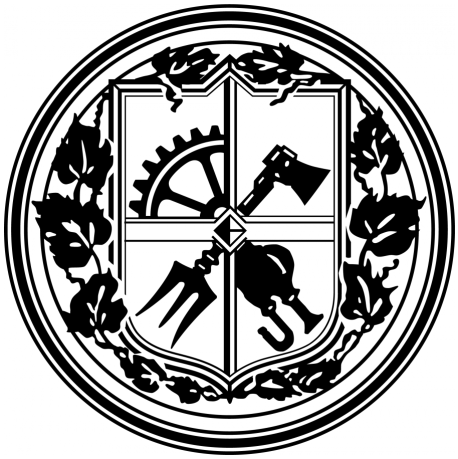
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ



Звіт за темою:

“ДОСЛІДЖЕННЯ СУЧАСНИХ АЛГЕБРАЇЧНИХ КРИПТОСИСТЕМ”

Виконали:
студенти групи ФІ-32мн
Баєвський Констянтин,
Шифрін Денис

Київ — 2023

ЗМІСТ

1 Мета практикуму

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

1. 1 Постановка задачі та варіант завдання

Бригада №4 ***Алгоритм LUOV***

Треба виконати	**Зроблено**
Детальний опис алгоритму та його складових	✓
Результати порівняльного аналізу з іншими алгоритмами	✓
Огляд наявних результатів досліджень алгоритму	✓
Результати порівняльного аналізу стійкоті з іншими алгоритмами з можливим застосуванням відомих атак	✓
Опис власних тестів, які проводилися з метою перевірки коректності реалізованої програми	✓
Детальний опис особливостей реалізації та приклади застосування і тестів	✓
Результати аналізу постквантової стійкості за наявними результатами аналізу	✓

2 Хід роботи та опис труднощів

Для початку було проаналізовано документацію щодо роботи алгоритму LUOV, аналізу проведених атак на даний алгоритм та відомих результататів досліджень. В процесі роботи було детально описано:

– криптографічний алгоритм LUOV та його складові частини;

– реалізації основних алгебраїчних операцій, які використовує даний алгоритм;

– результати порівняльного аналізу швидкодії даного алгоритму зі схожими алгоритмами (або модифікаціями алгоритму за допомогою заміни складових частин);

– наявні результати досліджень даного алгоритму;

– результати порівняльного аналізу стійкості даного алгоритму зі схожими алгоритмами з обґрунтуванням можливості застосування відомих атак

– власні тести, які проводились для перевірки коректності реалізованого алгоритму

– особливості реалізації

– результати аналізу постквантової стійкості алгоритму LUOV.

В результаті було отримано повний теоретичний опис алгоритму та реалізації основних алгебраїчних операцій, які використовує обраний алгоритм. Дані описи будуть використовуватись безпосередньо для подальшої реалізації даного алгоритму.

При виконанні практикуму ніяких труднощів не виникло.

3 Результати дослідження

В результаті проробленої роботи було детально проаналізовано та теоретично описано алгоритм LUOV та реалізації основних алгебраїчних операцій, які використовує даний алгоритм.

3. 1 Опис алгоритму

LUOV Signature Scheme for NIST PQC Project — це пост–квантовий алгоритм цифрового підпису, заснований на використанні еліптичних кривих. Він забезпечує безпеку підпису навіть у випадку, якщо квантові комп'ютери будуть створені.

Алгоритм LUOV Signature Scheme є одним із найперспективніших пост–квантових алгоритмів цифрового підпису. Він є безпечним, ефективним і відносно простим у реалізації.

LUOV = UOV + PRNG + Field Lifting + спрощений секретний ключ, де:

– UOV – одна з найстаріших і найкраще вивчених схем підпису в галузі багатоваріантної криптографії. Вона була розроблена Дж. Патаріном у 1997 році та витримала два десятиліття криптоаналізу. Схема UOV дуже проста, має маленькі підписи та швидка. Однак основним недоліком UOV є те, що відкритий ключ досить великий. LUOV покращує UOV, щоб прискорити алгоритм і зменшити розмір відкритого ключа.

– PRNG – у оригінальній схемі UOV спочатку навмання вибирається секретний ключ, а потім обчислюється відповідний відкритий ключ. Однак також можна випадково вибрати велику частину відкритого ключа, а потім обчислити відповідний секретний ключ і решту відкритого ключа. LUOV використовує цю техніку, так що більша частина відкритого ключа може бути згенерована з PRNG. Це значно зменшує розмір відкритого ключа.

