https://doi.org/10.31891/2219-9365-2024-80-44
УДК 004.056.55

ГРЕСЬ Олександр
Чернівецький національний університет імені Юрія Федьковича
https://orcid.org/0000-0002-8465-193X
e-mail::o.hres@chnu.edu.ua
ГАЛЮК Сергій
Чернівецький національний університет імені Юрія Федьковича
https://orcid.org/0000-0003-3836-2675
e-mail:s.haliuk@chnu.edu.ua
КРУЛІКОВСЬКИЙ Олег
Чернівецький національний університет імені Юрія Федьковича
https://orcid.org/0000-0001-5995-6857
e-mail:o.krulikovskyi@chnu.edu.ua
ЗАЯЦ Роман
Чернівецький національний університет імені Юрія Федьковича
https://orcid.org/0009-0009-2701-6380
e-mail:zaiats.roman@chnu.edu.ua
ЛАСТІВКА Галина
Чернівецький національний університет імені Юрія Федьковича
https://orcid.org/0000-0003-3639-3507
e-mail:g.lastivka@chnu.edu.ua
ШПАТАР Петро
Чернівецький національний університет імені Юрія Федьковича
https://orcid.org/0000-0003-4088-1458
e-mail:p.shpatar@chnu.edu.ua

РОЗРОБЛЕННЯ ГЕНЕРАТОРІВ ПСЕВДОВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ НА ОСНОВІ ХЕШ-ФУНКЦІЙ З ВИКОРИСТАННЯМ БАГАТОВИМІРНИХ ХАОТИЧНИХ СИСТЕМ

В даній роботі запропоновано алгоритм генерування хеш-послідовностей на основі багатовимірних хаотичних систем, зокрема восьмивимірної системи Лозі. Функція хешування основана на властивостях псевдовипадковості та чутливості до початкових умов, притаманних хаотичним системам. Відмінність даного генератора хеш функцій на основі багатовимірної хаотичної системи Лозі від інших подібного типу генераторів полягає у збуренні параметрів, які визначають хаотичність системи при введенні блоків інформації. На кожному кроці нові параметри отримуються в результаті сумісної дії її блоку інформації та поточного стану системи. Завдяки цьому здійснюється нелінійне введення та перетворення інформації при збереженні хаотичної динаміки. При виконанні ітерацій кожний блок інформації зазнає впливу як змінних стану системи так і значення параметра. Додатковим елементом захисту є використання ключа розміром 256 біт, на основі якого формується початковий стан системи. Розроблений алгоритм орієнтований на застосування арифметики з фіксованою комою з довжиною представлення числа в 32 біти, з яких один виділяється на представлення знаку числа, один для опису цілої частина числа, 30 біт – для дробової. Такий поділ забезпечує ефективне використання 32-го представлення числа. Використання арифметики з фіксованою комою, замість рухомої коми, забезпечує ідентичність реалізацій і виконання алгоритму на різних платформах. Алгоритм може бути адаптований для використання іншої арифметики або більшої довжини слова при відповідному доопрацюванні. В роботі також проведено дослідження запропонованого алгоритму генерування хеш функцій до вхідних повідомлень, ключа хешування та проведене дослідження хеш-функції як алгоритму для генерування псевдовипадкових послідовностей з використанням пакету статистичних тестів NIST STS.

Ключові слова: генератор, хеш функція, хаотична система, статистичні тести.

HRES Oleksandr, HALIUK Serhii, KRULIKOVSKIY Oleh,
ZAIATS Roman, LASTIVKA Halyna, SHPATAR Petro
Yury Fedkovich Chernivtsi National University

DEVELOPMENT OF GENERATORS OF PSEUDORANDOMS SEQUENCES BASED ON HASH FUNCTIONS USING MULTIDIMENSIONAL CHAOTIC SYSTEMS

In this paper, we propose hash-function based on multidimensional chaotic systems, specifically an eight-dimensional Lozi system. The hash function leverages the inherent properties of chaotic systems, including pseudo-randomness and extreme sensitivity to initial conditions. Unlike other similar chaotic-based hash generators, the proposed approach uses perturbation of the system parameters, those that define the chaotic behavior, by incorporating each incoming block of information. At every iteration, new parameters are derived from the joint interaction of the current system state and the respective information block. The input data is injected and transformed, while the chaotic dynamics of the system are preserved. Throughout the iterative process, each information block is influenced not only by the system's state variables but also by the current parameter values. A further layer of security is provided by utilizing a 256-bit key, which serves to initialize the system's starting state. The proposed algorithm is designed to operate using fixed-point arithmetic with 32-bit number representation: one bit is allocated for the sign, one for the

integer part, and the remaining 30 bits for the fractional part. This division ensures efficient use of the 32-bit format, efficient computing and provides platform-independent, consistent performance – unlike floating-point arithmetic, whose results can vary across different hardware and compilers. The algorithm can be adapted to alternate arithmetic or larger word sizes if needed. Additionally, this work investigates the proposed algorithm's resistance to differential attack and collisions, as well as its viability as a pseudorandom sequence generator. The statistical properties of sequences was assessed using the NIST Statistical Test Suite (NIST STS), which confirmed the capability of algorithm for cryptography applications.

Keywords: generator, hash function, chaotic system, statistical tests

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

Дослідження хаотичних систем та їх застосування у криптографії посідає значне місце у сучасній науці. Хаотичні системи, завдяки своїм властивостям псевдовипадковості, непередбачуваності та чутливості до початкових умов, мають широкі можливості застосування на практиці. Зокрема, їх використовують для генерування псевдовипадкових чисел, шуму, створення нових алгоритмів шифрування та хешування [1-3].

