Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Радуш Володимир Вячеславович,
студент групи РЗ-171

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Метод синтезу високоякісних S-блоків на основі функцій багатозначної логіки

Спеціальність:
125 Кібербезпека

Спеціалізація, освітня програма:
Кібербезпека

Керівник:
Соколов Артем Вікторович,
к.т.н., доцент

Одеса – 2022

Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Рівень вищої освіти другий (магістерський)
Спеціальність - 125 Кібербезпека
Освітня програма - Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри КБПЗ

_____
д.т.н.,проф. А.А.Кобозєва
____ _____202_р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Радушу Володимиру Вячеславовичу

1.Тема роботи: *Метод синтезу високоякісних S-блоків на основі функцій багатозначної логіки*,
керівник роботи *Соколов Артем Вікторович, к. ф.-м. н., доцент,* затверджені наказом ректора Національного університету «Одеська політехніка» від „____”_____ 20___ р. №_____ .
2.Зміст роботи*: аналіз проблемної області, постановка задачі, аналіз недоліків сучасних методів синтезу якісних S-блоків, розробка методу синтезу високоякісних S-блоків, оцінка криптографічних характеристик синтезованих блоків, оцінка якості роботи криптоалгоритму AES з новим синтезованим блоком*,
3. Перелік ілюстративного матеріалу: *скріншоти коду програми, графіки, слайди презентації*.

4. Консультанти розділів роботи

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

5. Дата видачі завдання "_____"_____2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	*Аналіз джерел з теми випускної кваліфікаційної роботи*	*19.09.2022*	*виконано*
2	*Розробка методу синтезу високоякісних S-блоків*	*24.10.2022*	*виконано*
3	*Розробка програмного забезпечення*	*14.11.2022*	*виконано*
4	*Підготовка тексту роботи*	*28.11.2022*	*виконано*
5	*Підготовка презентації та доповіді*	*30.11.2022*	*виконано*
7	*Попередній захист*	*02.12.2022*	*виконано*
8	*Нормоконтроль, рецензування*	*16.12.2022*	*виконано*
9	*Перевірка на плагіат*	*19.12.2022*	*виконано*

Здобувач вищої освіти _____ *Радуш В.В*

Керівник роботи _____ *Соколов А.В.*

АНОТАЦІЯ

Кваліфікаційна робота на тему «Метод синтезу високоякісних S-блоків на основі функцій багатозначної» на здобуття другого (магістерського) рівня вищої освіти за спеціальністю 125 Кібербезпека, спеціалізація, освітня програма: Кібербезпека, містить 15 рисунків, 12 таблиць, 1 додаток, 31 літературних джерел за переліком посилань. Робота виконана на 63 сторінках загального тексту і 45 сторінках основного тексту.

Метою роботи є створення методу синтезу високоякісних S-блоків великих довжин, що задовольняють суворому лавинному критерію компонентних 2- та 4-функцій та мають кореляційний імунітет у сенсі булевої логіки.

У цій роботі пропонується в якості матеріалу для побудови таких S-блоків використовуються S-блоки невеликої довжини, які задовольняють суворому лавинному критерію компонентних 4-функцій. На їх основі за допомогою отримання якісних 4-функцій формується множина блоків, що одночасно відповідають СЛК 2- та 4-логіки, а також мають кореляційний імунітет у сенсі булевої логіки. Синтезовані блоки показали кращі результати при роботі у складі криптоалгоритмів, ніж ті, що синтезовані за допомогою конструкцією Ніберг. Ці S-блоки можна рекомендувати для практичного використання для модернізації існуючих симетричних криптографічних алгоритмів, а також для побудови нових перспективних шифрів.

СИМЕТРИЧНИЙ БЛОКОВИЙ-ШИФР, S-БЛОК, БАГАТОЗНАЧНА ЛОГІКА, СУВОРИЙ ЛАВИННИЙ КРИТЕРІЙ, КОРЕЛЯЦІЙНИЙ ІМУНІТЕТ, ВІДСТАНЬ НЕЛІНІЙНОСТІ.

ANNOTATION

Qualification work entitled as «The method of synthesis of high-quality S-blocks based on multivalued functions» to obtain the second (master's) level of higher education in the specialty 125 Cybersecurity, specialization, educational program: Cybersecurity, has 15 pictures, 23 tables, 2 appendices, 42 literature sources according to the list of references. The work is performed on 63 pages of general text and 45 pages of main text.

The purpose of the work is to create a method for the synthesis of large lengths high quality S-boxes that satisfy the strict avalanche criterion of component 2- and 4-functions and have correlation immunity in the sense of Boolean logic.

As a raw material for the construction of such an S-boxes, small-length S-boxes are used that satisfy the strict avalanche criterion of component 4-functions. On their basis, using obtained qualitative 4-functions, a set of blocks is formed and simultaneously correspond to the SAC of 2- and 4-logic, and also have correlation immunity in the sense of Boolean logic. Synthesized S-boxes showed better results than S-boxes, that represented by Niberg construction. These S-boxes can be recommended for practical use to modernize existing symmetric cryptographic algorithms, as well as to build new perspective ciphers.

SYMMETRIC BLOCK CIPHER, S-BOX, MANY-VALUED LOGIC, STRICT AVALANCHE CRITERION, CRITERION OF MAXIMUM AVALANCHE EFFECT.

ЗМІСТ

ВСТУП

Результати досліджень, наведених у роботі, опубліковані в журналах «Інформатика та математичні методи в моделюванні» [1] та «Journal of Discrete Mathematical Sciences & Cryptography» [2].

У наш час фактично неможливо побудувати систему управління інформаційною безпекою без використання блокових симетричних шифрів. Варто зазначити, що еталонним шифром у цій галузі є криптоалгоритм AES, який відповідно є найпоширенішим. Більш того він реалізований на апаратному рівні фактично в кожному сучасному персональному комп'ютері/ноутбуці. Це обумовлено в першу чергу тим, що AES досить просто реалізується як на програмному, так і на апаратному рівні, а також має досить високу криптостійкість, незважаючи на те, що був введений більше 20 років тому [3].

Через тенденції до надшвидкого розвитку інформаційних технологій, а відповідно і методів криптоаналізу та криптоатак стає зрозуміло, що класичної структури цього криптоалгоритму може виявитися недостатньо, а тому необхідно своєчасно покращити її, насамперед звернувши увагу на криптографічні примітиви, що входять до складу AES.

Найбільшого поширення, а відповідно й практичного застосування набув так званий блок підстановки, також відомий як S-блок. Даний примітив я складовою частиною фактичного будь-якого блочного симетричного криптоалгоритму, адже фактично саме вони зумовлюють криптографічну якість всього алгоритму (загальний рівень конфузії і дифузіїї) [4], а також те, наскільки простою є задача технічною реалізації того чи іншого алгоритму.

На даний момент існує досить багато методів, що дозволяють синтезувати високоякісні S-блоки. Насамперед, це конструкція Ніберг, на основі якої був синтезований S-блок криптоалгоритму AES [5], а також все більшої популярності починає здобувати метод побудови S-блоків за допомогою динамічного хаосу [6].

На даний момент для оцінки якості будь-якого блоку підстановки використовується низка наступних критеріїв [7, 8]:

1. Висока відстань нелінійності.

2. Висока алгебраїчна степінь нелінійності.

3. Відповідність суворому лавинному критерію (СЛК).

4. Статистична незалежність виходу S-блоку підстановки від його входу (кореляційний імунітет).

Усі ці критерії формувалися відповідно до саме двійкового уявлення S-блоку у вигляді набору булевих функцій і на даний момент існує низка робіт, що присвячені саме методам оцінки даних критеріїв для S-блоку криптоалгориму AES [9, 10].

Проте, варто пам’ятати, що криптоаналітик не обмежений в використанні лише апарату булевих функцій, якими описано сучасні криптоалгоритми і може також використати апарат функцій багатозначної логіки, в якому, цілком ймовірно дані конструкції будуть більш вразливими [11].

Метою цієї роботи є підвищення ефективності сучасних шифрів шляхом розробки методу синтезу S-блоків, що відповідають суворому лавинному критерію, та критерію кореляційного імунітету при їх представленні функціями багатозначної логіки.

Завдання дослідження:

- провести аналітичний огляд та дослідження існуючих конструкцій S-блоків (зокрема, заснованих на методах теорії динамічного хаосу) на відповідність критеріям криптографічної якості при їх представленні за допомогою функцій багатозначної логіки;
- розробити метод синтезу S-блоків, що відповідають суворому лавинному критерію компонентних булевих та 4-функцій, відповідають критерію кореляційного імунітету при представленні булевими функціями, а також прийнятну відстань нелінійності;
- провести емпіричні дослідження роботи побудованих S-блоків у складі криптоалгоритму AES.

Об’єкт дослідження — процеси підвищення ефективності сучасних криптографічних алгоритмів.

Предмет дослідження — методи синтезу кореляційно імунних S-блоків, що відповідають критеріям криптографічної якості як у сенсі представлення булевими функціями, так і у сенсі представлення функціями багатозначної логіки.

Очевидно, що такий клас S-блоків має більшу практичну цінність, так як має стійкість одразу в двох представленнях, більш того у булевому представленні такий блок відповідає фактично 3 критеріям з 4 можливих одночасно. Такі особливості значно знижуються шанси криптоаналітика на успішну атаку, а також дозволяють оптимізувати роботу криптоалгоритму з точки зору швидкості роботи програми.

Зрозуміло, що такі S-блоки є більш практичними, так як мають стійкість одразу в двох своїх представленнях. Така особливість знижує можливості криптоаналітика для атаки, а також оптимізує роботу криптоалгоритму з точки зору швидкості роботи.

1 КРИТЕРІЇ КРИПТОГРАФІЧНОЇ ЯКОСТІ. МЕТОДИ СИНТЕЗУ ЯКІСНИХ S-БЛОКІВ

1.1 Суворий лавинний критерій: основні визначення та принцип оцінки

Лавинний ефект (англ. Avalanche effect) – криптографічне поняття, що використовується в криптографії відносно до блокових шифрів та хеш-функцій. Головна суть – забезпечити суттєві («лавинні») зміни у вихідній послідовності бітів при зміні навіть незначної кількості бітів у вхідній послідовності, тобто забезпечити залежність вихідних бітів від вхідних [12].

Відповідно, суворий лавинний критерій (СЛК) і визначає лавинні властивості функції q-значної логіки, а також S-блоку до складу якого вони входять. Варто зазначити, що q-значна логіка відповідає не лише булевому представленню, а усім можливим представленням степені 2, тобто 2, 4, 16 і т.д, так як кожне з цих представлень легко декомпозується в усі попередні.

Стає зрозуміло, що якість S-блоку є сумарною оцінкою усього набору функцій q-значної логіки, що входять до його складу. Таким чином, для дослідження якості S-блоку, представленого у вигляді набору функцій q-значної логіки, необхідно використати більш широкий математичний апарат.

Надамо визначення основних понять. Для початку розглянемо q-функцію k змінних *f(x)*, а також вектор $u = (u_1, u_2, ..., u_k)$.

Визначення 1 [13]. Функцією q-значної логіки k змінних називається відображення $\{0,1,2,...,q-1\}^k \to \{0,1,2,...,q-1\}$

Визначення 2 [13]. Вагою $\underline{\omega}(u)$ q-значного вектору назвемо кількість його ненульових компонент.

Визначення 3 [13]. Похідною функції f у напрямку вектору u назвемо функцію

$$D_u f(x) = f(x \underset{q}{\oplus} u) - f(x) (\mathrm{mod}\, q) \ , \qquad (1.1)$$

де $\underset{q}{\oplus}$ означає додавання по модулю q.

Визначення 4 [13]. Функція *q*-значної логіки *f(x)* відповідає критерію поширення щодо вектора $u \in V_k$ – КП(*u*), якщо її похідна по напрямку *u* є збалансованою функцією, тобто значення $0,1,2,...,q-1$ приймаються з рівними ймовірностями: $p(D_u f(x) = i(\mathrm{mod}\, q)) = 1/q$ для усіх $i = 0,1,2,...,q-1$. Інакше кажучи, $K^0 = K^1 = ... = K^{q-1}$, де K^i кількість наборів значень змінних, на які похідна приймає значення *i*.

Визначення 5 [13]. Функція *q*-значної логіки *f(x)* відповідає критерію поширення степеню *k* – КП(*k*), якщо вона відповідає критерію поширення КП(*u*) щодо всіх векторів *u* ваги $1 \le \underline{\omega}(u) \le k$.

Визначення 6 [13]. Функція *q*-значної логіки *f(x)* задовольняє суворому лавинному критерію (СЛК), якщо вона відповідає критерію поширення степеню 1 – КП (1).

Очевидно, що наведені визначення є універсальними стосовно будь-якого представлення S-блоку, включаючи двійкову логіку [14]. Однак, визначення СЛК є досить жорсткою вимогою, тому досить часто використовується визначення критерію максимального лавинного ефекту.

