НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ ЗАХИСТУ ІНФОРМАЦІЇ

«До захисту допущено»
В.о. завідувача кафедрою
_____ М.М.Савчук
(підпис) (ініціали, прізвище)
“___”_____2019 р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.040301 «Прикладна математика»
(код і назва)

на тему: Алгебраїчна атака на двійкові SNOW2.0-подібні потокові шифри

Виконала: студентка 4 курсу, групи ФІ-52
(шифр групи)
Овчарова Марина Андріївна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник професор кафедри ММЗІ, д-р. техн. наук. Олексійчук А.М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»
Фізико-технічний інститут

Кафедра математичних методів захисту інформації

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки - 6.040301 «Прикладна математика»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедрою

М.М.Савчук

_____
(підпис) (ініціали, прізвище)

«___»_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту

Овчаровій Марині Андріївні
(прізвище, ім'я, по батькові)

1. Тема роботи Алгебраїчна атака на двійкові SNOW2.0-подібні потокові шифри,
керівник роботи професор кафедри ММЗІ, д-р. техн. наук. Олексійчук Антон Миколайович,

затверджені наказом по університету від _____ р. № _____

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи узагальнення відомої алгебраїчної атаки на спрощену версію SNOW2.0 на довільні двійкові SNOW2.0-подібні потокові шифри.

4. Зміст роботи У цій роботі зроблен детальний аналіз існуючої алгебраїчної атаки на спрощену версію потокового шифру SNOW2.0 та теоретичних матеріалів, що потрібні для її розуміння, а також на базі цієї атаки запропонован спосіб побудови алгебраїчної атако на двійкові SNOW2.0-подібні потокові шифри.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) презетація

_____

_____

_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	провести огляд опублікованих джерел за заданою темою		
2	детальний аналіз існуючої алгебраїчної атаки на спрощену		
	версію потокового шифру SNOW2.0		
3	побудувати алгоритм, що дозволить розширити цю атаку		
	на двійкові SNOW2.0-подібні потокові шифри		
4	запропонувати практичну реалізацію цього алгоритму та		
	оцінити його ефективність		
5	описати спосіб побудови системи, вирішення якої надасть		
	змогу відновити початковий стан заданого шифру.		

Студент _____ Овчарова М.А
(підпис) (ініціали, прізвище)

Керівник роботи _____ Олексійчук А.М.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Кваліфікаційна робота містить: 61 сторінки, 7 рисунків, 21 джерело та 1 додаток.

У цій роботі зроблен детальний аналіз існуючої алгебраїчної атаки на спрощену версію потокового шифру SNOW2.0 та теоретичних матеріалів, що потрібні для її розуміння, а також на базі цієї атаки запропонован спосіб побудови алгебраїчної атаки на двійкові SNOW2.0-подібні потокові шифри.

Тема роботи: алгебраїчна атака на двійкові SNOW2.0-подібні потокові шифри.

Мета роботи: визначення параметрів систем нелінійних булевих рівнянь, які впливають на стійкість шифів відносно зазначеної алгебраїчної атаки на двійкові SNOW2.0-подібні потокові шифри.

Задача роботи: узагальнення відомої алгебраїчної атаки на спрощену версію SNOW2.0 на довільні двійкові SNOW2.0-подібні потокові шифри.

Об'єкт дослідження: процес перетворення інформації у двійкових SNOW2.0-подібних потокових шифрах.

Предмет дослідження: властивості компонент алгоритмів шифрування, що визначають їх стійкість відносно алгебраїчних атак.

Методи дослідження: методи теорії булевих функцій, абстрактної алгебри; методи системи компютерної алгебри $SageMath$.

У результаті цієї роботи запропоноване розширення відомої алгебраїчної атаки на спрощену версію потокового шифру SNOW2.0 на двійкові SNOW2.0-подібні потокові шифрів, яке у силу їх структури полягає у розробці алгоритму побудови системи рівнянь найменшого степеня, що описує нелінійну фунцію між регістрами пам'яті, яка може базуватися на будь-якому S-блоці, та у способі створення системи рівнянь, рішення якої надасть змогу відновити початковий стан двійкового SNOW2.0-подібного потокового шифру. Також представлена практична реалізація алгоритму побудови системи рівнянь найменшого

степеня за допомогою системи компютерної алгебри $SageMath$.

Результати цієї роботи були частково представлені на XVII Науково-практичній конференції студентів, аспірантів та молодих вчених "Теоретичні і прикладні проблеми фізики, математики та інформатики"(26-27 квітня 2019р., м. Київ).

SNOW2.0, ПОТОКОВІ ШИФРИ, СИМЕТРИЧНА КРИПТОГРАФІЯ, АЛГЕБРАЇЧНІ АТАКИ, СТІЙКІСТЬ ПОТОКОВИХ ШИФРІВ

РЕФЕРАТ

Квалификационная робота содержит: 61 страницы, 7 рисунков, 21 источник и 1 приложение.

В данной работе сделан детальный анализ существующей алгебраической атаки на упрощенную версию поточного шифра SNOW2.0 и теоретических материалов, необходимые для её понимания, на основе этой атаки предложен алгоритм, позволяющий расширить эту атаку на любые двоичные SNOW2.0-подобные потоковые шифры.

Тема работы: алгебраическая атака на двоичные SNOW2.0-подобные потоковые шифры.

Цель работы: формирование условий, которые определяют стойкость двоичных SNOW2.0-подобных потоковых шифров относительно известных алгебраических атак.

Задача работы: обобщить известную алгебраическую атаку на упрощенную версию SNOW2.0 на произвольные двоичные SNOW2.0-подобные потоковые шифры.

Объект исследования: процесс преобразования информации в двоичных SNOW2.0-подобных потоковых шифрах.

Предмет исследования: свойства компонент алгоритмов шифрования, определяющих их стойкость относительно алгебраических атак.

Методы исследования: методы теории булевых функций, абстрактной алгебры; методы системы компьютерной алгебры $SageMath$.

В результате этой работы предложено расширение известной алгебраической атаки на упрощенную версию потокового шифра SNOW2.0 на двоичные SNOW2.0-подобные потоковые шифры, которое в силу их структуры заключается в разработке алгоритма построения системы уравнений наименьшей степени, описывающая нелинейную фунцию между регистрами памяти, которая может базироваться на любом S -блоке, и в способе создания системы уравнений, решение

которой позволит восстановить исходное состояние двоичного SNOW2.0-подобного потокового шифра. Также представлена практическая реализация алгоритма построения системы уравнений наименьшей степени с помощью системы компьтерной алгебры $SageMath$.

Результаты этой работы были частично представлены на XVII Научно-практичной конференции студентов, аспирантов и молодых ученых "Теоретические и прикладные проблемы физики, математики и информатики"(26-27 апреля 2019г., г. Киев).

SNOW2.0, ПОТОКОВЫЕ ШИФРЫ, СИММЕТРИЧНАЯ КРИПТОГРАФИЯ, АЛГЕБРАИЧЕСКИЕ АТАКИ, СТОЙКОСТЬ ПОТОКОВЫХ ШИФРОВ

ABSTRACT

The thesis contains: 61 pages, 7 figures, 21 sources and 1 appendix.

In this thesis, a detailed analysis was made of the existing algebraic attack on a simplified version of the stream cipher SNOW2.0 and the theoretical materials that are needed for its understanding, and based on this attack, an algorithm was proposed that allows the attack to be extended to any binary SNOW2.0-type stream ciphers.

The theme of this thesis is an algebraic attack on binary SNOW2.0-type stream ciphers.

The goal of this thesis is definition of parameters of systems of nonlinear boolean equations that influence the stability of ciphers relative to the specified algebraic attack on binary SNOW2.0-type stream ciphers.

The task of this work is to generalize the well-known algebraic attack on the simplified version of SNOW2.0 to arbitrary binary SNOW2.0-type stream ciphers. The object of the research is the process of the mapping information in binary SNOW2.0-type stream ciphers.

The subject of the research are properties of the components of encryption algorithms, which determine their resistance against algebraic attacks.

Methods of research are methods of the theory of boolean functions, abstract algebra; methods of the system of computer algebra $SageMath$ were used.

As a result of this work, an extension of the known algebraic attack to the simplified version of the SNOW2.0 stream cipher is proposed on binary SNOW2.0-type stream ciphers, which, by their structure, consists in developing an algorithm for constructing a system of equations of the least degree describing a nonlinear function between registers of memory, which can be based on any S-block, and in a method for creating a system of equations, the solution of which will enable the initial state of the binary SNOW2.0-type stream cipher to be restored. The practical implementation of the algorithm

for constructing a system of equations of the lowest degree using the system of the computer algebra $SageMath$ is also presented.

The results of this thesis were partially presented at the XVII Scientific and Practical Conference of students, entrants and young specialists "Theoretical and Applied Problems of Physics, Mathematics and Informing"(April 26-27 2019, Kyiv).

SNOW2.0, STREAM CIPHERS, SYMMETRIC CRYPTOGRAPHY, ALGEBRAIC ATTACKS, RESISTANCE OF STREAM CIPHERS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

$\oplus$ — операція додавання за модулем 2.

$\boxplus$ — операція додавання за модулем 2^{32}.

РЗЛЗЗ — регістр зсуву з лінійним зворотним зв'язком.

СА — скінечний автомат.

$Rijndael(AES, AdvancedEncryptionStandard)$ — симетричний алгоритм блочного шифрування (розмір блока 128 біт, ключ 128/192/256 біт), фіналіст конкурсу AES.

S-блок — нелінійна таблиця заміни.

$SageMath$ — безкоштовна система математичне програмного забезпечення з відкритим кодом.

LaTeX— це високоякісна система з выльним доступом, що призначена для формування технічної та наукової документації.

ВСТУП

Актуальність дослідження. У сучасному світі існує достатньо потужних потокових шифрів, які забезпечують досить швидке шифрування інформації, проте деякі з них відрізняються досить складними перетвореннями та структурою. Постає питання, чи можна якось спростити обрахування у цих шифрах, не втрачаючи їх стійкість та не набуваючи нових уразливостей. Останнім часом високої популярності набирають потокові шифри, що базуються на потоковому шифрі SNOW2.0: "Струмок"(що є кандидатом на національний стандарт потокового шифрування України) [1], SNOW2.0-V(який розрахован на шифрування мобільних мереж 5G)[2]. У цій роботі проаналізована алгебраїчна атака спрощену версію потокового шифру SNOW2.0, та запропоноване її розширення на двійкові SNOW2.0-подібні потокові шифри. Виходячи із результатів цієї робити, далі можна буде розробити стійкий до цієї алгебраїчної атаки двійковий SNOW2.0-подібний потоковий шифр.

Мета дослідження: визначити параметри систем нелінійних булевих рівнянь, які впливають на стійкість шифів відносно зазначеної алгебраїчної атаки на двійкові SNOW2.0-подібні потокові шифри. Для досягнення мети необхідно розв'язати **задачу дослідження**, яка полягає в узагальненні відомої алгебраїчної атаці на спрощену версію потокового шифру SNOW2.0 на двійкові SNOW2.0-подібні потокові шифри. Для розв'язання задачі необхідно вирішити такі завдання:

1) провести огляд опублікованих джерел за тематикою дослідження;
2) зробити детальний аналіз існуючої алгебраїчної атаки на спрощену версію потокового шифру SNOW2.0;
3) видилити основні кроки побудови цієї атаки;
4) проаналізувати зміни, що можуть статися при приміненні цієї

атаки на двійкові SNOW2.0-подібні потокові шифри;

5) побудувати алгоритм, що дозволить розширити цю атаку на двійкові SNOW2.0-подібні потокові шифри;
6) запропонувати практичну реалізацію цього алгоритму та оцінити його ефективність;
7) описати спосіб побудови системи, вирішення якої надасть змогу відновити початковий стан двійкового SNOW2.0-подібного потокового шифру.

Об'єкт дослідження: процес перетворення інформації у двійкових SNOW2.0-подібних потокових шифрах.

Предмет дослідження: властивості компонент алгоритмів шифрування, що визначають їх стійкість відносно алгебраїчних атак.

При розв'язанні поставлених завдань використовувались такі *методи дослідження:* методи теорії булевих функцій, абстрактної алгебри, теорії кодування; для обчислення параметрів абстрактної алгебри використовувалась система компютерної алгебри $SageMath$.

Наукова новизна отриманих результатів полягає у введені поняття двійкових SNOW2.0-подібних потокових шифрів та в узагальненні відомої алгебраїчної атаки на спрощену версію потокового шифру SNOW2.0 на введені двійкові SNOW2.0-подібні потокові шифри, структура яких є більш узагальненою і представляє більшу кількість потокових шифрів.

Практичне значення Результати цієї роботи можуть бути використані для проведення алгебраїчної атаки на двійковий SNOW2.0-подібний потоковий шифр, а також для знаходження системи рівнянь мінімального степеня для будь-якого або таблиці істиності, що преставлена на прикладі S-блоку.

Апробація результатів та публікації. Результати цієї роботи були частково представлені на XVII Науково-практичній конференції студентів, аспірантів та молодих вчених "Теоретичні і прикладні проблеми фізики, математики та інформатики"(26-27 квітня 2019р., м. Київ).

1 ОГЛЯД ТЕОРЕТИЧНИХ МАТЕРІАЛІВ ЗА ЗАДАНОЮ ТЕМОЮ ДОСЛІДЖЕННЯ

У данному розділі проаналізовані існуючі джерела та виділені основні описи та терміни, що відносяться до теми цієї роботи, а саме описані потоковий шифр SNOW2.0, спрощена версія потокового шифру SNOW2.0, двійкові SNOW2.0-подібні потокові шифри та S-блок блочного шифру $Rijndael$, а також основні та необхідні означення із абстрактної алгебри у термінах булевих функцій.