Сучасні алгоритми хешування відіграють ключову роль у забезпеченні безпеки даних, їх цілісності та ефективному пошуку в комп'ютерних системах. Хеш-функції перетворюють вхідні дані (повідомлення, файли тощо) на фіксовану довжину рядка, що є унікальним представленням цих даних. У разі правильного вибору хеш-функції, навіть найменша зміна вхідних даних призводить до суттєвих змін у хеш-значенні. Незважаючи на активне використання таких хеш-функцій як: SHA-2, SHA-3, BLAKE2, MD5, RIPEMD-160, Argon2 тощо, активно ведеться пошук нових рішень, зокрема на основі дискретних хаотичних систем [1]. З цією метою використовують тентове відображення [4], каскадно-з'єднані логістичні відображенні [5], поєднання кількох одновимірних чи спеціально створених хаотичних систем [6, 7].

Супутніми проблемами, що мають бути вирішеними при розробленні хеш-функцій є запобігання деградації хаотичної динаміки та уникнення слабких ключів [2, 8]. Серед різноманітних математичних моделей хаотичних систем, відображення Лозі (або багатовимірні варіації цієї моделі) зарекомендували себе як потужні інструменти для створення нових алгоритмів генерування псевдовипадкових послідовностей та шифрування [9-11]. Широкому застосуванню таких функцій (алгоритмів) перешкоджає низька обчислювальна ефективність розрахунку траєкторій хаотичних відображень. Багатовимірна система Лозі, побудована на основі по'єднання обчислювано простих тентових відображень, через це характеризується меншою складністю обчислень, в той же час забезпечує генерацію рівномірно розподілених послідовностей у широкому діапазоні параметрів [11]. Багатовимірні хаотичні моделі можуть бути ефективно адаптовані для реалізації стійких криптографічних протоколів, які спроможні забезпечувати стійкість до сучасних атак.

Стаття присвячена дослідженню багатовимірного хаотичного відображення Лозі та його застосування для створення криптографічних хеш-алгоритмів. У роботі проведений аналізу розподілу послідовностей, біфуркаційних діаграм та показників Ляпунова, досліджено восьмивимірну систему Лозі та запропоновано функцію одностороннього стиснення інформації на її основі. Реалізація пропонованої хеш-функції показала її достатню стійкість до колізій та диференційної атаки.

ДОСЛІДЖЕННЯ БАГАТОВИМІРНОЇ СИСТЕМИ ЛОЗІ

Багатовимірна хаотична система (відображення) Лозі відноситься класу систем, що є дискретними в часі та мають неперервну множину значень реалізацій. Аналітично відображення описується системою рекурентних рівнянь [9]:

$$\begin{cases} x_{n+1}^{(1)} = 1 - 2\left|x_n^{(1)}\right| + k^{(1)} \times x_n^{(2)}, \\ x_{n+1}^{(2)} = 1 - 2\left|x_n^{(2)}\right| + k^{(2)} \times x_n^{(3)}, \\ \dots \\ x_{n+1}^{(p)} = 1 - 2\left|x_n^{(p)}\right| + k^{(p)} \times x_n^{(1)}, \end{cases} \quad (1)$$

Де p – кількість змінних (розмірність) системи, $i = [1 \dots p]$, $k^{(i)} = (-1)^{i+1}$, $\left|x_n^{(i)}\right|$ позначення операції отримання модуля від значення $x_n^{(i)}$.

На одне вихідне значення необхідне виконання лише однієї операції множення та двох операцій додавання. Ще дві операції – знаходження модуля та множення на $k^{(i)} = \pm 1$, можуть бути реалізовані через ефективні побітові обчислення.

Для забезпечення рівномірного розподілу розв'язків системи (1), тобто щоб траєкторії системи з однаковою ймовірністю потрапляти у всі області у фазовому просторі, накладаються обмеження на максимальне значення змінних через застосування операцій зсуву [9]:

$$\begin{cases} x_{n+1}^{(j)} < -1, x_{n+1}^{(j)} = x_{n+1}^{(j)} + 2, \\ x_{n+1}^{(j)} > 1, x_{n+1}^{(j)} = x_{n+1}^{(j)} - 2. \end{cases} \quad (2)$$

Правила (2) обмежують значення реалізацій x_i в діапазоні [-1,1] та підвищують псевдовипадковість при застосуванні системи для генерування псевдовипадкових чисел.

Структура багатовимірної хаотичної системі Лозі має вигляд послідовності підсистем з'єднаних у кільце як наведено на рис. 1. Наступне значення кожної підсистеми визначається лише її поточним станом та станом приєднаної підсистеми згідно з кільцевим зв'язком.

