**Рівномірна вибірка Гауса та його застосування для схеми підпису Falcon**

**Анотація:**

Вибірка Гауса для цілих чисел є ключовим інструментом у криптографії на основі решітки, але виявилось напрочуд складним завданням виконувати вибірку у типовий, ефективний спосіб з доведеною захищеністю. У цій роботі представлено блоковий фреймворк для генерації дискретних компонентів Гауса з довільним центром та стандартним відхиленням. Даний фреймворк надзвичайно простий, і саме ця простота дозволила зробити його простим у впровадженні, доказово захищеним, портативним, ефективним та доказово стійким проти часових атак. Відбірник, що буде розглядатися, є гарним кандидатом для будь-якого вибіркового аналізу, нещодавно його було впроваджено у схему ЕП Falcon. Другий внесок спрямований на систематизацію виявлення помилок реалізації у відбірниках Гауса. Ми пропонуємо набір статистичних випробувань для дискретних компонентів Гауса, який називається SAGA (Statistically Acceptable GAussian).

**Ключові слова:**

Криптографія, заснована на алгебраїчних решітках, Вибірка Гауса, рівномірна вибірка.

**1. Вступ**

Вибірка Гауса над цілими числами є центральним елементом криптографії на основі решіток, як в теорії, так і на практиці. Також відомо, що вибірка є процесом, який складно виконувати якісно та безпечно. Це доведено численними атаками бічними каналами на пробовідбірник Гауса, що застосовується у BLISS. З цієї причини деякі схеми обмежують або забороняють використання таких відбірників. Однак у деяких ситуаціях використання відбірника Гауса не уникнути. Найбільш яскравим прикладом є відбір проб: його проведення з іншими розподілами є поки не вирішеним питанням, за винятком одиничних випадків (вибірка з широкого класу розподілів), які спричиняють зростання складності до на виході, що, в свою чергу, призводить до зменшення рівня безпеки. Беручи до уваги незліченну кількість застосувань вибіркових процесів (хеш-сигнатури повного домену, шифрування на основі ідентичності (або IBE) , ієрархічне IBE, тощо), важливо прийти до відбірників Гауса над цілими числами, які були б не тільки ефективними, але й доказово захищеними, стійкими до часових атак та загалом простими в реалізації.

**2 Попередні дані**

**2.1 Функція Гауса**

Для з ми називаємо функцією Гауса параметрів і позначаємо через функцію, визначену над як . Зверніть увагу, що коли , це упускається в позначеннях індексу, наприклад . Параметр (відповідно, ) часто називають стандартним відхиленням (відповідно, центром) Гауса. Крім того, для будь-якої злічувальної множини ми грубо позначаємо через суму . Коли скінченна, позначаємо через і називаємо гауссовим розподілом параметрів розподіл над , визначений як Тут також, коли , ми опускаємо це в позначеннях індексу. Для позначення розподілу Бернуллі параметра використовується позначення .

**2.2 Відхилення Рені**

У цьому пункті згадується визначення відхилення Рені, яке буде активно використовуватися у доказах безпеки.

**Визначення 1 (відхилення Рені).** Нехай - два розподіли такі, що . Для визначимо відхилення Рені порядку як

До того ж, визначаємо відхилення Рені порядку як

Відхилення Рені перевіряє криптографічно корисні властивості, які буде перераховано нижче.

**Лема 2.** Для двох розподілів та двох сімейств розподілів відхилення Рені перевіряє наступні властивості:

* **Нерівність обробки даних.** Для будь-якої функції , де розподіл (відповідно, ) отримується шляхом застосування до (відповідно, ).
* **Мультиплікативність.** Припустимо, що і є спільним розподілом пари випадкових величин (). Припустимо також, що **незалежні**. Тоді позначаємо і . Тоді .
* **Збереження ймовірності.** Для будь-якої події та ,

**Лема 3.** Нехай - два розподіли однакової опори . Припустимо, що відносна похибка між і обмежена: так, що над . Тоді для :

У деяких випадках запити до розподілів не є незалежними, отже, не можна застосовувати мультиплікативність безпосередньо в наших доказах. Для обробки цих залежностей вводиться наступне твердження.

**Твердження 4.** Нехай і позначають два розподіли N-множини випадкових величин . Для припустимо, що (відповідно ) є граничним розподілом , і нехай позначає умовний розподіл , враховуючи, = . Нехай . Припустимо, що для всіх існує так, що для всіх i-множин у опорі для , обмежені першими змінними,

Тоді:

Доказ. Результат доводиться у випадку , тоді загальний випадок слідує за індукцією. Маємо:

На цьому доказ закінчується.

**2.3 Параметр згладжування**

Для параметр згладжування решітки Λ є найменшим значенням таким, що де позначає дуальну до Λ решітку. У літературі деякі визначення параметра згладжування масштабують наше визначення коефіцієнтом .

**2.4 Рівномірні (ізохронні) алгоритми**

**Визначення 5.** Нехай - це (імовірнісний або детермінований) алгоритм із набором вхідних змінних , набором вихідних змінних і нехай буде набором чутливих змінних. Ми говоримо, що є абсолютно рівномірним(ізохронним) відносно , якщо час його роботи є незалежним від жодної змінної в . Крім того, стверджується, що статистично рівномірний(ізохронний) щодо , якщо існує розподіл , незалежний від усіх змінних у , такий, що час роботи статистично близький (для чітко визначеного відхилення) до .

Зазначимо, що ми можемо визначити поняття обчислювально рівномірного(ізохронного) алгоритму. Для такого алгоритму обчислювально важко відновити чутливі змінні, навіть враховуючи розподіл часу роботи алгоритму.

**3 Відбірник**

У цьому розділі описано новий пробовідбірник із довільним стандартним відхиленням і центром. Основне припущенням в даному випадку - вважати, що всі стандартні відхилення обмежені і що центр знаходиться в [0, 1]. Іншими словами, позначаючи верхню та нижню межі на стандартному відхиленні як , ми представляємо алгоритм, який відбирає розподіл для будь-якого та .

