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

«До захисту допущено»
В.о. завідувача кафедри
_____ Михайло САВЧУК
«__» _____ 2021 р.

Дипломна робота

на здобуття ступеня бакалавра

зі спеціальності: 113 Прикладна математика
на тему: **«Квантовий криптоаналіз геш-функції «Купина»»**

Виконав: студент 4 курсу, групи ФІ-73
Ткаченко Артем Станіславович

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

Консультант: _ _____

Рецензент: к.ф., м.н., доц. кафедри ІБ Южакова Г.О. _____

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

Студент _____

Київ — 2021

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
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
_____ Михайло САВЧУК
«__» _____ 2021 р.

ЗАВДАННЯ
на дипломну роботу

Студент: Ткаченко Артем Станіславович

1. Тема роботи: «*Квантовий криптоаналіз геш-функції «Купина»*»,
керівник: к.ф., м.н., ст.викладач Фесенко А.В.,
затверджені наказом по університету №__ від «__» _____ 2021р.

2. Термін подання студентом роботи: 4 червня 2021р.

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

4. Зміст роботи: побудова квантової реалізації геш-функції «Купина», дослідження квантової атаки за допомогою алгоритма Гровера

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

6. Дата видачі завдання: 10 вересня 2020 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником	01-15 вересня 2020 р.	Виконано
2	Огляд опублікованих джерел за тематикою дослідження	Вересень-жовтень 2020 р.	Виконано
3	Аналіз методів квантового криптоаналізу	Жовтень-грудень 2020 р.	Виконано
4	Аналіз застосовності алгоритму Гровера до геш-функцій, розбір криптоаналізу масштабованих криптопримітивів	Січень-березень 2021 р.	Виконано
5	Ознайомлення із фреймворком Qiskit	Квітень-травень 2021 р.	Виконано
6	Робота над реалізацією програмних компонент	Травень 2021 р.	Виконано
7	Оформлення пояснювальної записки	Травень 2021 р.	Виконано
8	Формулювання результатів дослідження	Травень 2021 р.	Виконано

Студент _____ Ткаченко А.С.

Керівник _____ Фесенко А.В.

РЕФЕРАТ

Кваліфікаційна робота містить: 75 стор., 37 рисунків, 5 таблиць, 23 джерел.

Об'єктом дослідження є інформаційні процеси в системах криптографічного захисту інформації.

Предметом дослідження є складність застосування алгоритму Гровера до геш-функції «Купина» у квантовій моделі обчислень.

Метою роботи є побудова атаки з використанням алгоритму квантового пошуку (алгоритму Гровера) на криптографічну геш-функцію «Купина» у квантовій моделі обчислень.

В роботі досліджено внутрішню структуру криптографічної геш-функції «Купина». Реалізовано та оцінено складність імплементації компонент із яких складаються перестановки функції стиснення, а саме: повний сумматор, що реалізує складання за модулем 2, схема додавання за модулем 2^{64}, схема множення елментів у полі $\mathbf{GF}(2^8)$, пошук елемента для заміни за допомогою підстановок $\pi_0, \pi_1, \pi_2, \pi_3$. Це дозволило оцінити складності реалізації перестановок $T_l^{\oplus}$ та T_l^{+}. Вентильна складність та максимальна глибина схеми склали відповідно: $T_{512}^{\oplus} : O(2^{24.7}), O(2^{24.7})$; $T_{512}^{+} : O(2^{25.06}), O(2^{24.7})$; $T_{1024}^{\oplus} : O(2^{26.5}), O(2^{26.16})$; $T_{1024}^{+} : O(2^{26.6}), O(2^{26.16})$.

Отримані оцінки по перестановкам дозволили оцінити складність реалізації геш-функції «Купина» в цілому. Оцінка проводилася зверху у найгіршому випадку (при використанні максимального числа блоків повідомлення), використовуючи кількість вентилів групи Кліффорда, Т-вентилів та загальної глубини схеми. Отримані значення відповідно дорівнюють: для $l = 512$ – $2^{111.9}$, $2^{111.85}$ та $2^{112.7}$; для $l = 1024$ – $2^{112.65}$, $2^{112.35}$ та $2^{113.2}$.

За допомогою значень складності реалізації криптопримітиву проведено його криптоаналіз із боку застосування алгоритму Гровера. Порівнюючи отримані результати із NIST-метрикою, зроблено висновок

про застосовність алгоритму пошуку Гровера до версії $l = 512$, при складності, що становить $O(2^{307.6})$.

ГЕШ-ФУНКЦІЯ «КУПИНА», АЛГОРИТМ ГРОВЕРА, КВАНТОВИЙ ДИФЕРЕНЦІАЛЬНИЙ КРИПТОАНАЛІЗ

ABSTRACT

Qualification work contains: 75 pages, 37 figures, 5 tables, 23 sources.

The object of research is information processes in systems of cryptographic protection of information.

The subject of the research is the complexity of applying Grover's algorithm to the hash function «Kupyna» in the quantum model of calculations.

The goal of the work is to construct attacks on the hash function «Kupyna» by methods of quantum cryptanalysis using Grover's algorithm, and to assess the complexity of these attacks.

The internal structure of the cryptographic hash function «Kupyna» is investigated in the work. implemented and evaluated the complexity of the implementation of the components that make up the permutations of the compression function, namely: a complete adder that implements the addition of module 2, addition scheme by module 2^{64}, scheme of multiplication of elements in the field $\mathbf{GF}(2^8)$, search for an item to replace with substitutions $\pi_0, \pi_1, \pi_2, \pi_3$. This allowed us to assess the complexity of the implementation of the permutations $T_l^{\oplus}$ and T_l^{+}. The valve complexity and the maximum depth of the circuit were respectively: $T_{512}^{\oplus} : O(2^{24.7}), O(2^{24.7})$; $T_{512}^{+} : O(2^{25.06}), O(2^{24.7})$; $T_{1024}^{\oplus} : O(2^{26.5}), O(2^{26.16})$; $T_{1024}^{+} : O(2^{26.6}), O(2^{26.16})$.

The obtained permutation estimates allowed us to assess the complexity of the implementation of the hash function «Kupyna» in general. The evaluation was performed from above in the worst case (using the maximum number of message blocks), using the number of gates of the Clifford group, T-gates and the total depth of the scheme. The obtained values are respectively equal to: for $l = 512$ – $2^{111.9}$, $2^{111.85}$ and $2^{112.7}$; for $l = 1024$ – $2^{112.65}$, $2^{112.35}$ and $2^{113.2}$.

Using the values of permutation complexity, a cryptanalysis of this cryptocurrency was performed by the Grover algorithm. Comparing the obtained results with the NIST-metric, we concluded that the applicability of

the Grover search algorithm to the version $l = 512$, with a complexity of $O(2^{307.6})$.

"KUPYNA"HASH FUNCTION , GROVER'S ALGORITHM, QUANTUM DIFERENTIAL CRYPTOANALYSIS

ЗМІСТ

Перелік умовних позначень, скорочень і термінів 9
Вступ 10
1 Основні теоретичні дослідження 12
1.1 Опис геш-функції SM3 12
1.2 Атаки на SM3 у класичній моделі обчислень 14
1.3 Усічені варіанти геш-функцій 16
1.4 Структура геш-функції «Купина» 21
Висновки до розділу 1 30
2 Квантовий метод криптоаналізу за допомогою алгоритму Гровера ... 31
2.1 Квантова модель обчислень 31
2.2 Квантові вентилі 35
2.3 Алгоритм квантового пошуку (алгоритм Гровера) 38
2.4 Складність алгоритму у квантовій моделі обчислень 42
2.5 Застосування алгоритму Гровера до SM3 43
Висновки до розділу 2 48
3 Квантові атаки на геш-функцію «Купина» 49
3.1 Квантова реалізаці геш-функції «Купина» 49
3.2 Застосування алгоритму Гровера до перестановок геш-функції «Купина» 60
3.3 Атака на прообраз за допомогою алгоритма Гровера 63
Висновки до розділу 3 64
Висновки 66
Перелік посилань 68
Додаток А Тексти програм 71
А.1 Програма 1 71
Додаток Б Таблиці заміни для шару нелінійного бієктивного відображення 74

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

$\oplus$ — операція побітового додавання

$\mathcal{P}$ — множина відкритих текстів

$\mathcal{C}$ — множина шифротекстів

$\mathcal{K}$ — множина ключів

$E : \mathcal{P} \times \mathcal{K} \to \mathcal{C}$ — функція шифрування

$D : \mathcal{C} \times \mathcal{K} \to \mathcal{P}$ — функція розшифрування

$\mathbf{P}_\sigma, \mathbf{P}_{\sigma^{-1}}$ — оракули, що реалізують підстановки σ та σ^{-1}

ВСТУП

Актуальність дослідження. Поява ідей та робіт, щодо реалізації квантових комп'ютерів та квантових алгоритмів змінило вектор досліджень у сучасній криптографії. Найбільш популярні алгоритми з відкритим ключем, такі як RSA та криптосистеми на еліптичних кривих, стійкість яких грунтується на задачах дискретного логарифмування та факторизації. Однак, існує, наприклад, квантовий алгоритм Шора [1], який може розкласти число M на прості множники за час $O(\log^3 M)$, використовуючи при цьому $O(\log M)$ логічних кубітів. Така швидкодія алгоритму робить неможливим використання класичних реалізацій RSA та криптосистем на еліптичних кривих у квантовій моделі обчислень.

В області симетричної криптографії, вплив результатів, що можуть надати квантові комп'ютери, не такий великий, як у випадку криптографії з відкритим ключем. Алгоритм Гровера [2] спроможний знаходити секретний ключ довжиною n біт за $O(\sqrt{n})$. Наразі - це є методом атаки у квантовій моделі обчислень на симетричний шифр. Застосування алгоритму Гровера до симетричних шифрів є найкращим способом їх оцінки стійкості до криптоаналізу у квантовій моделі обчислень. Застосовність алгоритму Гровера до криптопримітивів у квантовій моделі обчислень вимагає їх імплементацію за допомогою квантового апарату. Це робить задачу переводу алгоритмів із класичної моделі у квантову доволі актуальною, зважаючи на розвиток квантових комп'ютерів.

Метою дослідження є оцінка необхідних квантових ресурсів для реалізації геш-функції «Купина» у квантовій моделі обчислень. Для досягнення мети необхідно виконати такі завдання:

1) Провести аналіз наявних методів оцінки необхідної кількості квантових вентелів для реалізації криптопримітивів;

2) Провести аналіз наявних атак на геш-функції у квантовій моделі

обчислень та дослідити підходи, які при цьому використовуються;

3) Дослідити особливості роботи геш-функції «Купина» національного стандарту ДСТУ 7564:2014;

4) Провести оцінку кількості необхідних квантових вентилів окремих складових геш-функції «Купина» в квантовій моделі обчислень, та обчислити загальну складність реалізації;

5) Побудувати атаки на два варіанти геш-функції «Купина»: версії, що використовують вектор ініціалізації розміром 512 біт та 1024 біт;

6) Отримати оцінки складності цих атак.

Об'єктом дослідження є інформаційні процеси в системах криптографічного захисту інформаціїї в квантовій моделі обчислень.

Предметом дослідження є складність реалізації геш-функції «Купина» у квантовій моделі обчислень та реалізація атаки за допомогою алгоритму Гровера

Методи дослідження: теорії складності алгоритмів, абстрактної та лінійної алгебри, теорії імовірності.

Наукова новизна отриманих результатів полягає у тому, що вперше отримано оцінку кількості необхідних квантових вентелів для реалізації геш-функції «Купина».

Практичне значення полягає у тому, що побудовано оцінки для перестановок $T_l^{\oplus}$ та T_l^{+}, які можуть бути використані при реалізації атаки на геш-функцію «Купина». Отримана оцінка складності реалізації геш-функції «Купина» дозволяє оцінити її стійкість у квантовій моделі обчислень на даний момент.

1 ОСНОВНІ ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

У даному розділі описана геш-функція SM3, як приклад криптопримітиву, до якого може бути застосовним квантовий метод криптоаналізу за допомогою алгоритму квантового пошуку (алгоритму Гровера). Також, розділ включає аналіз методу масштабування геш-функцій, що має назву *Toy*. Наприкінці описуються необхідні компоненти побудови криптографічної геш-функції «Купина», такі як структура Меркля-Дамгора, високорівнева конструкція Дєвіса-Майера інтерпретування односторонньої функції у вигляді блокового шифру, схема Евен-Мансура мінімального блокового шифру й SPN-структури; також проаналізована структура самого криптопримітиву.

1.1 Опис геш-функції SM3

SM3 [4] - це 256-бітова криптографічна геш-функція, що входить у національний стандарт Китая. В основі цього криптопримітву лежить структура Меркля-Дамгорда [15].

Ініціалізація необхідних компонент. Для подальшої роботи необхідно ввести ряд параметрів:

1. Вектори ініціалізації:
   - $V_0^{(0)} = 7380166f$
   - $V_1^{(0)} = 4914b2b9$
   - $V_2^{(0)} = 172442d7$
   - $V_3^{(0)} = da8a0600$
   - $V_4^{(0)} = a96f30bc$
   - $V_5^{(0)} = 163138aa$
   - $V_6^{(0)} = e38dee4d$
   - $V_7^{(0)} = b0fb0e4e$

2. Індексозалежні константи:

$$T_j = \begin{cases} 794519, 0 \leqslant j \leqslant 15 \\ 7a879d8a, 16 \leqslant j \leqslant 63 \end{cases}$$

3. Перестановки:

$P_0(X) = X \oplus (X \lll 9) \oplus (X \lll 17)$

$P_1(X) = X \oplus (X \lll 15) \oplus (X \lll 23)$

4. Булеві функції:

$$FF_j = \begin{cases} X \oplus Y \oplus Z, 0 \leqslant j \leqslant 15 \\ (X \wedge Y) \vee (Y \wedge Z) \vee (X \wedge Z), 16 \leqslant j \leqslant 63 \end{cases}$$

$$GG_j = \begin{cases} X \oplus Y \oplus Z, 0 \leqslant j \leqslant 15 \\ (X \wedge Y) \vee (\bar{X} \wedge Z), 16 \leqslant j \leqslant 63 \end{cases}$$

Доповнення та розширення повідомлення. SM3 використовує 32-бітові слова та повертає геш-значення довжиною 256 біт. На початку роботи відбувається доповнення вхідного повідомлення m довжини l біт за таким правилом: в кінець повідомлення записується один одиничний біт, після цього допусується ще k нульових бітів, де k - це мінімальне невід'ємне число, яке задовольняє співвідношенню $l + k + 1 \equiv 488 \ mod \ 512$. Далі потрібно додати до цього повідомлення 64-бітну строку довжини повідмолення l. У результаті вийде доповнене повідомлення $m^{'}$ яке кратне 512.

Повідомлення $m^{'}$ представляється у вигляді блоків $B^{(0)}, B^{(1)}, \ldots, B^{(n-1)}$, де $n = \frac{l+k+65}{512}$. Кожен блок $B^{(i)}$ розширяється до 132 слів $W_0, W_1, \ldots W_{67}, W_1^{'}, \ldots, W_{63}^{'}$ за допомогою наступних співвідношень:

$$W_j \leftarrow P_1(W_{j-16} \oplus W_{j-9} \oplus (W_{j-3} \lll 15)) \oplus (W_{j-13} \lll 7) \oplus W_{j-6} \qquad (1.1)$$

$$W_j^{'} = W_j \oplus W_{j+4}, (16 \leqslant j \leqslant 67) \qquad (1.2)$$

Алгоритм отримання геш-значення. Нехай A, B, C, D, E, F, G, H - це 8 слів регістрів та SS_1, SS_2, TT_1, TT_2 - це 4 допоміжні змінні. Після цих визначень можна визначити алгоритм отримання геш-значення:

Algorithm 1.1 Алгоритм отримання геш-значення SM3

for j = 0 to 63 **do**
  $SS_1 \leftarrow ((A \lll 12) + E + (T_j \lll (j \ mod \ 32)) \lll 7$
  $SS_2 \leftarrow SS_1 \oplus (A \lll 12)$
  $TT_1 \leftarrow FF_j(A, B, C) + D + SS_2 + W_j^{'}$
  $TT_2 \leftarrow GG_j(E, F, G) + H + SS_1 + W_j$
  $D \leftarrow C$
  $C \leftarrow B \lll 9$
  $B \leftarrow A$
  $A \leftarrow TT_1$
  $H \leftarrow G$
  $G \leftarrow F \lll 19$
  $F \leftarrow E$
  $E \leftarrow P_0(TT_2)$
end for
$V(i+1) \leftarrow ABCDEFGH \oplus V(i)$
return A, B, C, D, E, F, G, H

Тобто, кінцеве значення на виході є $y = ABCDEFGH$.

1.2 Атаки на SM3 у класичній моделі обчислень

На даний криптопримітив реалізовано досить багато різних атак. Розглянемо основні результати по найбільш популярним варіантам зламу геш-функцій, таких як атака знаходження прообразу, метод бумерангу та атака на пошук колізій.

Атака знаходження прообразу.

У роботі [5] реалізовано атаку на знаходження прообразу геш-функції SM3. Автори вперше отримали доволі великі значення імовірностей для усіченних диференціалів за допомогою слабкого місця в трансформації стану оновлення та процедурі лінійного розширення повідомлень. На основі цих результатів вони змогли зробити ефективну атаку на прообраз геш-функції, використовуючи диференціальний варіант

MiTM-атаки. Реалізація атаки працює за 32 раунди. Більше того, автори також змогли досягти перетворення атаки на прообраз у псевдоколізійну атаку. За результатами досліджень, часова складність атаки на прообраз, що використовує 29, 30, 31, 32 раунди склала 2^{245}, $2^{251.1}$, 2^{245} та $2^{251.1}$ відповідно. Часова складність псевдоколізіної атаки, що використовує 29, 30, 31, 32 раунди склала 2^{122}, $2^{125.1}$, 2^{122} та $2^{125.1}$ відповідно.

Атака на пошук колізій. Особливість цієї атаки [6] базується на використанні того факту, що структура геш-функції SM3 дуже схожа із сімейством геш-функції MD4, зокрема із SHA-256. Подібні атаки засновані на концепції узагальненої умови та автоматичних алгоритмів пошуку. У цій роботі автори використали результати отримані у [7] для SHA-256 та застосували їх до геш-функції SM3. Було представлено дві версії атаки - це 20-раундова та 24-раундова із вільним стартом. У результаті автори отримали такі оцінки часової складності: 20-раундова версія - $2^{117.1}$, 24-раундова - 2^{249}.

Метод бумерангу. Криптоаналіз, що використовує цей метод, представлений у роботах [8, 9]. У цій роботі представлено колізії другого порядку для функції стиснення SM3. Основна проблема, що вирішувалася під час криптоаналізу, була у суперечливості умов, які надавали нижній та верхній диференціали. Автори роботи зробили так, що ці увови вирішувалися на етапі модифікації повідомлення, використовуючи реалізацію переносу на лівій та правій грані «бумерангу». Кінцеві результати для оцінки часової склданості: 33-раундова версія - 2^{125}, 35-раундова версія - $2^{117.1}$.

