НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

СУЧАСНІ АЛГЕБРАЇЧНІ КРИПТОСИСТЕМИ

КОМП'ЮТЕРНИЙ ПРАКТИКУМ

Дослідження сучасних алгебраїчних криптосистем

Виконали:
Волинець Сергій ФІ-42мн
Сковрон Роман ФІ-42мн

Київ — 2025

Зміст

1 Мета проведення комп'ютерного практикуму

Дослідження особливостей реалізації сучасних алгебраїчних криптосистем на прикладі учасників першого раунду процесу стандартизації постквантової криптографії (NIST PQC).

2 Постановка задачі

Розробити програмну реалізацію обраного криптографічного алгоритму. Реалізація повинна містити всі можливі варіанти алгоритму. Коректність реалізації підтвердити за допомогою тестів, які використовують за наявності офіційні тестові вектори або офіційну реалізацію. Знайти схожі алгоритми та провести порівняльний аналіз швидкодії за різних умов та використання модифікацій складових частин. Навести повний теоретичний опис алгоритму з усіма деталями та відомими результатами досліджень. Провести теоретичний порівняльний аналіз обраного алгоритму зі схожими алгоритмами та дослідити можливість перенесення відомих атак на обраний алгоритм.

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

Хід виконання роботи почався з вибору криптографічного алгоритму для розробки програмної реалізації. Обравши алгоритм цифрового підпису CRYSTALS-Dilithium [DKL+17], було проведено детальний теоретичний аналіз принципів його роботи та механізмів безпеки, що закладені в алгоритмі. Далі, було реалізовано всі можливі варіанти алгоритму, а саме спрощену, та розширену версії. Окрім того, варти уваги той факт, що алгоритми, можуть мати екілька рівнів безпеки, в залежності від значення глобальних змінних.

Під час розробки виникли деякі труднощі, зокрема, з оптимізацією швидкодії алгоритму для різних умов та для різних обсягів даних. Для вирішення цих проблем було досліджено еталонну реалізацію наведену в документації, разом з описом.

4 Опис обраного криптографічного алгоритму та його складових частин

Алгоритм підпису Dilithium є частиною класу постквантових криптографічних алгоритмів, що забезпечує цифрові підписи з високим рівнем безпеки. В його основі лежать такі примітиви, як множення матриць, бітові та байтові операції, тощо. Основні складові частини, що описані в протоколі до алгоритму, це генерація ключів, підписування та перевірка підпису.

На початку генерується секретний ключ і публічний ключ. Секретний ключ включає випадкові значення, які генеруються з використанням сід-значень ρ і K. Публічний ключ $\mathbf{t}$ обчислюється як результат множення матриці $\mathbf{A}$ на вектор $\mathbf{s_1}$, після чого застосовується функція Power2Round для створення зменшеного представлення. Таким чином, публічний ключ містить значення $\mathbf{t_1}$, яке використовуватиметься в наступних етапах підписування та перевірки.

Для створення підпису, підписувач використовує секретний ключ і повідомлення M, для якого обчислюється його хеш μ. Потім генерується маска для випадкових чисел, що використовується в процесі підпису. Підпис складається з двох частин: вектора $\mathbf{z}$ і допоміжної інформації $\mathbf{h}$, яка містить позиції переповнень у підписі, і контрольного значення c, яке використовується для підтвердження дійсності підпису. Щоб уникнути використання чисто випадкових значень для підпису використовується геш функція SHAKE-256, що дозволяє ортимати геш значення довільної довжини.

Під час перевірки підпису за допомогою публічного ключа $\mathbf{t_1}$ і повідомлення M перевіряється правильність обчислень, а саме: чи співпадають контрольні значення та чи відповідає підпис заданим критеріям. Процес верифікації включає перевірку переповнень у векторі $c\mathbf{t_0}$ і коректність хешування, щоб забезпечити, що підпис не був підроблений.

Проблеми можуть виникати через необхідність значної кількості обчислень, особливо при створенні та перевірці підписів. Для оптимізації використовується спеціалізоване представлення матриць у вигляді NTT (Number Theoretic Transform), що дозволяє значно зменшити обсяг обчислень, необхідних для множення матриць і хешування. Однак, зменшення розміру публічного ключа за рахунок відкидання деяких коефіцієнтів може ускладнити перевірку

підпису, оскільки потрібна додаткова інформація про позиції переповнень, яка передається у вигляді допоміжних даних в підписі.

Таким чином, кожен з етапів алгоритму — від генерації ключів до підписування та перевірки підпису — має свої виклики, пов'язані з оптимізацією швидкості, збереженням пам'яті та забезпеченням високого рівня безпеки. Алгоритм Dilithium вирішує ці проблеми через використання ефективних математичних інструментів, зокрема NTT.