– Field Lifting – LUOV генерує пару ключів над бінарним полем F2, але підносить ці ключі до розширення поля (наприклад, F 27,F247), щоб використовувати їх для підпису та перевірки повідомлень. Це значно зменшує розмір ключів (оскільки кожен коефіцієнт є одним бітом), але не впливає на складність розв'язування системи поліномів, оскільки розв'язки живуть лише в полі розширення.

– Спрощений секретний ключ – UOV, як і багато інших схем MQ, має властивість, що секретний ключ не є унікальним. Існує величезна кількість секретних ключів, які можуть відповідати одному відкритому ключу. Однак не всі ці можливі секретні ключі однаково ефективні. LUOV вибирає певний секретний ключ, щоб генерація та підписання ключів відбувалися набагато швидше.

Алгоритм LUOV використовує функції розширюваного виведення SHAKE для надання криптографічно захищених псевдовипадкових потоків бітів.

Приватний ключ для схеми підпису LUOV – це послідовність із 256 випадкових бітів (які використовуються для заповнення Keccak1600 Sponge) і просто кодується як послідовність із 32 байтів.

Відкритий ключ схеми підпису LUOV – це послідовність із 32 байтів (які використовуються для заповнення Keccak Sponge) і матриці m на $\frac{m(m+1)}{2}$ із двійковими записами. Матриця кодується шляхом об'єднання стовпців і доповнення результату нульовими бітами, щоб отримати послідовність бітів, довжина яких ділиться на 8. Потім послідовність інтерпретується як послідовність байтів, де перші біти мають найменші значення.

Публічне початкове число, представлене 32 байтами, отримують із губки шляхом стиснення (видавлювання) 32 байтів.

Губка (sponge) — це криптографічна геш-функція з деревоподібною структурою, яка дозволяє ефективно абсорбувати дані та стискати вихід.

В алгоритмі LUOV в якості функції, яка використовує конструкцію губки виступає Keccak1600.

Розглянемо еліптичну криву E над полем F_p, де p – просте число. Нехай G – точка на кривій E, а n – деяке число, що ділить порядок групи $E(F_p)$.

Параметри алгоритму LUOV:

– Еліптична крива (наприклад, $E : y^2 = x^3 + 7x + 5$)

– Поле F_p (наприклад, F_{17})

– Базова точка G на кривій (наприклад, G =(3, 12))

– Геш-функція (наприклад, SHAKE-256)

– Функція стиснення(*sponge*) (наприклад, Keccak)

Алгоритм генерації ключів для LUOV

Алгоритм 0.1. Генерація ключів для LUOV.
Вхід: $private_seed$ – початкове значення для створення ключа.
Вихід: $(public\_seed, Q_2)$ – публічний ключ, $private\_seed$ – відповідний закритий ключ.

1) Ініціалізується губка з заданим $private\_seed$, результатом якої є $private\_sponge$.

2) Вичавлюється публічний набір з $private\_sponge$. Публічний набір використовується для отримання відкритого ключа і оприлюднюється.

3) Вичавлюється значення T з $private\_sponge$. Значення T використовується для отримання приватного ключа і зберігається в секреті.

4) Ініціалізується інша губка з $public\_seed$.

5) Вичавлюються три значення (C, L, Q_1) з $public\_sponge$. Значення C і L використовуються для отримання відкритого ключа, а значення Q_1 використовується для отримання приватного ключа.

6) Обчислюються значення Q_2 з Q_1 і T. Значення Q_2 є частиною відкритого ключа.

7) В результаті повертається відкритий ключ ($public\_seed$, Q_2) і приватний ключ ($private\_seed$).

Саме функція, яка обчислює значення Q_2 – FindQ2($Q1$, T) є важливою частиною алгоритму. Вона забезпечує безпеку алгоритму, оскільки її важко обчислити, знаючи лише значення Q_1 і T.

Алгоритм генерування підпису LUOV

Алгоритм 0.2. LUOV Signature Scheme працює наступним чином:
Вхід: відкритий ключ (s_1,s_2), приватний ключ $private\_seed$, повідомлення M.
Вихід: підпис s.