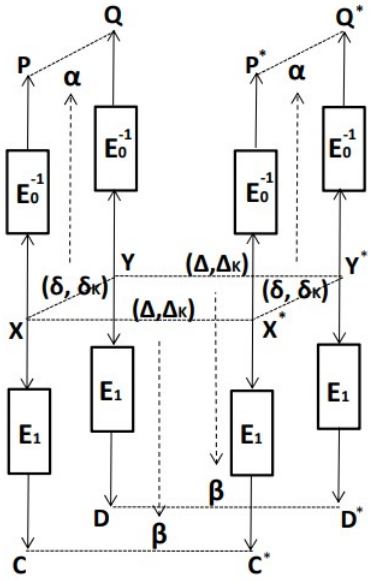


Рисунок 1.1 – Схематичне представлення методу «бумеранг»

1.3 Усічені варіанти геш-функцій

Для кращого розуміння певних аспектів криптоаналізу деяких криптопримітивів, іноді, застосовують масштабування цільового алгоритму (toy-версія) зі збереженням його стійкості. Такий підхід також застосовний до алгоритмів у асиметричній криптографії, де криптоаналітки намагються вирішити якомога більше випадків складної проблеми та екстраполювати результати для оцінки задачі в цілому.

Toy Sponge. Геш-функція Sponge [10] має доволі просту та гнучку структуру, що дозволяє будувати на її основі нові криптопримітиви. Стійкість Sponge базується на будові внутрішньої функції, яка часто обирається у вигляді ітераційної перестановки П.

Схематичний вигляд побудови цієї геш-функції зображений на Рисунку 1.2. Структура геш-функції умовно поділяється на два етапи: *поглинаюча фаза (absorbing phase)* та *стискаюча фаза (squeezing phase)*. На етапі поглинаючої фази вхідне повідомлення розбивається на блоки повідомлень $m_0, m_1, \ldots, m_{n-1}$ однакової довжини r біт. Кожен цей блок сумується за модулем 2 із станом геш-функції розміром $r + c$, де c - це

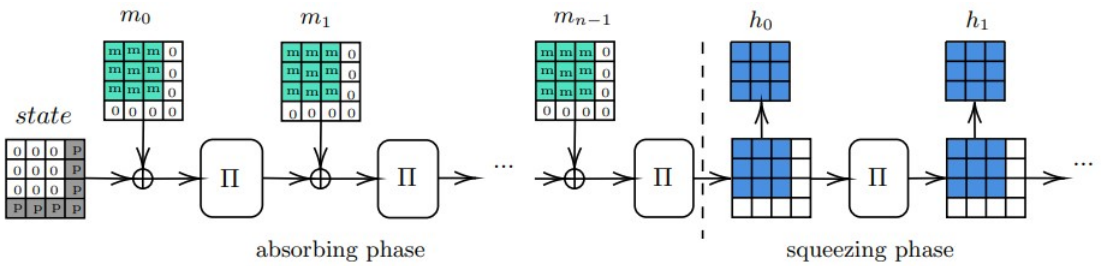


Рисунок 1.2 – Структура Sponge

об'єм Sponge. Зауважемо, що $r + c$ - це також і розмір вхідної та вихідної перестановки П. Початковий стан ініціалізується в залежності від фіксованого відомого значення для першої ітерації. Після того, як блоки повідомлень пройдуть поглинаючу фазу, починається стискаюча фаза у вигляді створення необхідної кількості вихідних блоків $h_0, h_1, \ldots$ розміром r, для досягнення потрібного розміру геш-значення.

В якості першого кроку потрібно впровадити просту ARX перестановку, конструкція якої є схожою із внутрішньою перестановкою у потоковому шифрі ChaCha20 [11]. Внутршній стан перестановки ChaCha20 представляє собою матрицю 32-бітних слів розміром 4×4. Дана перестановка отримується шляхом ітерації певної кількості раундів. Кожен раунд діє певним чином на колонки та діагоналі матриці стану за допомгою функції *QuarterRound* (QR). Так як квантовий симулятор, що використовувався у роботі міг керувати тільки станом розміром 16 біт, автори реалізували Toy Sponge, використовуючи матрицю стану 4-бітних слів розміром 2×2. Функція QR приймала на вхід два слова та повертала також два слова. Реалізація у вигляді цієї функції у вигляді псевдокоді представлена на наступних рисунках:

$\mathsf{QR}(a, b)$:
  $a = a + b$
  $b = (b \oplus a) \lll 2$
  $a = a + b$
  $b = (b \oplus a) \lll 1$
  return a, b

Рисунок 1.3 – Функція QR

$\mathsf{ColQR}(v)$:
  # update columns
  $v[0], v[2] = \mathsf{QR}(v[0], v[2])$
  $v[1], v[3] = \mathsf{QR}(v[1], v[3])$
  return v

Рисунок 1.4 – Оновлення колонок матриці стану

$\mathsf{DiagQR}(v)$:
  # update diagonals
  $v[0], v[3] = \mathsf{QR}(v[0], v[3])$
  $v[1], v[2] = \mathsf{QR}(v[1], v[2])$
  return v

Рисунок 1.5 – Оновлення діагоналі матриці стану

Для спрощення квантової симуляції було зроблене наступне масштабування: поглинання 8-бітового повідомлення та стиснення від 4 до 8-бітового геш-виходу може бути реалізовано за одну ітерацію внутрішньої перестановки. Toy-версія даної геш-функції передбачає

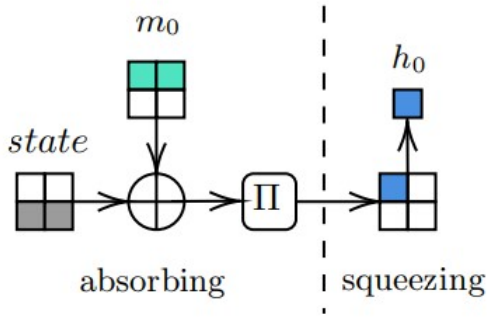


Рисунок 1.6 – Схема Toy Sponge

використання блоку повідомлення розміром $r = 8$ біт та об'єм Sponge, також, $c = 8$ біт. В якості перестановки використовується $ChaCha_\pi$ із потокового шифру ChaCha20.

Toy BLAKE. Даний криптопримітив [12] представляє собою ARX криптографічну геш-функцію зі структурою, яка надає можливість ефективної програмної реалізації. Основною перестановкою є спеціально налаштована перестановка, що використовується у потоковому шифрі ChaCha20.

Перед початком роботи, вхідне повідмолення m розбивається на n блоків $d_0, d_1, \ldots, d_{n-1}$, які є матрицями 64 або 32-бітних слів розміром 4×4. Кожен блок вводиться у функцію стиснення F разом із 8-ма словами стану h, котрий далі оновлюється за допомогою F. Функція стиснення F ініціалізує матрицю ϑ розміром 4×4, використовуючи стан h та ініціалізаційний вектор $i\vartheta$. Далі, для ρ раундів, функція стиснення F діє на колонки, а потім на діагоналі матриці ϑ за допомогою функції змішування G, яка також приймає перестановку поточного блоку як вхід. Структура геш-функції BLAKE та її функція стиснення зображені на Рисунку 1.7.

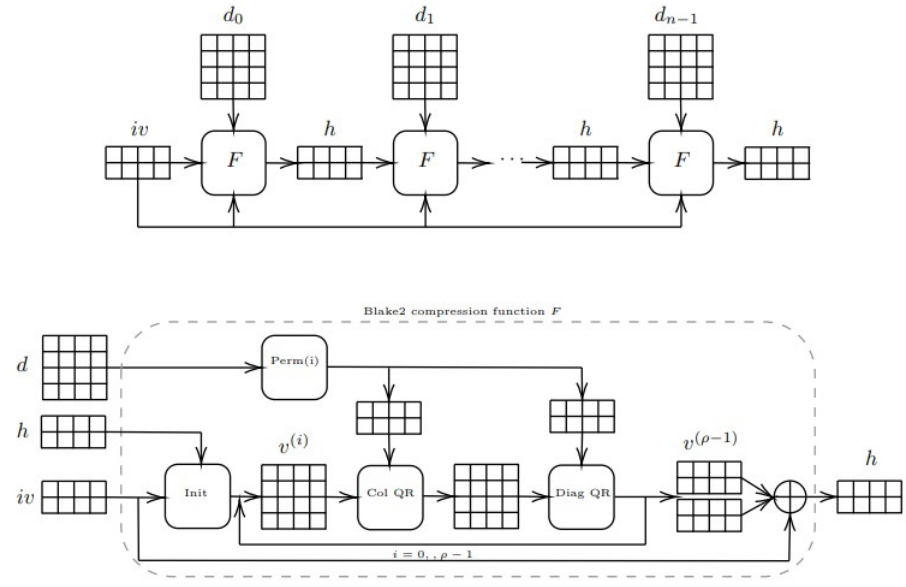


Рисунок 1.7 – Структура BLAKE

Для масштабування до Toy BLAKE необхідно зменшити розмір слова до 4 біт, а блоки повідомлень представити у вигляді матриць розміром 2×2 із 4-бітними словами. Toy-версія представлена на Рисунку 1.8.

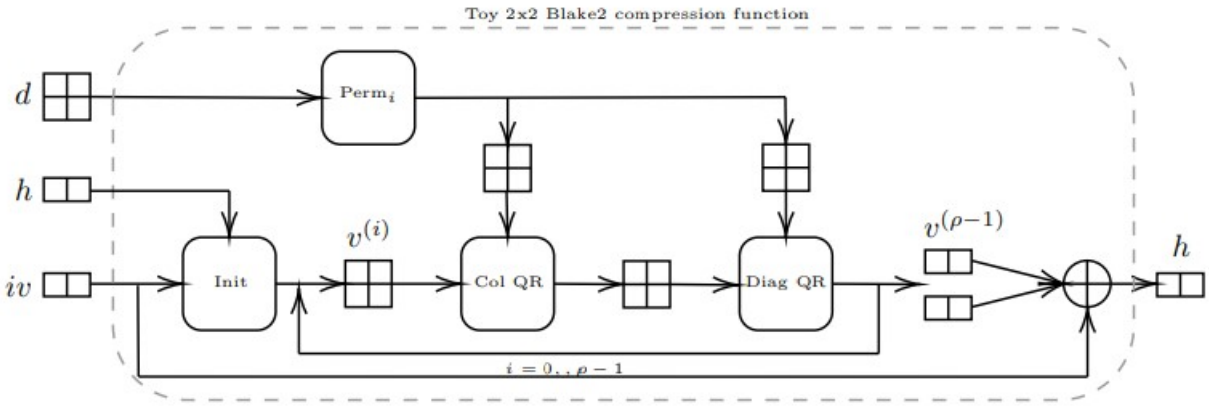


Рисунок 1.8 – Функція стиснення Toy BLAKE

Параметри h, $i\vartheta$ та ϑ визначаються наступним чином:

$$i\vartheta[0] = 0x8, i\vartheta[1] = 0xB,$$
$$\mathrm{h}[0] = \mathrm{i}\vartheta[0] \oplus 0x2, h[1] = i\vartheta[1],$$
$$\vartheta[0] = h[0], \vartheta[1] = h[1] \oplus (4 \ll t), \vartheta[2] = i\vartheta[0], \vartheta[3] = i\vartheta[1]$$

Змінна t визначає поточний блок 2×2 який обробляється, а ϑ ініціалізуєтсья на початку кожного раунду.

Функція внутрішнього змішування геш-функції Toy BLAKE визначається наступним чином:

$G(a, b, x, y)$:
$a = a + b + x$
$b = (b \oplus a) \ggg 2$
$a = a + b + y$
$b = (b \oplus a) \ggg 1$
return a, b

ColQR(v, d) :
update columns
$v[0], v[2] = G(v[0], v[2], d[s[0]], d[s[1]])$
$v[1], v[3] = G(v[1], v[3], d[s[2]], d[s[3]])$
return v

DiagQR(v, d) :
update diagonals
$v[0], v[3] = G(v[0], v[3], d[s[0]], d[s[1]])$
$v[1], v[2] = G(v[1], v[2], d[s[2]], d[s[3]])$
return v

Рисунок 1.9 – Функція внутрішнього змішування G

1.4 Структура геш-функції «Купина»

Структура геш-функції «Купина» складається з багатьох компонент, які необхідно розібрати, щоб зрозуміти, як даний криптопримітив працює в цілому.

SPN-структури. SPN-структура (substitution-permutation network)[13] є найпопулярнішою вискорівневою конструкцією ітеративних блокових шифрів. SPN-перетворення застосовується у багатьох криптопримітвах, таких як AES/Rijndael, «Калина».

Основними компонентами цієї констркуції є операції перемішування (diffusion) та розсіювання (confusion), що реалізовані за допомогою лінійних та нелінійних перетворень відповідно. Такий приниц побудови дає гарантію дотримання приниципів Шеннона.

Однією із особливостей SPN-структури є неможливість реалізації за допомогою апарату випадкових функцій. Це випливає с того, що для коректної процедури розшифрування у раундовому перетворенні необхідно використовувати бієктивні відображення (або перестановки).

Відштовхуючись від внутрішньої конструкції алгоритмів Square та AES/Rijndael, блок, який буде використовуватись у шифрі необхідно реалізовувати у вигляді матриці розміром $n_c \times n_b$ елементів, де розмір кожного елементу буде становити n_e біт, тоді як розмір самого блоку буде $2n = n_c \times n_b \times n_e$ біт. $n_c = n_b \cdot 2l$, $l \geqslant 1$ обирається таким чином, щоб було дотримано ряд моментів, які забезпечують стійкість шифру, та, щоб можно було реалізувати ефективну програмну версію криптопримітиву. Параметр n_e часто обирають рівним 8, враховуючи, що сучасні шифри мають байтову реалізацію.

У раундовому перетворенні SPN-структури використовуються випадкові перестановки для нівелювання впливу конкретних компонент. Така випадкова перестановка називається Super-S-box [14].

Процес шифрування на основі SPN-структури має наступний вигляд:

Перший блок у цій схемі відповідає за шар перестановок, а другий - за шар

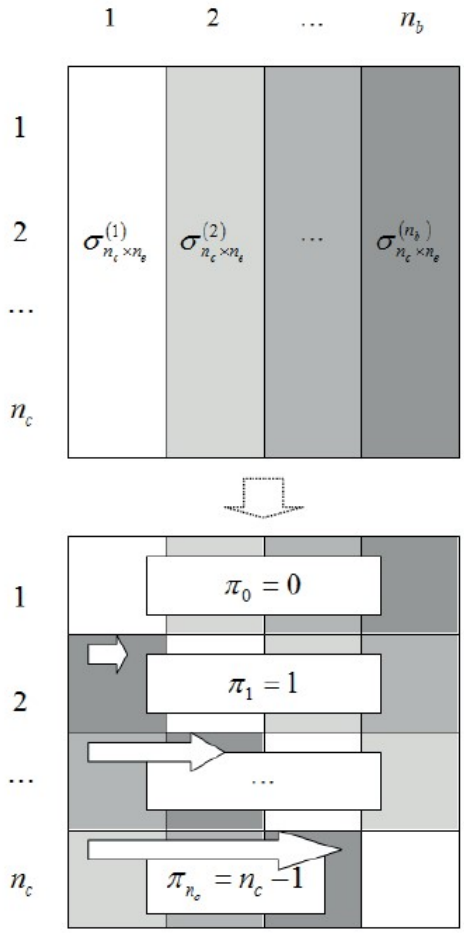


Рисунок 1.10 – Шифрування на основі алгоритму із SPN-перетворенням

обміну елементів між колонками матриці стану.

Отже, конструкцію шифруючого перетворення SPN-структури можна описати як послідовне застосування шару випадкових перестановок та процес обміну елементів у колонках матриці стану.

Для опису загального випадка, модель s-циклового шифру представляють у вигляді наступної формули

$$\vartheta = \prod_{i=1}^{s} (\pi_0 \cdot \pi_1 \cdot \ldots \cdot \pi_{n_c}) \circ (\sigma^{(1,i)}_{n_c \times n_e} \cdot \sigma^{(2,i)}_{n_c \times n_e} \cdot \ldots \cdot \sigma^{(n_b,i)}_{n_c \times n_e}) \qquad (1.3)$$

Шифруюче перетворення приймає на вході значення $x_i = (x_i^{(1)} \cdot x_i^{(2)} \cdot \ldots \cdot x_i^{(n_b)})$, а на виході отримується $y_i = (y_i^{(1)} \cdot y_i^{(2)} \cdot \ldots \cdot y_i^{(n_b)})$.

Схема Меркля-Дамгора. Дана схема [15] є методом/правилом побудови геш-функції, яка має наступні властивості:

– стійкість функції стиснення до колізійних атак;

– стійкість до колізій в цілому.

Для забезпечення вищенаведених властивостей, метод передбачає розбиття вхідного повідомлення на деяку кількість блоків, що мають однакову довжину, та послідовну роботу з кожним, використовуючи функцію стиснення. На кожній ітерації функція стиснення приймає на вхід поточний блок повідомлення та результуючий блок із попередньої ітерації. Робота цієї конструкції представлена на Рисунку 2.1.

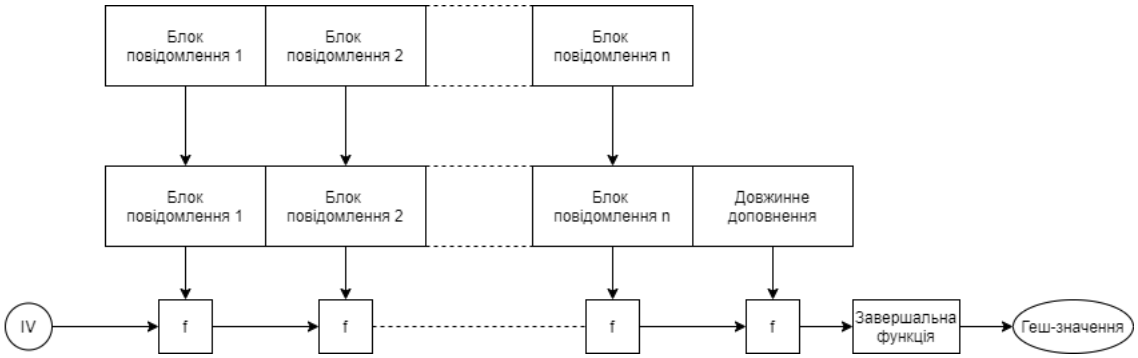


Рисунок 1.11 – Схема Меркля-Дамгора

Перед початком роботи, необхідно застосувати МД-сумісне перетворення задля доповнення вхідного повідомлення M до певної довжини. Це робиться по двум причинам: забезпечення відповідного рівня стійкості алгоритму та випадку, коли криптопримітив не має можливості обробляти вхід певної довжини. Після цього, доповнене повідомлення M' розбивається на певну кількість блоків, яка прописана у алгоритмі, однакової довжини. Після цих кроків функція стиснення може обробляти кожен блок повідомлень. На кожній ітерації функція стиснення приймає на вхід поточний блок повідомлення та результуючий блок із попередньої ітерації. Завершальним етапом роботи схеми є застосування методу довжинного доповнення (або підсилення Меркля-Дамгора), який кодує довжину вхідного повідомлення M. Це

робиться для забезпечення гарного рівня стійкості геш-функції.