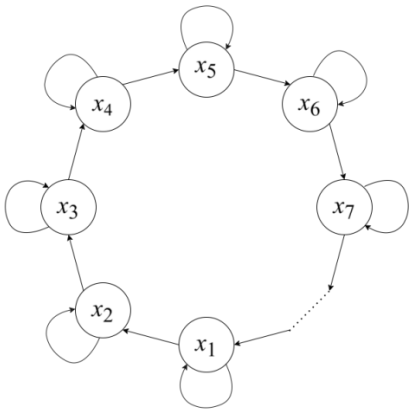

Рис. 1. Структура багатомірного відображення Лозі

Для побудови хеш-функції оберемо систему Лозі при $p = 8$. Для розширення можливостей керування системою введемо додаткові параметри [11]. Восьмивимірна система задається моделлю:

$$\begin{cases} x_1(n+1) = 1 - A_1|x_1(n)| + x_8(n), \\ x_2(n+1) = 1 - A_2|x_2(n)| - x_1(n), \\ \cdots, \\ x_8(n+1) = 1 - A_8|x_8(n)| - x_7(n), \end{cases} \quad (3)$$

де $x_1, \cdots, x_8$ – змінні системи, $A_1, \cdots, A_8$ – параметри. Перетворення (2) додатково застосовується до змінних системи після кожної ітерації.

Проведемо моделювання динаміки системи при однакових значеннях параметрів $A_1 = A_1 = \cdots = A_8 = A$. Приклад послідовності згенерованої системою (3) наведено на рис. 2.

Приклад розподілу хаотичної послідовності для 1000000 ітерацій при значенні параметрів $A_1 = A_2 = \cdots = A_8 = 2$ наведено на рис. 3. Множина значень всіх вихідних змінних $x_1, \cdots, x_8$ розбито на 100 непересічних під діапазонів, що не перетинаються, і знайдено кількість попадань у кожен. З рис. 3 випливає, що на кожен з інтервалів припадає приблизно 10000 точок, при цьому досліджені послідовності мають рівномірний розподіл.

Умовою використання хаотичної системи для криптографічних додатків є якомога ширший та неперервний діапазон значення параметрів керування у межах якого зберігається хаотична динаміка [2]. Метод біфуркаційних діаграм, що відображає залежності між динамічними режимами системи та її параметрами, використаний далі для вивчення динаміки відображення (3).

Діаграма біфуркацій, наведена на рис. 3, є суцільною майже у всьому досліджуваному діапазоні значення параметру. Це означає, що хаотичні часові ряди можна отримувати при зміні параметру A у широких межах в діапазоні $A = [0, 2]$. Це є ключовою перевагою багатовимірного відображення Лозі в порівнянні з відомими одновимірних хаотичними системами, такими як логістичне, квадратичне, кубічне рівняння. Варто звернути увагу, що у системі (3) також мають місце нехаотичні режими при значені параметру в околі $A \approx 0$. Таких значень параметрів слід уникати при застосуванні даної системи в криптографії, зокрема у пропонованому алгоритмі хешування. Суцільна біфуркаційна діаграма дає змогу уникнути вибору параметру при якому хаос відсутній, тобто можна уникнути слабких ключів.

Показники Ляпунова дають змогу оцінити характер коливань й інформаційні властивості системи. Якщо один чи більше показників Ляпунова нелінійної системи більше нуля, це гарантує, що в такій системі будуть хаотичні режими. Дискретна система хаотична, якщо є мінімум один показник Ляпунова більший нуля.