Алгоритм вибірки називається , він описаний в Алгоритмі 1. Позначимо як алгоритм, який відбирає елемент із фіксованим напівгаусовим розподілом . Перший крок полягає у використанні BaseSampler. Потім отриманий зразок перетворюється на , де - біт, рівномірно вибраний у множині {0, 1}. Позначимо через розподіл . Розподіл є дискретним бімодальним напіврозподілом Гауса для центрів 0 і 1. Більш формально,

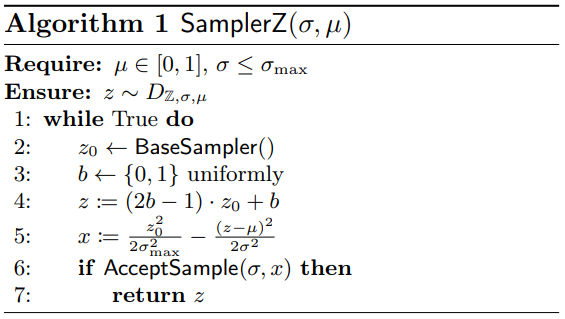


Рис. 3.1 – Алгоритм

Потім, щоб відновити бажаний розподіл для входів (), можна було б застосувати класичну техніку вибірки відкидання, застосовану до схем на основі решітки, і прийняти з імовірністю:

Елемент всередині exp обчислюється на кроці 5. Далі також вводиться алгоритм, позначений як . Він виконує вибірку відкидання (Алгоритм 2): використовуючи алгоритм , який повертає , він повертає вибірку Бернуллі з відповідною ймовірністю. Власне, щодо питань рівномірності(ізохронії), детально описаних у Розділі 6, імовірність останнього прийняття масштабується на коефіцієнт . Оскільки підпорядковується розподілу , після вибірки відкидання кінцевий розподіл () є пропорційним до , який після нормування точно дорівнює . Таким чином, за допомогою цієї конструкції можна вивести наступне твердження.

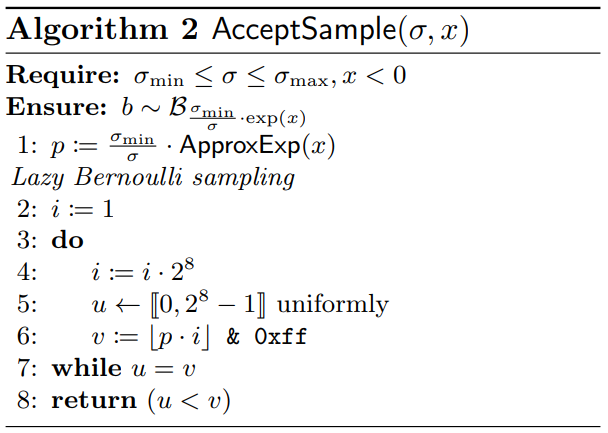


Рис. 3.2 – Алгоритм

**Твердження 6 (Правильність).** Припустимо, що всі рівномірні розподіли є досконалими і що та exp, тоді будова (в алгоритмах 1 та 2) така, що .

Для практичних реалізацій неможливо отримати ідеальні розподіли. Можливо лише отримання та exp. Розділ 6 доводить, що за певних умов на та та за певної кількості запитів вибірки, остаточний розподіл залишається невідмінним від .

**4 Доказ захищеності**

На рисунку 4.1 наведено позначення кількості викликів , а також розглянуті при створенні екземпляру для Falcon значення. Через вибірку відкидання на кроці 6, буде (потенційно нескінченна) кількість ітерацій циклу while. Пізніше в лемі 8 буде показано, що кількість ітерацій відповідає геометричному закону параметра . Позначимо як наближено максимальну кількість ітерацій. За аргументом центральної межі буде лише незначно вищою за очікувану кількість ітерацій. Щоб створити екземпляр значень на прикладі Falcon, беремо. Фактично, для параметрів Falcon.

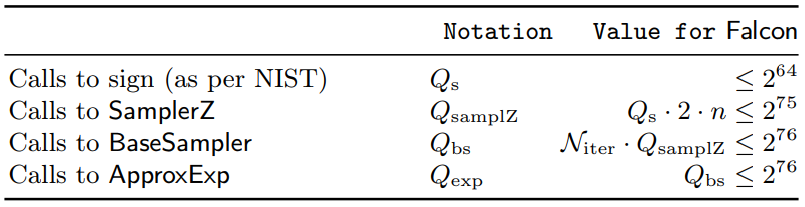


Рис. 4.1 – Кількість викликів

Наступна теорема оцінює безпеку , вона є незалежною від обраних значень кількості викликів.

**Теорема 7 (Безпека SamplerZ).** Розглянемо криптосистему, позначену , яка робить щонайбільше **незалежних** запитів до . Припустимо, що має λ-бітову захищеність від задачі пошуку. Тепер розглянемо копію останньої криптосистеми, позначеної , де єдиною відмінністю є застосування .. Якщо дотримано дві наступні умови:

тоді, буде мати (λ-2)-бітову захищеність від задачі пошуку.

Для отримання конкретних числових значень ми припускаємо, що для вихідної схеми заявлено 256 бітів захищеності, таким чином для реальної реалізації необхідно 254 біта захисту. Тоді для реалізації Falcon числові значення будуть наступними:

**4.1 Створення екземпляру**

Для досягнення умови (1) за допомогою використовується поліноміальне наближення експоненти на . Насправді можна зменшити параметр за модулем таким чином, щоб . Обчислюємо експоненціальні лишки для обчислення . Відзначаючи, що трапляються дуже рідко, тому можна наситити до 63, щоб уникнути переповнення без втрати точності.