1) Ініціалізація губки до нульового стану та поглинання даних з $private\_seed$. В результаті отримуємо значення sponge.

2) Вичавлювання значення $public\_seed$ з sponge.

3) Обчислення значення T зі sponge.

4) Ініціалізація губки до нульового стану та поглинання даних з $public\_seed$. В результаті отримуємо значення $public\_sponge$.

5) Обчислення значень матриць C, L і Q_1 шляхом відбору колонок з $public\_sponge$.

6) Ініціалізація губки до нульового стану та поглинання даних з конкатенації повідомлення M та його падінгу $0x00$. В результаті отримуємо значення $hash\_sponge$ для гешування повідомлення.

7) Обчислення геш-значення повідомлення M з $hash\_sponge$. В результаті отримуємо вектор геш-значень h.

8) Ініціалізація губки до нульового стану та поглинання даних з конкатенації повідомлення M, його падінгу $0x00$ та значення $private\_seed$. В результаті отримуємо губку з випадковими числами $vinegar\_sponge$ для генерування підпису.

9) Поки не знайдено такого s', яке задовольнить рівняння $F(s') = h$, виконуємо наступний цикл:

а) Вичавлювання байтів з губки $vinegar\_sponge$ та їх інтерпретація в кодування. В результаті отримуємо вектор v.

б) Побудова розширеної матриці A для лінійної системи $F(v||o) = h$. Для побудови беруться в якості вхідних даних C, L, Q_1, T, h, v. В результаті отримуємо значення матриці A – розв'язку лінійної системи методом Гауса.

в) Якщо система $F(v||o) = h$ має унікальне рішення o, то: $s' = (v||o)^T$.

10) Обраховуємо значення s: $s = \begin{pmatrix} 1_v & -T \\ 0 & 1_m \end{pmatrix} \cdot s'$

11) Повернення значення підпису s.

Алгоритм верифікації підпису LUOV

Алгоритм 0.3. Верифікація підпису LUOV.
Вхід: ($public\_seed$,Q_2) – публічний ключ, повідомлення M, підпис s.
Вихід: *Прийнято (accept)* - якщо підпис s дійсний, *Відхилено (reject)* - в інших випадках.
1) Ініціалізація губки до нульового стану та поглинання даних з конкатенації повідомлення M та його падінгу 0x00. В результаті отримуємо губку *sponge* для гешування.
2) Обчислення геш-значення повідомлення M з губки *sponge*. В результаті отримуємо вектор геш-значень h.
3) За алгоритмом EvaluatePublicMap перевіряє підпис за вхідними даними ($public\_seed, Q_2$) та s.
а) якщо значення e та h співпадають: алгоритм повертає *Accept*
б) в іншому випадку: алгортм повертає *Reject*.

Також схема підпису може використовуватись в **режимі відновлення повідомлень**. Використання відновлення повідомлення не впливає на алгоритм створення підпису. Ту саму пару ключів можна використовувати для підпису повідомлень у обох режимах підпису. Підпис для M у режимі доданого підпису не пов'язаний з підписом для того самого повідомлення в режимі відновлення повідомлення, оскільки інший байт додається до повідомлення в кожному режимі.

Безпека алгоритму:

Безпека алгоритму LUOV Signature Scheme базується на тому, що важко обчислити точку P, знаючи лише точку Q і пару чисел (a, b). Ця задача є NP–складною, і навіть якщо квантові комп'ютери будуть створені, то її рішення буде вимагати значних ресурсів.

Основні алгебраїчні операції, які використовує LUOV:

– Додавання еліптичних кривих:

$$Q = aP + bG \tag{0.1}$$

– Множення числа на точку еліптичної кривої:

$$s_1 Q = aP + bGs_1 \tag{0.2}$$

3. 2 Продуктивність LUOV

Рівень безпеки	**Розмір секретного ключа**	**Розмір відкритого ключа**	**Розмір підпису**	**Генерація ключів**	**Підписання**	**Перевірка**
1	32 Б	11,5 Кб	239 Б	1,1 М циклів	224 К цикли	49 К циклів
3	32 Б	35,4 Кб	337 Б	4,6 М циклів	643 К цикли	152 К цикли
5	32 Б	82,0 КБ	440 Б	9,7 М циклів	1,1 М циклів	331 К циклів

Таблиця 1 – Характеристики продуктивності оптимізованої реалізації LUOV

Характеристики продуктивності оптимізованої реалізації LUOV з постійним часом AVX2. Для покращення продуктивності дана реалізація виконує деякі попередні обчислення пари ключів. Для впровадження без інструкцій AVX2 або з різними рівнями попереднього обчислення звертаємося до документа NIST PQC Round2.