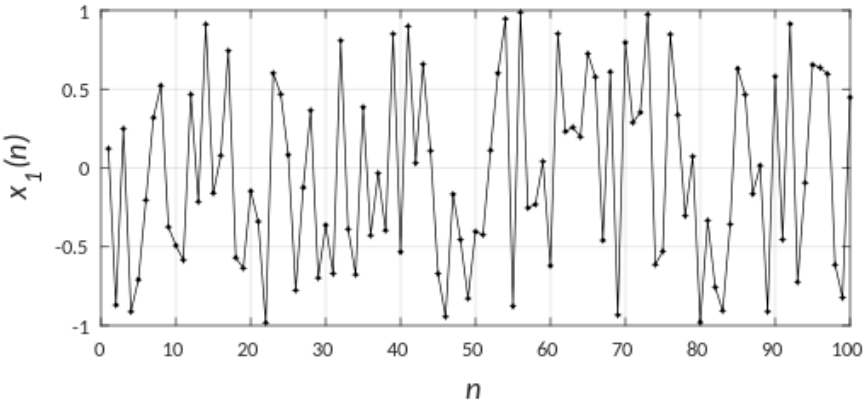

Рис. 2. Послідовності чисел восьмивимірної системи Лозі

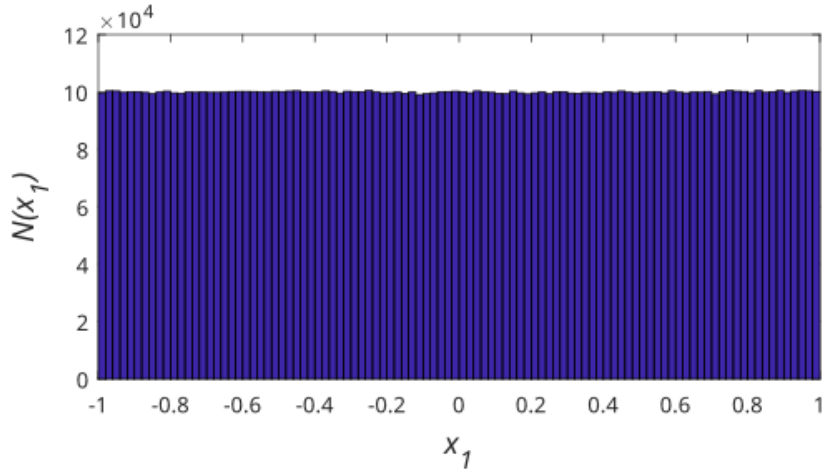

Рис. 3. Гістограма послідовності восьмивимірної системи Лозі

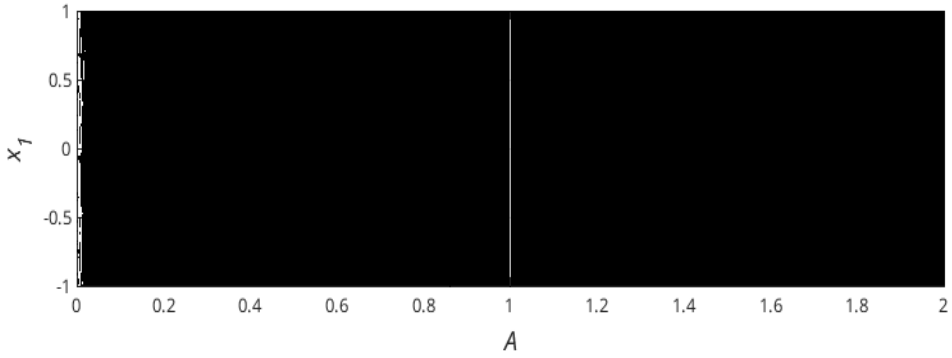

Рис. 4. Діаграма біфуркацій для восьмивимірної системи Лозі

Для досліджуваного відображення Лозі спектр показників Ляпунова $\lambda_1 \dots \lambda_4$ розрахований QR-методом наведено на рис. 5. При $A \in (1, 2]$ додатними є щонайменше 5 із 8 показників. Це підтверджує, що у системі Лозі спостерігаються складні гіперхаотичні режими.

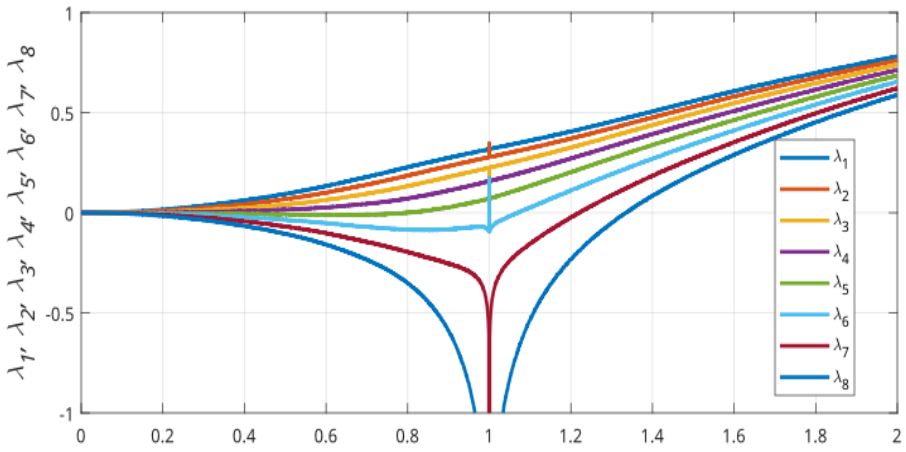

Рис. 5. Показники Ляпунова для восьмивимірної системи Лозі

МЕРЕЖА МЕРКЛА-ДАМГАРДА

Повідомлення M перед поділом на блоки спочатку адаптується шляхом додавання розширення для збільшення розміру, так щоб повідомлення для хешування за розміром стало кратним до блоку стиснення. У криптографії структура Меркла-Дамгарда є основною для створення криптографічно стійких до колізій хеш-функцій з використанням стійких односторонніх функцій стиснення. Такий принцип побудови хеш-алгоритму покладений в основі багатьох популярних хеш-додатків, наприклад, MD5, SHA1 і SHA2. Доведено, що коли односпрямована функція стиснення стійка до колізій, тоді хеш-функція побудована на її основі також буде стійкою до колізій [1].

Алгоритми хешування на основі структури Меркла-Дамгарда працюють згідно з рис. 6. Перший блок повідомлення M_1 подається на вхід функції стискання $F^{(*)}$ яка може бути проініціалізована секретним ключем K. У результаті перетворення функція стиснення отримується проміжне значення хешу H_1. Секретний ключ K може бути початковим значенням функції $F^{(*)}$, або ключем для проведення хешування. Далі наступний блок повідомлення M_2 та проміжний хеш H_1 є вхідними для функції стискання і т.д. Таким чином відбувається послідовна обробка всіх блоків повідомлення. Хешем повідомлення буде послідовність H_l на виході мережі.

Рис. 6. Мережа Меркла-Дамгарда

Відомо, що хаотичні системи є джерелами інформації. Звідси слідує, що інформація про початковий стан хаотичної системи з часом буде втрачатися або забуватися. З іншої сторони детермінізм однозначно пов’язує довільний початковий та відповідний йому кінцевий стани системи. Якщо початковими умовами хаотичної системи є певне повідомлення, то після деякої кількості ітерацій кінцевий стан системи буде однозначно пов'язаний з початковим. Проте зв'язок між вхідним станом та вихідним буде практично неможливо встановити. Це пов'язано з тим, що обернений розв'язок для хаотичних систем є практично неможливим. Виходячи з цих міркувань можна стверджувати, що хаотичні системи можуть слугувати функціями одностороннього перетворення або стиснення свого початкового стану.

