КІБЕРБЕЗПЕКА ТА ЗАХИСТ ІНФОРМАЦІЇ

УДК 004.056.55
DOI https://doi.org/10.32782/2521-6643-2026-1-71.18

Савченко Ю. В., кандидат технічних наук, доцент,
доцент кафедри кібербезпеки та інформаційних технологій
Університету митної справи та фінансів
ORCID: 0000-0002-7177-6311

Воскобойник В. О., кандидат технічних наук, доцент,
доцент кафедри інформаційної безпеки та наноелектроніки
Національного університету «Запорізька політехніка»
ORCID: 0000-0003-3786-8666

Корнейко О. В., кандидат технічних наук, професор,
професор кафедри інформаційних технологій та кібербезпеки
Національної академії внутрішніх справ
ORCID: 0000-0002-1882-9680

Зудова С. М., здобувач вищої освіти кафедри кібербезпеки
та інформаційних технологій Університету митної справи
та фінансів
ORCID: 0009-0008-6160-922X

ЛІНІЙНІ РЕКУРЕНТНІ СПІВВІДНОШЕННЯ У СИСТЕМІ СИМЕТРИЧНИХ АЛГОРИТМІВ ШИФРУВАННЯ

В статті запропоновано принципи функціонування симетричного шифрування на основі гами-шифру отриманого шляхом генерування лінійних рекурентних співвідношень. Досліджено вплив лінійного рекурентного співвідношення на криптографічну стійкість алгоритмів, а також наведено приклади практичного застосування рекурентних моделей у забезпеченні процесу шифрування. Отриманий методом гамування в основі якого задіяно лінійне рекурентне співвідношення, зашифрований текст є достатньо трудомістким для розкриття в тому випадку, якщо гама-шифр не містить повторюваних бітових послідовностей і змінюється випадковим чином для кожного зашифрованого слова. Якщо період гами перевищує довжину всього зашифрованого тексту і невідома жодна частина вихідного тексту, то шифр можна розкрити тільки прямим перебором множини ймовірних ключових значень в цьому випадку криптостійкість визначається розміром ключа. Те, на скільки утворена гама відповідає вимогам безпеки у більшості випадків залежить від початкових даних, на основі яких генерується лінійне рекурентне співвідношення. Максимальне значення періоду послідовності цілковито залежить від глибини лінійного рекурентного співвідношення та співвідношення, що її задає, а конкретні значення – від початкового стану послідовності. Підвищити криптостійкість алгоритму можна шляхом об'єднання кількох різних лінійних рекурентних співвідношеннь в контексті LFSRs та/або застосуванням нелінійних функцій в зворотному зв'язку регістра, нелінійної логіки і фільтрації вмісту регістра.

Симетричний алгоритм шифрування на основі лінійного рекурентного співвідношення допускає як програмну, так і апаратну реалізацію. З точки зору практичності, програмна реалізація допускає більшу гнучкість у використанні. Однак реалізація алгоритмів шифрування в основному коді програми провокує чисельні безпекові ризики пов'язані здебільш із несанкціонованим доступом та питаннями конфіденційності повідомлення. Для вирішення такої проблеми доречно застосовувати практику виносу функцій шифрування та/або дешифрування в окремі модулі або бібліотеки, а також передавати захищений ключ використовуючи лише захищені канали.

У статті описано, як функціонує на основі встановлення залежностей між членами невідомої послідовності та індексу лінійне рекурентне співвідношення. Поточний метод набув широкого застосування в аналізі алгоритмів, цифровій обробці сигналів, економіці, а в контексті криптографії нерідко використовується в рамках симетричного шифрування.

У статті визначено, що модульний підхід передбачає розділення програмного забезпечення на логічно незалежні частини, кожна з яких виконує окрему функцію. При перетворенні модуля на динамічну бібліотеку головна програма буде лише викликати функцію за потреби повністю приховуючи доступ до коду алгоритму. Таким чином криптографічні операції будуть реалізуватися ізольовано, що дозволить зменшити ризик несанкціонованого доступу до алгоритму шифрування.

Ключові слова: *симетричне шифрування, криптографія, шифрування, гама-шифр, лінійні рекурентні послідовності, алгоритм перетворення, дешифрування, алгоритмізація.*

Savchenko Iu. V., Voskoboinyk V. O., Korneiko O. V., Zydova S. M. Linear recurrent relations in the system of symmetric encryption algorithms

The article proposes the principles of functioning of symmetric encryption based on a gamma cipher obtained by generating linear recurrence relations. The influence of a linear recurrence relation on the cryptographic stability of algorithms is investigated, and examples of practical application of recurrent models in ensuring the encryption process are also given. The encrypted text obtained by the gamma method based on a linear recurrence relation is quite laborious to reveal if the gamma cipher does not contain repeated bit sequences and changes randomly for each encrypted word. If the gamma period exceeds the length of the entire encrypted text and no part of the original text is known, then the cipher can be revealed only by direct search of the set of probable key values, in this case, the cryptographic stability is determined by the size of the key. The extent to which the formed gamma meets the security requirements in most cases depends on the initial data on the basis of which the linear recurrence relation is generated. The maximum value of the sequence period depends entirely on the depth of the linear recurrence relation and the relation that sets it, and the specific values – on the initial state of the sequence. The cryptographic strength of the algorithm can be increased by combining several different linear recurrence relations in the context of LFSRs and/or by using nonlinear functions in the register feedback, nonlinear logic and filtering of the register contents. The symmetric encryption algorithm based on the linear recurrence relation allows for both software and hardware implementation. From the point of view of practicality, software implementation allows for greater flexibility in use. However, the implementation of encryption algorithms in the main program code provokes numerous security risks associated mainly with unauthorized access and message confidentiality issues. To solve such a problem, it is appropriate to apply the practice of removing encryption and/or decryption functions to separate modules or libraries, as well as transmitting the protected key using only protected channels. The article describes how a linear recurrence relation functions based on establishing dependencies between members of an unknown sequence and an index. The current method has found wide application in algorithm analysis, digital signal processing, economics, and in the context of cryptography it is often used within the framework of symmetric encryption. The article defines that a modular approach involves dividing software into logically independent parts, each of which performs a separate function. When converting a module into a dynamic library, the main program will only call the function, when necessary, completely hiding access to the algorithm code. Thus, cryptographic operations will be implemented in isolation, which will reduce the risk of unauthorized access to the encryption algorithm.

