Міністерство освіти і науки України

Харківський національний університет імені В.Н. Каразіна

Факультет комп'ютерних наук

Спеціальність 125 «Кібербезпека»

Освітня програма «Безпека інформаційних та комунікаційних систем»

«Допущено до захисту»

Зав.кафедрою БІСТ

Сергій РАССОМАХІН

_____

« » 2022 р.

Пояснювальна записка

до кваліфікаційної роботи магістра

на тему: «Евристичні методи генерації нелінійних вузлів заміни блокових симетричних шифрів»

оцінка « »

Голова ЕК

Доценко С.І. _____

Керівник проф. Кузнецов О. О.

Рецензент проф. Толстолузька Є. Г.

Виконавець : студент(ка) групи КБ-61

Заїченко Ю. О.

Харків – 2022

РЕФЕРАТ

Дипломна робота обсягом 59 сторінок містить: 2 таблиць, 10 рисунків та 45 джерел посилань.

Робота складається зі вступу, 3 розділів, висновків та переліку джерел посилань.

Об'єкт дослідження: евристичні методи генерації нелінійних вузлів ускладнення.

Предмет досліджень: Принципи реалізації методів генерації нелінійних вузлів заміни та методики порівняльного аналізу цих методів.

Мета досліджень: Аналіз евристичних методів генерації нелінійних вузлів ускладнення для досягнення більш швидкої генерації високонелінійних S-блоків та дослідження критеріїв та показників ефективності.

Актуальність дослідження полягає в поглибленому вивченні такого методу генерації нелінійних вузлів заміни з сімейства алгоритмів локального пошуку, як Hill Climbing.

Методи проведення досліджень. Аналіз та порівняльні дослідження методів генерації нелінійних вузлів заміни, порівняльні дослідження методів генерації нелінійних вузлів заміни, розробка математичної моделі та удосконаленого методу Hill Climbing для генерації нелінійних вузлів заміни.

Основні завдання досліджень:

- Проаналізувати критерії та показники ефективності нелінійних вузлів заміни блокових симетричних шифрів та їх зв'язок із ефективністю криптоперетворень;
- Провести порівняльні дослідження методів генерації нелінійних вузлів заміни;

- Виконати розробку математичної моделі та удосконаленого методу Hill Climbing для генерації нелінійних вузлів заміни.
- Описати удосконалений метод Hill Climbing для генерації нелінійних вузлів заміни математично.

У першому розділі описано поняття симетричного блокового шифру та показники, що його характеризують.

У другому розділі описано деякі існуючі евристичні методи генерації нелінійних вузлів ускладнення, детально розглянуто принцип їх реалізації.

У третьому розділі проаналізовано метод сходження на пагорб, описано реалізацію удосконаленого методу сходження на пагорб для генерації нелінійних вузлів заміни та наведено статистичні дані щодо генерації.

У ході роботи розроблено програмну реалізацію удосконаленого методу градієнтного спуску для генерації нелінійних вузлів заміни. Також проведено експериментальні дослідження даного методу з точки зору швидкодії та оптимальності згенерованих нелінійних вузлів заміни, проведено дослідження щодо найбільш безпечних значень критеріїв та показників.

У результаті проведеної роботи зроблені висновки щодо застосування евристичних методів для генерації S-блоків для симетричних криптоперетворень та розроблені рекомендації щодо практичного застосування і вибору критеріїв та показників ефективності нелінійних вузлів ускладнення.

Ключові слова: ГРАДІЄНТНИЙ СПУСК, СИМЕТРИЧНЕ КРИПТОПЕРЕТВОРЕННЯ, БЛОКОВІ СИМЕТРИЧНІ ШИФРИ, ЕВРИСТИЧНІ МЕТОДИ, S-БЛОКИ, БУЛЕВІ ФУНКЦІЇ, МЕТОДИ ГЕНЕРАЦІЇ ВУЗЛІВ, НЕЛІНІЙНІ ВУЗЛИ.

ABSTRACT

The qualifying work of a master, 59 pages, 3 tables, 10 images and 45 reference sources.

The work consists of an introduction, 3 chapters, conclusions and a list of reference sources.

The object of the research are heuristic methods of generating non-linear complication nodes.

The subject of the research are principles of implementation of methods of generation of non-linear replacement nodes and methods of comparative analysis of these methods.

The purpose of the research is analysis of heuristic methods of generating non-linear complication nodes complications to achieve faster generation of highly nonlinear S-boxes and research of criteria and performance indicators.

The relevance of the research lies in the in-depth study of such a method of generating nonlinear replacement nodes from the family of local search algorithms, such as Hill Climbing.

The methods of the research. Analysis and comparative studies of methods for generating nonlinear replacement nodes, comparative studies of methods for generating nonlinear replacement nodes, development of a mathematical model and an improved Hill Climbing method for generating nonlinear replacement nodes.

The main tasks of research:

- To analyze the criteria and performance indicators of non-linear nodes for replacing block symmetric ciphers and their relationship with the effectiveness of crypto-transformations;

- Conduct comparative studies of methods of generating non-linear replacement nodes;

- Develop a mathematical model and an improved Hill Climbing method for generating non-linear replacement nodes.

- Describe the improved Hill Climbing method for generating non-linear substitution nodes mathematically.

The first chapter describes the concept of a symmetric block cipher and its characteristics.

In the second chapter, some existing heuristic methods of generating nonlinear complication nodes are described, and the principle of their implementation is considered in detail.

The third chapter analyzes the hill-climbing method, describes the implementation of the improved hill-climbing method for the generation of non-linear replacement nodes, and provides statistical data on the generation.

In the course of the work, a software implementation of the improved gradient descent method for the generation of non-linear substitution nodes was developed. Experimental studies of this method were also carried out from the point of view of speed and optimality of generated non-linear replacement nodes, research was carried out on the safest values of criteria and indicators.

As a result of the work, conclusions were drawn regarding the use of heuristic methods for generating S-blocks for symmetric cryptotransformations, and

recommendations were developed for the practical application and selection of criteria and efficiency indicators of nonlinear complication nodes.

Keywords: HILL CLIMBING, SYMMETRICAL CRYPTOCONVERSION, BLOCK SYMMETRICAL CIPHERS, HEURISTIC METHODS, S-BOXES, BOOLEAN FUNCTIONS, NODE GENERATION METHODS, NON-LINEAR NODES.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

ACS	–	Ant Colony System
CO	–	задача комбінаторної оптимізації
GF	–	поле Галуа
HC	–	hill-climbing (метод сходження на пагорб)
PCP	–	посткросоверна популяція
PMX		частково відображений кросовер
PRNG	–	генератор псевдовипадкових чисел
SA	–	simulated annealing (метод імітованого відпалу)
TSP	–	проблема комівояжера
WHS	–	Walsh-hadamard sequences (послідовність Волша-Адамара)
АНФ		алгебраїчна нормальна форма
БСШ	–	блочний симетричний шифр
ГОСТ	–	Державний стандарт
ДСТУ	–	Державний стандарт України
ОС	–	операційна система
ПК	–	персональний комп’ютер
ЦП	–	центральний процесор

ВСТУП

У системах захисту інформації значного розвитку та використання зазнали симетричні криптографічні засоби. І при формуванні шифру із симетричним ключем однією з найбільш ресурсозатратних задач є генерація нелінійних підстановок, які є криптографічно стійкими.

Генерація нелінійних вузлів заміни для криптозасобів відбувається за допомогою різних методів. Проте найперспективнішим для формування високонелінійних S-блоків вважається напрям евристичних методів. До них, зокрема, належать алгоритми локального пошуку. Вони мають дві головні переваги: по-перше, вони вимагають дуже мало пам’яті і, по-друге, вони часто можуть знаходити значущі рішення у великих або нескінченних (безперервних) просторах станів, для яких систематичні алгоритми не підходять. Окрім пошуку цілей, алгоритми локального пошуку також корисні для розв’язання задач чистої оптимізації, де метою є пошук найкращого стану відповідно до цільової функції.

Актуальність даного дослідження полягає в поглибленому вивченні методів генерації нелінійних вузлів заміни, удосконалення методу сходження на пагорб с точки зору швидкодії: оптимізація евристичних методів задля забезпечення більш швидкого процесу генерації високонелінійних S-блоків.

На меті стояло докладне дослідження евристичних методів генерації нелінійних вузлів заміни і аналіз їх реалізації, розробка удосконаленого методу сходження на пагорб задля досягнення більшої швидкості генерації нелінійних вузлів заміни. Було розглянуто різні евристичні техніки генерації нелінійних вузлів підстановки, зокрема:

- Алгоритм імітованого відпалу
- Алгоритм сходження на пагорб
- Метод градієнтного спуску

- Меметичні алгоритми такі як генетичний алгоритм
- Мурашині алгоритми тощо.

У ході дослідження було застосовано такі методи, як аналіз та порівняльні дослідження методів генерації нелінійних вузлів заміни, розробка математичної моделі та удосконаленого методу Hill Climbing для генерації нелінійних вузлів заміни.

1 СИМЕТРИЧНІ БЛОКОВІ ШИФРИ

Комп’ютери та, загалом, електронні засоби передачі, зберігання та обробки інформації відіграють усе більшу роль у сучасному суспільстві. Для того, щоб інформаційні технології використовувалися в різних сферах, необхідно забезпечити їх надійність і безпеку. Безпека стосується здатності інформаційної системи зберігати свою цілісність і працездатність під випадковим або навмисним зовнішнім впливом.

Симетричні блокові шифри в даний час займають передову позицію в криптографії, і їх використання виходить далеко за рамки забезпечення конфіденційності. Хеш-функції [11], генератори псевдовипадкової послідовності [34] і коди автентифікації повідомлень [36] були побудовані на основі компонентів симетричних блокових шифрів, що дозволяє використовувати численні режими блокового шифру для ефективного перетворення одного примітиву. Різні призначення, роблять його універсальним.

1.1 Базові принципи розробки шифрів

Варто звернути увагу на те, що процес криптографічного захисту інформації здійснюється за заздалегідь заданою схемою шифрування, яку можна представити у вигляді алгебраїчної моделі. [22]

Спільним для усіх блочних шифрів є те, що для забезпечення безпеки необхідна нелінійна операція/елемент. Нехай X, Y і Q – деякі скінченні множини, відповідно, відкритого і зашифрованого тексту, а також ключові множини. Введемо такі $K1, K2 \in Q$, що на добутках $X \times Q$ і $Y \times Q$ множин X, Q і Y задано таку пару функції відображень $E_{K1} : X \times Q \to Y$ і $D_{K2} : Y \times Q \to X$ відповідно. До того ж, відображення E_{K1} визначає метод зашифрування відкритого тексту, а відображення D_{K2} – метод розшифрування зашифрованого тексту.

Алгебраїчною моделлю шифру є $\sum = (X, Y, Q, E_{K1}, E_{K2})$,

За умови, що виконуються наступні тези:

1) $\forall x \in X, \forall K1, K2 \in Q : D_{K2}(E_{K1}(x)) = x.$

2) $Y = \bigcup_{K1 \in Q} E_{K1}(X).$

У симетричних блокових шифрах в зашифрувальних і розшифрувальних процесах використовується один і той самий ключ $K = K1 = K2,$ враховуючи цей факт, отримуємо алгебраїчну модель у такому вигляді

$$\sum = (X, Y, Q, E_K, D_K). \quad (1.1)$$

Додатковим параметром для узагальненої алгебраїчної моделі симетричного блокового шифрування (1.1) інколи може бути введена також множина управління станом шифрувального процесу, позначена як T, елементи якої можуть бути відкритими та надавати блоковому шифру властивості випадковості [22].

Іншими словами, функції відображення, які складають процеси шифрування та дешифрування відповідно, приймають форму $E_K : X \times Q \times T \to Y$ і $D_K : Y \times Q \times T \to X$ відповідно, а алгебраїчна модель шифру тоді подається у наступному вигляді

$$\sum = (X, Y, Q, T, E_K, D_K). \quad (1.2)$$

Для одного і того ж блоку відкритого тексту можна отримати різні блоки зашифрованого тексту, використовуючи різні значення елементів набору T і один і той же незмінний секретний ключ K.

Вперше базові принципи, використовувані при розробці шифрів, були оприлюднені і визначені у роботі Огюста Керкгоффса [22]. В першу чергу, шифри слід розглядати у контексті їх поширеного використання, коли той самий метод шифрування використовується значною кількістю користувачів. Потім, коли важко довести теоретичну стабільність запропонованої криптосистеми, використовуються емпіричні методи дослідження, засновані на виконанні криптографічних атак, іншими словами з використанням криптоаналізу. Якщо результати емпіричних досліджень негативні, криптографічний шифр вважається стабільною з огляду на розвиток сучасних електронно-обчислювальних можливостей.

На думку Керкгоффса, криптографічні системи повинні відповідати наступним вимогам, актуальним і сьогодні [22]:

1) Якщо неможливо підтвердити теоретичну стійкість, система повинна бути практично стійкою до атак.
2) Чутливість архітектури системи не має залежати від складності її злому.
3) Перебіг модифікації, передачі та збереження закритих ключів має залишатися простим, без фізичних паперових записів.
4) Захищена інформація має бути переданою виключно через новітні інформаційні та комунікаційні мережі без застосування жодних допоміжних ускладнень.
5) Система – портативна і не потребує будь-яких обслуговуючих операторів.
6) Система повинна бути зрозумілою для використання.

Значний внесок у формулювання принципів на основі розробки криптографічних методів захисту інформації зробив К. Шеннон. У своїй роботі [19] він запропонував класифікацію шифрів і розділив їх на теоретично стійкі, практично стійкі і абсолютно стійкі.