РОЗРОБЛЕННЯ АЛГОРИТМУ ХЕШУВАННЯ НА ОСНОВІ СИСТЕМИ ЛОЗІ

Врахувавши результати аналізу хаотичної системи, нами запропоновано алгоритм хешування з функцією стиснення на базі восьмивимірного відображення Лозі (3). Вхідними даними є ключ хешування, що використовується для задання початкових умов алгоритму. Послідовність кроків алгоритму наступна:

1. Задаємо ключ хешування розміром 256 біт.
2. Ключ розділяється на вісім підключів по 32 біти, з яких формуються двійкові послідовності $x_{1,0}, \ldots, x_{8,0}$, які використовуються як початкові умови для ітерацій відображення (3). Початкові умови

утворюють перетворенням двійкової послідовності у знакове раціональне число з фіксованою комою, при цьому на знак виділяється 1 біт, на цілу частину – 1 біт, на дробову частину – 30 біт.

3. Початкові значення параметрів системи на початку роботи алгоритму встановлюються як $A_1 = A_2 = \cdots = A_8 = 2$.

4. Розбиваємо вхідне повідомлення на k блоків M_i довжиною 128 біт. При необхідності останній блок доповнюється послідовністю нулів.

5. Кожен блок повідомлення M_i розділяється на вісім частин $M_{i,1}, \dots, M_{i,8}$ довжиною 16 біт кожна. Кожна частина блоку повідомлення M_i обробляється окремо з початковими умовами згідно п. 2.

6. Оновлені значення параметрів $A_1, \dots, A_8$ для кожної ітерації системи (3) утворюють як:

6.1. Старші 2 біти параметрів її залишаються незмінними;

6.2. Біти 14 … 29 значень параметрів $A_1, \dots, A_8$ утворюються додаванням за модулем 2 блоків інформації $M_{i,1}, \dots, M_{i,8}$, $A_1, \dots, A_8$ та 14 … 29 бітів двійкового представлення попереднього значення відповідних параметрів.

6.3. Молодші 14 біт значення параметрів є результатом додавання за модулем 2 молодших 14 біт відповідного попереднього (початкового) стану та молодших 14 біт попереднього значення параметру (рис. 7).

7. Використовуючи значення параметрів $A_1, \dots, A_8$ та попереднього стану $x_{1,n}, \dots, x_{8,n}, n = 1,2, \dots, k$ знаходимо згідно (3) наступний стан системи.

8. Результат обробки частин блоку повідомлення M_i та наступний блок M_{i+1} додаються за модулем 2.

9. Кроки 5 – 8 повторюємо для всіх блоків повідомлення.

10. Для приховання кінцевого стану хаотичної системи після введення останнього блоку інформації виконуємо ітерації (3) N раз із початковими умовами та значеннями параметрів отриманими на кроці 7. Молодші 8 біт кожної змінної стану системи зберігаємо як частину вихідного хешу повідомлення (8*8 = 64 біти).

11. Виконуємо N ітерацій системи (3) із початковими умовами отриманими на кроці 10 та параметрами $A_1 = A_2 = \cdots = A_8 = 2$. Молодші 24 біти значень змінних кінцевого стану хаотичної системи зберігаємо як частину вихідного хешу повідомлення (8*24 = 192 біт).

12. Хеш повідомлення становить 256 біт і отримується об'єднанням послідовностей отриманих у п. 10 і п. 11 (64+192 = 256 біт).

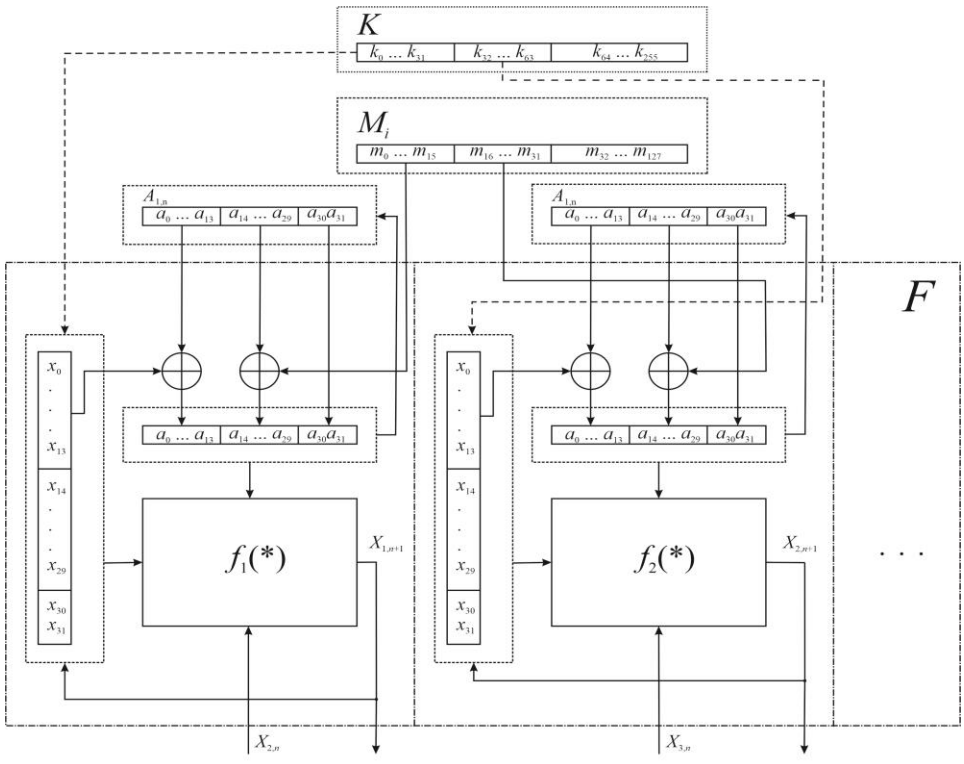

Рис. 7. Схема введення блоку інформації у функцію перетворення

Відмінність пропонованого алгоритму від відомих [4-7] є наступною. У даному алгоритмі параметри, які визначають хаотичність системи отримуються в результаті сумісної дії її блоку інформації та поточного стану системи. Таким чином на кожному кроці здійснюється нелінійне введення та перетворення інформації. При виконанні кожної ітерації кожний блок інформації зазнає впливу як змінних стану системи так і значення параметра. Процес виконання алгоритму обчислень однієї ітерації рівнянь (3) зображено на рис. 8.