5 Порівняльний аналіз швидкодії обраного алгоритму зі схожими алгоритмами

Алгоритм підпису Dilithium був розроблений як одна з найбільш ефективних постквантових схем підпису, що поєднує хорошу швидкодію та оптимізацію для зменшення розмірів публічних ключів і підписів. Щоб порівняти його з іншими алгоритмами постквантового підпису, важливо розглянути ключові параметри: розміри підпису та публічного ключа, швидкість підписування та перевірки, а також складність реалізації.

Однією з основних переваг Dilithium є використання операцій NTT (Number Theoretic Transform), що дозволяє значно прискорити операції множення поліномів порівняно з іншими методами, такими як прямі множення. Для оптимізації цих операцій Dilithium пропонує використовувати цілісні арифметичні операції на процесорах з підтримкою AVX2. В порівнянні з іншими постквантовими схемами, такими як NTRU або NTS, які вимагають використання гауссового вибірки, що є складнішим і повільнішим процесом, Dilithium показує кращу швидкодію, оскільки він використовує лише рівномірний розподіл, що значно зменшує складність реалізації та тестування. Середній час підписування для Dilithium в два рази швидший за схеми на основі NTRU, а перевірка підпису займає менше часу завдяки зменшеному розміру публічного ключа і використанню ефективних трансформацій.

У порівнянні з іншими криптографічними схемами, Dilithium має середній розмір підпису близько 2.7KB, що є більшим за деякі альтернативи, такі як схеми на основі багаточленових підписів, де підпис може бути меншим за 100 байт. Однак у порівнянні з іншими лінійними схемами підпису на основі решіток (наприклад, NTRU), де підписи можуть досягати 1.5KB, Dilithium пропонує найкращий баланс між розміром і швидкодією. Крім того, публічні ключі в Dilithium мають розмір, який також зменшено завдяки оптимізації параметрів, зокрема розміру коефіцієнтів у публічному ключі. У схемах, таких як Ring-LWE або Multivariate підписи, публічні ключі можуть бути значно більшими, що знижує ефективність їх використання в системах з обмеженими ресурсами.

У порівнянні з хешованими підписами, які мають невеликі публічні ключі (менше 100 байт), але значно більші підписи (30-40KB), Dilithium пропонує значно кращу швидкість підписування, що в середньому в 50 разів швидше. Інші схеми, такі як Multivariate Signature Schemes, можуть мати ще менші підписи, але потребують значно більших публічних ключів (понад 100KB) і часто мають меншу ефективність через велику кількість обчислень, необхідних для їх створення.

Загалом, Dilithium виділяється завдяки оптимальному балансу між швидкістю, розмірами ключів та підписів, що робить його потужним кандидатом для широкого використання в постквантових криптографічних системах.

6 Огляд наявних результатів досліджень обраного алгоритму

У дослідженнях самих авторів, що стосуються алгоритму Dilithium, важливу роль відіграє редукція ґратки та її зв'язок з розв'язуванням задачі пошуку короткого вектора в ґратці (SVP). Основним методом для знаходження коротких векторів у Евклідових ґратках є алгоритм BKZ (Block-Korkine-Zolotarev), який застосовує розв'язок задачі SVP для блоків розміром b. Безпека схеми Dilithium базується на складності вирішення задачі SVP для великих розмірів блоків b, оскільки час виконання класичних і квантових рішень цієї задачі є експоненційним відносно b. Найкращі відомі алгоритми для вирішення задачі SVP на класичних та квантових комп'ютерах мають складність порядку $2^{c_C b}$, де c є константами для класичних ($c_C \approx 0.292$) та квантових рішень ($c_Q \approx 0.265$). Тому зміна розміру блоку b без суттєвого прориву в теорії не дозволить значно зменшити складність.

Щодо алгебраїчних атак на систему Dilithium, дослідження показують, що використовувані параметри є стійкими до основних типів атак, таких як прямі та дуальні атаки на основі

матриць LWE (Learning With Errors). Крім того, алгебраїчні атаки на криптографічні схеми, такі як Ring-LWE, менш ефективні в разі використання Multivariate LWE (MLWE), що віддаляє систему від слабких структур на решітках. Навіть атаки, засновані на пошуку коротких векторів за допомогою алгоритмів BKZ, не дають суттєвих переваг без використання дуже великих параметрів блоку b. Враховуючи ці аспекти, атаки на схему Dilithium виявляються малоймовірними за умови належно налаштованих параметрів.

Сьогодні існують досліження інших авторів, наприклад [KPLG24] яка пропонує дві атаки на підпис, які послідовно виправляють підробку до тих пір, поки верифікація не буде успішою. Це є приклади атак з використанням побічної інформації, наприклад [RJH+18]. Існує і специфічна атака на основі витоку електроенергі [WGL+23], що є атакою на апаратне забезпечення.

7 Опис власних тестів, які проводилися з метою перевірки коректності реалізованої програми

Для проведення тестів, було переглянуто яким чином перевіряється коректність еталонної імплементації. В результаті, було визначено, що автори один раз виконують підписування та перевірку, для одного повідомлення, з подальшим заміром швидкодії кожного з трьох основних алгоритмів. Для наочності, результати швидкодії еталонної реалізації наведені в таблиці (*tab.* 1) разом з замірами нашої реалізації спрощеної схеми. Для більш точних оцінок, було проведено заміри на 100 випадкових текстах.

