ 2022 р.

3. Вихідні дані до роботи: опубліковані джерела за тематикою дослідження.

4. Зміст роботи:

1) аналіз наявних атак на алгоритми цифрового підпису у квантовій та класичній моделях обчислень;

2) аналіз проекту стандарту з алгоритмом «Вершина» та дослідження особливості роботи його алгоритму з різними режимами роботи;

3) аналіз стійкості алгоритму «Вершина» у квантовій та класичній моделях обчислень.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): Презентація доповіді

6. Дата видачі завдання: 10 вересня 2021 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником	01-15 вересня 2021 р.	Виконано
2	Огляд опублікованих джерел за тематикою дослідження	Вересень-жовтень 2021 р.	Виконано
3	Огляд наявних досліджень щодо особливостей та стійкості алгоритму Dilithium	Грудень 2021 р.	Виконано
4	Проведення аналізу проекту стандарту з алгоритмом «Вершина»	Лютий-березень 2022 р.	Виконано
5	Аналіз властивостей алгоритму «Вершина» для різних режимів роботи	Березень-квітень 2022 р.	Виконано
6	Отримання власних оцінок стійкості алгоритму «Вершина» до атаки підробки підпису без використання повідомлення у класичній та квантовій моделях обчислень	Квітень 2022 р.	Виконано
7	Отримання власних оцінок стійкості алгоритму «Вершина» до атаки відновлення ключа та екзистенційної підробки підпису у класичній та квантовій моделях обчислень	Травень 2022 р.	Виконано

Студент _____ Литвиненко Ю.С.

Керівник _____ Фесенко А.В.

РЕФЕРАТ

Кваліфікаційна робота містить: 54 стор., 0 рисунки, 6 таблиць, 36 джерел.

У роботі розглянуто алгоритм цифрового підпису CRYSTALS-Dilithium, який слугував прототипом для нового алгоритму цифрового підпису «Вершина». Також наведено особливості проекту національного стандарту цифрового підпису та побудови алгоритму «Вершина». У ході аналізу алгоритму обчислено очікувану кількість ітерацій, яку повинен виконати алгоритм для формування правильного підпису. Викладено основні теоретичні відомості про структуру Фіат-Шаміра з перериваннями та її стійкість у квантовій та класичній моделях обчислень. Отримані власні результати стійкості алгоритму «Вершина» до атаки підробки без використання повідомлення у класичній і квантовій моделях обчислень. Стйкість алгоритму «Вершина» до атаки відновлення ключа грунтується на припущенні складності задачі MLWE; а стійкість до екзистенційної підробки підпису грунтується на припущенні складності задачі MSIS. Відповідно у роботі обчислено очікуваний рівень складності розв'язку задач SIS та LWE, до яких існує зведення задач MSIS та MLWE.

Метою роботи є оцінка стійкості алгоритму цифрового підпису «Вершина» та аналіз складових алгоритмів проекту національного стандарту цифрового підпису та режимів його роботи.

Об'єктом дослідження є процеси перетворення інформації в алгоритмі цифрового підпису «Вершина».

Предметом дослідження є стійкість алгоритму цифрового підпису «Вершина» до криптоаналізу.

АЛГОРИТМ ЦИФРОВОГО ПІДПИСУ «ВЕРШИНА», ЗАДАЧІ НА РЕШІТКАХ, ОЦІНКИ СТІЙКОСТІ

ABSTRACT

The thesis contains: 54 pages, 0 figures, 6 tables, 36 sources.

In the thesis is reviewed the CRYSTALS-Dilithium digital signature algorithm, which was selected as a prototype for the new «Vershyna» digital signature algorithm. The features of the project of the national standard of digital signature and construction of the algorithm «Vershyna» are also given. During the analysis of the project was calculated the expected number of repetitions that the algorithm must go through to form a correct signature. The main theoretical information about the structure of Fiat-Shamir with interruptions and its stability in quantum and classical random oracle models is presented. We obtain our own results of the resistance of the «Vershyna» algorithm to the attack without the use of a message in classical and quantum oracle models. The resistance of the «Vershyna» algorithm to a key recovery attack is based on the assumption of the hardness of the MLWE problem; and resilience to existential signature forgery is based on the assumption of the hardness of the MSIS problem. In this thesis is calculated the expected level of hardness of SIS and LWE problems, to which there is a reductions from MSIS and MLWE problems.

The purpose of the thesis is to analyze the resistance to cryptanalysis of the proposed «Vershyna» digital signature algorithm and to analyze the constituent algorithms of the standard and modes of the digital signature scheme.

The research object is the processes of information transformation in the «Vershyna» digital signature.

The research subject is the resistance to cryptanalysis of the «Vershyna» digital signature.

«VERSHYNA» ALGORITHM, DIGITAL SIGNATURE SCHEME, LATTICE PROBLEMS, SECURITY ANALYSIS

ЗМІСТ

ВСТУП

Актуальність дослідження. Як відомо, станом на сьогодні стійкість більшості асиметричних криптосистем ґрунтується на складності розв'язання задач дискретного логарифмування та факторизації цілих чисел. На сьогодні не відомо поліноміальних алгоритмів розв'язку цих задач за допомогою класичних комп'ютерів. Однак ще у 1994 році американський математик Пітер Шор довів, що ці задачі ефективно розв'язуються у квантовій моделі обчислень [1]. Тож з настанням ери потужних квантових комп'ютерів більшість асиметричних криптосистем, які використовують зараз, можуть бути зламані.

Ще у грудні 2016 року NIST (Національний інститут стандартів та технологій у США) офіційно ініціював процес розробки та стандартизації нових постквантових криптографічних алгоритмів з відкритим ключем [2]. Тож наразі побудовою та перевіркою стійкості кандидатів у постквантові криптосистеми займаються спеціалісти з усього світу.

Нещодавно запропоновано проект нового національного стандарту цифрового підпису, який містить алгоритм цифрового підпису «Вершина». Прототипом для нього слугував алгоритм цифрового підпису CRYSTALS-Dilithium, який є одним з фіналістів конкурсу NIST.

Метою дослідження є оцінка стійкості алгоритму цифрового підпису «Вершина» та аналіз складових алгоритмів проекту національного стандарту цифрового підпису та режимів його роботи. Для досягнення мети необхідно виконати наступні завдання:

1) Провести огляд опублікованих джерел, необхідних для подальшого дослідження.

2) Провести аналіз наявних атак на алгоритми цифрового підпису у квантовій та класичній моделях обчислень.

3) Провести аналіз проекту стандарту з алгоритмом «Вершина» та дослідити особливості роботи його алгоритму з різними режимами роботи.

4) Отримати оцінки стійкості алгоритму «Вершина» до

- атак підробки підпису без використання повідомлення;
- атак відновлення ключа;
- екзистенційної підробки підпису.

5) Навести всі оцінки стійкості для класичної та квантової моделей обчислень.

Об'єкт дослідження: процеси перетворення інформації в алгоритмі цифрового підпису «Вершина».

Предмет дослідження: стійкість алгоритму цифрового підпису «Вершина» до криптоаналізу.

При розв'язанні поставлених завдань використовувались *методи дослідження*: теорії ймовірності, квантових обчислень, теорії складності алгоритмів, лінійної та абстрактної алгебри.

Наукова новизна отриманих результатів полягає у вперше наданих відкритих оцінках стійкості алгоритму цифрового підпису «Вершина».

Практичне значення результатів полягає у підтвердженні стійкості алгоритму цифрового підпису «Вершина» в класичній та квантовій моделях обчислень.

Апробація результатів та публікації.

Результати роботи представлені на XX Всеукраїнській науково-практичній конференції студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики». (15-16 червня 2022 року, м. Київ)

1 ОСОБЛИВОСТІ ПОБУДОВИ СХЕМИ ПІДПИСУ DILITHIUM ТА НАЯВНІ АТАКИ НА НЕЇ

У цьому розділі розглянуто алгоритм цифрового підпису на решітках CRYSTALS-Dilithium. Проаналізовано особливості його побудови та наявні модифікації. Проведено детальне дослідження наявних атак на алгоритм та ймовірності їх вдалого проведення. Наведено контрзаходи, які використовуються проти атак. Також розглянуто алгоритм проведення атаки екзистенційної підробки у моделі частково відомого особистого ключа.

1.1 Теоретичні відомості про алгоритм Dilithium та його модифікації

На сьогодні криптографічні протоколи, стійкість яких ґрунтується на складності розв'язання задач на решітках, вважаються одними з можливих варіантів заміни криптосистем при переході до поствантових алгоритмів. Зазвичай у якості задач, складність яких буде основою стійкості системи, обирають задачі знаходження найкороткротшого вектору (SVP), знаходження короткого цілого рішення (SIS) та навчання з помилками (LWE) [3]. Системи цього класу вважаються найбільш перспективними та пропонують кращу ефективність у порівнянні з іншими аналогами за рівних умов стійкості.

Цифрові підписи, напевне, є найбільш використовуваним криптографічним примітивом з відкритим ключем у практичних застосуваннях. Очевидно, що доволі багато зусиль вкладено в спроби побудувати такі схеми з використанням решіток.

Взагалі, першим застосував решітки у криптографії решітки Айтай ще у 1996 році [4]. Він у своїй роботі навів перший криптографічну схему

з решітками та довів її стійкість. Трохи згодом у наступній роботі Айтай і Дворк [5] надали схему шифрування з відкритим ключем на основі решітки. Було визнано, що його ідея варта уваги. Вже через рік Голдрайх, Голдвассер і Халеві (GGH) [6] опублікували схему шифрування з відкритим ключем і схему цифрового підпису, які використовують задачі решітки. На відміну від робіт Айтая і Дворка, для схеми GGH не було надано жодних гарантій безпеки в найгіршому випадку і схема підпису GGH пізніше була повністю зламана [7]. Однак головні ідеї, що лежали в основі схеми GGH, пізніше були відроджені та доведено, що вони є стійкими за найгіршого випадку. Ще однією схемою підпису, про яку варто згадати, є схема NTRUSign [8]. Вперше вона була представлена на Eurocrypt 2001 [9]. На схему підпису NTRU було проведено успішну атаку Нгуєном і Регевом [7] [10] всього через 3 роки після презентації. Наразі є відомою схема pqNTRUSign, яка також приймала участь у конкурсі NIST, хоча й не стала фіналістом. Наступні 2 алгоритми, на які варто звернути увагу, були опубліковані у 2007-2008 роках. Джентрі, Пейкерт і Вайкунтанатан у своїй роботі [11] запропонували схему цифрового підпису, але вона виявилася дуже неефективною на практиці, адже вихідні дані та ключі були завдовжки в мегабайти. Пізніше їх схема була дороблена і стала придатною для користування. Трохи пізніше Любашевський і Міччіансіо описали загальну структуру, яка перетворює певні типи лінійних геш-функцій, стійких до колізій, в одноразові підписи [12]. Однак головним недоліком було те, що ці підписи були саме одноразовими. Цю схему також допрацювали та отримали повноцінний алгоритм цифрового підпису, який можна застосовувати на практиці.

У 2016 році NIST офіційно ініціював процес розробки та стандартизації нових постквантових криптографічних алгоритмів з відкритим ключем. Алгоритм цифрового підпису Dilithium є одним з фіналістів конкурсу NIST PQC.

Розглянемо основні особливості алгоритму. Більшу частину подальших відомостей взято зі специфікації алгоритму Dilithium [13] [14].