Визначення 7 [14]. Булева функція відповідаю критерію максимального лавинного критерію (МЛК), якщо усі її похідні по всім напрямкам одиничної ваги $u \in V_k \mid wt(u) = 1$ мають вагу, що дорівнює як мінімум половині її довжини $wt(D_u f(x)) = N/2$, тобто:

$$p\{f(x) = f(x \oplus u)\} \ge 0,5 \;\; \forall u \in V_k, \;\; wt(u) = 1 \qquad (1.2)$$

1.2 Відстань нелінійності

Поняття нелініності прийнято для показника нелінійності, що використовує поняття ваги Хемінга та відстані Хемінга.

Визначення 8 [15] Вагою Хемінга чи просто вагою двійкового вектора називається число одиниць серед його компонент. Вага Хемінга булевої функції є вагою вектора її значень. Позначати вагу векотру чи функції будемо *wt(x)* та *wt(f)*.

Визначення 9 [15] Відстань Хемінга між двома функціями f та g є вага функції $f \oplus g$. Іншими словами – це число тих x $\oplus$ V_n , на яких $f(x) \neq g(x)$.

На практиці найчастіше всього використовують знаходження нелінійності за допомогою перетворення Уолша-Адамара. Перетворення Уолша-Адамара має наступний вигляд:

$$W_F(\nu) = \sum_{u \in Z_2^k} (-1)^{\langle u,\nu \rangle \oplus F(u)}, \tag{1.3}$$

Де F – булева функція, k – кількість змінних, Z_2^k – множина двійкових векторів довжини k.

Однак, більшої популярності здобуло матричне перетворення:

$$W_B(\omega) = BA, \omega = 0,1,...,2^{k-1}, \tag{1.4}$$

де A – матриця Уолша-Адамара порядку N.

Сама ж матриця Адамара H_N порядку N задається за допомогою конструкції Сильвестра:

$$H_{2^k} = \begin{bmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{bmatrix}, H_1 = 1, \tag{1.5}$$

Тоді відстань Хемінга дорівнює наступній величині

$$N_f = 2^{k-1} - \frac{1}{2} \max_{\nu \in Z_2^k} |W_F(\nu)|, \tag{1.6}$$

У випадку 4-логіки варто розглянути визначення афінної 4-функції, що має наступний вигляд:

$$f(x_0,\ldots,x_{k-1}) = b + a_0x_0 + a_1x_1 + \ldots + a_{k-1}x_{k-1} (\mathrm{mod}\, 4) = \sum_{i=0}^{k-1} a_i x_i (\mathrm{mod}\, 4) + b, \tag{1.7}$$

де $a_i, b \in \{0,1,2,3\}$.

Наприклад, для випадку k=2 ми можемо отримати усі афінні функції, що не є лінійною комбінацією x точки зору сумування зі значенням 1, 2 чи 3 за модулем 4, у результаті чого отримуємо такі функції:

$$\begin{array}{ll}\varphi_1 = 0; & \{0000\} \\ \varphi_5 = x_1; & \{0123\} \\ \varphi_9 = 2x_1; & \{0202\} \\ \varphi_{13} = 3x_1; & \{0321\}\end{array} \tag{1.8}$$

При переході в експоненційну форму наступним чином

$$\left\{e^{j\frac{2\pi}{4}\cdot 0} \quad e^{j\frac{2\pi}{4}\cdot 1} \quad e^{j\frac{2\pi}{4}\cdot 2} e^{j\frac{2\pi}{4}\cdot 3}\right\} \to \left\{e^{j0^o} \quad e^{j90^o} \quad e^{j180^o} e^{j270^o}\right\} \to \\ \to \{z_0 \quad z_1 \quad z_2 \quad z_3\}, \tag{1.9}$$

отримуємо матрицю Віленкіна-Крістенсона

$$V_4 = \begin{bmatrix} z_0 & z_0 & z_0 & z_0 \\ z_0 & z_1 & z_2 & z_3 \\ z_0 & z_2 & z_0 & z_2 \\ z_0 & z_3 & z_2 & z_1 \end{bmatrix} = \begin{bmatrix} e^{j\frac{2\pi}{4}\cdot 0} & e^{j\frac{2\pi}{4}\cdot 0} & e^{j\frac{2\pi}{4}\cdot 0} & e^{j\frac{2\pi}{4}\cdot 0} \\ e^{j\frac{2\pi}{4}\cdot 0} & e^{j\frac{2\pi}{4}\cdot 1} & e^{j\frac{2\pi}{4}\cdot 2} & e^{j\frac{2\pi}{4}\cdot 3} \\ e^{j\frac{2\pi}{4}\cdot 0} & e^{j\frac{2\pi}{4}\cdot 2} & e^{j\frac{2\pi}{4}\cdot 0} & e^{j\frac{2\pi}{4}\cdot 2} \\ e^{j\frac{2\pi}{4}\cdot 0} & e^{j\frac{2\pi}{4}\cdot 3} & e^{j\frac{2\pi}{4}\cdot 2} & e^{j\frac{2\pi}{4}\cdot 1} \end{bmatrix} \tag{1.10}$$

Більш того, ми можемо синтезувати таку матрицю для будь-якого заданого порядку $N = 4^k$

$$V_{4^{k+1}} = \begin{bmatrix} V_{4^k} & V_{4^k} & V_{4^k} & V_{4^k} \\ V_{4^k} & V_{4^k}+1 & V_{4^k}+2 & V_{4^k}+3 \\ V_{4^k} & V_{4^k}+2 & V_{4^k} & V_{4^k}+2 \\ V_{4^k} & V_{4^k}+3 & V_{4^k}+2 & V_{4^k}+1 \end{bmatrix} \tag{1.11}$$

Де «+» – операція складання за модулем 4, а матриці представлені у символьній формі, тобто підсумовування відбувається відносно індексів z_i. Таким чином, коефіцієнти перетворення Віленкіна-Крістенсона ми отримуємо наступним чином:

$$\Omega_A = A \cdot \overline{V}_N \tag{1.12}$$

Де А – матриця Віленкіна-Крістенсона, $\overline{V}_N$ – 4-функція у вигляді своєї таблиці істинності. Після чого коефіцієнт нелінійності знаходиться наступним чином:

$$q^k - \max\left\{\left|\Omega_A\right|\right\}, \qquad (1.13)$$

де k – кількість змінних, q – основа багатозначної системи числення (наприклад, 4, 16 тощо)

1.3 Кореляційний імунітет

Як вже зазначалося вище ще одним критерієм криптографічної якості є кореляційний імунітет, що забезпечує статистичну незалежність. Наприклад, представимо булеву функцію схематичним чином (рисунок 1.1).

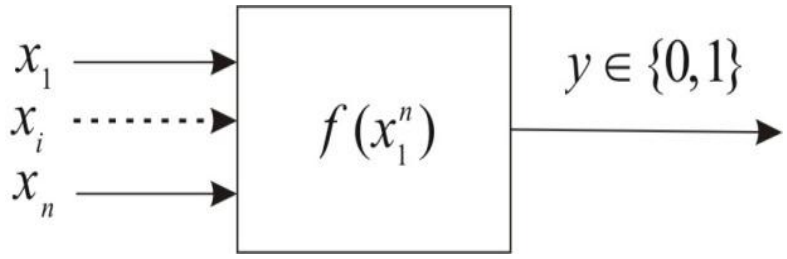

Рисунок 1.1 - схематичне представлення булевої функції

Визначення 10 [16] – булева функція *f(x)*, $x \in V_n$, називається кореляційно імунною порядку m, $1 \le m \le n$, якщо вихід $y=f(x_1^n)$ та будь-яка множина з m її вхідних змінних є статистично незалежними.

Визначення 11 [16] – підфункцією булевої функції *f(x)*, $x \in V_n$, називається функція f', що отримується шляхом підстановки у f деяких констант «0» чи «1» замість деяких змінних. Якщо підставимо до функції f константи $\sigma_{i_1},...,\sigma_{i_s}$ замість змінних $x_{i_1},...,x_{i_s}$, відповідно, то отримана підфункція позначається $f^{\sigma_{i_1},...,\sigma_{i_s}}_{x_{i_1},...,x_{i_s}}$. Якщо замість змінної x_i константа не підставлена, то x_i називається вільною змінною.

Наприклад, нехай булева функція $f(x_1,x_2,x_3)$, тоді її підфункціями будуть $f'(x_1,x_2,0)$, $f'(x_1,x_2,1)$, $f'(x_1,0,0)$, $f'(x_1,0,1)$ і т.д.

Визначення 12 [16] – Булева функція *f(x)*, $x \in V_n$, називається кореляційно-імунною порядку *m*, $1 \le m \le n$, якщо вага $wt(f') = wt(f)/2^m$, для будь-якої її підфункції f' від *n* - *m* змінних.

Визначення 13 [16] – Врівноважена кореляційно-імунна функція порядку *m* називається *m-стійкою*. Іншими словами, булева функція f називається *m-стійкою*, якщо вага $wt(f') = 2^{n-m-1}$, для будь-якої її підфункції f' від *n* - *m* змінних.

Визначення 14 [16] – Булева функція $f \in F_n$ має кореляційний імунітет порядку m, КІ(*m*), якщо її коефіцієнти перетворення Уолша-Адамара задовольняють умові $W_f(\omega) = 0$, для всіх $\omega \in V_n$, таких що вага $1 \le wt(\omega) \le m$.

Визначення 15 [16] – Булева функція $f \in F_n$ є кореляційно-ефективною, якщо не менш за половину її спектральних коефіцієнтів дорівнюють нулю, тобто $W_f(\omega) = 0$ для не менш ніж за половину $\omega \in V_n$.

Властивість 1 [16] – Кореляційно-імунна функція порядку *m* є також кореляційно-імунною будь-якого меншого порядку.

Що ж стосується визначення кореляційного імунітету стосовно функцій багатозначної логіки, то необхідно розглянути послідовність над алфавітом $\{0,1,...,q-1\}$

$$f_1 \in \{0,1,...q-1\},\ i = 0,1,...,N-1, \qquad (1.14)$$

Варто зазначити, що елементи даної послідовності можуть бути представлені в експоненційному представленні шляхом однозначного перетворення:

$$0 \leftrightarrow z_0 = e^{j\frac{2\pi}{q}0}, 1 \leftrightarrow z_1 = e^{j\frac{2\pi}{q}1}, ..., q-1 \leftrightarrow z_{q-1} = e^{j\frac{2\pi}{q}(q-1)}, \qquad (1.15)$$

Для даної послідовності ми можемо ввести вектор $K = \{K_0, K_1, ..., K_{q-1}\}$, де коефіцієнти K_u характеризують кількість появ символу $u \in \{0,1,...q-1\}$ у послідовності f.

Визначення 16 [17]. Розбалансом послідовності f назвемо значення модуля суми поелементних множень елементів вектора K на відповідні їм елементи експоненційного алфавіту $\{z_0, z_1, ..., z_{q-1}\}$

$$\Delta(f) = | K_0 z_0 + K_1 z_1 + ... + K_{q-1} z_{q-1} |. \quad (1.15)$$

Визначення 17 [17]. Говорять, що вихід q-функції не залежить від групи її вхідних змінних $\{x_i\}, i = 1,...m$ якщо при подстановці замість цих змінних будь-яких констант $\sigma_{i_1}, ..., \sigma_{i_s} \in \{0,1,..., q-1\}$, розбаланс отриманих таким чином підфункцій $\Delta_{f'} = \frac{\Delta_f}{q^m}$.

Визначення 18 [17]. Максимальним відхиленням S-блоку від критерію незалежості виходу від вхідних змінних при його представленні компонентними q-функціями називається максимум серед усіх його відхилень від критерію незалежності виходу від вхідних змінних його компонентних q-функцій.

Визначення 19 [17]. Інтегральним відхиленням S-блоку від критерію незалежності виходу від вхідних змінних при його представленні компонентними q-функціями називається сумарне значення всіх відхилень від критерію незалежності виходу від вхідних компонентних q-функцій:

$$\Lambda f_{qi} = \sum_{i=0}^{k} \Delta f_{qi} \quad (1.16)$$

1.4 Схема Кіма

У наш час одним з найпопулярніших методів синтезу криптографічних примітивів, що представлені у вигляді S-блоків та відповідають СЛК є схема Кіма. Дана схема надає можливість до синтезу S-блоків будь-якої довжини, яка відповідає усім можливим представленням ступенів двійки *k*, тобто 8, 16, 32, 64 і так до нескінченності. Необхідно зазначити, що уся множина S-блоків довжини 8, що відповідає СЛК знаходиться досить легко, до того ж використовуючи всього лише метод повного перебору.