Алгоритм	Еталонна розширена реалізація	Наша спрощена реалізація
GEN	0.05696 мс	6.1304 с
Sign	0.3186 мс	15.0656 с
Verify	0.06625 мс	3.8037 с

Табл. 1: Результати дослідження швидкодії

Разом із замірами швидкодії, проводилися заміри коректності. В результаті, було визначено, що алгоритм працює правильно. Нажаль, заміри для розширеної реалізації не вдалося провести, в наслідок того, що створена реалізація містить недоліки, які перешкоджають створенню правильного підпису. Ця проблема була виявлена при перевірці коректності.

8 Детальний опис особливостей реалізації та приклади застосування

При реалізації та дослідженні еталонної імплементації, було виявлено, що деякі змінні є глобальними та постійними, а отже немає потреби передавати їх в кожну функцію по ланцюжку, що збільшує зрозумілість коду. Основними реалізованими алгоритмами є такі алгоритми як Gen, Sign, Verify. Для їх роботи було реалізовано додаткові алгоритми:

1. matrix_mul, matrix_const_mul, matrix_sum, matrix_sub – для операцій з матрицями поліномів, таких як множення, множення на константу, додавання і віднімання.
2. modpm – реалізує операцію $a \mod^{\pm} q$.
3. random_polynomial_Zq – генерує випадковий многочлен з коефіцієнтами, обраними з скінченного поля $\mathbb{Z}_q$. Залежно від заданих параметрів, коефіцієнти створеного полінома можуть бути обмежені деяким малим значенням для обмеження шуму.
4. Norm_polyvec, Norm_polyvec_check, Norm_matr , Norm_matr_check
5. decompose, HighBits, LowBits – використовуються для розкладання заданого значення за модулем Q на дві складові, r_1 і r_0.

Як приклад застосування можна розглянути наступний код, що знаходить підпис до двох повідомлень і перевіряє їх.

```
print("Key generation...")
pk, sk = Gen()
print("Key is generated")
print("")

M1 = "Some text1"
```

```
M2 = "Some text2"
print("Sign first message...")
sigma1 = Sign(sk=sk, M=M1)
print("Sign second message...")
sigma2 = Sign(sk=sk, M=M2)
print("Messages are signed...")
print("")

print("Verify signature of the messages")
print(f"First signature correctness:", Verify(pk=pk, M=M1, sigma=sigma1))
print(f"Seccond signature correctness:", Verify(pk=pk, M=M2, sigma=sigma2))
print("")
```

9 Висновки до роботи

У процесі виконання роботи було реалізовано криптографічний алгоритм цифрового підпису CRYSTALS-Dilithium, що є частиною постквантових криптосистем, і проведено детальний теоретичний аналіз його принципів і механізмів безпеки. Було розглянуто різні варіанти реалізації алгоритму, включаючи спрощену і розширену версії, а також виконано оптимізацію швидкодії для різних обсягів даних. Під час розробки виникли труднощі, зокрема, при оптимізації швидкодії. Результати тестів підтвердили коректність реалізації алгоритму, хоча були зафіксовані проблеми з розширеною реалізацією через недоліки в коді.

Порівняльний аналіз показав, що Dilithium ефективно поєднує хорошу швидкодію та оптимізацію розміру публічних ключів і підписів. Завдяки використанню операцій NTT (Number Theoretic Transform), цей алгоритм показує значно кращу швидкість підписування та перевірки, ніж альтернативи, такі як NTRU, а також забезпечує високий рівень безпеки завдяки складності задачі пошуку короткого вектора у ґратці.

Алгоритм Dilithium не пройшов NIST. Швидше за все, основна причине це наявні вже на той момент атаки, слабкості та незрозумілості деяких рішень, що зазначені в коментарях суддів.

Література

[DKL+17] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium algorithm specifications and supporting documentation, 2017.

[KPLG24] Elisabeth Krahmer, Peter Pessl, Georg Land, and Tim Güneysu. Correction fault attacks on randomized CRYSTALS-dilithium. Cryptology ePrint Archive, Paper 2024/138, 2024.

[RJH+18] Prasanna Ravi, Mahabir Prasad Jhanwar, James Howe, Anupam Chattopadhyay, and Shivam Bhasin. Side-channel assisted existential forgery attack on dilithium - a NIST PQC candidate. Cryptology ePrint Archive, Paper 2018/821, 2018.

[WGL+23] Huaxin Wang, Yiwen Gao, Yuejun Liu, Qian Zhang, and Yongbin Zhou. In-depth correlation power analysis attacks on a hardware implementation of CRYSTALS-dilithium. Cryptology ePrint Archive, Paper 2023/1891, 2023.