Тут використовується інструмент апроксимації поліномів, наданий у GALACTICS. Цей інструмент генерує поліноміальні наближення, які дозволяють проводити обчислення фіксованої точності з обраним розміром коефіцієнтів та ступеня. Як приклад, для 32-розрядних коефіцієнтів і ступеня 10 отримуємо поліном , з:

Для будь-якого , підтверджує , що є достатнім для дотримання умови (1) у реалізації Falcon. Крім того, ми експериментально перевіряємо, чи підтверджує .

**4.2 Створення екземпляру**

Для досягнення умови (2) за допомогою ми спираємось на сукупну таблицю розподілу (CDT). Ми попередньо обчислюємо таблицю сукупної функції розподілу з певною точністю; тоді для отримання вибірки генерується випадкове значення в [0, 1] з такою ж точністю і повертаємо індекс останнього запису в таблиці, який перевищує це значення. У випадковий час вибірку можна зробити досить ефективно за допомогою двійкового пошуку, але реалізація з постійним часом, по суті, не має іншого вибору, як кожен раз читати всю таблицю та проводити кожне порівняння. Цей процес резюмується в алгоритмі 3. Параметри - це відповідно кількість елементів CDT та точність його коефіцієнтів. Нехай . Для визначення параметрів використовується простий скрипт, який враховуючи як вхідні дані:

1. Обчислює найменший зріз при якому відхилення Рені між ідеальним розподілом та його обмеженням до {0,. . . , w} (позначеного як ) підтверджує (;
2. Округлює таблицю щільності ймовірностей (PDT) для з бітами абсолютної точності. Це округлення виконується «розумно» шляхом усічення всіх значень PDT, крім найбільшого:
   * Для значення зрізається:
   * з метою мати розподіл імовірностей, .
3. Виводить і обчислює кінцевий

Взявши, і як вхідні дані, ми отримали .

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Експеримент показує, що для будь-якого , , що задовольняє умові 2 для реалізації Falcon.

**5 Аналіз опору часовим атакам**

Основою для припущення щодо безпеки криптографічних примітивів постквантової криптології є такі задачі решітки як: «коротке ціле рішення» (SIS), «задача найкоротшого вектора» (SVP) та «навчання з помилками» (LWE), які обчислювально є достатньо складними. В загальному випадку решітку можна визначити набором векторів, , які формують її базис, B, який визначається як . Для алгоритму Falcon припущення про стійкість генерації ключів базується на задачі NTRU. В ній стверджується, що обчислювально складною є процедура відновлення і для елемента кільця поліномів . Стійкість процедури підписання залежить від SVP, в якій стверджується, що при відомому базисі решітки важко знайти короткий вектор у ній.

В даному випадку всі операції виконуються в поліноміальному кільці . Позначимо якрозмір кільця і як –модуль. Елементи кільця можуть бути представлені як поліноми ступеня або вектори розмірності . Параметри а також рівні безпеки для кожного набору параметрів наведено в таблиці 2.1.

Таблиця 2.1 – Пропоновані параметри для схеми підпису Falcon.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Набір параметрів | Рівень  NIST | Рівень захищеності | Розмірність ( | Модуль( |
| Набір 1 | Рівень 1 | AES128 | 512 | 12289 |
| Набір 2 | Рівень 4  Рівень 5 | SHA384  AES256 | 1024  1024 | 12289  12289 |

**2.2 Схема підпису Falcon**

Falcon – геш-підписова схема, яка заснована на шифруванні на основі DLP та методах швидкої вибірки Фур’є. Завдяки цим методам, вдається забезпечити невеликі розміри закритого ключа та підвищену швидкодію процедури вибірки. Генерація ключів відбувається за Алгоритмом 1 (Рис.2.1). Після генерації ключів, отримуємо NTRU-поліноми. При цьому - особисті ключі, а – відкритий ключ.

Процес підписання показано в Алгоритмі 2 (Рис.2.2). Він відбувається шляхом знаходження короткого вектору в решітці NTRU з використанням закритого ключа. Для цього застосовується процедура ffSampling – показана в Алгоритмі 3 (Рис.2.3).

Перевірка підпису виконується шляхом перевірки модуля, який розміщується в межах необхідної межі β, за допомогою відкритого ключа. Ця процедура показана в Алгоритмі 4 (Рис.2.4).

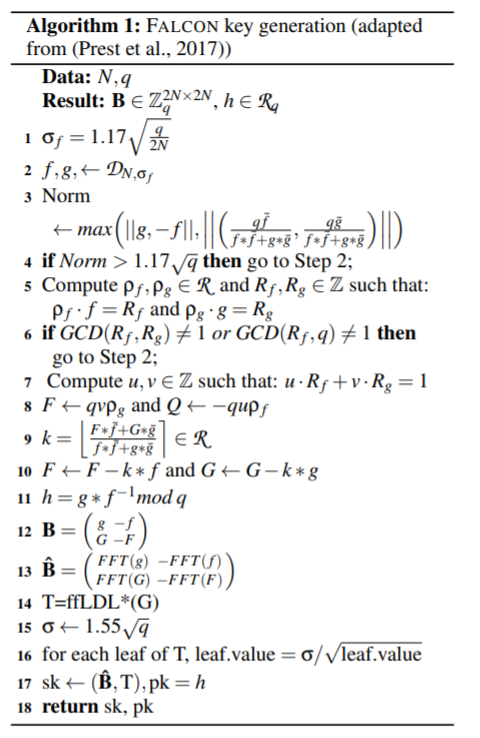


Рис. 2.1 - Алгоритм генерування підпису (Алгоритм 1)

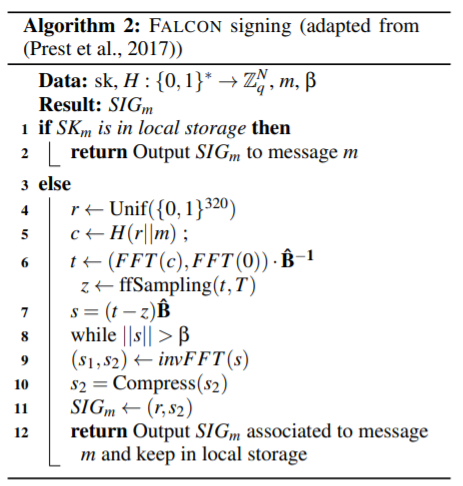


Рис. 2.2 - Алгоритм підписування (Алгоритм 2)