Розглянемо схему більш детально. Наприклад, в нас є є деякий S-блок довжини 8, який відповідає СЛК та буде позначений як S_k.

$$S_k = \{4 \quad 7 \quad 2 \quad 6 \quad 1 \quad 5 \quad 0 \quad 3\} \tag{1.17}$$

Крок 1. Виконуємо розкладання блоку (1.17) на компонентні булеві функції (Таблиця 1.1)

Таблиця 1.1 – Розбиття S-блоку на булеві функції

4	**7**	**2**	**6**	**1**	**5**	**0**	**3**
1	1	0	1	0	1	0	0
0	1	1	1	0	0	0	1
0	1	0	0	1	1	0	1

Крок 2. Необхідно обрати компоненту функцію F_m, що є найстаршою компонентною функцією серед представлених. Відповідно до Таблиці 1.1 це 10001101. Дану булеву функцію необхідно подвоїти використовуючи наступне правило горизонтальної конкатенації:

$$G_1[F_m] = \{F_m(x) \mid F_m(x \oplus e_k^{(a)}) \oplus 1\}, \quad x = 0,1,...,2^k - 1, \tag{1.18}$$

де $e_k^{(a)}$ – вектор довжини k, що містить 1 на позиції a, $|$ – символ горизонтальної конкатенації, $F_m(x \oplus e_k^{(a)})$ – похідна функції F_m по направленню $e_k^{(a)}$

Тоді при векторі $e_k^{(a)}$, такому що $e_k^{(a)} = \{0\ 0\ 1\}$ отримаємо:

$$G_1\left[F_m\right] = \{0\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0\} \tag{1.19}$$

Крок 3. Використовуємо правило горизонтальної конкатенації вже безпосередньо до S-блоку і подвоюємо його за наступним правилом:

$$G_0[S_{k+1}] = \{S_k(x) \mid S_k(x \oplus e_k)\}, \quad x = 0,1,...,2^k - 1 \tag{1.20}$$

Після чого знову беремо $e_k^{(a)}$, такий що $e_k^{(a)} = \{0\ 0\ 1\}$ та синтезуємо новий небієктивний (такий, що має однакові значення) S-блок:

$$G_0[S_{k+1}] = \{4\ 7\ 2\ 6\ 1\ 5\ 0\ 3\ 7\ 4\ 6\ 2\ 5\ 1\ 3\ 0\} \quad (1.21)$$

Крок 4. Перетворюємо небієктивний блок на бієктивний $S_{k+1} = S_4$, що відповідає критеріям СЛК за наступним правилом

$$S_{k+1} = \{G_1[F_m] \cdot 2^k + G_0[S_k]\}, \quad (1.22)$$

звідки отримуємо:

$$S_4 = \{4\ 7\ 2\ 14\ 1\ 13\ 8\ 11\ 15\ 12\ 6\ 10\ 5\ 9\ 3\ 0\} \quad (1.23)$$

Після знаходження ваги похідних компонентних булевих функцій $wt(D_{i,k})$ для (1.12) можна дійти до висновку, що даний S-блок відповідає СЛК (Таблиця 1.2) [18, 19].

Таблиця 1.2 – Ваги похідних функцій

e_j	$wt(D_{1,k})$	$wt(D_{2,k})$	$wt(D_{3,k})$	$wt(D_{4,k})$
0001	8	8	8	8
0010	8	8	8	8
0100	8	8	8	8
1000	8	8	8	8

Хоча схема Кіма і дозволяє синтезувати криптографічно якісні S-блоки з точки зору відповідності вимогам СЛК булевої логіки, проте ця особливість і зберігається лише у рамках математичного апарату булевої логіки. Така обмеженість даного методу у наш час є доволі суттєвою, так як сучасна динаміка розвитку IT-сфери, а відповідно і її криптографічного та криптоаналітичного аспекту вимагають вже більш якісних криптографічних примітивів, таких, що мали відповідність хоча б одному критерію поза межами лише одного математичного представлення. Наприклад мали відповідність вимогам СЛК у булевому та 4-представленні [19].

Можна дійти до висновку, що так як світовий стандарт шифрування AES був розроблений у 90-ті роки, ураховуючи лише можливості булевої логіки, тоді найбільш вірогідно, що і блоки конструкції Ніберг не відповідають СЛК 4-логіки, тим самим знижуючи можливості криптоалгоритму. Більш того, існуючі методи синтезу S-блоків не можуть надати рішення цієї проблеми. Віповідно до цього, необхідно створити новий метод синтезу, що задовольняв би хоча б МЛК в булевій системі та СЛК в 4-системі.

1.5 Побудова S-блоків відповідно до теорії динамічного хаосу

В останні роки досить великої популярності здобув синтез так званих S-блоків, побудованих за допомогою теорії динамічного хаосу. Теорія динамічного хаосу намагається відповісти на питання де пролягає кордон між регулярною, проте складно організованою структурою (тобто порядком) та безладом. Нині слово «безлад» замінюється на хаос.

Концепція динамічного хаосу уперше отримала своє обґрунтування на прикладі простої моделі статистичної механіки – більярду. Системи, що відповідають більярду зводяться до ряду задач статичної фізики, однак виявилось, що навіть добре детермінована система (в нашому випадку доволі проста, бо вона має лише два шари) в залежності від форми кордонів може мати хаотичні властивості (рис 1.2) [20].

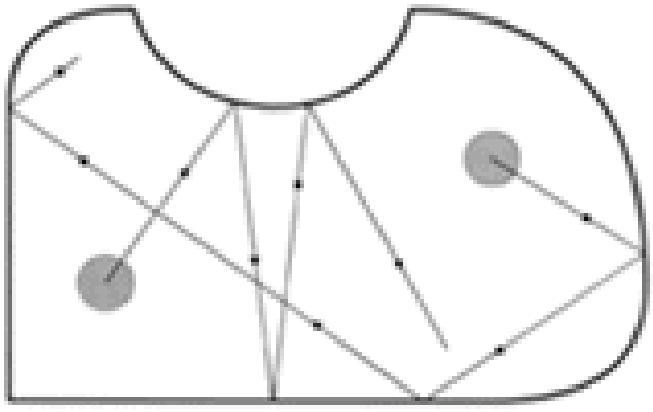

Рис.1.2 – Детермінована система з хаотичними властивостями

Особливості хаотичних систем, такі як висока періодичність, плутаність, висока чутливість до початкових умов роблять її перспективними кандидатами для розробки надійних систем безпеки для захисту зображення, аудіо, відео тощо.

Такі системи досліджуються заради синтезу нелінійних компонентів блокових шифрів, тобто блоків підстановки. Дослідники у галузі криптографії намагаються побудувати криптографічні якісні S-блоки на основі теорії динамічного хаосу, які мають бажані властивості для ускладнення диференціального, лінійного та іншого криптоаналізу.

На даний момент теорія динамічного хаосу є досить розвиненою, так як вже розроблено досить багато різноманітних методів класифікацій хаосу, знайдені закономірності його розвитку, а також створені методики, що надають нам змогу відрізнити хаос від білого шуму. І що, є найбільш цікавим відносно нашої предметної області це те, що було доведено, що складне просторово-часове розподілення середовищ з великою кількістю вільних ступенів може бути адекватно описано нелінійними системами невеликої розмірності. З точки зору криптографії це надає нам змогу розроблювати потенційно якісні криптографічні структури [21].

2 СИНТЕЗ ВИСОКОЯКІСНИХ S-БЛОКІВ

1.1 Дослідження S-блоків, синтезованих на основі теорії динамічного хаосу

Теорія динамічного хаосу є одним із перспективних інструментів для синтезу S-блоків, що відповідають критеріям криптографічної якості, які були описані вище. Проте подальший розвиток криптографії та методів криптоаналізу привів до розробки нових атак, заснованих на представленні шифрів за допомогою багатозначних логічних функцій, що робить необхідним дослідження криптографічної якості даних S-блоків не тільки тоді, коли вони представлені булевими функціями, а також для всіх можливих представлень функціями багатозначної логіки.

На даний момент створено багато методів синтезу криптографічно якісних S-боксів на основі теорії динамічного хаосу, найбільш поширені з них представлені в [22…31]. Усі зазначені S-блоки мають довжину $N = 256$, що відповідає архітектурі сучасних криптографічних алгоритмів. Таким чином, усі ці S-блоки можна представити у вигляді 8 компонентних булевих функцій, 4 компонентних 4-функцій або 2 компонентних 16-функцій, кожна з яких визначає криптографічну якість усієї конструкції та повинна бути ретельно досліджена.

Використовуючи математичний апарат для дослідження криптографічної якості S-блоків, описаний у розділі 1 цієї роботи, ми проводимо дослідження криптографічної якості S-блоків на основі теорії динамічного хаосу [5…14], результати якого представлені в таблицях 2.1-2.3

Таблиця 2.1 — Значення показників криптографічної якості S-блоків на основі теорії динамічного хаосу у випадку булевої логіки:

№	S-блок	ΔK_S	$\Delta_{\max} K_S$	N_{f_2}
1	[22]	520	32	106

Продовження таблиці 2.1

№	S-блок	ΔK_S	$\Delta_{\max} K_S$	N_{f_2}
2	[23]	656	32	98
3	[24]	552	28	102
4	[25]	544	24	104
5	[26]	524	28	96
6	[27]	484	24	100
7	[28]	484	24	106
8	[29]	636	28	104
9	[30]	512	28	104
10	[31]	**432**	**16**	**112**

Таблиця 2.2 — Значення показників криптографічної якості S-блоків на основі теорії динамічного хаосу у випадку 4-логіки:

№	S-блок	ΔK_S	$\Delta_{\max} K_S$	N_{f_4}
1	[22]	1232	18	209.1385
2	[23]	1120	18	213.2449
3	[24]	1312	43	202.3344
4	[25]	1108	24	213.4794
5	[26]	1088	28	214.7689
6	[27]	1024	20	216
7	[28]	1016	**16**	213.0582

8	[29]	1052	22	210.3054
9	[30]	1096	18	212.1366
10	[31]	**880**	**16**	**216.6046**

Таблиця 2.3 — Значення показників криптографічної якості S-блоків на основі теорії динамічного хаосу у випадку 16-логіки:

№	S-блок	ΔK_S	$\Delta_{\max} K_S$	$N_{f_{16}}$
1	[22]	3108	**12**	212.3220
2	[23]	2876	14	206.1523
3	[24]	3276	17	194.5649
4	[25]	2956	12	214.1403
5	[26]	2980	12	213.7293
6	[27]	2868	14	216.9470
7	[28]	2804	14	217.0569
8	[29]	2968	11	215.2830
9	[30]	2908	18	**219.1407**
10	[31]	**2728**	14	216.5184

Аналіз даних, наведених у табл. 2.1-2.3, показує, що криптографічні властивості S-блоків, побудованих на основі теорії динамічного хаосу, сильно відрізняються для різних відомих структур, як у випадку їх представлення компонентними булевими функціями, так і компонентними функції багатозначної логіки. При цьому за більшістю розрахованих показників найкращою є конструкція [31], яка характеризується як високим рівнем нелінійності, малими диференційними та інтегральними відхиленнями від СЛК, так і малими піками кореляційної залежності вихідних векторів зі вхідних векторів.

Відповідно до даного дослідження можна дійти до наступних висновків:

- використання будь-якої відомої криптографічної конструкції для побудови S-блоків передбачає ретельне дослідження властивостей результуючого S-блоку, як при його представленні за допомогою математичного апарату булевих функцій, так і за допомогою математичного апарату функцій багатозначної логіки;

- методи синтезу S-блоків на основі теорії динамічного хаосу є перспективними, а синтезовані на їх основі S-блоки можуть характеризуватися високим рівнем криптографічної якості. Тим не менш, при розробці конструкцій S-блоків, заснованих на теорії динамічного хаосу, важливо враховувати їх властивості не тільки при представленні булевими функціями, але і коли вони представлені за допомогою функцій багатозначної логіки.

- серед сукупності досліджуваних S-блоків була знайдена конструкція, що характеризується найкращим рівнем криптографічної якості за більшістю розглянутих показників. Цю конструкцію можна рекомендувати для практичного використання в існуючих і розроблених криптографічних алгоритмах.

2.2 Метод створення високоякісних S-блоків

На жаль, теорія динамічного хаосу також дотримується принципів «компромісу» показників криптографічної якості – намагання знайти допустимо максимальні значення для одного критерію з урахуванням особливостей іншого, тобто таким чином, щоб при досягненні максимального допустимого значення у межах визначених криптографом, а також особливостями методу синтезу та для яких практичних завдань даний S-блок розроблюється, значення інших показників криптографічної якості не постраждали чи їх значення знизились на допустимий рівень. Наприклад, якщо S-блок має відповідність вимогам лавинного критерію не повинно статися так, що в нього відсутній кореляційний імунітет чи відстань нелінійності досягає рівня нижче за 50% від максимально можливого значення чи взагалі відсутня. Тобто, максимально можливі значення для

показників криптографічної якості фактично не зустрічаються навіть для одного критерію.