Теоретично стійкий шифр задається у наступному вигляді $(X, Y, Q, E_K, D_K), Y = E_K(X \times Q), \quad X = D_K(Y \times Q)$ із вказаними імовірнісними розподілами $p(x), x \in X$ на X і $p(K), K \in Q$, якщо він досконалий за Шенноном, тобто при будь-якому $y \in Y$ і x [19]

$$p(x/y) = p(x)$$

де $p(x/y)$ – апостеріорна вірогідність отримання відкритого тексту за умови вибраного зашифрованого текст y. Крім того, необхідною і достатньою умовою теоретично стійкого шифру, при будь-якому і є реалізація цієї рівності [38]

$$p(y/x) = p(y)$$

де $p(y/x)$ – умовна ймовірність отримання зашифрованого тексту за умови, що обрано відкритий текст x;

$p(y)$ – імовірність отримання зашифрованого тексту y.

Такі шифри, зашифрований текст яких не містить достатньо інформації для однозначного визначення відповідного публічного тексту або ключа, також є *теоретично стійкими* щодо методів криптоаналізу [12, 32].

Практично стійкими називають шифри, для яких завдання злому еквівалентно протистоянню відомим методам криптографічного аналізу або мають досить велику обчислювальну складність, протягом якого інформація втратить цінність [22].

Абсолютно стабільний визначає такий шифр, де розмір ключової інформації дорівнює або перевищує розмір відкритого тексту, який у результаті буде зашифровуватися, і такий ключ використовується тільки один раз [22]. Абсолютно

стійкий шифр практично не використовується через свою неефективність, адже розмір ключової інформації досить великий і є необхідність у дотриманні високого рівня захищеності такої ключової інформації у період її зберігання.

К. Шеннон також встановив ряд критеріїв, висунутих до блокових шифрів під час їх розробки, які все ще актуальні [12, 22, 32, 38].

1) Секретний ключ повинен бути невеликим за розміром, але в той же час забезпечувати достатній ступінь криптографічної стійкості шифрування.
2) Невелика зміна секретного ключа або відкритого тексту призведе до значних змін у зашифрованому тексті.
3) Будь-який секретний ключ повинен забезпечувати один і той самий рівень криптографічної стійкості шифру.
4) Операції шифрування та дешифрування повинні бути простими задля забезпечення високої швидкості шифрувального процесу.
5) Кількість помилок, що виникають при кодуванні або передачі повідомлень, має бути зведена до мінімуму, щоб вони не ширилися на все повідомлення.
6) Розмір повідомлення після шифрування не збільшується.

Останнім часом до симетричних блокових шифрів висунуто додаткові вимоги [22]:

− підтримка можливості використання секретних ключів змінної довжини;

− підтримка паралельних обчислень;

− підтримка можливості використання криптографічних шифрів на різноманітних апаратних засобах;

− вартість шифрування не повинна перевищувати цінності інформації, що підлягає захисту.

Крім того, криптографічні перетворення, які використовуються для формування симетричних шифрів, повинні забезпечувати два основні принципи –

дифузію (розсіювання) та скремблування (перемішування) для створення криптографічно стабільного шифру.

Принцип скремблювання дозволяє задати складні статистичні залежності між відкритим і зашифрованим текстом. Мета такого перетворення – усунення зв'язку між відкритим і зашифрованим текстом.

Дифузія розуміється як таке криптографічне перетворення, яке дозволяє перерозподіляти надлишковість відкритого тексту шляхом його розподілу по всьому тексту, зашифрованому секретним ключем через вплив одного символу відкритого тексту на значущу кількість символів зашифрованого тексту.

У підсумку, найбільш стійкими шифрами є абсолютно стійкі шифри, але враховуючи, що в абсолютно стійких шифрах $|X| \le |Q|$, то в симетричних блокових шифрах використано секретні ключі K невеликого розміру (порядку 128-256 біт). Це призводить до використання повторюваних наборів операції перетворення даних у методології симетричних блокових шифрів. Шифри, реалізовані з використанням таких методів шифрування називаються ітераційними. Ітераційні методи блокового шифрування базуються на:

− мережі Фейстеля та її модифікаціях [22];

− підстановочно-перестановочної мережі [25, 24];

− еластичної мережі [32];

− неортодоксальних структур [12, 38].

Основу ітераційних СБШ складає ідея К. Шеннона [22], яка має на увазі модель послідовного застосування звичайних криптоперетворень, які дозволяють найшвидше та найефективніше реалізувати перемішування та розсіювання даних. Ці криптографічні перетворення включають в себе одну або декілька різних груп операцій, щоб побудувати криптографічно стійких функцій шифрування даних.

Одноразовий виклик функції шифрування даних називається раундом шифрування. Раундові ключі – різні підключі, що використовуються в кожному з

раундів. [22]. Раундові ключі отримують шляхом перетворення вхідного секретного ключа K на певну кількість підключів, менших або рівних за розрядністю, які використовуються у процессі шифрування даних для забезпечення високого рівня перемішування. У загальному випадку для кожного раунду шифрування використовуються різні раундові ключі, причому розмір раундового ключа може бути набагато більше розміру секретного ключа K, що значно підвищує стійкість процесу кодування даних.

До перетворень, які використовують під час формування раундових ключів, висувають наступні дві вимоги [3, 25]:

1) Процес відновлення секретного ключа шифрування K із заданого підключа має бути достатньо складним, тобто слід уникати лінійних перетворень.
2) Кожен біт секретного ключа шифрування K повинен впливати на кожен підключ, тобто повинен бути забезпеченим лавинний ефект.

Оскільки перетворення забезпечують високу ступінь дисперсії та змішування тексту відкритого та закритого ключів, використовують операції підстановки та перестановки, ефективність яких була продемонстрована К. Шенноном [22]. Крім згаданих вище операцій, використовують також наступні [3, 25]:

− логічні операції (І, АБО, виключне-АБО, інверсія тощо);

− арифметичні операції (додавання і множення за модулем);

− операції зсувів (звичайного і циклічного);

− спеціалізовані логічні функції.

На підставі алгебраїчної структури симетричного блочного шифру, наведених вище принципів побудови симетричного блочного шифру, основних криптографічних перетворень та ітераційної схеми криптоалгоритму можна прийти до наступних висновків.

Характерним недоліком мережі Фейстеля та її модифікацій є те, що розмір блоків даних, які підлягають обробці, не може перевищувати максимальний

бітрейт процесора, тому за один раунд шифруванню підлягає неповний блок відкритого тексту, що несе за собою низький ступінь розсіювання.

При реалізації мережі підстановки-перестановки розробники повинні розміщувати сурогатні таблиці в оперативній пам'яті, частий доступ сповільнить шифрування або обмежить сурогатні таблиці розміром кеша ЦП. Інший підхід полягає у використанні математичних або логічних функцій, які імітують правила підстановки або перестановки. Однак такі функції повинні мати високий ступінь нелінійності, і знайти їх досить складно. Тобто основним недоліком мережі підстановки-перестановки є неефективність її реалізації в сучасних комп’ютерних системах і мережах.

Еластична мережа гарантує подвійну криптографічну стійкість зашифрованого тексту, але, в той же час, час зашифрованого тексту та розмір зашифрованого тексту також збільшуються порівняно з відкритим текстом [32, 38]. Тому існує нагальна потреба в пошуку нових методів шифрування, здатних забезпечити достатньо високий рівень криптографічної стійкості та прості в апаратно-програмній реалізації з урахуванням особливостей сучасних процесорів.

В Україні рекомендовано до використання блоковий симетричний шифр ДСТУ ГОСТ 28147:2009 [26, 28], який був прийнятий на озброєння у 1989 році і вже має нижчу продуктивність, аніж багато сучасних шифрів, включаючи AES [34]. В останні роки успішно проведені теоретичні атаки на криптографічні алгоритмі, що зменшило складність пошуку ключів з 2^{256} до 2^{192} [29]. Однак складність у 2^{192} році недосяжна для сучасних комп'ютерів, що свідчить про практичну захищеність шифру [95].

На сьогодні є два основних перетворення в БСШ: блок заміни (S-блок) і блок перестановки (P-блок) [29, 3]. Можна показати, що будь-яке двійкове перетворення двійкового блоку фіксованої довжини зводиться до S-блоку, але на практиці,

оскільки структура n-розрядного S-блоку є достатньо складною при великих n, використовуються простіші конструкції [1].

У своїй загальній формі S-блок можна представити як декодер (комбінаційна схема), який перетворює n-розрядне двійкове повідомлення в однорозрядне повідомлення на основі 2^n, систему комутаторів внутрішніх з'єднань (всього з'єднань $2^n!$) і шифратор (комбінаційна схема), що переводить повідомлення з однорозрядного 2^n-ричного в n-розрядне двійкове [2]. Процес аналізу n-розрядного S-блоку при великому n вкрай складний, оскільки кількість можливих значень занадто велика ($2^n!$). У загальному випадку S-блок може також представляти собою і лінійне перетворення, проте на практиці використовуються нелінійні підстановки і з більш низькою розрядністю (24, 28), але в складі більш складних систем. Також у криптоалгоритмах основною задачею нелінійних перетворень є перемішування бітів [24, 3].

P-блок змінює положення символів і є лінійним пристроєм. Цей блок може мати дуже велику кількість входів-виходів, однак через лінійність систему не можна вважати криптостійкою. Аналіз ключового шифру для n-розрядного P-блоку проводиться шляхом подачі на вхід n-1 різних повідомлень, кожне з яких складається з n-1 нуля («0») та 1 одиниці («1») [29]. Головним завданням лінійних перетворень є розсіювання бітів даних під час шифрування [3].

1.2 Методики генерації нелінійних вузлів ускладнення

У сучасних блокових криптосистемах раундові шифри будуються в основному з використанням операцій заміни двійкових кодів невеликої розрядності (схеми, що реалізують цю нелінійну операцію, називаються S-блоками; як правило, саме від їх властивостей насамперед залежить стійкість усієї системи), перестановки елементів двійкових кодів , арифметичних та логічних операцій над двійковими кодами. Кожен раундовий шифр може бути перетворенням, слабким із криптографічної точки зору. Єдине обмеження при побудові складового шифру

полягає у забороні використання у двох сусідніх раундах шифрування перетворень, що мають загальну прозорість.

Вибір і подальше використання S-блоків з покращеними індикаторами шифру дозволяє покращити властивості симетричних перетворень. Це зменшує кількість ітерацій алгоритму, зберігаючи рівень стійкості до атак криптоаналізу. [31]. Перестановка з використанням найкращих криптографічних показників може покращити властивості симетричних перетворень і зменшити кількість ітерацій алгоритму. Однак генерація оптимальних S-блоків є обчислювально складним завданням, яке є неприйнятним при використанні S-блоків як домінуючого фактора, наприклад.

S-блок називається оптимальним [29], якщо він задовольняє наборові відомих на даний момент обмежень на показники, які визначають стійкість симетричних перетворень для диференціального, лінійного та алгебраїчного криптоаналізу. Випадкові S-блоки не є оптимальними, в основному, через їхню низьку стійкість відносно лінійного криптоаналізу [29].

Перші методи побудування S-блоків з покращеними криптографічними показниками базувалися на застосування збалансованих булевих функцій, що дозволило побудувати підстановки з високими показниками стійкості до диференційного та лінійного криптоаналізу [35]. Однак, стійкість таких S-блоків до алгебраїчної атаки залишається недостатньою. На теперішній час значного розвитку набули так звані евристичні методи генерації S-блоків, які передбачають перевірку згенерованих за певним алгоритмом підстановок на відповідність набору критеріїв. При цьому саме на перевірку S-блоків витрачається основна частина обчислювальних ресурсів. Більшість таких методів генерації S-блоків є недостатньо ефективними для отримання підстановок з оптимальними характеристиками на персональному комп’ютері. У деяких випадках це є серйозною проблемою (наприклад, при застосуванні підстановок в якості

довгострокових ключових елементів). Таким чином, актуальною є задача оптимізації існуючих методів генерації S-блоків.

1.3 Показники стійкості криптографічних булевих функцій

S-блоки (блок підстановок, s-box, substitution box) є одним із основних компонентів, що визначають нелінійність та рівень стійкості сучасних симетричних криптографічних алгоритмів. Зокрема, вони особливо важливі для стійкості проти диференціальних атак, лінійних атак, алгебраїчних атак та інших методів криптоаналізу. Можна зробити висновок, що S-блоки та їх властивості мають першорядне значення для безпеки криптографічного алгоритму загалом.

Для захисту криптографічних алгоритмів від різних типів атак S-блоки повинні відповідати низці критеріїв. Через велику кількість існуючих критеріїв, їх суперечливість або часткову взаємозалежність проблематично сформувати S-блок, що володіє всіма відомими заданими властивостями. Тому на практиці використовуються S-блоки, що задовольняють основним критеріям, суттєвим для конкретного симетричного алгоритму. Такі S-блоки прийнято називати оптимальними.

Критерії оптимального S-блоку можуть бути встановлені для цілого класу криптографічних алгоритмів, а також задані для окремо взятого криптопримітиву. При виборі S-блоків під час проектування криптоалгоритмів основними критеріями є нелінійність та диференціальна рівномірність.

Диференціальна рівномірність є показником стійкості проти диференційної атаки. Наприклад, для 8-бітних підстановок оптимальними значеннями диференціальної рівномірності є не більше 8.

Нелінійність є показником стійкості проти лінійної атаки. Оптимальними значеннями для 8-бітових підстановок є значення не менше 100.