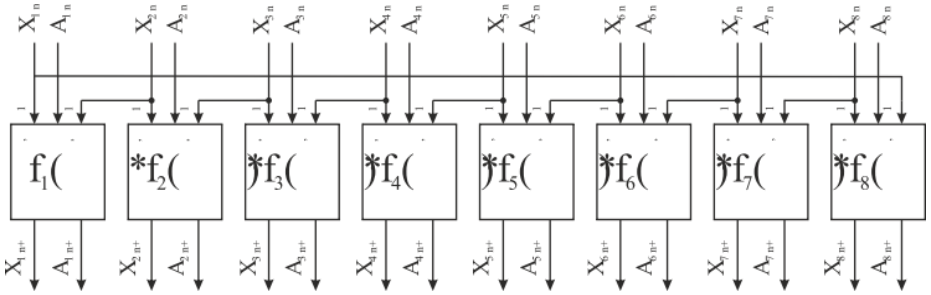

Рис. 8. Виконання однієї ітерації хаотичної системи при хешуванні

Розроблений алгоритм орієнтований на застосування арифметики з фіксованою комою з довжиною представлення числа в 32 біт, з яких один виділяється на представлення знаку числа, один для опису цілої частина числа, 30 біт – для дробової. Такий поділ забезпечує ефективне використання 32-го представлення числа. Використання обчислень з фіксованою комою, замість рухомої коми, полегшує реалізацію і виконання алгоритму на різних платформах.

Алгоритм також може бути адаптований для використання іншої арифметики або більшої довжини слова при відповідному доопрацюванні.

АНАЛІЗ СТІЙКОСТІ ХЕШУВАННЯ

Для проведення дослідження обрано коротке повідомлення “Сьогодні на дворі чудова погода” яке було збережено у двійковому файлі. Ключ хешування обрано наступним: 848DDA233F0FD7DD0411AA034E9800410F125D5CC7C3439E3A5D463060036B0. На основі заданого ключа сформовано початкові умови відповідно до описаного алгоритму.

Алгоритм хешування повинен бути чутливим до найменшої зміни повідомлення, тобто має забезпечувати лавинний ефект. Для заданого повідомлення отримано наступний хеш hash = D43DD7FC2E1C342DBB27F7093CD89ECDE69E95E71E3B12FEF21B0F61677F5D4E

Для зміненого повідомлення “Сьогодні на дворі чудова погода!” – в кінці додано символ “!” при незмінному ключі хеш становить hash = C1A602B8E68374CFC72F8644C8C019C439F45D3A3BA6F791 382123737D458352.

Для повідомлення “Сьогодні на дворі чудова погода” – додано символ пробілу між словами “на” та “дворі” отриманий хеш рівний hash = A17B199BF8CE5F9884382A5C16EDDB9EAB090CB32732C56D657E439583F1D37A.

Хеш значення повинно виглядати як випадкове число. Для перевірки цього правила було проведено статистичне дослідження у якому підраховано кількість символів “0” та “1” для великої кількості хеш-значень. Якщо хеш є випадковим за значенням тоді для двох значень хешу отриманих для різних повідомлень, або для одного і того самого повідомлення, але з різним ключем, кількість біт які не збігаються повинно становити в середньому 128 (для хешу довжиною 256 біт). Розподіл кількості не співпадаючих біт повинен мати вигляд гаусової кривої.

Для визначення чутливості алгоритму хешування до повідомлення було згенеровано пари випадкових повідомлень довжиною 64 байти, які відрізнялися значенням одного біту. Всі пари повідомлень піддавалися хешуванню за допомогою одного і того ж ключа. Розподіл кількості відмінних біт у парах хеш-значень наведено на рис. 9*а*. Можна констатувати, що цей розподіл нагадує гаусовий, тобто очікуваний теоретичний для подібної задачі.

Середнє значення кількості неоднакових бітів у хешах становить 128,05, що дає право стверджувати про достатню чутливість запропонованого алгоритму до вхідного повідомлення.

Далі було проведено дослідження чутливості хешування до ключа. Одне і теж повідомлення хешувалося ключами, що відрізнялися значенням останнього біту. Отриманий розподіл також є гаусовим, а середня кількість різних за значенням біт становить 128,03 (рис. 9*б*).