Емпіричним шляхом було виявлено, що рекурентна побудова S-блоків, які мають хороші лавинні властивості компонентних 2- та 4-функцій є досить перспективною, саме тому даний метод побудови покладено в основу розробленого методу і представлено у вигляді покрокового алгоритму.

Крок 1. В якості вхідного матеріалу для даного методу буде використовуватися множина S-блоків довжини $N=16$, що відповідають СЛК 4-функцій, які були побудовані у [19]. Дана множина має потужність $J=245760$.

У якості прикладу будемо розглядати S-блок, що декомпозується на булеві функції наступного вигляду:

$$S=\left[\begin{array}{c|cccccccccccccccc} Q & 0 & 1 & 3 & 7 & 14 & 2 & 10 & 9 & 6 & 12 & 11 & 4 & 13 & 8 & 15 & 5 \\ \hline f_{20} & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ f_{21} & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ f_{22} & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ f_{23} & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{array}\right] \qquad (2.9)$$

Крок 2. Задається функція F_m, яка представляє собою старшу компонентну 4-функцію в розкладанні S-блоку на 4-функції.

Якщо розглядати на прикладі S-блоку (2.9), то дана компонентна 4-функція має вигляд

$$F_m=[0001302213213231] \qquad (2.10)$$

Крок 3. Формується безліч з 4-х перестановок у відповідності з наступним правилом

$$p_j=x\underset{4}{\oplus}(j\circ d),\quad x=0,1,...,N-1,\quad j=0,1,2,3,, \qquad (2.11)$$

де d – один з векторів довжини $k=\log_4 N$ с 1 на одній зі своїх позицій, вектор x пробігає четвіркові представлення чисел від 0 до $N-1$, $\underset{4}{\oplus}$ – додавання по модулю 4, $\circ$ – символ поелементного множення четвіркового представлення числа d на значення j.

У випадку нашого прикладу $k = \log_4 16 = 2$, візьмемо значення $d = [0 \quad 1]$, відповідно множина перестановок буде мати наступний вигляд:

$$\begin{aligned} p_0 &= [0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \quad 11 \quad 12 \quad 13 \quad 14 \quad 15]; \\ p_1 &= [1 \quad 2 \quad 3 \quad 0 \quad 5 \quad 6 \quad 7 \quad 4 \quad 9 \quad 10 \quad 11 \quad 8 \quad 13 \quad 14 \quad 15 \quad 12]; \\ p_2 &= [2 \quad 3 \quad 0 \quad 1 \quad 6 \quad 7 \quad 4 \quad 5 \quad 10 \quad 11 \quad 8 \quad 9 \quad 14 \quad 15 \quad 12 \quad 13]; \\ p_3 &= [3 \quad 0 \quad 1 \quad 2 \quad 7 \quad 4 \quad 5 \quad 6 \quad 11 \quad 8 \quad 9 \quad 10 \quad 15 \quad 12 \quad 13 \quad 14]. \end{aligned} \tag{2.12}$$

Крок 4. Отримуємо функцію G_1 довжини $4N$, за наступним правилом

$$G_1[F_m] = \left\{ F_m \underset{4}{\oplus} c_1 \mid F_m(p_1) \underset{4}{\oplus} c_2 \mid F_m(p_2) \underset{4}{\oplus} c_3 \mid F_m(p_3) \underset{4}{\oplus} c_4 \right\}, \tag{2.13}$$

де $\{c_1, c_2, c_3, c_4\}$ – множина констант, що повинна містити всі значення з множини $\{0,1,2,3\}$. За замовчуванням візьмемо $c_1 = 0, c_2 = 1, c_3 = 2, c_4 = 3$.

У нашому прикладі отримуємо, що функція G_1 прийме наступний вигляд

$$G_1 = [000130221321323111211330032230202322001203311310033312310021021 2]. \tag{2.14}$$

Крок 5. Збільшуємо довжину S-блоку до значення $4N$ використовуючи наступну конструкцію

$$G_0 = \{S \mid S(p_1) \mid S(p_2) \mid S(p_3)\}. \tag{2.15}$$

Крок 6. Будуємо новий бієктивний S-блок довжини $4N$, що відповідає суровому лавинному критерію компонентних 4-функцій за наступним правилом

$$S_1 = \{G_1 \cdot 4^k + G_0\}, k = \log_4 N. \tag{2.16}$$

У нашому випадку отримуємо наступний S-блок довжини $N = 64$

$$\begin{aligned} S_{64} = [&0,1,3,23,62,2,42,41,22,60,43,20,61,40,63,21, \\ &17,19,39,16,18,58,57,14,12,59,36,38,56,15,37,13, \\ &35,55,32,33,10,9,30,34,11,52,54,28,31,53,29,8,7,48, \\ &49,51,25,46,50,26,4,6,44,27,5,45,24,47]. \end{aligned} \tag{2.17}$$

Отриманий S-блок (2.17) згідно до умов Визначення 6 має наступну вагову матрицю похідних компонентних булевих функцій (Таблиця 2.1)

Таблиця 2.1 – Вага синтезованого S-блоку

e_j	$wt(D_{1,k})$	$wt(D_{2,k})$	$wt(D_{3,k})$	$wt(D_{4,k})$	$wt(D_{5,k})$	$wt(D_{6,k})$

000001	32	40	32	24	32	40
000010	32	32	32	32	32	32
000100	32	32	32	32	32	32
001000	32	32	32	32	32	32
010000	32	40	32	24	32	24
100000	32	32	32	32	32	32

тобто не відповідає СЛК булевих функцій

Повторне використання розробленого методу до S-блоку (2.17) довжини $N = 64$ дозволяє отримати на його основі новий S-блок довжини $N = 256$

$$S_{256} = [0,1,3,87,254,2,170,169,86,252,171,84,253,168,255,85,81,\\ 83,167,80,82,250,249,14,12,251,164,166,248,15,165,13,163,247,\\ 160,161,10,9,94,162,11,244,246,92,95,245,93,8,7,240,241,243,\\ 89,174,242,90,4,6,172,91,5,173,88,175,145,147,231,144,146,58,\\ 57,78,76,59,228,230,56,79,229,77,227,55,224,225,74,73,158,\\ 226,75,52,54,156,159,53,157,72,71,48,49,51,153,238,50,154,68,\\ 70,236,155,69,237,152,239,64,65,67,151,62,66,234,233,150,60,\\ 235,148,61,232,63,149,35,119,32,33,138,137,222,34,139,116,\\ 118,220,223,117,221,136,135,112,113,115,217,46,114,218,132,\\ 134,44,219,133,45,216,47,128,129,131,215,126,130,42,41,214,\\ 124,43,212,125,40,127,213,209,211,39,208,210,122,121,142,140,\\ 123,36,38,120,143,37,141,199,176,177,179,25,110,178,26,196,\\ 198,108,27,197,109,24,111,192,193,195,23,190,194,106,105,22,\\ 188,107,20,189,104,191,21,17,19,103,16,18,186,185,206,204,187,\\ 100,102,184,207,101,205,99,183,96,97,202,201,30,98,203,180,\\ 182,28,31,181,29,200], \tag{2.18}$$

який теж відповідає СЛК компонентних 4-функцій, і характеризується наступною ваговою матрицею похідних компонентних булевих функцій (Таблиця 2.2)

Таблиця 2.2 — Значення синтезованого S-блоку

e_j	$wt(D_{1,k})$	$wt(D_{2,k})$	$wt(D_{3,k})$	$wt(D_{4,k})$	$wt(D_{5,k})$	$wt(D_{6,k})$	$wt(D_{7,k})$	$wt(D_{8,k})$
00000001	128	160	128	96	128	160	128	128
00000010	128	128	128	128	128	128	128	128
00000100	128	128	128	128	128	128	128	128

00001000	128	128	128	128	128	128	128	128
00010000	128	160	128	96	128	128	128	96
00100000	128	128	128	128	128	128	128	128
01000000	128	160	128	96	128	128	128	128
10000000	128	128	128	128	128	128	128	128

тобто не відповідає СЛК булевих компонентних функцій.

Проведені експерименти показують, що на основі повної множини S-блоків довжини $N=16$, шляхом використання розробленого методу можуть бути отримані S-блоки практично цінної довжини $N=256$, що відповідають одночасно СЛК компонентних 4-функций та критерію максимального лавинного критерію булевих функцій.

Варто зазначити, що кількість S-блоків сильно залежить від значення обраних параметрів c_1, c_2, c_3, c_4, а також від обраного вектору направлення d. Наприклад, нехай задані наступні параметри: довжина S-блоку $N=256$, параметри $c_1=0, c_2=1, c_3=2, c_4=3$ на першій та другій ітерації використання методу, а також значення $d=[0 \quad 1]$ на першій ітерації використання методу і $d=[0 \quad 0 \quad 1]$ на другій ітерації використання методу.

Тоді отримуємо, що з множини $J=245760$ S-блоків довжини $N=256$, отриманих на основі множини S-блоків довжини $N=16$ [19] $J_1=3968$ S-блоків одночасно відповідають СЛК компонентних 4-функцій точно і критерію максимального лавинного ефекту компонентних булевих функцій.

У якості прикладу приведемо один з даних S-блоків довжини $N=256$, компонентні 4-функції якого відповідають СЛК, побудованих на основі S-блоку $S=[0,1,2,12,7,11,3,8,15,9,6,13,5,4,10,14]$ (додаток В) та його вагову матрицю похідних булевих функцій (Таблиця 2.3)

Таблиця 2.3 – Вага синтезованого S-блоку

e_j	$wt(D_{1,k})$	$wt(D_{2,k})$	$wt(D_{3,k})$	$wt(D_{4,k})$	$wt(D_{5,k})$	$wt(D_{6,k})$	$wt(D_{7,k})$	$wt(D_{8,k})$

00000001	128	160	128	160	128	128	128	160
00000010	128	128	128	128	128	128	128	128
00000100	128	128	128	128	128	128	128	128
00001000	128	128	128	128	128	128	128	128
00010000	128	160	128	160	128	128	128	160
00100000	128	128	128	128	128	128	128	128
01000000	128	160	128	160	128	128	128	128

Продовження таблиці 2.3

e_j	$wt(D_{1j})$	$wt(D_{2j})$	$wt(D_{3j})$	$wt(D_{4j})$	$wt(D_{5j})$	$wt(D_{6j})$	$wt(D_{7j})$	$wt(D_{8j})$
10000000	128	128	128	128	128	128	128	128

яка підтверджує, що вказаний S-блок (додаток В) дійсно відповідає критерію максимального лавинного эффекту компонентних булевих функцій [9]. Однак вимоги СЛК все ще не виконуються.

Крок 7. Пробігаючи усю множину S-блоків, необхідно виконати їх декомпозицію на такі 4-функції, які містять у собі булеві функції, що задовольняють умовам СЛК. Фактично виходить, що завдяки одній високоякісній 4-функції ми отримуємо одразу 2 високоякісні булеві функції. Після такої декомпозиції ми отримуємо множину 4-функцій, потужність якої складає 14336 функцій. Одна з таких функцій має наступний вигляд:

Fun_4 = [0, 1, 3, 3, 2, 2, 2, 1, 2, 0, 3, 0, 1, 0, 3, 1, 1, 3, 3, 0, 2, 2, 1, 2, 0, 3, 0, 2, 0, 3, 1, 1, 3, 3, 0, 1, 2, 1, 2, 2, 3, 0, 2, 0, 3, 1, 1, 0, 3, 0, 1, 3, 1, 2, 2, 2, 0, 2, 0, 3, 1, 1, 0, 3, 2, 0, 0, 1, 3, 3, 2, 3, 1, 0, 1, 3, 1, 0, 2, 2, 0, 0, 1, 2, 3, 2, 3, 3, 0, 1, 3, 1, 0, 2, 2, 1, 0, 1, 2, 0, 2, 3, 3, 3, 1, 3, 1, 0, 2, 2, 1, 0, 1, 2, 0, 0, 3, 3, 3, 2, 3, 1, 0, 1, 2, 1, 0, 2, 1, 1, 2, 3, 0, 3, 0, 0, 1, 2, 0, 2, 1, 3, 3, 2, 1, 2, 3, 1, 3, 0, 0, 0, 2, 0, 2, 1, 3, 3, 2, 1, 2, 3, 1, 1, 0, 0, 0, 3, 0, 2, 1, 2, 3, 2, 1, 3, 3, 1, 1, 2, 0, 0, 3, 0, 2, 1, 2, 0, 2, 1, 3, 3, 2, 3, 0, 2, 0, 1, 1, 1, 3, 1, 3, 2, 0, 0, 3, 2, 3, 0, 2, 2, 1, 1, 1, 0, 1, 3, 2, 3, 0, 3, 2, 0, 0, 2, 2, 3, 1, 1, 0, 1, 3, 2, 3, 1, 3, 2, 0, 0, 2, 2, 3, 0, 1, 0, 1, 1, 2, 3, 1, 3, 2, 0, 0, 3 (2.19)

Крок 8. З отриманої множини 4-функцій, необхідно виділити лише такі, що є унікальними. Таким чином підвищується обчислювальна швидкість при генерації S-блоків, що робить можливим синтез усіх можливих варіантів за

допомогою використання методу звичайного перебору. Після проведення даної процедури видалення дублікатів з даної множини отримуємо в решті всього лише 769 таких функцій.