1.1 Структура шифру SNOW2.0

Потоковий шифр SNOW2.0 [3] складається з РЗЛЗЗ з шістнадцятьма 32-бітними словами та СА з двома 32-бітовими регістрами пам'яті. Схема роботи шифру представлена на рисунку 1.1.

СА призначен для породження нелінійності, з цією метою він реалізує нелінійну бієкцію між регістрами пам'яті $\mathcal{S}$, що базується на S-блоці блочного шифру $Rijndael$.

Слід зауважити, що РЗЛЗЗ визначен над $GF(2^{32})$, він складається з шістнадцяти 32 бітних слів, що зумовлює 512 бітний розмір внутрійшного стану. Поле може бути також описане, як $GF(2^{32}) = GF(2)(\alpha, \beta)$, де β є коренем поліному над

$$x^8 + x^7 + x^5 + x^3 + 1 \in \mathbb{F}(2)[x],$$

та α є коренем поліному:

$$x^4 + \beta^{23}x^3 + \beta^{245}x^2 + \beta^{48}x + \beta^{239} \in \mathbb{F}(2^8)[x].$$

Тоді поліном зворотнього зв'язку приймає вигляд:

$$\alpha x^{16} + x^{14} + \alpha^{-1} x^5 + 1. \in \mathbb{F}_{2^{32}}[x].$$

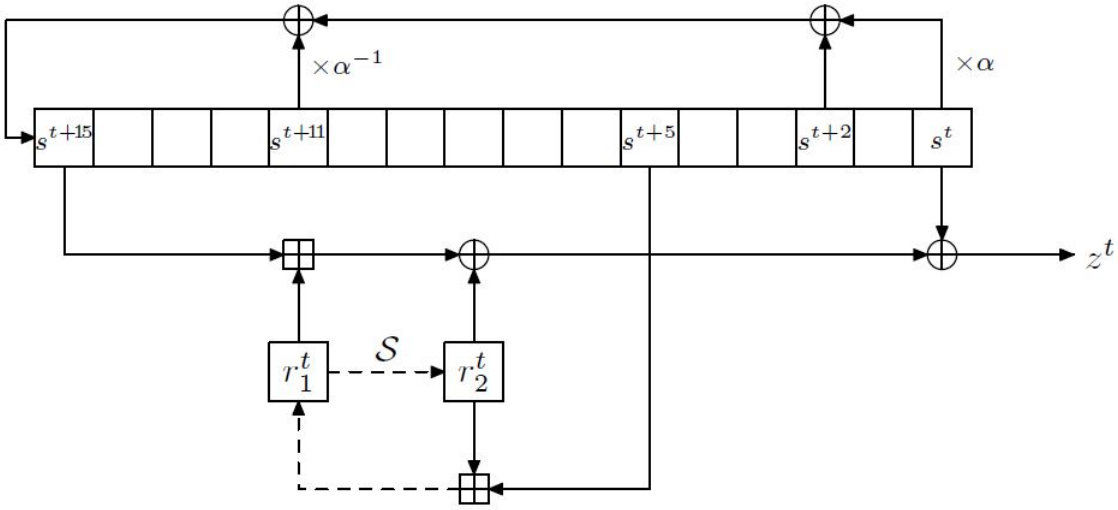

Рисунок 1.1 – Схема роботи шифру SNOW2.0

Правило роботи СА описується такою системою:

$$\begin{cases} r_2^{t+1} = \mathcal{S}(r_1^t), \\ r_1^{t+1} = r_2^t \boxplus s^{t+5}, \\ F^t = (r_1^t \boxplus s^{t+15}) \oplus r_2^t, \end{cases}$$

У нормальному режимі вихідний потік гами визначається, як $z^t = s^t \oplus F^t$, або:

$$z^t = s^t \oplus (r_1^t \boxplus s^{t+15}) \oplus r_2^t.$$

Спеціальний режим роботи (ініціалізація шифру, заповнення початкового стану регістру та СА; потік гами при цьому не виробляється) описується трохи іншим співвідношенням:

$$s^{t+16} = \alpha^{-1} s^{t+11} \oplus s^{t+2} \oplus \alpha s^t \oplus F^t.$$

1.2 Опис спрощеної версії потокового шифру SNOW2.0

У спрощеній версії потокового шифру SNOW2.0 додавання за модулем – $\boxplus$ замінюється на $\oplus$ в його описі, все інше залишається ідентичним, як на рисунку 1.2 [4].

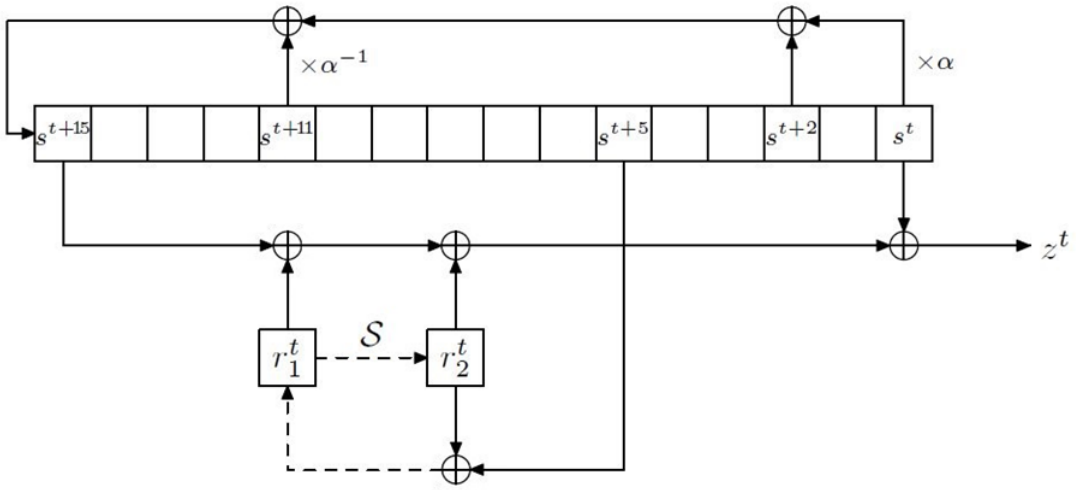

Рисунок 1.2 – Схема роботи спрощеної версії шифру SNOW2.0

У випадку спрощеної схеми правило роботи СА описується такою системою співвідношень:

$$\begin{cases} r_2^{t+1} = \mathcal{S}(r_1^t), \\ r_1^{t+1} = r_2^t \oplus s^{t+5}, \\ F^t = r_1^t \oplus s^{t+15} \oplus r_2^t, \end{cases}$$

Тепер у нормальному режимі вихідний потік гами визначається як $z^t = s^t \oplus F^t$, або

$$z^t = s^t \oplus r_1^t \oplus s^{t+15} \oplus r_2^t,$$

а спеціальний режим роботи – як

$$s^{t+16} = \alpha^{-1} s^{t+11} \oplus s^{t+2} \oplus \alpha s^t \oplus F^t.$$

1.3 Характеристика S-блоку блочного шифру $Rijndael$

S-блок є перестановкою на множині 32-бітових векторів, що базується на раундовій функції блочного шифру $Rijndael$ [5]. Нехай $w = (w_3, w_2, w_1, w_0)$ є вхідними даними S-блоку, де w_i, $i = 0..3$, – чотири байти з w та w_3 – найбільш значущий байт. Нехай

$$w = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix}$$

є вектором, що будемо подавати на вхід до S-блоку. Спочатку застосовуємо S-блок до кожного байту вектора, отримуємо:

$$\begin{pmatrix} S[w_0] \\ S[w_1] \\ S[w_2] \\ S[w_3] \end{pmatrix}$$

У перетворенні $MixColumn$ раундової функції блочного шифру $Rijndael$ кожен байт розглядається як елемент скінченного поля $\mathbb{F}_{2^8}$, породженого незвідним поліномом

$$x^8 + x^4 + x^3 + x + 1 \in \mathbb{F}_2[x].$$

Відповідно, кожен 4-байтовий вектор може бути представлений поліномом не більше ніж 3-го степеня над $\mathbb{F}_{2^8}$, коефіцієнтами якого виступають байти-координати даного слова. Під час виконання $MixColumn$ поліном, який

представляє вхідний вектор, множиться на фіксований поліном

$$c(y) = (x+1)y^3 + y^2 + y + x \in \mathbb{F}_{2^8}[y]$$

за модулем $y^4 + 1 \in \mathbb{F}_{2^8}[y]$; координати результуючого поліному (також степеня не вище 3) утворюють вектор, що є вихідним значенням процедури [6, 7].

Описане перетворення векторів через множення на поліном (як у блочному шифрі *Rijndael*) може бути обчислено еквівалентним чином через множення матриць:

$$\begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \end{pmatrix} = \begin{pmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{pmatrix} \begin{pmatrix} S[w_0] \\ S[w_1] \\ S[w_2] \\ S[w_3] \end{pmatrix}$$

де (r_3, r_2, r_1, r_0) – вектор вихідних байтів S-блоку. Ці байти конкатенуються та формують вихідне слово з S-блоку $r = \mathcal{S}(w)$ [8, 9].

1.4 Опис двійкових SNOW2.0-подібних потокових шифрів

У оригінальному SNOW2.0 та у його спрощеній версії, описаних вище, нелінійність у СА досягається за допомогою описаного відображення між регістрами пам'яті $\mathcal{S}$, що заснована на S-блоці блочного шифру *Rijndael*.

У випадку двійкових SNOW2.0-подібних потокових шифрів, використовуючи заміну модульного додавання $\boxplus$ на $\oplus$, як у спрощеній версії, ми відходимо від S-блоку блочного шифру *Rijndael* та розглядаємо функцію $\mathcal{N}$, що,базується на будь-якому S-блоці, який забезпечує нелінійність, тобто тепер нелінійна бієкція між регістрами

пам'яті має такий вигляд:

$$r_2^t = \mathcal{N}(r_1^{t-1}) \tag{1.1}$$

Структура двійкових SNOW2.0-подібних потокових шифрів зображена на рисунку 1.3.

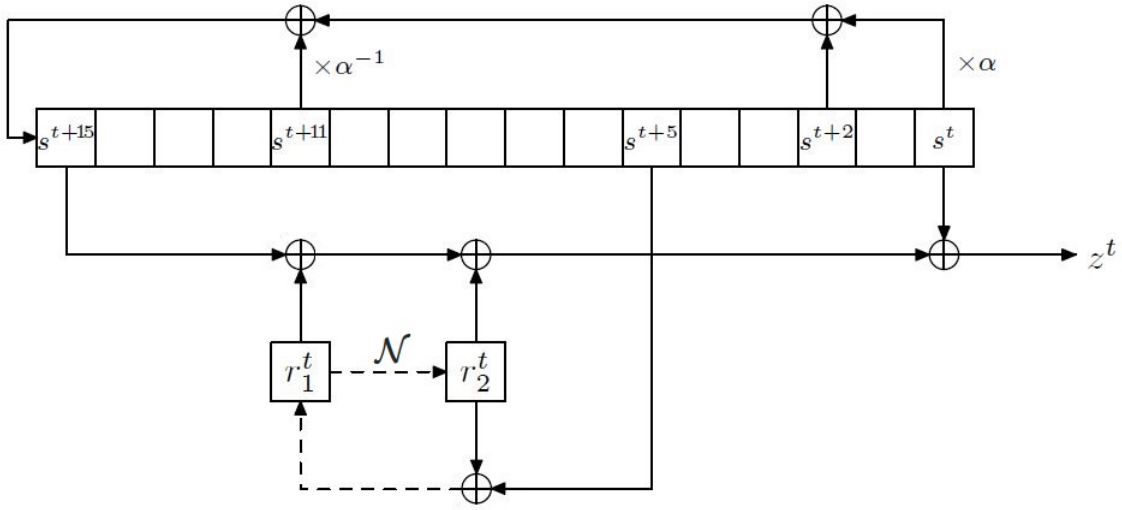

Рисунок 1.3 – Схема роботи двійкового SNOW2.0-подібного потокового шифру

Тепер правило роботи CA описується такою системою:

$$\begin{cases} r_2^{t+1} = \mathcal{N}(r_1^t), \\ r_1^{t+1} = r_2^t \oplus s^{t+5}, \\ F^t = r_1^t \oplus s^{t+15} \oplus r_2^t, \end{cases}$$

Усе інше залишається аналогічним спрощеній версії потокового шифру SNOW2.0.

1.5 Необхідні відомості абстрактної алгебри у термінах булевих функцій

Для будь-якого натурального n позначимо V_n множину двійкових векторів довжини n, B_n – множину булевих функцій від n змінних. Множина B_n є комутативним кільцем відносно операцій покомпонентного додавання та множення булевих функцій:

$$\forall f, g \in B_n : \ (f \oplus g)(x) = f(x) \oplus g(x), \ \ (fg)(x) = f(x)g(x), \ x \in V_n.$$

Означення 1.1. Множина $I \subseteq B_n$ називається **ідеалом кільця** B_n, якщо виконується умова: $\forall f \in B_n \forall g_1, g_2 \in I : \ g_1 \oplus g_2 \in I, \ fg_1 \in I$.
Запис $I \lhd B_n$ означає, що I є ідеалом кільця B_n.

Ідеал, породжений множиною $\{g_1, ..., g_m\} \subseteq B_n$, визначається за формулою:

$$< g_1, ..., g_m >= \{f_1 g_1 \oplus ... \oplus f_m g_m : \ f_1, ..., f_m \in B_n\}.$$

Для будь-яких $I \lhd B_n$, $M \subseteq B_n$ покладемо:

$$V(I) = \{x \in V_n | \ \forall g \in I : \ g(x) = 0\}, \tag{1.2}$$

$$J(M) = \{g \in B_n | \ \forall x \in M : \ g(x) = 0\}. \tag{1.3}$$

Означення 1.2. Множина (1.2) називається алгебраїчним многовидом [10] або множиною нулів ідеалу I.