Алгебраїчний ступінь та алгебраїчний імунітет є показниками стійкості проти алгебраїчних атак. У разі 8-бітових підстановок оптимальними значеннями

алгебраїчного ступеня є значення не менше 7, а максимальним значенням імунітету алгебри вважається 3 при 441 рівняннях. А в разі підстановок 4 в 4 біта критерій імунітету алгебри не відіграє великої ролі, так як вони можуть бути описані системою рівнянь другого ступеня. Але водночас не може дорівнювати 1.

Ще одним критерієм є відсутність циклів довжини 1, тобто. нерухомих (фіксованих) точок. Існує і багато інших критеріїв. Досі не було доведено необхідності більшості критеріїв. Багато які з них не застосовуються до блокових шифрів, але в той же час застосовуються в потокових шифрах.

Багато досліджень показують, що ідеальних S-блоків, найімовірніше, немає. Тому було введено поняття оптимального S-блоку, критерії якого визначаються для конкретного криптографічного алгоритму або класу криптографічних алгоритмів) та є оптимальними з точки зору захисту від існуючих видів атак.

Булеві функції являються неодмінним криптографічним примітивом, а також відіграють важливу роль при розробці блокових і потокових шифрів. [39]

Булева функція – це така функція, що виконує відображення з поля $GF(2^n)$ усіх двійкових векторів $x = (x_1,...,x_2)$ довжини n у полі $GF(2)$. [40]

Основними способами представлення S-блоку є таблиця істинності, нормальна алгебраїчна форма, перетворення Фур'є, перетворення Уолша, автокореляційна функція. Між цими представленнями існує однозначна відповідність булевої функції, і кожна з них відображає інший аспект властивостей, необхідних для криптографії.

Двійкова таблиця істинності для булевої функції в полі F_2^n – це (0,1)-послідовність $\left(f(\alpha_1), f(\alpha_2),..., f\left(\alpha_{2^n-1}\right)\right)$, а послідовністю функції $f(x)$ є (1, −1)-послідовність, визначена як

$$\left((-1)^{f(\alpha_0)},(-1)^{f(\alpha_1)},...,(-1)^{f\left(\alpha_{2^n-1}\right)}\right),$$

де

$$\alpha_0 = (0,0,\ldots,0), \alpha_1 = (0,0,\ldots,1), \ldots, \alpha_{2^n-1} = (1,1,\ldots,1).$$

n-змінна булева функція *f(x)* є збалансованою, якщо вивід у двійковій таблиці істинності містить однакову кількість 0 і 1 (або 1 і -1). Таблиця істинності S-блоку є конкатенація таблиць істинності всіх координатних функцій.

Алгебраїчна нормальна форма, перетворення Фур'є, перетворення Уолша, автокореляційна функція визначаються за допомогою відповідних уявлень компонентних функцій S-блоку. Алгебраїчна нормальна форма (АНФ) булевої функції *f(x)* є

$$\begin{aligned} f(x_0, x_1, \ldots, x_n) &= \alpha_0 + \alpha_1 x_1 + \cdots + \alpha_n x_n + \\ &+ \alpha_{12} x_1 x_2 + \alpha_{13} x_1 x_3 + \cdots + \alpha_{12\ldots n} x_1 x_2 \cdots x_n \end{aligned}, \quad (1.3)$$

де додавання і множення виконуються в полі F_2.

Зазвичай булеві функції подаються в нормальній алгебраїчній формі. Інша назва алгебраїчної нормальної форми булевої функції – поліном Жегалкіна.

Генерація оптимальних S-блоків є трудомістким завданням. Існуючі методи отримання S-блоків можна розділити на три основні напрямки: алгебраїчні конструкції, псевдовипадкова генерація, евристичний підхід.

У першому підході S-блоки проектуються відповідно до деяких доведених математичних співвідношень та принципів. Найбільш відомими представниками цього підходу є бієктивні $(n \times n)$ S-блоки (перестановки), що ґрунтуються на інверсії в кінцевому полі $GF(2^n)$. Вони є найкращими S-блоками, знайденими та одночасно оптимальними по відношенню до більшості бажаних критеріїв. Наприклад, S-блок AES є таким S-блоком, який має високий алгебраїчну ступінь - 7, високу нелінійність - 112, низьку автокореляцію - 32 і низьку диференціальну

рівномірність – 4. Незважаючи на те, що такі S-блоки часто переважають через їх чудові криптографічні властивості, існують деякі проблеми, пов'язані з їх простою алгебраїчною структурою і можливою майбутньою вразливістю до атак алгебри. Крім того, кількість цих S-блоків невелика, і всі вони афінно еквівалентні.

Другий підхід полягає у побудові S-блоків із таблиці випадкових чисел з подальшою перевіркою його відповідності. Цей підхід приречений на провал із самого початку, так як більшість криптографічних критеріїв, що шукаються, часто суперечать один одному, що значно зменшує кількість S-блоків, які хороші за всіма критеріями, і зменшує ймовірність підбору хорошого S-блоку.

При третьому підході відбувається процес ітеративного поліпшення S-блоку або цілого набору S-блоків по відношенню до однієї або кількох властивостей. На відміну від алгебраїчних конструкцій, евристичні методи здатні створювати великі набори S-блоків, оскільки вони використовують методи прямого пошуку. Найчастіше криптографічні властивості S-блоків, отримані з допомогою евристичних алгоритмів, не такі вдалі, як в алгебраїчно побудованих S-блоків. Однак останніми роками різниця між цими властивостями стає дедалі незрозумілішою. Останнє досягається за допомогою деяких специфічних евристичних методів, таких як метод пошуку сходженням до вершини, метод імітації відпалу, генетичні алгоритми або їх комбінації.

Хоча більшість описаних методів дають хороші результати для побудови бієктивних S-блоків лише по одному з основних критеріїв, це стає набагато складнішим, коли слід розглядати як нелінійність, так і диференціальну рівномірність.

Відомий метод генерації високо нелінійних S-блоків на основі градієнтного спуску [29] вимагає послідовного застосування кількох критеріїв для кожної сформованої підстановки. У роботі [18] представлено удосконалення цього методу шляхом вибору порядку застосування критеріїв, що знижує необхідну обчислювальну потужність для генерації S-блоків. У роботі [8] пропонується

евристичний метод генерації великих множин $(n \times n)$ бієктивних S-блоків, що володіють хорошим поєднанням цільових властивостей, таких як висока нелінійність, висока алгебраїчна ступінь, низька диференціальна однорідність і низька автокореляція, заснований на використанні специфічного мистецтва у поєднанні з модифікацією методу сходження до вершини для S-блоків. У роботі [7] обґрунтовано перспективи подальших досліджень із метою вдосконалення евристичних методів синтезу випадкових S-блоків. На роботі [5] описується узагальнена методологія проектування і тестування S-блоків для симетричних шифрів. Ця методологія включає застосування трьох добре зарекомендувавших себе тестів, які повинні використовуватися для проектування і тестування кожного S-блоку. Дослідження також показує, що принаймні деякі математичні методи проектування S-блоків, будучи безпечними, не безпечніші, ніж математичні методи, але більш обчислювально інтенсивні.

Таким чином, актуальним питанням є аналіз існуючих критеріїв для S-блоків та обґрунтований вибір необхідного набору критеріїв для конкретних криптографічних алгоритмів або класів криптографічних алгоритмів; пошук та розробка теоретично обґрунтованих ефективних практичних методів отримання оптимальних S-блоків, що забезпечують високі показники стійкості у симетричних криптографічних алгоритмах. Проведений аналіз критеріїв та методів дозволить побудувати найефективніший алгоритм генерації оптимальних S-блоків.

1.4 Функція вартості

Основне завдання евристичних технік – зниження (або у деяких випадках збільшення) функції вартості, яка пов’язана з бажаною властивістю S-блоку. Під час роботи алгоритму пошуку виконується наближення характеристик поточного S-блоку до бажаного значення.

Слід зазначити, що для вдалої генерації дуже важливо правильно обрати функції вартості, а отже – треба обережно підходити також до вибору її параметрів.

[42] Наразі існують деякі варіації реалізації функцій вартості, зокрема найвідоміші з них наступніі:

- функція вартості Кларка (Clark's cost functions WHS) [44] та її модифікація [10, 18];
- функція вартості Пічека (Picek's cost functions PCF) [43];
- функція вартості Фрейре – Ечеваррія (Freyre – Echevarría cost functions WCF) [45].

1.4.1 Функція вартості Кларка

Практично в усіх виконаних до цього часу роботах з оптимізації, спрямованих на отримання високо нелінійних функцій, зазвичай було використано саму нелінійність як функцію придатності, тобто придатність функції *f* на *n* вхідних змінних, що задається формулою[44]

$$fitness(f) = N_f = \frac{1}{2}(2^n - \max_{\omega} |\hat{F}(\omega)|) \tag{1.4}$$

або, якщо це розглядається як проблема мінімізації, функція витрат забезпечується за рахунок наступних факторів

$$cost(f) = WH_{\max}(f) = \max_{\omega} |\hat{F}(\omega)| \tag{1.5}$$

Подібним чином, при низькій автокореляції в якості цілі використовується сама автокореляція як функція витрат, тобто функція витрат задана формулою

$$cost(f) = AC(f) = \max_{s \neq 0} |\sum_{x} \hat{f}(x)\hat{f}(x \oplus s)| = \max_{s \neq 0} |\hat{r}(s)| \tag{1.6}$$

Попередні оптимізаційні підходи до розвитку булевих функцій із бажаними криптографічними властивостями були узагальнені для випадку множинного виходу.

Для нелінійності функцією витрат було:

$$cost(f) = \min_{\beta \in B^k} \sum_{\omega \in B^n} \left| \left| \hat{F}_{\beta}(\omega) \right| - X \right|^R \tag{1.7}$$

Для автокореляції функцією вартості було:

$$cost(f) = \max_{\beta \in B^k} \sum_{s \in B^n} \left| \left| \hat{r}_\beta(s) \right| - X \right|^R \tag{1.8}$$

У 2000 році було запропоновано нове сімейство функцій витрат, яке передбачало суттєві вдосконалення для випадку з одним виходом. Після значних експериментів ця функція витрат показала здатність виробляти функції з винятковими профілями критеріїв безпеки. Замість того, щоб ґрунтувати вартість на екстремальних значеннях (згідно з визначенням нелінійності та автокореляції), вона визначила вартість у всьому спектрі Уолша Адамара та спектрі автокореляції. [41]. Було прийнято надавати набір функцій витрат нижче:

$$cost(f) = \sum_{\omega} \left| \left| \hat{F}(\omega) \right| - X \right|^R, \tag{1.9}$$

де X і R є дійсними параметрами.

Оскільки спектральні підходи дали цікаві результати для випадку з одним виходом, тепер можна визначити дві функції витрат для використання в розвитку S-блоку. Функція витрат на основі спектрів Уолша-Адамара задається формулою

$$cost(f) = \sum_{\beta \in B^k} \sum_{\omega \in B^n} \left| \left| \hat{F}_\beta(\omega) \right| - X \right|^R \tag{1.10}$$

і подібна функція витрат на основі спектрів автокореляції задана формулою

$$cost(f) = \sum_{\beta \in B^k} \sum_{\omega \in B^n} \left| \left| \hat{r}_\beta(s) \right| - X \right|^R \tag{1.11}$$

До кожної функції, визначеної як лінійна комбінація виходів, були застосовані функції одиничних витрат та виходи, а результати додавалися за усіма такими комбінаціями.

1.4.2 Функція вартості Пічека

Для булевої функції найбільш звичною формою подання є форма таблиці істинності. Там для отримання збалансованого рішення досить встановити рівно половину нулів і половину одиниць у вихідному векторі. Однак, говорячи про S-

бокс, недостатньо того, щоб кожен стовпець (булева функція) був збалансованим, також потрібно збалансувати всі лінійні комбінації.

Надалі увага зосереджена на поданні Уолша-Адамара та єдиному сценарії булевої функції. У цьому поданні кожен коефіцієнт є числом у *Z*. Однак не існує прямолінійного способу встановлення цих значень. Тільки якщо ми хочемо створити вигнуту (де зігнуті функції існують лише для *n* парних) логічної функції, ми знаємо, що спектр Уолша-Адамара є плоским, тобто $W_F(\vec{v}) = 2^{\frac{n}{2}}$. Однак зігнуті функції - не збалансовані і тому не підходять для більшості застосувань у криптографії [43]. Окрім цієї інформації, існує також теорема Парсеваля:

$$\sum_{\vec{v} \in F_2^n} W_f(\vec{v})^2 = 2^{2n} \tag{1.12}$$

На жаль, ця теорема дає лише необхідну умову для булевої функції, а не достатню. Отже, для розвитку булевої функції, представленої спектром Уолша-Адамара, необхідно було б постійно перевіряти, чи дійсно рішення є булевою функцією, та реалізовувати оператор, який фіксував би ті рішення, які не відповідають дійсності (альтернативно, такі рішення потрібно викинути). Перейти до S-боксу було б ще складніше, оскільки існує низка булевих функцій, які нам потрібно перевірити.

Нарешті, залишилось подати таблицю підстановки, де у випадку, коли *n* = *m*, можна представити S-бокс як перестановку. У такому випадку S-бокс завжди збалансований, і необхідності перевіряти цю властивість немає.

Функція витрат, розроблена Пічеком, використовує лише один параметр, який визначає, скільки значень (*N*) спектра Уолша-Адамара буде використано при оцінці рішення. Автори зробили змогу **зменшити** кількість коефіцієнтів, які мають

максимальне абсолютне значення. Як тільки число максимальних коефіцієнтів абсолютного значення зменшується до нуля, нелінійність S-боксу збільшується.