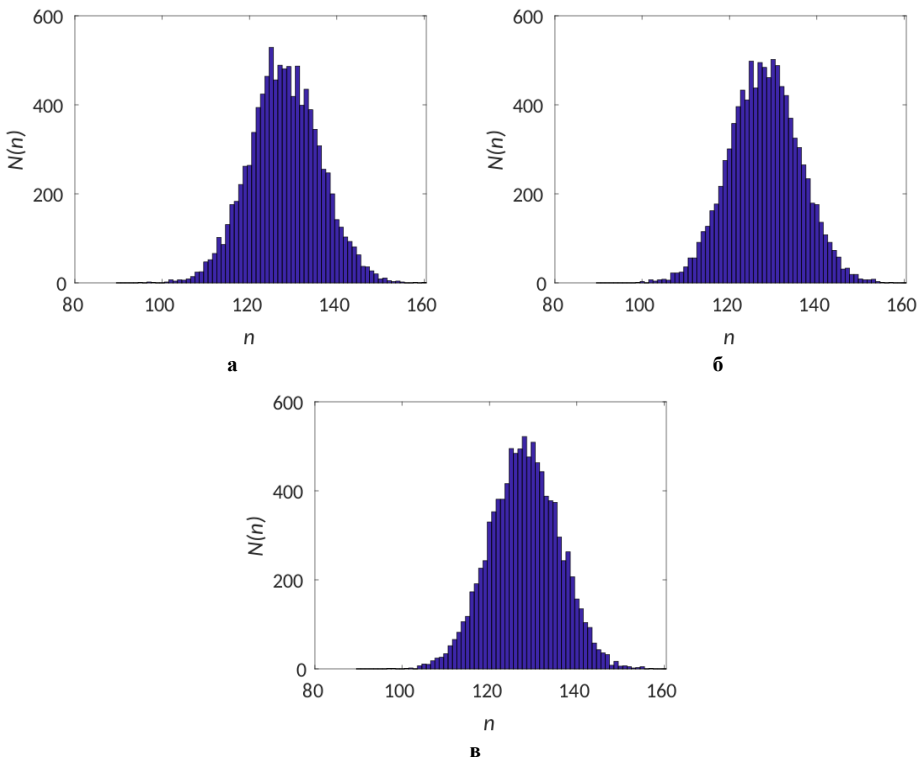

Рис. 9. Гістограма кількості неоднакових бітів при хешуванні: фіксованим ключем пар повідомлень, що відрізняються значенням останнього біту – а; випадкових повідомлень ключами, що відрізняються значенням останнього біту – б; пар різних випадкових повідомлень з випадковими ключами – в

Третій експеримент полягав у визначенні кількості побітових відмінностей для хеш-значень отриманих для різних випадкових повідомлень та різних ключів. Гістограма таких відмінностей наведена на рис. 9*в*. У даному випадку також отримано гаусовий розподіл, із середнім значенням 128,01.

Як слідує з рис. 9 у результатах всіх трьох експериментів колізій не виявлено. Колізія в даному випадку означає, що два хеші є однаковими, тобто кількість побітових відмінностей дорівнює нулю.

Якщо до пропонованого алгоритму застосувати атаку днів народження, то у випадку рівномірного розподілу хеш-значень колізії можуть виникати в середньому при 2^{128} спроб підбору повідомлення.

Підсумовуючи можна зробити висновок, що описаний алгоритм хешування є достатньо чутливим до повідомлення та ключа.

СТАТИСТИЧНЕ ТЕСТУВАННЯ ХЕШ-ПОСЛІДОВНОСТЕЙ

Хеш-послідовності повинні відповідати всім вимогам, що висуваються до випадкових чисел. Одним із основних інструментів, який використовується для дослідження псевдовипадкових чисел на відповідність до істинно випадкових є статистичні тести NIST SP 800-22 [12]. Пакет NIST STS 2.1.2 містить 15 тестів, за допомогою яких намагаються виявити різні закономірності у псевдовипадкових послідовностях. Якщо тестові послідовності успішно проходять всі 15 тестів, то такі послідовності можна вважати статистично безпечними.

Реалізацію та тестування запропонованого алгоритму було здійснено у середовищі MatLab. Для дослідження згенеровано хеш-послідовності сумарною довжиною $2*10^8$ біт та проведено їх тестування. Результати тестування можна побачити у табл. 1, а отриманий статистичний портрет – на рис. 10. Мінімальне прохідне значення для кожного статистичного тесту (для 1000 протестованих послідовностей), за винятком тестів random excursion (variant) приблизно = 0.965. Мінімальне значення для проходження тестів random excursion (variant) становить приблизно 0.960.

З результатів, наведених в табл. 1, випливає, що послідовності хеш-значень успішно пройшли всі тести.

З результатів дослідження (табл. 1) можна зробити висновок, що запропонований генератор хеш-послідовностей на основі багатовимірної хаотичної системи Лозі задовольняє вимогам криптографічних тестів та має задовільні статистичні характеристики, а генеровані бітові послідовності задовольняють вимогам статистичних тестів NIST, тобто їх можна вважати статистично безпечними.

Таблиця 1.

Результати тестування хеш-послідовностей тестами NIST

Назва тесту	P - value	Пропорція	Результат
Frequency	0,176	0,97	Пройдено
BlockFrequency	0,263	0,968	Пройдено
CumulativeSums	0,948	0,976	Пройдено
CumulativeSums	0,670	0,989	Пройдено
Runs	0,349	0,991	Пройдено
LongestRun	0,807	0,983	Пройдено
Rank	0,968	0,991	Пройдено
FFT	0,807	0,997	Пройдено
OverlappingTemplate	0,855	0,972	Пройдено
Universal	0,489	0,995	Пройдено
ApproximateEntropy	0,916	0,995	Пройдено
Serial	0,540	0,975	Пройдено
Serial	0,430	0,993	Пройдено
LinearComplexity	0,824	0,99	Пройдено
NonOverlappingTemplate, 148 тестів	-	-	Пройдено
RandomExcursions, 8 тестів	-	-	Пройдено
RandomExcursionsVariant, 18 тестів	-	-	Пройдено