Означення 1.3. Множина (1.3) є ідеалом, що складається з усіх булевих функцій, які обертаються в нуль на M.

Твердження 1.1. $\forall I \lhd B_n, M \subseteq B_n$

$$J(V(I)) = I, \ V(J(M)) = M$$

Зокрема, існує взаємно однозначна відповідність між ідеалами кільця B_n та підмножинами множини V_n (так, що кожен ідеал однозначно визначається множиною його нулів). Крім того, кожен ідеал $I \lhd B_n$ породжується єдиною булевою функцією χ_I, яка визначається за формулою:
$chi_I(x) = 0$, якщо $x \in V(I)$; $chi_I(x) = 1$ – у протилежному випадку, $x \in V(I)$.

Означення 1.4. Нехай I – довільний ідеал кільця B_n. Тоді множина

$$Ann(I) = \{f \in B_n | \forall \in I : \ fg = 0\}$$

також є ідеалом, який називається **анулятором ідеалу** I. Анулятор функції $f \in B_n$ визначається як анулятор ідеалу, що породжується цією функцією: $Ann(f) = Ann(< f >)$.

Твердження 1.2. *Для будь-якого $I \lhd B_n$ кільце B_n розкладається в пряму суму ідеалів I та $Ann(I)$. Іншими словами, кожна функція допускає однозначне представлення у вигляді $f = g \oplus g^{\perp}$, де $g \in I$, $g^{\perp} \in Ann(I)$ Крім того, якщо $I =< g_0 >$, то $Ann(I) =< g_0 \oplus 1 >$.*

Кожна функція $f \in B_n \setminus \{0\}$ має єдине представлення у вигляді полінома Жегалкіна, тобто полінома вигляду $f(x) = \bigoplus_{\alpha \in V_n} c_\alpha x^\alpha$, де $c_\alpha \in \{0, 1\}$, $x^\alpha = x_1^{\alpha_1}, ..., x_n^{\alpha_n}$, $x = (x_1, ..., x_n)$, $\alpha = (\alpha_1, ..., \alpha_n \in V_n)$. Число $|\alpha| = \alpha_1 + ... + \alpha_n$ називається степенем монома x^α, а число $degf = max\{|\alpha| : c_\alpha = 1, \alpha \in V_n\}$ – степенем функції f. Мінімальний степінь ідеалу $I \lhd B_n$ визначається за формулою $min\ degI = min\{deg\ f : \ f \in I \setminus \{0\}\}$.
Розглянемо векторну булеву функцію (S-блок) $S : V_n \to V_n$ з координатними функціями $S_1, ..., S_n$ та ідеал кільця булевих функцій від $2n$ змінних $x = (x_1, ..., x_n)$, $y - (y_1, ..., y_n)$ який визначається за формулою

$$I(S) =< y_1 \oplus S_1(x), ..., y_n \oplus S_n(x) > .$$

Алгебраїчна імунність функції S визначається рівністю [11]:

$$AI(S) = min\ degI(S). \tag{1.4}$$

Твердження 1.3. *Алгебраїчна імунність векторної функції* $S : V_n \to V_n$ *дорівнює:*

1) мінімуму степенів усіх функцій $g \in B_{2n}$, *що задовольняють умові* $\forall x, y \in V_n : S(x) = y \Rightarrow g(x, y) = 0$ *(в цьому випадку говорять, що рівняння* $g(x, y) = 0$ *описує векторну функцію* S*);*

2) мінімальному степеню ідеалу $Ann(f_S)$, *де функція* $f_S : V_{2n} \to \{0, 1\}$ *визначається за формулою:*
$f_S(x, y) = 1$ *, якщо* $S(x) = y$;
$f_S(x, y) = 0-$ *у протилежному випадку,* $x, y \in V_n$.

Таким чином, для обчислення алгебраїчної імунності векторної функції S достатньо визначити функцію f_S та знайти найменший степінь ненульових булевих функцій, що її анулюють.

Означення 1.5. Позначимо N_0^n множину векторів довжини n з невід'ємними цілими координатами. Ця множина є полугрупою відносно операції + покоординатного додавання векторів. Відношення часткового порядку $\leq$ на множині N_0^n визначається за формулою:

$$\forall \alpha = (\alpha_1, .., \alpha_n),\ \beta = (\beta_1, ..., \beta_n) \in N_0^n :\ \alpha \leq \beta \Leftrightarrow (\alpha_i \leq \beta_i,\ i = 1, 2, ..., n).)$$

Число $|\alpha| = \alpha_1 + ... + \alpha_n$ називається мультистепенем вектора

$$\alpha = (\alpha_1, ..., \alpha_n) \in N_0^n.$$

Означення 1.6. Будь-яке відношення лінійного порядку $\preceq$ на множині N_0^n, що задовольняє умовам:

1) $\forall \alpha, \beta \in N_0^n :\ \alpha \leq \beta \Rightarrow \alpha \preceq \beta$;

2) $\forall \alpha, \beta, \gamma \in N_0^n :\ \alpha \preceq \beta \Rightarrow \alpha + \gamma \preceq \beta + \gamma$,

називається **мономіальним впорядкуванням** на N_0^n. Мономіальне впорядкування $\preceq$ називається **степеневим**, якщо для будь-яких $\alpha,\ \beta \in N_0^n$ виконується умова $|\alpha| \leq |\beta| \Rightarrow \alpha \preceq \beta$.

В подальшому запис $\alpha \prec \beta$ $(\alpha < \beta)$ означає, що $\alpha \preceq \beta$ $(\alpha \leq \beta)$ та $\alpha \neq \beta$. Будь-яке мономіальне впорядкування $\preceq$ дозволяє лінійно впорядкувати булеві мономи $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ за правилом: $x^\alpha \preceq x^\beta \Leftrightarrow \alpha \preceq \beta,\ \alpha,\ \beta \in V_n$ (тут і далі множина двійкових векторів V_n розглядається як підмножина полугрупи N_0^n) [12]. Зазначений порядок на множині мономів дозволяє визначити старший моном будь-якої ненульової булевої функції $f(x) = \bigoplus_{\alpha \in V_n} c_{\alpha,f} x^\alpha$, де $c_{\alpha,f} \in \{0,1\}, \alpha \in V_n$:

$$HM_{\preceq}(f) = \max_{\preceq}\{x^\alpha :\ c_{\alpha,f} = 1\}.$$

За означенням моном x^α ділиться на моном x^β, якщо виконується умова $\alpha \geq \beta,\ \alpha, \beta \in V_n$ [13].

Нехай I – ненульовий ідеал кільця булевих функцій від n змінних.

Означення 1.7. Система функцій $g_1, ..., g_m \in I$ називається **базисом Грьобнера** ідеалу I відносно мономіального впорядкування $\preceq$ на множині N_0^n, якщо старший моном будь-якої функції $f \in I$ ділиться на один з мономів $HM_{\preceq}(g_i), i = 1, 2, ..., m$. Базис Грьобнера $g_1, ..., g_m$ називається **мінімальним**, якщо $HM_{\preceq}(g_i)$ не ділиться на $HM_{\preceq}(g_j)$ для будь-яких $i \neq j$ [14].

Твердження 1.4. *[15] Для будь-якого ненульового ідеалу $I \triangleleft B_n$ існує мінімальний базис Грьобнера.*

Твердження 1.5. *[11] Нехай $S : V_n \to V_n$ – векторна булева функція, $\preceq$ – довільне степеневе мономіальне впорядкуванням на множині N_0^n, G – мінімальний базис Грьобнера ідеалу $I(S)$ відносно цього впорядкування. Нехай, далі, $g_1, ..., g_m$ є усі функції з G, що мають найменший степінь d. Тоді*

1) $AI(S) = d$;

2) кожна функція $f \in I(S)$ степеня d допускає єдине представлення у вигляді

$$f = c_1 g_1 \oplus \cdots \oplus c_m g_m$$

де $c_i \in,\ i = 1, 2, ..., m$. Зокрема, ідеал $I(S)$ містить точно 2^m функцій степеня d.

Висновки до розділу 1

У цьому розділі проаналізовані джерела та наведені поняття, що знадобляться для подальшої роботи, а саме описані шифр SNOW2.0, спрощена версія шифру SNOW2.0 та S-блок блочного шифру *Rijndael*, на якому базується нелінійна бієкція між регістрами пам'яті $\mathcal{S}$, також наведені основні та необхідні означення із абстрактної алгебри у термінах булевих функцій, які знадобляться для покращення систем рівнянь описуючих S-блок.

2 АНАЛІЗ КОНСТРУКЦІЇ ІСНУЮЧОЇ АЛГЕБРАЇЧНОІ АТАКИ НА СПРОЩЕНУ ВЕРСІЮ ПОТОКОВОГО ШИФРУ SNOW2.0

У цьому розділі буде проаналізована конструкція алгебраїчноі атаки на спрощену версію потокового шифру SNOW2.0 [4] та побудову системи квадратних рівнянь у S-блоці блочного шифру *Rijndael* [5].

2.1 Алгебраїчна атака на спрощену версію потокового шифру SNOW2.0

Після опису спрощеної версії потокового шифру SNOW2.0 у першому розділі перейдемо до самої атаки.

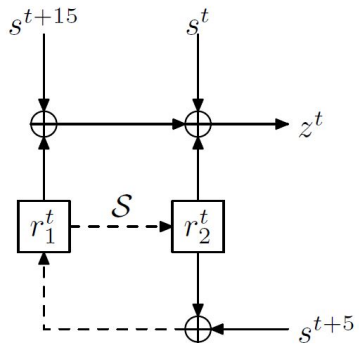

Рисунок 2.1 – Спрощена версія потокового шифру SNOW2.0

Отримуємо систему рівнянь:

$$\begin{cases} z^t = s^{t+15} \oplus r_1^{t+15} \oplus s^t \oplus r_2^t, \\ r_1^t = r_2^{t-1} \oplus s^{t+4}, \end{cases}$$

Яке перетворюється на $r_2^t = r_2^{t-1} \oplus z^t \oplus s^{t+15} \oplus s^{t+4} \oplus s^t$, з цього отримуємо вираз для регістра r_2 для будь-якого кроку t, що включає потік ключа, початкове заповнення РЗЛЗЗ s^0, ..., s^{15} та початкове заповнення r_2^0 регістру r_2. Заносимо до рівняння, для кожного кроку t, відомі коефіцієнти e_t^i, що

$$\boldsymbol{r_2^t = r_2^0 \bigoplus_{i=0}^{t} z_i \bigoplus_{j=0}^{15} e_t^j s_j}$$

,

Нехай t=0 для першої вихідної послідовності гами. Легко перевірити, що регістр r_1 оновлюється за правилом $r_1^{t+1} = r_2^t \oplus s^{t+5}$ (також початковий стан регістру r_1 можна отримати зі зтану r_2^0 та з відношення $r_1^0 = r_2^0 \oplus s^0 \oplus s^{15} \oplus z^0$). Іншими словами, ми позбулися вливання пам'яті на кроки t>0, тому що можливе лінійне вираження через початкове заповнення РЗЛЗЗ та r_2^0.

Властивість, що дозволяє визначити потік гами за допомогою лінійної функції з $r_2^0, s^0, ..., s^{15}$, що містяться у r_1 та r_2, можуть бути представлені на рисунку 2.2.

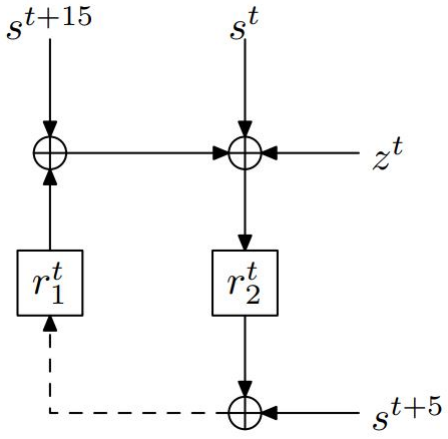

Рисунок 2.2 – відстеження регістрів пам'яті r1 та r2

Виходячи с того, що функція $\mathcal{S}$ побудована на S-блоці шифру *Rijndael*, отримуємо 156 квадратних рівнянь між $r_1^t = r_2^t \oplus s^t \oplus s^{t+15} \oplus z^t$ та r_2^{t+1}.

Відновлення початкового стану РЗЛЗЗ та ключа можна звести до

знаходження розв'яку такої системи, проте існує дві різні стратегії.

Перша базується на лінеарізації системи, проте у такому випадку вона задіє приблизно 2^{51} операцій.

Інша стратегія полягає у пошуку розв'язка систем квадратичних рівнянь без лінеризації. У такому випадку складність рішення системи значно вища за вирішення лінеарізованої.

Як тільки початковий стан $s^0, ..., s^{15}$ та r_2^0 буде відновлений, використовуючи метод лінеарізації, r_1^0 буде отримано з виразу $r_1^0 = r_2^0 \oplus s^0 \oplus s^{15} \oplus z^0$, так що стане відомо повний стан шифру при t=0 [9, 16]. Для того щоб отримати секретний ключ K - і таким чином бути у змоззі передбачити подальшу послідовність гами для інших початкових станів - достатньо запустити один раз шифр у зворотньому порядку у нормальному режимі та 32 рази у спеціальному режимі зворотнього зв'язку. З цього випливає, що переходи станів у потоковому шифрі SNOW2.0 у нормальному та у спеціальному режимах є оберненими. Виходячи з ціого, ми зможемо отримати стан РЗЛЗЗ під час ініціалізації, який отриман із знання початкового стану, значення секретного ключа .