Значення початкової версії функції витрат $CF_i(S)$ при застосуванні до S-боксу S обчислюється як число коефіцієнтів Уолша-Адамара максимального абсолютного значення (позначене тут як $H(S)_l$):

$$CF_i(S) = H(S)_l \tag{1.13}$$

Для поліпшення функції витрат розглянуто до N коефіцієнтів у спектрі Уолша-Адамара. «Важливість» зменшення кількості коефіцієнтів абсолютної величини пропорційна абсолютним значенням коефіцієнтів. Тобто, найважливішим є зменшення кількості максимальних коефіцієнтів абсолютного значення, а потім - у порядку зменшення кількості всіх коефіцієнтів абсолютного значення.

Нехай $\vec{H}(S)$ - гістограма абсолютних значень коефіцієнтів Уолша-Адамара для S-боксу S. Це вектор, що має в положенні i число коефіцієнтів, рівне $|4i|$ у спектрі Уолша-Адамара S-боксу S. Нехай l - максимальне (останнє) положення в цьому векторі з ненульовим значенням. Тоді максимальне абсолютне значення коефіцієнтів Уолша-Адамара для S-боксу S становить $4l$, і цей коефіцієнт визначає нелінійність S-боксу. Тепер наша функція витрат виглядає так:

$$CF(S) = \sum_{i=0}^{N-1} 2^{-i} H(S)_{l-i}, \tag{1.14}$$

де $CF(S)$ - функція витрат, застосована до S-боксу S, $H(S)_k$ - k-та складова нульового індексованого вектора $\vec{H}(S)$, і ми встановлюємо $H(S)_k = 0, \quad \forall k < 0$.

1.4.3 Функція витрат Фрейре – Ечеваррія

Основним внеском роботи[45] є побудова функції вартості для еволюції продуктивності високонелінійних S-блоків без залежності від будь-якого зовнішнього параметра, на відміну від функцій вартості Кларка та Пічека, які залежать від зовнішніх параметрів.

Нелінійність одного S-блоку залежить від найвищого абсолютного значення спектра Уолша–Адамара.

Нехай C – набір усіх абсолютних коефіцієнтів, менших або рівних межі. Тоді ми маємо наступне:

$C = \left\{ 0, 4, ... 2^{\frac{n+1}{2}} \right\}$, якщо n - непарне;

$C = \left\{ 0, 4, ... 2^{\frac{n}{2}+1} \right\}$, якщо n - парне

Нелінійність одного S-блоку S є максимальною тоді і тільки тоді, коли всі абсолютні значення коефіцієнтів у спектрі Уолша–Адамара S містяться в C.

Треба звести демонстрацію до випадку, коли n - непарне, оскільки необхідно переглянути ті самі умови, коли n парне. Спочатку продемонструємо прямий наслідок, тобто якщо нелінійність S максимальна, то всі абсолютні значення коефіцієнтів у його спектрі Уолша–Адамара містяться в C.

Для непарного n, якщо нелінійність з S є максимальною, тоді $N_S = 2^{n-1} - 2^{\frac{n-1}{2}}$ задовольняє рівність $N_S \le 2^{n-1} - 2^{\frac{n-1}{2}}$.

Крім того, результат $L_S \ge 2^{\frac{n+1}{2}}$ означає, що рівність у $N_S \le 2^{n-1} - 2^{\frac{n-1}{2}}$ призводить до $L_S = 2^{\frac{n+1}{2}}$. Оскільки L_S є найбільшим абсолютним значенням коефіцієнтів у спектрі Уолша–Адамара S, тоді $L_S \ge |X|$ для будь-якого довільного

коефіцієнту X в спектрі. Згідно з визначенням, $max(C) = 2^{\frac{n+1}{2}}$, отже, $max(C) = L_S \geq |X|$. Отже, усі абсолютні значення коефіцієнтів у спектрі Уолша–Адамара S містяться в C.

І навпаки, якщо всі коефіцієнти в спектрі Уолша–Адамара S містяться в C, то ми маємо $L_S \leq 2^{\frac{n+1}{2}}$.

Нехай x – ціле число, а K – скінченна множина натуральних чисел, що $|x| \in K$ Тоді має місце рівність

$$P = \prod_{i \in K} (|x| - i) = 0$$

Нехай M — розмір K. Формулу P можна розкласти на множення M віднімань наступним чином

$$П = (|x| - i_1) \cdot (|x| - i_2) \cdot \ldots \cdot (|x| - i_M)$$

де термін i_t позначає t-й елемент K. Якщо $|x| \in K$, існують такі, що $|x| = i_t$, тобто $|x| - i_t = 0$. Отже, підставляючи в розклад попереднє віднімання, маємо $P = (|x| - i_1) \cdot (|x| - i_2) \cdot \ldots \cdot 0 \cdot \ldots \cdot (|x| - i_M)$, що призводить до $P = 0$.

На основі вище описаних тверджень і ґрунтується функцію витрат. Нехай $S : F_2^n \to F_2^m$ – s-блок. Наступна функція витрат визначається як

$$C_S = \sum_{y \in F_2^n} \sum_{x \in F_2^m} \prod_{z \in C} \left| |W_S(x, y)| - z \right| , \qquad (1.15)$$

де W_S – перетворення Уолша–Адамара S.

Тобто маємо, що будь-який коефіцієнт у спектрі Уолша–Адамара, абсолютне значення якого міститься в C, не заважає кінцевому результату цієї функції витрат. Отже, розрахунок C_S виводиться з коефіцієнтів, абсолютні значення яких

перевищують межу. Крім того, мінімальне значення C_S досягається, коли найбільше абсолютне значення в спектрі Уолша–Адамара дорівнює найбільшому коефіцієнту в C, де C_S дорівнює нулю, що означає, що S має максимальну нелінійність.

Як виявилося, дана функція вартості дозволяє досить швидко згенерувати бієктивні 8-бітні S-блоки випадковим чином, обійшовши результати розрахунків із використанням формул, описаних в п. 1.4.1 та 1.4.2.

2 МЕТАЕВРИСТИЧНІ МЕТОДИ ПОШУКУ

Метаевристичні методи пошуку (також звані «комбінаторною оптимізацією» та «методами пошуку, натхненними природою») були застосовані до дуже широкого кола проблем у кількох різних областях, таких як маршрутизація мережі, планування робочих навантажень машини в алюмінієвому сплаві. ливарні виробництва, створення університетських розкладів і проектування схем. [10] Вони також застосовувалися в різних галузях криптології - переважно в криптографії, оскільки їх використовували для створення компонентів шифру, а також у криптоаналізі. Ми обговоримо це більш детально пізніше, але перш за все пояснимо концепцію метаевристики.

У метаевристичному пошуковому алгоритмі користувач намагається вирішити проблему, створюючи або «розвиваючи» сутність певної форми, будь то бітовий рядок із певними властивостями чи дизайн компонента механізму... Користувач спочатку визначає «функцію вартості» або «функцію відповідності (фітнес-функцію)», яка приймає сутність типу, що розвивається, і виводить скалярне значення. В алгоритмах, які використовують функції вартості, високоякісні рішення задачі повинні відповідати низьким вихідним значенням функції, а погані рішення – високим значенням вартості.

Для алгоритмів, які використовують фітнес-функції, вірно навпаки.

Жорсткого визначення «метаевристичного» методу пошуку немає. Різні методи пошуку під цим прапором мають різні спільні риси, але продовження досліджень у деяких випадках узагальнило визначення.

- Більшість метаевристичних алгоритмів, як зазначено, вимагають від користувача чіткого визначення класу об’єкта, що розвивається, а потім визначення функції вартості C, яка повертає одне скалярне значення $C(x)$

таким чином, що чим менше значення $C(x)$, тим кращим вважається кандидат у рішення x.

(Як зазначено, для деяких алгоритмів функція пристосованості F повинні бути визначені таким чином, щоб високі значення $F(x)$ відповідають якісним рішенням.)

У випадку, якщо є різні критерії, які нам потрібно оптимізувати, деякі з яких можуть суперечити іншим, багатоцільові еволюційні алгоритми [4] є узагальненою формою метаевристики, в якій для кожного критерію визначаються окремі функції вартості або придатності, і одна з різних стратегій використовується для їх одночасної оптимізації за допомогою більш складного методу, ніж одна звичайна функція витрат, що обчислює загальну вартість з виходи окремих функцій.

- Алгоритм не є детермінованим. Він часто використовує генератор псевдовипадкових чисел (PRNG) під час прийняття рішень або побудови варіантів рішень, і два запуски того самого алгоритму з різними початковими значеннями PRNG повинні поводитися по-різному.
- Алгоритм є ітераційним алгоритмом. У кожній ітерації він створює або один кандидат-рішення, або «популяцію» з кількох кандидатів.
- У всіх ітераціях, окрім першої, алгоритм використовує інформацію (можливо, включаючи вартість) з варіантів рішень, створених у деяких або всіх попередніх ітераціях, щоб побудувати рішення на заміну або його набір. При цьому середня вартість на ітерації i не гарантовано буде нижчим за середню вартість на ітерації $i-1$, з часом вартість повинна зменшуватися.

  Існують різні підходи до цього. Меметичні алгоритми використовують наявну популяцію для створення нових рішень, утворюючи нову популяцію, а потім вносять кілька поступових змін до кожного члена

популяції. Насправді більшість метаевристик працюють з сутностями так, що в сутність можна внести невеликі зміни з обмеженим впливом на вартість. Імітований відпал використовує ці зміни для постійної зміни єдиного кандидатського рішення. Мурашині алгоритми зберігають інформацію з попередніх рішень, яку вони використовують для побудови нових рішень, але не зберігають ні попередні рішення, ні достатньо інформації для їх реконструкції. Негенераційні еволюційні алгоритми подібні до меметичних алгоритмів, але лише замінюють кількох членів популяції новими кандидатами на кожному етапі.

- Має бути можливість визначити «простір пошуку». Це набір кандидатів у рішення, що повністю складається з сутностей форми, що розвивається. Будь-яка функція, що використовується під час пошуку, яка генерує новий кандидат на рішення, чи то з нуля, чи то з використанням інформації з попередніх ітерацій, повинна мати цей набір або деяку його підмножину як кодомен. Будь-яка функція, яка змінює кандидатське рішення під час пошуку, також повинна мати цей набір або деяку його підмножину (можливо, частково визначену кандидатом попередньої модифікації) як свій кодомен - таким чином простір пошуку закрито для всіх визначених функцій, які змінюють кандидатські рішення. У просторі пошуку не повинно бути жодного елемента, який би алгоритм пошуку не міг на якомусь етапі вважати його кандидатом.
- У деяких випадках може здаватися (неправильно!), ніби це порушення, оскільки не кожен елемент у просторі пошуку насправді відповідає правильному рішенню проблеми, яку бажає вирішити користувач. Можливо, простір пошуку складається з набору елементів, які можуть бути перетворені в дійсні рішення, а функція вартості базується на успіху перетворення в досягненні цього [9]. У деяких випадках поняття *оподіл генотип-фенотип* призводить до того, що простір пошуку повністю

складається з сутностей одного типу, а функція вартості штрафує ті, які не можуть бути перетворені в сутності другого типу, а також винагороджує відповідні якості сутностей другого типу, коли трансформація вдається.

- Вище ми посилалися на «Будь-яку функцію, яка змінює кандидатське рішення під час пошуку». Такі функції відомі як «функції переміщення», а одне застосування функції переміщення є «переміщенням». Вони впливають на визначення простору пошуку, оскільки:
    - Будь-яка сутність може розглядатися як така, що має різні $x \geq 1$, «x-переміщення околиці», підмножина простору пошуку, що складається з кандидатів на рішення, які можна отримати від вихідної сутності шляхом x рухається.
    - Де рішення S_a є членом x-переміщення околиці рішення S_b, S_b також має бути членом x-переміщення околиці S_a.

  Визначення ходу залежить від сутності, яка розвивається. Наприклад, якби ми намагалися розвинути бітовий рядок, ми могли б вибрати один із його бітів і перевернути його. Або ми могли б поміняти місцями нуль і одиницю в бітовому рядку. Один хід, як правило, має лише незначний вплив на значення функції витрат.
- Продуктивність алгоритму спочатку трохи краща, ніж алгоритм, який вибирає елементи з набору варіантів рішень випадковим чином. Хоча ці ранні рішення можуть, залежно від алгоритму, підлягати певній формі (можливо, детермінованої) оптимізації, ранні етапи алгоритму призначені для дослідження широкої області простору пошуку. З часом поведінка алгоритму відходить від дослідження та стає все більше зосередженою на підвищенні якості рішень у поточному регіоні простору пошуку.

Ми наводимо два важливих визначення

Локальний оптимум є кандидатом на рішення таким чином, що його вартість менша, ніж вартість будь-якого іншого варіанта рішення в його околиці з 1 ходом. Запобіжні заходи обов'язково мають бути прийняті для того, щоб запобігти тому, щоб алгоритм пошуку занадто рано «застряг» у локальному оптимумі, коли можуть бути кращі локальні оптимуми або навіть «глобальний» оптимум, як визначено нижче, для пошуку.

Глобальний оптимум є варіантом рішення таким чином, що його вартість,в, менше або дорівнює вартості всіх інших учасників простору пошуку. Таких може бути більше одного, залежно від функції вартості та простору пошуку.

2.1 Підйом на гору

Підйом на пагорб — це метод математичної оптимізації, який належить до сімейства локального пошуку. Це ітераційний алгоритм, який починається з довільного вирішення проблеми та поступово змінює рішення, щоб спробувати знайти краще рішення. Якщо зміна призводить до кращого рішення, у нове рішення вносяться подальші поступові зміни, які повторюються, доки не буде видно жодних покращень.

