

Атаки сторонніми каналами на Dilithium та оцінка контрзаходів щодо них

Ярослав Дерев'янку¹, Олена Качко²

¹ науковий співробітник-консультант АТ "ІТ", в. Коломенська, 15, 61166, Харків, e-mail:

yarik0009258@gmail.com

² к. т. н., професор, АТ "ІТ", в. Коломенська, 15, 61166, Харків, e-mail: ekachko@gmail.com

У даній роботі надається комплексне дослідження атак сторонніми каналами (SCA – side-channel attacks) і атак помилками (FIA – fault injection attacks) на схеми на основі решітки, з акцентом на схемі підпису Dilithium, яка є провідним кандидатом в процесі стандартизації Національного Інституту Стандартизації NIST для постквантової криптографії. Також у роботі перевіряється низка індивідуальних контрзаходів, здатних забезпечити/покрацити захист від існуючих SCA/FIA. Проведена оцінка продуктивності показує, що представлені контрзаходи спричиняють розумний рівень втрати продуктивності при реальному застосуванні.

Ключові слова: криптографія; алгебраїчні решітки; атаки сторонніми каналами; атаки помилками

Вступ. Постквантовий алгоритм ЕП Dilithium стикається як з математичними, так і з фізичними атаками. Фізичні атаки включають вилучення секретної інформації з витoku сторонніми каналами, такої як час виконання, споживання електроенергії та електромагнітне випромінювання. Подібні атаки призводять до атак екзистенційної підробки або атак відновлення ключа, обидві з яких є катастрофічними для комп'ютерної системи. Існуючі дослідження підтверджують ризик витoku сторонніми каналами для Dilithium.

1. Попередні дані

Dilithium – це безпечна схема підпису на основі решітки, безпека якої базується на задачі модульного навчання з помилками (M-LWE) і модульного короткого цілого рішення (M-SIS). Існує два варіанти Dilithium: детермінований та імовірнісний/рандомізований.

Процедура генерації ключа складається з генерації екземпляра LWE t . Згодом екземпляр LWE розбивається на біти вищого та нижчого порядку t_1 і t_0 відповідно, де t_1 утворює частину відкритого ключа, а t_0 стає частиною секретного (приватного) ключа.

Процедура підписання Dilithium базується на структурі «Fiat-Shamir with Aborts», де підпис неодноразово генерується та відхиляється, доки не

задовольнить заданому набору умов. Підпис має вигляд $\sigma = (z, h, c)$.

Процедура перевірки використовує підпис σ та відкритий ключ pk для повторного обчислення полінома виклику \bar{c} . Якщо всі перевірки пройдено, то процедура перевірка успішна, інакше невдала.

2. Атаки помилками на Dilithium

Диференціальні атака помилками: Першу таку атаку запропонували Bruinderink та Pessl у роботі [1]. Атака вимагала лише однієї випадкової помилки будь-де протягом великого вікна приблизно у 68% часу виконання процедури підписання, щоб відновити повний секретний ключ.

Основна ідея такої атаки полягає в наступному. Зловмисник дозволяє цілі створити дійсний підпис для повідомлення m . Нехай відповідний підпис буде $z = s_1 \cdot c + u$. Згодом він робить запит до цілі з метою підписати те саме повідомлення, але тепер вводить випадкову помилку, щоб порушити обчислення c до c^* , але також переконується у використанні того самого попсе u для генерації помилкового підпису z^* . Різниця між правильним та помилковим підписом дає $\Delta z = s_1 \cdot \Delta c$, що можна легко розв'язати шляхом елімінації Гауса для відновлення секрету s_1 . Ця атака позначатиметься як *Загальна ДАП*.

Згодом у роботі Ravi та ін. [2] було представлено практичну атаку помилками із пропуском додавання з використанням введення електромагнітної помилки (EMFI). Вони запропонували демаскувати окремі коефіцієнти попсе u шляхом порушення кінцевої операції додавання окремих коефіцієнтів $(s_1 \cdot c)$ з попсе u . Якщо зловмисник порушує додавання першого коефіцієнта z , тобто $z[0]$, тоді Δz дає перший коефіцієнт $s_1 c$, тобто $s_1 c[0]$. Маючи достатню кількість помилкових підписів, зловмисник може побудувати лінійну систему рівнянь, яку можна повністю розв'язати, щоб відновити повний секрет s_1 . Ця атака позначатиметься як *Пропуск Додавання*. Вищезазначені атаки у стилі DFA можна застосовувати лише до детермінованого варіанту.