2.2 Побудова системи рівнянь, що описують S-блок блочного шифру *Rijndael*

Зазначимо, S-блок блочного шифру *Rijndael* - це відображення $S : GF(2)^8 \to GF(2)^8$, що представлене інверсійною функцією $I(x)$, яка є зворотньою за модулем многочлена $m(t) = t^8 + t^4 + t^3 + t + 1$ та функція афінного перетворення $A(x)$, де x - байтова змінна, що складається з біт x_i, де $i = 0, ..., 7$ та x_7 представляє старший біт у

$$x = \sum_{i=0}^{7} x_i t^i \tag{2.1}$$

Тепер визначимо інверсійну функцію $I(x)$:

$$I(x) = x^{254} \mod m(t) \tag{2.2}$$

Також слід зазначити, що інверсія (2.2) відображає 0 у 0, а афінне перетворення $A(x)$ може бути представлено у вигляді модулярного полінома:

$$A(x) = a\ x \mod (t^8 + 1) \oplus b, \tag{2.3}$$

де a ='1F' та b ='63'. Ці двузначні числа у шістнадцятковій системі позначають константний байт, який можна представити у вигляді поліному. Наприклад, число '63' має представляння у двійковому векторі $(01100011)^T$ або у вигляді поліному $t^6 + t^5 + t + 1$. Анагічно $A(x)$ можна записати у матричному вигляді, де коефіцієнти байтових змінни будуть представленні у векторному вигляді:

$$A(x) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \tag{2.4}$$

Тому весь S-блок блочного шифру *Rijndael* можна представити:

$$S(x) = A \circ I = A(I(x))$$

Позначимо $y = I(x)$ та $z = S(x) = A(I(x)) = A(y)$. Із інверсійності $y = I(x)$ слідує, що $xy = 1$, якщо $x \neq 0$, у поліноміальній формі:

$$\left(\sum_{i=0}^{7} x_i t^i\right)\left(\sum_{j=0}^{7} x_j t^j\right) \mod m(t) = \sum_{k=0}^{7} 0 \cdot t^k + 1 \tag{2.5}$$

Зазначене вище ділення за модулем виконується аналітично та потім проводиться порівняння коефіцієнтів членів однакового порядку за $t^k, 0 \leq k \leq 7$. Це дає змогу визначити перші вісім багатовимірних квадратичних рівнянь для S-блоку блочного $Rijndael$.

Скористаємося середою $Cocalc$ [17], що базується на $SageMath$ [18, 19] для моделювання та оцінки таких рівнянь. Для роботи з поліномами виду (2.1) ця середа надає модулі для побудови кільця багатовимірних поліномів. У рядку 1 лістингу 2.1 для зручності визначена змінна кількості біт. У рядку 2 створюється список назв трьох байтових поліномів($['x0', 'x1', ...'z7']$).

Лістинг 2.1 – Байтові поліноми над факторним кільцем

```
1. nb = 8
2. varl = [c+str(p) for c in 'xyz' for p in range (nb)]
3. B = BooleanPolynomialRing(names = varl)
4. B.inject_variables()
5. P.<p> = PolynomialRing (B)
6. Byte.<t> = P.quotient_ring(p^8 + p^4 + p^3 + p + 1)
7. X = B.gens()[:nb]
8. Y = B.gens()[nb:2*nb]
9. x = sum([X[j]*t^j for j in range (nb)])
10. y = sum([Y[j]*t^j for j in range (nb)])
```

У рядку 3 створюється булеве кільце поліномів із зазначеними коефіцієнтами та присвоює їм властивості $GF(2)$. Рядок 4 надає змогу використовувати назви коефіцієнтів у якості змінних. У рядку 5

створюється поліноміальне кільце над булевим поліноміальним кільцем B, та потім, у рядку 6, отримуємо кінцеве фактор кільце із модулем $m(t)$. Для зручності у рядках 7 та 8 генерується список коефіцієнтів змінних цього фактор кільця. За допомогою створених списків у рядках 9 та 10 моделюються поліноми x та y.

Лістинг 2.2 – Формування рівнянь для коефіцієнтів

```
1. E3 = x*y
2. eqs3 = E3.list()
3. latex(eqs3)
```

У другому рядку у лістингу 2.2 отримуємо коефіцієнти усіх степеней t у рінянні, зазначеному у рядку 1. Третій рядок формує ці рівняння та подає у виді документу LaTeX[20]:

$$
\begin{aligned}
c_0 &= x_0y_0 + x_1y_7 + x_2y_6 + x_3y_5 + x_4y_4 + x_5y_3 + x_5y_7 + x_6y_2 \\
&\quad + x_6y_6 + x_6y_7 + x_7y_1 + x_7y_5 + x_7y_6 \\
c_1 &= x_0y_1 + x_1y_0 + x_1y_7 + x_2y_6 + x_2y_7 + x_3y_5 + x_3y_6 + x_4y_4 \\
&\quad + x_4y_5 + x_5y_3 + x_5y_4 + x_5y_7 + x_6y_2 + x_6y_3 + x_6y_6 + x_7y_1 \\
&\quad + x_7y_2 + x_7y_5 + x_7y_7 \\
c_2 &= x_0y_2 + x_1y_1 + x_2y_0 + x_2y_7 + x_3y_6 + x_3y_7 + x_4y_5 + x_4y_6 \\
&\quad + x_5y_4 + x_5y_5 + x_6y_3 + x_6y_4 + x_6y_7 + x_7y_2 + x_7y_3 + x_7y_6 \\
c_3 &= x_0y_3 + x_1y_2 + x_1y_7 + x_2y_1 + x_2y_6 + x_3y_0 + x_3y_5 + x_3y_7 \\
&\quad + x_4y_4 + x_4y_6 + x_4y_7 + x_5y_3 + x_5y_5 + x_5y_6 + x_5y_7 + x_6y_2 \\
&\quad + x_6y_4 + x_6y_5 + x_6y_6 + x_6y_7 + x_7y_1 + x_7y_3 + x_7y_4 + x_7y_5 \\
&\quad + x_7y_6 + x_7y_7 \\
c_4 &= x_0y_4 + x_1y_3 + x_1y_7 + x_2y_2 + x_2y_6 + x_2y_7 + x_3y_1 + x_3y_5 \\
&\quad + x_3y_6 + x_4y_0 + x_4y_4 + x_4y_5 + x_4y_7 + x_5y_3 + x_5y_4 + x_5y_6
\end{aligned}
$$

$$
\begin{aligned}
& +x_6y_2 + x_6y_3 + x_6y_5 + x_7y_1 + x_7y_2 + x_7y_4 + x_7y_7 \\
c_5 \;=\; & x_0y_5 + x_1y_4 + x_2y_3 + x_2y_7 + x_3y_2 + x_3y_6 + x_3y_7 + x_4y_1 \\
& +x_4y_5 + x_4y_6 + x_5y_0 + x_5y_4 + x_5y_5 + x_5y_7 + x_6y_3 + x_6y_4 \\
& +x_6y_6 + x_7y_2 + x_7y_3 + x_7y_5 \\
c_6 \;=\; & x_0y_6 + x_1y_5 + x_2y_4 + x_3y_3 + x_3y_7 + x_4y_2 + x_4y_6 + x_4y_7 \\
& +x_5y_1 + x_5y_5 + x_5y_6 + x_6y_0 + x_6y_4 + x_6y_5 + x_6y_7 + x_7y_3 \\
& +x_7y_4 + x_7y_6 \\
c_7 \;=\; & x_0y_7 + x_1y_6 + x_2y_5 + x_3y_4 + x_4y_3 + x_4y_7 + x_5y_2 + x_5y_6 \\
& +x_5y_7 + x_6y_1 + x_6y_5 + x_6y_6 + x_7y_0 + x_7y_4 + x_7y_5 + x_7y_7
\end{aligned} \tag{2.6}
$$

Ці рівняння практично ідентичні рівнянням (2.5), відмінність лише у порядку доданків у кожному рівнянні. Коеефіцієнти на i-тому місці відповідають коефіцієнтам при i-тому порядку($c_i = eqs3[i], i = 0, ...7$).

З рівняння (2.3), використовуючи $z = A(y)$ отримуємо:

$$y = a^{255}(z + b) \mod (t^8 + 1) \tag{2.7}$$

Виходячи з того, що $a^{255}a \mod (t^8 + 1) = 1$. Записуючи по бітах рівняння (2.7) отримуємо:

$$
\begin{cases}
y_7 = z_6 + z_4 + z_1 \\
y_6 = z_5 + z_3 + z_0 \\
y_5 = z_7 + z_4 + z_2 \\
y_4 = z_6 + z_3 + z_1 \\
y_3 = z_5 + z_2 + z_0 \\
y_2 = z_7 + z_4 + z_1 + 1 \\
y_1 = z_6 + z_3 + z_0 \\
y_0 = z_7 + z_5 + z_2 + 1
\end{cases} \tag{2.8}
$$

Можна замінити y на (2.8) у рівняннях (2.6), щоб отримати остаточну форму перших восьми рівнянь, що представляють S-блок блочного

шифру *Rijndael*. За допомогою рівняння (2.7) у лістингу (2.3) генеруємо цю заміну.

Лістинг 2.3 – Генерація заміни змінних у

```
1. Baff.<u> = P.quotient_ring (p^8 + 1)
2. Z = B.gens()[2*nb:][:nb]
3. z = sum ([Z[ j ]*u^j for j in range (nb)])
4. a = u^4 + u^3 + u^2 + u + 1
5. b = u^6 + u^5 + u + 1
6. eqs4 = dict(zip(Y,( a^255*(z + b)).list()))
```

У першому рядку визначаємо фактор кільце за модулем $t^8 + 1$. У наступних чотирьох рядках визначаємо байтову змінну z та два константних полінома a та b. Рівняння (2.7) зазначено, як `a^255 * (z + b)` у лістингу (2.3). Резльтатом отримуємо у лістингу (2.4):

Лістинг 2.4 – Заміна змінних у

```
{y7: z6 + z4 + z1,
y6: z5 + z3 + z0,
y5: z7 + z4 + z2,
y4: z6 + z3 + z1,
y3: z5 + z2 + z0,
y2: z7 + z4 + z1 + 1,
y1: z6 + z3 + z0,
y0: z7 + z5 + z2 + 1}
```

Тепер підставимо це у рівняння (2.6) за допомою команди:

```
eqs5 = [_.substitute(eqs4) for _ in eqs3]
```

Результатом знову маємо набір із восьми багатовимрних квадратичних рівнянь, шляхом прирівнювання байтвої змінної до 1:

$$
\begin{aligned}
1 &= x_0z_2 + x_0z_5 + x_0z_7 + x_0 + x_1z_1 + x_1z_4 + x_1z_6 + x_2z_0 \\
&\quad + x_2z_3 + x_2z_5 + x_3z_2 + x_3z_4 + x_3z_7 + x_4z_1 + x_4z_3 + x_4z_6 \\
&\quad + x_5z_0 + x_5z_1 + x_5z_2 + x_5z_4 + x_5z_5 + x_5z_6 + x_6z_0 + x_6z_3 \\
&\quad + x_6z_5 + x_6z_6 + x_6z_7 + x_6 + x_7z_2 + x_7z_4 + x_7z_5 + x_7z_6 + x_7z_7 \\
0 &= x_0z_0 + x_0z_3 + x_0z_6 + x_1z_1 + x_1z_2 + x_1z_4 + x_1z_5 + x_1z_6 \\
&\quad + x_1z_7 + x_1 + x_2z_0 + x_2z_1 + x_2z_3 + x_2z_4 + x_2z_5 + x_2z_6 \\
&\quad + x_3z_0 + x_3z_2 + x_3z_3 + x_3z_4 + x_3z_5 + x_3z_7 + x_4z_1 + x_4z_2 \\
&\quad + x_4z_3 + x_4z_4 + x_4z_6 + x_4z_7 + x_5z_0 + x_5z_2 + x_5z_3 + x_5z_4 \\
&\quad + x_5z_5 + x_6z_1 + x_6z_2 + x_6z_3 + x_6z_4 + x_6z_7 + x_6 + x_7z_0 \\
&\quad + x_7z_2 + x_7z_3 + x_7z_4 + x_7 \\
0 &= x_0z_1 + x_0z_4 + x_0z_7 + x_0 + x_1z_0 + x_1z_3 + x_1z_6 + x_2z_1 \\
&\quad + x_2z_2 + x_2z_4 + x_2z_5 + x_2z_6 + x_2z_7 + x_2 + x_3z_0 + x_3z_1 \\
&\quad + x_3z_3 + x_3z_4 + x_3z_5 + x_3z_6 + x_4z_0 + x_4z_2 + x_4z_3 + x_4z_4 \\
&\quad + x_4z_5 + x_4z_7 + x_5z_1 + x_5z_2 + x_5z_3 + x_5z_4 + x_5z_6 + x_5z_7 \\
&\quad + x_6z_0 + x_6z_2 + x_6z_3 + x_6z_4 + x_6z_5 + x_7z_1 + x_7z_2 + x_7z_3 \\
&\quad + x_7z_4 + x_7z_7 + x_7 \\
0 &= x_0z_0 + x_0z_2 + x_0z_5 + x_1z_6 + x_1z_7 + x_1 + x_2z_5 + x_2z_6 \\
&\quad + x_3z_1 + x_3z_5 + x_3z_6 + x_3 + x_4z_0 + x_4z_4 + x_4z_5 + x_5z_1 \\
&\quad + x_5z_3 + x_5z_6 + x_5z_7 + x_6z_0 + x_6z_1 + x_6z_2 + x_6z_4 + x_6z_5 \\
&\quad + x_6 + x_7z_0 + x_7z_3 + x_7z_6 + x_7z_7 \\
0 &= x_0z_1 + x_0z_3 + x_0z_6 + x_1z_0 + x_1z_1 + x_1z_2 + x_1z_4 + x_1z_5 \\
&\quad + x_1z_6 + x_2z_0 + x_2z_3 + x_2z_5 + x_2z_6 + x_2z_7 + x_2 + x_3z_2 \\
&\quad + x_3z_4 + x_3z_5 + x_3z_6 + x_3z_7 + x_4z_3 + x_4z_5 + x_4 + x_5z_1
\end{aligned}
$$