3. 3 Атака QuantumHammer

Схема Oil and Vinegar є однією з найкраще вивчених багатоваріантних схем підпису, яка, за умови правильного вибору параметрів, витримує всі криптоаналізи з 1997 року.

Гібридна атака QuantumHammer — це комбінація двох атак: атаки трасування бітів, увімкненої за допомогою впровадження помилки Rowhammer, і атаки розділяй і володарюй, яка використовує трасування бітів як оракул. Використовуючи трасування бітів, зловмисник, маючи доступ до помилкових підписів, зібраних за допомогою атаки Rowhammer, може відновити секретні ключові біти, хоча й повільно. Використовується атака «розділяй і володарюй», яка в свою чергу використовує структуру в частині LUOV, яка генерує ключі, і ефективніше розв'язує систему рівнянь для секретного ключа з кількома бітами ключа, відновленими за допомогою трасування бітів. Було продемонстровано першу успішну атаку в дикій природі на LUOV, яка відновила всі 11 тисяч бітів ключа менш ніж за 4 години активної атаки Rowhammer. Частина постобробки є дуже паралельною, тому її можна тривіально пришвидшити, використовуючи скромні ресурси. QuantumHammer не робить жодних нереалістичних припущень, вимагає лише спільного розміщення програмного забезпечення (без фізичного доступу), і тому може використовуватися для націлювання на спільні хмарні сервери або в інших середовищах ізольованого програмного середовища.

Таким чином атака Rowhammer може призвести до серйозних наслідків через перевертання бітів в інших процесах і витік ключової інформації. QuantumHammerattack поєднує в собі обидві слабкі сторони, щоб розпочати успішну атаку з відновленням повного секретного ключа схеми.

3. 4 Атака Nested Subset Differential Attack

Модифікована версія диференціальної атаки підполя під назвою «Вкладена диференціальна атака підмножини» повністю порушує половину параметрів, встановлених у раунді 2 версії Lifted Unbalanced Oil and Vinegar. Автори звели атаку на ці набори параметрів до проблеми розв'язання квадратних рівнянь над простим полем

F2. Це робить їхню атаку достатньо ефективною для практичного виконання. Оскільки дана атака не використовувала незбалансовану структуру LUOV, її можна розглядати як метод вирішення піднятих квадратичних систем загалом. Автори вважають, що необхідні додаткові дослідження для розв'язання такого типу квадратичних систем за допомогою атаки NSDA. Також було проведено експериментальні атаки на фактичні параметри LUOV і змогли підробити підпис менш ніж за 210 хвилин.

Також на алгоритм підпису LUOV проводились такі "класичні атаки"як DirectAttack та HashCollisionAttack.

3. 5 Опис власних тестів

Для реалізації алгоритму LUOV проводились наступні тести:

• Тест на швидкодію в залежності від розміру вхідного повідомлення.

Для тесту використовувались повідомлення різних розмірів (100 символів, 1000 символів та 10000 символів). В результаті було отримано, що швидкодія підписання та перевірки підпису не залежить від довжини вхідного повідомлення. Середня швидкодія при виконанні даного тесту лежить в межах проміжку [600,900] мс.

• Тест генерації підпису.

В результаті було отримано правильний підпис для повідомлення.

• Тест верифікації підпису повідомлення.

В результаті було отримано значення true після верифікації підпису для повідомлення.

• Тест верифікації пошкодженого повідомлення.

В результаті було отримано значення false після верифікації підпису для пошкодженого повідомлення.

• Тест використання різних параметрів кривої. В результаті було показано використання алгоритму підпису LUOV для 2-ох можливих варіантів набору параметрів.

3. 6 Детальний опис особливостей реалізації та приклади застосування

При реалізації алгориму цифрового підпису LUOV посилалися на деякі інші відомі реалізації алгоритму, включно з документацією. Ось деякі з них:

1) https://github.com/WardBeullens/LUOV
2) https://github.com/saadislamm/QuantumHammer
3) https://github.com/Danny-xyy/LUOV

4) https://github.com/EddAngulo/luov_java_implementation

5) https://github.com/WardBeullens/ThesisCode

Основні функції реалізації алгоритму:

1) Можливість вибору поля з параметрами для роботи з алгоритмом серед запропонованих:

– GF(207): r=7,m=57,v=197.