На Рисунку 2.2 функція стиснення позначена f. f є відображенням із векторного простору розміру l у такий же простір, тобто $f : V_l \to V_l$. Для почтаку роботи алгоритму необхідно подати деяке вхідне значення - інілізаційний вектор (IV).

В кінці роботи алгоритму, для кращого змішування та гарного рівня лавинного ефекту, результат необхідно подати обробку завершальній функції. Це, також, робиться для ускладнення криптоаналізу.

Структура Девіса-Мейєра. Дана структура [16] є прикладом односторонньої функції стиснення, що будуэться на основы блокового шифру. Схема складається з блокового шифру E на вхід якому подається блок повідомлення m_i та попереднє геш-значення H_{i-1} в якості ключа й блоку відкритого тексту відповідно. Наприкінці, для того, щоб отримати наступне геш-значення H_i результуючий зашифрований блок складається за модулем 2 із значенням на попередній ітерації H_{i-1}.

Отже, математично, дана конструкція представляється у вигляді наступного співвідношення:

$$H_i = E_{m_i}(H_{i-1}) \oplus H_{i-1}. \tag{1.4}$$

Схематично, дану структуру, можна представити наступним чином:

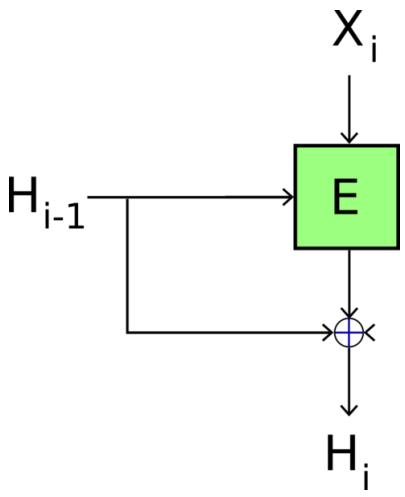


Рисунок 1.12 – Одностороння функція стиснення Девіса-Мейєра

Схема Евен-Мансура. Дана схема є прикладом мінімальної конструкції блокового шифру зі збереженням його стійкості [17][18]. Під мінімальністю розуміється число елементів у схемі шифру, а під стійкістю - будь-яку формально вірну оцінку знизу складностей реалізацій атак на цей шифр.

Схема використовує тільки одну підстановку π із множини S_{2^n} яка обирається випадково. За умовою побудови криптопримітиву вважається, що частина деякого ключа не може включатися у перестановку.

Нехай задано множини $\mathcal{P}$, що є множиною відкритих текстів, та - множина закритих текстів (шифротекстів). За побудовою $\mathcal{P} \equiv$. Нехай, також, є деяка обрана підстановка π із множини $S_{|\mathcal{P}|}$ ($S_{|\mathcal{P}|}$ - це множина, яка включає в себе всі можливі підстановки над множиною $\mathcal{P}$) та визначен обернена підстановка до π - π^{-1}. Перестановка будь-якого елементу із множини $\mathcal{P}$ та обернена перестановка будь-якого елементу із $\mathcal{C}$, за умовою, легко обчислюється просто як значення відповідних підстановок. Також є варіант обчислення через виклики деяких оракулів $\mathbf{P}_\pi$ і $\mathbf{P}_\pi^{-1}$, що задаються на початку роботи алгоритму.

Множини $\mathcal{P}$ та задаються як бітові послідовності довжини n - $\{0,1\}^n$, а множина ключів, є множиною бітових векторів довжини $2n$ - $\{0,1\}^{2n}$. Особливість побудови цієї схеми - це представлення секретного ключа $\underline{K}$ у виді кортежу із підключів $\underline{K} = \langle \underline{K_1}, \underline{K_2} \rangle$. Вибір кожного підключа відбувається із імовірністю $\frac{1}{2^n}$ із множини $\{0,1\}^n$.

$\underline{K}$ зберігається у таємниці й використовується тільки підтвердженими користувачами системи. Відповідно є ключем для зашифруваня та розшифрування.

Процедура шифрування відкритого тексту P за допомогою секретного ключа $\underline{K} = \langle \underline{K_1}, \underline{K_2} \rangle$ та фіксованої підстановки π описується наступною формулою:

$$E_{\underline{K}}(P) = E(P, \langle \underline{K_1}, \underline{K_2} \rangle) = \pi(P \oplus \underline{K_1}) \oplus \underline{K_2}, \tag{1.5}$$

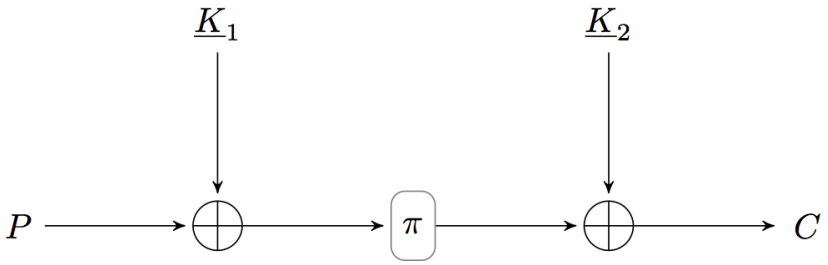


Рисунок 1.13 – Схема Евен-Мансура

а розшифрування шифротексту C за допомогою секретного ключа $\underline{K}$ та фіксованої підстановки відбувається наступним чином:

$$D_K(C) = D(C, \langle \underline{K_1}, \underline{K_2} \rangle) = \pi^{-1}(C \oplus \underline{K_2}) \oplus \underline{K_1}. \tag{1.6}$$

Геш-функція «Купина». «Купина» - це AES-подібна ітерована геш-функція зі структурою SPN, що використовую функцію стиснення Меркля-Дамгорда. Дана геш-функція може повертати геш-значення довжиною від 8 до 512 біт. Рекомендовані версії до використання - «Купина-256», «Купина-384», «Купина-512».

На вхід геш-функції подається деяке повідомлення M у бітовому форматі. Нехай довжина цього повідомленя дорівнює N. Спочатку перевіряється чи є N кратне l, де l - це довжина блоків, на які буде розбиватися повідомлення M. Значення l обирається в залежності від версії геш-функції:

$$l = \begin{cases} 512, 8 \leqslant n \leqslant 256 \\ \\ 1024, 256 < n \leqslant 512 \end{cases} \tag{1.7}$$

Якщо $N \not\vdots l$, тоді повідомлення M доповнюється до кратної довжини специфічним чином: у кінець повідомлення записується один одиничний біт, $d = (-N - 97)\ mod\ n$ нульових бітів, а також ще 96 біт, які відповідають довжині повідомлення N. Далі, доповнене повідомлення M' розбивається на k блоків $M_0, M_1, ..., M_{k-1}$.

Щоб згенерувати геш-значення необхідно виконати наступні кроки:

$CV_0 \leftarrow IV$,

$CV_{i+1} \leftarrow CF(CV_i, M_i), i = \overline{0, k-1}$,

$h = Trunc(T_l^{\oplus}(CV_k) \oplus CV_k)$,

де IV - це ініціалізаційний вектор, що залежить від l:

$$IV = \begin{cases} 1 <<< 510, l = 512 \\ 1 <<< 1024, l \neq 512 \end{cases} \tag{1.8}$$

$CF(CV_i, M_i)$ - функція стиснення, яка обраховується за наступною формулою:

$$CF_l(CV_i, M_i) = T_l^{\oplus}(CV_i \oplus M_i) \oplus T_l^{+}(M_i) \oplus M_i, \tag{1.9}$$

$Trunc$ - це функція, що повертає n значущих бітів із вхідного блоку повідомлення довжиною $l(n < l)$, а результат записується в молодші n біт обчисленого значення.

Схематично, отримання геш-значення представлено на Рисунку 2.4.

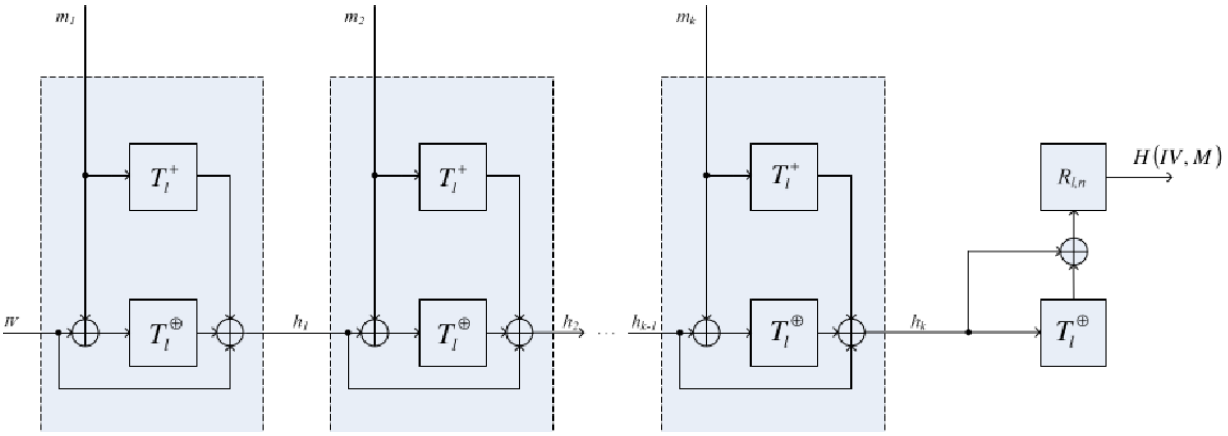


Рисунок 1.14 – Схема роботи геш-функції «Купина»

Перестановки $T_l^{\oplus}$ та T_l^{+} є бієктивними відображеннями виду $T_l^{\oplus}$, T_l^{+}:$V_l \rightarrow V_l$, $l = \{512, 1024\}$. Кожне відображення є композицією деяких функцій, що приймають в якості аргументу $x \in V_l$ матрицю розміром

$8 \times 8(l = 512)$ байтів, або $8 \times 8(l = 1024)$ байтів. Елементи матриці представляються у вигляді елементів поля $\mathbf{GF}(2^8)$.

Матриця внутрішнього стану визначається наступним чином: $G = (g_{i,j}), i = \overline{0,7}, j = \overline{0,c-1}$. Матриця вхідної послідовності G_{input} буде заповнюватися зверху-вниз вхідними байтами $B_1, B_2, ..., B_{\frac{l}{8}}$.

$$G_{input} = \begin{bmatrix} B_1 & B_9 & B_{17} & B_{25} & B_{33} & B_{41} & B_{49} & B_{57} \\ B_2 & B_{10} & B_{18} & B_{26} & B_{34} & B_{42} & B_{50} & B_{58} \\ B_3 & B_{11} & B_{19} & B_{27} & B_{35} & B_{43} & B_{51} & B_{59} \\ B_4 & B_{12} & B_{20} & B_{28} & B_{36} & B_{44} & B_{52} & B_{60} \\ B_5 & B_{13} & B_{21} & B_{29} & B_{37} & B_{45} & B_{53} & B_{61} \\ B_6 & B_{14} & B_{22} & B_{30} & B_{38} & B_{46} & B_{54} & B_{62} \\ B_7 & B_{15} & B_{23} & B_{31} & B_{39} & B_{47} & B_{55} & B_{63} \\ B_8 & B_{16} & B_{24} & B_{32} & B_{40} & B_{48} & B_{56} & B_{64} \end{bmatrix}$$

Для випадку $l = 512$ матриця G_{input} виглядає наступним чином:

$$G_{input} = \begin{bmatrix} 00 & 08 & 10 & 18 & 20 & 28 & 30 & 38 \\ 01 & 09 & 11 & 19 & 21 & 29 & 31 & 39 \\ 02 & 0a & 12 & 1a & 22 & 2a & 32 & 3a \\ 03 & 0b & 13 & 1b & 23 & 2b & 33 & 3b \\ 04 & 0c & 14 & 1c & 24 & 2c & 34 & 3c \\ 05 & 0d & 15 & 1d & 25 & 2d & 35 & 3d \\ 06 & 0e & 16 & 1e & 26 & 2e & 36 & 3e \\ 07 & 0f & 17 & 1f & 27 & 2f & 37 & 3f \end{bmatrix}$$

Для $l = 1024$ заповнення відбувається аналогічно.

Перестановки $T_l^{\oplus}$ та T_l^{+} визначаються за наступними формулами:

$$T_l^{\oplus} = \prod_{\vartheta=0}^{t-1}(\psi \circ \tau^{(l)} \circ \pi^{'} \circ \kappa_{\vartheta}^{(l)}), \tag{1.10}$$

$$T_l^{+} = \prod_{\vartheta=0}^{t-1}(\psi \circ \tau^{(l)} \circ \pi^{'} \circ \kappa_{\eta}^{(l)}), \tag{1.11}$$

де t - це кількість раундів, яка обирається також в залежності від l: якщо $l = 512$, то $t = 10$, якщо $l = 1024$, то $t = 14$,

$\kappa_{\vartheta}^{(l)}$-це функція, яка додає вектор $\omega_j^{(\vartheta)} = ((j << 4) \oplus \vartheta, 0, 0{,}0{,}0{,}0{,}0{,}0{,}0)^T, \omega_j^{(\vartheta)} \in V_{64}$ до кожного стовпця матриці G за $mod\ 2$, ϑ - це номер ітерації,

$\eta_{\vartheta}^{(l)}$ - додає вектор $\zeta_j^{(\vartheta)} = (0xF3{,}0xF0{,}0xF0{,}0xF0{,}0xF0{,}0xF0{,}0xF0{,}((c{-}1{-}j) << 4) \oplus \vartheta)^T, \zeta_j^{(\vartheta)} \in V_{64}$ до кожного стовпця матриці внутрішнього стану G за модулем 2^{64}, прицьому при додаванні $0xF3$ - молодші 8 біт вектору $\zeta_j^{(\vartheta)}$, $g_{0,j}$ - молодші 8 біт вектору G_j,

$\pi^{'}$ - виконує заміну кожного елементу матриці внутрішнього стану $G = (g_{i,j})$ на елемент підстановки π_{imod4},

$\tau^{(l)}$ - виконує циклічний зсув вправо елементів матриці внутрішнього стану в залежності від номеру рядка: елементи у рядках з номерами $i = 0{,}1{,}2{,}3{,}4{,}5{,}6$ будуть зсуватися на i елементів відповідно, а елементи у 7 рядку будуть зсуватися на 7 елементів, якщо $l = 512$, або на 11, якщо $l = 1024$,

ψ - для застосування цієї функції, кожен елемент матриці внутрішнього стану $G = (g_{i,j})$ переводиться у поле $\mathbf{GF}(2^8)$, яке утворене незвідним поліномом $\vartheta(x) = x^8 + x^4 + x^3 + x^2 + 1$.

Результуюча матриця стану $U(x) = (u_{i,j})$ отримується за наступною формулою:

$$u_{i,j} = (\nu >>> i) \otimes G_j, \tag{1.12}$$

де $\nu = (0x01,0x01,0x05,0x01,0x08,0x06,0x07,0x04)$ - вектор, що утворює циркулятну матрицю МДР-коду.

Висновки до розділу 1

У розділі розглянуто геш-функції на які було здійснено успішні атаки у квантовій моделі обчислень. Криптоаналіз проводився за допомогою алгоритму Гровера для двох типів криптопримітивів: звичайний варіант та усічений (*toy*-варіант). Щодо усічених варіантів розглянуті дві геш-функції: BLAKE2 та Sponge. Їх структура дозволила зрозуміти до якого моменту потрібно проводити процедуру масштабування так, щоб зберігався порядок стійкості криптопримітиву.

Другою частиною розділу є опис необхідних компонент на яких будується криптографічна геш-функція національного стандарту України - «Купина». Після проведення аналізу структури цього криптопримітиву було виділено такі складові побудови:

– Структура Меркля-Дамгора - є головною конструкцією для отримання геш-значення.

– Високорівнева конструкція Девіса-Майєра - ця схема є одної із компонент для реалізації ітерації; за допомогою неї відбувається представлення односторонньої функції у виді деякого блокового шифру.

– Схема Евен-Мансура - це конструкція, яка дозволяє проектувати мінімальну за будовою схему блокового шифру; також є компонентою для реалізації ітерації.

Маючи опис цих складових, вдалося зрозуміти принципи роботи та внутрішню будову геш-функції «Купина» та провести паралелі із деякими шифрами, такими як AES та «Калина».

2 КВАНТОВИЙ МЕТОД КРИПТОАНАЛІЗУ ЗА ДОПОМОГОЮ АЛГОРИТМУ ГРОВЕРА

Цей розділ включає необхідну теорію, яка використовується для реалізації низки атак на криптопримітиви у квантовій моделі. В якості прикладу використання квантового алгоритму пошуку проаналізовано атаку на геш-функцію SM3.

2.1 Квантова модель обчислень

У якості основного інструменту квантова модель використовує математичний апарат лінійної алгебри. Фундаментом квантових обчислень є той факт, що носіями інформації є квантово-механічні системи, отже, всі їх можливі стани та перетворення описується за допомогою постулатів квантової механіки. Наведемо основні визначення об'єктів та операцій над ними.

Позначемо за $\mathcal{H}^d$ - d-мірний комплексно-лінійний векторний простір в якому визначено операцію скалярного множення. Такий простір ще називають *гільбертовим*. Належність елементів до цього простору вводять за допомогою нотації Дірака *«бра-кет»*:

$$\langle\psi| \text{ - вектор рядок,}$$
$$|\psi\rangle \text{ - вектор стопчик.}$$

Вираз $\langle\psi_1|\psi_2\rangle$ позначає скалярне множення.

Означення 2.1. Тензорне множення. $A \otimes B$ - *тензорне(праве кронекерове) множення* матриць A та B.

Якщо $A = \begin{pmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \dots & a_{m,n} \end{pmatrix}$, $B = \begin{pmatrix} b_{1,1} & \dots & b_{1,k} \\ \vdots & \ddots & \vdots \\ b_{l,1} & \dots & b_{l,k} \end{pmatrix}$, тоді, за означенням, $A \otimes B = \begin{pmatrix} a_{1,1}B & \dots & a_{1,n}B \\ \vdots & \ddots & \vdots \\ a_{m,1}B & \dots & a_{m,n}B \end{pmatrix}$.