Key words: *symmetric encryption, cryptography, encryption, gamma cipher, linear recurrent sequences, transformation algorithm, decryption, algorithmization.*

Постановка проблеми. Проблема забезпечення захисту інформації шляхом її цілеспрямованого перетворення, яке унеможливлює несанкціоноване сприйняття змісту сторонніми особами, є актуальною з найдавніших часів [7]. В рамках поточної проблематики виникли два фундаментальні напрями – криптографія та криптоаналіз, що мають між собою цілком протилежні цілі. Криптографія спрямована на розроблення та теоретичне обґрунтування математичних методів перетворення інформації. Криптоаналіз, у свою чергу, досліджує методи розкриття або обходу криптографічних систем без знання ключів шифрування.

Проблема використання криптографічних методів в інформаційних системах стала особливо актуальною через стрімке зростання обсягів електронних даних, розвиток цифрових технологій і збільшення кількості кіберзагроз. У сучасному світі інформація є одним із найцінніших ресурсів, тому її захист від несанкціонованого доступу, підроблення чи втрати набуває першочергового значення.

Криптографічні системи розділяються на симетричні і асиметричні (з відкритим ключем). У випадках, коли головним пріоритетом є швидкодія – симетричні криптографічні алгоритми постають найбільш ефективним рішенням в порівнянні з асиметричними, адже в них не використовуються складні арифметичні операції, а деякі етапи зводяться до простих перетворень. Тому їх доречно застосовувати для шифрування повідомлень або файлів великих розмірів [11–13].

В симетричному шифруванні широко використовується метод гамування, принцип якого полягає в генерації гами-шифру за допомогою псевдовипадкової послідовності і накладення отриманої гами-шифру (певної послідовності γ) на відкриті дані (відкритий текст Т) зверненим способом. Шифри гамування (адитивні шифри) постають одними з найефективніших серед симетричних шрифтів з огляду на стійкість та процес швидкості перетворень (процедур шифрування і дешифрування). У гама-шифрах в загальному випадку використовується операція додавання по модулю N і в окремих випадках (що орієнтовані на програмну реалізацію) додавання по модулю 2.

Оскільки в симетричному шифруванні для шифрування і для розшифрування використовується той самий ключ, під час гама-шифрування однією з важливих проблем, постає генерування непередбачуваних

двійкових послідовностей великої довжини. Для розв'язання цієї проблеми широко використовуються генератори двійкових псевдовипадкових послідовностей. серед яких можна виділити лінійні рекурентні послідовності (далі – ЛРП) (такі послідовності, ще іноді називають зворотними послідовностями).

Теорія лінійних рекурентних послідовностей є точним аналогом теорії лінійних диференціальних рівнянь з постійними коефіцієнтами. Частковими випадками лінійних рекурентних послідовностей є такі послідовності як: арифметична прогресія геометрична прогресія, числа Фібоначчі, числа Люка, числа трибоначчі, послідовності Люка. Також рекурентні співвідношення мають принципове значення в економіці, цифровій обробці сигналів, комбінаториці, біології, аналізі алгоритмів (з метою описати час роботи певного алгоритму), тощо [1, 8, 9].

Мета статті. Метою статті є дослідження рекурентних співвідношень в системі симетричних криптографічних алгоритмів з метою виявлення закономірностей їх побудови та оцінки їх впливу на стійкість алгоритму гама-шифрування.

Виклад основного матеріалу

Огляд лінійних рекурентних послідовностей. На практиці найчастіше в якості гами-шифру γ виступає двійкова (бітова) послідовність, оскільки до неї дуже зручно використовувати для накладання операцію виключного АБО (XOR) за формулою 1:

$$P = T \oplus \gamma. \tag{1}$$

Оскільки в симетричному шифруванні один і той самий ключ має бути наявний і у відправника, і у отримувача, а гама-шифр має таку ж довжину, як і повідомлення – її недоцільно передавати по каналам зв'язку.

Поточна проблема може бути вирішена шляхом незалежного генерування гами-шифру на боці відправника і на боці отримувача завдяки генераторам псевдовипадкових послідовностей. Відповідно по захищеним каналам зв'язку передається не сама гама, а лише початковий стан генератора. Генератор псевдовипадкових послідовностей може функціонувати на основі лінійних рекурентних послідовностей (ЛРП).

Рекурентна послідовність виступає одним з видів рівняння, що визначає послідовність чисел, де кожен член залежить від попередніх на основі деякого правила. Загальних правил розв'язання рекурентних співвідношень не існує. Проте існує клас рекурентних співвідношень, який розв'язується єдиним методом. Це однорідне співвідношення із сталими коефіцієнтами, що задається у вигляді: (2)

$$x_{k+n} = a_1 \cdot x_{k+n-1} \oplus a_2 \cdot x_{k+n-2} \oplus ... \oplus a_{n-1} \cdot x_{k+1} \oplus a_n \cdot x_k, \tag{2}$$

де $k = 1,2, \ldots$

Члени послідовності x_i і коефіцієнти a_i – значення із множини {0, 1}.

Величина n – глибина послідовності, що визначається різницею між найстаршим і наймолодшим індексами елементів в запису послідовності. Початковим станом псевдовипадкової послідовності виступають перші n значень x_i ($i = 1, 2, ..., n$).