Для початку варто сказати про схему, яку використовували автори алгоритму Dilithium. Основний підхід схеми ґрунтується на структурі Фіат-Шаміра з перериваннями [15], тоді як сама схема є вдосконаленим варіантом схеми підпису на основі решітки, запропонованої Баєм і Гелбрейтом [16]. Однією з її особливостей є те, що вона вимагає генерування одноразового значення (нонсу) для постановки підпису. Відповідно використання однакових нонсів для підпису повідомлень значно впливає на стійкість системи. Парадигма Фіат-Шаміра є одною з найбільш уживаних при побудові постквантових схем цифрового підпису. Шаблон алгоритму цифрового підпису Dilithium наведено в додатку А.

Алгоритм Dilithium є одним з компонентів криптографічного набору на алгебраїчних решітках (CRYSTALS). Тож наступною особливістю, яку варто згадати, є використання алгебраїчних решіток. Формальне визначення наведено у розділі 2, означення 2.1.

Варто також зауважити, що всі криптографічні операції в алгоритмі Dilithium виконуються в кільці $\mathbb{Z}_q[X]/(X^n + 1)$.

Як і в більшості схем з використанням решітки, які використовують відповідні операції над кільцями поліномів, автори алгоритму Dilithium вибрали кільце так, щоб операція множення мала дуже ефективну реалізацію через теоретичне перетворення чисел (NTT). Перетворення NTT насправді є лише частковим випадком дискретного перетворення Фур'є (FFT), яка використовується при роботі зі скінченними полями. Щоб можна було використовувати перетворення NTT, потрібно обрати просте число q так, щоб група Z_q^* мала елемент порядку $2n$. У такому випадку найскладнішими за часом виконання будуть операції власне NTT перетворення та обернене NTT перетворення.

До речі, як вже було згадано вище, існує декілька задач, на складності яких ґрунтується стійкість всіх алгоритмів на решітках. Стійкість алгоритму цифрового підпису Dilithium ґрунтується на складності розв'язку задач модульного навчання з помилками (MLWE) та

модульного короткого цілого рішення (MSIS). Зокрема, стійкість до атак відновлення ключа ґрунтується на припущенні складності задачі MLWE; а стійкість до екзистенційної підробки підпису ґрунтується на припущенні складності задачі MSIS [17].

Найкомпактніші схеми підпису з використанням решітки будують з використанням дискретного гаусового розподілу (означення 2.3). Створення відповідних випадкових послідовностей у спосіб, захищений від атак з боку побічних каналів, є дуже нетривіальною задачею і може легко призвести до небезпечних реалізацій. На відміну від своїх попередників, автори алгоритму Dilithium надали перевагу використанню рівномірного розподілу при створенні псевдовипадкових послідовностей. Враховуючи відмову від використання дискретного гаусового розподілу, вважається, що алгоритм Dilithium має найменшу комбінацію розмірів підпису та відкритого ключа серед усіх постквантових схем підпису за рівних умов стійкості.

Як бачимо на схемі з додатку А, в алгоритмі використовується деяка геш-функція H. В алгоритмі Dilithium у якості такої геш-функції використовують функція SHAKE-256.

Цифровий підпис Dilithium є параметризованою система, тож залишається можливість модифікації оригінального алгоритму. Однією з таких модифікації власне і є алгоритм «Вершина», який міститься у проекті національного стандарту цифрового підпису.

Основні положення та відмінності алгоритму «Вершина» від алгоритму Dilithium розглянуто у розділі 3

Звісно, змінюючи параметри та функції формування псевдовипадкових послідовностей та гешування можна створити велику кількість різних варіацій одного й того самого алгоритму. Однак є ще одна цікава можливість модифікувати алгоритм Dilithium. Наприклад створити з оригінального алгоритму агрегований цифровий підпис. Агрегований підпис — це об'єднання декількох непов'язаних підписів, виданих різними сторонами, які підписали різні повідомлення, в один

єдиний узагальнений підпис. Взагалі ідеальним варіантом вважається ситуація, коли це об'єднання проводить деяка інша третя сторона.

Вперше ідея була запропонована Дорозом та іншими у 2020 році [18]. Однак вона використовувала задачу часткового відновлення Фур'є (PFR - Partial Fourier Recovery problem), яка не досить детально вивчалася криптоаналітиками. Тож, можливо, знайдуться суттєві недоліки цієї схеми. Алгоритм Dilithium таким чином модифікували Катаріна Будгуст і Аделін Ру-Ланглуа [19]. Також вони надали доведення стійкості побудованої системи.

1.2 Наявні атаки на алгоритм Dilithium та контрзаходи

Від самого початку в офіційному заклику NIST до розробки нових постквантових алгоритмів серед таких вимог як параметризованість, можливість паралелізації та інше, було згадано стійкість до атак за побічними каналами (side-channel attacks) [20]. Відповідно, більшість атак, які проведено на алгоритм Dilithium саме з використанням різних методів атаки за побічними каналами. Методами атаки за побічними каналами є не тільки атаки помилки, а й простий аналіз потужності, диференціальний аналіз потужності, атака на кеш (CA), атака шаблонів тощо [21].

Розглядають такі основні типи атак:

1) Класичний криптоаналіз (CC) — відповідно до назви, це класичний математичний аналіз криптосистеми.

2) Часовий аналіз (STA) використовує різницю часу виконання алгоритму. На основі зміни часу можна зробити деякі припущення щодо секрету.

3) Атаки помилки (FA) — це методи навмисного спонукання до помилок для знаходження особистого ключа.

4) Простий (SPA), диференціальний та кореляційний (APA) аналізи потужності використовують варіації споживання енергії криптографічним

алгоритмом.

5) Електромагнітні атаки (EM) використовують електромагнітні випромінювання від криптографічного алгоритму.

6) Атаки шаблонів (TA) створюють профіль цільового пристрою, щоб після цього провести атаку з його використанням і швидко знайти особистий ключ.

7) Атака холодного перезавантаження (CB) полягає у використанні залишків пам'яті, які не одразу ж видаляються із пам'яті комп'ютера після його вимкнення.

Деніелом Апоном і Джеймсом Хоу було наведено таблицю [22] (відповідна частина щодо алгоритмів підпису наведена у таблиці 1.1). У ній вони відзначили стійкість до яких атак перевірялася у фіналістів конкурсу NIST PQC. Таблиця наведена за результатами 2-го етапу конкурсу.

Таблиця 1.1 – Типи проведених атак на фіналістів NIST PQC

Схема підпису	CC	STA	FA	SPA	APA	EM	TA	CB	CM
Dilithium			✓			✓			✓
Falcon			✓						
Rainbow	✓				✓			✓	

Прапорець «✓» означає, що велися дослідження щодо стійкості кандидату до даного виду атаки.

У цій таблиці також присутній стовпчик (CM). Він позначає контрзаходи проти атак на алгоритм. Захист відбувається за допомогою методів приховування або маскування. Докладніше про контрзаходи можна дізнатися у підрозділі 1.2.

Як можна побачити з таблиці 1.1 основні дослідження щодо стійкості алгоритму Dilithium велися у напрямах атак помилки та електромагнітних атак. Відповідно також розглядалися можливі контрзаходи, які б захищали алгоритм.

Взагалі аналіз побічного каналу (SCA) — це пасивна атака, при якій зловмисник може збирати інформацію про споживання енергії, електромагнітне випромінювання, час виконання тощо. Миттєве значення кожного з цих джерел залежить від даних, що обробляються в криптосистемі, які, зі свого боку, залежать від секретного ключа. SCA є основною загрозою безпеці, оскільки, наприклад, витік потужності дає локалізовану інформацію про кожну операцію в межах криптографічного алгоритму. Отже, зловмисник може націлитися на ті операції, в яких фігурують невеликі частини секретного ключа (ці невеликі частини також називають підключами). Тому підвищення рівня безпеки (розміру ключа) криптографічної схеми не перешкоджає SCA, оскільки кожен підключ можна відновити окремо.

Далі розглянемо деякі з вищеназваних методів. Спочатку розглянемо атаки помилки, потім перейдемо до атак профілювання та диференціального аналізу потужності. Відповідно для кожного класу атак наведено контрзаходи, які використовують для захисту алгоритму.

Диференціальні атаки помилки (FA) є однією з загроз з якою доводиться зустрічатися схемам цифрового підпису, як використовуваним зараз, так і кандидатам на цю роль. Ця доля не оминула і Dilithium.

Надалі буде йти мова про детермінований варіант алгоритму.

У алгоритмі Dilithium унікальний вектор підпису має вигляд $\mathbf{z} = \mathbf{y} + c\mathbf{s}$, де c - деякий результат геш-функції, $\mathbf{s}$ - секрет, $\mathbf{y}$ - нонс, який обчислюють детермінованим чином. Задача атаки: створити сценарій з повторним використанням нонсу і тим самим створити тривіально розв'язуваний екземпляр задачі LWE [23]. Як згадувалося раніше сама схема Фіат-Шаміра вимагає одноразовості нонсу. Тож при даній атаці зловмисник намагається шляхом додавання деяких помилок залишити нонс без змін. Якщо йому вдасться отримати різні підписи для одного й того самого повідомлення, що використовують однаковий нонс, то він також може відновити секретне значення $\mathbf{s}$ за допомогою методів лінійної алгебри [24]. На щастя, та ж сама схема Фіат-Шаміра може

знизити ймовірність успіху проведення цієї атаки, тобто не всі помилки можна використати для знаходження секрету.

Тож схема атаки FA виглядає так [25]:

1) Надаємо підписувачу двічі однаковий текст M. Але при першому підписуванні нічого не робимо і отримуємо коректний підпис $(\mathbf{z}, \mathbf{h}, c)$. Коли підпис обчислюється вдруге, вносяться деякі помилки на певному етапі виконання алгоритму таким чином щоб отримати значення $\mathbf{z}' = \mathbf{y} + c'\mathbf{s}_1$.

2) Обчислюють $\Delta\mathbf{z} = \mathbf{z} - \mathbf{z}'$, і аналогічно Δc ($\Delta\mathbf{y} = 0$ оскільки умовою є зберегти незмінним нонс).

3) Складають і розв'язують відносно $\mathbf{s}_1$ рівняння: $\Delta\mathbf{z} = \Delta c \cdot \mathbf{s}_1$. Якщо існує обернений елемент до Δc, то це рівняння має розв'язок, а саме: $\mathbf{s}_1 = \Delta c^{-1} \cdot \Delta\mathbf{z}$.

Через те що у Dilithium використовується схема Фіат-Шаміра з перериваннями, виникає ще одна умова успішності атаки: обчислення обох підписів повинні бути перервані на одній і тій самій ітерації циклу переривання.

Помилки можуть вноситися на різних етапах обчислень. У таблиці 1.2 наведено можливі сценарії атак на помилки та ймовірності успіху проведення цих атак.

Таблиця 1.2 – Типи атак помилки та ймовірність їх успіху

Назва	Короткий опис	Ймовірність, %
fA_ρ	пошкоджує ρ під час імпорту ключа	14.3
fA_E	випадкова помилка під час розширення $\mathbf{A}$	54.4
fY	випадкова помилка під час обчислення $\mathbf{y}$	$23.9 - 24.8$
fW	випадкова помилка у множенні поліномів	$25.4 - 90.3$
fH	випадкова помилка під час виклику H	91.0

Якщо уважно розглянути алгоритм А.2, то побачимо, що для проведення атаки необхідно змінити значення $c = SampleInBall(\widetilde{c})$, де $\widetilde{c} = H(\mu, \mathbf{w}_1)$. Функція $SampleInBall(\widetilde{c})$ є варіантом алгоритму

Фішера – Йєтса. Її застосовують для створення псевдовипадкового елементу c з рівно τ коефіцієнтами рівними ± 1 та $n - \tau$ нульовими коефіцієнтами, використовуючи для цього значення $\widetilde{c}$. Інтуїтивно хочеться внести помилки саме в момент, коли обчислюють значення $\widetilde{c}$. Логічно, що можна або змінити вхідні значення μ, $\mathbf{w}_1$, або безпосередньо ввести помилку в саму геш-функцію H (сценарій fH).