$$
\begin{aligned}
& +x_5z_2 + x_5z_6 + x_6z_0 + x_6z_1 + x_6z_5 + x_6 + x_7z_0 + x_7z_1 \\
& +x_7z_6 + x_7z_7 + x_7 \\
0 \;=\; & x_0z_2 + x_0z_4 + x_0z_7 + x_1z_1 + x_1z_3 + x_1z_6 + x_2z_0 + x_2z_1 \\
& +x_2z_2 + x_2z_4 + x_2z_5 + x_2z_6 + x_3z_0 + x_3z_3 + x_3z_5 + x_3z_6 \\
& +x_3z_7 + x_3 + x_4z_2 + x_4z_4 + x_4z_5 + x_4z_6 + x_4z_7 + x_5z_3 \\
& +x_5z_5 + x_5 + x_6z_1 + x_6z_2 + x_6z_6 + x_7z_0 + x_7z_1 + x_7z_5 + x_7 \\
0 \;=\; & x_0z_0 + x_0z_3 + x_0z_5 + x_1z_2 + x_1z_4 + x_1z_7 + x_2z_1 + x_2z_3 \\
& +x_2z_6 + x_3z_0 + x_3z_1 + x_3z_2 + x_3z_4 + x_3z_5 + x_3z_6 + x_4z_0 \\
& +x_4z_3 + x_4z_5 + x_4z_6 + x_4z_7 + x_4 + x_5z_2 + x_5z_4 + x_5z_5 \\
& +x_5z_6 + x_5z_7 + x_6z_3 + x_6z_5 + x_6 + x_7z_1 + x_7z_2 + x_7z_6 \\
0 \;=\; & x_0z_1 + x_0z_4 + x_0z_6 + x_1z_0 + x_1z_3 + x_1z_5 + x_2z_2 + x_2z_4 \\
& +x_2z_7 + x_3z_1 + x_3z_3 + x_3z_6 + x_4z_0 + x_4z_1 + x_4z_2 + x_4z_4 \\
& +x_4z_5 + x_4z_6 + x_5z_0 + x_5z_3 + x_5z_5 + x_5z_6 + x_5z_7 + x_5 \\
& +x_6z_2 + x_6z_4 + x_6z_5 + x_6z_6 + x_6z_7 + x_7z_3 + x_7z_5 + x_7
\end{aligned}
\tag{2.9}
$$

Остання система (2.9) ідентична до системи (2.7). Сім із восьми рівняннь цієї системи є нулями, та це справедливо з ймовірністю 1. Восьме рівняння є справедливим тільки при $x \neq 0$, отже, з ймовірністю 255/256. Крім того для $\forall x \neq 0 \;\; x = x^2y$, проте воно акож справедлво при $x = 0$, тому має місце запис:

$$
\forall\, x \in GF(2^8) \begin{cases} x &= yx^2 \\ x^2 &= y^2x^4 \\ & \vdots \\ x^{128} &= y^{128}x^{256} = y^{128}x \end{cases}
\tag{2.10}
$$

Візьмемо останнє рівняння із системи (2.10) та запишемо два симетричних рівняння для створення додаткового набору з 16 рівнянь

для S-блок блочного шифру $Rijndael$.

$$\begin{cases} x^{128} &= y^{128}x \\ y^{128} &= x^{128}y \end{cases} \tag{2.11}$$

Використаємо перше рівняння та (2.8) та додамо їх у лістинг (2.4):

```
E7 = x^128 + y^128 * x
eqs7 = [_.substitute(eqs4) for _ in E7.list()]
latex(eqs7)
```

У результаті отримуємо список рівнянь:

$$\begin{aligned}
0 &= x_0z_0 + x_0z_1 + x_0z_2 + x_0z_6 + x_1z_1 + x_1z_3 + x_1z_5 + x_1z_7 \\
&\quad + x_2z_3 + x_2z_4 + x_2z_5 + x_2z_6 + x_2z_7 + x_3z_2 + x_3z_3 + x_3z_5 \\
&\quad + x_3z_6 + x_3 + x_4z_0 + x_4z_1 + x_4z_3 + x_4z_4 + x_5z_0 + x_5z_1 \\
&\quad + x_5z_2 + x_5z_3 + x_5z_5 + x_5z_6 + x_5z_7 + x_5 + x_6z_0 + x_6z_3 \\
&\quad + x_6z_5 + x_6z_7 + x_7z_1 + x_7z_3 + x_7z_5 + x_7z_6 \\
0 &= x_0z_1 + x_0z_2 + x_0z_3 + x_0z_4 + x_0z_5 + x_0z_6 + x_0z_7 + x_0 \\
&\quad + x_1z_0 + x_1z_2 + x_1z_3 + x_1z_5 + x_1z_6 + x_1z_7 + x_2z_1 + x_2z_4 \\
&\quad + x_2z_6 + x_2 + x_3z_2 + x_3z_4 + x_3z_7 + x_3 + x_4z_0 + x_4z_1 \\
&\quad + x_4z_2 + x_4z_4 + x_4z_5 + x_4z_6 + x_5z_2 + x_5z_4 + x_5z_5 + x_5z_6 \\
&\quad + x_5z_7 + x_6z_1 + x_6z_2 + x_6z_6 + x_7z_0 + x_7z_1 + x_7z_6 + x_7z_7 + x_7 \\
0 &= x_0z_0 + x_0z_1 + x_0z_3 + x_0z_4 + x_0z_5 + x_0z_6 + x_0z_7 + x_1z_1 \\
&\quad + x_1z_2 + x_1z_3 + x_1z_4 + x_1z_5 + x_1z_6 + x_1z_7 + x_1 + x_2z_0 \\
&\quad + x_2z_2 + x_2z_3 + x_2z_5 + x_2z_6 + x_2z_7 + x_2 + x_3z_1 + x_3z_4 \\
&\quad + x_3z_6 + x_3 + x_4z_2 + x_4z_4 + x_4z_7 + x_4 + x_5z_0 + x_5z_1 \\
&\quad + x_5z_2 + x_5z_4 + x_5z_5 + x_5z_6 + x_5 + x_6z_2 + x_6z_4 + x_6z_5 \\
&\quad + x_6z_6 + x_6z_7 + x_7z_1 + x_7z_2 + x_7z_6
\end{aligned}$$

$$
\begin{aligned}
0 &= x_0z_0 + x_0z_2 + x_0z_6 + x_1z_0 + x_1z_4 + x_1z_6 + x_1 + x_2z_1 + x_2z_2 \\
&\quad + x_2 + x_3z_0 + x_3z_7 + x_4z_0 + x_4z_3 + x_4z_6 + x_5z_0 + x_5z_1 + x_5z_3 \\
&\quad + x_5z_4 + x_5z_5 + x_5z_6 + x_6z_1 + x_6z_2 + x_6z_3 + x_6z_4 + x_6z_6 \\
&\quad + x_6z_7 + x_6 + x_7z_1 + x_7z_2 + x_7z_3 + x_7z_4 + x_7z_7 + x_7 \\
0 &= x_0z_0 + x_0z_1 + x_0z_3 + x_0z_4 + x_1z_0 + x_1z_1 + x_1z_2 + x_1z_3 \\
&\quad + x_1z_5 + x_1z_6 + x_1z_7 + x_1 + x_2z_0 + x_2z_3 + x_2z_5 + x_2z_7 \\
&\quad + x_3z_1 + x_3z_3 + x_3z_5 + x_3z_6 + x_3 + x_4z_1 + x_4z_3 + x_4z_4 \\
&\quad + x_4z_7 + x_4 + x_5z_1 + x_5z_2 + x_5z_5 + x_5z_7 + x_6z_1 + x_6z_4 \\
&\quad + x_6z_6 + x_6z_7 + x_7z_2 + x_7z_4 + x_7z_5 + x_7z_7 \\
0 &= x_0z_2 + x_0z_3 + x_0z_5 + x_0z_6 + x_1z_0 + x_1z_1 + x_1z_3 + x_1z_4 \\
&\quad + x_1 + x_2z_0 + x_2z_1 + x_2z_2 + x_2z_3 + x_2z_5 + x_2z_6 + x_2z_7 \\
&\quad + x_3z_0 + x_3z_3 + x_3z_5 + x_3z_7 + x_3 + x_4z_1 + x_4z_3 + x_4z_5 \\
&\quad + x_4z_6 + x_4 + x_5z_1 + x_5z_3 + x_5z_4 + x_5z_7 + x_5 + x_6z_1 \\
&\quad + x_6z_2 + x_6z_5 + x_6z_7 + x_7z_1 + x_7z_4 + x_7z_6 + x_7z_7 \\
0 &= x_0z_3 + x_0z_4 + x_0z_5 + x_0z_6 + x_0z_7 + x_1z_2 + x_1z_3 + x_1z_5 \\
&\quad + x_1z_6 + x_1 + x_2z_0 + x_2z_1 + x_2z_3 + x_2z_4 + x_3z_0 + x_3z_1 \\
&\quad + x_3z_2 + x_3z_3 + x_3z_5 + x_3z_6 + x_3z_7 + x_3 + x_4z_0 + x_4z_3 \\
&\quad + x_4z_5 + x_4z_7 + x_5z_1 + x_5z_3 + x_5z_5 + x_5z_6 + x_6z_1 + x_6z_3 \\
&\quad + x_6z_4 + x_6z_7 + x_6 + x_7z_1 + x_7z_2 + x_7z_5 + x_7z_7 \\
0 &= x_0z_1 + x_0z_3 + x_0z_5 + x_0z_7 + x_1z_3 + x_1z_4 + x_1z_5 + x_1z_6 \\
&\quad + x_1z_7 + x_1 + x_2z_2 + x_2z_3 + x_2z_5 + x_2z_6 + x_3z_0 + x_3z_1 \\
&\quad + x_3z_3 + x_3z_4 + x_3 + x_4z_0 + x_4z_1 + x_4z_2 + x_4z_3 + x_4z_5 \\
&\quad + x_4z_6 + x_4z_7 + x_5z_0 + x_5z_3 + x_5z_5 + x_5z_7 + x_5 + x_6z_1 \\
&\quad + x_6z_3 + x_6z_5 + x_6z_6 + x_6 + x_7z_1 + x_7z_3 + x_7z_4 + x_7z_7
\end{aligned}
\tag{2.12}
$$

Тепер зробимо те саме для іншого рівняння:

```
E8 = y^128 + x^128 * y
```

```
eqs8 = [_.substitute(eqs4) for _ in E8.list()]
latex(eqs8)
```

Для того, щоб отримати інший набор із вісьми рівнянь для S-блок блочного шифру *Rijndael*. Отримуємо:

$$
\begin{aligned}
0 &= x_0z_2 + x_0z_5 + x_0z_7 + x_0 + x_1z_1 + x_1z_2 + x_1z_5 + x_1z_6 \\
&\quad + x_1z_7 + x_1 + x_2z_1 + x_2z_4 + x_2z_6 + x_3z_0 + x_3z_5 + x_4z_0 \\
&\quad + x_4z_3 + x_4z_5 + x_5z_4 + x_5z_7 + x_6z_2 + x_6z_4 + x_6z_7 + x_7z_1 \\
&\quad + x_7z_3 + x_7z_4 + x_7 + z_0 + z_1 + z_2 + z_6 + 1 \\
0 &= x_0z_0 + x_0z_3 + x_0z_6 + x_1z_0 + x_1z_1 + x_1z_3 + x_1z_5 + x_1 \\
&\quad + x_2z_1 + x_2z_2 + x_2z_4 + x_2z_5 + x_2z_6 + x_2z_7 + x_2 + x_3z_0 \\
&\quad + x_3z_1 + x_3z_2 + x_3z_6 + x_3z_7 + x_3 + x_4z_0 + x_4z_1 + x_4z_3 \\
&\quad + x_4z_4 + x_4z_5 + x_4z_6 + x_5z_0 + x_5z_4 + x_5z_5 + x_5z_7 + x_6z_0 \\
&\quad + x_6z_2 + x_6z_3 + x_6z_4 + x_6z_5 + x_6z_7 + x_7z_1 + x_7z_3 + x_7z_7 \\
&\quad + x_7 + z_1 + z_2 + z_3 + z_4 + z_5 + z_6 + z_7 + 1 \\
0 &= x_0z_1 + x_0z_4 + x_0z_7 + x_0 + x_1z_1 + x_1z_4 + x_1z_5 + x_1z_6 \\
&\quad + x_1z_7 + x_1 + x_2z_0 + x_2z_3 + x_2z_6 + x_3z_0 + x_3z_1 + x_3z_3 \\
&\quad + x_3z_5 + x_3 + x_4z_1 + x_4z_2 + x_4z_4 + x_4z_5 + x_4z_6 + x_4z_7 \\
&\quad + x_4 + x_5z_0 + x_5z_1 + x_5z_2 + x_5z_6 + x_5z_7 + x_5 + x_6z_0 \\
&\quad + x_6z_1 + x_6z_3 + x_6z_4 + x_6z_5 + x_6z_6 + x_7z_0 + x_7z_4 + x_7z_5 \\
&\quad + x_7z_7 + z_0 + z_1 + z_3 + z_4 + z_5 + z_6 + z_7 \\
0 &= x_0z_0 + x_0z_2 + x_0z_5 + x_1z_0 + x_1z_1 + x_1z_2 + x_1z_5 + x_1z_7 \\
&\quad + x_1 + x_2z_6 + x_2z_7 + x_2 + x_3z_0 + x_3z_1 + x_3z_4 + x_3z_6 \\
&\quad + x_3z_7 + x_3 + x_4z_5 + x_4z_6 + x_5z_0 + x_5z_1 + x_5z_3 + x_5z_4 \\
&\quad + x_5z_5 + x_5z_7 + x_5 + x_6z_1 + x_6z_5 + x_6z_6 + x_6 + x_7z_0 \\
&\quad + x_7z_2 + x_7z_3 + x_7z_4 + x_7z_6 + x_7z_7 + z_0 + z_2 + z_6
\end{aligned}
$$