– GF(261): r=61,m=60,v=261.

2) Функція генерації публічного та приватного ключів: keyGen().

3) Функція генерації підпису повідомлення: sign().

4) Функція перевірки підписаного повідомлення: verify().

Приклади використання реалізації та проведених тестів:

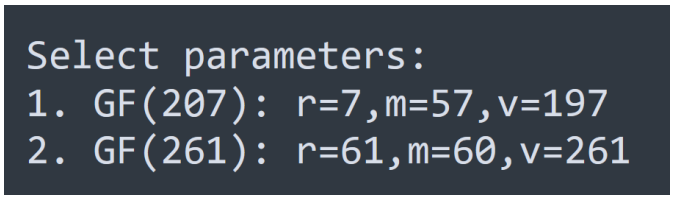


Рисунок 1 – Можливість вибору параметрів.

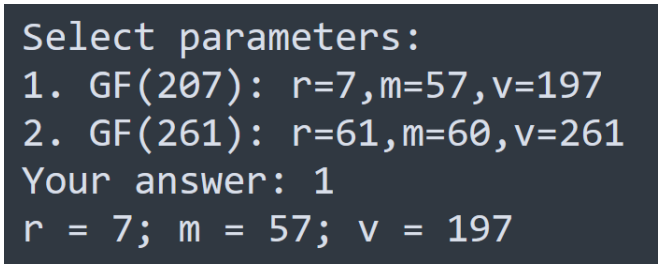


Рисунок 2 – Вибір 1-го типу параметрів.

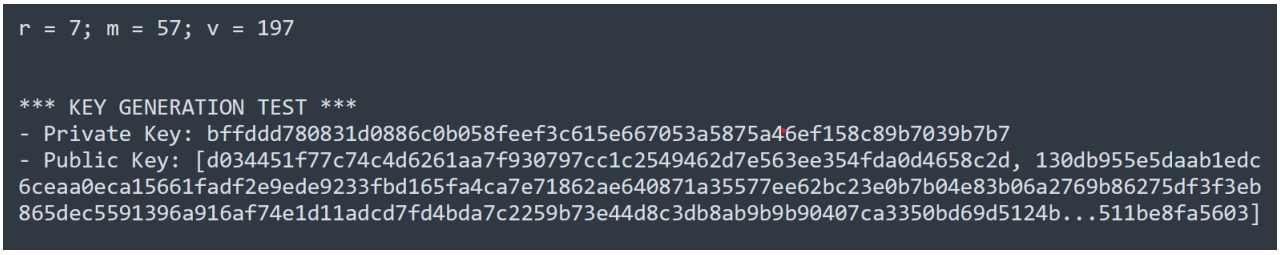


Рисунок 3 – Генерація ключів для 1-го типу параметрів.