Крок 9. Виконуючи композицію отриманих 4-функцій між собою, генеруємо множину S-блоків, вишукуючи лише ті, що мають властивості бієктивності – усі значення числової послідовності мають бути унікальними. В нашому випадку це уся множина чисел в діапазоні від 0 до 255. Після виконання даної процедури синтезу S-блоків на основі вже максимально якісних компонентних 4-функцій з отриманих на усіх попередніх кроках отримуємо множину, потужність якої складає 117588 унікальних S-блоків. Одним з таких блоків є блок, представлений на таблиці 2.4.

Таблиця 2.4 Приклад високоякісного S-блоку, що відповідає СЛК 2- та 4-логіки

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

F	F5	25	F0	FF	BD	10	06	5C	86	A3	6A	12	99	6B	CB	4C

Варто зазначити, що отриманий S-блок дійсно є високоякісною конструкцією, як мінімум відносно одного критерію, так як він був побудований на основі високоякісних 4-функцій, які відповідають СЛК четвіркової логіки, які в свою чергу декомпозуються на такі ж якісні булеві функці, але вже у сенсі булевої логіки. Дослідження лавинних властивостей S-блоку дійсно демонструє його повну відповідність вимогам СЛК як у сенсі булевої (таблиця 2.5), так і у сенсі 4-логіки (таблиця 2.6)

Таблиця 2.5 — Відповідність вимогам СЛК булевої логіки синтезованого високоякісного S-блоку

e_j	$wt(D_{1,k})$	$wt(D_{2,k})$	$wt(D_{3,k})$	$wt(D_{4,k})$	$wt(D_{5,k})$	$wt(D_{6,k})$	$wt(D_{7,k})$	$wt(D_{8,k})$
00000001	128	128	128	128	128	128	128	128
00000010	128	128	128	128	128	128	128	128
00000100	128	128	128	128	128	128	128	128
00001000	128	128	128	128	128	128	128	128
00010000	128	128	128	128	128	128	128	128
00100000	128	128	128	128	128	128	128	128
01000000	128	128	128	128	128	128	128	128
10000000	128	128	128	128	128	128	128	128

Таблиця 2.6 — Відповідність вимогам СЛК 4-логіки синтезованого високоякісного S-блоку

e_j	$wt(D_{1,k})$	$wt(D_{2,k})$	$wt(D_{3,k})$	$wt(D_{4,k})$
0001	64	64	64	64
0002	64	64	64	64
0003	64	64	64	64
0010	64	64	64	64

0020	64	64	64	64
0030	64	64	64	64
…	64	64	64	64
3333	64	64	64	64

Відповідно, можна стверджувати, що S-блоки даного класу мають максимально можливі лавинні властивості, що робить їх практичного цінними, наприклад для використання у таких криптографічних конструкціях як хеш-функції, де лавинні властивості є найголовнішим критерієм, що визначають ступінь якості хеш функції.

Однак, інші криптографічні якості цього S-блоку також виявилися досить якісними при тому, що відповідність вимогам суворого лавинного критерію зберіглася одразу в двох системах числення. Наприклад, матриця кореляції як для булевого, так і для 4-представлення має одні нулі, що свідчить про кореляційний імунітет як мінімум для булевого представлення, тобто забезпечується максимально можлива статистична незалежність параметрів виходу S-блоку від параметрів входу.

Ще одна особливість полягає в тому, що й відстань нелінійності для двійкового представлення складає 64 (57% від максимального значення 112) та 155 (66% від максимального значення 240).

2.3 Порівняння ентропійних властивостей криптоалгоритму AES при використанні оригінальних та синтезованих S-блоків.

При вирішенні більшості практичних завдань необхідно знати середню кількість інформації $H(A)$, яка припадає на символ алфавіту, яка визначається шляхом усереднення $-\log_2 p(a_i)$ по всій множині (обсягу) алфавіту:

$$H(A) = -\sum_{i=1}^{m} p(a_i) \log_2 p(a_i) \left[\frac{біт}{символ} \right], \qquad (2.20)$$

де $H(A)$ – середня кількість інформації, $p(a_i)$ – статистика появи символу

a_i

Даний вираз відомий у теорії зв’язку як формула Шеннона для ентропії джерела дискретних повідомлень. Ентропія сприймається як міра невизначеності у формуванні джерелом того чи іншого повідомлення (літери).

Спосіб вимірювання кількості інформації, запропонований К. Шенноном, є узагальненням способу Хартлі у разі появи нерівноймовірних незалежних символів [32].

Для тестування ентропійних властивостей криптоалгоритму AES були використано 3 S-блоки, що побудовані у [33-35], де з кожної множини було взято по одному представнику, де перший S-блок – оригінальний S-блок з конструкції Ніберг, другий S-блок з відповідністю СЛК 4-логіки та МЛК 2-логіки, а третій синтезований у даній роботі.

Для кожної модифікації криптоалгоритму AES було виконано тестування 10000 різноманітних текстових записів з початковою ентропією 0.5 з метою виявлення швидкості наближення кожної з них до значення ентропії 1. Показники рахувались для кожного раунду окремо, після чого були вирахувані усі середні значення. Дані досліджень зображені на рисунку 2.1.

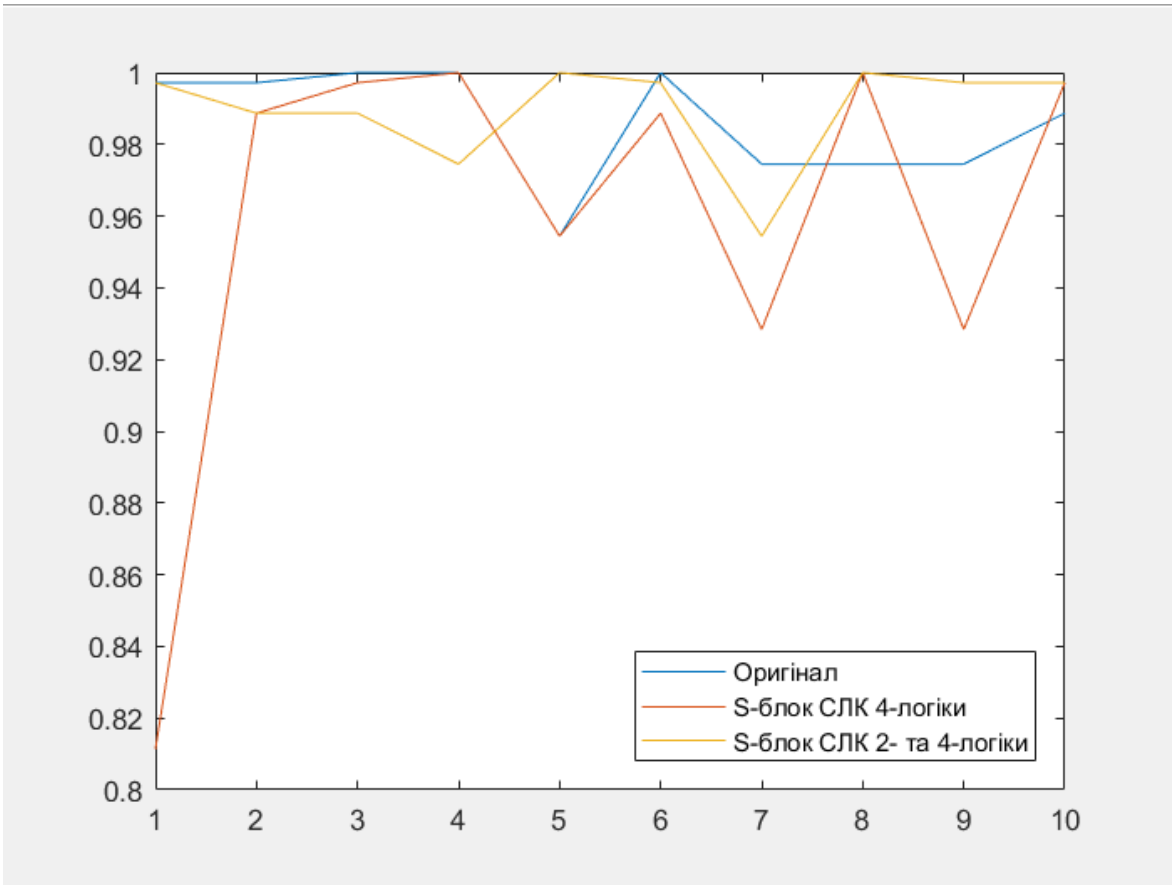

Рисунок 2.1 – Графіки швидкості досягнення максимальної ентропії трьох різноманітних модифікацій криптоалгоритму AES.

Відповідно до рисунку 2.1 можна зазначити, що швидше всього максимального значення ентропії досягає оригінально побудований криптоалгоритм AES. На 1 ітерацію довше максимум досягається з використанням S-блоку з відповідністю МЛК 2-логіки та СЛК 4-лоігки та ще на 2 ітерації довше його досягає AES, що використовує S-блок, що синтезований у даній роботі.

Однак, якщо розглянути наскільки сильно коливається значення ентропії (для зручності будемо розглядати апроксимовані графіки), то бачимо результати, що представлені на рисунку 2.2.

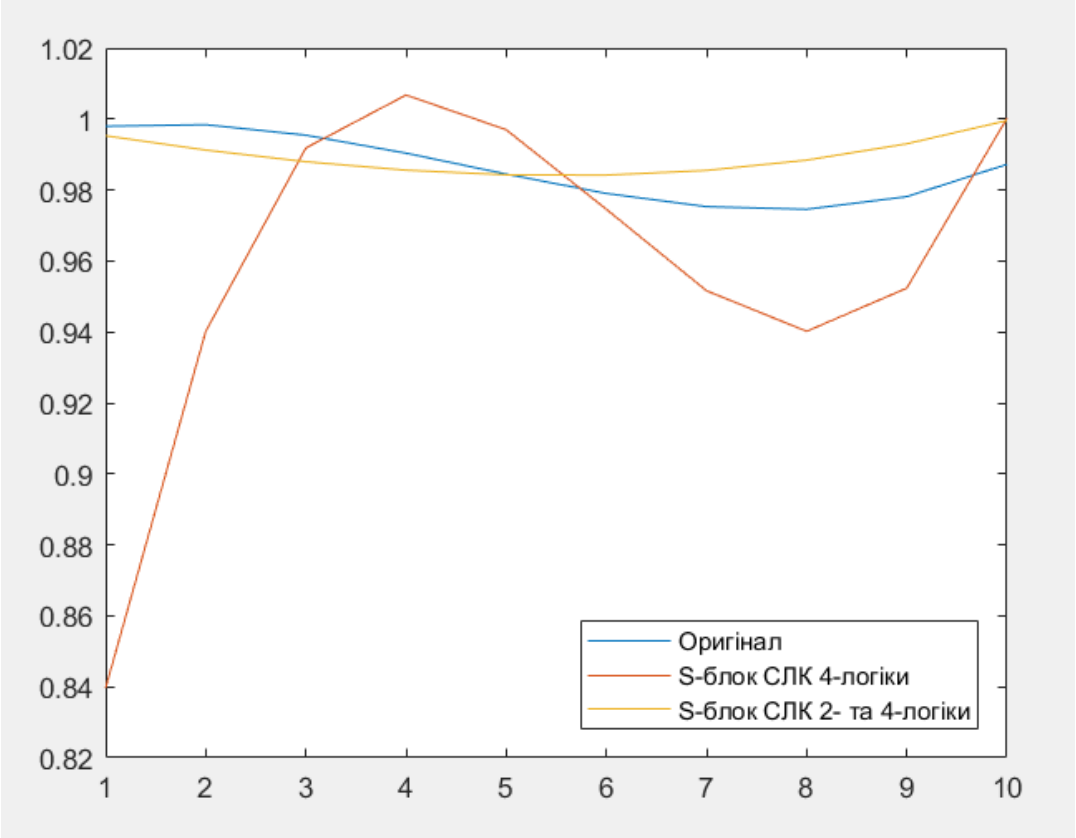

Рисунок 2.2 – Апроксимовані графіки ентропії 3-х різним модифікацій AES

Як бачимо, найбільш стабільним виявився AES, що використовує S-блоки, побудовані у даній роботі. Варто зазначити, що усі значення знаходяться у діапазоні допустимої похибки, тому фактично всі ці три модифікації показують досить непогані результати.