$$
\begin{aligned}
0 &= x_0z_1 + x_0z_3 + x_0z_6 + x_1z_0 + x_1z_2 + x_1z_3 + x_1 + x_2z_0 \\
&\quad + x_2z_1 + x_2z_2 + x_2z_4 + x_2z_5 + x_2z_6 + x_3z_1 + x_3z_2 + x_3z_7 \\
&\quad + x_3 + x_4z_0 + x_4z_3 + x_4z_5 + x_4z_6 + x_4z_7 + x_4 + x_5z_0 \\
&\quad + x_5z_1 + x_5z_6 + x_5 + x_6z_2 + x_6z_4 + x_6z_5 + x_6z_6 + x_6z_7 \\
&\quad + x_7z_0 + x_7z_5 + x_7z_7 + z_0 + z_1 + z_3 + z_4 \\
0 &= x_0z_2 + x_0z_4 + x_0z_7 + x_1z_1 + x_1z_3 + x_1z_4 + x_1 + x_2z_1 \\
&\quad + x_2z_3 + x_2z_6 + x_3z_0 + x_3z_2 + x_3z_3 + x_3 + x_4z_0 + x_4z_1 \\
&\quad + x_4z_2 + x_4z_4 + x_4z_5 + x_4z_6 + x_5z_1 + x_5z_2 + x_5z_7 + x_5 \\
&\quad + x_6z_0 + x_6z_3 + x_6z_5 + x_6z_6 + x_6z_7 + x_6 + x_7z_0 + x_7z_1 \\
&\quad + x_7z_6 + x_7 + z_2 + z_3 + z_5 + z_6 \\
0 &= x_0z_0 + x_0z_3 + x_0z_5 + x_1z_4 + x_1z_7 + x_2z_2 + x_2z_4 + x_2z_7 \\
&\quad + x_3z_1 + x_3z_3 + x_3z_4 + x_3 + x_4z_1 + x_4z_3 + x_4z_6 + x_5z_0 \\
&\quad + x_5z_2 + x_5z_3 + x_5 + x_6z_0 + x_6z_1 + x_6z_2 + x_6z_4 + x_6z_5 \\
&\quad + x_6z_6 + x_7z_1 + x_7z_2 + x_7z_7 + x_7 + z_3 + z_4 + z_5 + z_6 + z_7 \\
0 &= x_0z_1 + x_0z_4 + x_0z_6 + x_1z_0 + x_1z_5 + x_2z_0 + x_2z_3 + x_2z_5 \\
&\quad + x_3z_4 + x_3z_7 + x_4z_2 + x_4z_4 + x_4z_7 + x_5z_1 + x_5z_3 + x_5z_4 \\
&\quad + x_5 + x_6z_1 + x_6z_3 + x_6z_6 + x_7z_0 + x_7z_2 + x_7z_3 + x_7 \\
&\quad + z_1 + z_3 + z_5 + z_7
\end{aligned}
\tag{2.13}
$$

Далі підрахуємо за допомогою лістингу (2.5) кількість рівнянь у S-блоці блочного шифру *Rijndael* та кількість мономів у цих рівняннях, а також максимальну, мінімальну та загальну кількість різних мономів (докладніше усі кроки цього перетворення викладені у роботі [21]).

Лістинг 2.5 – Створення першої системи рівнянь S-блоку блочного шифру *Rijndael*

```
1. mq1 = eqs5[1:] + eqs7 + eqs8
2. len(mq1)
```

```
3. lmon1 = [len(_.monomials()) for _ in mq1]
4. min(lmon1)
5. max(lmon1)
6. len(set(flatten([_.monomials() for _ in mq1])))
```

Як описано вище, ймовірність справедливості першого рівняння 255/256, тому воно відкидається, тому отримуємо 23 рівняння, що складаються з від 28 до 49 мономів, та в загалом із 81 унікальних мономів. Пошук 256 рішень для цієї системи рівнянь представляють у виді значень таблиці S-блоку блочного шифру *Rijndael*, він здійснен у лістингу (2.6).

Лістинг 2.6 – Рішення системи рівнянь

```
1. mq2 = eqs7 + eqs8
2. len(mq2)
3. lmon2 = [len(_.monomials()) for _ in mq2]
4. min(lmon2)
5. max(lmon2)
6. len(set(flatten([_.monomials() for _ in mq2])))
```

На виході отримали 16 лінійно незалежних, що описують S-блок блочного шифру *Rijndael*, мають від 28 до 49 мономів, а у загалом 81 унікальних мономів.

Висновки до розділу 2

У цьому розділі проаналізовано алгебраїчну атаку на спрощену версію потокового шифру SNOW2.0, а також розібрана побудова системи

рівнянь найменшого степеня, що описує S-блок блочного шифру $Rijndael$, що дозволяє видиліти основні етапи атаки та аспекти, які треба описати при розширенні її дії на двійкових SNOW2.0-подібні потокові шифри.

3 РОЗРОБКА АЛГОРИТМУ ПОБУДОВИ АЛГЕБРАЇЧНОІ АТАКИ НА ДВІЙКОВІ SNOW2.0-ПОДІБНІ ПОТОКОВІ ШИФРИ

У цьому розділі запропоновані алгортими для побудови алгебраїчноі атаки на двійкові SNOW2.0-подібні потокові шифри, а також практична реалізація її частини, що безпосередньо відрізняє двійкові SNOW2.0-подібні потокові шифри від спрощеної версії потокового шифру SNOW2.0, побудову системи рівнянь найменшого степеня, що описує функцію між регістрами пам'яті, представлену випадковим S-блоком.

3.1 Етапи побудови алгебраїчноі атаки на двійкові SNOW2.0-подібні потокові шифри

Розглядаючи алгебраїчну атаку на спрощену версію потокового шифру SNOW2.0, описану у розділі 2.1, її можно умовно поділити два этапа:

1) побудувати систему рівнянь, що описує нелінійну бієкції між регістрами пам'яті(фукцію, що базується на деякому S-блоці);
2) скласти систему рівняннь між символами гами, вихідної послідовності, та початковим заповненням регістрів.

Після виконяння обох цих етапів початковий стан шифру буде встановлено однозначно, а тому, алгебраїчна атака буде здійснена.

У розділах 3.2 та 3.4 запропоновані способи для проведення обох етапів наведеної алгебраїчної атаки.

3.2 Перший етап: алгоритм пошуку системи рівнянь найменшого степеня, що описує залежність між регістрами пам'яті

На першому етапі із схожості побудови двійкових SNOW2.0-подібних потокових шифрів, рисунок 1.3, та спрощеної версії потокового шифру SNOW2.0, рисунок 1.2, варто розглянути саме той аспект, де вони відрізняються, а саме у нелінійній бієкції між регістрами пам'яті $\mathcal{S}$ та $\mathcal{N}$. У спрощеній версії потокового шифру SNOW2.0 функція $\mathcal{S}$ базується на S-блоці блочного шифру $Rijndael$, побудову системи рівнянь найменшого степеня для нього будо розглянуто у розділі 2.

Алгоритм 3.1 — Пошук системи рівнянь найменшого степеня, що описує вхідний S-блок.

Вхід: S-блок

1 визначити координатні функції S-блоку;

2 скласти за допомогою виходу із S-блоку та координатних функцій S-блоку систему, що буде описувати цей S-блок;

3 побудувати ідеал цієї системи;

4 скласти базис Грьобнера цього ідеалу;

5 визначити алгебраїчну імунність S-блоку;

6 за допомогою алгебраїчної імунності знайти рівняння із найменшими степенями;

Вихід: система рівнянь найменшого степеня, що описує S-блок.

У двійкових SNOW2.0-подібних потокових шифрах функція $\mathcal{N}$ (1.1) може базуватися на будь-якому S-блоці, а тому потребує алгоритму побудови системи рівнянь, що буде його описувати. Варто зауважити, що

ця система повина бути мінімального степеня, бо інакше на її вирішення на великих S-блоках буде витрачено занадто багато часу.

Алгоритм 3.1 дозволяє, використуючи на абстрактну алгебру, створити таку систему для будь-якого S-блоку.

Хоча цей алгоритм описан, та він надає змогу зменшити кількість рівнянь у системі, що описує S-блок, проте важко судити про його ефективність без програмної реалізації, приклад якої представлен та розібран у розділі 3.3.

3.3 Приклад реалізації алгортиму пошуку системи рівнянь найменшого степеня, що описує випадковий S-блок

Для реалізації алгоритму будемо використовувати середовище $CoCalc(SageMathCloud)$ [17], що є онлайн-сервісом для запуску онлайн-розрахунків $SageMath$, а також офіційною документацією $SageTutorial$ [19].

Розглянемо випадковий S-блок, зазначений у вигляді таблиці перестановок на рисунку 3.1.

Його координатні функцій у векторній формі приймають вигляд:

$$f_0 = (1000010101111001),$$

$$f_1 = (0101110001100011),$$

$$f_2 = (1110100000110101),$$

$$f_3 = (0110010110110010);$$

та у вигляді поліному Жегалкіна:

$$f_0 = 1 \oplus x_3 \oplus x_2 \oplus x_2\ x_3 \oplus x_1 \oplus x_1\ x_2 \oplus x_1\ x_2\ x_3 \oplus x_0 \oplus x_0\ x_1\ x_2,$$

$$f_1 = x_3 \oplus x_1 \oplus x_1\ x_3 \oplus x_1\ x_2 \oplus x_0\ x_2 \oplus x_0\ x_1 \oplus x_0\ x_1\ x_2,$$

$$f_2 = 1 \oplus x_2\ x_3 \oplus x_1\ x_3 \oplus x_1\ x_2 \oplus x_0 \oplus x_0\ x_2 \oplus x_0\ x_2\ x_3,$$

$$f_3 = x_3 \oplus x_2 \oplus x_1\ x_2 \oplus x_0 \oplus x_0\ x_2 \oplus x_0\ x_2\ x_3 \oplus x_0\ x_1 \oplus x_0\ x_1\ x_3.$$

	x0	x1	x2	x3
0x	22	13	03	10
1x	12	31	00	21
2x	01	30	33	23
3x	20	02	11	32

	x00	x01	x10	x11
00x	1010	0111	0011	0100
01x	0110	1101	0000	1001
10x	0001	1100	1111	1011
11x	1000	0010	0101	1110

Рисунок 3.1 – Таблиця перестановок випадкового S-блоку

Позначимо вихідні біти із S-блоку, як $y_0,\ y_1,\ y_2,\ y_4$:

$$(\ y_0,\ y_1,\ y_2,\ y_3\ )\ =\ S\ [\ (\ x_0,\ x_1,\ x_2,\ x_3\ )\ ].$$

Лістинг 3.1 – Складання системи S-блоку

```
1. nb = 4
2. varl=[c+str(p) for c in 'xy' for p in range (nb)]
3. R = BooleanPolynomialRing(names = varl,
                             order='degrevlex')
4. R.inject_variables()
5. f0 = 1 + x3 + x2 + x2~x3 + x1 + x1~x2 + x1~x2~x3 +
       + x0 + x0~x1~x2
6. f1 = x3 + x1 + x1~x3 + x1~x2 + x0~x2 + x0~x1 +
       + x0~x1~x2
7. f2 = 1 + x2~x3 + x1~x3 + x1~x2 + x0 + x0~x2 +
       + x0~x2~x3
```

```
8. f3 = x3 + x2 + x1~x2 + x0 + x0~x2 + x0~x2~x3 +
       + x0~x1 + x0~x1~x3
9. B = [y0 + f0, y1 + f1, y2 + f2, y3 + f3,
   x0~x0 + x0, x1~x1 + x1, x2~x2 + x2, x3~x3 + x3,
   y0~y0 + y0, y1~y1 + y1, y2~y2 + y2, y3~y3 + y3]
```

У першому рядку лістингу 3.1 зазначається кількість вхідних та вихідних змінних. У другому рядку створюється поле над $GF(2)$ рангу, зазначеному у першому рядку, по кожній змінній. Створення поля, із змінними x_0, x_1, x_2, x_4, y_0, y_1, y_2, y_4, що ініціалізуються у четвертому рядку, продемонстровано на рисунку 3.2.

```
Boolean PolynomialRing in x0, x1, x2, x3, y0, y1, y2, y3
```

Рисунок 3.2 – Створення поля

Рядки 5-8 містять координатні функції заданого вище S-блоку. У рядку 9 задається система, що описує S-блок.

Далі побудуємо ідеал цієї системи та базис Грьобнера цього ідеалу, як у лістингу 3.2.