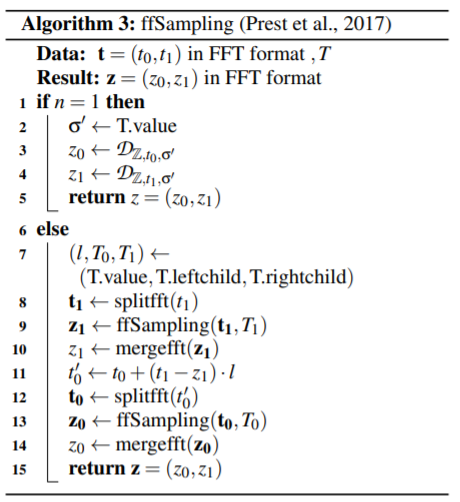


Рис. 2.3 - Алгоритм ffSampling (Алгоритм 3)

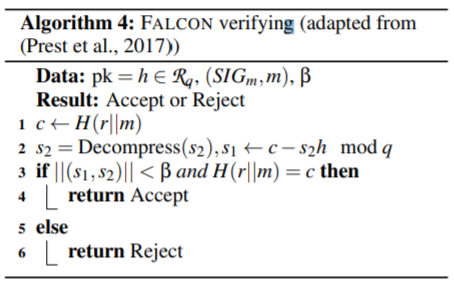


Рис. 2.4 - Алгоритм перевірки (Алгоритм 4)

**2.3 Схема підпису DLP**

Розглянемо атаку на геш-підписову схему DLP на основі GPV, запропоновану Еспітау (2016, 2018).

Модель помилки для їхньої атаки виглядає наступним чином: на початку етапу Гауссової вибірки решітки GPV, зловмисник може перервати цикл в алгоритмі підписання. Змушуючи процес перериватися на певній ітерації при вибірці векторних коефіцієнтів полінома для підпису ступеня , вони отримують помилковий підпис у підрешітці рангу .

Цей процес повторюється до моменту отримання підписів. Коли необхідна кількість помилкових підписів отримана, зловмисник має змогу згенерувати вихідну решітку і відновити короткий вектор з дуже високою ймовірністю (86%), що в свою чергу дасть змогу отримати приватний ключ, через відновлення структури решітки NTRU.

**3 Часові атаки на Falcon**

Деякі математичні компоненти, які використовуються в алгоритмі ЕЦП Falcon, є вразливими до атак, заснованих на аналізі часових показників. Це пов’язано з певними складнощами реалізації цих компонентів за постійний час. В цьому розділі коротко описано вразливі місця та запропоновано контрзаходи, щодо часових атак на них.

**3.1 Відбірник Гауса**

Саме Гаусові відбірники, як правило, є вразливим місцем для часових атак проти криптографії на решітках. Для схеми Falcon Гаусів відбірник виконує роль відбірника коротких векторів у решітках.

**3.2 Теоретичне перетворення числа (NTT)**

NTT є потенційно вразливим до часових атак, оскільки використовує велику кількість модульної арифметики, яку важко реалізувати за постійний час. У схемі Falcon NTT використовується для пришвидшення множення кільцевих многочленів.

**3.3 HashtoPoint**

Процес гешування повідомлення до точки в кільці поліномів „може бути складно здійсненним при постійному часі” – зазначають автори алгоритму Falcon. Тому функція є потенційно вразливою до часових атак.

**3.4 Запропоновані контрзаходи проти часових атак**

Одним з ефективних на даний момент заходів проти часових атак є алгоритм BlindVector, запроваджений Saarinen (2017). Він є розширеним та вдосконаленим алгоритмом перемішування Фішера-Йейтса (Fisher et al., 1938) який, як правило, використовується для ефективної випадкової перестановки коефіцієнтів вектора, розподіленого за Гаусом. У BlindVector покращено випадкові перемішування для протистояння атакам по бічних каналах. Цей алгоритм використовується у даній реалізації, крім того гарантується постійна за часом реалізацію навіть для подальшого збільшення складності атак. Цикли алгоритму є незалежними від даних, а операції завжди виконуються, незалежно від того, замінено якесь значення чи ні.

Для протидії часовим атакам, функції NTT та FFT також реалізовані постійними за часом. Це стає можливим завдяки обробці всіх необхідних змінних гілки та відмови від логіки для більш тривалих арифметичних операцій. Для зменшення впливу на продуктивність даних контрзаходів, використовується «ледаче» скорочення (lazy reduction) та векторизація SIMD. Для забезпечення можливості виконання множення двох кілець у домені NTT застосовується поточкове модульне множення, яке забезпечує обчислення кожного елементу незалежно. Дані операції є послідовними та безумовними, тому можна говорити про їх усталеність у часі. Ще однією особливістю є те, що у перетворенні застосовується вдосконалений NTT алгоритм Cooley-Tukey. Покращення застосовуються для підвищення продуктивності, увімкнення автоматичної векторизації та гарантування того, що не використовуються умовні операції розгалуження. З метою підвищення продуктивності в реалізації, що розглядається, автоматична векторизація виконується окремо від модульного скорочення. Крім того, конструкція модульного скорочення в даному випадку є спрощеним обмеженням діапазону. Це дозволяє перейти від логічних операцій до арифметичних операцій, які в більшості випадів є усталеними за часом.