Розглянемо випадок, коли помилки вносяться у вхідні змінні μ, $\mathbf{w}_1$. Значення μ не можна змінювати, адже воно входить як вхідний параметр до функції, яка обчислює $\mathbf{y}$ (у детермінованому варіанті алгоритму $\mu = CRH(tr\|M) = \rho'$). Таким чином залишається тільки можливість змінювати $\mathbf{w}_1$, а отже потрібно внести помилки у обчислення значення $\mathbf{w} = \mathbf{A}\mathbf{y}$. Це можна зробити двома шляхами: це може бути просто помилка у множенні поліномів (fW), або помилка, яка виникає ще в момент обчислення матриці $\mathbf{A}$ (fA_E, fA_ρ).

У всіх попередніх сценаріях атаки вважають, що $\mathbf{y}' = \mathbf{y}$. Однак залишився ще один нерозглянутий сценарій, а якось по іншому змінити c без зміни $\mathbf{y}$ не можливо. Тож в останньому сценарії (fY) дозволяється зміна нонсу. Так, наприклад, нонс може детерміновано збільшуватись на якусь константу. Через те, що нонс все ж таки змінюється отримуємо 2 дійсні підписи (на відміну від попередніх атак, де тільки перший підпис є дійсним). У такому випадку є неможливим виявити помилку шляхом звичайної перевірки підпису. Однак внаслідок детермінованості зміни нонсу ця атака все одно може бути проведена.

Розглянемо способи, якими можна спробувати захиститися від атак помилки.

Так, наприклад, можна додати перевірку підпису кожного разу після формування цього підпису. Оскільки при більшості атак на помилки отримуємо підпис з помилкою, то таку звичайну перевірку він не пройде. Ця міра захисту також є доволі ефективною за часом виконання, адже перевірка підпису проходить достатньо швидко. Однак, як вже було згадано, від сценарію fY така перевірка не захистить і ніяких помилок

виявлено не буде.

Звісно багато помилок можна виявити, просто запустивши обчислення підпису двічі та перевіривши виходи на рівність. Однак час виконання відповідно також збільшується. Ще одним недоліком цього варіанту захисту є те, що він є неефективним, якщо обидва рази вноситься ідентична помилка. Звісно складно гарантувати однаковість помилки, але забезпечити таку подію можливо.

Ще один і, напевне, найпростіший спосіб захисту: внести додаткову випадковість при детермінованому виборі $\mathbf{y}$. Наприклад, додати випадковий вектор до входу функції, яка обчислює $\mathbf{y}$. Цей спосіб захисту є найшвидшим у реалізації та перешкоджає усім вище згаданим атакам внаслідок додавання випадковості. Більш того цей вид захисту також допомагає захиститися від інших методів атак за побічними каналами, які розглянуто далі.

Атака профілювання — одна з найпотужніших атак за побічними каналами. Вона складається з фази навчання та фази атаки.

Під час фази навчання створюється профіль (за допомогою навчального пристрою із такими ж чи подібними специфікаціями, що й у цільового пристрою), який буде використовуватися на етапі атаки. Для цього визначаємо, які значення слід вивчати та які моделі використовувати. Їх називають маркуванням і моделюванням відповідно.

Такого типу атака була проведена Кімом та ін. [21]. Вони вперше показали атаку з використанням всього одного сліду на алгоритм Dilithium. Крім того, їхня команда не обмежилася атакуванням лише незахищеного варіанту алгоритму. Вони також провели атаку і на алгоритм, до якого було застосовано маскування.

Зауваження. Щойно згадано, що у своїй атаці Кім та ін. використали всього один слід. Варто зауважити, що це атаки, які використовують простий аналіз потужності (SPA). За таких атак використовують лише один спостережуваний слід витоку (або середнє значення багатьох слідів витоку, але входи для кожного обчислення

обов'язково повинні бути *однаковими*) Цей вид атаки показує гарні результати, коли рівень шуму значно менший, за об'єм витоку [26].

У якості цілі для атаки на незамаскований Dilithium була обрана функція NTT. З іншого боку вони також розглянули атаку на схему замаскованого алгоритму підпису Dilithium. У якості схеми маскування розглядали схему, яку запропонували Мільоре та ін. [27]. Через проблеми з обмеженнями ефективності Мільоре та ін. зосередилися на оптимізованій версії модуля. А саме у якості модуля було взяте не просте число, а число, яке дорівнює степені двійки. Тому перетворення NTT у захищеному варіанті алгоритму є недоступним. З цієї причини Кім та ін. взяли поліноміальне множення, а не перетворення NTT, як цільову операцію для замаскованого алгоритму Dilithium.

В обох випадках методом відновлення ключа слугувала атака профілювання на основі машинного навчання. І, як відомо, потрібно визначити які значення слід вивчати та які моделі використовувати.

У своїй роботі Кім та ін. в якості значення маркування обрали секретні ключі, а для моделювання автори використали багатошаровий персептрон (MLP). Персептрон складається з трьох шарів (вхідний шар, прихований шар, вихідний шар), і кожен шар має вузол, який відновлює секретні ключі, змінюючи вагу.

У своїй роботі вони зосередилися на пошуку двох частин секретного ключа: $\mathbf{s}_1$, $\mathbf{s}_2$. Тож компоненту $\mathbf{t}_0$ легко знайти. Відповідно ця команда запропонувала атаку, яка відновлює повний ключ повністю незалежно від рівня оптимізації.

Як вже було згадано, якщо шуму небагато у порівнянні з об'ємом витоку інформації, то можна провести атаки SPA. Однак, якщо трапилася інша ситуація (тобто витік відносно шуму є набагато меншим), застосовуються такі статистичні методи, як **диференціальний аналіз потужності (DPA)**. Метод DPA на відміну від SPA збирає декілька слідів витоку, які отримані на *різних* входах. Велика кількість вибірок використовують з метою зменшення шуму [26].

Метод DPA є більш серйозною загрозою. Для проведення атаки потрібно діяти за таким алгоритмом [28]:

1) Необхідно обрати чутливу змінну в алгоритмі. Чутлива змінна — це проміжна змінна в криптографічному алгоритмі, яка залежить як від ключа (як правило, лише 8 бітів або менше), так і від відомих вхідних або вихідних даних (відкритий текст або зашифрований текст).

2) Після цього збирають дані про витоки (споживання енергії) криптографічного модуля під час обробки багатьох різних вхідних даних.

3) На тих самих входах потрібно обчислити значення чутливої змінної.

4) Перетворюють отримане значення в гіпотетичне споживання електроенергії за допомогою моделі витоку.

5) І, врешті-решт, порівнюють гіпотетичне споживання енергії з фактичним, шукаючи ключ, який веде до найкращого збігу.

Атака DPA на незахищений варіант алгоритму Dilithium проведена Мільоре та ін. [27]. Їхні результати підтверджують, що зловмисник, який має фізичний доступ до пристрою, може легко виконати відновлення ключа за допомогою атаки за побічними каналами у стандартній реалізації алгоритму Dilithium. Вони також запропонували деякі рекомендації для ефективного захисту алгоритму Dilithium. Таким чином ними надано аналіз витоку як для немаскованої, так і для замаскованої версії Dilithium. Цей аналіз показав, що для успішного знаходження деяких витоків функцій декомпозиції та відхилення у незахищеному варіанті потребується не більше ніж 500 слідів. В той час як захищена версія показала значно кращі результати: навіть з використанням 10000 слідів витоку не було виявлено.

Також варто згадати про **контрзаходи проти атак за побічними каналами**.

На щастя, атаки за побічними каналами не завжди можна провести. По-перше, вони вимагають фізичного доступу до пристрою обчислення. Більш того криптографічний модуль уразливий до цих атак, якщо

виконуються такі умови:

1) Зловмисник може передбачити значення чутливої змінної.

2) Значення чутливої змінної впливає на витік деяким передбачуваним чином. Тож детерміновані системи є уразливими до цього виду атак.

3) Інформацію з кількох спостережень можна об'єднати.

Контрзаходи поділяються на три категорії. Кожна категорія відповідно порушує одну з вищезазначених умов таким чином:

1) Маскування: під час роботи алгоритму додають нову випадкову змінну, яку використовують для рандомізації введених даних і всіх обчислень. Її слід видаляти лише в кінці обчислень. Такі дії виконують з метою зашкодити зловмиснику правильно передбачити проміжні змінні.

2) Приховування: перший метод приховування полягає у розробці окремого резервного модуля обробки доповнення вхідних даних. Таким чином споживана потужність системи залишається незмінною. По-друге, якщо це є можливим, гарною ідеєю є дещо рандомізувати порядок виконання програми (це можна зробити там, де немає залежності даних від попередніх обчислень). Такі дії називають перемішуванням. Третій метод — спроектувати додатковий генератор шуму. Метою цих інструментів є мінімізація відношення сигнал/шум. Це обмежує вплив проміжних змінних на витік потужності.

3) Стійкість до витоку: використовують додатковий механізм оновлення ключа. Він призначений для обмеження терміну життя будь-якого ключа лише кількома криптографічними операціями. Таким чином виток інформації про ключ дуже обмежується.

Як метод захисту від атак DPA найчастіше використовується маскування. Двома найпоширенішими підходами є булеве (логічне) та арифметичне маскування.

Булеве : $x' = x \oplus r$

Арифметичне : $x' = (x \pm r) \mod n$

Тут x — змінна, яку хочемо замаскувати за допомогою деякого

випадкового значення r. Після маскування отримаємо іншу змінну x'.

Використання маскування іноді може виявитися нетривіальною задачею, адже зазвичай алгоритми вимагають використання обох типів маскування. Таким чином, захист усього алгоритму вимагає перетворення між арифметичним і булевим маскуванням. Варіант алгоритму виконання таких перетворень був запропонований Короном та Губеном [29].

Якщо маємо справу з цифровими підписами, то не можна не згадати про **екзистенційні підробки**.

Більшість атак дозволяють з великою імовірністю успіху знайти секретне значення $\mathbf{s}_1$ шляхом аналізу витоків та введення помилок до незахищеного алгоритму. Звичайно першим бажанням зловмисника є знайти значення $\mathbf{s}_2$, щоб повністю зламати систему.

Перший метод полягає в спробі відновити секрет $\mathbf{s}_2$ за допомогою статистичного аналізу. Знаючи значення $\mathbf{s}_1$, зловмисник може тривіально обчислити компонент $\mathbf{w}$ для всіх дійсних підписів. Оскільки також відомо, що всі вихідні підписи дотримуються умови, що $HighBits_q(\mathbf{w} - c\mathbf{s}_2,\ 2\gamma_2) \leq \gamma_2 - \beta$, можна вивести певні умови на можливі значення $\mathbf{s}_2$. Функція $HighBits_q(x,\alpha)$ представляє елемент x у вигляді $x = x_1 \cdot \alpha + x_0$ та повертає значення старших бітів x_1. Але такий підхід вимагає великої кількості підписів та обчислювальних зусиль, що підвищує складність цієї атаки [25].