Атаки помилками шляхом переривання циклу: У роботі Espitau et al. [3] пропонується нова атака помилками, спрямована безпосередньо на попсе u . Пропустивши цикл, який відбирає окремі коефіцієнти u , існує висока ймовірність того, що ці не відібрані коефіцієнти мають значення 0. Якщо так, помилковий підпис z містить кілька коефіцієнтів, які є нічим іншим, як незамаскованими коефіцієнтами добутку $s_1 \cdot c$. Оскільки ця атака не включає диференційний аналіз, вона застосовна як до імовірного, так і до детермінованого варіантів Dilithium. Ця атака позначатиметься як *Переривання Циклу*.

Атаки, націлені на процедуру перевірки: Однією з головних мотивацій є примусове прийняття недійсних підписів через помилки для будь-якого повідомлення за вибором зловмисника. Однією з очевидних цілей впровадження

помилку є пропуск операції остаточного порівняння, яка визначає дійсність отриманих підписів. Ця атака позначатиметься як *Обхід_Перевірки*.

3. Атаки сторонніми каналами на Dilithium

Атаки на NTT: Існуючі роботи в основному спрямовані на поліноміальний множник, який використовується в Dilithium. Враховуючи, що NTT використовується для поліноміального множення, атаки *NTT_Витоку*, запропоновані в [4], також стають актуальними для Dilithium, хоча і з відповідними модифікаціями. Хоча докази витоку з інших операцій, таких як функції округлення (LowBits, HighBits) і вибірка з відхиленням, були показані Migliore та ін. у роботі [5], атака повного відновлення ключа, націлена на будь-яку з цих операцій на Dilithium не була виконана.

4. Захист Dilithium від атак сторонніми каналами/помилками

Захист від ДАП: Уважне спостереження за помилковими підписами, створеними атаками *Загальна_ДАП* в [1-2], показує, що створені помилкові підписи є недійсними і їх неможливо правильно перевірити. Таким чином, перевірка згенерованих підписів служить конкретним засобом протидії таким атакам. Цей контрзахід позначатиметься як *Перевірка_Після_Підпису* для Dilithium.

Захист вибірки nonce у: Атаці *Переривання_Циклу* можна легко запобігти, просто призначивши лічильник циклу для відстеження кількості коефіцієнтів вибірки u . Це гарантує, що зловмисник не зможе просто пропустити вибірку кількох коефіцієнтів. Також, можна додати другий рівень захисту шляхом ініціалізації u випадковими значеннями, щоб пропуск вибірки все ще гарантував, що u містить ненульові випадкові значення, тим самим запобігаючи атаці. Третій рівень захисту можна додати шляхом перевірки розподілу коефіцієнтів вибірки u . Цей контрзахід позначатиметься як *Захист_У*.

Захист процедури перевірки: Попередній аналіз процедури перевірки Dilithium з бібліотеки `pqm4` свідчить про те, що зловмисник може використати одну помилку пропуску, щоб пропустити перевірку обчисленого полінома виклику s . Щоб запобігти цьому, додається захист від пропуску операції порівняння, для чого використовується лічильник динамічного циклу, щоб відстежувати кількість порівнюваних значень. Якщо кількість значень для порівняння дорівнює v , тоді лічильник циклу i ініціалізується випадковим значенням $k \cdot v$, де k вибирається випадковим чином під час кожного виконання. Потім i зменшується на k для кожного значення, що порівнюється. Таким чином, якщо $i = 0$ після порівняння, то можна впевнитись, що всі значення були порівняні. Цей контрзахід позначатиметься як *Захист_Перевірки*.

Захист NTT: Для запобігання атакам *NTT_Витоку* впроваджуються контрзаходи перемішування та маскування, запропоновані Ravі та ін. у роботі [25]. Ці контрзаходи будуть позначатися як *Перемішування_NTT* та *Маскування_NTT*.

5. Експериментальна оцінка

Платформа на базі ARM Cortex-M4: Цільовою платформою є плата STM32F407G-DISC1, на якій розміщено мікроконтролер STM32F407. Тактова частота становить 24 МГц. Контрзаходи [6] були реалізовані на оптимізованій для M4 реалізації Dilithium, доступній у загальнодоступній бібліотеці rpm4.