Також серед запропонованих заходів є відкидання зразку. Даний конрзахід полягає в наступному – з випадкових адрес зчитується додатковий кеш з метою спотворити статистику для SCA і після цього зайві зчитування відкидаються. Діапазон норм відкидань у реалізації, що розглядається, обрано на рівні 6,25%, 12,5% та 25%. Ще одним нововведенням є ефективна конструкція методики гаусової вибірки (CDT) з постійним часом. Перевагою даної конструкції є чітка кількість операцій пошуку-зчитування, а також використання однакової арифметики, незалежно від гілки, обраної згідно з CDT. Верхня межа кількості необхідних пошукових операцій дорівнюватиме тому кожен виклик пробовідбірника доповнюється до найближчого ступеня двійки для того, щоб виконати таку ж кількість тактових циклів. Побудований таким чином пробовідбірник здатен забезпечити кращу швидкодію, ніж пробовідбірники Knuth-Yao, дискретний відбірник Ziggurat’а та відбірник Бернуллі, для виконання за усталений час.

**4 BEARZ атака на Falcon**

BEARZ – атака помилками на схему ЕЦП Falcon, запропонована Sarah McCarthy та ін. (2019). Принцип дії атаки полягає у вилученні базису через переривання рекурсії або обнулення. Атака добуває закриту ключову інформацію, шляхом спричинення помилок в роботі. Модель зловмисника передбачає, що він може пропускати команди та обнуляти змінні.

**4.1 Рекурсія алгоритму Falcon**

Виходячи з того, що алгоритм вибірки для Falcon є рекурсивною формою GPV відбірника, атака буде спиратися на переривання рекурсивного виклику на початку (Perst et al., 2017, Alg. 18). Схема Falcon має два рекурсивні виклики на верхньому рівні алгоритму ffSampling. Для даного цільового вектора , алгоритм діє спочатку на , рекурсивно, справа наліво, а потім на . Кожен елемент - і потім - безперервно ділиться на два вектори довжини поки не буде дорівнювати 2, а потім відбирає коефіцієнти Гаусового розподілу, щоб отримати вектор вибірки . Через це алгоритм верхнього рівня містить дві рекурсивні гілки. Зауважимо, що всі процедури виконуються в домені FFT. Для вектора довжини , перші значень будуть представлені реальними коефіцієнтами, а другі - уявними.

Спираючись на структуру рекурсії Falcon, для успішної атаки слід було б перервати рекурсивний виклик у необхідній точці, щоб тільки місць були б заповнені. На жаль це не є можливим через характер функцій FFT.

**4.2 FFT: злиття та розділення**

Функції злиття та розділення для схеми Falcon виконуються у домені FFT. Це спричиняє проблему для атаки помилками, оскільки нульові вхідні дані не представлені нулем у FFT. Тому є необхідність відстеження коефіцієнтів зразків під час руху вгору по рекурсивному дереву. Після отримання вектору решітки із Гаусового зразка, до нього застосовується зворотня функція FFT (. Це необхідно, щоб переконатися, що підпис не знаходиться в домені FFT. Після цього може бути використана та сама після-обробка що і для DLP-атаки.

Атаки шляхом переривання рекурсії можна класифікувати, в залежності від місця переривання. Але усі методи атаки призводять до однакового вихідного векторного формату для ; перші коефіцієнів примусово задаються нулями.

**4.2.1 Переривання другої рекурсії (для )**

Атака полягає в перериванні виклику алгоритму вибірки в кінці, після першого рекурсивного виклику (рядок 10 алгоритму 3). Після виконання атаки заповниться вибраними коефіцієнтами, а залишиться повністю нульовим. Це також можна виконати для , з метою обнулити перші коефіцієнтів.

**4.2.2 Обнулення або атака пропуску (для )**

Можливі два варіанти даної атаки. Перший - при передостанньому злитті ( = Від 256 до 512), встановіть необхідні (наприклад, перша половина, якщо ) з вихідних коефіцієнтів в нуль шляхом пропуску операцій або обнулення. Операції, що пропускаються: , з коду (Prest et al., 2017)( merge\_fft() функція у falcon\_fft.c). Другий - встановити необхідну кількість перших коефіцієнтів в нуль перед обчисленням відповідного вектора решітки, тобто перезаписати вихід відбірника - .

**4.2.3 Переривання середньої рекурсії ( )**

Даний тип атаки вимагає одноразового попереднього обчислення, проте дозволяє застосовувати помилку на стадії одновимірного Гаусового відбірника, що сприяє простоті фізичного вбудовування.

Якщо нам необхідно прирівняти до нуля ліву половину , то вектор лівої сторони (LHS) при останньому виклику merge\_fft() повинен містити у першій половині нульові значення, а у векторі правої сторони (RHS) перша половина коефіцієнтів має бути рівною другій половині. Кожен реальний коефіцієнт вектору генерується як та , де - розмірність вектора вищого рівня, і . Для того, щоб прирівняти і до нуля, встановлюється і для кожного . Тому, наприклад, для = 512, перші = 128 коефіцієнтів 256-мірного вектора LHS встановлюються рівними нулю, а перші 128 коефіцієнтів 256-мірного вектора RHS встановлюються рівними другим 128 коефіцієнтам цього ж вектора.

Щодо 256-розмірного RHS вектору - він повинен містити в першій половині значення, рівні значенням другої половини. Щоб це могло статися, 128-мірний вектор LHS повинен мати реальні значення, що дорівнювали б уявним значенням, а 128-розмірний RHS вектор повинен бути нульовим. Це зумовлено тим, що необхідно отримати та рівним для кожної ітерації . Функція merge\_fft () виконує це наступним чином: і . Щоб умова вище виконувалася, встановлено наступні правила - перша половина коефіцієнтів дорівнюватиме другій половині, і . Тому рівняння будуть просто залежати від вектора подачі LHS.

Для RHS кожну нижню гілку можна встановити в нуль. Візьмемо LHS 128-розмірний вектор: 64-мірний вектор LHS повинен мати рівні реальні та уявні значення, а RHS має бути нульовим. Будь-який рівний нулю вектор повинен мати обидва вектори подачі рівними нулю, тому гілки нижче цієї можуть бути обнулені. Цей метод може застосовуватися для будь-якого такого, що , де .