Лістинг 3.2 – Обчислення ідеалу та базису Грьобнера

```
1. I = Ideal(B)
2. latex(I)

3. GB = I.groebner_basis()
4. latex(GB)
```

У першому рядку формується ідеал системи, у другому рядку він

виводиться у виді документу LaTeX:

$x_0x_1x_2 + x_1x_2x_3 + x_1x_2 + x_2x_3 + x_0 + x_1 + x_2 + x_3 + y_0 + 1,$

$x_0x_1x_2 + x_0x_1 + x_0x_2 + x_1x_2 + x_1x_3 + x_1 + x_3 + y_1,$

$x_0x_2x_3 + x_0x_2 + x_1x_2 + x_1x_3 + x_2x_3 + x_0 + y_2 + 1,$

$x_0x_1x_3 + x_0x_2x_3 + x_0x_1 + x_0x_2 + x_1x_2 + x_0 + x_2 + x_3 + y_3.$

У третьому рядку будується базис Грьобнера ідеалу системи, у четвертому рядку він виводиться у виді документу LaTeX:

$y_0y_1y_3 + x_3y_3 + y_1y_3 + x_0 + x_1 + x_2 + y_1 + y_3,$

$y_0y_2y_3 + y_0y_2 + x_3y_3 + y_1y_3 + y_2y_3 + x_3 + y_1 + y_2,$

$x_0x_1 + y_0y_1 + y_0y_2 + x_3y_3 + y_2y_3 + x_0 + x_2 + y_0 + y_2 + 1,$

$x_0x_2 + x_3y_3 + y_0y_3 + y_2y_3 + x_0 + x_1 + x_3 + y_0 + 1,$

$x_1x_2 + y_0y_1 + y_0y_2 + y_0y_3 + y_1y_3 + y_2y_3 + x_1 + x_2 + y_3 + 1,$

$x_0x_3 + y_0y_1 + x_3y_3 + y_2y_3 + x_1 + y_1 + y_2 + y_3 + 1,$

$x_1x_3 + y_0y_2 + y_0y_3 + y_2y_3 + x_0 + y_0 + y_3,$

$x_2x_3 + y_0y_1 + y_0y_3 + y_1y_3 + x_0 + y_0 + y_1 + y_2 + y_3,$

$x_0y_0 + y_0y_2 + x_2y_3 + x_3y_3 + y_1y_3 + x_2 + x_3 + y_1 + y_3 + 1,$

$x_1y_0 + y_0y_1 + y_0y_2 + x_3y_3 + y_1y_3 + x_0 + x_1 + x_2 + y_0 + y_1 + y_3,$

$x_2y_0 + y_0y_2 + x_3y_3 + y_0y_3 + x_0 + x_1 + y_0 + y_1 + y_2 + 1,$

$x_3y_0 + y_0y_1 + y_0y_2 + x_2y_3 + y_1y_3 + x_0 + x_1 + x_2 + x_3 + y_2 + y_3,$

$x_0y_1 + y_0y_1 + x_3y_3 + y_0y_3 + y_1y_3,$

$x_1y_1 + x_3y_3 + y_0y_3 + y_1y_3 + x_0 + y_0 + y_2 + y_3,$

$x_2y_1 + y_0y_1 + x_3y_3 + y_0y_3 + x_0 + y_0 + y_1 + y_2 + y_3,$

$x_3y_1 + x_2y_3 + x_3y_3 + y_1y_3 + x_1 + x_3 + y_1 + y_2 + y_3 + 1,$

$x_0y_2 + y_0y_2 + y_2y_3 + x_0 + y_0 + y_3,$

$x_1y_2 + x_2y_3 + y_0y_3 + x_2 + y_0 + y_3 + 1,$

$x_2y_2 + y_0y_2 + x_3y_3 + y_0y_3 + y_1y_3 + y_2y_3 + x_2 + y_0 + y_2 + y_3 + 1,$

$x_3y_2 + y_0y_2 + x_3y_3 + y_0y_3 + y_2y_3 + x_0 + y_0 + y_3,$

$y_1y_2 + x_3y_3 + y_0y_3 + x_0 + y_0 + y_2 + y_3,$

$x_0y_3 + y_0y_3 + y_2y_3 + y_3,$

$x_1y_3 + x_2y_3 + y_0y_3 + y_1y_3 + y_2y_3 + x_0 + x_1 + x_2 + y_1 + y_3$

У лістингу 3.3 проходимо по всіх рівняннях, що входят до базису Грьобнера.

Лістинг 3.3 – Обчислення алгебраїчної імунності

```
1. for i in range (len(GB)):
2.     if (GB[i].degree() < AI):
3.         AI = GB[i].degree()
4.         N_min = i
```

У випадку виявлення рівняння, степень якого менша за алгебраїчну імунність у даний момент часу, алгебраїчна імунність набуває значення, рівному степені цього рівняння, а також запам'ятовуємо номер цього поліному. За допомогою лістингу 3.3 ми знаходимо номер рівняння із найменшим степенем.

Лістинг 3.4 – Пошук рівняння із найменшим степенем

```
1. for i in range (len(GB)):
2.     if (GB[i].degree() == AI):
3.         Q_min = Q_min+1
4.         P.append(GB[i])
5. latex(P)
```

Далі знайдемо ті рівняння, степень яких співпадає із значенням алгебраїчної імунності, а також підрахуємо кількість цих рівнянь, це продемонстровано у лістингу 3.4.

За допомогою останнього рядка у лістингу 3.4 виводимо результати роботи у виді документу LaTeX.

На виході отримуємо:

2

23

2

21

$1 : x_0x_1 + y_0y_1 + y_0y_2 + x_3y_3 + y_2y_3 + x_0 + x_2 + y_0 + y_2 + 1,$

$2 : x_0x_2 + x_3y_3 + y_0y_3 + y_2y_3 + x_0 + x_1 + x_3 + y_0 + 1,$

$3 : x_1x_2 + y_0y_1 + y_0y_2 + y_0y_3 + y_1y_3 + y_2y_3 + x_1 + x_2 + y_3 + 1,$

$4 : x_0x_3 + y_0y_1 + x_3y_3 + y_2y_3 + x_1 + y_1 + y_2 + y_3 + 1,$

$5 : x_1x_3 + y_0y_2 + y_0y_3 + y_2y_3 + x_0 + y_0 + y_3,$

$6 : x_2x_3 + y_0y_1 + y_0y_3 + y_1y_3 + x_0 + y_0 + y_1 + y_2 + y_3,$

$7 : x_0y_0 + y_0y_2 + x_2y_3 + x_3y_3 + y_1y_3 + x_2 + x_3 + y_1 + y_3 + 1,$

$8 : x_1y_0 + y_0y_1 + y_0y_2 + x_3y_3 + y_1y_3 + x_0 + x_1 + x_2 + y_0 + y_1 + y_3,$

$9 : x_2y_0 + y_0y_2 + x_3y_3 + y_0y_3 + x_0 + x_1 + y_0 + y_1 + y_2 + 1,$

$10 : x_3y_0 + y_0y_1 + y_0y_2 + x_2y_3 + y_1y_3 + x_0 + x_1 + x_2 + x_3 + y_2 + y_3,$

$11 : x_0y_1 + y_0y_1 + x_3y_3 + y_0y_3 + y_1y_3,$

$12 : x_1y_1 + x_3y_3 + y_0y_3 + y_1y_3 + x_0 + y_0 + y_2 + y_3,$

$13 : x_2y_1 + y_0y_1 + x_3y_3 + y_0y_3 + x_0 + y_0 + y_1 + y_2 + y_3,$

$14 : x_3y_1 + x_2y_3 + x_3y_3 + y_1y_3 + x_1 + x_3 + y_1 + y_2 + y_3 + 1,$

$15 : x_0y_2 + y_0y_2 + y_2y_3 + x_0 + y_0 + y_3,$

$16 : x_1y_2 + x_2y_3 + y_0y_3 + x_2 + y_0 + y_3 + 1,$

$17 : x_2y_2 + y_0y_2 + x_3y_3 + y_0y_3 + y_1y_3 + y_2y_3 + x_2 + y_0 + y_2 + y_3 + 1,$

$18 : x_3y_2 + y_0y_2 + x_3y_3 + y_0y_3 + y_2y_3 + x_0 + y_0 + y_3,$

$19 : y_1y_2 + x_3y_3 + y_0y_3 + x_0 + y_0 + y_2 + y_3,$

$20 : x_0y_3 + y_0y_3 + y_2y_3 + y_3,$

$21 : x_1y_3 + x_2y_3 + y_0y_3 + y_1y_3 + y_2y_3 + x_0 + x_1 + x_2 + y_1 + y_3.$

Це значить, що алгебраїчна імунність заданого S-блоку становить 2, у базису Грьобнера знаходиться 23 рівняння, номер першого моному із мінімальним степенем, що дорівнює 2(нумерація починається з нуля), а також кількість та самі рівняння, що входять до системи рівнянь найменшого степеня, яка описує заданий S-блок.

Скористаємося спроможністтю $SageMath$ вирішувати системи рівнянь та отримаємо однозначне співставлення виходів до входів у

S-блок:

$$
\begin{aligned}
(\,0,\,0,\,0,\,0\,) &= S\,[\,(\,0,\,1,\,1,\,0\,)\,]\\
(\,1,\,0,\,0,\,0\,) &= S\,[\,(\,1,\,1,\,0,\,0\,)\,]\\
(\,0,\,1,\,0,\,0\,) &= S\,[\,(\,0,\,0,\,1,\,1\,)\,]\\
(\,1,\,1,\,0,\,0\,) &= S\,[\,(\,1,\,0,\,0,\,1\,)\,]\\
(\,0,\,0,\,1,\,0\,) &= S\,[\,(\,1,\,1,\,0,\,1\,)\,]\\
(\,1,\,0,\,1,\,0\,) &= S\,[\,(\,0,\,0,\,0,\,0\,)\,]\\
(\,0,\,1,\,1,\,0\,) &= S\,[\,(\,0,\,1,\,0,\,0\,)\,]\\
(\,1,\,1,\,1,\,0\,) &= S\,[\,(\,1,\,1,\,1,\,1\,)\,]\\
(\,0,\,0,\,0,\,1\,) &= S\,[\,(\,1,\,0,\,0,\,0\,)\,]\\
(\,1,\,0,\,0,\,1\,) &= S\,[\,(\,0,\,1,\,1,\,1\,)\,]\\
(\,0,\,1,\,0,\,1\,) &= S\,[\,(\,1,\,1,\,1,\,0\,)\,]\\
(\,1,\,1,\,0,\,1\,) &= S\,[\,(\,0,\,1,\,0,\,1\,)\,]\\
(\,0,\,0,\,1,\,1\,) &= S\,[\,(\,0,\,0,\,1,\,0\,)\,]\\
(\,1,\,0,\,1,\,1\,) &= S\,[\,(\,1,\,0,\,1,\,1\,)\,]\\
(\,0,\,1,\,1,\,1\,) &= S\,[\,(\,0,\,0,\,0,\,1\,)\,]\\
(\,1,\,1,\,1,\,1\,) &= S\,[\,(\,1,\,0,\,1,\,0\,)\,]
\end{aligned}
$$

За допомогою цієї програми, що була продемонстрована та розібрана у цьому розділі та яка у повному обсязі представлена у додатку А.2, можна побудувати та розв'язати систему найменшого степеня, що буде описувати заданий S-блок.

$SageMath$ дозволяє також замірити витрачений час на розв'язування системи рівнянь, що допомогає оцінити ефективність побудови системи

рівняньнайменшого степеня, відносно інших відомих систем.

Лістинг 3.5 – Порівняння часу рішення різних систем рівнянь

```
1. 1000 loops, best of 3: 216 ms per loop
2. 1000 loops, best of 3: 317 ms per loop
3. 1000 loops, best of 3: 488 ms per loop
4. 1000 loops, best of 3: 617 ms per loop
```

У лістингу 3.5 зображено результати вимірів часу, який *SageMath* витратив на рішення систем рівнянь, вирішення кожної системи проводилось 1000 разів. У першому рядку зазначено час пошуку рішення у системи рівнянь найменшого степеня, у другому - базису Грьобнера, у третьому - ідеалу та у четвертому - початкової системи рівнянь. Із отриманих результатів можна зробити висновок, що на знаходження рішення системи витрачається найменьший час, а тому це доказує ефективність її побудови.

3.4 Другий етап: спосіб відновлення початкового заповнення регістрів двійкового SNOW2.0-подібного потокового шифру

За допомогою відношення $z^0 = s^0 \oplus r_1^0 \oplus s^{15} \oplus r_2^0$ можна однозначно відновити початковий стан регістру r_1, знаючи параметри $s_0, ..., s_{15}, r_2^0$, а тому їх можна вважати початковим заповненням, яке треба відновити на цьому, другому, етапі.

Скористаємось такими рекурентними рівняннями, що виходять із структури двійкового SNOW2.0-подібного потокового шифру,

зображеного на рисунку 1.3:

$$z^t = s^t \oplus r_1^t \oplus s^{t+15} \oplus r_2^t, \tag{3.1}$$

$$r_1^t = r_2^{t-1} \oplus s^{t+4}. \tag{3.2}$$

Піставимо (3.2) у (3.1) рекурентно для таких $t = 1..i,\ i \geq 1$ та виразимо відносно r_2^t, отримуємо:

$$\begin{aligned} r_2^i &= z^i \oplus r_2^{i-1} \oplus s^i \oplus s^{i+4} \oplus s^{i+15}, \\ r_2^{i-1} &= z^{i-1} \oplus r_2^{i-2} \oplus s^{i-1} \oplus s^{i+3} \oplus s^{i+14}, \\ &\dots \\ r_2^1 &= z^1 \oplus r_2^0 \oplus s^0 \oplus s^4 \oplus s^{15}. \end{aligned}$$

Після складання цих рівностей можна побачити, що r_2^i залежить від i знаків гами, які є відомими, r_2^0 та деякого набору $s^t,\ t = 1..(i+15)$, кожен элемент якого можна виразити через $s_0, ..., s_{15}$, а тому маємо змогу представити r_2^i, як деяку афінну функцію, що залежить від $s_0, ..., s_{15}, r_2^0$. Аналогічно із рівняння (3.2) r_1^t можна також представити, як деяку іншу афінну функцію від $s_0, ..., s_{15}, r_2^0$. Тому знадобиться принаймні 17 таких рівнянь та 17 знаків гами.