Якщо задано певну кількість векторів, нехай q, тоді, для позначення тензорного множення $|a_1\rangle, |a_2\rangle, ..., |a_q\rangle \in \mathcal{H}^2$ використовують наступний запис:

$$|a_1\rangle \otimes ... \otimes |a_q\rangle = |a_1,...,a_q\rangle \in \mathcal{H}^{2q}.$$

Означення 2.2. Стандартним обчислювальним базисом в $\mathcal{H}^2$ є базис, що задається наступними векторами:

$$|0\rangle = \begin{pmatrix} 1 \\ \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ \\ 1 \end{pmatrix}$$

Для випадку $\mathcal{H}^d$, $d > 2$ вводяться наступні записи кожного стану:

$$\begin{gathered} |1\rangle = |00\dots0\rangle = (100\dots0\dots0)^T \\ |2\rangle = |00\dots1\rangle = (010\dots0\dots0)^T \\ \vdots \\ |i\rangle = |(i-1)_2\rangle = (000\dots1\dots0)^T \\ \vdots \\ |d\rangle = |11\dots1\rangle = (000\dots0\dots1)^T \end{gathered}$$

Перевірка даних рівностей робиться за допомогою послідовного застосування операції скалярного множення: наприклад, ми розглянемо вектор стовпчик, що відповідає $|010\rangle = |3\rangle$:

$$|10\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 1 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$|010\rangle = |0\rangle \otimes |10\rangle = \begin{pmatrix} 1 \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \\ 0 \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = |3\rangle$$

Отже, можна зробити висновок, що така система базисних векторів $|1\rangle, |2\rangle, \ldots |d\rangle$ є ортанормованою у просторі $\mathcal{H}^d$. Дана система прийнята за стандартний обчислювальний базис. Всі подальші розрахунки проводяться виключно у цьому базисі.

Розглянемо таке поняття, як *простір станів*. Посилаючись на перший постулат, із будь-якою ізольованою фізичною системою можна поставити у відповідність деякий гільбертів простір, що буде мати назву *простір станів*. У даному випадку стан системи буде повністю визначатися вектором стану - це одиничний вектор, що належить простору станів.

Означення 2.3. Кубіт.Кубіт визначається як об'єкт гільбертового простору розмірності 2 $\mathcal{H}^2$, що представляє собою квантово-механінчу

систему, яка задається у вигляді наступного вектору:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle,$$

$|0\rangle$ та $|1\rangle$ позначаються вектори, які є ортонормованим базисом гільбертового простору $\mathcal{H}^2$, а α та β є комплексними константами підібраними так, щоб задовольняти умову нормування $|\alpha|^2 + |\beta|^2 = 1$.

Отже, таке визначення показує основну відмінність між класичним бітом та кубітом: кубіт має властивість знаходитися у суперпозиції своїх станів, що представлені векторами $|0\rangle$, $|1\rangle$, а комплекснозначені числа α та β виступають у ролі амплітуд.

Використовуючи той факт, що α та β є комплексними числами, можна позначати загальний стан кубіту $|\psi\rangle$, використовуючи поняття *фази* θ та *фазового множника* $e^{i\phi}$. Це представлення виглядає наступним чином:

$$|\psi\rangle = \cos(\tfrac{\theta}{2})\,|0\rangle + e^{i\phi}\sin(\tfrac{\theta}{2})\,|1\rangle,$$

де $0 \leqslant \phi < 2\pi, 0 \leqslant \theta \leqslant \pi$.

Якщо зробити деякі перевозначення, то можна прийти до геометричної інтерпретації «поведінки» кубіту. Позначемо за $e^{i\phi}\frac{\theta}{2} = x + iy$ та $z = \cos(\frac{\theta}{2})$. Звідси можна отримати, що виконується рівність $x^2 + y^2 + z^2 = 1$. Отже, виходить так, що стан кубіту можна представити у вигляді положення точки на сфері із полярними координатами θ і ϕ. Утворений об'єкт носить назву у квантовій моделі, як *сфера Блоха*.

Така геометрична побудова дає можливість легко описувати положення кубіту у будь-який момент часу. А у випадку, коли значення α та β є дійсними числами, тоді, очевидно, що стан кубіту описується положенням точки на колі в $\mathbb{R}^2$.

Квантові схеми. Нехай є деяка модель обчислень, яка може обробляти вхідні слова фіксованої довжини. Далі, опис квантових схем буде розглянутий у розрізі саме таких моделей обчислень. Тенденції квантових обчислень показують, що використання квантових схем у розрахунках є одним із найпопулярнішим підходом на даний час.

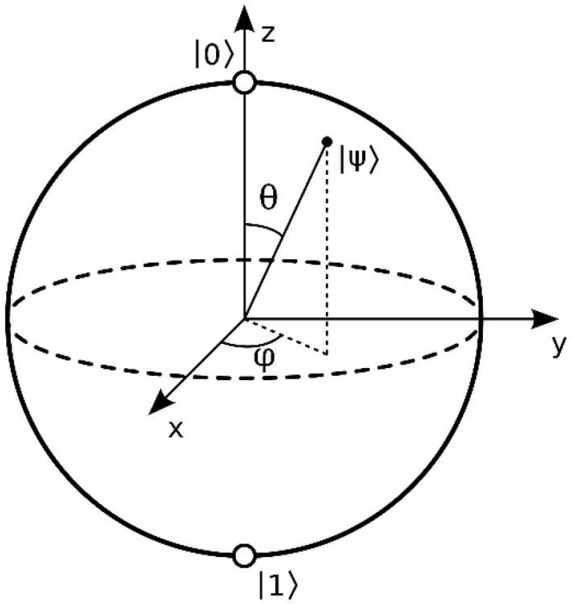


Рисунок 2.1 – Сфера Блоха

Основною компонентою квантової схеми є об'єкт, що має назву *квантовий вентиль*.

Означення 2.4. *Квантовим вентилем на q кубітах називають об'єкт, що є унітарним відображенням у гільбертовому просторі* $\mathcal{H}^{2q} = \mathcal{H}^2 \otimes \ldots \otimes \mathcal{H}^2$, *яке діє нетривіальним чином на фіксоване (незалежне від q) число кубіт.*

Означення 2.5. *Квантова схема на q кубітах - це унітарне відображення у гільбертовому просторі* $\mathcal{H}^{2q} = \mathcal{H}^2 \otimes \ldots \otimes \mathcal{H}^2$, *яке може бути представлено як послідовне з'єднання кінцевого числа квантових вентелей.*

У загальному випадку квантова схема може представлятися наступним чином:

2.2 Квантові вентилі

Для моделювання обчислень у класичниій моделі квантові комп'ютери використовують низку різних вентилів. На даний момент

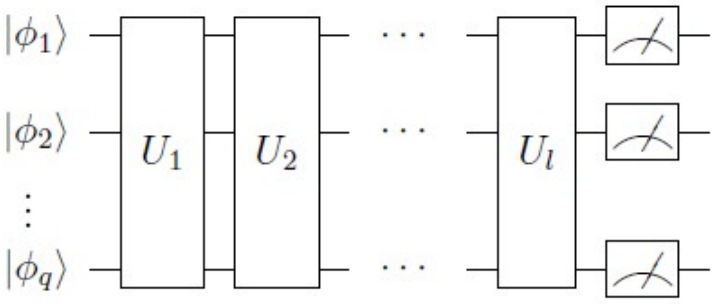


Рисунок 2.2 – Приклад квантової схеми

найбільш популярними є CNOT та Тоффолі.

CNOT.Цей логічний елемент має двухкубітову основу, де один вхідний кубіт виступає у ролі керуючого, а інший - керованого. Схематично такий вентиль можна представити наступним чином:

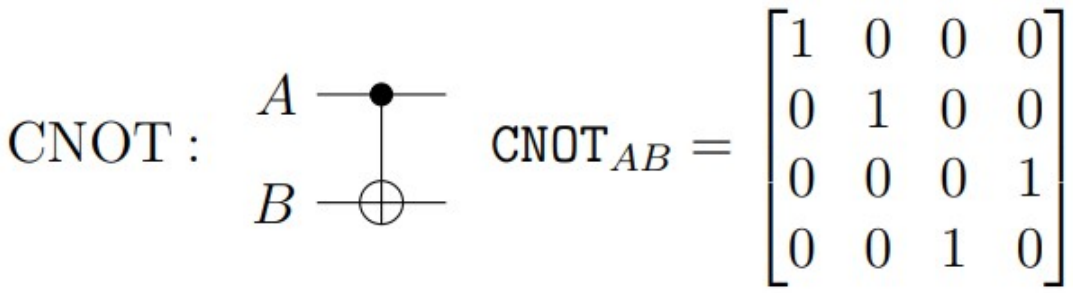


Рисунок 2.3 – Схематичне та матричне зображення CNOT

На схемі верхня лінія є керуючим кубітом, а нижня - керованим кубітом. Процесс обчислення описується наступним правилом: керований кубіт не змінює свого значення у випдаку, коли керуючий кубіт приймає значення 1. У випдаку, коли керуючий кубіт є 1, значення керованого кубіту повинно змінюватися. Це можна описати наступною послідовністю перетворень станів:

$$|00\rangle \to |00\rangle\,, |01\rangle \to |01\rangle\,, |10\rangle \to |11\rangle\,, |11\rangle \to |00\rangle.$$

Перехід станів можна переписати і через операцію XOR: $|A,B\rangle \to |A,B \oplus A\rangle$. Отже, кубіт A залишається незімінним, а у результат операції записується сума за модулем 2 керуючого та керованого кубіту.

Також, для реалізації вентилів використовується матричне представлення. Матриця для CNOT представлена на Рисунку 2.7. Матриця будується таким чином, що кожен стовпчик представляє собою опис перетворень, які відбуваються зі станами.

Вентиль Тоффолі. Даний вентиль представляє собою схему на трьох вхідних та вихідних кубітах. Було доведено, що використовуючи тільки цей логічний елемент, можна будувати будь-які логічні схеми у класичній моделі. Використовується у квантовій моделі для побудови оборотних схем. Схематично елемент представляється наступним чином:

Inputs			Ouputs		
a	b	c	a'	b'	c'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

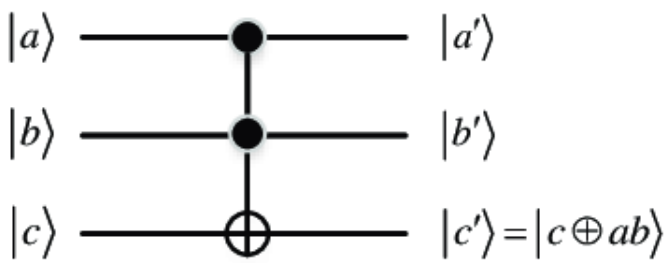


Рисунок 2.4 – Схематичне та матричне зображення Toffoli вентиля

У схемі наявні два керуючих кубіта, які не змінюються протягом роботи елемента. Третій біт є керованим. Його значення змінюється тільки у випадку, коли два керуючих кубіта дорівнюють 1 одночасно. Також, даний елемент має властивість оборотності, тобто:

$$(a,b,c) \rightarrow (a,b,c \oplus a \cdot b) \rightarrow (a,b,c)$$

Конструкція елементу Тоффолі дозволяє використовувати його для побудови вентилів NAND, та моделювати виконання операції FANOUT. А за допомогою цих операцій можна моделювати будь-які операцій у класичній моделі обчислень.

Елемент Тоффолі також можна побудувати й у квантовій моделі. У

цьому випадку квантова реалізація вентиля Тоффолі змінює стан обчислювального базису аналогічно до класичного варіанту вентиля.

2.3 Алгоритм квантового пошуку (алгоритм Гровера)

Для кращого розуміння роботи алгоритму пошуку має сенс почати з формулювання за допомогою *оракулу*. Такий підхід к опису алгоритму дозволить повністю описати процедуру пошуку та зробити геометричну інтерпретацію роботи кожного кроку алгоритму [2].

Оракул. Нехай є деякий N-елементний простір по якому можна виконувати процедуру пошуку. Основна ідея алгоритму полягає у зміні процедури пошуку не по конкретним елементам, а по їх індексам. Тобто, пошук виконується за номерами від 0 до $N-1$. Для роботи у бітовому форматі вважаємо, що $N = 2^n$. Для опису, також, потрібно ввести поняття, як кількість рішень задачі пошуку M. За умовою M буде лежати у діапазоні $1 \leqslant M \leqslant N$. У загальному випадку дана задача описується деякою функцією $f(x)$, $x = \{0,1,...,N-1\}$ яка приймає наступні значення:

$$f(x) = \begin{cases} 1, & x \text{ - розв'язок задачі} \\ 0, & \text{у протилежному випадку} \end{cases} \tag{2.1}$$

Процедура алгоритму. В основі алгоритму лежить лише одни регістр на n кубітах. У кінцевому результаті потрібно знайти розв'язок, що використовує мінімальну кількість викликів оракулу.

$|0\rangle^{\otimes n}$ визначає початковий стан комп'ютеру. Використовуючи перетворення Адамара відбувається перевденння стану комп'ютера у наступний стан:

$$|\psi\rangle = \frac{1}{N^{\frac{1}{2}}} \sum_{x=0}^{N-1} |x\rangle \,. \tag{2.2}$$

Наступним кроком є послідовне застосування квантової підпрограми

- *ітерації(оператор) Гровера.* Позначемо як G.

Ітерація Гровера складається з 4 кроків:

1) застосування оракулу O,

2) застосування перетворення Адамара $H^{\otimes n}$,

3) застосування до регістру умовного зсуву фази - кожний стан обчислювального базису, за винятком стану $|0\rangle$, набуває фазовий зсув -1:

$$|x\rangle \to -(-1)^{\delta_{x0}}|x\rangle,$$

4) застосування перетворення Адамара $H^{\otimes n}$.

Кроки 2, 3 та 4 можуть будти описані наступним співвідношенням:

$$H^{\otimes n}(2\,|0\rangle\,\langle 0| - I)H^{\otimes n} = 2\,|\psi\rangle\,\langle\psi| - I, \tag{2.3}$$

де $|\psi\rangle$ - це стани, які мають однакову вагу, та знаходяться у суперпозиції. Таким чином, ітерацію Гровера можна записати як $G = (2\,|\psi\rangle\,\langle\psi| - I)O$.

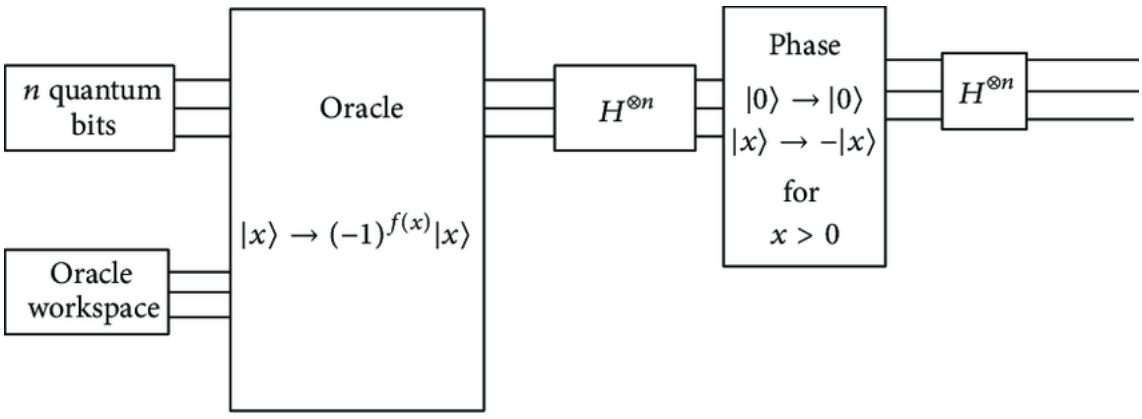


Рисунок 2.5 – Квантова схема алгоритму Гровера

Візуальне представлення алгоритму. Основною ідею для візуального представлення алгоритму пошуку є інтерпретування ітерації Гровера як *повороту* у двовимірному просторі. Цей простір породжений двома компонентами: початковим вектором $|\psi\rangle$ та станом суперпозиції рішень задачі пошуку, що мають однакову вагу.

Введемо такі позначення: $\sum_x^{'}$ - це сума по всім x, що є розв'язками задачі пошуку, а $\sum_x^{''}$ - сума по всім x, що не є розв'язками задачі пошуку.

Ці нормовані стани мають наступний вигляд:

$$|\alpha\rangle \equiv \frac{1}{\sqrt{N-M}} \sum_{x}^{''} |x\rangle \,, \tag{2.4}$$

$$|\beta\rangle \equiv \frac{1}{\sqrt{M}} \sum_{x}^{'} |x\rangle \,. \tag{2.5}$$

Перезапишемо вектор $|\psi\rangle$:

$$|\psi\rangle = \frac{1}{N^{\frac{1}{2}}} \sum_{x=0}^{N-1} |x\rangle = \sqrt{1-\frac{M}{N}}\,|\alpha\rangle + \sqrt{\frac{M}{N}}\,|\beta\rangle \tag{2.6}$$

Отримали, що два вектори: $|\alpha\rangle$ та $|\beta\rangle$ породжуть простір в якому знаходиться початковий стан квантового комп'ютера.

Опишемо послідовно дії, що виконує оператор G. Є площина, яка породжена векторами $|\alpha\rangle$ та $|\beta\rangle$. У цій площині оракул O виконує відображення вектора $|\alpha\rangle$. Ця процедура описується так: $O(a\,|\alpha\rangle + b\,|\beta\rangle) = a\,|\alpha\rangle - b\,|\beta\rangle$. У свою чергу, оператор $2\,|\psi\rangle\langle\psi| - I$ виконує відображення вектору $|\psi\rangle$ у тій самій площині. За властивістю композиції відображень, приходимо до висновку, що застосування одного разу ітерації Гровера G представляє собою поворот у просторі. Звідси шукається й кут повороту. З виду коєфіцієнтів при $|\alpha\rangle$ та $|\beta\rangle$ у виразі для $|\psi\rangle$, можна переписати іх наступним чином: $\cos(\frac{\theta}{2}) = \sqrt{1-\frac{M}{N}}$, $\sin(\frac{\theta}{2}) = \sqrt{\frac{M}{N}}$. Звідси, у свою чергу, випливає, що $|\psi\rangle = \cos(\frac{\theta}{2})\,|\alpha\rangle + \sin(\frac{\theta}{2})\,|\beta\rangle$. Після того, як відбудуться два відображення, вектор $|\psi\rangle$ перейде у стан

$$G\,|\psi\rangle = \cos(\frac{3\theta}{2})\,|\alpha\rangle + \sin(\frac{3\theta}{2})\,|\beta\rangle \,, \tag{2.7}$$

а це й значить, що кут повороту - це θ. k-кратне застосування G виглядає наступним чином:

$$G^k\,|\psi\rangle = \cos(\frac{2k+1}{2}\theta)\,|\alpha\rangle + \sin(\frac{2k+1}{2}\theta)\,|\beta\rangle \,. \tag{2.8}$$

З цього всього випливає головна геометрична характеристика оператору G - це поворот на кут θ у просторі, утвореного векторами $|\alpha\rangle$ та $|\beta\rangle$. Застосовуючи ітерацію знову і знову буде все ближче повертати вектор стану до $|\beta\rangle$. У момент досить близького розташування вектору $G^k |\psi\rangle$ до $|\beta\rangle$, якщо зробити вимірювання, то можна отримати із досить великою імовірністю якийсь доданок $|\beta\rangle$. Після цього буде знайдене рішення задачі пошуку.