**4.3 Обробка після нападу**

Після отримання недійсного підпису атака передбачає відновлення таємного базису з нього. Наприклад, у нас мається коефіцієнтів вектора , які дорівнюють нулю. Тоді обчислюємо підпис Falcon як:

де - базова матрицею в домені FFT. Але нам потрібна лише друга половина - , де:

і так як встановлено в нуль:

Крім того, перші коефіцієнтів дорівнюють нулю, що означає:

і оскільки деякі з коефіцієнтів дорівнюють нулю, ми нарешті маємо:

Тим самим отримується підрешітка решітки, згенерованої за допомогою F. Таким чином, з кількома недійсними підписами можна знайти решітку, породжену , (де є коротким вектором у цій решітці), а завдяки алгоритму BKZ можна знайти цей короткий вектор. Після цього можна отримати з відкритого ключа , і таким чином можна знайти таємний базис NTRU решітки.

**4.4 Результати атаки**

Для перевірки моделі цієї атаки Sarah McCarthy та ін. (2019) було виконано тестування кожного методу атаки за допомогою програмного моделювання. Після успішного виконання всіх тестів, було зібрано недійсних підписів для значень нулів, наведених в таблиці 2. Для отримання полінома приватного ключа було застосовано алгоритм BKZ (FPLLL Development Team, 2016). Таблиця 2 містить часові результати запуску алгоритмів.

Таблиця 4.1 – Часи (у секундах) на застосування алгоритму BKZ для недійсних підписів для отримання базового полінома (працює на одному ядрі процесора Intel Core i7-6700HQ на частоті 2,60 ГГц.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Набір параметрів | m | l=m+1 | l=m+2 | l=m+3 |
| Набір 1 | 64 | 0.07 | 0.07 | 0.08 |
| 128 | 0.35 | 0.30 | 0.36 |
| 256 | 2.17 | 2.80 | 2.36 |
| Набір 2 | 128 | 0.28 | 0.29 | 0.35 |
| 256 | 2.24 | 2.55 | 2.67 |

**4.5 Модель помилки**

Модель помилки передбачає аналіз бічних каналів, завдяки якому може бути виявлено вікно між першим і другим рекурсивним викликом. В межах цього вікна алгоритм може бути перервано.

Для застосування атаки обнуленням можна використати момент зберігання вектору в оперативній пам’яті, обнуливши під час цього необхідні біти. (Naccache et al., 2005). Як альтернативу можна розглядати пропуск рядків. Його можна виконати за допомогою перепадів тактової частоти процесора. (Blömer et al.,2014).

**4.6 Контрзаходи проти BEARZ**

Для запобігання атаці BEARZ існують певні контрзаходи. Одним з найпростіших методів виявлення атак помилками є подвійне обчислення підпису. Це подвоює час підписання. При перевірці відразу після підписування підписувач може переконатися в тому, що обладнання не піддавалося атакам помилками. (Bruinderink та Pessl, 2018). Також простим та ефективним методом виявлення ВEARZ атаки є перевірка того, що вибраний вектор не йде до нуля в певній точці вздовж своєї довжини в кінці ffSampler алгоритму.

**5 Результати та оцінка**

Даний розділ показує результати роботи модифікованого алгоритму Falcon із застосуванням контрзаходів, які були описані в минулому розділі, а також порівнює його швидкодію з чистою реалізацією алгоритму.

**5.1 Falcon з контрзаходами**

Вплив на швидкодію алгоритму при застосуванні контрзаходів показані на рисунку 5.1.

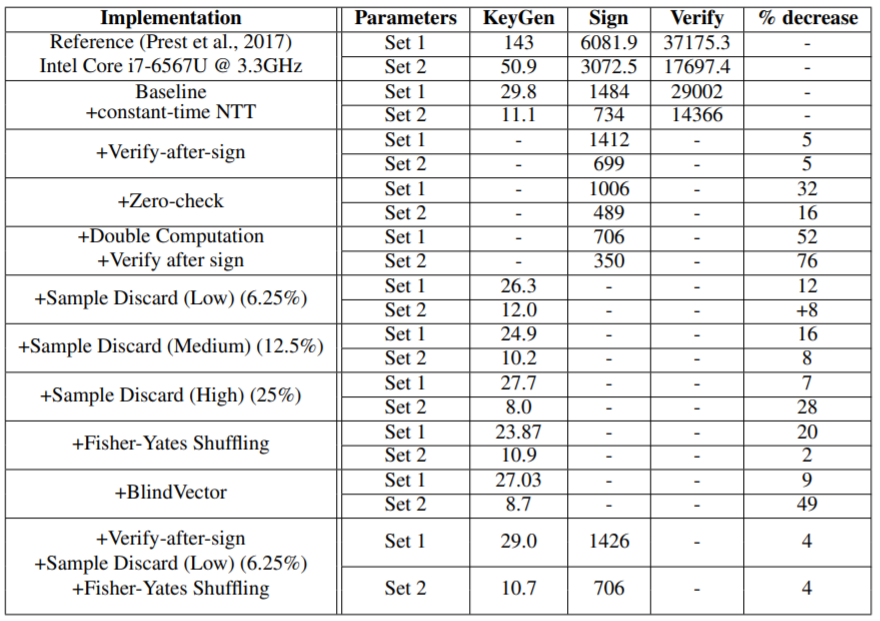


Рис. 5.1 - Результати продуктивності (ops / sec) для Falcon із контрзаходами SCA, включаючи відсоткове зменшення задіяних компонентів, на Intel E5-1620 при 3,7 ГГц.

Як видно з рисунку 5.1 – контрзаходи не мають значного впливу на швидкодію Falcon. Загальна рекомендована комбінація контрзаходів уповільнює генерацію ключів та підписання менш ніж на 5% для набору параметрів 1. Такий результат зумовлений ефективністю процесів вибірки та перевірки.

**5.2 Ефективність контрзаходів**

**5.2.1 Часові атаки**

Для перевірки контрзаходу проти атак помилками, було 100 разів проведено процедуру підписання та визначено, що для набору 1, він потрапляв в діапазон 50 операцій на секунду, а для набору 2 - 20 операцій на секунду. Це свідчить про його ефективність проти часових атак.