Повертаючись до рівняння (1.1), підставимо це представлення регістрів у вигляді афінних функції замість них самих та побудуємо за алгоритмом 3.1 систему найменшого степеня. В отриманій системі маємо 544 невідомих над $GF(2)$(17 станів $s_0, ..., s_{15}, r_2^0$, кожен з яких є 32-бітним). Отримана система буде складатися із $17 * \texttt{Q\_min}$ рівнянь того самого найменшого степеня, тому що у підстановці були задіяні лише афінні функції, де `Q_min`, як у розділі 3.3, є кількістю рівнянь у системі рівнянь найменшого степеня, що описує нелінійну функцію між регістрами пам'яті.

Вирішення такої системи можливо методом лінеарізації. Як вже було

зазначено, маємо $544(32 * 16 + 32 = 2^5(2^4 + 1))$ невідомих. Нехай AI, як зазначено у розділі 3.3, визначає степінь системи, яка побудована за алгоритмом 3.1, тоді маємо $N = \sum_{i=0}^{AI}\binom{544}{i}$ змінних після лінеарізації, трудоємність рішення лінеарізованої системи:

$$\mathcal{O}(N^3) = \mathcal{O}(\ (\ \sum_{i=0}^{AI}\binom{544}{i}\ )^3\ ) = \mathcal{O}(\ (\ \binom{544}{AI}\ )^3\ ) \approx \mathcal{O}(\ (\ \frac{544^{AI}}{AI!})^3\ ) \approx$$

$$\approx \mathcal{O}(\ (\ \frac{2^{9*AI}}{AI!}\ )^3\ ) = \mathcal{O}(\ \frac{2^{27*AI}}{AI!}\ ).$$

Зазнчимо, що отримана складність значно менша за повний перебор, який дорівнює 2^{544}, а тому зазначену алгебрїчну атаку можна вважати досить успішною.

Висновки до розділу 3

У цьому розділі було запропоноване розширення відомої алгебраїчної атаки на спрощену версію потокового шифру SNOW2.0 на двійкові SNOW2.0-подібні потокові шифрів, яке у силу їх структури полягає у розробці алгоритму побудови системи рівнянь найменшого степеня, що описує нелінійну фунцію між регістрами пам'яті, яка може базуватися на будь-якому S-блоці, та у способі створення системи рівнянь, рішення якої надасть змогу відновити початковий стан двійкового SNOW2.0-подібного потокового шифру.

Також для алгоритму побудови системи рівнянь найменшого степеня представлена практична реалізація за допомогою $SageMath$, яка продемострувала доцільність використання даного алгоритму, а саме швидкість її вирішення.

ВИСНОВКИ

У результаті цієї роботи були проаналізовані та стисло описані джерела за тематикою дослідження, а саме шифр SNOW2.0, спрощена версія шифру SNOW2.0 та S-блок блочного шифру $Rijndael$, на якому базується нелінійна бієкція між регістрами пам'яті $\mathcal{S}$, також були наведені основні та необхідні означення із комутативної алгебри у термінах булевих функцій. Також було проаналізовано алгебраїчну атаку на спрощену версію потокового шифру SNOW2.0 та розібрана побудова системи рівнянь найменшого степеня, відносно цієї атаки, а саме S-блоку блочного шифру $Rijndael$.

Запропоноване розширення відомої алгебраїчної атаки на спрощену версію потокового шифру SNOW2.0 на двійкові SNOW2.0-подібні потокові шифрів, яке у силу їх структури полягає у розробці алгоритму побудови системи рівнянь найменшого степеня, що описує нелінійну фунцію між регістрами пам'яті, яка може базуватися на будь-якому S-блоці, та у способі створення системи рівнянь, рішення якої надасть змогу відновити початковий стан двійкового SNOW2.0-подібного потокового шифру.

Зокрема представлена практична реалізація алгоритму побудови системи рівнянь найменшого степеня за допомогою $SageMath$, яка продемострувала доцільність використання даного алгоритму, а саме швидкість її вирішення.

У подальшій роботі планується розробити стійкий до цієї алгебраїчної атаки двійковий SNOW2.0-подібний потоковий шифр.

ПЕРЕЛІК ПОСИЛАНЬ

1. Андрушкевич А.В. Іваненко Д.В. Луценко М.С. Кухар Ю.В. Алгоритм потокового криптоперетворення «Струмок» // Труды научно-технической конференции с международным участием «Компьютерное моделирование в наукоемких технологиях», 26-31 мая 2016 г. – Х.: ХНУ имени В.Н. Каразина. — 2016. — С. 187–190.

2. Ekdahl P. Johansson T Maximov A. Yang J. A new SNOW stream cipher called SNOW-V [Електронний ресурс]. — 2018. — Режим доступу: https://eprint.iacr.org/2018/1143.pdf.

3. Ekdahl P. Johansson T. A New Version of the Stream Cipher SNOW [Електронний ресурс] // Springer, Heidelberg. — 2003. — С. 47–61. — Режим доступу: https://link.springer.com/chapter/10.1007/3-540-36492-7_5.

4. Billet O. Gilbert H. Resistance of SNOW 2.0 Against Algebraic Attacks [Електронний ресурс] // Lecture Notes in Computer Science, vol 3376. Springer, Berlin, Heidelberg. — 2005. — Режим доступу: https://link.springer.com/chapter/10.1007/978-3-540-30574-3_3.

5. Daemen J. Rijmen V. The design of Rijndael [Електронний ресурс] // Springer Verlag Series on Information Security and Cryptography, Springer Verlag. — 2002. — Режим доступу: https://link.springer.com/book/10.1007/978-3-662-04722-4.

6. Dworkin M. Advanced Encryption Standard (AES) [Електронний ресурс]. — 2001. — Режим доступу: https://doi.org/10.6028/NIST.FIPS.197.

7. Martin V. Algebraic attack on stream ciphers // Bratislava. — 2007. — Режим доступу: http://www.dcs.fmph.uniba.sk/diplomovky/obhajene/getfile.php/master-mv.pdf?id=132&fid=219&type=application%2Fpdf.

8. Nyberg K. Perfect nonlinear S-boxes [Електронний ресурс] // EUROCRYPT 1991. Lecture Notes in Computer Science, vol 547. Springer,

Berlin, Heidelberg. — 1991. — Режим доступу: https://link.springer.com/chapter/10.1007/3-540-46416-6_32.

9. Courtois N. T. Pieprzyk J. Cryptanalysis of block ciphers with overdefined systems of equations [Електронний ресурс] // ASIACRYPT 2002, volume 2501 of Lecture Notes in Computer Science. Springer-Verlag. — 2002. — С. 267–287. — Режим доступу: https://eprint.iacr.org/2002/044.pdf.

10. Cox D. Little J. O'Shea D. Ideals, Varieties, and Algorithms // Springer-Verlag New York. — 2007. — Режим доступу: https://doi.org/10.1007/978-0-387-35651-8.

11. Ars G. Faugère J.-C. Algebraic immunities of functions over finite fields [Електронний ресурс] // Technical report, INRIA. — 2005. — Режим доступу: https://hal.inria.fr/inria-00070475/document.

12. Biryukov A. de Canniere C. Block ciphers and systems of quadratic equations [Електронний ресурс] // Fast Software Encryption. – FSE'03, Proceedings. – Springer-Verlag. — 2003. — С. 274 – 289. — Режим доступу: https://www.esat.kuleuven.be/cosic/publications/article-14.pdf.

13. . Carlet C. On the algebraic immunities and higher order nonlinearities of vectorial Boolean functions [Електронний ресурс] // Workshoop ACPTECC, Veliko Tavrono, Bulgaria. — 2009. — С. 104 – 116. — Режим доступу: http://ebooks.iospress.nl/volumearticle/23788.

14. М. Олексійчук А. Алгебраїчна імунність векторних булевих функцій та булеві базиси Грьобнера [рукопис].

15. Armknecht F. Krause M. Constructions single- and multi-output Boolean functions with maximal algebraic immunity [Електронний ресурс] // Automata, Languages and Programming, Proceedings. – LNCS 4052. – Springer-Verlag. — 2006. — С. 180 – 191. — Режим доступу: https://link.springer.com/chapter/10.1007/11787006_16.

16. Faug'ere. J.-C. A New Efficient Algorithm for Computing Groebner Bases without Reduction to Zero (F5) [Електронний ресурс] //

In Proceedings of ISSAC. — 2002. — С. 75–83. — Режим доступу: https://www3.risc.jku.at/research/theorema/Groebner-Bases-Bibliography/gbbib_files/publication_502.pdf.

17. CoCalc (SageMathCloud) [Електронний ресурс]. — Режим доступу: https://cocalc.com/.

18. SageMath [Електронний ресурс]. — Режим доступу: http://www.sagemath.org/.

19. Sage Tutorial Release 8.7 [Електронний ресурс] / The Sage Development Team. — 2019. — Режим доступу: http://doc.sagemath.org/pdf/en/tutorial/SageTutorial.pdf.

20. LaTeX [Електронний ресурс]. — Режим доступу: https://www.latex-project.org/.

21. Cui J. Hong Zh.. Generation and Optimization of Rijndael S-box Equation System [Електронний ресурс] // Information Technology Journal, 13. — 2014. — С. 2482–2488. — Режим доступу: https://www.researchgate.net/publication/287519749_Generation_and_Optimization_of_Rijndael_S-box_Equation_System.

ДОДАТОК А ТЕКСТИ ПРОГРАМ

Цей додаток містить повні коди програм, що реалізують створення мінімальної системи рівнянь, що повністю описують заданий S-блок.

А.1 Програма 1

Ця програма містить повний код, що розглядається у розділі 2, який реалізує створення мінімальної системи рівнянь, що повністю описують S-блок блокового шифру *Rijndael* [4].

```
1. nb = 8
2. varl=[c+ str(p) for c in 'xyz' for p in range (nb)]
3. B = BooleanPolynomialRing(names = varl)
4. B.inject_variables()

5. P.<p> = PolynomialRing (B)
6. Byte.<t> = P.quotient_ring(p^8 + p^4 + p^3 + p + 1)
7. X = B.gens()[:nb]
8. Y = B.gens()[nb:2*nb]
9. x = sum([X[j]*t^j for j in range (nb)])
10. y = sum([Y[j]*t^j for j in range (nb)])

11. E3 = x*y
12. eqs3 = E3.list()
13. latex(eqs3)

14. Baff.<u> = P.quotient_ring (p^8 + 1)
15. Z = B.gens()[2*nb:][:nb]
```

```
16. z = sum ([Z[ j ]*u^j for j in range (nb)])
17. a = u^4 + u^3 + u^2 + u + 1
18. b = u^6 + u^5 + u + 1
19. eqs4 = dict(zip(Y,( a^255*(z + b)).list()))

20. eqs5 = [_.substitute(eqs4) for _ in eqs3]

21. E7 = x^128 + y^128 * x
22. eqs7 = [_.substitute(eqs4) for _ in E7.list()]
23. latex(eqs7)

24. E8 = y^128 + x^128 * y
25. eqs8 = [_.substitute(eqs4) for _ in E8.list()]
26. latex(eqs8)

27. mq1 = eqs5[1:] + eqs7 + eqs8
28. len(mq1)
29. lmon1 = [len(_.monomials()) for _ in mq1]
30. min(lmon1)
31. max(lmon1)
32. len(set(flatten([_.monomials() for _ in mq1])))

33. mq2 = eqs7 + eqs8
34. len(mq2)
35. lmon2 = [len(_.monomials()) for _ in mq2]
36. min(lmon2)
37. max(lmon2)
38. len(set(flatten([_.monomials() for _ in mq2])))
```

A.2 Програма 2

Ця програма містить повний код, що розглядається у розділі 3, який реалізує створення мінімальної системи рівнянь, що повністю описують заданий на рисунку 3.1 S-блок.

```
1. nb = 4
2. varl=[c+str(p) for c in 'xy' for p in range (nb)]
3. R = BooleanPolynomialRing(names = varl,
                             order='degrevlex')
4. R.inject_variables()
5. f0 = 1 + x3 + x2 + x2~x3 + x1 + x1~x2 + x1~x2~x3 +
       + x0 + x0~x1~x2
6. f1 = x3 + x1 + x1~x3 + x1~x2 + x0~x2 + x0~x1 +
       + x0~x1~x2
7. f2 = 1 + x2~x3 + x1~x3 + x1~x2 + x0 + x0~x2 +
       + x0~x2~x3
8. f3 = x3 + x2 + x1~x2 + x0 + x0~x2 + x0~x2~x3 +
       + x0~x1 + x0~x1~x3
9. B = [y0 + f0, y1 + f1, y2 + f2, y3 + f3,
   x0~x0 + x0, x1~x1 + x1, x2~x2 + x2, x3~x3 + x3,
   y0~y0 + y0, y1~y1 + y1, y2~y2 + y2, y3~y3 + y3]
10. I = Ideal(B)
11. latex(I)
12. GB = I.groebner_basis()
13. latex(GB)
14. AI = nb*2
```

```
15. N_min = -1
16. Q_min = 0
17. P = []

18. for i in range (len(GB)):
19.     if (GB[i].degree() < AI):
20.         AI = GB[i].degree()
21.         N_min = i

22. for i in range (len(GB)):
23.     if (GB[i].degree() == AI):
24.         Q_min = Q_min+1
25.         P.append(GB[i])

26. AI
27. len(GB)
28. N_min
29. Q_min
30. latex(P)

31. J=ideal([f for f in P])
32. latex(J)
33. J.variety()

34. TimeGB=ideal([f for f in GB])
35. TimeGB.variety()
36. TimeB=ideal([f for f in B])

37. from sage.misc.sage_timeit import SageTimeitResult
38. timeit("J.variety()", number = 1000)
39. timeit("TimeGB.variety()", number = 1000)
40. timeit("I.variety()", number = 1000)
41. timeit("TimeB.variety()", number = 1000)
```