Задача розв'язання рекурентного співвідношення полягає в знаходженні невідомої послідовності. Теоретично, така послідовність має нескінченно багато розв'язків. Зазвичай, кількість таких розв'язків обмежується заданням значень початкових членів невідомої послідовності. Рекурентне співвідношення k-го порядку називається лінійним, якщо кожен наступний член цього рекурентного співвідношення є лінійною комбінацією k попередніх членів.

Розв'язки рекурентного співвідношення поділяють на загальний та частковий. Загальним розв'язком вважається такий, що залежить від довільних сталих, які можуть бути підібраними при будь-яких початкових умовах, він визначаєть як сума однорідних і часткових розв'язків. Частковим розв'язком в свою чергу називається будь-який розв'язок, що може бути отриманий із загального, встановленням значень його сталих [2].

Наприклад, для заданої ЛРП співвідношенням $x_{k+3} = x_{k+2} \oplus x_k$, усього можливих ключів: $2^3 = 8$. При початковому стані, наприклад $x_1 = 1, x_2 = 1, x_3 = 0$ розв'язання системи набуває наступного вигляду:

$$x_4 = x_2 \oplus x_1 = 1 \oplus 1 = 0; \quad x_{11} = x_9 \oplus x_8 = 1 \oplus 1 = 0;$$
$$x_5 = x_3 \oplus x_2 = 0 \oplus 1 = 1; \quad x_{12} = x_{10} \oplus x_9 = 0 \oplus 1 = 1;$$
$$x_6 = x_4 \oplus x_3 = 0 \oplus 0 = 0; \quad x_{13} = x_{11} \oplus x_{10} = 0 \oplus 0 = 0;$$
$$x_7 = x_5 \oplus x_4 = 1 \oplus 0 = 1; \quad x_{14} = x_{12} \oplus x_{11} = 1 \oplus 0 = 1;$$
$$x_8 = x_6 \oplus x_5 = 0 \oplus 1 = 1; \quad x_{15} = x_{13} \oplus x_{12} = 0 \oplus 1 = 1;$$
$$x_9 = x_7 \oplus x_6 = 1 \oplus 0 = 1; \quad x_{16} = x_{14} \oplus x_{13} = 1 \oplus 0 = 1;$$
$$x_{10} = x_8 \oplus x_7 = 1 \oplus 1 = 0; \quad x_{17} = x_{15} \oplus x_{14} = 1 \oplus 1 = 0;$$

Звідси можна помітити, що після виконання 7 такту починається перехід у початковий стан. Отже лінійна рекурентна послідовність набуває вигляду 1100101…. Після формування ЛРП, для шифрування

повідомлення (наприклад “Coding”) наступним кроком буде переведення вхідного тексту у двійковий вигляд для накладення отриманої гами-шифру (1100101) через операцію додавання за модулем (XOR). Отже, процес шифрування виглядає наступним чином:

Coding = 01000011 01101111 01100100 01101001 01101110 01100111
XOR
Гама шифр = 11001011 10010111 00101110 01011100 10111001 01110010
Результат = 10001000 11110000 01001010 00110101 11010111 00010101 в hex цей вираз набуде вигляду “88F04A35D715”.

Процес дешифрування гами-шифру в основі якого лежить створення гами на основі ЛРП здійснюється для повторної генерації гами-шифру сформованого лише при наявності відомого ключа та накладанні такої ж гами на зашифровані дані, згідно формулі 1.

$$T = P \oplus \gamma. \tag{3}$$

Отже, першим кроком для розшифрування є переведення зашифрованого повідомлення у бітовий вигляд. Надалі, знаючи глибину ЛРП (n), для розкриття закону формування лінійно-рекурентної послідовності має бути перехоплено (або відтворено методом підбору) 2^n поряд розміщених вірних бітів, що формують ділянку ЛРП $\{x_{k+1}, x_{k+2}, \ldots, x_{k+2n}\}$. Внаслідок, отриманої ділянки послідовності, криптоаналітик може вираховувати наступні коефіцієнти відповідного рекурентного співвідношення, розв’язавши систему відносно a_i [3].

Наприклад для розглянутого вище зашифрованого повідомлення, за умови, що криптоаналітику відомий початковий стан ($x_1 = 1, x_2 = 1, x_3 = 0$) процес шифрування виглядає наступним чином:

88f04a35d715 = 01000011 01101111 01100100 01101001 01101110 01100111
XOR
Гама-шифр = 11001011 10010111 00101110 01011100 10111001 01110010
Результат = 10001000 11110000 01001010 00110101 11010111 00010101 що відповідає слову «Coding».

Однак, в умовах, коли початковий стан залишається невідомим, для дешифрування повідомлення можна скористатися (за їх наявності) відомостями про глибину послідовності n.

Наприклад, якщо відомо, що кількісне значення ключів – 3 біти: (x_1, x_2, x_3), усього можливих варіантів ключів для дешифрування 2^n, що в поточному випадку відповідає $2^3 = 8$. Отже, для дешифрування повідомлення треба перебрати всі можливі варіанти комбінацій ключа (в даному випадку, такими комбінаціями постають 000, 111, 001, 011, 110, 101, 010, 100) й на основі результатів після утворення від кожного відповідного ключа послідовності, виконати із кожною послідовністю операцію додавання за модулем. Після цього, перебравши всі отримані 2^n варіантів результатів – віднайти той, за яким виходить осмислений результат.

Загалом, для реалізації криптографічно стійкого потокового симетричного на основі гами-шифру пред’являються три основні вимоги:

1) Важливою з точки зору криптографічної стійкості характеристикою генератора на ЛРП є період генерованої послідовності, оскільки період гами повинен бути досить великим для шифрування повідомлень різної довжини.