**5.2.2 BEARZ Атака**

Перевірка після підписування може не виявити атаки помилками, яка націлена на вибірку, якщо одна і та ж помилка успішно реалізована двічі. Тому атака все ще може сформувати дійсний підпис і залишитись невиявленою. Однак контрзахід нульової перевірки повинен виявити атаку зі 100% успіхом. Тому його можна рекомендувати як мінімальний і достатній контрзахід.

**5.3 Порівняння з іншими схемами підписів на основі решітки**

У цьому підрозділі проводиться порівняння результатів даної реалізації Falcon з Dilithium (Lyubashevsky et al., 2017) та Bliss-B (Ducas, 2014) з використанням пропонованих контрзаходів. Застосування перевірки після підпису спричиняє менші втрати продуктивності для алгоритму Falcon порівняно з алгоритмом Dilithium, як показано на рисунку 5.3. Навіть при високому рівні безпеки його продуктивність знижується не так помітно, як у BLISS-B, як показано на рисунку 5.2. Це можна пояснити ефективністю компонентів алгоритму ЕЦП Falcon.

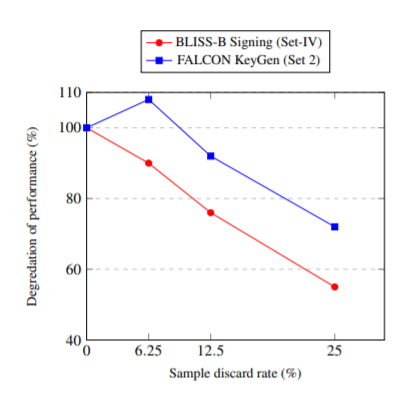


Рис. 5.2 – Ефект контрзаходу відкидання зразків на BLISS-B та Falcon для відповідного (192 біти) рівня захищеності

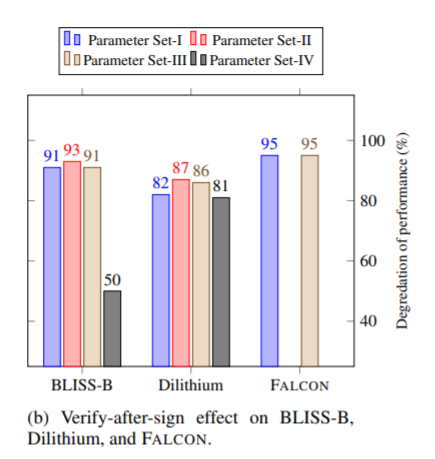


Рис. 5.3 – Ефект перевірки після підпису на BLISS-B, Dilithium та Falcon

BLISS-B пропонує широкий спектр контрзаходів для захисту від запропонованих атак, і, як видно з рисунку 5.3, їх вплив на продуктивність може коливатися від додаткових 10% до 50%, залежно від реалізації. Однією з переваг схеми Dilithium є захищеність від атак на відбірник Гауса, проте його захист від атак помилками Bruinderink та Pessl (2018) залишається проблемою. При застосуванні до нього контрзаходів, його ефективність сповільнюється майже на 20%. А це є навіть гіршими показниками, ніж показники BLISS-B при застосуванні контрзаходу для захисту Гаусового відбірника.

**6 Висновки**

У цьому дослідженні було розглянуто атаку на схему підпису Falcon - BEARZ. Було показано, що певні математичні компоненти, які використовуються в алгоритмі ЕЦП Falcon, є вразливими до атак, заснованих на аналізі часових показників, а також, що Falcon є вразливим до атак помилками на відбірник Гауса. Через це при стандартизації чи впровадженні слід розглядати можливість фізичних атак. Було розглянуто можливі контрзаходи для протидії спеціальним атакам, показано вплив даних контрзаходів на швидкодію алгоритму. Крім того, було порівняно швидкодію різних схем, заснованих на решітках, і показано, що Falcon є конкурентним кандидатом другого раунду, навіть із застосуванням запропонованих контрзаходів.

**Список джерел**

[1] Ajtai, M. and Dwork, C. (1997). A public-key cryptosystem with worst-case/average-case equivalence. In STOC ’97 Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, pages 284–293.

[2] Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q., Liu, Y.-K., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., and Smith-Tone, D. (2019). Status Report on the First Round of the NIST PostQuantum Cryptography Standardization Process. http://aiweb.techfak.uni-bielefeld.de/ content/bworld-robot-control-software/. [Online; accessed February 2019].

[3] Alkim, E., Ducas, L., Pöppelmann, T., and Schwabe, P. (2016). Post-quantum key exchange-a new hope. In USENIX Security Symposium, volume 2016. Bindel, N., Buchmann, J., and Krämer, J. (2016). Lattice-based signature schemes and their sensitivity to fault attacks. In Fault Diagnosis and Tolerance in Cryptography (FDTC), 2016 Workshop on, pages 63–77. IEEE.

[4] Bindel, N., Kramer, J., and Schreiber, J. (2017). Special session: hampering fault attacks against lattice-based signature schemes-countermeasures and their efficiency. In Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2017 International Conference on, pages 1–3. IEEE.

[5] Blömer, J., Silva, R. G. D., Günther, P., Krämer, J., and Seifert, J.-P. (2014). A practical second-order fault attack against a real-world pairing implementation. In Fault Diagnosis and Tolerance in Cryptography (FDTC), 2014 Workshop on, pages 123– 136. IEEE.

[6] Bruinderink, L. G., Hülsing, A., Lange, T., and Yarom, Y. (2016). Flush, Gauss, and reload-a cache attack on the BLISS lattice-based signature scheme. In International Conference on Cryptographic Hardware and Embedded Systems, pages 323–345. Springer. Bruinderink, L. G. and Pessl, P. (2018). Differential fault attacks on deterministic lattice signatures. IACR Transactions on Cryptographic Hardware and Embedded Systems, pages 21–43.