Інший метод полягає у спробі відновити секрет $\mathbf{s}_2$ за допомогою, наприклад, атаки за побічними каналами. Але в практичних умовах за допомогою атаки SCA не завжди можливо отримати секрети повністю. Проблеми можуть виникнути через необхідність або аналізувати велику кількість спостережень для відновлення секретного ключа, або вводити велику кількість успішних помилок (якщо проводимо атаку помилок). Таким чином, зловмиснику, можливо, доведеться перебрати коефіцієнти елементу $\mathbf{s}_2$, які не вдалося дістати в результаті атаки, щоб повністю відновити ключ. Цей перебір також вимагає значних обчислювальних

зусиль, що унеможливлює успішне знаходження секрету.

Очевидно, що тоді постає питання підробки підпису під бажаним повідомленням з використанням лише значення $\mathbf{s}_1$.

Екзистенційну підробку в моделі частково відомого особистого ключа у своїй роботі розглянули П. Раві та інші [30]. Вони довели, що знання значення $\mathbf{s}_1$, як єдиного елементу з усього особистого ключа (ρ, K, tr, $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{t}_0$), достатньо для підробки підписів. Таким чином показано, що секретність значення $\mathbf{s}_1$ є ключовою для безпеки алгоритму Dilithium.

До речі, у 2018 році було 2 паралельних дослідження можливого алгоритму екзистенційної підробки [25] [30]. У роботі [25] вважалося, що значення $\mathbf{s}_1$ здобувають за допомогою атаки помилок, а у [30] для цього використовується SCA.

Однак у дослідженні [25] бракує чіткого аналізу основних факторів, які дозволяють підробку підпису, лише зі знанням $\mathbf{s}_1$. Автори роботи вважали, що значення $\mathbf{s}_1$ насправді є єдиним секретом. Це випливало з таких міркувань: ρ є частиною і секретного, і публічного ключа; tr можна просто обчислити за формулою $tr = CRH(\rho \| \mathbf{t}_1)$ (CRH — деяка геш-функція); K не може бути відновлено, однак зловмисник може просто вибрати будь-яке випадкове значення K і все одно створювати дійсні підписи. А значення $\mathbf{s}_2$, $\mathbf{t}_0$ взагалі не беруть участі у підробці підпису.

З іншого боку Раві та ін. навели схему підробки підпису на основі саме доведення можливості використання для підпису лише часткового особистого ключа. Тож ці 2 паралельні дослідження призвели до дещо різних екзистенційних підробок. Схеми відповідних підробок можна побачити у додатку Б.

Висновки до розділу 1

У розділі розглянуто алгоритм цифрового підпису Dilithium, який слугував прототипом для алгоритму «Вершина». Наведено основні особливості алгоритму Dilithium та декілька його наявних модифікацій.

Крім того, проаналізовано наявні атаки на алгоритм цифрового підпису Dilithium. Детально розглянуто такі атаки як атаки за побічними каналами та атаки помилки з обчисленням ймовірності їхнього успішного проведення. Наведено три категорії, на які поділяються контрзаходи проти атак за побічними каналами: маскування, приховування та стійкість до витоку. Оскільки розглядається алгоритм цифрового підпису також досліджено атаки екзистенційної підробки, які проводяться в моделі частково відомого особистого ключа.

2 ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

Для подальших досліджень необхідна теоретична база, яка надасть розуміння основних моментів. Таким чином надалі надаються основні означення та уточнюються деякі позначення, які будуть використовуватись у подальшій роботі. Також надаються теоретичні основи, які стосуються обчислення оцінки захищенності алгоритму. Вказується, з яких елементів складається захищенність схеми у моделі випадкового оракула (ROM) та в чому відмінність від моделі квантового випадкового оракулу (QROM).

2.1 Основні необхідні означення та позначення

Оскільки «Вершина» є алгоритмом цифрового підпису, який використовує алгебраїчні решітки, потрібно ввести поняття решітки та пов'язані з ним визначення.

Означення 2.1. *m-вимірна решітка* — це дискретна адитивна підгрупа групи $\mathbb{R}^m$, яку представляють як множину цілочисельних лінійних комбінацій з n лінійно незалежних векторів $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$. Тобто

$$\mathfrak{L}(\mathbf{B}) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z} \ \forall i \in [1,n] \right\},$$

де $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ називають *базисом решітки* $\mathfrak{L}$, а n — *рангом решітки.*

Детермінантом решітки $\mathfrak{L}$ називають величину:

$$det(\mathfrak{L}) = \sqrt{Det(\mathbf{B}^T\mathbf{B})},$$

де Det — детермінант матриці.

Означення 2.2. Для довільних значень $s > 0$ та $\mathbf{y} \in \mathbb{R}^m$ *функцію*

Гауса визначають як:

$$\rho_{\mathbf{c},s}(\mathbf{y}) = e^{-\pi\|\mathbf{y}-\mathbf{c}\|^2/s^2}.$$

Означення 2.3. Для довільних значень $s > 0$ та $\mathbf{c} \in \mathbb{R}^m$ *дискретним гаусовим розподілом* $D_{\mathbb{L}+\mathbf{c},c}$ на $\mathbb{L} + \mathbf{c}$ називають функцію:

$$D_{\mathbb{L}+\mathbf{c},c} = \frac{\rho_s(\mathbf{x})}{\rho_s(\mathbb{L}+\mathbf{c})},$$

де $\mathbf{x} \in \mathbb{L} + \mathbf{c}$ і $\rho_s(\mathbb{L}+\mathbf{c}) = \sum\limits_{\mathbb{L}+\mathbf{c}} \rho_s(\mathbf{x})$.

Величину $\sigma = s/\sqrt{2\pi}$ називають *стандартним відхиленням* для функції $D_{\mathbb{L}+\mathbf{c},c}$.

Як вже згадувалося автори алгоритму Dilithium відійшли від використання дискретного гаусового розподілу. Відповідно у алгоритмі «Вершина» також було надано перевагу рівномірному розподілу. Приклади оцінки відстані між двома розподілами можна переглянути у роботі [32].

Як вже було зазначено раніше алгоритм «Вершина», як і Dilithium, використовують структуру Фіат-Шаміра з перериваннями, яка є одною з найуживаніших парадигм при побудові постквантових схем цифрового підпису. Аналіз захищенності подібних систем велися різними вченими і результати можна подивитися, наприклад, у роботах [32], [24], [31] та [33].

Зосередимося на структурі Фіат-Шаміра, та введемо необхідні означення. Перетворення Фіата-Шаміра поєднує канонічну схему ідентифікації та геш-функцію H для отримання схеми цифрового підпису $FS = FS[ID, H]$.

Означення 2.4. *Канонічною схемою ідентифікації* ID називають набір алгоритмів $ID := (IGen, P, ChSet, V)$.

1) Алгоритм створення ключів $IGen$ приймає системні параметри par як вхідні дані і повертає відкритий і особистий ключ $(pk,\ sk)$. Припускають, що відкритий ключ pk визначає $ChSet$ (набір викликів), $WSet$ (набір зобов'язань) і $ZSet$ (набір відповідей).

2) Алгоритм доведення $P = (P_1, P_2)$ розбивають на два алгоритми. Алгоритм P_1 приймає як вхід особистий ключ sk і повертає зобов'язання $W \in WSet$ і стан St; Алгоритм P_2 приймає в якості вхідних даних особистий ключ sk, зобов'язання W, виклик c і стан St і повертає відповідь $Z \in ZSet \cup \{\perp\}$, де $\perp \notin ZSet$ – спеціальний символ, що вказує на помилку.

3) Перевіряльник V приймає відкритий ключ pk і стенограму розмови (означення 2.5) як вхідні дані та виводить детерміновані рішення: 1 (прийняття) або 0 (відхилення).

Означення 2.5. *Транскрипт* (або *стенограма*) — це кортеж $(W, c, Z) \in (WSet \times ChSet \times ZSet) \cup \{\perp\ ,\ \perp\ ,\ \perp\}$.

Його називають *дійсним* (по відношенню до відкритого ключа pk), якщо $V(pk, W, c, Z) = 1$.

Зауваження. В означенні 2.4 набір викликів (тобто множину значень поліному c) позначено як $ChSet$ для наочності (Challenge Set). Насправді, поліном c приймає значення з множини B_τ, яка містить поліноми з рівно τ коефіцієнтами рівними ± 1, а решта $n - \tau$ коефіцієнтів є нульовими.

Означення 2.6. Якщо найбільш ймовірне значення випадкової величини W, що вибирається з дискретного розподілу D, зустрічається з імовірністю $2^{-\alpha}$, то *мінімальна ентропія* дорівнює α.
Позначають: $min - entropy(W \mid W \leftarrow D) = \alpha$.

Означення 2.7. Канонічна схема ідентифікації ID має α *бітів мінімальної ентропії*, якщо

$$\underset{(pk,sk)\leftarrow IGen(par)}{Pr}[min - entropy(W \mid (W,St) \leftarrow P_1(sk)) \geq \alpha] \geq 1 - 2^{-\alpha}.$$

Формальний опис задач MLWE, MSIS та SelfTargetMSIS

Оскільки стійкість «Вершини» ґрунтується на складності задач MLWE, MSIS та SelfTargetMSIS, то необхідно їх навести. Нижче наведено необхідні формальні означення цих трьох задач.

Означення 2.8. Для цілих чисел m, k і розподілу ймовірностей $D : R_q \to [0, 1]$ функцією переваги (advantage) алгоритму A у розв'язанні задачі $MLWE_{m,k,D}$ над кільцем R_q є значення, яке дорівнює

$$Adv^{MLWE}_{m,k,D} := |Pr[A(\mathbf{A},\mathbf{t}) = 1 \mid \mathbf{A} \leftarrow R_q^{m\times k},\mathbf{t} \leftarrow R_q^m]$$

$$-Pr[A(\mathbf{A},\mathbf{As}_1 + \mathbf{s}_2) = 1 \mid \mathbf{A} \leftarrow R_q^{m\times k},\mathbf{s}_1 \leftarrow D^k,\mathbf{s}_2 \leftarrow D^m]|.$$

Означення 2.9. З алгоритмом A пов'язують функцію переваги $Adv^{MSIS}_{m,k,\gamma}(A)$ для вирішення задачі $MSIS_{m,k,\gamma}$ над кільцем R_q як

$$Adv^{MSIS}_{m,k,\gamma}(A) := Pr[0 < \|\mathbf{y}\|_\infty \leq \gamma \wedge [\mathbf{I} \mid \mathbf{A}] \cdot \mathbf{y} = 0 \mid \mathbf{A} \leftarrow R_q^{m\times k},\mathbf{y} \leftarrow A(\mathbf{A})].$$

Означення 2.10. Припустимо, що $H : \{0, 1\}^* \to B_\tau$ є криптографічною геш-функцією. Тут B_τ - це множина з якої обираються поліноми виклику c. Всі коефіцієнти полінома приймають значення з множини $\{-1,\ 0,\ 1\}$ і кожний поліном містить рівно τ ненульових коефіцієнтів. З алгоритмом A пов'язують функцію переваги $Adv^{SelfTargetMSIS}_{H,m,k,\gamma}(A)$ для розв'язання задачі $SelfTargetMSIS_{H,m,k,\gamma}$ над кільцем R_q як

$$Adv^{SelfTargetMSIS}_{H,m,k,\gamma}(A) := Pr[\|\mathbf{y}\|_\infty \leq \gamma \wedge H([\mathbf{I} \mid \mathbf{A}] \cdot \mathbf{y}\|M) = c \mid \mathbf{A} \leftarrow R_q^{m\times k};$$

$$\left(y := \begin{bmatrix} r \\ c \end{bmatrix},\ M\right) \leftarrow A^{|H>}(\mathbf{A})].$$

2.2 Теоретичні основи оцінки стійкості цифрового підпису у класичній та квантовій моделях обчислень

Безпека схеми цифрового підпису Фіат-Шаміра в моделі випадкового оракула (ROM).