Чи належить метод підйому на гору до метаевристичних – досі спірне питання. Він використовується як компонент у різноманітних метаевристичних алгоритмах пошуку та життєво важливий для меметичних алгоритмів зокрема, але також може використовуватися як самостійний алгоритм пошуку.

Існує багато різних алгоритмів підйому на пагорб, але загалом вони реалізують алгоритм, який перевіряє частину або всю околицю з одним ходом поточного кандидата на рішення S, або заміна S відразу, коли знайдено кандидата з нижчою вартістю, або завершивши тести, а потім замінивши S із знайденим найкращим покращенням. Це повторюється або для певної фіксованої кількості ітерацій, або до тих пір, поки в одній з ітерацій не буде знайдено жодного покращувального кандидата.

Сходження на гору - це метаевристика, що базується на локальному пошуці, в якій рішення-кандидати, які вона перевіряє на кожній ітерації, знаходяться в межах 1-ходового околиці поточного кандидата.

Псевдокод нижче показує два алгоритми підйому на пагорб, які ми включили в наші алгоритми пошуку в цій статті.

Так званий «дробовик» підйому на пагорб, який виконує кілька алгоритмів підйому з кількох різних, випадково вибраних початкових точок S_0, і прийняти найкращий результат, часто використовувався в криптоаналізі старих шифрів на олівці та папері [10] та машинних шифрів - зокрема, дозволяючи криптоаналіз німецької Enigma, коли зашифрований текст відомий, але немає відповідного відкритого тексту [6].

```
S0 позначає початкового кандидата
S ← S0
repeat
    Sbest ← S
    ACCEPTS_IN_THIS_LOOP ←false
    for x ←0, sizeof(1-переміщення охрестя S) do
        Sx позначає x-го члена 1-ходового переміщення охрестя
    S.
        cost_dif ←C(Sx) - C(Sbest)
        if cost_dif <0 then
            ACCEPTS_IN_THIS_LOOP ←true
            Sbest ← Sx
        end if
    end for
    if ACCEPTS_IN_THIS_LOOP =true then
        S ← Sbest
    end if
until ACCEPTS_IN_THIS_LOOP=false
return S
```

Рисунок 2.1 - Псевдокод для детермінованого алгоритму сходження на пагорб (Алгоритм 1)

```
Цей алгоритм використовувався лише в ситуаціях, коли інший був
неможливим. S_0 позначає початкового кандидата
S ← S_0
repeat
    S_best ← S
    ACCEPTS_IN_THIS_LOOP ←false
    for x ←0, CANDIDATES_PER_LOOP do
        S_x позначає випадково обраного члена 1-ходового
    переміщення охрестя S.
        cost_dif f←C(S_x) - C(S)
        if cost_dif f <0 then
            ACCEPTS_IN_THIS_LOOP ←true
            S_best ← S_x
        end if
    end for
    if ACCEPTS_IN_THIS_LOOP =true then
        S ← S_best
    end if
until ACCEPTS_IN_THIS_LOOP=false
return S
```

Рисунок 2.2 - Псевдокод для недетермінованого алгоритму сходження на пагорб (Алгоритм 2)

Ядром усіх евристичних методів є метод сходження на гору (HC), представлений у [19]. Метод HC дозволяє збільшити нелінійність булевої функції, зокрема випадково згенерованої. Метод HC можна ефективно використовувати з методами генетичного та імітованого відпалу.

2.2 Імітований відпал

Імітований відпал (SA) — добре відомий метод метаевристичного пошуку, який використовувався для вирішення багатьох задач комбінаторної оптимізації. Це алгоритм підйому на гору з доданою можливістю виходу за межі локального

оптимуму в просторі пошуку. Це хороше рішення, але вимагає багато часу порівняно з простою процедурою підйому в гору.

Імітований відпал — це ще один алгоритм, заснований на локальному пошуку, багато в чому схожий на більш складну форму підйому на пагорб. Його надихає метод, який використовується в металургії для усунення дефектів у кристалічних структурах у зразках металу.

При моделюванні відпалу деяке вихідне рішення-кандидат, S_0, зазвичай вибирається випадковим чином, вводиться в алгоритм SA разом із такими параметрами:

- Функція витрат C.
- Початкове значення T_0 для «температури». Чим вища температура в поточній ітерації, тим більша ймовірність, що алгоритм пошуку прийме хід, який призведе до рішення-кандидата з вищою вартістю, ніж поточний кандидат (тобто зберегти вказане рішення-кандидат як «поточний кандидат»). З часом температура падає, через що алгоритм приймає менше рухів, що не пов’язані з покращенням, і, отже, відходить від дослідження до оптимізації. Ближче до кінця пошуку алгоритм надзвичайно рідко приймає рух без покращення, і його поведінка дуже схожа на поведінку алгоритму підйому на пагорб.
- При виборі значення T_0, різні джерела стверджують, що його слід вибирати так, щоб певна частка рухів приймалася при температурі T_0. Існує дуже мало інформації чи порад щодо того, якою має бути ця пропорція. В одній із найперших робіт про моделювання відпалу [10] зазначено, що підійде будь-яка температура, яка веде до початкового рівня прийнятності 80% або більше; проте наші початкові експерименти показали, що це було занадто високо для більшості експериментів у цій

дипломній роботі. Зазвичай ми зупинялися на початковому рівні прийняття 0,5 або 0,6 замість 0,8.

Вибравши початкову швидкість прийняття, експериментатор виконує алгоритм відпалу з різними T_0 поки не буде знайдено температуру, яка досягає частки, достатньо близької до цієї. Ми почали з температури 0,1 і кілька разів запускали алгоритм, подвоювали температуру та повторно запускали алгоритм, доки не було отримано принаймні такий високий рівень прийнятності, як вказаний. Де T_a була температура, при якій це було досягнуто, і $T_b = T_a / 2$, ми потім використали двійковий пошуковий алгоритм, щоб отримати температуру між T_a і T_b це призведе до рівня прийняття $\approx 50\%$.

- Значення α; «коефіцієнт охолодження», що визначає, наскільки знижується температура на кожній ітерації алгоритму.
- Ціле значення: МАКС_ВНУТРІШНІ_ПЕТЛІ (MAX_INNER_LOOP), що визначає кількість ходів, які може зробити алгоритм локального пошуку при кожній температурі.
- Необхідно також вказати критерій зупинки. Ми використовували МАКС_ЗОВНІШНІ_ПЕТЛІ (MAX_OUTER_LOOP) значення, яке вказує, скільки разів алгоритм мав знизити температуру та продовжити пошук, перш ніж він зупиниться.
- Ми також вказали МАКС_ЗАМОРОЖЕННЯ_ЗОВНІШНІХ_ПЕТЕЛЬ $(MAX\_FROZEN\_OUTER\_LOOP)$ параметр. Якби алгоритм на будь-якій стадії виконав стільки зовнішніх циклів, не прийнявши жодного руху, він би вважався вкрай малоймовірним, щоб зробити щось інше, окрім того, щоб залишатися повністю нерухомим з цього моменту, і отримав би вказівку завершити роботу раніше.

Термін «імітований відпал» базується на фізичному процесі відпалу в твердому стані, під час якого відбувається кристалізація речовини та намагається мінімізувати енергію системи шляхом повільного охолодження атомів, доки вони не досягнуть стабільного стану. Технологія повільного охолодження дозволяє атомам металу вирівнятися та сформувати правильну кристалічну структуру з високою щільністю та низькою енергією. Переходи атомів з однієї клітини в іншу відбуваються з постійною ймовірністю, і ймовірність зменшується зі зниженням температури. Початкова температура і швидкість зниження температури називаються графіком відпалу.

Використовуючи SA для вирішення задачі комбінаторної оптимізації (CO), ми починаємо з конкретного можливого вирішення проблеми. Потім ми намагаємося оптимізувати це рішення за допомогою методу, подібного до твердофазного відпалу. Околиці для цього перехрестя створюються за допомогою відповідного методу, і розраховується вартість (або доцільність) нового перехрестя. Якщо нове рішення є кращим за поточне рішення з точки зору економії коштів (або підвищення придатності), нове рішення приймається. Якщо нове рішення не краще поточного рішення, нове рішення приймається з певною ймовірністю. Імовірність прийняття зазвичай встановлюється на $exp(-\Delta / T)$, де Δ - це зміна вартості між старим та новим розв'язком, а T - поточна температура. Таким чином, ймовірність експоненційно зменшується в міру погіршення руху.

Процедура SA має менше шансів застрягти в місцевому оптимумі порівняно з простим hill-climbing, оскільки погані шляхи все ще мають шанс бути здійсненими. Спочатку температуру відпалу вибирають високою, так що ймовірність прийняття також буде високою, і приймаються майже всі нові розв'язки. Потім температуру поступово знижують, так що ймовірність прийняття низькоякісних розв'язків буде дуже малою, і алгоритм працює більш-менш, як hill-climbing, тобто високі температури дозволяють краще досліджувати пошуковий

простір, тоді як нижчі температури дозволяють регулювання кращого розв'язку. Цей процес повторюється до тих пір, поки температура не наблизиться до нуля або поки неможливо досягти подальшого поліпшення. Це аналогічно атомам твердого тіла, що переходять у кристалізований стан.

2.3 Меметичні алгоритми

Меметичні алгоритми поєднують локальну оптимізацію з існуючою метаевристикою «генетичних алгоритмів» і виявилися надзвичайно ефективними методами пошуку.

Існують певні відмінності в їх роботі – зокрема, не кожна реалізація для кожної предметної області проходитиме через чотири основні етапи в тому самому порядку, як у нашій реалізації, а деякі використовуватимуть більш складні методи машинного навчання на стадії локальної оптимізації, у той час як ми робимо простий підйом на пагорб. Зазначивши це, ми продовжуємо наш опис.

Меметичний алгоритм підтримує «популяцію» потенційних рішень у формі мультинабору з розміром, який визначається параметром *popsize*. Протягом кількох ітерацій; або «поколінь» - аналогічно зовнішнім петлям імітованого відпалу - нові популяції P_i будуть походити від своїх безпосередніх попередників. Члени популяції P_0 на початку алгоритму генеруються випадковим чином і підіймаються до локальних оптимумів. У нашій реалізації «проміжний» мультинабір містить результати застосування різних етапів алгоритму до P_i - для цілей цього розділу ми позначаємо цю множину *PCP* (Посткросоверна популяція). *PCP* очищається на початку кожного покоління та повторно заповнюється операцією «перехресного» у зазначеному поколінні. Члени *PCP* потім випадково змінюються під час фази «мутації» ітерації, підіймаються на пагорб і під час фази «відбору» використовуються для генерації P_{i+1}.

```
S←S0
bestsol← S0
T←T0
ZERO_ACCEPT_LOOPS ← 0
for x ← 0, MAX_OUTER_LOOPS - 1 do
     ACCEPTS_IN_THIS_LOOP ← false
     for y ← 0, MAX_INNER_LOOP S - 1 do
          Виберіть кілька S_n після 1го руху сусідства з S.
          Cost_dif f ← C(S_n)-C(S)
          if cost dif f < 0 then
               S ← S_n
               ACCEPTS_IN_THIS_LOOP ← true
               if C(S_n) < C(bestsol) then
               bestsol ← S_n
          end if
     else
          u ← Rnd(0,1)
          if u < exp(-cost_dif f/T) then
               S ← S_n
               ACCEPTS_IN_THIS_LOOP ← true
          end if
     end if
end for
if ACCEPTS_IN_THIS_LOOP = false then
     ZERO_ACCEPT_LOOPS ← ZERO_ACCEPT_LOOPS + 1
     if ZERO_ACCEPT_LOOPS = MAX_FROZEN_OUTER_LOOPS then
                                        Алгоритм завершується раніше
          return bestsol
     end if
end if
T ← T × α
end for
return bestsol
```

Рисунок 2.3 - Псевдокод для імітованого алгоритму відпалу (Алгоритм 3)

Кількість поколінь є одним з параметрів – *NO_OF_GENERATIONS*, аналогічний до *MAX_OUTER_LOOPS* параметр імітованого відпалу.

Функція кросовера, *cross*(p_1, p_2), приймає два «батьківських» рішення-кандидати p_1 і p_2 як вхідні дані та виводить «дочірнє» рішення-кандидат o_1яка певним чином поєднує риси обох p_1 і p_2. Зауважте, що o_1= *cross*(p_1, p_2), не обов'язково дорівнює o_2= *cross*(p_2, p_1).

Кілька різних алгоритмів кросинговеру були розроблені для еволюції біективних функцій (або, справді, будь-якої сутності, яку можна представити як перестановку на наборі цілих чисел), і вважається надзвичайно важливим вибрати

хороший алгоритм кросинговеру для проблемної області. У розділі цієї дипломної роботи ми намагаємося розвинути біективні функції над кінцевим полем $GF\left(2^n\right)$, ми порівнюємо два різні методи кросинговеру; *PMX* («частково відображений кросовер») і циклічний кросовер [10]. Ці два методи кросинговеру були обрані через їхню увагу до параметру *x* у зіставленні з кожним виходом замість того порядку, в якому ці виходи з'являлися.

• *Циклічний кросовер* працює наступним чином

PARENT 1: a b c **d** e f g h i j

PARENT 2: c f a **j** h d i g b e

(Випадково вибрана початкова точка циклу виділена жирним шрифтом.)

Елемент *Parent 1* у початковій точці циклу копіюється в дочірній елемент у тій же позиції:

CHILD: ? ? ? d ? ? ? ? ? ?

Елемент у тій же позиції в *Parent 2* є наступним для копіювання в дочірній елемент.