У ЛРП максимальний період послідовності глибини n дорівнює T_{max}. Значення T_{max} цілковито залежить від початкових умов ЛРП та підкріплюється визначенням характеристичного многочлену ЛРП. Характеристичним – є многочлен, який отримують з однорідного лінійного рекурентного рівняння, замінивши член послідовності a_n на змінну λ, а попередні члени – на відповідні степені цієї змінної, при цьому в якості коефіцієнтів многочлена постають коефіцієнти рекурентного рівняння). Характеристичний многочлен лінійної рекурентної послідовності певної глибини послідовності n називається многочленом ступеня n: (

$$f(\lambda) = \lambda_n + a_1 \cdot \lambda^{n-1} + a_2 \cdot \lambda^{n-2} + \ldots + a_{n-1} \cdot \lambda + a_n. \tag{4}$$

Отриманий многочлен, який не можна розкласти на множники нижчого ступеня, називається незвідним та виступає аналогом простого числа. Незвідними многочленами першого і другого ступеня є: λ, λ + 1, $\lambda^2 + \lambda + 1$. При цьому, якщо многочлен є незвідним, то він є дільником многочлена $\lambda^{2n-1} + 1$. Порядком незвідного многочлена $f(\lambda)$ називається найменше число s, таке що многочлен $\lambda s + 1$ ділиться на многочлен $f(\lambda)$. Якщо характеристичний многочлен послідовності є незвідним, то період послідовності дорівнює порядкові цього многочлена.

Відомо, що многочлен $\lambda^m + 1$ ділиться на многочлен $\lambda^k + 1$ тільки у тому випадку, якщо m ділиться на k. Таким чином, порядок s незвідного многочлена є дільником числа $2^n - 1$. Звідси, якщо характеристичний многочлен лінійної рекурентної послідовності є незвідним, то період послідовності T буде дільником значення T_{max}. Отже, якщо характеристичний многочлен ЛРП є незвідним і T_{max} – просте число, то період послідовності дорівнює T_{max} при будь-яких початкових значеннях, окрім всіх нулів. Звідси, якщо характеристичний многочлен ЛРП є незвідним і при цьому T_{max} – просте число, то період послідовності дорівнює T_{max} при будь-яких початкових значеннях (винятком є послідовність, що повністю складається з нулів) [4].

2) Гама повинна бути практично непередбачуваною, що означає неможливість передбачити наступний біт гами, навіть якщо відомі тип генератора й попередній відрізок гами.

Для ускладнення задачі криптоаналізу і забезпечення поточним вимогам, поряд із ЛРП в чистому вигляді нерідко застосовують наступні методи підвищення криптостійкості алгоритму:

Застосування ЛРП у генераторах регістрів зсуву (LFSRs – Linear Feedback Shift Registers). Цей метод представляє собою комбінування декількох генераторів псевдовипадкових послідовностей на основі ЛРП, що результаті об'єднуються в один, синхронізуючи їхні виходи через відповідну операцію. Такі генератори можуть мати як однакову так і різну глибину послідовності.

Як приклад, розглянемо генерування гами-шифром методом комбінації двох послідовностей для повідомлення довжиною 6 байт. Нехай глибина лінійного рекурентного співвідношення $n = 3$, при цьому значення ключів першого випадку $x_1 = 1$, $x_2 = 1$, $x_3 = 0$ та значеннях ключів другого випадку відповідно $x_1 = 1$, $x_2 = 0$, $x_3 = 1$ (період послідовності в цьому випадку 1011100…). За поточних умов, гама-шифр отримується наступним чином:

$ЛРП_1$ = 01000011 01101111 01100100 01101001 01101110 01100111
XOR
$ЛРП_2$ = 10111001 01110010 11100101 11001011 10010111 00101110
Гама-шифр = 11111010 00011101 10000001 10100010 11111001 01001001

Як бачимо, в отриманому гама шифрі довжиною 48 біт чіткий період не прослідковується, отже можна зробити висновок, що перевагою методу комбінації кількох ЛРП для формування гамми є те, що результуюча послідовність має значні періоди, в результаті чого, отриманий гама-шифр набуває більшої непередбачуваності.

Таким чином, для здійснення дешифрування недостатньо знати лише початкові умови або розмірність якогось одного рекурентного рівняння. Криптоаналітику в цьому випадку, як мінімум треба володіти інформацією як про кількість послідовностей, що приймають участь у створені гами-шифру, і мати уявлення про відповідні рекурентні рівняння (знати їх глибину та/або початкові значення (ключі).

Теоретично в розглянутому методі, кількість різних згенерованих ЛРП для синхронізації кількісно не обмежується, якщо щоразу передавати послідовності різних розмірностей. Однак, у випадку якщо для генерації гами-шифру використовувати ЛРП при статичній глибині, то максимально можлива кількість послідовностей для синхронізації визначається відповідно до максимальної кількості значень її ключів – 2^n.

Також, ускладнити створення гами можна використовуючи перетворення через нелінійні функції в зворотному зв'язку регістра або нелінійну логіку і фільтрацію вмісту регістра. Розв'язки нелінійних рекурентних співвідношень можна знаходити за допомогою генератрис [5].

3) Генерування гами не повинне викликати великих технічних складностей. Розглянутий симетричний алгоритм шифрування на основі ЛРП є досить зрозумілим через застосування простих математичних перетворень і відповідно допускає як програмну, так і апаратну реалізацію. Однак, з точки зору практичності, програмна реалізація допускає більшу гнучкість у використанні.

Зважаючи на це, в рамках дослідження було розроблено відповідну прикладну програмну реалізацію, що включає в себе автоматичну обробку розглянутих алгоритмів симетричного шифрування/дешифрування методом накладання гами-шифру на основі генератора ЛПР на мові програмування Python за допомогою створення відповідних функцій crypto() та decrypt().