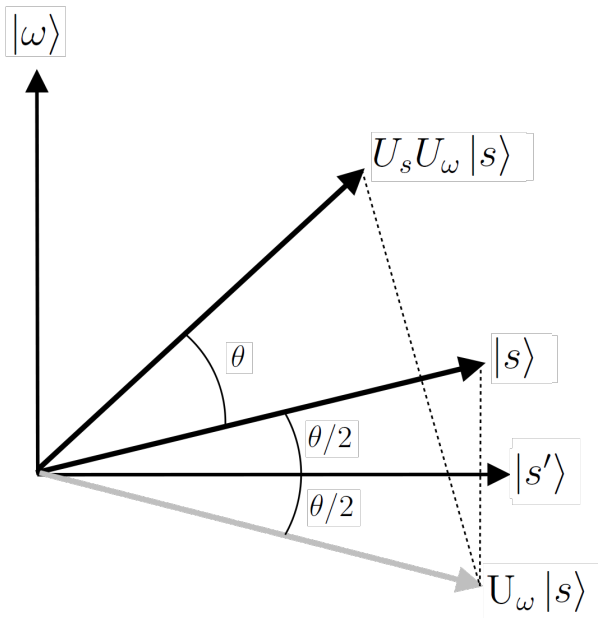


Рисунок 2.6 – Однократна дія ітерації Гровера

Задача пошуку вирішується з вискою імовіврністю у випдаку виконання $R = O(\sqrt{\frac{N}{M}})$ ітерацій Гровера(та запитів до оракулу)

Algorithm 2.1 Квантовий пошук($M = 1$)

1: Початковий стан: $|0\rangle^{\otimes n} |0\rangle$
2: Застосувати $H^{\otimes n}$ до перших n кубітів та HX до останнього кубіту: $\rightarrow \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$
3: Застосувати ітерацію Гровера R приблизно $\left[\frac{\pi\sqrt{2^n}}{4}\right]$ раз $\rightarrow [(2 |\psi\rangle \langle\psi| - I)O]^{\otimes R} \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right] \approx |x_0\rangle \left[\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right]$
4: Виміряти перших n кубітів: $\rightarrow x_0$

2.4 Складність алгоритму у квантовій моделі обчислень

Складність алгоритму у класичній моделі обчислень описується двома характеристиками: $T(n)$ - часова складність, $S(n)$ - просторова складність, де n - це довжина входу. У квантовій моделі обчислень, разом із характеристиками класичного алгоритму, також аналізують вентильну складність. За вимогами NIST, вентильная складність повинна обов'язково включатись у комплексний аналіз атаки на певний алгоритм у квантовій моделі обчислень.

Вентильна складність описується також двома характеристиками - це кількість використаних вентилів для побудови схеми та глибина самої схеми. Ці характеристики використовують таке поняття, як *вентильний базис*, що описує вентилі, які можуть бути використані при побудові схеми. Найбільш використовуваним, на даний час, є базис, що складається з *вентилів групи Кліффорда* та *T-вентиля*.

Означення 2.6. Група Кліффорда. Група Кліффорда - це група, що складається іх трьох логічних вентилів: $X, CNOT$ та S.

Матричне представлення вентилів групи Кліффорда:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Якщо розглядати тільки вентилі групи Кліффорда для побудови схем, аналізу квантових версій алгоритмів, то це не дуже ефективний спосіб для оцінки, так як за теоремою Готтесмана-Найла неможливо досягти квантової переваги на даних вентилях. Тобто, група Кліффорда

не є функціонально повною.

Теорема 2.1. ***Готтесмана – Найла [19]****. Схема у квантовій моделі, що використовує вентилі групи Кліффорда, може бути ефективно реалізована у класичній моделі обчислень.*

В основі швидкодії алгоритмів у квантовій моделі лежить квантова заплутаність станів, що досягаються використанням вентилів групи Кліффорда. Ця теорема пояснює, що при використанні тільки вентилв групи Кліффорда ми не зможемо досягнути певної переваги над алгоритмами у класичній моделі.

Для нівелювання цієї особливості вводять ще один вентиль - *T-вентиль*. Таке доповнення робить базис функціонально повним.

Означення 2.7. T-вентиль - це квантовий вентиль, матричне представлення якого описується наступним чином: $\begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix}$

Твердження 2.1. *Група вентилів Кліффорда разом із T-вентилем є функціонально повною.*

Тобто, тепер є заданий базис для проведення оцінок алгоритмів у квантовій моделі. Враховуючи, що він описується вентилями, які відносяться до двох типів, то й оцінка розглядається комплексно: складність за кількістю вентилів групи Кліффорда та за кількістю T-вентилів. Також є характеристика, як *загальна глибина квантового кола* та *T-глибина*.

2.5 Застосування алгоритму Гровера до SM3

Розглянемо квантовий метод криптоаналізу із боку використання алгоритму квантового пошуку до геш-функції SM3, яка була розібрана у Розділі 1. Це дозволить краще зрозуміти із чого потрібно починати при

реалізації атаки на геш-функцію у квантовій моделі за допомогою обраного метода [12].

Квантова реалізація SM3. Квантова реалізація SM3 передбачає оцінку квантових ресурсів у випдаку використання вхідного повідомлення яке доповнюється до довжини у 512 біт. Основна ідея полягає в оновленні кубітів повідомлення за допомогою змішування доповнених повідомлень із функцією розширення та функцією стиснення. Два слова повідомлення W_j, $W_j^{'}$ $(j = 0, 1, 2, \ldots, 63)$ включаються один раз для оновлення регістру із функцією стиснення. По-перше, метод оновлює $W_j^{'}$ $(j = 0, 1, 2, \ldots, 63)$ до W_j $(j = 0, 1, 2, \ldots, 63)$ та зберігає кубіти через переробку. По-друге, було показано як оновлювати та використовувати наявні кубіти в розподілі замість окремого розподілу кубітів для тимчасових змінних (SS_1, SS_2, TT_1) та TT_2, що використовується класичною версією SM3.

Падінг повідомлення. Для того, щоб зберегти повідомлення B довжиною 512 біт у $W_0, W_1, \ldots, W_{15}$ та обновити $W_{16}, W_{17}, \ldots, W_{67}$ використовувалися операції перестановки та вентилі CNOT. Оновлені значення $W_0, W_1, \ldots, W_{67}$ використовується для першої функції стиснення, а після цього перероблюються для оновлення $W_0^{'}, W_1^{'}, \ldots, W_{63}^{'}$ для другої функції стиснення без використання додаткових кубітів. Тому, відбувається наступне розділення: функція падінгу та функція стиснення поділяються на першу функцію падінгу та другу функцію падінгу, та першу і другу функцію стиснення відповідно, і використовуються разом.

Наступний алгоритм реалізує квантову схему для падінгу, що оновлює $W_{16}, W_{17}, \ldots, W_{67}$:

У першому алгоритмі падінгу повідомлення, $W_j (16 \leqslant j \leqslant 67)$ генерується за допомогою W_{j-16}, W_{j-9}, W_{j-3}, W_{j-13}, W_{j-6} $(16 \leqslant j \leqslant 67)$. Так як кубіти не можуть виконувати прості операції розподілу, то значення роботи вентиля CNOT у рядках 4 і 5 зберігається у W_{j-16}. Враховуючи, що W_j генерується, то попереднє значення повідомлення не потрібно змінювати. Оновлене значення результату зберігається у W_j, а значення W_{j-16}, що змінилося в процесі оновлення, повертається

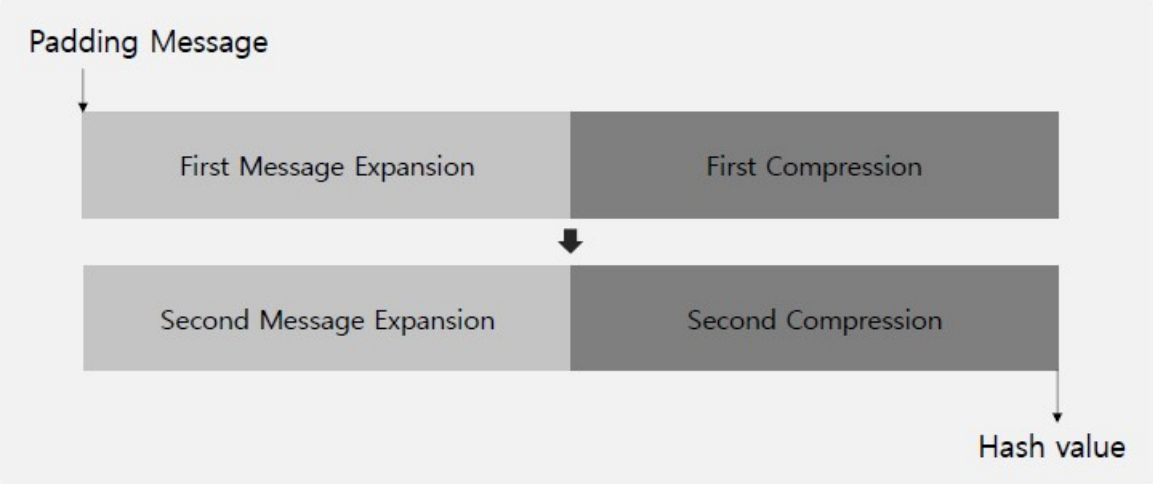


Рисунок 2.7 – Розділення функції падінгу та стиснення

Input: $W_0, W_1, ..., W_{15}$.
Output: $W_{16}, W_{17}, ..., W_{67}$.
1: **Update:**
2: **for** $i = 0$ **to** 31 **do**
3: $W_{j-16}[i] \leftarrow \text{CNOT}(W_{j-9}[i], W_{j-16}[i]),\ j = 16, ..., 67$
4: $W_{j-16}[i] \leftarrow \text{CNOT}(W_{j-3}[(i+15)\%32], W_{j-16}[i]),\ j = 16, ..., 67$
5: **end for**
6: $Permutation_{p1}(W_{j-16})$
7: **for** $i = 0$ **to** 31 **do**
8: $W_j[i] \leftarrow \text{CNOT}(W_{j-16}[i], W_j[i]),\ j = 16, ..., 67$
9: $W_j[i] \leftarrow \text{CNOT}(W_{j-13}[(i+15)\%32], W_j[i]),\ j = 16, ..., 67$
10: $W_j[i] \leftarrow \text{CNOT}(W_{j-6}[(i+15)\%32], W_j[i]),\ j = 16, ..., 67$
11: **end for**
12: **Update**(reverse)
13: **return** $W_{16}, W_{17}, ..., W_{67}$

Рисунок 2.8 – Квантова схема падінгу для оновлення $W_{16}, W_{17}, \ldots, W_{67}$

оберненим. Функція $Permutation_{P_1}$ у 6 рядку описує перестановку класичної версії SM3. Після першої функції падінгу, продовжується перша функція стиснення. Вже після цього виконується друга функція падінгу та друга функція стиснення.

У другій функції падінгу, застосовується CNOT-вентиль для $W_0, W_1, \ldots, W_{67}$, який використовувався у першій функції стиснення, та виводиться нове повідомлення $W_0', W_1', \ldots, W_{63}'$. Таким чином, кубіт був використаний повторно. Повідомлення $W_0', W_1', \ldots, W_{63}'$ згенероване

Input: $W_k, W_{k+4}, \quad k = 0, ..., 63.$
Output: $W'_t, \quad t = 0, ..., 63.$
1: **for** $i = 0$ **to** 31 **do**
2: $\quad W'j_[i] \leftarrow \text{CNOT}(Wj_[i], W_{j+4}[i]), \quad j = 0, ..., 63$
3: **end for**
4: **return** $W'_t, \quad t = 0, ..., 63$

Рисунок 2.9 – Квантова схема для другої функції падінгу

другою функцією падінгу, використовується другою функцією стиснення.

Отримання геш-значення. Після використання першої функції падінгу, перша функція стиснення, друга функція падінгу та друга функція стиснення ітеративно застосовуються 64 рази по порядку. Після завершення ітерацій, оновлені регістри A, B, C, D, E, F, G, H складаються за модулем 2 із попередніми регістрами A, B, C, D, E, F, G, H.

Перестановки. У геш-функції SM3 використовуються дві перестановки:

$$P_0(X) = X \oplus (X \lll 9) \oplus (X \lll 17)$$
$$P_1(X) = X \oplus (X \lll 15) \oplus (X \lll 23)$$

Дані перестановки використовують CNOT-вентиль. Якщо значення операції збережено, тоді важко знайти вихідне значення кубіту, без проблем у наступних операціях. У звичайних випадках, слід зберігати вихідні значення кубіт та повторно використовувати вже задіяні кубіти.

Коли $A = a_{31}, a_{30}, \ldots, a_0$ задано, а a_0 - найбільш значущий біт, то вентиль CNOT використовується у порядку $a_{31}, \ldots, a_{17}$. Знайти a_{31} важко при обчисленнях тільки a_{16}. Отже, процедура пошуку цього значення за допомогою багаторазового використання вентилю CNOT.

Цей алгоритм обчислює a_{16} як частину перестановки P_0. У цей момент, вентиль CNOT неодноразово застосовувався для використання a_{31}. Зміна кожного стану представлена на наступній таблиці:

Таблиця 2.1 - Зміна станів у алгоритмі на Рисунку 2.10

Input: a_{16}.
Output: $a_{16} \leftarrow a_{16} \oplus a_7 \oplus a_{31}$.
1: $a_{16} \leftarrow \text{CNOT}(a_7, a_{16})$
2: $a_{16} \leftarrow \text{CNOT}(a_{31}, a_{16})$
3: $a_{16} \leftarrow \text{CNOT}(a_{22}, a_{16})$
4: $a_{16} \leftarrow \text{CNOT}(a_{14}, a_{16})$
5: $a_{16} \leftarrow \text{CNOT}(a_{13}, a_{16})$
6: $a_{16} \leftarrow \text{CNOT}(a_5, a_{16})$
7: **return** $a_{16} \leftarrow a_{16} \oplus a_7 \oplus a_{31}$

Рисунок 2.10 – Частина обчислень перестановки P_0

Строка	Кубіт	Стан
1	a_{16}	$a_{16} \oplus a_7$
2	a_{16}	$a_{16} \oplus a_7 \oplus a_{31} \oplus a_{22} \oplus a_{14}$
3	a_{16}	$a_{16} \oplus a_7 \oplus a_{31} \oplus a_{14} \oplus a_{13} \oplus a_5$
4	a_{16}	$a_{16} \oplus a_7 \oplus a_{31} \oplus a_{13} \oplus a_5$
5	a_{16}	$a_{16} \oplus a_7 \oplus a_{31} \oplus a_5$
6	a_{16}	$a_{16} \oplus a_7 \oplus a_{31}$

Оцінка. Для оцінки використовувавя емулятор IBM ProjectQ. За допогою програмної реалізації було оцінено кількість використаних кубітів, вентилів Тоффолі, CNOT та T-вентилів.

За результатими тестів було отримано наступні оцінки:

Таблиця 2.2 - Зміна станів у алгоритмі на Рисунку 2.10

Алгоритм	Кубіти	Вентилі Тоффолі	CNOT вентилі	X-вентилі
SHA2 [20]	2402	57184	534272	—
SHA3 [20]	3200	84480	33269760	85
SM3	2721	43248	134144	2638

Висновки до розділу 2

Цей розділ включає опис квантової теорії, яка необхідна для розуміння методів квантового криптоаналізу. До цих методів входить застосування алгоритму квантового пошук (алгоритм Гровера). Також досліджено тонкощі побудови криптопримітиву у квантовій моделі - який базис вентилів необхідно обирати при конструюванні схеми найбільш ефективним способом, визначення оптимальної кількості кубітів та ін. Загалом, з'ясувалося, що найбільш популярним є застосування базису, який складається із вентелів групи Кліффорда та T-вентилів.

В якості прикладу успішного застосування алгоритму Гровера, досліджено атаку на криптографічну геш-функцію SM3. Спосіб, який реалізований у роботі[], показав як саме потрібно підходити до квантового криптоаналізу та як вирішувати труднощі перенесення компонент криптопримітиву у квантову модель.

3 КВАНТОВІ АТАКИ НА ГЕШ-ФУНКЦІЮ «КУПИНА»

На даний момент найбільш результативним методом побудови квантової атаки на певний криптопримітив є застосування деякого квантового алгоритму перебору. Стандартизація NIST тепер вимагає не тільки підрахунок кількості викликів шифруючого перетворення в якості аналізу атаки, а й обчислення складності реалізації самого перетворення, що значно покращує точність оцінки.

У цьому розділі буде розглянуто як раз складність реалізації перетворень, що застосовуються у ітераційному алгоритмі «Купини», а також буде проведений аналіз щодо імплементації даного криптопримітиву у квантову модель.

3.1 Квантова реалізаці геш-функції «Купина»

Алгоритм отримання геш-значення даного криптопримітиву використовує функцію стиснення, яка описується наступним перетворенням:

$$CF_l(CV_i, M_i) = T_l^{\oplus}(CV_i \oplus M_i) \oplus T_l^{+}(M_i) \oplus M_i.$$

Формула включає застосування двух перестановок - $T_l^{\oplus}$, T_l^{+}. Згідно опису геш-функції, потрібно розглянути два випадки:

$$l = 512 : CF_{512}(CV_i, M_i) = T_{512}^{\oplus}(CV_i \oplus M_i) \oplus T_{512}^{+}(M_i) \oplus M_i,$$
$$l = 1024 : CF_{1024}(CV_i, M_i) = T_{1024}^{\oplus}(CV_i \oplus M_i) \oplus T_{1024}^{+}(M_i) \oplus M_i.$$

На даному етапі оцінимо спрощений варіант реалізації перестановок: вважаємо, що складність у квантовій моделі буде еквівалентна складності побудови вже заданого вектору. Тобто, буде розглянуто випадки, коли на вікд перестановкам $T_l^{\oplus}$ та T_l^{+} подається матриці, що описані в стандарті.