Безпека структури Фіат-Шаміра у моделі ROM може бути доведена у два кроки. По-перше, якщо базова схема ідентифікації має статистичну

властивість HVZK (Honest-Verifier Zero-Knowledge), то UF-CMA (стійкість до атак на основі обраного повідомлення) і UF-NMA (стійкість до атак без використання повідомлення) є жорстко еквівалентними (UF-NMA означає, що зловмиснику не дозволено робити запити на підпис будь-яких повідомлень). По-друге, для підтвердження безпеки UF-NMA в моделі випадкового оракула (ROM) використовується «лема про розгалуження» (Forking Lemma) [34].

Квантова модель випадкового оракула (QROM).

Квантові комп'ютери можуть виконувати всі примітиви, такі як геш-функція, для довільних суперпозицій, що спонукало до впровадження квантової моделі випадкового оракула (QROM). Тобто в моделі стійкості UF-CMA для підписів у QROM зловмисник має квантовий доступ до ідеальної геш-функції H і класичний доступ до оракула підписання. Допомога в побудові безпечних підписів UF-CMA з доказовою (постквантовою) безпекою в QROM є основною мотивацією роботи [31].

Стандартним поняттям безпеки для цифрових підписів є стійкість в моделі UF-CMA, тобто захист від атак з використанням обраного повідомлення. У цій моделі зловмисник має відкритий ключ і має доступ до оракула підписання для підпису повідомлень на свій вибір. Мета зловмисника — знайти правильний підпис нового повідомлення без використання оракула. Трохи сильніша вимога захищеності, яка також є корисною в деяких випадках, — це SUF-CMA (сильна стійкість до атак на основі обраного повідомлення). У цій моделі зловмисник досягає успіху, якщо йому вдається створити інший підпис повідомлення, яке він уже обирав.

Можна показати, що у моделі випадкового оракула (ROM) алгоритм «Вершина», так само як і Dilithium, мають властивість захищеності від атак виду SUF-CMA завдяки складності розв'язку стандартних задач на решітках: задача MLWE (є узагальненням задач LWE і Ring-LWE) та задача MSIS (є узагальненням задач SIS

і Ring-SIS). Крім того, також потрібно враховувати захищеність схеми, коли зловмисник може надіслати запити до оракула обчислення геш-функції з використанням квантової суперпозиції вхідних даних (тобто, захищеність в моделі квантового випадкового оракула – QROM). Оскільки класичне доведення стійкості використовує «лему про розгалуження», то в загальному випадку зведення не є правильним у моделі QROM.

Варто зазначити, що існує задача, яка повністю еквівалентна, навіть за квантових зведень, властивості UF-CMA. Ця задача, по суті, є комбінацією основної математичної задачі (наприклад, задачі MSIS або дискретного логарифмування) з криптографічною геш-функцією H. Таким чином отримуємо третю задачу, на складності якої ґрунтується стійкість схеми. SelfTargetMSIS є вищезгаданою задачею, яка побудована як комбінація задачі MSIS і геш-функції H.

Якщо зловмисник A має доступ до оракулу H в класичній моделі, то існує зведення, яке використовує лему про розгалуження (forking lemma), щоб довести, що

$$Adv^{SelfTargetMSIS}_{H,m,k,\gamma}(B) \approx \sqrt{Adv^{MSIS}_{m,k,2\gamma}(A)/Q_H},$$

де Q_H – кількість класичних запитів до оракулу H. Це зведення є стандартним і неявно вказано в (класичних) доведеннях захисту цифрових підписів, стійкість яких ґрунтується на складності задачі MSIS.

Безпека алгоритму Dilithium у моделі QROM може бути виражена наступним чином:

$$Adv^{sUF-CMA}_{Dilithium}(A) \leq Adv^{MLWE}_{k,l,D}(B) + Adv^{SelfTargetMSIS}_{H,k,l+1,\zeta}(C)$$

$$+Adv^{MSIS}_{k,l,\zeta'} + 2^{-\alpha+1},$$

де D рівномірний розподіл по S_η,

$$\zeta = max\{\gamma_1 - \beta, 2\gamma_2 + 1 + 2^{d-1}w_c\} \leq 4\gamma_2,$$

$$\zeta^{'} = max\{2(\gamma_1 - \beta), 4\gamma_2 + 2\} \leq 4\gamma_2 + 2,$$

де w_c — кількість ± 1 у поліномі $c \in B_\tau$.

Нижче наведено формулу для оцінки значення α:

$$\Pr_{\mathbf{A} \leftarrow R_q^{k \times l}} \left[\forall \mathbf{w}_1 : \Pr_{\mathbf{y} \leftarrow S^l_{\gamma^{'}-1}} [HighBits_q(\mathbf{Ay}, 2\gamma) = \mathbf{w}_1] \leq \left(\frac{2\gamma + 1}{2\gamma^{'} - 1} \right)^n \right] > 1 - \left(\frac{n}{q} \right)^{kl}$$

Вона нагадує означення мінімальної ентропії для схеми ідентифікації. За допомогою цієї формули та означення 2.7 стає можливою оцінка значення α.

Найвідомішим алгоритмом знаходження дуже коротких ненульових векторів у решітках є алгоритм Блока-Коркіна-Золотарьова (BKZ), запропонований Шнорром і Ойхнером у 1991 році.

Алгоритм BKZ з розміром блоку b здійснює виклики до алгоритму, який розв'язує задачу знаходження найкоротшого вектора у решітці розмірності b (SVP). Стійкість алгоритму «Вершина» залежить від необхідності виконувати алгоритм BKZ з великим розміром блоку b і того факту, що складність розв'язання задачі SVP є експоненційною відносно значення b. Найбільш ефективний класичний алгоритм для розв'язку задачі SVP працює за час $\approx 2^{0.292 \cdot b}$. Найбільш ефективний квантовий алгоритм для розв'язку задачі SVP виконує пошук за час $\approx 2^{0.265 \cdot b}$. Можна сподіватися покращити час виконання до $\approx 2^{0.2075 \cdot b}$. Наразі більш ефективного алгоритму розв'язку задачі SVP не знайдено.

Існує також ще один метод за допомогою якого можна оцінити стійкість схеми цифрового підпису, а саме **ентропія полінома виклику** [32].

У алгоритмі «Вершина» введено метод для зниження ймовірності відхилення для досягнення кращої ефективності: кількість ненульових коефіцієнтів полінома виклику c змінюється залежно від рівня безпеки. Оскільки всі коефіцієнти полінома виклику знаходяться в множині $\{-1, 0, 1\}$, ентропія полінома виклику, обраного з множини B_τ,

дорівнює $\log \binom{256}{\tau} + \tau$ біт.

Однак невелика ентропія призводить до можливості прямої атаки підробки без будь-якого дійсного повідомлення для схеми Фіат-Шаміра з перериваннями. Враховуючи відкритий ключ $pk = (\mathbf{A}, \mathbf{t})$, атаку підробки можна здійснити, знайшовши підпис $\sigma^{'} = (z^{'},c^{'})$ такий, що виконуються такі умови на його складові:

$$\begin{gathered} \|z^{'}\|_{\infty} < \gamma_1 - \beta \\ c^{'} = H(M\|\mathbf{w}_1) = H(M\|HighBits_q(\mathbf{A}z^{'} - c^{'}\mathbf{t},2\gamma_2)). \end{gathered} \tag{2.1}$$

Тут $HighBits_q$ — функція виділення старшої частини поліному, H — геш-функція, M — повідомлення. Пояснення до параметрів наведено у розділі 3.1.

Для будь-якого фіксованого $c^{'} \in B_\tau$ зловмисник може підробити підпис, вибравши деяке $z^{'}$, і перевірити, чи виконуються умови 2.1 на значення $c^{'}$ та $z^{'}$. Оскільки ентропія значення $z^{'}$ є набагато більшою, ніж ентропія значення $c^{'}$, складність проведення атаки буде складати $O\left(\log \binom{256}{\tau} + \tau\right)$ у класичній моделі (ROM). Крім того, розглядаючи формулу для значення $c^{'}$ як функцію від $z^{'}$, алгоритм Гровера можна використовувати для досягнення квадратичного прискорення з часовою складністю $O\left(\sqrt{\log \binom{256}{\tau} + \tau}\right)$ у квантовій моделі обчислень (QROM).

Зауваження. Алгоритм Гровера [35], запропонований у 1996 році, є квантовим алгоритмом швидкого пошуку у базі даних. Цей алгоритм може квадратично прискорити проблему неструктурованого пошуку, але його можна використовувати для отримання квадратичного покращення часу виконання для низки інших алгоритмів.

Висновки до розділу 2

У розділі наведено основні теоретичні відомості, необхідні для подальшої роботи. Визначено основні терміни і уточнено позначення. Серед них означення решітки та пов'язані з ним означення. Надано формальний опис задач, на складності яких ґрунтується стійкість алгоритму «Вершина», а саме MLWE, MSIS та SelfTargetMSIS. Надано теоретичні відомості щодо стійкості алгоритмів цифрового підпису, які використовують схему Фіат-Шаміра з перериваннями, у моделях ROM та QROM. Вказано відмінність захищенності схеми у класичній та квантовій моделях обчислень. Визначено поняття ентропії поліному виклику та показано як обчислити складність проведення атаки підпробки.

3 КРИПТОАНАЛІЗ АЛГОРИТМУ ЦИФРОВОГО ПІДПИСУ «ВЕРШИНА»

Цей розділ присвячено результатам дослідження алгоритму цифрового підпису «Вершина». Проведено аналіз проекту національного стандарту з алгоритмом «Вершина», окреслено його особливості та можливі проблеми, які з ними пов'язані. Проведено аналіз обраних значень параметрів для алгоритму підпису. Обчислено розміри ключів, як особистого, так і відкритого, та отриманого цифрового підпису. Оцінено очікувану кількість ітерацій, яку виконає алгоритм для постановки дійсного підпису. Досліджено стійкість алгоритму до атак підробки у класичній та квантовій моделях обчислень і виконано порівняння з заявленими авторами рівнями стійкості. Обчислено складність розв'язку задач SIS та LWE.

3.1 Зауваження до проекту стандарту з алгоритмом «Вершина»

Проект стандарту «Вершина» складається з таких частин:

1) Частина, яка включає в себе сферу застосування, нормативні посилання, позначення та скорочення і загальні положення.

2) Алгоритми створення асиметричних пар ключів, формування та перевірки цифрового підпису. Всі алгоритми подані у вигляді псевдокоду.

3) Системні та додаткові параметри разом з призначенням та формулами для обчислень.

4) Передобчислення для перетворень NTT (прямого і оберненого) для значень $n = 256$ та $n = 512$.

5) Тестові вектори для функції гешування (ДСТУ 7564:2014), для створення псевдовипадкових послідовностей (ДСТУ 8845:2019), для

створення ключів (перший та останній компоненти матриці $\mathbf{A}$, векторів $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{t}_0$, $\mathbf{t}_1$ і власне самі асиметричні ключі), формування та перевірки цифрового підпису. Всі тестові вектори наявні для 4-х режимів роботи алгоритму.

6) Бібліографія.

Виділимо основні положення та деякі відмінності алгоритму «Вершина» від Dilithium:

Параметри

Як вже зазначалось, автори проекту з алгоритмом «Вершина» модифікували алгоритм Dilithium та додали 2 набори параметрів для нових рівней стійкості. Перші 2 набори параметрів запозичено в алгоритму Dilithium. Тож алгоритм «Вершина» передбачає 4 режими роботи в залежності від рівня криптографічної стійкості, який потрібно забезпечити. Для цих додаткових наборів параметрів є необхідним відповідний аналіз стійкості.