Програмна реалізація шифрування на основі ЛРП.

```
#Програмна реалізація шифрування на основі ЛРП
import itertools, random
def crypto(text: str, key_len: int) -> str:
    LRP = []
    text_bit, Bit_res = ‘’, ‘ ‘
    for i in text:
        for j in format(ord(i), ‘08b’):
            text_bit += str(j)
    #print(‘Вхідне повідомлення у бітах:’, text_bit)
    for i in range(key_len):
        LRP.append(random.randint(0, 1))
    #print(‘Ключ:’, LRP)
    for i in range(len(text_bit) - key_len):
        sub_seq = LRP[i:key_len + i]
        for j in range(key_len - 2):
            if sub_seq[j] != sub_seq[j + 1]:
                sub_seq.append(1)
            else:
```

```
                sub_seq.append(0)
        LRP.append(sub_seq[-1])
    #print('Гамма шифр: ', *LRP)
    for i in range(len(text_bit)):
        if text_bit[i] != str(LRP[i]):
            Bit_res += '1'
        else:
            Bit_res += '0'
    return Bit_res
def encrypt(code: str, key_len: int) -> list:
    Bit_res = []
    bit = ''
    LRP = [list(p) for p in itertools.product([0, 1], repeat=key_len)]
    #print('Можливі комбінації ключа')
    #for i in range(len(LRP)):
        #print(LRP[i])
    for i in range(len(LRP)):
        for j in range(len(code) - key_len):
            sub_seq = LRP[i][j:key_len + j]
            for k in range(key_len - 2):
                if sub_seq[k] != sub_seq[k + 1]:
                    sub_seq.append(1)
                else:
                    sub_seq.append(0)
            LRP[i].append(sub_seq[-1])
    #print('Можливі гама-вставки: ')
    for i in range(len(LRP)):
        #print(LRP[i])
        for j in range(len(code)):
            if code[j] != str(LRP[i][j]):
                bit += '1'
            else:
                bit += '0'
        Bit_res.append(bit[1::])
        bit = ''
    return Bit_res
my_text = input('Вхідне повідомлення: ')
key = random.randint(3, 10)
code = crypto(my_text, key)
print('Зашифроване повідомлення: ', code)
encode = encrypt(code, key)
print('Розшифровані варіації повідомлення: ')
for i in range(len(encode)):
    print(f'{i + 1}) {encode[i]}')
```

Функція crypto() створена з метою кодування інформації. Вона оброблює дві складові (вхідні дані):

text – первинне повідомлення зміст якого необхідно приховати. Таке повідомлення, представляється різними способами, найчастіше, у вигляді текстів, записаних у деякому алфавіті (наприклад повідомлення текстом “Math”;

key_len – глибина послідовності, що задається в діапазоні від 3 до 10 (варто зазначити, що поточний діапазон може бути змінений або розширений для конкретних індивідуальних випадків. Однак в рамках написання прикладної програми були обрані значення, результат яких можна представити більш наочно).

В першу чергу, вхідне повідомлення переводиться в бітовий вигляд. Надалі, відбувається генерація сукупність даних, які визначають вибір конкретного перетворення з усієї множини перетворень, що реалізуються шифром. Отримавши глибину ЛРП, на основі випадковості ключові значення $x_1, x_2, \ldots, x_n$ обираються випадковим чином із множини {0, 1} методом random.randint(0, 1).

Як бачимо, ключ знаходиться тільки у внутрішньому середовищі функції, тому для подальшого дешифрування ці значення будуть залишатися невідомими. В поточному випадку, згенероване значення глибини послідовності key_len = 3, а відповідний згенерований ключ набув вигляду $x_1 = 0, x_2 = 0, x_3 = 1$.

На основі початкових (згенерованих) значень ключа, використовуючи загальну формулу формування ЛРП, формується відповідна гама що за своїм розміром дорівнює бітовій довжині вхідного повідомлення. Після цього, методом вийняткової дизн’юкції (більш відома як операція XOR) визначається умова: якщо

значення біту вхідного повідомлення на певній позиції $i = 1, 2, 3\ldots$ не відповідає значенню біту геми-шифру на тій самій позиції (коли лише один із вхідних операндів є «істинним») – результуюча послідовність (зашифроване повідомлення) доповнюється значенням 1. В іншому випадку, якщо біти однакові – послідовність доповнюється значенням 0 (рис. 1).

```
Вхідне повідомлення: Math
Вхідне повідомлення у бітах: 01001101011000010111010001101000
Ключ: [0, 0, 1]
Гамма шифр:  0 0 1 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0
Зашифроване повідомлення:   01100011001111011100110100011010
```

Рис. 1. Виконання алгоритму шифрування

Функція encrypt() в свою чергу слугує для розшифрування повідомлення і також приймає два аргументи:

code – закодоване повідомлення, зміст якого необхідно розкрити (отримане в ході застосування функції crypto() до повідомлення у попередньому прикладі);

key_len – глибина послідовності, що має те ж значення, що передавалося у функцію для шифрування (key_len = 3). Тут варто зазначити, що параметр глибини послідовності не доречно було б генерувати всередині функції-кодувальника. Оскільки, за відсутності інформації про ключі $x_1, x_2, \ldots, x_n$ які знаходяться лише в середовищі функції code(), тільки за умови відомого значення про глибину послідовності, ми можемо відтворити формулу рекурентного співвідношення, без знання якого – розшифрування повідомлення фактично неможливе.

Звісно, для криптоаналітика, у варіанті відсутності інформації про довжину, залишається тільки послідовний перебір значень, для знаходження глибини послідовності $n = 2, 3\ldots$). Однак, для довіреного адресата інформація про довжину ключа постає необхідністю. У звичайних умовах системи шифрування, з міркувань безпеки і забезпечення конфіденційності інформації що передається, такий ключ необхідно передавати лише санкціонованому користувачеві і лише по захищеним каналам зв'язку використовуючи систему закритого ключа, адже така система в рамках симетричного шифрування була створена саме для таких випадків, коли один і той самий ключ необхідно використати для шифрування та розшифрування інформації. Передати ключ на основі ЛРП можна задіявши технології угоди про ключ (keyagreement), через встановлення авторизованого TLS/SSH-сеансу і передачі ключа всередині нього, використання служб управління ключами, тощо.