Однак він копіюється в ту саму позицію, в якій він зустрічається в батьківському 1:

CHILD: ? ? ? d ? ? ? ? ? j

Цей процес триває, доки процес не поверне нас до вихідної початкової точки циклу - іншими словами, коли створено «петля» або «цикл». В цьому випадку:

(d, j)→(j, e)→(e, h)→(h, g)→(g, i)→(i, b)→(b, f)→(f, d)→(d, j) знову.

CHILD: ? b ? d e f g h i j

Будь-які інші вакантні позиції в дочірньому елементі потім заповнюються шляхом копіювання відповідних значень із батьківського елемента 2:

CHILD: c b a d e f g h i j

•Кросовер *PMX* починається випадковим вибором двох «точок перетину», як показано вертикальними лініями нижче. Елементи *Parent 1* між цими точками копіюються в дочірній елемент:

PARENT 1: a b | c d e f | g h i j

PARENT 2: c f | a j h d | i g b e

CHILD: ? ? | c d e f | ? ? ? ?

Далі будь-які елементи батьківського елемента 2, які ще не були скопійовані в дочірній елемент, копіюються в:

PARENT 1: a b | c d e f | g h i j

PARENT 2: c f | a j h d | i g b e

CHILD: ? ? | c d e f | i g b ?

C уже скопійовано з батьківського елемента 1, тому елемент у позиції [0] не може дорівнювати *C*. Ми бачимо, що елемент батьківського елемента 2 у тій же позиції, що й C у батьківському елементі 1, - *A*, і скопіюємо його в позицію [0]. Подібним чином кінцевий елемент не може дорівнювати *E*, тому ми розміщуємо *H* у цій позиції, батьківський елемент 2 у ту саму позицію, що й *E* батьківського елемента 1.

PARENT 1: a b | c d e f | g h i j

PARENT 2: c f | a j h d | i g b e

CHILD: a ? | c d e f | i g b h

Остаточна нерозподілена позиція складніша. Ми не можемо скопіювати *F*, оскільки він вже присутній у дитини. Ми шукаємо *F* у Батьківському 1 і знаходимо *D* у відповідній позиції Батьківського 2. На жаль, *D* також було скопійовано в дочірній елемент! Далі ми шукаємо *D* у Батьківському 1 і знаходимо, що *J* присутній у тій самій позиції Батьківського 2 і не був скопійований у дочірній, що дозволяє нам завершити процес:

PARENT 1: a b | c d e f | g h i j

PARENT 2: c f | a j h d | i g b e

CHILD: ? ? | c d e f | i g b ?

Який би метод кросинговеру ми не вибрали, задіяні наступні два параметри:

• *no_of_children*: Коли p_1 і p_2 вибираються з P_i, це визначає, чи буде застосована функція перехресного переходу лише для додавання $o_1 = cross(p_1, p_2)$, до *PCP*, або чи $o_2 = cross(p_2, p_1)$, також буде обчислено та додано.

Якщо функція кросинговеру не застосована, це визначає, що один p_1 або обидва p_1 і p_2 додаються до *PCP*.

• *crossover_probability*: Коли p_1 і p_2 вибираються з P_i під час фази кросинговеру це визначає ймовірність застосування функції кросинговеру - тобто чи o_1(і o_2, залежно від попереднього параметра), або p_1 і можливо p_2, додаються до *PCP* в цьому поколінні.

Нам також потрібна «функція мутації», *mutate(c)*, взявши варіант рішення з *PCP* як вхідні дані, роблячи невелику випадкову зміну («мутацію») певної форми та повертаючи результат (який замінює оригінал у *PCP*). У наших експериментах, функція мутації робить один хід, як визначено тією ж методологією локального пошуку, що використовується для симуляції відпалу та підйому на пагорб; у випадку еволюції біективних функцій це означає, що два елементи таблиці істинності міняються місцями. Для цього співвідносяться два параметри:

• максимальна кількість можливих мутацій(*max_possible_mutations*): під час «фази мутації» алгоритму це визначає максимальну кількість мутацій, які можуть бути застосовані до будь-якого окремого кандидата.

• ймовірність мутації (*mutation_probability*): кожна потенційна мутація (до кількості, визначеної критерієм вище) виникає випадково з цією ймовірністю, незалежно від інших потенційних мутацій.

Мутація додає аспект дослідження в меметичний пошук, дозволяючи йому уникнути локальних оптимумів.

Третій етап, підйом на пагорб (*hill-climbing*), майже ідентичний алгоритму підйому на пагорб, визначеному раніше. Зауважте, однак, що оскільки меметичні алгоритми використовують функцію придатності замість функції вартості, алгоритм має бути налаштованим відповідно до цього. На цьому етапі учасники *PCP* всі піднімаються на пагорб до локальних оптимумів щодо зазначеної функції придатності.

Нарешті, ми маємо фазу «відбору», яка сама по собі поділяється на різні підфази. Виконавець може вирішити відсортувати елементи *PCP* за їх значеннями придатності для ефективності на початку фази відбору, якщо так, - це сортування є першою підфазою.

Після того, як сортування виконано (чи ні), наступною підфазою є підфаза «елітарності». Якщо параметр *рівень елітарності*(*elitism_level*) має ненульове значення, тоді *elitism_level* учасник з P_i з найвищими значеннями придатності копіюються безпосередньо в P_{i+1}. Якщо це призводить до повної популяції (що не бажано!), етап відбору завершується. Якщо ні, нам потрібно використовувати метод вибору, щоб продовжувати вибирати елементи з *PCP* щоб додати до P_{i+1}.

Дозволяє |*PCP*| бути позначеним *M*. У цій дипломній роботі ми експериментуємо з двома методами відбору:

1) Вибір колеса рулетки(*Roulette-whell selection*): Для виведення цього методу необхідна фітнес-функція значення ≥ 0 для всіх можливих входів. Нехай $\sum_{i=0}^{M} fitness(c_i)$ позначено як *Z*, а кількість місць, що залишилися в популяції, - *r*. Потім ми виконуємо процедуру в псевдокоді для алгоритму 4:

```
for i←1, r do
Один учасник PCP вибирається випадково з ймовірністю fitness(c_i) / Z
                                    Усі вибірки r незалежні і випадкові.
                                Кандидат може бути обраний кілька разів.
Копію цього учасника додано до P_{i+1}.
Початковий учасник поміщається назад в PCP.
end for
```

Рисунок 2.4 - Псевдокод для вибору колеса рулетки (Алгоритм 4)

2) Вибір рангу(*Rank selection*): Для цього методу відбору кандидати в PCP необхідно відсортувати за придатністю. Індексація відбувається від 1 вгору, тобто *PCP[1]* - це кандидат з найнижчою придатністю; *PCP[M]* - кандидат з найвищою.

Як і раніше, в кожному з *r* незалежних випробувань вибирається кандидат *PCP*. Копія цього кандидата розміщена в P_{i+1}, і кандидат замінюється на *PCP*. Різниця між цим і вибором колеса рулетки полягає в ймовірності, з якою буде обрано кожного кандидата:

$$P\left(PCP\left[i\right]\right)=\frac{2i}{M\left(M+1\right)}$$

2.4 Мурашині алгоритми

Першим методом оптимізації мурашиної колонії була Ant System, спочатку описана [13] як метаевристика, яку можна застосувати до проблеми комівояжера (TSP). Пізніші удосконалення створили більш ефективну систему мурашиних колоній [13] [10], яка використовувала більш елітарний підхід і досягла кращих результатів проти TSP.

Будь-яку задачу, до якої можна застосувати мурашині алгоритми, необхідно представити у вигляді графіка. Для експериментів із S-box вузлами графіка є

значеннях, а графік спрямований - ребро від вузла x до вузла y означає, що $S(x)=y$. Крім того, кожне ребро несе з собою вартість - і, на відміну від звичайного TSP, край, що йде від y до x може мати не таку ж вартість, як у x до y (роблячи дану проблему більш схожою на Asymmetric TSP).

Проблема також повинна дозволити розробити корисну функцію витрат, щоб під час побудови кожного потенційного рішення вартість починалася з нуля і збільшувалася щоразу, коли додається новий компонент, доки не буде отримано остаточну вартість. Компонент має бути представлений як ребро на графіку.

```
                                                    Етап 1: Кросовер.
Скидаємо PCP до порожнього мультинабору.
while size(PCP) < POST_CROSSOVER_POPULATION_SIZE do
     Вибираємо p1 і p2 з Pi рівномірно довільно
     if Rnd(0,1) < crossover_probability then
          o1 ← cross(p1, p2)
          Додаємо o1 до PCP
          if no_of_children = 2 then
               o2 ← cross(p2, p1)
               Додаємо o2 до PCP
          end if
     else
          Додаємо p1 до PCP
          if no_of_children = 2 then
               Додаємо p2 до PCP
          end if
     end if
end while
```

Рисунок 2.5 - Псевдокод для меметичного алгоритму (Алгоритм 5, Етап 1: Кросовер)

```
                                                    Етап 2: Мутація
for i ← 0, POST_CROSSOVER_POPULATION_SIZE − 1 do
     for j ← 0, max_possible_mutations − 1 do
          if Rnd(0,1) < mutation_probability then
               Виконуємо один рух (як визначено для локального
          пошуку) до PCP[i]
          end if
     end for
end for
```

Рисунок 2.6 - Псевдокод для меметичного алгоритму (Алгоритм 5, Етап 2: Мутація)

```
                                                    Етап 3: Підйом на гору
for i ← 0, POST_CROSSOVER_POPULATION_SIZE − 1 do
Підйом на гору PCP[i] до локального оптимуму.
end for
```

Рисунок 2.7 - Псевдокод для меметичного алгоритму (Алгоритм 5, Етап 3: Підйом на гору)

```
                                                    Етап 4: Відбір.
Скидаємо сукупність до порожнього мультинабору.
if elitism_level вказано then
     копіюємо elitism_level членів P_i в P_{i+1}
end if
while size(population) < popsize do
     використовуємо функцію вибору, щоб вибрати наступного члена
PCP, щоб додати до P_{i+1}.
end while
```

Рисунок 2.8 - Псевдокод для меметичного алгоритму (Алгоритм 5, Етап 4: Відбір)

Значення d_{ij} позначає суму, на яку збільшується вартість, якщо край від вузла i до вузла j додається, тобто якщо $S(i)$ присвоюється значення j. Хоча для деяких проблем (наприклад, TSP) d_{ij} є константою, тут на нього впливають значення таблиці істинності, які вже були призначені, тому їх потрібно перераховувати кожного разу, коли нам потрібно додати значення для $S(i)$.

Задіяні такі параметри:

- Особливий тип мурашиного алгоритму. У наших експериментах ми порівнювали Ant System, оригінальну Ant Colony System Доріго та версію ACS, визначену в «Основах метаевристики» Шона Люка [10]. Існують інші алгоритми; наприклад «AntNet» [13], спеціалізований варіант, призначений для вирішення проблем мережевої маршрутизації.
- гірські стежки (hillclimb_trails). Це логічне значення, яке визначає, чи використовується локальна оптимізація (тобто підйом на пагорб побудованих рішень) під час побудови маршруту.
- метод наступного індексу(next_index_method). Після додавання ребра від i до j, цей параметр визначає, з якого вузла мураха має спробувати додати ребро, починаючи з наступного. У цій дипломній роботі ми експериментуємо з «циклом», у якому наступний вузол є вузол j, і «інкремент», у якому наступний вузол є вузлом $(i+1)$.
- α і β є значеннями з плаваючою комою. Значення α визначає ступінь впливу рівнів феромонів на відбір краю та значення β визначає вплив d_{ij}.
- e - параметр оновлення елітарного феромону (використовується лише у версіях ACS етапу глобального оновлення).
- τ_0 - початкова кількість феромону на кожному ребрі.
- no_of_ants - кількість мурашок.
- Q - скалярне значення, на яке множиться кількість феромону, депонованого в глобальному оновленні. У статті, в якій була спочатку описана система Ant [10], після експериментів з Q=1, 100 і 10000, 100 було прийнято як «експериментально визначене оптимальне значення». Однак у більш пізніх описах ACS [13], Q =1 було використано неявно, а інші значення не згадувалися.

- q_0 - для алгоритмів ACS це визначає ймовірність того, що даний вибір краю використовуватиме елітарний метод відбору замість дослідницького методу Ant System. Для Ant System, q_0 фактично завжди дорівнює нулю.
- ρ - неелітарний параметр оновлення феромонів (також відомий як «швидкість випаровування»).

3 ЕВРИСТИЧНІ МЕТОДИ СХОДЖЕННЯ НА ГОРУ (англ. Hill Climbing)

3.1 Генерація булевих функцій методом градієнтного спуску

Метод побудови булевих функцій на основі градієнтного спуску був представлений у [30, 23]. Нагадаємо, що булеві функції є підкласом векторних булевих функцій з $m=1$.

Основна ідея методу полягає у зниженні нелінійності заданих бент-послідовностей. Іншими словами, у заданій бент-послідовності (таблиці істинності) змінюються деякі біти таким чином, щоб нова послідовність була збалансованою, а нелінійність була близька до нелінійності бент-функції.

Очевидно, що довільна зміна бітів може негативно позначитися на нелінійності послідовності, що вийшла. Тому для інвертування вибираються позиції, зміна яких тягне за собою збільшення або зменшення максимального значення перетворення Волша на рівну кількість, так щоб нелінійність функції, що вийшла, була близька до попередньої.

Цей метод, наприклад, може бути використаний для генерації криптографічних функцій, що застосовуються в поточних шифрах. Однак, для сучасних ПШ функція, що фільтрує, повинна бути як мінімум 20 біт або більше [29]. У зв'язку з чим ефективність даного методу знижується, оскільки, крім знаходження самої послідовності, ще необхідно відновлювати алгебраїчну нормальну форму [2], яка повинна містити мінімальну кількість мономів для ефективної реалізації алгоритму шифрування.