Відповідно до усіх вищеперерахованих факторів можна коротко резюмувати практичні результати дослідження S-блоків, заснованих на теорії динамічного хаосу, а також отриманих криптографічних примітивів на основі запропонованого методу синтезу високоякісних S-блоків:

- S-блоки, що були побудовані за рахунок різноманітних математичних моделей, які базуються на теорії динамічного хаосу мають досить непогані лавинні властивості та у деяких випадках мають фактично максимально можливу відстань нелінійності, однак жоден з критеріїв не досягає максимального значення, тобто в криптографічних примітивах такого класу зберігається принцип «компромісу»;

- розроблено метод синтезу S-блоків, що побудований на рекурентному методі побудови S-блоків та дозволяє синтезувати S-блоки практичної цінної довжини 256, яка є фактично стандартом для сучасним криптографічних методів, наприклад для криптоалгоритму AES;
- отримана множина S-блоків є досить великою, а потужність множини складає 117588 блоків, що дозволяє застосовувати ці блоки також у якості криптографічних ключів. Більш того, шляхом різноманітних криптографічних перетворень (наприклад інверсії чи віддзеркалення) дана множина може бути потенційно збільшена щонайменше до 4 разів;
- S-блоки даного класу мають відповідність СЛК у сенсі булевої та четвіркової логіки, тобто одразу у двох системах числення, що робить їх лавинні властивості фактично унікальними серед знайдених на цей час S-блоків практично цінної довжини 256;
- даний набір блоків підстановки окрім відповідності СЛК у двох системах числення, також має кореляційний імунітет у сенсі булевої логіки, що забезпечує статистичну незалежність виходу від входу даних криптографічних примітивів;
- незважаючи на відповідність одразу двом критеріям (СЛК та кореляційному імунітету), конструкції даної множини також визначаються відстанями нелінійності, які перевищують половину від можливого максимального значення в обох представленнях таких S-блоків: у якості компонентних булевих та 4-функцій;
- з огляду на відповідність вимогам фактично 3 критеріїв, дані S-блоки можуть бути запропоновані для практичного використання, наприклад для модернізації вже існуючих криптографічних алгоритмів, а також для розробки нових.

2 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ СИНТЕЗУ ЯКІСНИХ S-БЛОКІВ

2.1 Середовище програмування

При написанні програмного продукту, що реалізує синтез високоякісних S-блоків, що були побудовані у ході даної роботи було використано мову програмування Python, що відносно нещодавно здобула широкого вжитку до того ж у найрізноманітніших сферах.

У нашому випадку було використано мову програмування Python версії 3.11.0. На вибір саме цієї мови програмування у якості мови реалізації приведеного у даній роботі метода синтезу високоякісних S-блоків вплинули наступні фактори.

1. Зручність написання коду – Python вважається і фактично визнано однією з найпростіших мов програмування серед усіх аналогів (простіша тільки Scratch, однак для серйозних розробок вона не використовується), а парадигма того, що все є об’єктами значно полегшує написання алгоритмів через наявність широкого вбудованого інструментарію для роботи з різними видами об’єктів.

2. Велика кількість вбудованих бібліотек – популярність мови Python та її застосування у різноманітних галузях IT-сфери, таких як веб-розробка чи аналітика даних, зумовило досить велику її популярність і відповідно розширило кількість IT-спеціалістів, що її використовують. Для зручності роботи цих самих фахівців було розроблено досить об’ємну базу різноманітних вбудованих бібліотек, що направлені на різні задачі, а також з кожним роком виходять все нові бібліотеки, що розроблені сторонніми спеціалістами, а старі систематично оновлюються. Варто зазначити, що дана мова також має досить великий математичний апарат, що надає змогу виконувати різноманітні математичні операції.

3. Модульний підхід – дана мова програмування дозволяє розподілити код на різноманітні модулі, тобто винесення частин коду, що відповідають за різні сфери роботи програми у окремі файли, а при необхідності використовувати їх функціональні можливості у роботі головного модуля програми або ж обмінюватись функціональними властивостями між собою, що допомагає уникати дублювання функціоналу, а також підвищує читабельність коду, що значно спрощує його подальшу підтримку.

4. Графічний інтерфейс – Python має декілька різноманітних інструментів для створення графічного інтерфейсу, що дозволяє пересічним користувачам без усіляких проблем використовувати програмні продукти, що розроблені за допомогою цієї мови програмування. Найбільш популярними бібліотеками графічного інтерфейсу є PyQt, Tkinter та PySimpleGui. Останню було використано для реалізації програмного продукту даної роботи.

5. Високі вимоги до написання коду – дане середовище потребує додержання відступів при написанні коду, що фактично примушує спеціаліста писати код естетично і надає змогу тим, хто захоче надалі ознайомитися з цим кодом більш легко орієнтуватися і, відповідно, швидше освоювати написаний функціонал, тобто також підвищує читабельність.

6. Кросплатформеність – дане середовище програмування підтримує різні платформи, зокрема і найпопулярніші, такі як Windows, Linux, MacOS, Android. Такий широкий спектр платформ дозволяє відносно легко розроблювати програмні продукти без переймань щодо підтримки розроблено програмного продукту на ті чи іншій платформі. Даний факт є дуже важливим, оскільки зняття такого обмеження дозволяє використовувати програмні продукти, що розроблені за допомогою Python усім можливим користувачам в яких є свій ПК.

7. Широке застосування провідними ІТ-компаніями світу, що дає можливість імплементувати створені розробки до програмного середовища більшості ІТ-компаній.

8. Швидкість виконання операцій – фактично дана перевага була надана мові програмування Python починаючи з версії 3.11, що дало змогу підвищити швидкість усіх процесів фактично до 25% від початкової швидкості, що в сукупності з великим обсягом математичних обчислень, які використовуються в ході синтезу високоякісних S-блоків робить дану мову програмування ідеальним кандидатом для різноманітних наукових досліджень. Тобто дані покращення мови Python надали їй змогу максимально ефективно використовувати обчислювальні ресурси.

Для зручності використання програми її основні функціональні блоки було

розподілено на декілька модулів.

1. base2.py – модуль програми, що відповідає за основні обчислення, що проводяться у двійкові системі у рамках проекту, такі як розбиття блоку на булеві функції, створення зворотного блоку, композиція булевих функцій в один S-блок, представлення функцій у строковому вигляді тощо.

2. base4.py – модуль програми, що відповідає за основні обчислення, що проводяться у четвірковій системі у рамках проекту, такі як операція XOR, розбиття блоку на 4-функції блоку тощо, створення зворотного блоку, композиція 4-функцій в один S-блок, представлення функцій у строковому вигляді тощо.

3. findWeight2base.py – модуль програми, у якому реалізовано знаходження СЛК S-блоку у його булевому представленні.

4. findWeight2baseFUN.py – модуль, у якому реалізовано знаходження СЛК булевої функції.

5. findWeight4base.py – модуль програми, у якому реалізовано знаходження СЛК S-блоку у його четвірковому представленні.

6. ci2.py – програмний модуль для розрахунку матриці кореляції.

7. nonlin.py – модуль програми, в якому реалізовано розрахунок відстані нелінійності S-блоку у його представленні у вигляді набору булевих функцій.

8. nonlin4.py – модуль програми, в якому реалізовано розрахунок відстані нелінійності S-блоку у його представлені у вигляді набору 4-функцій.

9. matrixToStr – модуль перетворення числового варіанту матриць до строкового, так яку це матричне представлення є простішим для презентації у графічному компоненті програми.

10. ui.py – модуль програми, в якій реалізовано графічний інтерфейс користувача та представлено декомпозицію S-блоку, його відстань нелінійності, матрицю кореляційного імунітету, а також матриці суворого лавинного критерію у сенсі 2 та 4-логіки.

Розглянемо наведені модулі більш детально.

3.2 Опис програмного коду продукту

Основна логіка роботи програми, в якій поєднані та використані усі можливості функціональних модулів, що були описані вище, представлена саме у модулі ui.py.

Даний програмний модуль містить у собі всього один клас Ui_MainWindow, в якому і реалізовано логіку роботи програми, що поєднана с його візуальною складовою – інтерфейсом. Зазначимо, що інтерфейс реалізовано за допомогою сторонньої бібліотеки – PyQt. До складу Ui_MainWindow входять наступні методи:

- setupUI;
- retranslateUI;
- decomposeBlock;
- calcualteResults;
- generateGood;
- base2results;
- base4results

Варто зазначити, що у рамках даної програми присутній список 4 заздалегідь підібраних S-блоків, що відповдіюать МЛК 2-логіки та СЛК 4-логіки на основі яких і генерується високоякісний S-блок, який описаний у даній роботі, один з низ представлений на рисунку 3.1.

```
'[0, 4, 12, 85, 93, 174, 255, 89, 162, 81, 247, 251, 8, 166, 243, 170, 20, 28, 101, 16,
190, 207, 105, 109, 97, 199, 203, 178, 182, 195, 186, 24, 44, 117, 32, 36, 223, 121,
125, 142, 215, 219, 130, 113, 211, 138, 40, 134, 69, 48, 52, 60, 73, 77, 158, 239, 235,
146, 65, 231, 154, 56, 150, 227, 68, 76, 149, 64, 238, 63, 153, 157, 145, 55, 59, 226,
230, 51, 234, 72, 92, 165, 80, 84, 15, 169, 173, 254, 7, 11, 242, 161, 3, 250, 88, 246,
181, 96, 100, 108, 185, 189, 206, 31, 27, 194, 177, 23, 202, 104, 198, 19, 112, 116,
124, 133, 141, 222, 47, 137, 210, 129, 39, 43, 120, 214, 35, 218, 140, 213, 128, 132,
127, 217, 221, 46, 119, 123, 34, 209, 115, 42, 136, 38, 229, 144, 148, 156, 233, 237,
62, 79, 75, 50, 225, 71, 58, 152, 54, 67, 160, 164, 172, 245, 253, 14, 95, 249, 2, 241,
87, 91, 168, 6, 83, 10, 180, 188, 197, 176, 30, 111, 201, 205, 193, 103, 107, 18, 22,
99, 26, 184, 21, 192, 196, 204, 25, 29, 110, 191, 187, 98, 17, 183, 106, 200, 102, 179,
208, 212, 220, 37, 45, 126, 143, 41, 114, 33, 135, 139, 216, 118, 131, 122, 228, 236,
53, 224, 78, 159, 57, 61, 49, 151, 155, 66, 70, 147, 74, 232, 252, 5, 240, 244, 175,
9, 13, 94, 167, 171, 82, 1, 163, 90, 248, 86]',
```

Рисунок 3.1 – один з S-блоків, що використовується у програмі

Покажемо часткову програмну реалізацію методу setupUi (рисунок 3.2).

```
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(906, 869)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)

        sizePolicy.setHorizontalStretch(1)
        sizePolicy.setVerticalStretch(1)
        sizePolicy.setHeightForWidth(MainWindow.sizePolicy().hasHeightForWidth())
        MainWindow.setSizePolicy(sizePolicy)

        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")

        self.list = QtWidgets.QListWidget(self.centralwidget)
        self.list.setGeometry(QtCore.QRect(10, 10, 861, 181))
        self.list.setObjectName("list")
        self.list.addItems(Sboxes)
        self.list.itemClicked.connect(self.decomposeBlock)
```

Рисунок 3.2 – метод setupUi

Даний метод є методом, який відповідає за графічну складову даного програмного продукту – інтерфейс. Варто зазначити, що даний метод отримує властивості модуля, на базі якого він побудований – модуля PyQt, за допомогою передачі об’єкту MainWindow до представленого методу. Даний об’єкт відповідає за генерацію головного (батьківського) вікна програми на якому власне і розташовані усі графічні елементи керування програмою – кнопки, поля вводу/виводу тощо.

Метод MainWindow.setObjectName(«MainWindow») – задає ім’я батьківському вікну, а метод MainWindow.resize(906, 869) відповідає за розміри даного вікна, відповідно ширина встановиться у розмірі 906 пікселів, а висота 869 пікселів.

Далі завдяки об’єкту sizePolicy вже генерується поведінка вікна відповідно до змін його розмірів, а саме: вирівнювання контрольних елементів, їх розташування відносно одне одного тощо.

Тепер на прикладі елементу list розглянемо основні властивості елементів керування, що представлені у графічній бібліотеці PyQT:

- QtWidgets.QtListWidget(MainWindow) – виконує створення елементу списку за допомогою конструктору QtWidgets, що має метод QtListWidget, який власне і відповідає за створення графічного елементу списку, приймаючи на вхід функції вікно до якого і буде закріплений – у нашому випадку це батьківське вікно MainWindow;

- setGeometry(QtCore.QRect(10, 10, 861, 181)) – задає розмір для елемента, що буде відображений за допомогою звернення до вбудованого класу QtCore, що має метод Qrect, який приймає на свій вхід координати верхньої лівої точки вікна та правої нижньої точки вікна;

- setObjectName(«list») – задає ім’я для створюваного елементу задля того, що даний об’єкт міг бути розпізнаний у контексті програми;

- addItems(Sboxes) – додає до даного списку набір з 4 S-блоків, на основі яких синтезується високоякісний S-блок та які буде відображено у графічному інтерфейсі.