Оцінка $T_l^{\oplus}$ та T_l^{+} у спрощеному варіанті. Для початку

побудуємо алгебраїчні нормальні форми для кожного вихідного вектору. Це реалізовано за допомогою швидкого перетворення Мебіуса:

```
def moebius_transform(table: list) -> list:
    ANF_sbox = table.copy()
    for i in range(6): # 8
        for j in range(64): # 256
            if (j >> i) & 1:
                ANF_sbox[j] ^= ANF_sbox[j ^ (1 << i)]

    return ANF_sbox
```

Рисунок 3.1 – Фрагмент коду реалізації швидкого перетворення Мебіуса

Для побудови перестановок було використано фреймворк на *qiskit*[21], що приймає на вхід АНФ деякої булевої функції від 6 або 8 змінних:

```
def quantum_estimate_anf(anf: list) -> QuantumCircuit:

    x = QuantumRegister(6, 'x')
    out = QuantumRegister(6, 'out')
    anc_qubit = AncillaRegister(6, 'ancilla')

    circ = QuantumCircuit(x, out, anc_qubit)

    for i in range(1 << 6):
        if anf[i] != 0:
            ind = weight_of_n_significant_bit(i)
            weight = unitary_bit_indices(i)
            for ind_output in unitary_bit_indices(anf[i]):
                if weight == 0:
                    circ.x(out[ind_output])
                elif weight == 1:
                    circ.cx(x[ind[0]], out[ind_output])
                elif weight == 2:
                    circ.ccx(x[ind[0]], x[ind[1]], out[ind_output])
                else:
                    circ.mcx(x[ind], out[ind_output], anc_qubit, 'v-chain')

    kupyna_one_iteration_estimate(circ)

    return circ
```

Рисунок 3.2 – Фрагмент коду реалізації квантової схеми за АНФ

Побудовані квантові схеми мають наступні характеристики:

Таблиця 3.1 - Вентильна складність перестановок $T_l^{\oplus}$, T_l^{+}

Перестановка	Вентилі групи Кліффорда	Т-вентилі	Глибина схеми
$T^{\oplus}_{512}$	3512	2902	5099
T^{+}_{512}	3191	2631	4577
$T^{\oplus}_{1024}$	29134	23886	42764
T^{+}_{1024}	28880	23676	42379

Як бачимо, в залежності від версії геш-функції, складність перестановок має однаковий порядок. У середньому для $l = 512$ виходить $2^{11.7}$ вентилів Кліффорда, $2^{11.4}$ Т-вентилів та глибина схеми становить $2^{12.2}$. Аналогічно для $l = 1024$: $2^{14.8}$ вентилів Кліффорда, $2^{14.5}$ Т-вентилів та середня глибина схеми $2^{15.4}$.

Реалізація та оцінка складності процедури ініціалізації. Для отримання оцінки переведомо бітові вектори $1 \ll 512$ та $1 \ll 1024$ у байтовий вигляд та подамо їх на вхід функції, що будує квантову схему за АНФ матриці. Отримані результати приведені у наступній таблиці:

Таблиця 3.2 - Складність процедури ініціалізації

l	Вентилі Кліффорда	Т-вентилі	Глубина схеми
512	1183	975	1665
1024	7205	5905	10467

Реалізація та оцінка складності додавання за модулем 2^{64}. Для оцінки цієї операції спочатку потрібно реалізувати *повний суматор* на фреймворці *qiskit* та проаналізувати отримані характеристики квантової схеми.

Твердження 3.1. *Повний суматор переводить квантовий регістр зі стану* $|a\rangle\,|b\rangle\,|CarryBit\rangle\,|0\rangle$ *у стан* $|a\rangle\,|(a+b+CarryBit) mod 2\rangle$.

Доведення. Для доведення цього факту розглянемо покроково кожен стан квантового регістру.

$$reg_0 = ccx(q[0], q[1], q[3]) \Rightarrow q[3] = a \cdot b,$$
$$reg_1 = cx(q[0], q[1]) \Rightarrow q[1] = a \oplus b,$$

$$reg_2 = ccx(q[1], q[2], q[3]) \Rightarrow q[3] = (a \cdot b) \oplus CarryBit \cdot (a \oplus b),$$
$$reg_3 = cx(q[1], q[2]) \Rightarrow q[2] = a \oplus b \oplus CarryBit,$$
$$reg_4 = cx(q[0], q[1]) \Rightarrow q[1] = b,$$

Переберемо всі можливі результати в залежності від значення пари (a, b):
$a = b = 1 \Rightarrow \forall CarryBit_{i-1} : a \oplus b = 0, CarryBit_i = 1$
$a = b = 0 \Rightarrow CarryBit_i = 1$
$a \oplus b = 1 \Rightarrow CarryBit_i = 1 \Leftrightarrow CarryBit_{i-1} = 1,$
де i - це номер поточної ітерації.

□

Використовуючи вигляд схеми для квантової версії повного сумматора [22], створимо функцію за допомогою *qiskit*:

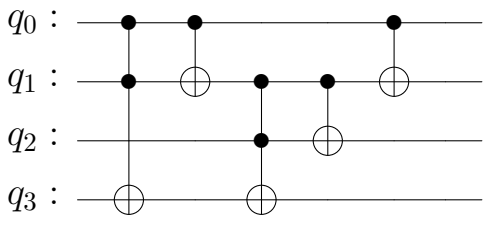
```
def full_adder() -> QuantumCircuit:
    q = QuantumRegister(4, 'q')
    circ = QuantumCircuit(q)

    circ.ccx(q[0], q[1], q[3])
    circ.cx(q[0], q[1])
    circ.ccx(q[1], q[2], q[3])
    circ.cx(q[1], q[2])
    circ.cx(q[0], q[1])

    return circ
```

Рисунок 3.3 – Фрагмент коду реалізації квантової схеми повного сумматора

У результаті отримуємо схему для повного сумматора:



Твердження 3.2. *Квантова схема для бітового повного сумматора використовує 19 вентелей групи Кліффорда та 14 T-вентиль. Глубина схеми складає 23 базових вентиля.*

Після побудови базового повного суматора, наступним кроком буде

його послідовне застосування необхідне число раз для побудови схеми, що реалізує сумування за модулем 2^n. Реалізація такої схеми представлена на наступному рисунку:

```
def modulo_adder(length: int) -> QuantumCircuit:
    start_adder = full_adder().to_instruction()

    a = QuantumRegister(length, name = 'a')
    b = QuantumRegister(length, name = 'b')
    output = QuantumRegister(length, name = 'output')
    carry_bit = QuantumRegister(1, name = 'carry bit')

    res = QuantumCircuit(a, b, output, carry_bit)

    for i in range(0, length):
        res.append(start_adder, [a[i], b[i], carry_bit[0], output[i]])

    return res
```

Рисунок 3.4 – Фрагмент коду реалізації схеми для сумування за модулем 2^n

Твердження 3.3. *Квантова схема, що реалізує схему додавання за модулем* 2^{64} *потребує 1216 вентелей групи Кліффорда, 896 T-вентилей. Глубина схеми складає 716 базових вентилей.*

Оцінка складності реалізації підстановок $\pi_0, \pi_1, \pi_2, \pi_3$. Дані оцінки вже були реалізовані, тому тут буде приведено тільки результати:

Таблиця 3.3 - Складність реалізація підстановок $\pi_0, \pi_1, \pi_2, \pi_3$

Підстановка	Вентилі групи Кліффорда	T-вентилі	Глубина схеми
π_0	28027	22966	41122
π_1	29440	24118	43240
π_2	29556	24227	43365
π_3	27851	22830	40825

Реалізація та оцінка складності множення у полі $\mathbf{GF}(2^m)$. Для того, щоб оцінити дану операцію, скористаємося такою теоремою про множення елементів у полі $\mathbf{GF}(2^m)$ та знайдемо матрицю редукції Q для реалізації схеми множення:

Теорема 3.1. ***Множення у полі*** $\mathbf{GF}(2^8)$.*Результат множення*

двох елементів a і b із поля $\mathbf{GF}(2^m)$ *може бути отриманий за допомогою наступного співвідношення:*

$r = a \cdot b = d + Q^T \cdot e,$

$$d = (d_0, d_1, \ldots, d_{n-1})^T = L \cdot b = \begin{pmatrix} a_0 & 0 & \ldots & 0 \\ a_1 & a_0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-1} & \ldots & a_0 \end{pmatrix} \cdot (b_0, b_1, \ldots, b_{n-1})^T,$$

$$e = (e_0, e_1, \ldots, e_{n-2})^T = U \cdot b = \begin{pmatrix} 0 & a_{n-1} & \ldots & a_2 & a_1 \\ 0 & 0 & \ldots & a_3 & a_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & a_{n-1} \end{pmatrix} \cdot (b_0, b_1, \ldots, b_{n-1})^T.$$

Матриця Q може бути розрахована за наступною теоремою:

Теорема 3.2. ***Обчислення матриці редукції*** Q. *Нехай поле* $\mathbf{GF}(2^m)$ *породжене деяким примітивним поліномом* $p(x)$, *тоді матриця редукції* Q *розміром* $(m-1) \times m$ *може бути обчислена за наступною формулою:*

$$\overrightarrow{\beta} = Q \cdot \overrightarrow{\alpha}(mod(p(x))),$$

де β *задається як степені коренів* α *від* m *до* $2(m-1)$ $\beta = (\alpha^m, \alpha^{m+1}, \ldots, \alpha^{2(m-1)})^T$.

Тепер, маючи дану теорему, можна знайти матрицю редукції полінома, що використовується у стандарті: $p(x) = x^8 + x^4 + x^3 + x^2 + 1$. Згідно вигляду поліномів, які задають степені кореня α, отримуємо наступну матрицю редукції Q:

$\alpha^8 : \alpha^8 + \alpha^4 + \alpha^3 + \alpha^2 + 1 = 0 \Rightarrow \alpha^8 = \alpha^4 + \alpha^3 + \alpha^2 + 1,$

$\alpha^9 : \alpha^9 = \alpha^8 \cdot \alpha = \alpha^5 + \alpha^4 + \alpha^3 + \alpha,$

$\alpha^{10} : \alpha^{10} = \alpha^9 \cdot \alpha = \alpha^6 + \alpha^5 + \alpha^4 + \alpha^2,$

$\alpha^{11} : \alpha^{11} = \alpha^{10} \cdot \alpha = \alpha^7 + \alpha^6 + \alpha^5 + \alpha^3,$

$\alpha^{12} : \alpha^{12} = \alpha^{11} \cdot \alpha = \alpha^8 + \alpha^7 + \alpha^6 + \alpha^4 = \alpha^4 + \alpha^3 + \alpha^2 + 1 + \alpha^7 + \alpha^6 + \alpha^4 =$
$= \alpha^7 + \alpha^6 + \alpha^3 + \alpha^2 + 1,$

$\alpha^{13} : \alpha^{13} = \alpha^{12} \cdot \alpha = \alpha^8 + \alpha^7 + \alpha^4 + \alpha^3 + \alpha = \alpha^4 + \alpha^3 + \alpha^2 + 1 + \alpha^7 + \alpha^4 + \alpha^3 + \alpha =$
$= \alpha^7 + \alpha^2 + \alpha + 1,$

$\alpha^{14} : \alpha^{14} = \alpha^{13} \cdot \alpha = \alpha^8 + \alpha^3 + \alpha^2 + \alpha = \alpha^4 + \alpha^3 + \alpha^2 + 1 + \alpha^3 + \alpha^2 + \alpha = \alpha^4 + \alpha + 1.$

Тоді, матриця редукції Q виглядає наступним чином:

$$Q = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

```python
def multiplication(circ) -> QuantumCircuit:
    a = QuantumRegister(8, 'a')
    b = QuantumRegister(8, 'b')
    output = QuantumRegister(8, 'output')

    a_m_b = QuantumCircuit(a, b, output)

    for i in range(1, 8):
        for j in range(8-i):
            a_m_b.ccx(a[i+j], b[7-j], output[i-1])

    for i in range(8):
        for j in range(i+1):
            a_m_b.ccx(a[j], b[i-j], output[i])

    return a_m_b
```

Рисунок 3.5 – Реалізація множення елементів у $\mathbf{GF}(2^8)$

Твердження 3.4. *Операція множення елементів у полі* $\mathbf{GF}(2^8)$ *потребує використання 512 вентелів групи Кліффорда та 448 T-вентелів. Загальная глубина схеми складає 207 базових вентилів.*

Тепер, коли оцінена складність операцій, що фігурують у перестановках $T_l^{\oplus}$ та T_l^{+}, можна обчислити грубу оцінку реалізації цих перестановок у квантовій моделі.

Розглянемо перший випадок: коли $l = 512$. Такі версії перестановок використовують у якості вході деяку байтову матрицю розміром 8×8. Тоді, для аналізу, потрібно зрозуміти яким чином операції, що задають перестановку, діють на елементи матриці протягом одного раунду:

– $\kappa_{\vartheta}^{(l)}$ - функція що додає деякий вектор до кожного стовпця матриці стану. За побудовою тільки перша компонента є числим відмінним від нуля, отже, складність реалізації операції буде складати **8·|Складність операції додавання за модулем|**. Позначемо як $Estimate_{\oplus}$.

– $\eta_{\vartheta}^{(l)}$ - функція додавання вектору за модулем 2^{64}. Компоненти цього вектора відмінні від нуля, тоді, складність становитиме **64·|Складність реалізації додавання за модулем** 2^{64}**|**. Позначемо як $Estimate_{2^{64}}$.

– $\pi^{'}$ - функція заміни кожного елемента поточної матриці стану на елемент підстановки $\pi^{'}$. Звідси випливає, що складність цієї функції складає **64·|Складність реалізації підстановки** $\pi^{'}$**|**. Позначемо як $Estimate_{\pi}$.

– $\tau^{(l)}$ - складність реалізації цієї функції приймається за $O(1)$, так як циклічний зсув можна реалізувати на класичному комп'ютері.

– ψ - множення деякого вектора на кожен стовпчик матриці внутрішнього стану у полі $\mathbf{GF}(2^8)$. Тоді складність операції буде становити **64·|Складність операції множення у** $\mathbf{GF}(2^8)$**|**. Позначемо як $Estimate_{\otimes}$.

Тоді, враховуючи складність кожної операції, можна отримати формулу для обчислення склданості реалізації перестановок $T_{512}^{\oplus}$ та T_{512}^{+}:

$$Estimate_{T_{512}^{\oplus}} = 10 \cdot (8 \cdot Estimate_{\oplus} + 64 \cdot Estimate_{\pi} + 64 \cdot Estimate_{\otimes})$$

$$Estimate_{T_{512}^{+}} = 10 \cdot (64 \cdot Estimate_{2^{64}} + 64 \cdot Estimate_{\pi} + 64 \cdot Estimate_{\otimes})$$

Твердження 3.5. *Реалізація перестановки* $T_{512}^{\oplus}$ *потребує* 12374170 *(*$2^{23.55}$*) вентилів групи Кліфорда та* 15323950 *(*$2^{23.86}$*) T-вентелів.*

Загальна глубина схеми складає 26915390 *(*$2^{24.68}$*) вентилів.*

Доведення. Обчислимо складність одного раунду. Кількість вентилів групи Кліффорда: $8 \cdot 19 + 64 \cdot 2^{14.2} + 64 \cdot 512 = 1237417$, кількість T-вентилів: $8 \cdot 14 + 64 \cdot 2^{14.52} + 64 \cdot 448 = 1532395$, загальна глубина схеми обирається у найгіршому випдаку: $64 \cdot 2^{15.36} = 2691539$. □

Твердження 3.6. *Реалізація перестаноки* T^{+}_{512} *потребує* $19362680(2^{24.2})$ *вентилів групи Кліффорда та* $15896270(2^{23.9})$ *T-вентилів. Загальна глубина схеми складає* 26915390 *(*$2^{24.68}$*) вентилів.*

Доведення. Обчислимо складність одного раунду. Кількість вентилів групи Кліффорда: $64 \cdot 1216 + 64 \cdot 2^{14.8} + 64 \cdot 512 = 1936268$, кількість T-вентилів: $64 \cdot 896 + 64 \cdot 2^{14.52} + 64 \cdot 448 = 1589627$, загальна глубина схеми обирається у найгіршому випдаку: $64 \cdot 2^{15.36} = 2691539$. □

По аналогічним міркуванням можна вивести формули для випдаку $l = 1024$. При такій версії геш-функції змінюється тільки розмір матриці, що подається на вхід перестановкам - тепер 8×16.

Отже, складність реалізації перестановок за такої версії становить:

$$Estimate_{T^{\oplus}_{1024}} = 14 \cdot (16 \cdot Estimate_{\oplus} + 128 \cdot Estimate_{\pi} + 128 \cdot Estimate_{\otimes})$$
$$Estimate_{T^{+}_{1024}} = 14 \cdot (128 \cdot Estimate_{2^{64}} + 128 \cdot Estimate_{\pi} + 128 \cdot Estimate_{\otimes})$$

Твердження 3.7. *Реалізація перестановки* $T^{\oplus}_{1024}$ *потребує* $52040702(2^{25.62})$ *вентилів групи Кліффорда та* $42907060(2^{25.34})$ *T-вентилів. Загальна глубина схеми складає* $75363092(2^{26.16})$ *вентилів.*

Доведення. Обчислимо складність одного раунду. Кількість вентилів групи Кліффорда: $16 \cdot 19 + 128 \cdot 2^{14.8} + 128 \cdot 512 = 3717193$, кількість T-вентилів: $16 \cdot 14 + 128 \cdot 2^{14.52} + 128 \cdot 448 = 3064790$, загальна глубина схеми обирається у найгіршому випдаку: $128 \cdot 2^{15.36} = 5383078$. □

Твердження 3.8. *Реалізація перестановки* T^{+}_{1024} *потребує* $54215518(2^{25.68})$ *вентилів групи Кліффорда та* $44509556(2^{25.4})$ *T-вентилів. Загальна глубина схеми складає* $75363092(2^{26.16})$.

Доведення. Обчислимо складність одного раунду. Кількість вентилів групи Кліффорда: $128 \cdot 1216 + 128 \cdot 2^{14.8} + 128 \cdot 512 = 3872537$, кількість T-вентилів: $128 \cdot 896 + 128 \cdot 2^{14.52} + 128 \cdot 448 = 3179254$, загальна глубина схеми обирається у найгіршому випадку: $128 \cdot 2^{15.36} = 5383078$. □

Маючі ці результати, можемо вивести формулу для обчислення загальної складності реалізації геш-функції «Купина» у квантовій моделі обчислень. Ця оцінка складається із таких компонент:

– Складність процедури ініціалізаціїї: $CV_0 \leftarrow IV$. Позначемо як $Estimate_{init}$.

– Складність реалізації функції стиснення: $CF(CV_i, M_i)$, $i = \overline{0,k-1}$. Позначемо як $Estimate_{CF}$.

– Остання операція $T_l^{\oplus}(CV_k) \oplus CV_k$ без врахування операції $Trunc$, що обрізає повідомлення. Позначемо як $Estimate_{end}$.