Інша область застосування полягає в наборі булевих функцій у нелінійний вузол заміни, який заздалегідь має високі криптографічні властивості.

3.1 Евристичний метод градієнтного підйому

Суть методу градієнтного підйому полягає в оптимізації рівня нелінійності булевої функції шляхом додавання позицій до таблиці істинності вихідної функції. Унікальні вхідні дані призначені для кожної позиції у таблиці істинності. Використовуючи цей метод, можливо створити повний список вхідних даних для функції, де додавання вихідної позиції, що відповідає цьому входу, у таблиці істинності збільшує нелінійність цієї функції. Список таких позицій в таблиці істинності позначений цифрою 1 - Покращений набір (*Improvement Set*) функції *f(x)*, або $1-IS_f$ [15]:

$g(x)=f(x)\oplus 1$для $x=x_a$ і $g(x)=f(x)$для всіх інших *x*. Якщо $N_g > N_f$, тоді $x_a \in 1-IS_f$.

У [15] швидкий систематичний метод визначення множини $1-IS_f$ певної булевої функції представлено за допомогою її таблиці істинності та перетворень Уолша-Адамара. Щоб знайти набір $1-IS_f$ заданої булевої функції спочатку потрібно визначити значення коефіцієнта перетворення Уолша-Адамара, яке відповідає значенню, близькому до абсолютного значення найбільшого коефіцієнта, $WH_{\max}$.

$f(x)$ є булевою функцією з перетворенням Уолша-Адамара $F(w)$, де $WH_{\max}$ позначено максимальне абсолютне значення $F(w)$. Одна або кілька лінійних функцій $L_w(x)$ мають найменшу відстань до функції f(x), а для даних w виконується рівність $\left|F(w)\right| = WH_{\max}$ буде існувати.

Наступний набір визначається як:

$$W_1^+ = \left\{w : F(w) = WH_{\max}\right\} i$$
$$W_1^- = \left\{w : F(w) = -WH_{\max}\right\}$$

Також встановлює *w*, для яких значення *WHT* близькі до максимуму визначаються:

$$W_2^+ = \{w : F(w) = WH_{\max} - 2\};$$
$$W_2^- = \{w : F(w) = -(WH_{\max} - 2)\};$$
$$W_3^+ = \{w : F(w) = WH_{\max} - 4\};$$
$$W_3^- = \{w : F(w) = -(WH_{\max} - 4)\};$$

Одна зміна в таблиці істинності призводить до зміни +2 або -2 у всіх значеннях *WHT*. Щоб збільшити нелінійність, усі значення *WHT* у наборі W_1^- слід змінити на -2, усі значення *WHT* у наборі W_1^+ слід змінити на 2, а всі значення *WHT* у наборі W_2^+ слід змінити на -2, всі WHT-значення в наборі W_2^- змінюються на 2. Якщо перші дві умови очевидні, то наступні дві умови застосовуються, аби всі інші значення $|F(w)|$ були менше $WH_{\max}$. Ці умови можна представити у вигляді простих тестів.

Згідно до теореми [27], булева функція *f(x)* з *WHT* $F(w)$ задано, а множини визначено наступним чином

$$W^+ = W_1^+ \cup W_2^+$$
та
$$W^- = W_1^- \cup W_2^-$$

Тоді для деякого введення *x* елемент із набору покращень існують і виконуються такі дві умови: $f(x) = L_w(x)$ для усіх $w \in W^+$, і $f(x) \neq L_w(x)$ для усіх $w \in W^-$.

Критерій градієнтного пошуку полягає в тому, щоб максимізувати відстань Хеммінга між згенерованою послідовністю та послідовністю лінійної функції. Аналогічна операція виконується після оновлення алгебраїчної форми булевої функції.

Було виконано *WFT*-перетворення Уолша-Адамара та знайдено максимум коефіцієнтів перетворення. Створено ряд зразків удосконалення. Є елемент послідовності функцій, який відповідає елементу послідовності найближчої лінійної форми. Інвертуйте елементи апроксимації, щоб зробити функцію більш нелінійною, «віддаляючись» від найближчої лінійної функції. Виконуються наступні ітерації, подібні до описаних вище.

Проведені дослідження показали, що розглянута процедура підйому градієнта є обчислювально інтенсивною, вимагає значної кількості ітераційних ітерацій з огляду на велику кількість аргументів булевої функції. Щоб зменшити кількість обчислень як вхідних даних, [11] пропонує градієнтний спуск із використанням послідовностей кривих.

3.2 Евристичний метод градієнтного спуску

Запропонований метод побудови криптографічних булевих функцій є подальшим розвитком методу евристичного градієнтного зростання. Цей метод заснований на використанні властивостей нелінійних послідовностей. За допомогою ітераційної процедури додавання позицій послідовності кривих для градієнтного пошуку збалансованої булевої функції відповідно до критерію максимізації відстані Хеммінга між згенерованою послідовністю та послідовністю всіх лінійних функцій добре відомо, що вона відрізняється від евристичних методів. що Булева функція з потрібними криптографічними властивостями.

Основна ідея градієнтного спуску полягає в тому, щоб ефективно зменшити нелінійність даних бент-послідовності для кожного з $2^{n/2-1}$ її обов'язкових доповнень.

3.3 Модифікований метод сходження на пагорб

Як показано в [15], складним підходом до проектування булевої функції є зміна поведінки конкретної булевої функції шляхом внесення добре вибраних змін в одному або двох місцях. Це один з засобів покращення нелінійності. Показано, що кожна модифікація таблиці істинності призводить до $\Delta WHT \in \{-2, 2\}$ для усіх *w*. Будь-які дві зміни викликають $\Delta WHT \in \{-4, 0, 4\}$.

Якщо два значення функції задовольняють нерівності $f(x_1) \neq f(x_2)$, вага Хеммінга залишається незмінною. Починаючи зі збалансованих функцій, автори змогли перейти до більш нелінійних збалансованих функцій, використовуючи метод, представлений у [33]. Цей підхід не змінив таблицю істинності, якщо не були присутні нелінійності покращені такими змінами. У цій статті я трохи змінив метод сходження на пагорб, але не змінив таблицю істинності для викривленої послідовності, якщо модифікація не посилила нелінійність. Висока нелінійність і низька автокореляція збалансованих булевих функцій можуть бути досягнуті в результаті ітераційної операції.

Коли два значення функції задовольняють $f(x_1) \neq f(x_2)$, то вага Хеммінга не зміниться. Починаючи зі збалансованої функції, автори могли піднятися до більш нелінійної збалансованої функції за допомогою методу, представленого в [33]. Такий підхід не вносив змін у таблицю істинності, якщо тільки нелінійність не була покращена такою зміною. У цьому документі ми трохи модифікували метод сходження на пагорб, але не внесли зміни в таблицю істинності викривленої послідовності, якщо тільки нелінійність не була гіршою через цю зміну. В результаті ітераційних операцій ми можемо досягти високої нелінійності та низької автокореляції для збалансованої булевої функції.

Добре відомо, що бент-функції мають хороші криптографічні властивості, такі як найвища нелінійність і найнижча автокореляція. Однак вони не

збалансовані. Бент-послідовність довжиною 2^n має 2^{n-1} одиниці та $2^{n-1}+2^{n/2-1}$ нулі, або навпаки. Доповнення $2^{n/2-1}$ одиниць (нулів) у бент-послідовності дозволяє отримати збалансовану послідовність з нелінійністю не менше [27]

$$nl(f) \geq 2^{n-1} - 2^{n/2} \tag{3.1}$$

Ця властивість є важливим інструментом, застосованим у нашому методі для порятунку від незбалансованої та досяжної високої нелінійності модифікованої послідовності вигинів.

Основна ідея нашого методу полягає в наступному. Як зазначено в [15, 27], одне доповнення до таблиці істинності призведе до кожного $F(w)$ змінити на ± 2. З точки зору нелінійності, це означає, що доповнення будь-якої позиції збільшить нелінійність на +1, якщо $F(w)$ змінено на −2, і навпаки, зменшить нелінійність на −1 якщо $F(w)$ змінено на +2. Таким чином, щоб змінити бент-послідовність на дуже нелінійну булеву функцію, ми повинні знайти позиції, зміна яких призведе до зменшення нелінійності.

Нелінійність можна збільшити простим способом. Щоб збільшити нелінійність, ми повинні не тільки знайти позиції n^- доповнення, яке призводить до зменшення нелінійності, але тоді ми повинні знайти позиції n^+ доповнення, що призводить до збільшення в нелінійності. Звичайно, що сума цих позицій, n^- і n^+, дорівнює $2^{n/2-1}$.

3.4 Опис реалізації генетичного алгоритму

1) Виконуємо процедуру генерації s-боксів методом генетичних алгоритмів. Генерується початкова популяція розміру *mutations_per_parent*child_per_parent*.

Метод може працювати як у первинному, так і вторинному режимі. Генерація початкової популяції у первинному режимі полягає у генерації випадкового S-боксу.

Лямбда-вираз для генерації початкової популяції у вторинному режимі. У вторинному режимі популяція складається з мутацій початкового S-боксу. Мутація полягає в обміні двох випадкових позицій в s-боксі

```
int idx1 = distrib(gen) % 256;
int idx2 = distrib(gen) % 256;
std::swap(new_sbox[idx1], new_sbox[idx2]);
```

Якщо параметр *input* (покажчик на початковий S-бокс)– не нульовий, то метод викликається як вторинний. У цьому випадку використовується *secondary_initial_generation*. Інакше – *primary_initial_generation* – первинний метод.

2) У головному циклі виконується вибірка кращих представників популяції, і *iterations_count* разів повторюються наступні дії для створення нової популяції:
    a. Викликається метод селекцій. Дана програмна реалізація підтримує 3 види селекторів:

- *basic_selection_method*. Обирає з відсортованого списку тільки кращих представників. Якщо два представника, що йдуть підряд, мають однакові значення цільової функції, то другий представник відкидається. Ця умова задана через необхідність виходу з локальних мінімумів.
- *rank_selection_method*. Обирає представників випадково з відсортованого за зменшенням цільової функції масиву. Ймовірність, що s-бокс буде обраний складає $\frac{2i}{M\left(M+1\right)}$, де i – позиція в масиві, M – число елементів в масиві.

- *roulette_wheel_selection.* Обирає представників випадково з відсортованого за зменшенням цільової функції масиву. Ймовірність, що s-бокс буде обраний складає $\frac{\cos t_i}{\sum_{j=1}^{M} \cos t_i}$, де i – позиція в масиві, M – число елементів в масиві, *cost* – значення функції вартості.

  Метод селекцій залишає *child_per_parent* представників з популяції

  b. Викликається метод кроссовера для тих, хто пройшов селекцію. У даній реалізації генетичного алгоритму застосовано два різні методи кросинговеру: *PMX* («частково відображений кросовер») і циклічний кросовер.

- *Циклічний кросовер* працює наступним чином:

За допомогою генератора випадкових чисел заповнюються S-бокси *parent1* та *parent2*. Випадково вибирається початкова точка циклу.

Елемент *parent1* у початковій точці циклу копіюється в дочірній елемент у тій же позиції: Елемент у тій же позиції в *parent2* є наступним для копіювання в дочірній елемент. Однак він копіюється в ту саму позицію, в якій він зустрічається в батьківському 1. Цей процес триває, доки процес не поверне нас до вихідної початкової точки циклу - іншими словами, коли створено «петля» або «цикл».

Будь-які інші позиції, по яким не пройшов цикл в перший раз, в дочірньому елементі потім заповнюються шляхом копіювання відповідних значень із *parent2*

- *Кросовер PMX* починається заповненням S-боксів *parent1* та *parent2* і випадковим вибором проміжку. Елементи *parent1* у цьому проміжку копіюються в дочірній елемент:

Далі копіюються елементи *parent2*, які ще не були скопійовані в дочірній елемент. Якщо значення *parent2* вже було використано, тоді шукаємо перше не використане.

```
while (index_values[value] == true) {
   for (int j = 0;j < 256;j++) {
      if (parent1[j] == value) {
         value = parent2[j];
         break;
      }
   }
}
```

Рисунок 3.1 – Процес заповнення S-боксів *parent1* та *parent2* і випадковим вибором проміжку для кросоверу *PMX*

Усі новоутворені S-бокси додаються в популяцію. Створюємо нову популяцію з кращих представників минулої популяції.

c. Тепер реалізуються безпосередні розрахунки за методом генетичних алгоритмів. Над кожним S-боксом в популяції, що утворилася після селекції та кросоверу, проводяться мутації.

Таким чином, додаємо в нову популяцію *mutations_per_parent* мутацій представника за основу береться поточний представник минулого покоління. обираються дві випадкові позиції і міняються місцями:

```
for (int i = 0; i < method_data.mutations_per_parent; i++)
{
      SBox mutant = successor;
      int pos_1 = distrib(gen) % 256;
      int pos_2 = distrib(gen) % 256;
      std::swap(mutant[pos_1], mutant[pos_2]);
```

Рисунок 3.2 - Розрахунки за методом генетичних алгоритмів

Процедура реалізує безпосередні розрахунки за методом генетичних алгоритмів у кожному потоці. Тут оновлюються значення функції вартості, порівнюються значення нелінійності, цільової функції, дельта рівномірності, мінімального степеня та алгебраїчного імунітету нового s-боксу. Якщо будь-який з цих параметрів не дорівнює необхідному, то починаємо нову спробу пошуку.

Коли s-бокс з потрібними характеристиками знайдено, додаємо його в популяцію.

Отже, над кожним S-боксом проводиться *mutations_per_parent* мутацій, і всі мутанти додаватимуться в популяцію.