[7] Ducas, L. (2014). Accelerating BLISS: the geometry of ternary polynomials. Cryptology ePrint Archive, Report 2014/874. https://eprint.iacr.org/ 2014/874.

[8] Ducas, L., Lyubashevsky, V., and Prest, T. (2014). Efficient identity-based encryption over ntru lattices. In International Conference on the Theory and Application of Cryptology and Information Security, pages 22–41. Springer.

[9] Ducas, L. and Prest, T. (2016). Fast fourier orthogonalization. In Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, pages 191–198. ACM.

[10] Espitau, T., Fouque, P., G’erard, B., and Tibouchi, M. (2018). Loop-abort faults on lattice-based signature schemes and key exchange protocols. IEEE Transactions on Computers, 67(11):1535–1549.

[11] Espitau, T., Fouque, P.-A., Gérard, B., and Tibouchi, M. (2016). Loop-abort faults on lattice-based fiatshamir and hash-and-sign signatures. In International Conference on Selected Areas in Cryptography, pages 140–158. Springer.

[12] Fisher, R. A., Yates, F., et al. (1938). Statistical tables for biological, agricultural and medical research. Statistical tables for biological, agricultural and medical research.

[13] FPLLL Development Team, O. T. (2016). fplll, a lattice reduction library. Available at https:// github.com/fplll/fplll.

[14] Gentry, C. and Boneh, D. (2009). A fully homomorphic encryption scheme, volume 20. Stanford University Stanford.

[15] Hodgers, P., Regazzoni, F., Gilmore, R., Moore, C., and Oder, T. (2016). State-of-the-art in physical side-channel attacks and resistant technologies. Technical report.

[16] Howe, J., Khalid, A., Rafferty, C., Regazzoni, F., and O’Neill, M. (2016). On practical discrete Gaussian samplers for lattice-based cryptography. IEEE Transactions on Computers.

[17] Howe, J., Pöppelmann, T., O’Neill, M., O’Sullivan, E., and Güneysu, T. (2015). Practical lattice-based digital signature schemes. ACM Transactions on Embedded Computing Systems (TECS), 14(3):41.

[18] Karmakar, A., Roy, S. S., Reparaz, O., Vercauteren, F., and Verbauwhede, I. (2018). Constant-time discrete gaussian sampling. IEEE Transactions on Computers.

[19] Khalid, A., Howe, J., Rafferty, C., and O’Neill, M. (2016). Time-independent discrete gaussian sampling for post-quantum cryptography. In 2016 International Conference on Field-Programmable Technology (FPT), pages 241–244. IEEE.

[20] Khalid, A., Oder, T., Valencia, F., O’Neill, M., Güneysu, T., and Regazzoni, F. (2018). Physical protection of lattice-based cryptography: Challenges and solutions. In Proceedings of the 2018 on Great Lakes Symposium on VLSI, pages 365–370. ACM.

[21] Longa, P. and Naehrig, M. (2016). Speeding up the number theoretic transform for faster ideal latticebased cryptography. In International Conference on Cryptology and Network Security, pages 124– 139. Springer.

[22] Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., and Stehle, D. (2017). CRYSTALS-Dilithium. Technical report, National Institute of Standards and Technology. available at https://csrc.nist.gov/ projects/post-quantum-cryptography/ round-1-submissions.

[23] Micciancio, D. and Walter, M. (2017). Gaussian sampling over the integers: Efficient, generic, constanttime. In Annual International Cryptology Conference, pages 455–485. Springer.

[24] Naccache, D., Nguyen, P. Q., Tunstall, M., and Whelan, C. (2005). Experimenting with Faults, Lattices and the DSA. In International Workshop on Public Key Cryptography, pages 16–28. Springer.

[25] NIST (2016a). Post-quantum crypto project. http://csrc.nist.gov/groups/ST/ post-quantum-crypto/.

[26] NIST (2016b). Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. https: //csrc.nist.gov/csrc/media/projects/ post-quantum-cryptography/documents/ call-for-proposals-final-dec-2016.pdf.

[27] Pessl, P. (2016). Analyzing the shuffling side-channel countermeasure for lattice-based signatures. In International Conference in Cryptology in India, pages 153–170. Springer.

[28] Pessl, P., Bruinderink, L. G., and Yarom, Y. (2017). To BLISS-B or not to be: Attacking strongSwan’s Implementation of Post-Quantum Signatures. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 1843–1855. ACM.

[29] Prest, T., Fouque, P.-A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., and Zhang, Z. (2017). Falcon. Technical report, National Institute of Standards and Technology. available at https://csrc.nist.gov/ projects/post-quantum-cryptography/ round-1-submissions.

[30] Primas, R. (2017). Side-channel attacks on efficient lattice-based encryption. Master’s thesis, Graz University of Technology, Graz.

[31] Regev, O. (2005). On lattices, learning with errors, random linear codes, and cryptography. In Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005, pages 84–93.

[32] Regev, O. (2009). On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM), 56(6):34:1–34:40.

[33] Roy, S. S., Reparaz, O., Vercauteren, F., and Verbauwhede, I. (2014). Compact and side channel secure discrete Gaussian sampling. IACR Cryptology ePrint Archive, 2014:591.

[34] Saarinen, M.-J. O. (2015). Gaussian sampling precision and information leakage in lattice cryptography. IACR Cryptology ePrint Archive, 2015:953.

[35] Saarinen, M.-J. O. (2017). Arithmetic coding and blinding countermeasures for lattice signatures. Journal of Cryptographic Engineering, pages 1– 14.

[36] Scott, M. (2017). A note on the implementation of the number theoretic transform. In IMA International Conference on Cryptography and Coding, pages 247–258. Springer.

[37] Shor, P. W. (1999). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Review, 41(2):303–332.

[38] Verbauwhede, I., Karaklajic, D., and Schmidt, J.- M. (2011). The fault attack jungle-a classification model to guide you. In 2011 Workshop on Fault Diagnosis and Tolerance in Cryptography, pages 3–8. IEEE.