Після отримання функцією вхідних даних, починається прямий перебір всіх можливих ключових значень утворюючи множину з ключів $x_1, x_2, \ldots, x_n$ розамахом у 2^n варіантів (рис. 2).

```
Вхідне повідомлення: Math
Зашифроване повідомлення:   01100011001111011100110100011010
Можливі комбінації ключа
[0, 0, 0]
[0, 0, 1]
[0, 1, 0]
[0, 1, 1]
[1, 0, 0]
[1, 0, 1]
[1, 1, 0]
[1, 1, 1]
```

Рис. 2. Перебір ключів глибини послідовності $n = 3$

Теоретично будь-який з цих варіантів ключів міг бути початковим значенням ЛРП яка утворила гама-шифр, що був покладений в основу шифрування повідомлення. Зважаючи на це, під кожну варіацію ключів формується відповідна послідовність (використовуючи рекурентне рівняння), довжина якої дорівнює бітовій довжині закодованого повідомлення (рис. 3).

В результаті, із кожною зі сформованих гам виконується операцію вийняткової дизн'юкції із закодованим повідомленням. В цьому випадку, варто звернути увагу на перевагу операції XOR, оскільки вона володіє властивістю самозворотності (повторне застосування цієї операції з тим самим ключем відновлює початкові дані.). Відповідно, алгоритм накладання гама-вставки можна описати формулою 3:

$$(T \oplus \gamma) \oplus \gamma = T.$$

Де M – початкове повідомлення; γ – гама шифр; $\oplus$ – операція XOR.

Після проведення вийняткової дизн'юкції із кожним елементом множини гама-вставок, отримуємо наступну множину результатів ймовірного повідомлення. З отриманих 2^n варіантів тільки один результат

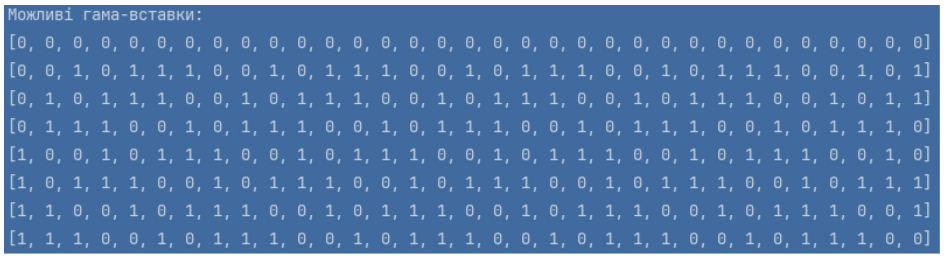

Рис. 3. Формування можливих гама-вставок

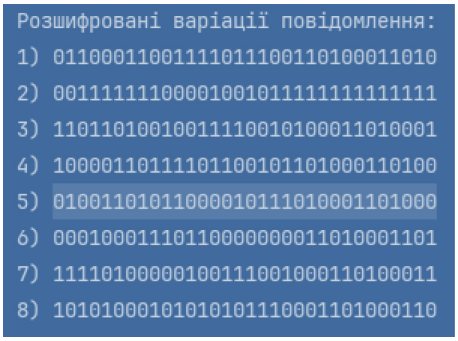

Рис. 4. Множина розв’язків

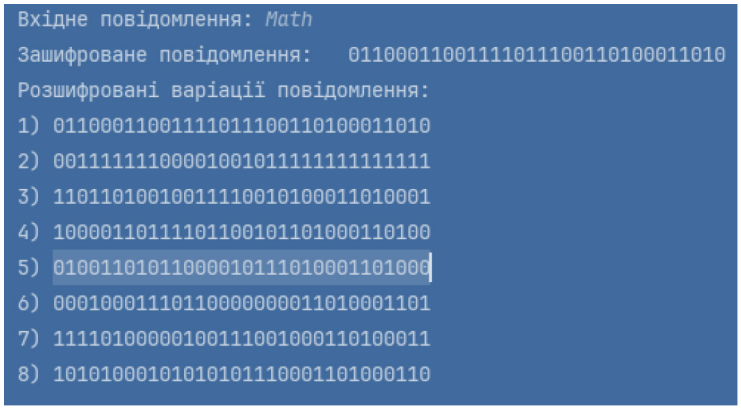

Рис. 5. Результат роботи програми з шифрування та дешифрування в загальному випадку

еквівалентний бітовому вигляду закодованого вхідного тексту. В розглянутому прикладі, вірним результатом є п’ятий результат, що дорівнює бітовому вигляду повідомлення «Math» (рис. 4).

Звичайно під час шифрування, з міркувань безпеки, у ході виконання програми недоречно виводити проміжні результати, тому в загальному випадку, користувач має можливість ознайомитися лише із вхідними та вихідними варіаціями повідомлення (рис. 5.)

Також, з метою підвищення зручності сприйняття, результати обчислень можуть бути представлені у шістнадцятковому або символьному форматі ASCII, що забезпечує більш наочне відображення.

Відомим недоліком систем, створених на основі «відкритої архітектури» як у представленому прикладному застосунку, є принципова можливість доступу до інформації з боку осіб, для яких вона не призначена. В рамках програмної реалізації гама-шифрування на основі генерування ЛРП – реалізація основних математичних перетворень безпосередньо в основному коді програми часто призводить до зниження гнучкості, безпеки та масштабованості системи. Тому все більшого поширення набуває практика виносу функцій шифрування в окремі модулі або динамічні бібліотеки з розширенням.dll.