Для подальшої роботи наведемо пояснення до параметрів алгоритму «Вершина»:

1) n — степінь поліномів;

2) q — модуль, за яким зводять усі коефіцієнти поліномів;

3) l — розмір вектору s_1;

4) k — розмір вектору s_2;

5) η визначає діапазон значень для коефіцієнтів поліномів, з яких складаються вектори s_1 та s_2, які входять до складу особистого ключа;

6) ω визначає максимальну кількість одиниць в матриці $\mathbf{h}$;

7) β — параметр, який застосовують для обчислення норми перетворень;

8) γ_1 — параметр, який застосовують при обчисленні вектору для маскування;

9) γ_2 — параметр, який застосовують для обчислення норми перетворень та обчислення молодшої та старшої частин поліномів;

10) d — кількість бітів для завдання молодшої частини компонентів

вектору $\mathbf{t}_0$, який входить до складу відкритого ключа;

11) $HASH\_OCTETS$ — кількість октетів в геш-значенні;

12) $SEED\_OCTETS$ — кількість октетів у псевдовипадкових послідовностях.

Функція гешування

Для отримання тестових векторів для алгоритму «Вершина» у якості геш-функції використовують алгоритм національного стандарту гешування ДСТУ 7564:2014 («Купина»). Функція стиснення алгоритму Купини складається з двох фіксованих підстановок (S-блоків), структура яких запозичена у шифра «Калина». Однак самі розробники зазначають, що насправді для обчислення геш-значення можна застосовувати й інші дозволені алгоритми обчислення геш-значення. Такою є, наприклад, геш-функція SHAKE-256.

Функція створення псевдовипадкових послідовностей

Ще однією відмінністю є те, що замість геш-функції SHAKE-128, для створення псевдовипадкових послідовностей обрано алгоритм національного стандарту потокового шифрування ДСТУ 8845:2019 («Струмок»). Для обчислення псевдовипадкових послідовностей також можна застосовувати інші дозволені алгоритми, наприклад, алгоритми AES та SHAKE-256. В залежності від обраного алгоритму визначають розмір блоку в октетах, адже псевдовипадкові послідовності в алгоритмі «Вершина» формуються блоками. Так автори вказують, що для шифру «Струмок» розмір блоку дорівнює 128 бітів, для AES — 64 біти, а для SHAKE-256 — 136 бітів.

Алгоритми цифрового підпису «Вершина»

Всі алгоритми подані у вигляді псевдокоду, що не завжди є зручним для реалізації. Звісно, з одного боку, це зручно для програмістів. Однак з іншого боку, всі реалізації повинні чітко відповідати саме псевдокоду наведеному у стандарті. Це унеможливлює вибір реалізацій алгоритмів, які будуть, наприклад, використовувати менший об'єм пам'яті, чи швидше працювати. Адже для різних сценаріїв є доцільними різні

реалізації алгоритмів і не варто обмежуватись лише одним. Розробники намагалися обирати найефективніші алгоритми, але математика розвивається і виникатимуть кращі і швидші схеми виконання тих самих операцій.

Передобчислення

В алгоритмі «Вершина» використовують такі передобчислення:

1) ρ - випадковий рядок для матриці $\mathbf{A}$;

2) ρ_1 - випадковий рядок для векторів $\mathbf{s}_1$, $\mathbf{s}_2$;

3) key - випадковий рядок, який є частиною особистого ключа sk;

4) $z256$, $z256\_$, $z512$, $z512\_$ - масиви, які використовуються для прямого і оберненого NTT перетворень і відповідно значення $n256\_ = 256^{-1} \pmod q$ та $n512\_ = 512^{-1} \pmod q$, які також використовують для NTT перетворень.

Передобчислення пришвидшують роботу алгоритму, але збільшують необхідний об'єм пам'яті, що може бути критичним для пристроїв з малою кількістю пам'яті.

Формування цифрового підпису

Алгоритм «Вершина» передбачає 2 варіанта формування підпису: однаковий для однакових повідомлень та кожного разу різний для однакових повідомлень. Це досягається введенням нової змінної $variant$ та використанням випадкового компоненту певної довжини, якщо обрано варіант з різними підписами.

Варто зазначити ще одну особливість алгоритму формування підпису у алгоритмі «Вершина». Вона полягає у тому, що автори алгоритму «Вершина» знехтували однією з перевірок, які відбуваються для постановки правильного підпису.

Твердження 3.1. *Якщо* $\|c\mathbf{s}_2\|_\infty \leq \beta$ *з імовірністю близькою до 1 та* $\|\mathbf{r}_0\|_\infty < \gamma_2 - \beta$, *то* $\mathbf{r}_1 = \mathbf{w}_1$.

Доведення. Для доведення скористаємося тим, що $\|c\mathbf{s}_2\|_\infty \leq \beta$ з імовірністю близькою до 1.

Якщо це справджується і

$$\|\mathbf{r}_0\|_\infty = \|LowBits_q(\mathbf{w} - c\mathbf{s}_2,\, 2\gamma_2)\|_\infty < \gamma_2 - \beta$$

то виконується рівність:

$$\mathbf{r}_1 = HighBits_q(\mathbf{w} - c\mathbf{s}_2,\, 2\gamma_2) = HighBits_q(\mathbf{w},\, 2\gamma_2) = \mathbf{w}_1$$

□

Отже виконання умови $\|\mathbf{r}_0\|_\infty < \gamma_2 - \beta$ завжди гарантує те, що $\mathbf{r}_1 = \mathbf{w}_1$. У версії алгоритму Dilithium, яка надана на 3-й раунд конкурсу NIST, ця перевірка також не використовується. Однак для того щоб позбутися умови $\mathbf{r}_1 = \mathbf{w}_1$ необхідно виконання умови на параметри, яку розглянуто у пункті 3.2.

3.2 Аналіз властивостей та визначення рівней стійкості алгоритму «Вершина»

Для подальшої роботи буде необхідна таблиця значень параметрів для різних режимів роботи алгоритму. Таблиця В.1 наведена у додатку В.

Зосередимося спочатку на самих параметрах. У таблиці В.1 наявний параметр β. Надалі буде показано, що чим меншим є значення β тим менше повторів потрібно буде зробити до вироблення правильного підпису. Однак виконання умови $\|c\mathbf{s}_2\|_\infty \leq \beta$ є необхідним для досягнення ідеального розподілу на множині дійсних підписів. Тривіальним рішенням є обрати границю для значення β як $w_c \cdot \eta$. І як можна побачити з таблиці саме таким чином було обрано значення параметру β для перших двох режимів роботи алгоритму «Вершина». Це саме ті режими для яких параметри рівні параметрам алгоритму Dilithium. Для нових режимів роботи автори алгоритму «Вершина» вирішили значно зменшити значення β. Замість $\beta = 385$ для 3-го режиму вони обрали

значення $\beta = 62$, а замість 236 для 4-го режиму — 74.

Як зазначають автори Dilithium у своїй роботі [36] менше значення параметра β дійсно можна обрати з умови

$$Pr_{\mathbf{s},c}[\|c\mathbf{s}_2\|_\infty > \beta] \approx \varepsilon$$

де ε є достатньо малим (наприклад, приблизно 2^{-80}).

Звісно значення ε можна обирати і більшим за 2^{-80}. Однак хоча ще не відомо атаки, яка б це використовувала, надмірне його збільшення не рекомендується. З іншого боку іноді може бути обґрунтовано сценарій за якого потребується зменшити параметр β і тим самим пришвидшити формування підпису, наприклад, вдвічі.

Обчислення довжин ключів та підпису.

Одним з найважливіших показників, які характеризують схеми цифрового підпису, є розміри ключів та обчисленого підпису. Адже за наявності двох однакових за стійкістю схем підпису доцільним буде обрати схему з меншими вказаними розмірами. Всі розміри вимірюються в октетах (байтах). Для цього використовують функцію:

$OCTETS(x)$ — функція для обчислення мінімально достатньої кількості октетів для запису змінної, яка складається із заданої кількості бітів x: $OCTETS(x) = \lceil x/8 \rceil$.

За формулами (3.1), (3.2) та (3.3) можливо обчислити необхідні значення. Пояснення до параметрів, які використано у формулах, наведено у розділі 3.1.

Загальний розмір *відкритого ключа* дорівнює:

$$PK\_OCTETS = SEED\_OCTETS + OCTETS(k \cdot n \cdot (log_2 q - d)). \quad (3.1)$$

Загальний розмір *особистого ключа* дорівнює:

$$\begin{aligned} SK\_OCTETS = {} & 2 \cdot SEED\_OCTETS + HASH\_OCTETS \\ & + OCTETS(n \cdot ((l+k) \cdot log_2(2\eta+1) + k \cdot d)). \end{aligned} \quad (3.2)$$

Загальний розмір отриманого цифрового підпису дорівнює:

$$DS\_OCTETS = SEED\_OCTETS + OCTETS(log_2(2\gamma_1) \cdot n \cdot l) + \omega + k \cdot n/256. \tag{3.3}$$

Також важливою є **очікувана кількість ітерацій**, необхідна для формування правильного підпису. Ця характеристика обчислюється за формулою:

$$Exp.\ reps \approx e^{n \cdot \beta(l/\gamma_1 + k/\gamma_2)} \tag{3.4}$$

Таблиця 3.1 – Розміри виходу та очікувана кількість ітерацій для режимів роботи Вершина 128/64, 256/128, 384/192, 512/256 біт стійкості відповідно до класичного та квантового криптоаналізу

Mode	128/64	256/128	384/192	512/256
$pk\ size\ (bytes)$	1312	1952	4528	5824
$sk\ size\ (bytes)$	2355	3740	8673	10271
$sig\ size\ (bytes)$	2420	3293	6612	10552
$Exp.reps$	4.25	5.1	3.2	6.55

На перший погляд може здатися, що отримуємо дуже довгі ключі та підпис (особливо, якщо дивитися на числа в байтах). Однак підхід, використаний у схемі цифрового підпису Dilithium і згодом запозичений авторами «Вершини», є одним з найбільш оптимальних серед усіх.

Обґрунтування формули 3.4 наведено у специфікаціях алгоритму Dilithium [13]. Вона також справедлива і для алгоритму «Вершина» оскільки це обернена величина до ймовірності пройти всі перевірки (на схемі з додатку А див. алгоритм А.2, рядок 11). Всі ті самі перевірки наявні і у алгоритмі «Вершина».

Наступним важливим моментом є **дослідження стійкості алгоритму** до атак.

Як вже згадувалося, стійкість алгоритму «Вершина» до різних атак ґрунтується на складності таких задач як MLWE, SelfTargetMSIS та MSIS.

Задачу $MLWE_{l,k,D}$ для деякого розподілу D можна розглядати як задачу LWE розмірності $n \cdot l$ та $n \cdot k$. Існує два типи атак: основна атака (primal attack) і двоїста атака (dual attack).