- itemClicked.connect(self.decomposeBlock) – закріплює обробник події натискання на елемент списку шляхом прив'язки до події матоду decomposeBlock (рисунок 3.2)

```
def decomposeBlock(self):
    Sbox = eval(self.list.currentItem().text())
    funcs = base2.funcs2int(base4.divideToFunctions(Sbox, False, len(Sbox)))

    self.decomposes.clear()
    self.decomposes.addItems([str(f) for f in funcs])
```

Рисунок 3.2 – Програмний код функції decomposeBlock()

Даний код виконує декомпозицію обраного S-блоку на 4 компоненті 4-функції. Розглянемо даний метод більш детально. Перший рядок коду функції виконує перетворення обраного текстового рядку у список шляхом інтерпретації тексту як програмного коду задля того, щоб було можливо проводити необхідні

математичні операції. З операцію перетворення відповідає нативна функція мови Python eval().

Далі завдяки власноруч спроектованому модулю base4 та його методу divideToDunctions() виконується декомпозиція S-блоку на його складові 4-функції. Метод divideToDunctions() приймає три вхідних аргументи:

- S-блок, що буде декомпозуватись і представлений у вигляді списку;

- булева змінна, що надає змогу повертати найстаршу компоненту функцію, якщо вона виставлена у значенні істини;

- довжина S-блоку, який буде представлено у вигляді компонентних 4-функцій.

Далі матриця отриманих значень перетворюється зі строкового формату до числового завдяки ще одній функції написаного модулю base2 – funcs2int(). Наступним кроком виконується очистка списку у графічному інтерфейсі, що репрезентує S-блок у вигляді компонентних 4-функцій. За це відповідає рядок self.decomposes.clear(), де викликається вбудований до класу бібліотеки PyQt метод clear(), що видаляє всі значення списку.

Після описаних вище процедур виконується заповнення списку decompose отриманими значеннями, завдяки вбудованому методу addItems, попередньо виконавши перетворення матриці цілих чисел у матрицю строкових представлень цілих чисел для можливості коректного відображення у графічному інтерфейсі.

Що ж стосується методу calculateResults() (рисунок 3.3), то тут виконуються усі процедури, що стосуються обчислення криптографічних характеристик досліджуваного S-блоку.

```
def calculateResults(self):
    content = self.list.currentItem().text()
    Sbox = eval(content)

    current_mode = self.dropdown.currentText()

    if current_mode == '2-logic':
        self.base2results(Sbox)

    else:
        self.base4results(Sbox)
```

Рисунок 3.3 – Програмний код функції calculateResults()

Першим кроком даний метод отримує поточне значення обраного елементу списку (S-блок), яке представлено строковому форматі, після чого за допомогою методу eval() інтерпретується у якості програмного коду і перетворюється на список цілих значень в діапазоні від 0 до 255.

Змінна current_mode виступає у якості маркеру задля того, щоб програма розуміла з якою саме системою числення ми працюємо. Значення вона бере з випадаючого списку dropdown, який і дозволяє користувачу обирати режим обчислювання. Якщо обрано значення, що є маркером булевої логіки, то викликається метод base2results(), який обчислює СЛК, відстань нелінійності та кореляційний імунітет у сенсі булевої логіки, інакше – викликається метод base4results, який калькулює ті ж самі значення, але вже у представленні S-блоку у якості набору 4-функцій.

Розглянемо роботу методу base2results () (рисунок 3.4).

```
def base2results(self, Sbox):
    f2 = findWeight2base.calculateWeight(Sbox).createFuns()
    self.SAC_input.setText(matrixToStr.matrixToStr(f2))

    n2 = nonlin.calculateSbox(Sbox)
    self.nonlin_input.setText(str(min(n2)))

    c2 = ci2.correlationCalc(Sbox)
    self.CI_input.setText(matrixToStr.matrixToStr(c2))
```

Рисунок 3.4 – Програмний код методу base2results()

Спочатку дана функція отримує доступ до модулю findWeight2base, який і виконує розрахунки матриці суворого лавинного критерію. Далі отриманий результат за допомогою функції, що вбудована до PyQt – setText() записує отримані дані задля того, щоб вивести їх у графічний інтерфейс. Наступним кроком виконується виклик модулю nonlin, який відповідає за розрахунок відстані нелінійності поточного S-блоку. Знову відбувається виклик методу setText і вже на цей раз виконується додавання результатів відносно відстані нелінійності до графічного інтерфейсу.

Останній блок роботи функції відповідає за розрахунок матриці кореляції отриманого S-блоку. Дана процедура відбувається за рахунок виклику модулю ci2, де вже по аналогії з попередніми частинами коду виводить результат у користувацький інтерфейс за допомогою функції setText().

Що ж стосується функції base4results (рисунок 3.5), то вона працює аналогічним чином.

```
def base4results(self, Sbox):
    f4 = findWeight4base.calculateWeight(Sbox).createFuns()
    self.SAC_input.setText(matrixToStr.matrixToStr(f4))

    n4 = nonlin4.calculateSbox(Sbox)
    self.nonlin_input.setText(str(round(min(n4))))

    c4 = ci2.correlationCalc(Sbox)
    self.CI_input.setText(matrixToStr.matrixToStr(c4))
```

Рисунок 3.5 – Програмний код методу base4results()

Представлений на рисунку 3.5 код повністю дублює логіку роботи своєї попередниці base2results() за тим винятком, що тут відбуваються розрахунки вже представленні S-блоку вже у якості компонентних 4-функцій (окрім матриці кореляції, так як її матриця незалежно від представлення завжди однакова).

Особливу увагу варто приділити методу genereteGood, що синтезує високоякісний S-блок (рисунок 3.6).

```
def generateGood(self):
    self.list.clear()
    self.list.addItem(str(decompose_method.generateBlock()))
```

Рисунок 3.6 – Програмний код методу generateGood()

Першим кроком відбувається очистка списку, який до натискання кнопки «Generate» містив у собі 4 S-блоки, що були задані як константи. Далі вже відбувається додавання згенерованого високоякісного S-блоку за допомогою методу PyQt addItem().

3.3 Модуль генерації високоякісного S-блоку

Так як робота методу generateGood(), що був описаний у минулому розділі цілком спирається на роботу модуля decompose_method варто розглянути його більш детального. Основна мета цього методу як раз практично показати роботу синтезу високоякісних S-блоків на прикладі 4 статично заданих S-блоків.

Сам модуль містить у собі всього один метод generateBlock(), який розподіляється на декілька логічних блоків. Перший блок (рисунок 3.7) має на меті перетворити усі S-блоки на набір 4-функцій.

```
funs4 = []

for Sbox in Sboxes:
    strFuns = base4.divideToFunctions(Sbox, False, 256)

    for strFun in strFuns:
        funs4.append([int(num) for num in strFun])

goodFuns = []
```

Рисунок 3.7 – перший логічний блок

Спочатку у межах циклу перебираються усі S-блоки, після чого за допомогою використання модулю base4 та його методу divideToFunctions відбувається декомпозиція поточного блоку на 4-функції. Наступним кроком вже перебираються усі отримані 4-функції задля їх запису у загальний список funs4, що буде містити усі функції отримані після розкладання блоків.

Друга логічна частина програми (рисунок 3.8) вже направлена на роботу з безпосередньо 4-функціями, що були отримані у першому блоці.

```
for fun in funs4:
    binfuns = base4ToBase2(fun)

    b1 = findWeight2baseFUN.calculateWeight(binfuns[0])
    b1.calculateFun()

    b2 = findWeight2baseFUN.calculateWeight(binfuns[1])
    b2.calculateFun()

    if b1.presentMatrix() == [128, 128, 128, 128, 128, 128, 128, 128] and b2.presentMatrix() == [128, 128, 128, 128, 128, 128, 128, 128]:
        goodFuns.append(fun)

goodFuns = np.unique(goodFuns, axis=0)
length = len(goodFuns)
```

Рисунок 3.8 – Другий логічний блок функції

Спочатку перебирається увесь набір отриманих 4-функцій, кожна з яких в свою чергу декомпозується на дві булеві функції, які записується до списку binfuns. Далі кожна з цих функцій перевіряється на відповідність вимогам СЛК, використовуючи метод calculateWeight, що належить до модуля findWeight2base. Останньою дією у циклі проводиться порівняння отриманої матриці значень з необхідною (вісім значень 128 у матриці). Якщо обидві булеві функції відповідають вимогам, то їх батьківська 4-функція заноситься до списку goodFuns.

Далі список goodFuns перевіряється на вміст дублікатів, які видаляються за допомогою методу unique(), що вбудований у бібліотеку numpy. Останнім кроком даного модулю є отримання довжини списку goodFuns.

Третій і останній логічний блок (рисунок 3.9) виконує основне завдання – синтезує високоякісний S-блок.

```
for i in range(0, length):
    temp1 = goodFuns[i]
    for j in range(i+1, length):
        temp2 = goodFuns[j]
        for k in range(j+1, length):
            temp3 = goodFuns[k]
            for l in range(k+1, length):
                temp4 = goodFuns[l]
                Sbox = base4.compose([
                    temp1,
                    temp2,
                    temp3,
                    temp4
                ])
                Su = set(Sbox)
                if (len(Su) == 256):
                    return Sbox
```

Рисунок 3.9 – Третій логічний блок функції

Даний блок є комбінацією декількох вкладених циклів. Фактично все зводиться до того, що пробігти фактично усі можливі поєднання блоків з метою синтезу високоякісного блоку. На кожній ітерації кожного циклу створюється змінна, індекс якої більше на 1 від попередньої, відповідно кожна нова 4-послідовність береться одразу після попередньої.

У останньому внутрішньому циклі відбувається композиція S-блоку з 4-функцій, які були отримані на попередніх кроках. За допомогою функції base4 та її методу compose(), що якраз і відповідає за перетворення наданого набору 4-функцій на один S-блок.

Отриманий S-блок оброблюється за допомогою вбудованої функції мови Python set(), що залигає лише унікальні значення, що ідеально підходить для перевірки бієктивності (наявності унікальних чисел в діапазоні від 0 до 255). Таким чином, якщо отримана послідовність проходить перевірку довжини на відповідність значенню 256, то ми будемо вважати таку послідовність бієктивною і таким чином отримуємо високоякісний S-блок (рисунок 3.10).

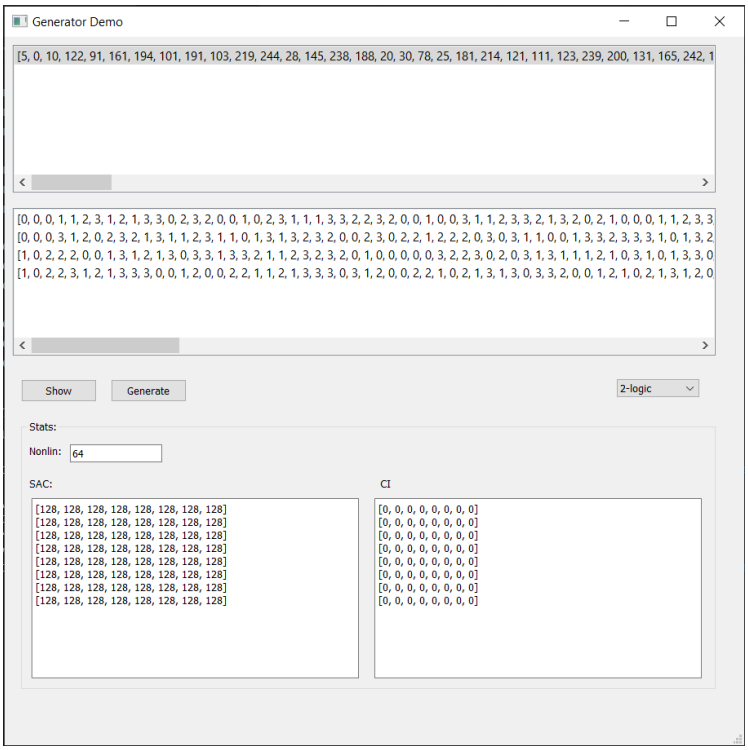

Рисунок 3.10 – Інтерфейс програми зі згенерованим високоякісним S-блоком.

3.4 Практичне використання

Для перевірки актуальності побудованої множини високоякісних S-блоків було модифіковано алгоритм AES за допомогою S-блоків з МЛК 2-логіки та СЛК 4-логіки, а також за допомогою S-блоків з СЛК 2- та 4-логіки. Для зручності будемо називати їх AES, AES-1 та AES-2 відповідно. У якості перевірки використаємо файл з розширенням .exe (рисунок 3.11) розміру 1.33 Мб та зашифруємо його за допомогою усіх трьох варіантів алгоритму та перевіримо результати роботи, використовуючи тести якості від NIST.