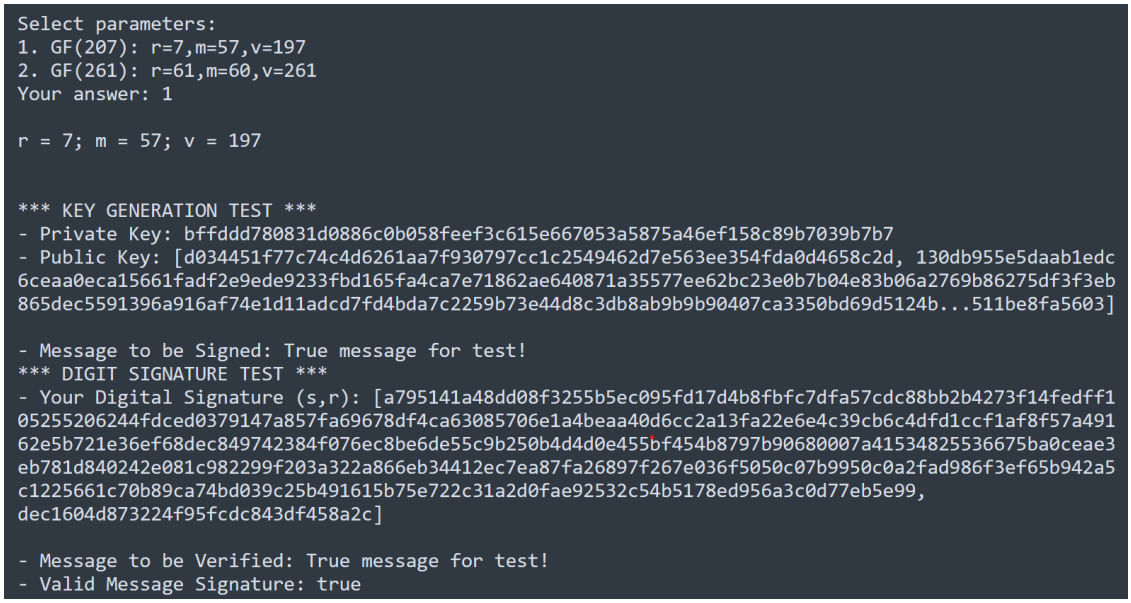


Рисунок 4 – Підписування повідомлення для 1-го типу параметрів.

Рисунок 5 – Пошкодження повідомлення при верифікації для 1-го типу параметрів.

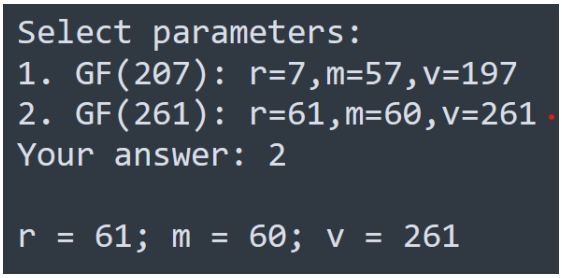


Рисунок 6 – Вибір 2-го типу параметрів.

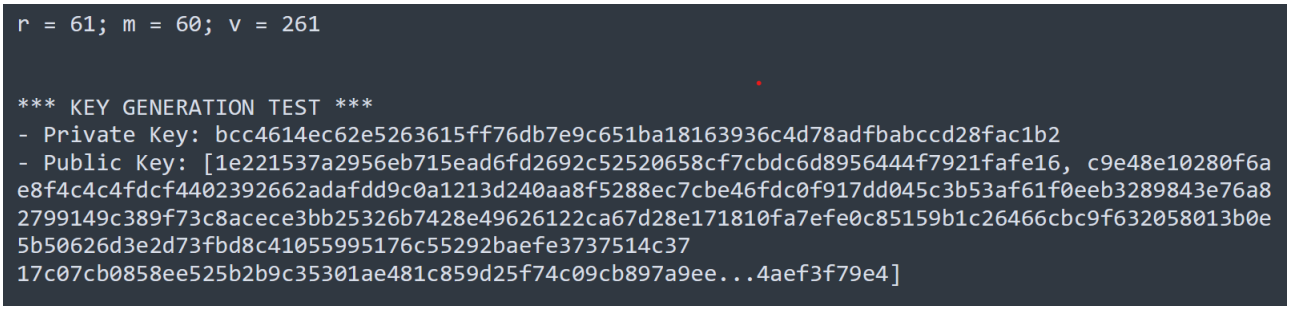


Рисунок 7 – Генерація ключів для 2-го типу параметрів.

Рисунок 8 – Підписування повідомлення для 2-го типу параметрів.

А також прикріплюємо результат тесту верифікації підпису з пошкодженим повідомленням для 2-го типу параметрів кривої:

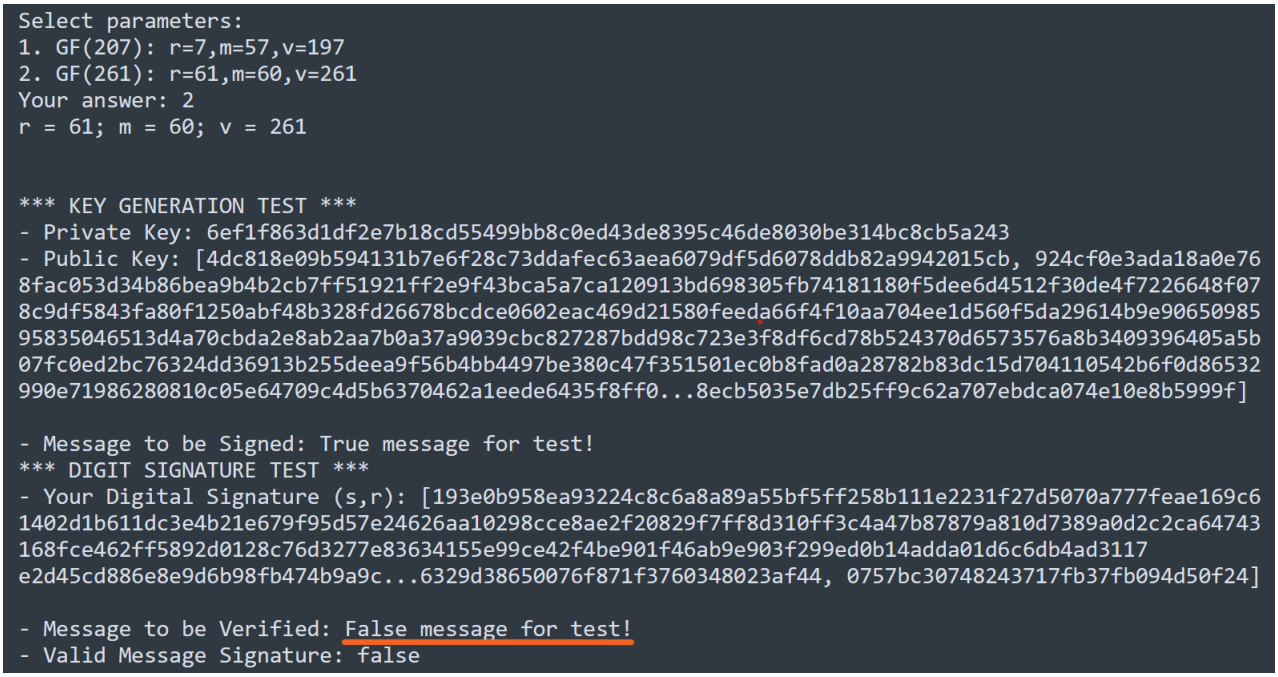


Рисунок 9 – Пошкодження повідомлення при верифікації для 2-го типу параметрів.

3. 7 Результати аналізу постквантової стійкості

Постквантову стійкість можна перевірити оцінкою складності прямої атаки на LUOV з одним з можливих наборів параметрів, наприклад (r = 7, m = 57, v = 197); можемо визначити рівень безпеки, який досягає цей набір. Для цього можна скористатися використанням методу Тома і Вольфа. Таким чином можна звести пошук розв'язку цієї недовизначеної системи до пошуку розв'язку детермінованої системи з $57 + 1 - \lfloor(57 + 197)/57\rfloor = 54$ рівняння. Ми припускаємо, що ця система та системи, отримані шляхом фіксації ряду змінних, є напіврегулярними. Якщо ми фіксуємо k додаткових змінних, то ступінь регулярності дорівнює ступеню першого члена в степеневому ряді:

$S_{54,54-k}(x) = \frac{(1-x^2)^{54}}{(1-x)^{54-k}}$, який має недодатний коефіцієнт.

Для k = 0 ми маємо $S_{54}(x) = (1 + x)54$, тому ступінь регулярності дорівнює 55. Продовжуючи обчислення далі визначимо, що складність прямої атаки перевищує нижню межу 2146×1,1, як вимагається. А якщо використати пошук Гровера замість частини гібридного підходу грубої сили, щоб прискорити пряму атаку. Таким чином продовжуючи обчислення далі дійдемо до того, що для всіх вибраних параметрів і всіх практичних значень k складність навіть одного обчислення базису Гробнера перевищує 264, і алгоритм Гровера повинен виконувати велику кількість цих обчислень послідовно, щоб отримати помітне прискорення порівняно з класичним грубим методом.

В результаті отримаємо, що алгоритм LUOV має постквантову стійкість до атак.

4 Висновки

В даному практикумі наведено повний теоретичний опис алгоритму з усіма деталями та відомими результатами досліджень. Проведено теоретичний порівняльний аналіз обраного алгоритму зі схожими алгоритмами та дослідження відомих атак на даний алгоритм. Крім цього в ході практикуму було реалізовано алгоритм LUOV для 2-ох варіантів наборів парметрів кривої. А також проведено власні тести для реалізації алгоритму, проаналізовано постквантову стійкість даного алгоритму.