Основна атака полягає у знаходженні короткого ненульового вектора в решітці

$$\mathfrak{L} = \{\mathbf{x} \in \mathbb{Z}^d : \mathbf{Mx} = \mathbf{0} \mod \mathbf{q}\},$$

де $\mathbf{M} = (rot(\mathbf{A}_{[1:m]})|\mathbf{I}_m|vec(\mathbf{t})_{[1:m]})$ — це матриця розмірності $m \times d$, $d = 256 \cdot l + m + 1$ та $m \leq 256 \cdot k$. Функція $vec(\cdot)$ відображає вектор елементів кільця у вектор, отриманий шляхом конкатенації коефіцієнтів його координат. Функція $rot(\mathbf{A})$ отримується шляхом заміни всіх елементів матриці $\mathbf{A} : a_{ij} \in \mathbb{R}_q$ на матриці розмірності 256×256, z-ий стовпець якої дорівнює $vec(x^{z-1} \cdot a_{ij})$.

В свою чергу двоїста атака полягає у знаходженні короткого ненульового вектора в решітці

$$\mathfrak{L} = \{(\mathbf{x},\mathbf{y}) \in \mathbb{Z}^m \times \mathbb{Z}^d : \mathbf{M}^T\mathbf{x} + \mathbf{y} = \mathbf{0} \mod \mathbf{q}\},$$

де $\mathbf{M} = (rot(\mathbf{A}_{[1:m]}))$ — це матриця розмірності $m \times d$, $d = 256 \cdot l$ та $m \leq 256 \cdot k$.

Розв'язок задачі SelfTargetMSIS передбачає або порушення стійкості геш-функції H, або розв'язок задачі $MSIS_{k,l+1,\zeta}$. Аналогічно як і у випадку задачі $MLWE$ задачу $MSIS_{k,l+1,\zeta}$ можна розглядати як задачу $SIS_{256\cdot k,256\cdot(l+1),\zeta}$

У таблиці 3.2 наведено результати обчислень очікованого рівня складності розв'язку задач SIS та LWE у класичній (Best Known Classical bit-cost) та квантовій (Best Known Quantum bit-cost) моделях обчислень. Також наведено оцінки рівня складності розв'язку цих задач найбільш

ефективним алгоритмом, який наразі відомий (Best Plausible bit-cost). Оскільки існує 2 типи атак при розв'язанні задачі LWE у таблиці 3.2 наведено менше значення з двох можливих.

Таблиця 3.2 – Очікуваний рівень складності розв'язку задач SIS та LWE для всіх 4-х режимів роботи алгоритму «Вершина»

Mode	128/64	256/128	384/192	512/256
BKZ block-size b to break SIS	417	602	1720	2318
Best Known Classical bit-cost	121	176	503	677
Best Known Quantum bit-cost	110	159	456	614
Best Plausible bit-cost	86	124	356	481
BKZ block-size b to break LWE	422	622	1496	2312
Best Known Classical bit-cost	123	181	437	676
Best Known Quantum bit-cost	111	164	396	613
Best Plausible bit-cost	87	129	310	479

Далі розглянуто метод оцінки стійкості алгоритму «Вершина» до атаки підробки без дійсного повідомлення. Тобто використовується значення **ентропії поліному виклику** $c \in B_\tau$.

Нагадаємо формулу ентропії, за якою будуть обчислені значення з таблиці нижче:

$$Entropy = \log \binom{256}{\tau} + \tau$$

Значення ентропії в точності відповідає кількості бітів стійкості до атак у класичній моделі ROM. Точніше складність дорівнює $O(Entropy)$. У квантовій моделі QROM складність зменшується за рахунок можливості використання алгоритму Гровера.

У таблиці 3.3 використані позначення:

1) Classical forgery attack — складність проведення класичної атаки підробки (атака підробки у моделі ROM).

2) Відповідно Quantum forgery attack — складність проведення квантової атаки підробки (атака підробки у моделі QROM).

Таблиця 3.3 – Очікуваний рівень стійкості для всіх 4-х режимів роботи алгоритму Вершина

Mode	128/64	256/128	384/192	512/256
Classical forgery attack	194	226	386	512
Quantum forgery attack	97	113	193	256

Як видно з таблиці 3.3 для трьох з 4-х рівнів стійкості запропонованих авторами «Вершини» справджується заявлений рівень стійкості. Однак режим Вершина 256/128 є більш вразливим, ніж заявлено, як до класичних атак підробки, так і до квантових. Цей набір параметрів запозичений у алгоритму Dilithium. Варіант рішення цієї проблеми запропоновано у роботі Ченга та інших [32]. Для того щоб позбутися цього недоліку вони запропонували дещо інший метод переривань і замість одного параметру β використовували β_1 та β_2. Це зроблено для збалансування двох проблем, для яких використовується один параметр β.

Висновки до розділу 3

У розділі проаналізовано алгоритм «Вершина» і представлено низку можливих проблем, які можуть виникнути при реалізації алгоритму, описаного в проекті національного стандарту. У результаті аналізу значень параметрів виявлено зменшення параметру β з метою

пришвидшення роботи алгоритму цифрового підпису. Виконано обчислення довжин ключів та підпису. Оцінено кількість ітерацій, яку виконає алгоритм до зупинки і формування правильного підпису. Обчислення проведено для усіх режимів роботи алгоритму «Вершина». Досліджено стійкість алгоритму до атаки підробки підпису. Виявлено, що для режиму роботи Вершина 256/128 рівень стійкості є нижчим за заявлений авторами. Обчислено складність розв'язку задач SIS та LWE згідно з параметрами усіх режимів роботи алгоритму «Вершина». Вона відповідає стійкості алгоритму «Вершина» до екзистенційної підробки та атак відновлення ключа відповідно. Для режимів роботи Вершина 128/64 та Вершина 256/128 реальний рівень стійкості у класичній моделі обчислень є нижчим за заявлений авторами.

ВИСНОВКИ

Наближення ери квантових комп'ютерів стало поштовхом розробки та стандартизації нових постквантових криптографічних алгоритмів. У першому розділі розглянуто алгоритм цифрового підпису Dilithium та наведено його модифікації. Проведено аналіз особливостей алгоритму Dilithium. Наведено його наявні варіації та наявні атаки на нього. Окрім цього, наведено теоретичні основи щодо обчислення оцінки стійкості алгоритму до:

- атак підробки без використання повідомлення;
- атак відновлення ключа;
- екзистенційної підробки підпису.

Вказано відмінність між стійкістю схеми у класичній моделі обчислень та стійкістю у квантовій моделі обчислень.

З використанням розглянутих робіт та наданих у другому розділі теоретичних відомостей отримано такі власні результати:

1) проведено аналіз алгоритму «Вершина» та значень параметрів для різних режмів роботи. У результаті аналізу виявлено зменшення параметру β у 6.2 рази порівняно з очікуваним значенням для режиму Вершина 384/192 та у 3.2 рази для режиму Вершина 512/256;

2) обчислено очікувану кількість ітерацій, яку виконає алгоритм «Вершина» до формування дійсного підпису.

3) проведено дослідження стійкості алгоритму цифрового підпису «Вершина» до атак підробки без використання повідомлення, атак відновлення ключа та екзистенційної підробки підпису. У результаті дослідження виявлено, що для режиму роботи Вершина 256/128 рівень стійкості до атаки підробки підпису без використання повідомлення є нижчим за заявлений авторами. Він складає 226 та 113 бітів стійкості до класичного та квантового криптоаналізу відповідно. Окрім цього обчислено складність розв'язку задач LWE та SIS, які відповідають

стійкості алгоритму «Вершина» до атаки відновлення ключа та екзистенційної підробки підпису відповідно. За отриманими результатами можна зробити висновок, що всі чотири режими роботи забезпечують заявлений авторами рівень стійкості до цих двох атак у квантовій моделі обчислень. Однак для режимів роботи Вершина 128/64 та Вершина 256/128 реальний рівень стійкості у класичній моделі обчислень виявився дещо нижчим за очікуваний. Також варто зауважити, що з використанням найбільш ефективного відомого на сьогодні алгоритму розв'язку задач LWE та SIS забезпечується необхідний рівень стійкості у моделі QROM для всіх режимів роботи окрім Вершина 256/128.

Оскільки алгоритм цифрового підпису «Вершина» міститься у проекті національного стандарту, то гарною їдеєю щодо напрямку подальших досліджень є оцінка стійкості алгоритму «Вершина» до інших атак та з використанням інших методів криптоаналізу. Необхідно дослідити алгоритм, наприклад, на стійкість до атак помилки чи електромагнітних атак, як це зроблено для алгоритму Dilithium. Також корисним є дослідження можливих контрзаходів проти наявних атак на алгоритм «Вершина».

ПЕРЕЛІК ПОСИЛАНЬ

1. Peter W. Shor. Polynomial-Time Algorithms for Prime Factorizationand Discrete Logarithms on a Quantum Computer.

2. Announcing Request for Nominations for Public-Key Post-Quantum Cryptographic Algorithms https://csrc.nist.gov/news/2016/public-key-post-quantum-cryptographic-algorithms

3. Dong Pyo Chi, Jeong Woon Choi, Jeong San Kim and Taewan Kim. Lattice Based Cryptography for Beginners.

4. Mikl´os Ajtai. Generating hard instances of lattice problems (Extended Abstract).

5. M. Ajtai and C. Dwork. The first and fourth public-key cryptosystems with worst-case/average-case equivalence (Extended Abstract).

6. O. Goldreich, S. Goldwasser, and S. Halevi. Public-key cryptosystems from lattice reduction problems.

7. P. Q. Nguyen and O. Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures.

8. J. Hoffstein, N. A. Howgrave Graham, J. Pipher, J. H. Silverman, and W. Whyte. NTRUSIGN: Digital signatures using the NTRU lattice.

9. J. Hoffstein, N. A. Howgrave Graham, J. Pipher, J. H. Silverman, and W. Whyte. NSS: An NTRU Lattice-Based Signature Scheme.

10. Phong Q. Nguyen. A Note on the Security of NTRUSign.

11. Craig Gentry, Chris Peikert and Vinod Vaikuntanathan. Trapdoors for Hard Lattices and New Cryptographic Constructions.

12. V. Lyubashevsky and D. Micciancio. Asymptotically efficient lattice-based digital signatures.

13. Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler and Damien Stehlé CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation.

14. Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter

Schwabe, Gregor Seiler, Damien Stehlé. CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme.

15. Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures.

16. Shi Bai and Steven D Galbraith. An Improved Compression Technique for Signatures Based on Learning with Errors.

17. Prasanna Ravi and Mahabir Prasad Jhanwar and James Howe and Anupam Chattopadhyay and Shivam Bhasin. Exploiting Determinism in Lattice-based Signatures - Practical Fault Attacks on pqm4 Implementations of NIST candidates.

18. Yarkın Doröz and Jeffrey Hoffstein and Joseph H. Silverman and Berk Sunar. MMSAT: A Scheme for Multimessage Multiuser Signature Aggregation.

19. Katharina Boudgoust and Adeline Roux-Langlois. Compressed Linear Aggregate Signatures Based on Module Lattices.

20. Dustin Moody. Announcement and outline of NIST's Call for Submissions. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/pqcrypto-2016-presentation.pdf

21. Il-Ju Kim and Tae-Ho Lee and Jaeseung Han and Bo-Yeon Sim and Dong-Guk Han. Novel Single-Trace ML Profiling Attacks on NIST 3 Round candidate Dilithium.

22. Daniel Apon and James Howe. Attacks on NIST PQC third round candidates. https://iacr.org/submit/files/slides/2021/rwc/rwc2021/22/slides.pdf

23. Prasanna Ravi, Debapriya Basu Roy, Shivam Bhasin, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Number "Not Used" Once-Practical Fault Attack on pqm4 Implementations of NIST Candidates.

24. Горбенко, Ю., Дроздова, О. (2020). Analysis of Dilithium post-quantum electronic signature resistance to fault attacks. Radiotekhnika, 3(202), 49–56.