Модульний підхід передбачає розділення програмного забезпечення на логічно незалежні частини, кожна з яких виконує окрему функцію. Отже, наведені в прикладному застосунку функції crypto() та encode() доречно розділити на два окремих, незалежних один від одного модулі. При перетворенні такого модуля на динамічну бібліотеку головна програма буде лише викликати функцію за потреби повністю приховуючи доступ до коду алгоритму. Таким чином криптографічні операції будуть реалізуватися ізольовано, що дозволить зменшити ризик несанкціонованого доступу до алгоритму шифрування [6].

Висновки. Лінійне рекурентне співвідношення (рекурентне рівняння) функціонує на основі встановлення залежностей між членами невідомої послідовності та індексу. Поточний метод набув широкого застосування в аналізі алгоритмів, цифровій обробці сигналів, економіці, а в контексті криптографії нерідко використовується в рамках симетричного шифрування.

Отриманий методом гамування в основі якого задіяно лінійне рекурентне співвідношення, зашифрований текст є достатньо трудомістким для розкриття в тому випадку, якщо гама-шифр не містить повторюваних бітових послідовностей і змінюється випадковим чином для кожного зашифрованого слова. Якщо період гами перевищує довжину всього зашифрованого тексту і невідома жодна частина вихідного тексту, то шифр можна розкрити тільки прямим перебором множини ймовірних ключових значень (наприклад підібрати ключ використовуючи атаку типу «brute force») в цьому випадку криптостійкість визначається розміром ключа.

Те, на скільки утворена гама відповідає вимогам безпеки у більшості випадків залежить від початкових даних, на основі яких генерується лінійне рекурентне співвідношення. Адже максимальне значення періоду послідовності цілковито залежить від глибини ЛРП та співвідношення, що її задає, а конкретні значення – від початкового стану послідовності.

Підвищити криптостійкість алгоритму можна шляхом об'єднання кількох різних лінійних рекурентних співвідношень в контексті LFSRs (LFSRs – Linear Feedback Shift Registers) та/або застосуванням нелінійних функцій в зворотному зв'язку регістра, нелінійної логіки і фільтрації вмісту регістра.

Симетричний алгоритм шифрування на основі лінійного рекурентного співвідношення допускає як програмну, так і апаратну реалізацію. З точки зору практичності, програмна реалізація допускає більшу гнучкість у використанні. Однак реалізація алгоритмів шифрування в основному коді програми провокує чисельні безпекові ризики пов'язані здебільш із несанкціонованим доступом та питаннями конфіденційності повідомлення. Для вирішення такої проблеми доречно застосовувати практику виносу функцій шифрування та/або дешифрування в окремі модулі або бібліотеки, а також передавати захищений ключ використовуючи лише захищені канали.

Список використаних джерел:

1. Основи криптології : навч. посіб. / Галкін О. В., Шкільняк О. С. Київ : КНУ ім. Шевченка, 2023. 199 с.
2. Теорія алгоритмів : навч. посіб. / Арсенюк І. Р., Колодний В. В., Яровий А. А. Вінниця : ВНТУ, 2006. 150 с.
3. Проектування та аналіз обчислювальних алгоритмів: вступ до алгоритмів : навч. посіб. / Федорін І. В. Київ : КПІ ім. Ігоря Сікорського, 2022. 116 с.
4. Прикладна криптологія : навч. посіб. / Палагін В. В., Палагіна А. О., Івченко О.В. Черкаси : ЧДТУ, 2023. 218 с.
5. Мєркєлов І. В. Огляд методів генерації лінійних псевдовипадкових послідовностей для псевдовипадкового переналаштування робочої частоти. *Universum. Електроніка та телекомунікації*. 2023. № 3. С. 127–132.
6. Технології захисту інформації в інформаційно-телекомунікаційних системах: навч. посіб. Жилін А. В, Шаповал О. М., Успенський О. А. Київ : КПІ ім. Ігоря Сікорського, 2020. 214 с.
7. Основи бездротових технологій / О. Г. Бєдняк, Ю. В. Савченко, В. О. Воскобойник, А. В. Тіменко, Н. В. Кіцель, О. О. Шаповал. Кременчук : Видавництво «НОВАБУК», 2025. – 300 с.
8. V. O. Voskoboinyk, Iu. V. Savchenko, L. M. Karpukov, O. A. Parshyna, D. I. Prokopovych-Tkachenko. ASSESSMENT OF THE STATE OF INFORMATION SECURITY USING EXPERT SYSTEMS. *Системи та технології*, № 1 (67), 2024 DOI https://doi.org/10.32782/2521-6643-2024-1-67.11
9. Козіна Г. Л., Савченко Ю. В., Воскобойник В. О., Прокопович-Ткаченко Д. І. Математичний підхід до підвищення швидкодії програмної реалізації криптоалгоритму SM4. *Системи та технології*, № 2 (68), 2024 DOI https://doi.org/10.32782/2521-6643-2024-2-68.9
10. Karpukov L., Tarasenko Y., Voskoboynyk V., Savchenko I., Shapoval O. Modeling of Safe Object Detection by Near-Field and Nonlinear Radar Systems. In: Solovieva, V., Hushko, S. (eds) Sustainable Development in Economics, Technology and Environmental Engineering. ISC SAI 2023. Sustainable Economy and Ecotechnology. Springer, Cham. 2025. pp 363–369. https://doi.org/10.1007/978-3-031-91953-4_40
11. Воскобойник В. О., Савченко Ю. В., Семеренко П. О. Застосування штучного інтелекту у багатофакторній автентифікації. Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: XII Міжн. наук.-практ. конф., 10–12 грудня 2024 р. : тези доповідей. Запоріжжя, 2024. С. 237–240.
12. Тарасенко Ю. С., Савченко Ю. В. ГЕОРАДІОЛОКАЦІЙНІ АСПЕКТИ БЕЗПЕКИ ПРИПОВЕРХНЕВИХ ОБ'ЄКТІВ КРИТИЧНОЇ ІНФРАСТРУКТУРИ. *Системи та технології*, 2023. 66 (2). С. 57–65. DOI https://doi.org/10.32782/2521-6643-2023.2-66.7
13. Тарасенко Ю. С., Савченко Ю. В. РИЗИК-ОРІЄНТОВАНІ ПРОЦЕСИ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ОБ'ЄКТІВ КРИТИЧНОЇ ІНФРАСТРУКТУРИ. Системи та технології, 2023. 65 (1). С. 66–76. DOI https://doi.org/10.32782/2521-6643-2023.1-65.9

References:

1. Halkin, O. V., Shkil′nyak, O. S. (2023). Osnovy kryptolohiyi [Fundamentals of cryptology]: navch. posib. Kyyiv : KNU im. Shevchenka, 199 s.