Результати експериментів: У таблиці 1 надано дані про рівень витрат продуктивності спричинених контрзаходами перетасування та маскуванню проти атак *NTT_Vитоку* на Dilithium, який працює на пристрої ARM Cortex-M4. Н/з – незахищена реалізація, З – захищена реалізація з контрзаходами, %в – рівень затрат продуктивності у відсотках:

Таблиця 1

Порівняння продуктивності реалізацій без контрзаходів та із їх застосуванням
(перемішування та маскуванню)

Схема	Такти ($\times 10^6$)					
	Генерація ключа			Підписання		
	Н/з	З	%в	Н/з	З	%в
Перемішування_NTT						
Dilithium2	≈ 1.6	≈ 2.1	32.2	≈ 4.1	≈ 9.5	131.8
Dilithium3	≈ 2.8	≈ 3.5	24.6	≈ 6.7	≈ 14.3	112.9
Маскування_NTT						
Dilithium2	≈ 1.6	≈ 2.1	29.9	≈ 4.1	≈ 10.5	155.5
Dilithium3	≈ 2.8	≈ 3.4	22.7	≈ 6.7	≈ 16.1	140.7

На пристрої ARM Cortex-M4 можна бачити вплив на продуктивність у діапазоні 23–33% для генерації ключів і 113–156% для процедури підписання. Зауважте, що контрзаходи реалізовано на основі реалізацій NTT/INTT на основі C, що призводить до високих витрат продуктивності. Таким чином, можна отримати значно менші витрати продуктивності за умови, що захищені NTT/INTT реалізовані на асемблері.

У таблиці 2 надано дані про рівень витрат продуктивності спричинених контрзаходами *Перевірка_Після_Підпису*, *Захист_Y* і *Захист_Перевірки* на Dilithium, який працює на пристрої ARM Cortex-M4. Н/з – незахищена реалізація, З – захищена реалізація з контрзаходами, %в – рівень затрат продуктивності у відсотках.

На пристрої ARM Cortex-M4 M4 ці контрзаходи створюють дуже розумний рівень витрат продуктивності в діапазоні 6-8%, 26-37% і $\approx 0\%$ для різних наборів параметрів Dilithium. Результати вказуються на те, що за винятком контрзаходів перемішування та маскуванню, інші контрзаходи створюють доволі розумний рівень витрат продуктивності Dilithium на оцінюваній платформі.

Висновки. У роботі було представлено дослідження атак сторонніми каналами і атак помилками на Dilithium, а також розглянуто спеціальні контрзаходи для захисту від відомих атак. Оцінка продуктивності показує, що за винятком

контрзаходів перемішування та маскування, інші контрзаходи створюють доволі розумний рівень витрат продуктивності для Dilithium.

Таблиця 2

Порівняння продуктивності реалізацій без контрзаходів та із їх застосуванням

Схема	Такти ($\times 10^6$)		
	Н/з	З	%в
Перевірка_Після_Підпису (підпис)			
Dilithium2	≈ 4.1	≈ 4.4	7.5
Dilithium3	≈ 6.8	≈ 7.2	5.5
Захист_У (підпис)			
Dilithium2	≈ 4.1	≈ 5.6	36.5
Dilithium3	≈ 6.8	≈ 8.6	26.3
Захист_Перевірки (перевірка)			
Dilithium2	≈ 1.6	≈ 1.6	≈ 0
Dilithium3	≈ 2.8	≈ 2.8	≈ 0

Література

- [1] Leon Groot Bruinderink and Peter Pessl. Differential Fault Attacks on Deterministic Lattice Signatures. 2018. – pp. 21-43.
- [2] Prasanna Ravi, Mahabir Prasad Jhanwar, James Howe, Anupam Chattopadhyay, and Shivam Bhasin. Exploiting determinism in lattice-based signatures: practical fault attacks on pqm4 implementations of NIST candidates. 2019 – pp. 427–440.
- [3] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Loop-abort faults on lattice-based fiat-shamir and hash-and-sign signatures. 2016 – pp. 140-158.
- [4] Peter Pessl and Robert Primas. More practical single-trace attacks on the number theoretic transform. 2019 – pp. 130-149.
- [5] Vincent Migliore, Benoît Gérard, Mehdi Tibouchi, and Pierre-Alain Fouque. Masking dilithium. 2019 – pp. 344-362.
- [6] Prasanna Ravi, Anupam Chattopadhyay, Anubhab Baksi. Side-channel and Fault-injection attacks over Lattice-based Post-quantum Schemes (Kyber, Dilithium): Survey and New Results. 2022 – p. 737.

Side-channel attacks against Dilithium and countermeasures evaluation

Yaroslav Derevianko, Olena Kachko

This paper provides a comprehensive study of side-channel attacks (SCA) and fault injection attacks (FIA) on lattice-based schemes, with an emphasis on the Dilithium signature scheme. The paper also examines a number of individual countermeasures capable of providing/improving protection against existing SCA/FIAs. The performance evaluation shows that the considered countermeasures cause a reasonable level of performance loss in real-world applications.

Отримано 17.03.23