Рисунок 3.11 – Тестовий файл

Дані порівняння результатів роботи приведено у таблицях 3.1 -3.3 , де PASS – пройдено, FAIL – провалено.

Таблиця 3.1 – Результати роботи класичного AES

Тест	Значення	Статус
Однакові біти	0.5204028429914505	PASS
Частотний блоковий	0.2958396608331729	PASS
Найдовша послідовність однакових значень	0.370829727170488	PASS
Найдовша послідовність одиниць	0.30229178236946214	PASS
Ранг бінарних матриць	0.8290388802137041	PASS
Спектральний	0.596452746002457	PASS
Непересічені шаблони	0.9997893007338	PASS

Продовження таблиці 3.1

Тест	Значення	Статус
Пересічені шаблони	0.5515309572181177	PASS
Універсальний тест Маурера	0.7751898821461352	PASS
Лінійна складність	0.8080513506590122	PASS
Підпослідовності	0.08496420524163752	PASS
Апроксимована ентропія	0.2825119133044632	PASS
Кумулятивні суми	0.1406216113867469	PASS
Довільне відхилення	0.002560236460540112	FAIL
Довільне варіативне відхилення	0.11945350158090245	PASS

Таблиця 3.2 – Результати роботи AES-1

Тест	Значення	Статус
Однакові біти	0.7747862541735437	PASS
Частотний блоковий	0.9218271109557018	PASS
Найдовша послідовність однакових значень	0.7420408084866765	PASS
Найдовша послідовність	0.9159415448967893	

одиниць		PASS
Ранг бінарних матриць	0.5583784623058192	PASS
Спектральний	0.8884030266963212	PASS
Непересічені шаблони	0.9999747313997819	PASS
Пересічені шаблони	0.12948968748163295	PASS
Універсальний тест Маурера	0.8835746655887791	PASS
Лінійна складність	0.006304389728741894	FAIL
Підпослідовності	0.3443740556249118	PASS
Апроксимована ентропія	0.6479305979729352	PASS
Кумулятивні суми	0.499041332863583	PASS
Довільне відхилення	0.11578882878900323	PASS
Довільне варіативне відхилення	0.20579609177663172	PASS

Таблиця 3.3 – Результати роботи AES-2

Тест	Значення	Статус
Однакові біти	0.9737917962925495	PASS

Продовження таблиці 3.3

Тест	Значення	Статус
Частотний блоковий	0.4638476263796343	PASS
Найдовша послідовність однакових значень	0.0517294209372034	PASS
Найдовша послідовність одиниць	0.9159415448967893	PASS
Ранг бінарних матриць	0.9173301366568777	PASS
Спектральний	0.8547347928040594	PASS
Непересічені шаблони	0.9781112038747598	PASS
Пересічені шаблони	0.05536471990878023	PASS
Універсальний тест Маурера	0.7680704135173523	PASS
Лінійна складність	0.9366351221016609	PASS
Підпослідовності	0.610153193719409	PASS
Апроксимована ентропія	0.6654770475404569	PASS
Кумулятивні суми	0.9931040124038137	PASS
Довільне відхилення	0.32120675775034657	PASS
Довільне варіативне відхилення	0.13990915398451198	PASS

Неважко помітити, що AES на базі блоку, синтезованого у цій роботі показує значно кращі результати, ніж його попередники. Для більшої наочності дані порівняння наведені у таблиці 3.4.

Таблиця 3.4 – Порівняння роботи 3 варіантів AES

Тест	Значення	Переможець
Однакові біти	0.9737917962925495	AES-2
Частотний блоковий	0.9218271109557018	AES-1
Найдовша послідовність однакових значень	0.0517294209372034	AES-2
Найдовша послідовність одиниць	0.30229178236946214	AES
Ранг бінарних матриць	0.9159415448967893	AES-2
Спектральний	0.8884030266963212	AES-1
Непересічені шаблони	0.9997893007338	AES

Продовження таблиці 3.4

Тест	Значення	Переможець
Пересічені шаблони	0.05536471990878023	AES-2
Універсальний тест Маурера	0.8835746655887791	AES-1
Лінійна складність	0.9366351221016609	AES-2
Підпослідовності	0.610153193719409	AES-2
Апроксимована ентропія	0.6654770475404569	AES-2
Кумулятивні суми	0.9931040124038137	AES-2
Довільне відхилення	0.32120675775034657	AES-2
Довільне варіативне відхилення	0.20579609177663172	AES-1

На практиці було доведено, що синтезовані S-блоки мають досить високі характеристики, а враховуючи результати тестів NIST, можна підсумувати, що AES на основі блоку синтезованого у даній роботі показав себе найкраще у 9/16 тестів, AES-1 у 4/16, а AES 2/16. До того ж, відповідно до результатів таблиць 3.1-3.3 можна помітити, що тільки AES на основі високоякісного S-блоку пройшов

усі тести з 16 доступних. Це ще раз доводить його переваги у порівнянні з аналогами.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було:

- досліджено S-блоки, що побудовані на теорії динамічного хаосу на предмет їх відповідності вимогам СЛК, кореляційного імунітету та відстані нелінійності, а також обрано найкращий з них;
- розроблено метод синтезу високоякісних S-блоків, що мають відповідність СЛК одразу у двох системах числення – булевої та 4 логіки, наявність кореляційного імунітету у сенсі булевої логіки та відстань нелінійності що складає вище 57% від максимально можливого значення. До того ж кількість таких S-блоків досить значна і перевищує 100 тисяч.
- проведено емпіричні дослідження роботи побудованих S-блоків у складі криптоалгоритму AES, де синтезований S-блок показав кращі результати, ніж AES з оригінальним S-блоком.

ПЕРЕЛІК ПОСИЛАНЬ

1. Соколов А.В., Радуш В.В. Метод синтезу кореляційно-імунних S-блоків, що відповідають суворому лавинному критерію при представленні функціями багатозначної логіки. Інформатика та математичні методи в моделюванні. 2022. Прийнята до друку.

2. Sokolov A.V., Radush V.V. Cryptographic properties of s-boxes constructed on the basis of dynamic chaos theory when represented using many-valued logic functions. Journal of Discrete Mathematical Sciences & Cryptography. Accepted for the publication.

3. Advanced Encryption Standard (AES). GeeksForGeeks. A computer science portal for geeks. URL: https://www.geeksforgeeks.org/advanced-encryption-standard-aes/

4. Shannon, C.E. A Mathematical Theory of Cryptography. Bell *System Technical Memo* MM 45-110-02., 1945. 132 p.

5. FIPS 197. Advanced encryption standard. URL: http://csrc.nist.gov/publications/

6. Zaibi G., Peyrard F., Kachouri A., Fournier-Prunaret D. On dynamic chaotic S-Box. Information Infrastructure Symposium, 2009. GIIS '09. Global. URL: https://www.researchgate.net/publication/224611268_On_dynamic_chaotic_S-Box

7. Жданов О.Н. Методика выбора ключевой информации для алгоритма блочного шифрования. М.: ИНФРА-М, 2013. 90 с.

8. Соколов, А.В. Новые методы синтеза нелинейных преобразований современных шифров. Lap Lambert Academic Publishing, Germany, 2015. 100 с.

9. Горбенко І.Д., Потій О.В., Ізбенко Ю.А. Дослідження аналітичних і статистичних властивостей булевих функцій криптоалгоритму RIJNDAEL (FIPS 197). *Радіотехніка: всеукр. міжвідом. наук.-техн. зб.* Харків, 2004 . Т. 126, С. 132 – 138.

10. Мазурков М.И., А.В. Соколов. Криптографические свойства нелинейного преобразования шифра Rijndael на базе полных классов

неприводимых полиномов. *Праці Одеського політехнічного університету,* 2012. №2(39), С. 183–189.

11. Sokolov A.V., Zhdanov O.N. Prospects for the Application of Many-Valued Logic Functions in Cryptography. *International Conference on Theory and Applications of Fuzzy Systems and Soft Computing*, 2018. P. 331–339.

12. Строгий лавинный критерий. *Вікіпедія:* веб-сайт. URL: https://ru.wikipedia.org/wiki/Лавинный_эффект

13. Sokolov A.V., Zhdanov O.N. Strict avalanche criterion of four-valued functions as the quality characteristic of cryptographic algorithms strength.. *Siberian Journal of Science and Technology*, 2019. Vol. 20. No. 2, P. 183–190.

14. Логачев О.А., Сальников А.А., Ященко В.В. Булевы функции в теории кодирования и криптологии. М: Издательство МЦНМО, 2004. 472 с.

15. Sokolov A.V., Zhdanov O.N., Regular synthesis method of a complete class of ternary bent-sequences and their nonlinear properties, *Journal of Telecommunication, Electronic and Computer Engineering*, vol. 8 no. 9, pp. 39-43, 2016.

16. Sokolov A.V., Zhdanov O.N. Strict avalanche criterion of four-valued functions as the quality characteristic of cryptographic algorithms strength. *SJST*, 2019. Vol. 20, No. 2. P.183–190.

17. Kazakova Nadiia, Sokolov Artem, Troyanskiy Alexander. Correlation Immunity of Many-Valued Logic Component Functions of Modern Cryptographic Algorithm S-Boxes. *International Scientific and Practical Conference «Intellectual Systems and Information Technologies»*: Conference Proceedings / Odessa State Environmental University. Odessa, 2021. P. 268-275.

18. Kim K., Matsumoto T., Imai H. A recursive construction method of S-boxes satisfying strict avalanche criterion. *Proc. of CRYPTO'90, Springer. Verlag*. 1990. P.565–574

19. Соколов, А.В. Конструктивный метод синтеза нелинейных S-блоков подстановки, соответствующих строгому лавинному критерию. *Известия высших учебных заведений. Радиоэлектроника. 2013*. Т. 56, N. 8, С. 43–52.

20. Лоскутов А. Нелінійна динаміка, теорія динамічного хаосу та синергетика (перспективи та додатки) "Компьютерра" № 47, 1998.

21. Ahmad M., Chugh H., Goel A., Singla P. A Chaos Based Method for Efficient Cryptographic S-box Design. International Symposium on Security in Computing and Communication. 2013. p 130–137.

22. Y. Tian, L. Zhimao, Chaotic S-Box: Intertwining Logistic Map and Bacterial Foraging Optimization, Mathematical Problems in Engineering, pp. 1-11, 2017. DOI: 10.1155/2017/6969312

23. E. Tanyildizi, F. Özkaynak, A New Chaotic S-Box Generation Method Using Parameter Optimization of One Dimensional Chaotic Maps, IEEE Access, vol. 7, pp. 117829-117838, 2019. DOI: 10.1109/access.2019.293644

24. S. Farwa, T. Shah, N. Muhammad et al., An Image Encryption Technique based on Chaotic S-Box and Arnold Transform. International Journal of Advanced Computer Science and Applications, Vol. 8, No. 6, pp.360-364, 2017. DOI: 10.14569/ijacsa.2017.080647

25. Q. Lu, C. Zhu, X. Deng, An Efficient Image Encryption Scheme Based on the LSS Chaotic Map and Single S-Box. IEEE Access, vol. 8, pp. 25664-25678, 2020. DOI: 10.1109/ACCESS.2020.2970806

26. M. Asim, V. Jeoti, Efficient and Simple Method for Designing Chaotic S-Boxes. ETRI Journal, vol. 30, no. 1, pp. 170-192, 2008. DOI: 10.4218/etrij.08.0207.0188

27. J. Wang, Y. Zhu, C. Zhou, Z. Qi, Construction Method and Performance Analysis of Chaotic S-Box Based on a Memorable Simulated Annealing Algorithm, Symmetry, 2020. DOI: 10.3390/sym12122115

28. Lambić D. S-box design method based on improved one-dimensional discrete chaotic map. Journal of Information and Telecommunication, vol. 2, issue 2, pp. 181-191, 2018. DOI: 10.1080/24751839.2018.1434723

29. Lu Q., Zhu C., Wang G. A Novel S-Box Design Algorithm Based on a New Compound Chaotic System, Entropy, no. 21(10), pp. 1004, 2019. DOI: 10.3390/e21101004

30. Q. Lai, A. Akgul, C. Li, G. Xu, U. Çavuşoğlu, A New Chaotic System with Multiple Attractors: Dynamic Analysis, Circuit Realization and S-Box Design, Entropy, no. 20(1), pp. 12, 2018. DOI: 10.3390/e20010012

31. I. Hussain, A. Anees, T. Al-Maadeed, M. Mustafa, Construction of S-Box Based on Chaotic Map and Algebraic Structures. Symmetry, no. 11(3), pp. 351, 2019. DOI: 10.3390/sym11030351