2. Arsenyuk, I. R., Kolodnyy, V. V., Yarovyy, A. A. (2006). Teoriya alhorytmiv [Theory of algorithms]: navch. posib. Vinnytsya : VNTU, 150 s.

3. Fedorin I. V. (2022). Proektuvannya ta analiz obchyslyuval′nykh alhorytmiv [Design and analysis of computational algorithms]: vstup do alhorytmiv : navch. posib. Kyyiv : KPI im. Ihorya Sikors′koho, 116 s.

4. Palahin, V. V., Palahina, A. O., Ivchenko, O. V. (2023). Prykladna kryptolohiya [Applied cryptology]: navch. posib. Cherkasy : CHDTU, 218 s.

5. Myerkyelov, I. V. (2023). Ohlyad metodiv heneratsiyi liniynykh psevdovypadkovykh poslidovnostey dlya psevdovypadkovoho perenalashtuvannya robochoyi chastoty [Overview of methods for generating linear pseudorandom sequences for pseudorandom retuning of the operating frequency]. Universum. *Elektronika ta telekomunikatsiyi*. № 3. S. 127–132.

6. Zhylin, A. V, Shapoval, O. M., Uspens′kyy, O. A. (2020). Tekhnolohiyi zakhystu informatsiyi v informatsiyno-telekomunikatsiynykh systemakh [Information protection technologies in information and telecommunication systems]: navch. posib. Kyyiv: KPI im. Ihorya Sikors′koho, 214 s.

7. Byednyak, O. H., Savchenko, Iu. V., Voskoboynyk, V. O., Timenko, A. V., Kitsel′, N. V., Shapoval, O. O. (2025). Osnovy bezdrotovykh tekhnolohiy [Wireless Technology Basics]. Kremenchuk : Vydavnytstvo "NOVABUK", 300 s.

8. Voskoboinyk, V. O., Savchenko, Iu. V., Karpukov, L. M., Parshyna, O. A., Prokopovych-Tkachenko, D. I. (2024). ASSESSMENT OF THE STATE OF INFORMATION SECURITY USING EXPERT SYSTEMS. *Systemy ta tekhnolohiyi*, № 1 (67), DOI https://doi.org/10.32782/2521-6643-2024-1-67.11

9. Kozina, H. L., Savchenko, Iu. V., Voskoboynyk, V. O., Prokopovych-Tkachenko, D. I. (2024). Matematychnyy pidkhid do pidvyshchennya shvydkodiyi prohramnoyi realizatsiyi kryptoalhorytmu SM4 [Mathematical approach to increasing the speed of software implementation of the SM4 crypto algorithm]. *Systemy ta tekhnolohiyi*, № 2 (68), DOI https://doi.org/10.32782/2521-6643-2024-2-68.9

10. Karpukov, L., Tarasenko, Y., Voskoboynyk, V., Savchenko, Iu., Shapoval, O. (2025). Modeling of Safe Object Detection by Near-Field and Nonlinear Radar Systems. In: Solovieva, V., Hushko, S. (eds) Sustainable Development in Economics, Technology and Environmental Engineering. ISC SAI 2023. Sustainable Economy and Ecotechnology. Springer, Cham. pp 363–369. https://doi.org/10.1007/978-3-031-91953-4_40

11. Voskoboynyk, V. O., Savchenko, Iu. V., Semerenko, P. O. (2024). Zastosuvannya shtuchnoho intelektu u bahatofaktorniy avtentyfikatsiyi [Application of artificial intelligence in multi-factor authentication]. Suchasni problemy i dosyahnennya v haluzi radiotekhniky, telekomunikatsiy ta informatsiynykh tekhnolohiy: XII Mizhn. nauk.-prakt. konf., 10–12 hrudnya 2024 r. : tezy dopovidey.– Zaporizhzhya, C. 237–240.

12. Tarasenko YU. S., Savchenko Iu.V. (2023). HEORADIOLOKATSIYNI ASPEKTY BEZPEKY PRYPOVERKHNEVYKH OB'YEKTIV KRYTYCHNOYI INFRASTRUKTURY [GEORADIOLOCATION ASPECTS OF SECURITY OF SURFACE OBJECTS OF CRITICAL INFRASTRUCTURE]. Systemy ta tekhnolohiyi, 66 (2). S. 57–65. DOI https://doi.org/10.32782/2521-6643-2023.2-66.7

13. Tarasenko Yu. S., Savchenko Iu.V. (2023). RYZYK-ORIYENTOVANI PROTSESY ZABEZPECHENNYA BEZPEKY OB″YEKTIV KRYTYCHNOYI INFRASTRUKTURY [RISK-ORIENTED PROCESSES OF SECURITY OF CRITICAL INFRASTRUCTURE OBJECTS]. Systemy ta tekhnolohiyi, 65 (1). S. 66–76. DOI https://doi.org/10.32782/2521-6643-2023.1-65.9

Дата першого надходження статті до видання: 23.11.2025
Дата прийняття статті до друку після рецензування: 22.12.2025
Дата публікації (оприлюднення) статті 27.01.2026