3.5 Обґрунтування отриманих у ході дослідження результатів

Направленням практичних досліджень є знаходження оптимальних підстановок для застосування в сучасних симетричних криптоалгоритмах. Для рішення задач, пов’язаних з формуванням вузлів нелінійної заміни, необхідний великий об’єм ресурсів.

Найбільш поширеною високопродуктивною мовою програмування для розподілених обчислень на сьогодні є C/C++. Тому для пошуку оптимальних підстановок було обрано саме цю мову.

Вимірювання швидкодії генерації S-блоків здійснювалося на ПК з процесором Intel Core i5-6200U і тактовою частотою 2.30-2.40 GHz, операційна система Windows 10 x32;

Реалізація передбачає застосування двох генераторів для генерації S-блоку. Перший генератор використовується для знаходження S-блоку, максимально близького за значенням до шуканого. Другий генератор доводить отриманий S-блок до цільових характеристик. У якості первинного використано генератор на основі метода генетичних алгоритмів, у якості вторинного – генератор на основі методу градієнтного підйому.

Для первинного генератора маємо наступні параметри:

Таблиця 3.1 – Параметри методу генетичних алгоритмів

Параметр	Значення
target_nonlinearity	Цільова нелінійність
target_delta_uniformity	Цільова дельта рівномірність

Продовження таблиці 3.1 – Параметри методу генетичних алгоритмів

target_min_degree	Цільова мінімальна степінь
target_algebraic_immunity	Цільовий алгебраїчний імунітет
limit	Верхнє обмеження на значення функції вартості, при якому s-блок вважається валідним
mutations_per_parent	Кількість мутацій в одному представнику минулого покоління
child_per_parent	Кількість представників минулого покоління з яких формується нове
iterations_count	Кількість поколінь
Comparator	Метод порівняння s-блоків
selection_method	Метод відбору найкращих представників популяції
crossover_method	Метод змішування/кросоверу
cost_function_name	Найменування функції вартості

Для вторинного методу маємо набір параметрів:

Таблиця 3.2 – Параметри методу градієнтного підйому

Параметр	Значение
target_nonlinearity	Цільова нелинейность
target_delta_uniformity	Цільова дельта рівномірність
target_min_degree	Цільова мінімальна степінь

Продовження таблиці 3.2 – Параметри методу градієнтного підйому

target_algebraic_immunity	Цільовий алгебраїчний імунітет
max_try_count	Максимальна спільна кількість спроб генерації нових s-блоків
max_frozen_outer_loops	Максимальна кількість пройдених зовнішніх циклів, коли не було виконано жодних покращень функції вартості
changed_positions_count	Кількість позицій, що змінюється в початковому s-блоці за раз
cost_function_name	Найменування функції вартості

У якості функції вартості при генерації використовувалася функція Уолша-Адамара

$$\sum_{i=1}^{256*256} \prod_{\substack{j=start \\ j+=step}}^{end} (|WHT[i]| - j),$$

де $start, step$ та end – цілочисленні параметри.

Визначено, що за умови застосування функції вартості для генерації нелінійних вузлів перетворення процес формування підстановок значно пришвидшується. Тому було використано функцію вартості Фрейре-Ечеваррія [45], задану в формулі (1.15)

Розрахунки проводилися над випадково згенерованими S-блоками.

Модифікація методу градієнтного спуску призвела до значного прискорення процесу формування цільових підстановок.

ВИСНОВКИ

Застосування математичного апарату векторних булевих функцій дозволяє знаходити ефективні методи формування нелінійних конструкцій підстановки для симетричних криптографічних примітивів з оптимальними показниками.

Застосування математичного апарату векторних булевих функцій дозволяє спростити опис основних елементів симетричних алгоритмів. Таке уявлення дає можливість узагальнення безлічі критеріїв, у тому числі й застосовуваних до підстановок, одночасно дозволяючи оцінити кореляційні, алгебраїчні та інші властивості S-блоків. Як показують дослідження, застосування теоретичного підходу не завжди може задовольнити практичні потреби, зокрема щодо генерації підстановок із заданими ненасиченими показниками.

Альтернативним напрямом є евристичний підхід, методи якого спрямовані на генерацію псевдовипадкових підстановок з подальшою перевіркою їх криптографічних властивостей.

Основна мета при використанні евристичних методів полягає у зменшенні показнику функції вартості, що, безпосередньо, пов’язано з цільовою властивістю S-блоків. При роботі алгоритму пошуку виконуються максимально можливе приближення характеристик поточного вузлу ускладнення до бажаних значень Обмеження евристичних методів обумовлені обчислювальними можливостями сучасних комп'ютерів та складністю методів генерації. На жаль, у більшості випадків алгоритми генерації підстановок засновані на деякому випадковому або псевдовипадковому процесі, що призводить до неможливості зменшення складності генерації блоків S до поліноміальної.

Таким чином, незважаючи на безліч існуючих рішень в області симетричної криптографії, залишається актуальним питання щодо знаходження підстановок,

застосування яких в алгоритмах шифрування забезпечує захист від існуючих і перспективних видів атак. Для розвитку та вдосконалення методів генерації нелінійних вузлів заміни необхідно вирішити завдання обґрунтування критеріїв та продовжити дослідження в галузі методів криптоаналізу та теорії векторних булевих функцій.

Проведений аналіз показує, що більшість сучасних методів генерації підстановок засновані на теоретичному підході, де за основу беруться векторні функції булевих з теоретично доведеними властивостями. Однак, у більшості випадків розглядається лише одне або два з них, тоді як на практиці необхідно враховувати чотири і більше.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Announcing development of a federal information processing standard for advanced encryption standard. 1997. URL: http://csrc.nist.gov/archive/aes/pre-round1/aes_9701.txt (дата звернення: 20.09.2022)
2. Carlet C., Crama Y., Hammer P. Vectorial Boolean Functions for Cryptography. Cambridge: *Cambridge University Press*. 2010. – P. 398–469.
3. Daemen J., Rijmen V. The design of Rijndael: AES-the advanced encryption standard. – Berlin: *Springer*. 2002.
4. Multiobjective Evolutionary Algorithms and Applications / Khor E.F. etc. *Springer*. 2005.
5. Easttom C. A generalized methodology for designing non-linear elements in symmetric cryptographic primitives. Las Vegas: *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. 2018. P. 444-449.
6. Sullivan G., Weierud F. Breaking German army ciphers. *Cryptologia*. 29(3): P. 193–232, 2005.
7. Random S-Boxes Generation Methods for Symmetric Cryptography / Gorbenko I. etc. Lviv, Ukraine: *2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON)*. 2019. P. 947-950.
8. Ivanov G., Nikolov N., Nikova S. Cryptographically Strong S-Boxes Generated by Modified Immune Algorithm. *Springer*. 2016. P. 31-42.
9. Almost Boolean functions: The design of Boolean functions by spectral inversion / Clark J.A. etc. *Computational Intelligence*, 20(3). 2004. P. 450–462,
10. McLaughlin J. D. Applications of search techniques to cryptanalysis and the construction of cipher components : дис. докт. техн. наук – York, 2012. – 271 p.
11. Heuristic Methods for the Design of Cryptographic Boolean Functions / Kuznetsov

O.O. etc.: Collective monograph. *ASC Academic Publishing*. USA. 2018. P. 45– 74.

12. Knudsen L.R. Block Ciphers Analysis, Design and Applications: PhD thesis. *Computer Science Department*. Denmark. 1994.
13. Dorigo M., Gambardella L.M. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation:* 1(1). April 1997. P. 53–66.
14. Nizam Chew L.C., Ismail E.S. S-box Construction Based on Linear Fractional Transformation and Permutation Function. *Symmetry:* 12. 2020. 826 p.
15. Rodinko M., Oliynykov R. An Approach to Search for Multi-Round Differential Characteristics of Cypress-256. *International Scientific-Practical Conference Scopus*. Kharkiv. 2018. P. 659–662.
16. Rodinko M., Oliynykov R. Comparing Performances of Cypress Block Cipher and Modern Lighweight Block Ciphers on Different Platforms. *Proceedings of 2019 IEEE International Scientific-Practical Conference Scopus.* Kyiv. 2019. P. 113–116.
17. Rodinko M., Oliynykov R. The Method of Searching for Differential Trails of ARX-based Block Cipher Cypress. *Proceedings of 2020 IEEE 11th International Conference Scopus*. Kyiv. 2020. P. 157–160.
18. Rodinko M., Oliynykov R., Gorbenko Y. Optimization of the high nonlinear s-boxes generation method. *Tatra Mountains Mathematical Publications, Mathematical Institute.* Bratislava. 2017. Vol. 70. P. 93-105.
19. Millan W., Clark A., Dawson E., Smart Hill Climbing Finds Better Boolean Functions, *Proceedings of the Workshop on Selected Areas on Cryptography SAC 97*. 1997. P. 50-63.
20. Izbenko Y., Kovtun V., Kuznetsov A. The Design of Boolean Functions by Modified Hill Climbing Method. *2009 Sixth International Conference on Information Technology: New Generations*. Las Vegas. 2009. P. 356-361.
21. Zahid A.H., Arshad M.J. An Innovative Design of Substitution-Boxes Using Cubic

Polynomial Mapping. *Symmetry:* 11. 2019. 437 p.

22. Бабенко В. Г. Методологія синтезу операцій перетворення інформації для комп'ютерної криптографії : дис. докт. техн. наук. Черкаси. 2020. 468 с.
23. Вероятностная модель формирования нелинейных узлов замен для симметричных криптографических средств защиты информации / Л. С. Сорока и др. *Системи обробки інформації.* Харків, 2009. С. 286–294.
24. Перспективный блочный шифр "Калина" основные положения и спецификация / Горбенко И. Д. и др. *Прикладная радиоэлектроника*, 2007, №2.
25. Горбенко І. Д., Горбенко Ю. І. Прикладна криптологія: монографія. Харків : Форт, 2012. 868 с.
26. ГОСТ 28147–89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. Київ : Изд-во стандартов, 1989. 28 с.
27. Дослідження методів формування випадкових нелінійних вузлів замін симетричних шифрів / О. О. Кузнецов ін. *Прикладна радіоелектроніка.* 2018. №3. С. 88–95.
28. ДСТУ ГОСТ 28147:2009. Системы обработки информации. Защита криптографическая. Алгоритм криптографического. Київ : Изд-во стандартов, 2009.
29. Казимиров О. В. Методи та засоби генерації нелінійних вузлів заміни для симетричних криптоалгоритмів : дис. канд. техн. наук. Харків, 2014. 190 с.
30. Кузнецов А. А., Избенко Ю. А., Московченко И. Метод построения криптографически стойких булевых функций на основе градиентного спуска. Харків : ХУПС, 2007. С. 63–66.
31. Экспериментальные данные по определению динамических показателей прихода блочных симметричных шифров к состоянию случайной подстановки / Лисицкая И.В. и др. *Радіоелектроніка, інформатика, управління*. 2017. № 1.

С. 129–141.

32. Мао Венбо. Современная криптография: теория и практика. *Вильямс*, 2005. 768 с.
33. Московченко І. В. Математичні моделі та обчислювальні методи імовірнісного формування нелінійних вузлів замін симетричних криптографічних засобів захисту інформації : автореф. дис. канд. техн. наук. Харків, 2009. 20 с.
34. Олейников Р. В., Шумов А. И., Руженцев В. И. Построение переопределенной системы уравнений для описания алгоритма шифрования AES. *Безпека інформації в інформаційно-телекомунікаційних системах*. Київ, 2004.
35. Родінко М. Ю. Методи побудови та дослідження властивостей малоресурсних блокових шифрів та їх компонентів : дис. докт. техн. наук. Харків, 2020. 201 с.
36. Родінко М. Ю., Олійников Р. В. Дослідження продуктивності малоресурсного блокового шифру «Кипарис» на різних платформах. *Радіотехніка*. 2020. Вип. 200. С. 51–57.
37. Родінко М. Ю., Олійников Р. В. Математична модель оцінки властивостей неін’єктивних схем розгортання ключів симетричних блокових шифрів. *Прикладная радиоэлектроника.* 2016. Т. 15, № 3. С. 179–183.
38. Хорошко В.А., Чекатков А. А. Методи й засоби захисту інформації. Київ. *Юніор*, 2003. 504 с.
39. Chuan-Kun W., Dengguo F. Boolean Functions and Their Applications in Cryptography. *Springer-Verlag Berlin Heidelberg*. 2016. 267 p.
40. Дослідження методів формування випадкових нелінійних вузлів замін симетричних шифрів / Кузнєцов О.О та ін., *Науково-технічний журнал «Прикладна радіоелектроніка»,* 2018. Том 17. №3, 4. С. 88-95.
41. Оптимізація параметрів алгоритму локального пошуку для генерації нелінійних підстановок / Кузнєцов О.О та ін., *Науково-технічний журнал «Радіоелектроніка»,* 2021. Вип. 206. С. 64-76.

42. Дослідження нової функції вартості для генерації випадкових підстановок симетричних шифрів / Кузнєцов О.О та ін., *Науково-технічний журнал «Радіоелектроніка»,* 2022. Вип. 209. С. 71-82.
43. A New Cost Function for Evolution of S-Boxes / Picek S. etc. *Evolutionary Computation.* 2016. Vol. 24, № 4. P. 695–718.
44. The design of S-boxes by simulated annealing / Clark J.A. etc. *New Gener Comput.* 2005. Vol. 23, № 3. P. 219–231.
45. Evolving Nonlinear S-Boxes With Improved Theoretical Resilience to Power Attacks / Freyre-Echevarría A. etc. *IEEE Access.* 2020. Vol. 8. P. 202728–202737.