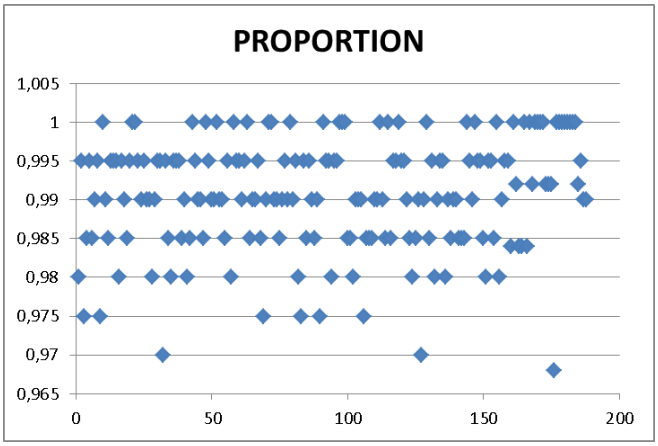

Рис. 10. Статистичний портрет хеш-послідовностей

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

В статті описано алгоритм генерування хеш-послідовностей на основі багатовимірної хаотичної системи Лозі. Відмінність запропонованого генератора хеш-функцій від інших подібного типу полягає в сумісному впливі блоку інформації та поточного стану системи на вихідне хеш-значення. Повідомлення одночасно збурює стан хаотичної системи та її параметри, що призводить до складних і незворотних нелінійних перетворень. Розроблений алгоритм генерування хеш-функцій орієнтований на застосування арифметики з фіксованою комою з довжиною чисел в 32 біти. Використання арифметики з фіксованою комою, замість рухомої коми, забезпечує можливість реалізації алгоритму на різні платформи з обмеженими обчислювальними ресурсами. Алгоритм також може бути адаптований для використання іншої арифметики або більшої довжини слова при відповідному доопрацюванні.

Дослідження запропонованого алгоритму з використанням пакету статистичних тестів NIST STS показали, що запропонований генератор хеш-послідовностей на основі багатовимірної хаотичної системи Лозі генерує хеш-послідовності, які можуть вважатися статистично безпечними.

Запропонований генератор хеш-послідовностей на основі багатовимірної хаотичної системи Лозі має задовільні статистичні характеристики, а генеровані бітові послідовності задовольняють вимогам статистичних тестів NIST, тобто їх можна вважати статистично безпечними.

Подяки

Дослідження підтримано грантом № 2023.04/0150 від Національного фонду досліджень України.

References

1. Chaos-Based cryptography / ed. by L. Kocarev, S. Lian. – Berlin, Heidelberg : Springer Berlin Heidelberg, 2011.

2. Alvarez G. Some basic cryptographic requirements for chaos-based cryptosystems / Gonzalo Alvarez, Shujun Li // International journal of bifurcation and chaos. – 2006. – Vol. 16, no. 08. – P. 2129–2151.

3. A Deterministic Chaos-Model-Based Gaussian Noise Generator / Serhii Haliuk [et al.] // Electronics. – 2024. – Vol. 13, no. 7. – P. 1387.

4. Amin M. Chaos-based hash function (CBHF) for cryptographic applications / Mohamed Amin, Osama S. Faragallah, Ahmed A. Abd El-Latif // Chaos, solitons & fractals. – 2009. – Vol. 42, no. 2. – P. 767–772.

5. One-Way hash function based on cascade chaos / Fei Xiang [et al.] // The open cybernetics & systemics journal. – 2015. – Vol. 9, no. 1. – P. 573–580.

6. A New Hash Function Based on Chaotic Maps and Deterministic Finite State Automata / Moatsum Alawida [et al.] // IEEE Access. – 2020. – Vol. 8. – P. 113163–113174.

7. Ayubi P. Chaotic Complex Hashing: A simple chaotic keyed hash function based on complex quadratic map / Peyman Ayubi, Saeed Setayeshi, Amir Masoud Rahmani // Chaos, Solitons & Fractals. – 2023. – Vol. 173. – P. 113647.

8. Krulikovskyi O.V. Osoblyvosti vyboru khaotychnykh system dlia pobudovy heneratoriv psevdovypadkovykh poslidovnostei // O. V. Krulikovskyi, S.D. Haliuk, L.F. Politanskyi // Telecommunication and Informative Technologies. – 2017. – №2(55). – S. 31-40.

9. Lozi R. Noise-resisting ciphering based on a chaotic multi-stream pseudo-random number generator / René Lozi, Estelle Cherrier // 2011 International conference for internet technology and secured transactions, Abu Dhabi, 2011. – P. 91–96.

10. Garasym O. High-speed encryption method based on switched chaotic model with changeable parameters [Electronic resource] / Oleg Garasym, Ina Taralova // 2013 8th International Conference for Internet Technology and Secured Transactions (ICITST), London, United Kingdom, 9–12 December 2013. Pp. 37-42.

11. Krulikovskyi O.V. Testing timeseries ring-coupled map generated by on FPGA / O.V. Krulikovskyi, S.D. Haliuk, L.F. Politanskyi // Telecommunication and Informative Technologies. – 2016. – №4(53). – C. 24-29.

12. On the interpretation of results from the NIST statistical test suite / M. Sys [et al.] // Romanian journal of infor mation science and technology. – 2015. – Vol. 18, № 1. – P. 18–32.