25. Leon Groot Bruinderink and Peter Pessl. Differential fault attacks on deterministic lattice signatures.

26. Suresh Chari, Josyula R. Rao and Pankaj Rohatgi. Template Attacks.

27. Vincent Migliore , Benoit Gerard , Mehdi Tibouchi and Pierre-Alain Fouque. Masking Dilithium: Efficient Implementation and Side-Channel Evaluation.

28. Mostafa Taha and Thomas Eisenbarth. Implementation Attacks on Post-Quantum Cryptographic Schemes.

29. Jean-Sebastien Coron and Louis Goubin. On Boolean and Arithmetic Masking against Differential Power Analysis.

30. Prasanna Ravi, Mahabir Prasad Jhanwar, James Howe, Anupam Chattopadhyay and Shivam Bhasin. Side-channel Assisted Existential Forgery Attack on Dilithium - A NIST PQC candidate.

31. Eike Kiltz, Vadim Lyubashevsky and Christian Schaffner. A Concrete Treatment of Fiat-Shamir Signatures in the Quantum Random-Oracle Model.

32. Zhongxiang Zheng, Anyu Wang and Lingyue Qin. Rejection Sampling Revisit: How to Choose Parameters in Lattice-Based Signature.

33. Dominique Unruh. Post-quantum security of fiat-shamir.

34. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. Journal of Cryptology, 13(3):361–396, 2000.

35. Lov Kumar Grover (1996). "A fast quantum mechanical algorithm for database search"

36. Léo Ducas, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, Damien Stehlé.CRYSTALS – Dilithium: Digital Signatures from Module Lattices.

ДОДАТОК А ШАБЛОН АЛГОРИТМУ ЦИФРОВОГО ПІДПИСУ DILITHIUM.

Псевдокод для детермінованої і рандомізованої версій алгоритму Dilithium. Єдина відмінність між двома версіями полягає в рядку 4 алгоритму формування підпису, де $\rho^{'}$ є або функцією від ключа та повідомлення, або вибирається повністю випадково.

Алгоритм А.1. Алгоритми створення асиметричних пар ключів Gen

1: $\zeta \leftarrow \{0, 1\}^{256}$
2: $(\rho,\varsigma,K) \in \{0, 1\}^{256\times3} := H(\zeta)$
3: $(\mathbf{s}_1,\mathbf{s}_2) \in S_\eta^l \times S_\eta^k := H(\varsigma)$
4: $\mathbf{A} \in R_q^{k\times l} := ExpandA(\rho)$
5: $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
6: $(\mathbf{t}_0, \mathbf{t}_1) := Power2Round_q(\mathbf{t},d)$
7: $tr \in \{0, 1\}^{384} := CRH(\rho\|\mathbf{t}_1)$
8: **return** $(pk = (\rho,\mathbf{t}_1),\ sk = (\rho,K,tr,\mathbf{s}_1,\mathbf{s}_2,\mathbf{t}_0))$

Алгоритм А.2. Алгоритм формування підпису $Sign(sk,\ M)$

1: $\mathbf{A} \in R_q^{k\times l} := ExpandA(\rho)$
2: $\mu \in \{0, 1\}^{384} := CRH(tr\|M)$
3: $\kappa := 0,\ (\mathbf{z},\ \mathbf{h}) :=\perp$
4: $\rho^{'} \in \{0, 1\}^{384} := CRH(tr\|M)\ \ (or\ \rho^{'} \leftarrow \{0,1\}^{384})$
5: **while do**$(\mathbf{z},\ \mathbf{h}) :=\perp$
6: $\quad \mathbf{y} \in S_{\gamma_1-1}^l := ExpandMask(\rho^{'},\kappa)$
7: $\quad \mathbf{w} := \mathbf{A}\mathbf{y}$
8: $\quad \mathbf{w}_1 := HighBits_q(\mathbf{w},2\gamma_2)$
9: $\quad \widetilde{c} \in \{0,1\}^{256} := H(\mu\|\mathbf{w}_1)$
10: $\quad c \in B_\tau := SampleInBall(\widetilde{c})$

11: $\mathbf{z} := \mathbf{y} + c\mathbf{s}_1$

12: $\mathbf{r}_0 := LowBits_q(\mathbf{w} - c\mathbf{s}_2, 2\gamma_2)$

13: **if** $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ *or* $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$ **then** $(\mathbf{z}, \mathbf{h}) := \perp$

14: **else**

15: $\mathbf{h} := MakeHint_q(-c\mathbf{t}_0, \mathbf{w} - c\mathbf{s}_2 + c\mathbf{t}_0, 2\gamma_2)$

16: **if** $\|c\mathbf{t}_0\|_\infty \geq \gamma_2$ *or* $\#\ 1's\ in\ \mathbf{h} > \omega$ **then** $(\mathbf{z}, \mathbf{h}) := \perp$

17: $\kappa := \kappa + l$

18: **return** $\sigma = (\mathbf{z}, \mathbf{h}, \widetilde{c})$

Алгоритм А.3. Алгоритми перевірки підпису $Verify(pk, M, \sigma = (\mathbf{z}, \mathbf{h}, \widetilde{c}))$

1: $\mathbf{A} \in R_q^{k \times l} := ExpandA(\rho)$

2: $\mu \in \{0, 1\}^{384} := CRH(CRH(\rho \| \mathbf{t}_1) \| M)$

3: $c := SampleInBall(\widetilde{c})$

4: $\mathbf{w}_1^{'} := UseHint_q(\mathbf{h}, \mathbf{Az} - c\mathbf{t}_1 \cdot 2^d, 2\gamma_2)$

5: **return** $[\|\mathbf{z}\|_\infty < \gamma_1 - \beta]$ *and* $[\widetilde{c} = H(\mu \| \mathbf{w}_1^{'})]$ *and* $[\#\ 1's\ in\ \mathbf{h} \leq \omega]$

ДОДАТОК Б СХЕМИ ЕКЗИСТЕНЦІЙНИХ ПІДРОБОК АЛГОРИТМУ DILITHIUM

Алгоритм Б.4. Схема підробки, яку запропонували Бруйндерінк і Песл для версії алгоритму Dilithium, представленної на 2-му раунді конкурсу

1: $tr \in \{0,1\}^{384} := CRH(\rho \| \mathbf{t}_1)$
2: $K \leftarrow \{0,1\}^{256}$
3: $\mathbf{u} := \mathbf{A}\mathbf{s}_1 - \mathbf{t}_1 \cdot 2^d$
4: $\mathbf{A} \in \mathbb{R}_q^{k \times l} := ExpandA(\rho)$
5: $\mu \in \{0,1\}^{384} := CRH(tr \| M)$
6: $\kappa := 0$, $(\mathbf{z}, \mathbf{h}) := \perp$
7: **while** **do** $(\mathbf{z}, \mathbf{h}) := \perp$
8: $\quad \mathbf{y} \in S^l_{\gamma_1 - 1} := DeterministicSample(K \| \mu \| \kappa)$
9: $\quad \mathbf{w} := \mathbf{A}\mathbf{y}$
10: $\quad \mathbf{w}_1 := HighBits_q(\mathbf{w}, 2\gamma_2)$
11: $\quad c \in B_\tau := H(\mu \| \mathbf{w}_1)$
12: $\quad \mathbf{z} := \mathbf{y} + c\mathbf{s}_1$
13: $\quad \mathbf{h} := MakeHint(-c\mathbf{u}, \mathbf{w} + c\mathbf{u})$
14: $\quad$ **if** $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ **then** $(\mathbf{z}, \mathbf{h}) := \perp$
15: $\quad$ **else**
16: $\quad\quad$ **if** not $Verify(pk, M, (\mathbf{z}, \mathbf{h}, c))$ **then** $(\mathbf{z}, \mathbf{h}) := \perp$
17: $\quad \kappa := \kappa + 1$
18: **return** $\sigma = (\mathbf{z}, \mathbf{h}, c)$

У третьому раунді автори алгоритму Dilithium дещо змінили алгоритм обчислення поліному $c \in B_\tau$. Ця схема підробки все ще може застосовуватись і до оновленого алгоритму за умови відповідних змін у обчисленні поліному c.

Алгоритм Б.5. Схема підробки, яку запропонували Раві та ін. для версії алгоритму Dilithium, представленної на 2-му раунді конкурсу

1: $\mathbf{A} \in \mathbb{R}_q^{k \times l} := ExpandA(\rho)$

2: $tr \in \{0,\,1\}^{384} := CRH(\rho \| \mathbf{t}_1)$

3: $\mu := CRH(tr \| M)$

4: $\widetilde{\mathbf{y}} \leftarrow S^l_{\gamma_1 - 1}$

5: $\widetilde{\mathbf{w}} := \mathbf{A}\widetilde{\mathbf{y}}$

6: $\widetilde{\mathbf{w}}_1 := HighBits_q(\widetilde{\mathbf{w}}, 2\gamma_2)$

7: $c \in B_\tau := H(\mu \| \widetilde{\mathbf{w}}_1)$

8: $\widetilde{\mathbf{z}} := \widetilde{\mathbf{y}} + c\widetilde{\mathbf{s}}_1$

9: $$\widetilde{\mathbf{h}} = \begin{bmatrix} h_{10} & \cdots & h_{1(n-1)} \\ h_{20} & \cdots & h_{2(n-1)} \\ & \cdots\cdots & \\ h_{k0} & \cdots & h_{k(n-1)} \end{bmatrix} = \begin{bmatrix} 0 & \cdots & 0 \\ 0 & \cdots & 0 \\ & \cdots & \\ 0 & \cdots & 0 \end{bmatrix}$$

10: $\mathbf{\Theta} = UseHint_q(\widetilde{\mathbf{h}}, \mathbf{A}\widetilde{\mathbf{z}} - c\widetilde{\mathbf{t}}_1 \cdot 2^d, 2\gamma_2)$

11: **for** $i = 1$ *to* k **do**

12: **for** $j = 0$ *to* $n - 1$ **do**

13: **if** $\Theta_{ij} \neq [\widetilde{\mathbf{w}}_1]_{ij}$ **then**

14: $h_{ij} = 1$

15: **if** $\mathbf{\Theta} \neq \widetilde{\mathbf{w}}_1$ *or* $\|\widetilde{\mathbf{z}}\|_\infty \geq \gamma_1 - \beta$ *or* $\#\ 1's\ in\ \widetilde{\mathbf{h}} > \omega$ **then**

16: *Go to* 4

17: **else**

18: **return** $\sigma = (\widetilde{\mathbf{z}}, \mathbf{h}, c)$

Ця схема підробки також може застосовуватись і до оновленого алгоритму за умови відповідних змін у обчисленні поліному c.

ДОДАТОК В ТАБЛИЦЯ ЗНАЧЕНЬ СИСТЕМНИХ ТА ДОДАТКОВИХ ПАРАМЕТРІВ АЛГОРИТМУ «ВЕРШИНА»

Таблиця В.1 – Значення системних та додаткових параметрів цифрового підпису для режимів роботи Вершина 128/64, 256/128, 384/192, 512/256 біт стійкості відповідно до класичного та квантового криптоаналізу

Mode	128/64	256/128	384/192	512/256
n	256	256	512	512
q	8380417	8380417	8380417	8380417
γ_1	131072	524288	524288	524288
γ_2	95232	261888	261888	261888
(k, l)	(4, 4)	(6, 5)	(7, 5)	(9, 8)
η	2	4	5	2
w_c	39	49	77	118
β	78	196	62	74
d	13	13	13	13
ω	80	55	150	230
$SEED\_OCTETS$	32	32	48	64
$HASH\_OCTETS$	32	64	96	128