Тоді, загальна складність за кількістю вентилів становить:

$$Estimate_l = Estimate_{init} + k \cdot Estimate_{CF} + Estimate_{end} \tag{3.1}$$

$$Estimate_l = Estimate_{init} + k \cdot Estimate_{CF} + Estimate_{end} =$$
$$= Estimate_{init} + k \cdot (Estimate_{T_l^+} + Estimate_{T_l^{\oplus}} + 2 \cdot SizeM \cdot Estimate_{\oplus}) +$$
$$+ Estimate_{T_l^{\oplus}} + SizeM \cdot Estimate_{\oplus} = Estimate_{init} + k \cdot Estimate_{T_l^+} +$$
$$+ (k+1) \cdot Estimate_{T_l^{\oplus}} + (2k+1) \cdot SizeM \cdot Estimate_{\oplus}$$

Аналогічно обчислюється загальна глибина схеми як сума глибин окремих компонент:

$$Depth_l = Depth_{init} + k \cdot Depth_{CF} + Depth_{end} \tag{3.2}$$

$Depth_l = Depth_{init} + k \cdot Depth_{T_l^+} + (k+1) \cdot Depth_{T_l^{\oplus}} + (2k+1) \cdot SizeM \cdot Depth_{\oplus}$, де $SizeM$ -це розмір байтового вхідного повідомлення.

Маючі ці формули можна обчислити кінцеві характеристики для кожного варіанту геш-функції «Купина»:

1) $l = 512$:

– Вентилі групи Кліффорда:

$Estimate_{512}^{Clifford} = 2^{\log_2(1183)} + k \cdot 10 \cdot 2^{20.88} + (k+1) \cdot 10 \cdot 2^{20.23} + (2k+1) \cdot 64 \cdot 19 \approx$
$\approx 2^{10.2} + k \cdot 2^{24.2} + (k+1) \cdot 2^{23.6} + (2k+1) \cdot 2^{10.24} \leqslant 2^{10.2} + 2^{87} \cdot 2^{24.2} +$
$+ 2^{87} \cdot 2^{23.6} + 2^{88} \cdot 2^{10.24} \approx 2^{111.9}$.

– T-вентилі:

$Estimate_{512}^{T-gate} = 2^{\log_2(975)} + k \cdot 10 \cdot 2^{20.6} + (k+1) \cdot 10 \cdot 2^{20.54} + (2k+1) \cdot 64 \cdot 14 \approx$
$\approx 2^{9.9} + k \cdot 2^{23.9} + (k+1) \cdot 2^{23.8} + (2k+1) \cdot 2^{9.8} \leqslant 2^{9.9} + 2^{87} \cdot 2^{23.9} +$
$+ 2^{87} \cdot 2^{23.8} + 2^{88} \cdot 2^{9.8} \approx 2^{111.85}$.

– Глубина схеми:

$Depth_5 12 = 2^{\log_2(1665)} + k \cdot 10 \cdot 2^{21.36} + (k+1) \cdot 10 \cdot 2^{21.36} + (2k+1) \cdot 64 \cdot 23 \approx$
$\approx 2^{10.7} + (2k+1) \cdot 2^{24.7} \leqslant 2^{10.7} + 2^{88} \cdot 2^{24.7} \approx 2^{112.7}$.

2) $l = 1024$:

– Вентилі групи Кліффорда:

$Estimate_{1024}^{Clifford} = 2^{\log_2(7205)} + k \cdot 14 \cdot 2^{21.88} + (k+1) \cdot 14 \cdot 2^{21.82} + (2k+1) \cdot 128 \cdot 19 \approx$
$\approx 2^{12.8} + k \cdot 2^{25.7} + (k+1) \cdot 2^{25.6} + (2k+1) \cdot 2^{11.2} \leqslant 2^{12.8} + 2^{86} \cdot 2^{25.7} +$
$+ 2^{86} \cdot 2^{25.6} + 2^{87} \cdot 2^{11.2} \approx 2^{112.65}$.

– T-вентилі:

$Estimate_{1024}^{T-gate} = 2^{\log_2(5905)} + k \cdot 14 \cdot 2^{21.6} + (k+1) \cdot 14 \cdot 2^{21.54} + (2k+1) \cdot 128 \cdot 14 \approx$
$\approx 2^{12.5} + k \cdot 2^{25.4} + (k+1) \cdot 2^{25.3} + (2k+1) \cdot 2^{10.8} \leqslant 2^{12.5} + 2^{86} \cdot 2^{25.4} +$
$+ 2^{86} \cdot 2^{25.3} + 2^{87} \cdot 2^{10.8} \approx 2^{112.35}$.

– Глубина схеми:

$Depth_{1024} = 2^{\log_2(10467)} + k \cdot 14 \cdot 2^{22.36} + (k+1) \cdot 14 \cdot 2^{22.36} + (2k+1) \cdot 128 \cdot 23 \approx$
$\approx 2^{13.3} + (2k+1) \cdot 2^{26.2} \leqslant 2^{13.3} + 2^{87} \cdot 2^{26.2} \approx 2^{113.2}$.

3.2 Застосування алгоритму Гровера до перестановок геш-функції «Купина»

Наразі, для оцінки деякого алгоритму у квантовій моделі, використовується метрика NIST. Вона включає в себе дві характеристики: максимальна глибина квантової схеми ($MaxDepth$) та число використаних вентилів у схемі ($GateCount$). У якості еталону із яким порівнюють став шифр AES[23]. Після проведення багатьох тестів прийшли до таких порівняльних констант:

$$MaxDepth = 2^{40}$$
$$GateCount = 2^{130}.$$

Ці значення відповідають основним параметрам атаки на версію шифру AES-128. Тобто, якщо знаходиться така атака, яка витрачає менше ресурсів, то такий злам вважається успішним.

Для загального випадку оцінки стійкості деякого криптопримітиву використовується наступна формула:

$$GateCount = \frac{SecurityLevelConstant}{MaxDepth}, \tag{3.3}$$

де $SecurityLevelConstant$ - це константа, що визначається в залежності від обраного рівня стійкості. В залежності від довжини ключа AES, використовують такі значення цієї константи:

Таблиця 3.4 - *SecurityLevelConstant* **для різних довжин ключів**

Довжина ключа AES	*SecurityLevelConstant*
128	2^{170}
192	2^{233}
256	2^{298}

Інакше кажучи, криптопримітив вважається вразливим, якщо виконується наступне співвідношення:

$$GateCount \cdot MaxDepth \leqslant SecurityLevelConstant, \tag{3.4}$$

для відповідного рівня стійкості. Але, також, накладаються певні обмеження на параметр $MaxDepth$: значення не повинно бути більше ніж 2^{96}. Якщо говорит про оптимальні атаки, то це значення знаходиться у межах 2^{40}.

Введемо процедуру пошуку прообразу за алгоритмом Гровера за допомогою оракулу для функції $f : \{0,1\}^k \to \{0,1\}^k$. У загальному випадку схема виглядає наступним чином:

Алгоритм робить $\lfloor \frac{\pi}{4} 2^{\frac{k}{2}} \rfloor$ викликів до ітерації Гровера G. За побудовою ця ітерація розбивається на два етапи: U_g - реалізує відображення $g : \{0,1\}^k \to \{0,1\}$, що переводить x в 1 тоді і тільки тоді, коли $f(x) = y$; кожен виклик U_g реалізує два виклики до оберненої реалізації f та один виклик схеми порівняння для перевірки рівності $f(x) = y$. Другою частиною ітерації Гровера є реалізація перетворення $2 |0\rangle \langle 0| - I$, що має назву оператору дифузії.

Для реалізації пошуку введемо наступні кроки:

1) Визначити послідовність всіх прообразів $\frac{1}{\sqrt{|Preimages|}} \sum\limits_{x \in Preimages} |x\rangle$

2) Виконати $O(\sqrt{|Preimages|})$ викликів оракулу Гровера

3) Підготувати регістр, який можна виміряти і отримати результат у

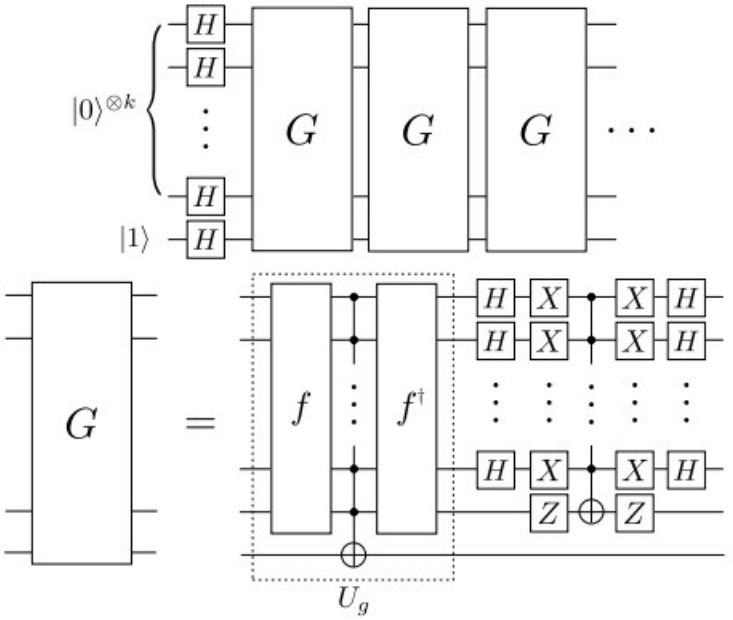


Рисунок 3.6 – Алгоритм пошуку Гровера із оракулом для функції f

вигляді шуканого прообразу.

Використовуючи оцінки, що були отримані для перестановок $T_l^{\oplus}$ та T_l^{+} і нерівності (3.2), можемо оцінити застосовність алгоритму Гровера до цих перестановок:

$$GateCount_{T_{512}^{\oplus}} = 10 \cdot (2^{20.23} + 2^{20.54}) \approx 2^{24.7},$$
$$GateCount_{T_{512}^{+}} = 10 \cdot (2^{20.88} + 2^{20.6}) \approx 2^{25.06},$$
$$MaxDepth_{T_{512}^{\oplus}} = MaxDepth_{T_{512}^{+}} = 10 \cdot 2^{21.36} \approx 2^{24.7},$$

Отримуємо наступні оцінки за допомогою нерівності (3.2) для випадку $l = 512$:

$$GateCount_{T_{512}^{\oplus}} \cdot MaxDepth_{T_{512}^{\oplus}} = 2^{24.7} \cdot 2^{21.36} = 2^{46.06},$$
$$GateCount_{T_{512}^{+}} \cdot MaxDepth_{T_{512}^{+}} = 2^{25.06} \cdot 2^{24.7} = 2^{49.76}$$

Так як оракул Гровера у цьому випадку потрібно застосувати $O(\sqrt{2^{512}})$ разів, то отримуємо остаточні оцінки для цих добутків:

$$T_{512}^{\oplus} : 2^{46.06} \cdot 2^{256} = 2^{302.06} \Rightarrow O(2^{302.06}),$$
$$T_{512}^{+} : 2^{49.76} \cdot 2^{256} = 2^{305.76} \Rightarrow O(2^{305.76}).$$

Отримали, що атака за допомогою пошуку Гровера застосовна до $T_{512}^{\oplus}$ та T_{512}^{+}. За аналогічною схемою обчислимо оцінку для перестановок $T_{1024}^{\oplus}$ і

T^{+}_{1024}:

$$GateCount_{T^{\oplus}_{1024}} = 14 \cdot (2^{21.82} + 2^{21.54}) \approx 2^{26.5},$$
$$GateCount_{T^{+}_{1024}} = 14 \cdot (2^{21.88} + 2^{21.6}) \approx 2^{26.6},$$
$$MaxDepth_{T^{\oplus}_{1024}} = MaxDepth_{T^{+}_{1024}} = 14 \cdot 2^{22.36} \approx 2^{26.16},$$

Отримуємо наступні оцінки за допомогою нерівності (3.2) для випадку $l = 1024$:

$$GateCount_{T^{\oplus}_{1024}} \cdot MaxDepth_{T^{\oplus}_{1024}} = 2^{26.5} \cdot 2^{26.16} = 2^{52.66},$$
$$GateCount_{T^{+}_{1024}} \cdot MaxDepth_{T^{+}_{1024}} = 2^{26.6} \cdot 2^{26.16} = 2^{52.76}$$

Так як оракул Гровера у цьому випадку потрібно застосувати $O(\sqrt{2^{1024}})$ разів, то отримуємо остаточні оцінки для цих добутків:

$$T^{\oplus}_{1024} : 2^{52.66} \cdot 2^{512} = 2^{564.66} \Rightarrow O(2^{564.66}),$$
$$T^{+}_{1024} : 2^{52.76} \cdot 2^{512} = 2^{564.76} \Rightarrow O(2^{564.76}).$$

3.3 Атака на прообраз за допомогою алгоритма Гровера

З вигляду схеми Меркля-Дамгора, яка є основною структурою для побудови геш-функції «Купина», достатньо обчислити необхідні квантові ресурси по застосовності алгортму Гровера для одної ітерації. Одна ітерація - це застосування функції стиснення:

$$CF(CV, M) = T^{\oplus}_{l}(CV \oplus M) \oplus T^{+}_{l}(M) \oplus M.$$

Компонентами складності реалізації є реалізації перестановок T^{+}_{l}, $T^{\oplus}_{l}$ та складання за модулем:

$$GateCount^{1round}_{l} = GateCount_{T^{+}_{l}} + 2 \cdot SizeM \cdot GateCount_{\oplus} + GateCount_{T^{\oplus}_{l}} \tag{3.5}$$

$$MaxDepth^{1round}_{l} = MaxDepth_{T^{+}_{l}} + 2 \cdot SizeM \cdot MaxDepth\oplus + MaxDepth_{T^{\oplus}_{l}} \tag{3.6}$$

1) $l = 512$:

– GateCount:

$GateCount_{512}^{1round} \approx 2^{25.06} + 2 \cdot 64 \cdot 33 + 2^{24.7} \approx 2^{25.9}$

– MaxDepth:

$MaxDepth_{512}^{1round} \approx 2 \cdot 2^{24.7} + 2 \cdot 64 \cdot 23 \approx 2^{25.7}$

Так як оракул Гровера потрібно використати $O(\sqrt{2^{512}})$ раз, отримуємо кінцеве значення для оцінки по застосовності алгоритму:

$$l = 512 : 2^{25.9} \cdot 2^{25.7} = 2^{51.6} \Rightarrow O(2^{256} \cdot 2^{51.6}) = O(2^{307.6})$$

Степінь цього значення відрізняється від порогових значень метрики NIST лише у десятках, тому атака є застосовною до цієї версії «Купини».

2) $l = 1024$:

– GateCount:

$GateCount_{1024}^{1round} \approx 2^{26.6} + 2 \cdot 128 \cdot 33 + 2^{26.5} \approx 2^{27.5}$

– MaxDepth:

$MaxDepth_{1024}^{1round} \approx 2 \cdot 2^{26.16} + 2 \cdot 128 \cdot 23 \approx 2^{27.1}$

У цьому випадку оракул Гровера застосовується $O(\sqrt{2^{1024}})$, тому кінцеве значення оцінки застосовності алгоритму наступне:

$$l = 1024 : 2^{27.5} \cdot 2^{27.1} = 2^{54.6} \Rightarrow O(2^{512} \cdot 2^{54.6}) = O(2^{566.6})$$

У даному випадку, алгоритм, можливо є незастосовним, враховуючи метрики NIST.

Висновки до розділу 3

Отже, був проведений аналіз складності побудови перестановок $T_l^{\oplus}$ та T_l^{+}. Для отримання кінцевих результатів використовувалося допоміжне програмне забезпечення - фреймфорк *Qiskit*. За допомогою нього було оцінено функції, які фігурують у перестановках, а саме: сумування за модулем 2^m, множення в $\mathbf{GF}(2^8)$ та реалізація пошуку елемента для заміни у підстановках $\pi_0, \pi_1, \pi_2, \pi_3$. Внутрішня структура

деяких операцій дозволила знехтувати їх реалізацією - циклічні зсуви бітів та просто зсуви можна обчислювати у класичній моделі, тому їх склданість була обрана за $O(1)$.

За допомогою отриманих значень складності реалізації внутрішніх функції перестановок та операцій, що використовуються у ітераційному алгоритмі, оцінено складність реалізації «Купини» в залежності від кількості блоків повідомлень k:

– $l = 512$:

$$\text{Вентилі групи Кліффорда: } 2^{10.2} + k \cdot 2^{24.2} + (k+1) \cdot 2^{23.6} + (2k+1) \cdot 2^{10.24}$$
$$\text{T-вентилі: } 2^{9.9} + k \cdot 2^{23.9} + (k+1) \cdot 2^{23.8} + (2k+1) \cdot 2^{9.8}$$
$$\text{Глубина схеми: } 2^{10.7} + (2k+1) \cdot 2^{24.7}$$

– $l = 1024$:

$$\text{Вентилі групи Кліффорда: } 2^{12.8} + k \cdot 2^{25.7} + (k+1) \cdot 2^{25.6} + (2k+1) \cdot 2^{11.2}$$
$$\text{T-вентилі: } 2^{12.5} + k \cdot 2^{25.4} + (k+1) \cdot 2^{25.3} + (2k+1) \cdot 2^{10.8}$$
$$\text{Глубина схеми: } 2^{13.3} + (2k+1) \cdot 2^{26.2}.$$

Використовуючи алгоритм Гровера, проведений аналіз застосовності цього алгоритму до:

– Перестановок, що використовуються в ітераційному алгоритмі геш-функції. Для випадку $l = 512$ отрмані наступні оцінки: $O(2^{302.06})$ та $O(2^{305.76})$ для $T^{\oplus}_{512}$ та T^{+}_{512} відповідно. Стійкість перестановок $T^{\oplus}_{1024}$ та T^{+}_{1024} залишається під питанням, так як, можливо, значення кінцевої складності залежить від способу реалізації перестановки.

– Самої геш-функції. Для оцінки застосовності алгоритма Гровера тут достаньо було розглянути одну ітерацію геш-функції - застосування функції стиснення. Після проведення обчислень отримано наступні результати: для версії $l = 512$ - $O(2^{307.6})$, $l = 1024$ - $O(2^{566.6})$. Враховуючи той факт, що NIST вводила порогові значення для ключів довжини включно до 256 бітів, можна зробити висновок, що версія $l = 512$ геш-функції «Купина» у квантовій моделі є нестійкою, а стійкість версії $l = 1024$ є невизначеною.

ВИСНОВКИ

1) У ході цього дослідження проаналізовані методи щодо оцінки складності реалізації криптопримітву у квантовій моделі обчислень на прикладі геш-функції SM3, що входить у націоанальний стандарт Китая. Підхід, який використовувався у роботі, дозволив зрозуміти як потрібно підходити до процедури переведення геш-функції у квантову модель обчислень.

2) В якості підходів, що використовуються при квантовому криптоаналізі геш-функцій, проаналізован метод, який масштабує криптопримітив до спрощеної версії (*toy*-версії) зі збереженням приблизно такої ж за порядком стійкості, що й класичний варіант. У цій роботі, даний метод масштабування, розібран на прикладі двох геш-функцій: BLAKE2 та Sponge.

3) Досліджено тонкощі побудови та внутрішню стуктуру криптографічної геш-функції «Купина». Після аналізу вдалося виділити наступні компоненти побудови цього криптопримітиву: схема Меркля-Дамгора, високорівнева структура Девіса-Майєра, що застосовується у зв'язці із схемою Евен-Мансура в кожній ітерації алгоритма отримання геш-значення.

4) Першим етапом оцінки складності реалізації геш-функції «Купина» у квантовій моделі обчислень була реалізація функцій, які фігурують у перестановках $T_l^{\oplus}$ і T_l^{+} та обчислення загальної складності їх побудови. Вентильна складність та максимальна глибина схеми склали відповідно:

$$T_{512}^{\oplus} : O(2^{24.7}), O(2^{24.7});\ T_{512}^{+} : O(2^{25.06}), O(2^{24.7});$$
$$T_{1024}^{\oplus} : O(2^{26.5}), O(2^{26.16});\ T_{1024}^{+} : O(2^{26.6}), O(2^{26.16}).$$

Після отримання оцінок для перестановок, виведено формули, за якими можна оцінювати загальну складність реалізації геш-функції «Купина» у

квантовій моделі обчислень. За допомогою цих формул отримано оцінку зверху у випадку використання максимального числа блоків повідомлення. Оцінка складається з кількості вентилів групи Кліффорда, T-вентилів та загальної глибини схеми. Отримані значення відповідно дорівнюють: для $l = 512$ – $2^{111.9}$, $2^{111.85}$ та $2^{112.7}$; для $l = 1024$ – $2^{112.65}$, $2^{112.35}$ та $2^{113.2}$.

5) Обчисленні результати підтверджені за допомогою фреймоврка *Qiskit*, який повертає характеристики квантової схеми.

6) Використовуючи алгоритм квантового пошуку (алгоритм Гровера) реалізовано атаку на знаходження прообразу геш-функції. Для цього було оцінено вентильну складність та загальну глибину схеми, що реалізує одну ітерацію алгоритма отримання геш-значення. За результатами дослідження отримано наступні оцінки складності:

$$l = 512 : O(2^{307.6}), l = 1024 : O(2^{566.6}).$$

Ці результати показують, що атака за алгоритмом Гровера може бути реалізована у випадку, коли версія «Купини» використовує матриці стану розміром 8×8 (512 біт). Цей висновок зроблений на основі метрики NIST, що надає порогові значення для константи $SecurityLevelConstant$ із якою порівнюють результати обчислень складності реалізації атак у квантовій моделі. Щодо версій, які використовують матриці стану розміром 8×16 (1024 біт), оцінка, на даний момент, є невизначеною, враховуючи той факт, що NIST-матрика реалізована для розміру ключів до 256 біт включно.

Основним напрямком подальших досліджень є криптоаналіз «Купини» із боку алгоритма Саймона. Це є доволі цікавим варіантом атаки на геш-функцію, так як кожна ітерація «Купини» є зв'язкою двох схем: Евен-Мансура та конструкції Дєвіса-Майера. А як відомо за роботою Кувакадо та Мораї, алгоритм Саймона є застосовним до зламу схеми Евен-Мансура у квантовій моделі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Shor P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computers// Foundations of Computer Science: Conference Publications.—1997.—P. 1484-1509.

2. Lov K. Grover. A fast quantum mechanical algorithm for database search. In Gary L.Miller, editor, Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC 1996), pages 212-219. ACM, 1996.

3. ДСТУ 7564:2014. Інформаційні технології. Криптографічний захист інформації. Функція гешування. — Введ. 01-04-2015. — К.: Мінекономрозвитку України, 2015.

4. SM3 cryptographic hash algorithm [Електронний ресурс]. — Режим доступу: https://www.oscca.gov.cn/sca/xxgk/2010-12/17/1002389/files/302a3ada057c4a73830536d03e683110.pdf.

5. Preimage and pseudo-collision attacks on step-reduced SM3 hash function [Електронний ресурс]. — Режим доступу:https://www.sciencedirect.com/science/article/abs/pii/S0020019013000513?via%3Dihub

6. Finding Collisions for Round-Reduced SM3 [Електронний ресурс]. — Режим доступу:https://link.springer.com/chapter/10.1007%2F978-3-642-36095-4_12

7. F. Mendel, T. Nad, and M. Schlaer. Cryptanalysis of Round-Reduced HAS-160.In H. Kim, editor, ICISC, volume 7259 of LNCS, pages 3347. Springer, 2011.

8. Boomerang and Slide-Rotational Analysis of the SM3 Hash Function [Електронний ресурс]. — Режим доступу:https://link.springer.com/chapter/10.1007%2F978-3-642-35999-6_20

9. Improved boomerang attacks on round-reducedSM3 and keyed permutation of BLAKE-25 [Електронний ресурс]. — Режим доступу: https://ietresearch.onlinelibrary.wiley.com/doi/epdf/10.1049/iet-ifs.2013.0380

10. Duplexing the sponge: single-pass authenticated encryption and other applications [Електронний ресурс]. — Режим доступу:https://keccak.team/files/SpongeDuplex.pdf

11. ChaCha20 cipher [Електронний ресурс]. — Режим доступу:https://libsodium.gitbook.io/doc/advanced/stream_ciphers/chacha20

12. Grover on SM3 [Електронний ресурс]. — Режим доступу:https://eprint.iacr.org/2021/668

13. Кайдалов Д. С. Оценка эффективности SPN-структуры блочного симметричного шифра / Д. С. Кайдалов, Р. В. Олейников // Восточно-Европейский журнал передовых технологий. - 2014. - № 6(9). - С. 4-10. [Електронний ресурс]. — Режим доступу: http://nbuv.gov.ua/UJRN/Vejpte_2014_6(9)__2..

14. Thomas Peyrin. The Super-Sbox Cryptoanalysis. CCRG Seminar - Nanyang Technological University. Singapore - October 26, 2010. [Електронний ресурс]. — Режим доступу: http://www1.spms.ntu.edu.sg/~ccrg/documents/Presentation%20labo%20CCRG%20NTU%202010.pdf.

15. Coron J., Dodis Y., Malinaud C., Puniya P. Merkle-Damgrad Revisited: How to Construct a Hash Function.// Advances in Cryptology — CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings / V.Shoup — Springer - Verlag, 2005. — P. 430-448. — 572 p. — ((Lecture Notes in Computer Science; Vol. 3621)). [Електронний ресурс]. — Режим доступу: https://link.springer.com/content/pdf/10.1007%2F11535218_26.pdf.

16. Авезова Яна Эдуардовна. Современные подходы к построению хеш-функций на примере финалистов конкурса sha-3. Москва, 2015. [Електронний ресурс]. — Режим доступу: https://cyberrus.com/wp-content/uploads/2015/09/vkb_11_8.pdf.

17. Минимализм в криптографии, или схема Even-Mansour. [Електронний ресурс]. — Режим доступу: https://habr.com/ru/post/265607/.

18. Orr Dunkelman, Nathan Keller, Adi Shamir. Minimalism in

Cryptography: The Even-Mansour Scheme Reviseted. Computer Science Department University of Haifa. Haifa 31905, Israel. [Електронний ресурс]. — Режим доступу: https://eprint.iacr.org/2011/541.pdf.

19. Aaronson Scott, Gottesman Daniel. Improved simulation of stabilizer circuits // Physical Review A. — 2004. — Nov. — Vol. 70, no. 5.[Електронний ресурс]. — Режим доступу: http://dx.doi.org/10.1103/PhysRevA.70.052328.

20. M. Amy, O. D. Matteo, V. Gheorghiu, M. Mosca, A. Parent, and J. Schanck, "Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3,"2016.

21. Qiskit [Електронний ресурс]. — Режим доступу: https://qiskit.org/documentation/getting_started.html

22. Quantum full-adder circuit [Електронний ресурс]. — Режим доступу:https://quantumcomputing.stackexchange.com/questions/1654/how-do-i-add-11-using-a-quantum-computer

23. Quantum reversible circuit of AES-128 / Mishal Almazrooie, Azman Samsudin, Rosni Abdullah, Kussay Mutter // Quantum Information Processing. — 2018. — 03. — Vol. 17.

ДОДАТОК А ТЕКСТИ ПРОГРАМ

Тексти інструментальних програм для проведення експериментальних досліджень необхідно виносити у додатки.

А.1 Програма 1

```
from typing import Union
from qiskit import Aer, QuantumCircuit, QuantumRegister, transpile, AncillaRegister
from qiskit.circuit import Qubit, Instruction

simulator1 = Aer.get_backend ('qasm_simulator')[0]
simulatir2 = Aer.get_backend('aer_simulator')[0]
simulator.base_gates = ['x', 'cx', 'h', 's', 't', 'tdg']
base = ['x', 'cx', 'h', 's', 't', 'tdg']
```

Рисунок А.1 – Підготовичй етап для створення симуляції

```
def kupyna_one_iteration_estimate(circuit: QuantumCircuit, opt_level = 4):
    circuit.barrier()

    for realization_level in range(opt_level):
        iteration = transpile(circuit, simulator1, optimization_level = realization_level, basis_gates = base)
        print('Depth of circuit:', iteration.depth())
        print('Gate counts:', iteration.count_ops())
```

Рисунок А.2 – Характеристики схеми

```
def moebius_transform(table: list) -> list:
    ANF_sbox = table.copy()
    for i in range(6): # 8
        for j in range(64): # 256
            if (j >> i) & 1:
                ANF_sbox[j] ^= ANF_sbox[j ^ (1 << i)]

    return ANF_sbox
```

Рисунок А.3 – Швидке перетворення Мебіуса

```
def unitary_bit_indices(vector: int) -> list:
    indices = []
    vector_cut = list(bin(vector)[2:])[::-1][:6] #8

    l1 = []

    if len(vector_cut) <= 6: # 8
        for i in range(0, len(vector_cut)):
            if vector_cut[i] == '1':
                indices.append(i)

    if len(vector_cut) > 6: #8
        l1 = vector_cut[:6] #8
        for i in range(0, len(l1)):
            if vector_cut[i] == '1':
                l1.append(i)
        indices = l1

    return indices
```

Рисунок А.4 – Кількість перших n значущих бітів

```
def unitary_bit_indices(vector: int) -> list:
    indices = []
    vector_cut = list(bin(vector)[2:])[::-1][:6] #8

    l1 = []

    if len(vector_cut) <= 6: # 8
        for i in range(0, len(vector_cut)):
            if vector_cut[i] == '1':
                indices.append(i)

    if len(vector_cut) > 6: #8
        l1 = vector_cut[:6] #8
        for i in range(0, len(l1)):
            if vector_cut[i] == '1':
                l1.append(i)
        indices = l1

    return indices
```

Рисунок А.5 – Індекси одиничних бітів

```
def quantum_estimate_anf(anf: list) -> QuantumCircuit:

    x = QuantumRegister(6, 'x')
    out = QuantumRegister(6, 'out')
    anc_qubit = AncillaRegister(6, 'ancilla')

    circ = QuantumCircuit(x, out, anc_qubit)

    for i in range(1 << 6):
        if anf[i] != 0:
            ind = weight_of_n_significant_bit(i)
            weight = unitary_bit_indices(i)
            for ind_output in unitary_bit_indices(anf[i]):
                if weight == 0:
                    circ.x(out[ind_output])
                elif weight == 1:
                    circ.cx(x[ind[0]], out[ind_output])
                elif weight == 2:
                    circ.ccx(x[ind[0]], x[ind[1]], out[ind_output])
                else:
                    circ.mcx(x[ind], out[ind_output], anc_qubit, 'v-chain')

    kupyna_one_iteration_estimate(circ)

    return circ
```

Рисунок А.6 – Побудова схеми за АНФ

```
def multiplication(circ) -> QuantumCircuit:
    a = QuantumRegister(8, 'a')
    b = QuantumRegister(8, 'b')
    output = QuantumRegister(8, 'output')

    a_m_b = QuantumCircuit(a, b, output)

    for i in range(1, 8):
        for j in range(8-i):
            a_m_b.ccx(a[i+j], b[7-j], output[i-1])

    for i in range(8):
        for j in range(i+1):
            a_m_b.ccx(a[j], b[i-j], output[i])

    return a_m_b
```

Рисунок А.7 – Реалізація множення елементів у $\mathbf{GF}(2^8)$

ДОДАТОК Б ТАБЛИЦІ ЗАМІНИ ДЛЯ ШАРУ НЕЛІНІЙНОГО БІЄКТИВНОГО ВІДОБРАЖЕННЯ

Підстановка π_0:

```
A8 43 5F 06 6B 75 6C 59 71 DF 87 95 17 F0 D8 09
6D F3 1D CB C9 4D 2C AF 79 E0 97 FD 6F 4B 45 39
3E DD A3 4F B4 B6 9A 0E 1F BF 15 E1 49 D2 93 C6
92 72 9E 61 D1 63 FA EE F4 19 D5 AD 58 A4 BB A1
DC F2 83 37 42 E4 7A 32 9C CC AB 4A 8F 6E 04 27
2E E7 E2 5A 96 16 23 2B C2 65 66 0F BC A9 47 41
34 48 FC B7 6A 88 A5 53 86 F9 5B DB 38 7B C3 1E
22 33 24 28 36 C7 B2 3B 8E 77 BA F5 14 9F 08 55
9B 4C FE 60 5C DA 18 46 CD 7D 21 B0 3F 1B 89 FF
EB 84 69 3A 9D D7 D3 70 67 40 B5 DE 5D 30 91 B1
78 11 01 E5 00 68 98 A0 C5 02 A6 74 2D 0B A2 76
B3 BE CE BD AE E9 8A 31 1C EC F1 99 94 AA F6 26
2F EF E8 8C 35 03 D4 7F FB 05 C1 5E 90 20 3D 82
F7 EA 0A 0D 7E F8 50 1A C4 07 57 B8 3C 62 E3 C8
AC 52 64 10 D0 D9 13 0C 12 29 51 B9 CF D6 73 8D
81 54 C0 ED 4E 44 A7 2A 85 25 E6 CA 7C 8B 56 80
```

Підстановка π_1:

```
CE BB EB 92 EA CB 13 C1 E9 3A D6 B2 D2 90 17 F8
42 15 56 B4 65 1C 88 43 C5 5C 36 BA F5 57 67 8D
31 F6 64 58 9E F4 22 AA 75 0F 02 B1 DF 6D 73 4D
7C 26 2E F7 08 5D 44 3E 9F 14 C8 AE 54 10 D8 BC
1A 6B 69 F3 BD 33 AB FA D1 9B 68 4E 16 95 91 EE
4C 63 8E 5B CC 3C 19 A1 81 49 7B D9 6F 37 60 CA
E7 2B 48 FD 96 45 FC 41 12 0D 79 E5 89 8C E3 20
30 DC B7 6C 4A B5 3F 97 D4 62 2D 06 A4 A5 83 5F
2A DA C9 00 7E A2 55 BF 11 D5 9C CF 0E 0A 3D 51
7D 93 1B FE C4 47 09 86 0B 8F 9D 6A 07 B9 B0 98
18 32 71 4B EF 3B 70 A0 E4 40 FF C3 A9 E6 78 F9
8B 46 80 1E 38 E1 B8 A8 E0 0C 23 76 1D 25 24 05
F1 6E 94 28 9A 84 E8 A3 4F 77 D3 85 E2 52 F2 82
50 7A 2F 74 53 B3 61 AF 39 35 DE CD 1F 99 AC AD
72 2C DD D0 87 BE 5E A6 EC 04 C6 03 34 FB DB 59
B6 C2 01 F0 5A ED A7 66 21 7F 8A 27 C7 C0 29 D7
```

Підстановка π_2:

```
93 D9 9A B5 98 22 45 FC BA 6A DF 02 9F DC 51 59
4A 17 2B C2 94 F4 BB A3 62 E4 71 D4 CD 70 16 E1
49 3C C0 D8 5C 9B AD 85 53 A1 7A C8 2D E0 D1 72
A6 2C C4 E3 76 78 B7 B4 09 3B 0E 41 4C DE B2 90
25 A5 D7 03 11 00 C3 2E 92 EF 4E 12 9D 7D CB 35
10 D5 4F 9E 4D A9 55 C6 D0 7B 18 97 D3 36 E6 48
56 81 8F 77 CC 9C B9 E2 AC B8 2F 15 A4 7C DA 38
1E 0B 05 D6 14 6E 6C 7E 66 FD B1 E5 60 AF 5E 33
87 C9 F0 5D 6D 3F 88 8D C7 F7 1D E9 EC ED 80 29
27 CF 99 A8 50 0F 37 24 28 30 95 D2 3E 5B 40 83
B3 69 57 1F 07 1C 8A BC 20 EB CE 8E AB EE 31 A2
73 F9 CA 3A 1A FB 0D C1 FE FA F2 6F BD 96 DD 43
52 B6 08 F3 AE BE 19 89 32 26 B0 EA 4B 64 84 82
6B F5 79 BF 01 5F 75 63 1B 23 3D 68 2A 65 E8 91
F6 FF 13 58 F1 47 0A 7F C5 A7 E7 61 5A 06 46 44
42 04 A0 DB 39 86 54 AA 8C 34 21 8B F8 0C 74 67
```

Підстановка π_3:

```
68 8D CA 4D 73 4B 4E 2A D4 52 26 B3 54 1E 19 1F
22 03 46 3D 2D 4A 53 83 13 8A B7 D5 25 79 F5 BD
58 2F 0D 02 ED 51 9E 11 F2 3E 55 5E D1 16 3C 66
70 5D F3 45 40 CC E8 94 56 08 CE 1A 3A D2 E1 DF
B5 38 6E 0E E5 F4 F9 86 E9 4F D6 85 23 CF 32 99
31 14 AE EE C8 48 D3 30 A1 92 41 B1 18 C4 2C 71
72 44 15 FD 37 BE 5F AA 9B 88 D8 AB 89 9C FA 60
EA BC 62 0C 24 A6 A8 EC 67 20 DB 7C 28 DD AC 5B
34 7E 10 F1 7B 8F 63 A0 05 9A 43 77 21 BF 27 09
C3 9F B6 D7 29 C2 EB C0 A4 8B 8C 1D FB FF C1 B2
97 2E F8 65 F6 75 07 04 49 33 E4 D9 B9 D0 42 C7
6C 90 00 8E 6F 50 01 C5 DA 47 3F CD 69 A2 E2 7A
A7 C6 93 0F 0A 06 E6 2B 96 A3 1C AF 6A 12 84 39
E7 B0 82 F7 FE 9D 87 5C 81 35 DE B4 A5 FC 80 EF
CB BB 6B 76 BA 5A 7D 78 0B 95 E3 AD 74 98 3B 36
64 6D DC F0 59 A9 4C 17 7F 91 B8 C9 57 1B E0 61
```