Харківський національний університет імені В. Н. Каразіна
Міністерство освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

Родінко Марія Юріївна

УДК 681.3.06:006.354

ДИСЕРТАЦІЯ
«МЕТОДИ ПОБУДОВИ ТА ДОСЛІДЖЕННЯ ВЛАСТИВОСТЕЙ МАЛОРЕСУРСНИХ БЛОКОВИХ ШИФРІВ ТА ЇХ КОМПОНЕНТІВ»

Спеціальність 122 – Комп'ютерні науки
(Галузь знань 12 – Інформаційні технології)
Подається на здобуття ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.


М. Ю. Родінко

Науковий керівник: Олійников Роман Васильович, доктор технічних наук, доцент.

Усі принципові положення дисертації ідентичні тим, що захищалися
Голова спеціалізованої вченої ради Д 64.051.019 Валентин ПАЗУРА

Харків – 2020

АНОТАЦІЯ

Родінко М. Ю. Методи побудови та дослідження властивостей малоресурсних блокових шифрів та їх компонентів. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття ступеня доктора філософії за спеціальністю 122 – Комп'ютерні науки (Галузь знань 12 – Інформаційні технології). – Харківський національний університет імені В. Н. Каразіна Міністерства освіти і науки України, Харків, 2020.

Дисертація присвячена розробці та удосконаленню методів аналізу криптографічних властивостей компонентів симетричних блокових шифрів та побудові перспективних криптографічних перетворень.

Метою дисертаційної роботи є підвищення продуктивності симетричних криптографічних перетворень і удосконалення методів аналізу їх стійкості.

У першому розділі дисертації *(Актуальний стан та перспективи розвитку методів аналізу та синтезу симетричних блокових шифрів)* виконано аналіз актуального стану розвитку технологій симетричного блокового шифрування та постановку задач дослідження. Зокрема, проаналізовані основні високорівневі конструкції, а також принципи побудови циклової функції та схеми розгортання ключів симетричних блокових шифрів. Виконано аналіз сучасних малоресурсних блокових шифрів та вимог до симетричних примітивів, що застосовуються у пристроях з обмеженою кількістю споживання енергії. За результатами аналізу виявлені недоліки та невирішені питання, що стосуються існуючих малоресурсних блокових шифрів, виходячи з яких сформульовані задачі дисертаційного дослідження.

У другому розділі *(Розробка удосконаленого методу генерації оптимальних S-блоків з високими нелінійністю та алгебраїчним імунітетом)* вирішена задача щодо підвищення швидкості (зниження обчислювальної складності) генерації оптимальних S-блоків на доступній обчислювальній

техніці. Отримано *перший науковий результат*: *удосконалено* метод градієнтного спуску для генерації нелінійних таблиць заміни, що відрізняється від відомого обґрунтованим оптимальним порядком застосування критеріїв при перевірці підстановок на відповідність криптографічним показникам, що дозволяє суттєво знизити складність генерації оптимальних S-блоків.

Отримано *перший практичний результат*: удосконалений метод градієнтного спуску дозволяє зменшити середній час генерації оптимального байтового S-блока з нелінійністю 104, δ-рівномірністю 8, алгебраїчним імунітетом 3 та мінімальною степінню булевих функцій 7 на персональному комп'ютері з 2,5 годин до 30 хвилин (скорочення часу розрахунків у 5 разів).

В основу оптимізації методу градієнтного спуску покладено той факт, що критерії відбору підстановок є частково взаємозалежними. Таким чином, час перевірки підстановки на відповідність набору критеріїв суттєво залежить від порядку застосування цих критеріїв у процесі перевірки. У зв'язку з цим, запропоновано аналітичний вираз для визначення порядку застосування критеріїв, за якого час перевірки підстановки буде мінімальним.

В процесі перевірки запропонованого підходу для чотирьох обраних критеріїв були розраховані значення часу перевірки підстановки (та, відповідно, часу генерації) для всіх можливих комбінацій критеріїв. Розрахунки показали, що вибір найоптимальнішого порядку застосування критеріїв дозволяє зменшити час генерації S-блока майже в 5 разів. Аналітично отримані значення були підтверджені й експериментально. Середній час генерації оптимальних підстановок за допомогою програмної реалізації з урахуванням запропонованого вдосконалення склав 30 хвилин.

У третьому розділі *(Розробка методу оцінки колізійних властивостей неін'єктивних схем розгортання ключів симетричних блокових шифрів)* вирішена задача удосконалення оцінок щодо ймовірності співпадіння потужностей множини послідовностей циклових ключів, які формуються неін'єктивною схемою розгортання циклових ключів, і множини ключів шифрування для симетричних шифрів. Отримано *другий науковий результат*:

отримав подальший розвиток математичний метод оцінки колізійних властивостей неін’єктивних схем розгортання ключів блокових шифрів, що відрізняється застосуванням вдосконаленої математичної моделі та більш ефективного математичного апарату та дозволяє отримати уточнену оцінку ймовірності колізії двох послідовностей циклових ключів.

Отримано *другий практичний результат*: удосконалено значення оцінки нижньої границі стійкості діючого національного стандарту ДСТУ 7624:2014, а саме доведено, що складність атак переборного типу на неін’єктивні схеми розгортання ключів практично дорівнює складності атак на ін’єктивні схеми.

За допомогою удосконаленого математичного методу оцінки ймовірності співпадіння потужностей множини послідовностей циклових ключів, які формуються неін’єктивною схемою розгортання ключів, і множини ключів шифрування, доведено, що для повномасштабного шифру ця ймовірність практично дорівнює 1. При цьому неін’єктивні схеми розгортання ключів забезпечують додаткову стійкість до атак на реалізацію та деяких інших методів криптоаналізу. Таким чином, при побудуванні перспективного симетричного блокового шифру доцільно використовувати саме неін’єктивну схему розгортання циклових ключів.

У четвертому розділі *(Побудова та аналіз властивостей перспективного постквантового малоресурсного блокового шифру)* вирішені наступні задачі:

– розробка малоресурсного блокового симетричного шифру, що задовольняє вимогам щодо забезпечення високого та надвисокого рівня стійкості (за класифікацією NESSIE), а також має компактну реалізацію та високу продуктивність на різних програмно-апаратних платформах;

– розробка математичної моделі для оцінки стійкості визначеного класу ARX-подібних шифрів до диференційного криптоаналізу, що базується на відомих положеннях теорії криптоаналізу та обґрунтованих припущеннях щодо властивостей компонентів таких шифрів;

– розробка методів пошуку найбільш ймовірних одноциклових диференційних характеристик визначеного класу ARX-шифрів з урахуванням

особливостей побудови циклової функції, що використовується у шифрах визначеного класу;

– удосконалення методів пошуку багатоциклових диференційних характеристик визначеного класу ARX-шифрів із використанням відомих підходів та результатів застосування розроблених методів пошуку одноциклових диференційних характеристик.

Отримано *третій та четвертий наукові результати*:

– *вперше* запропоновано два методи пошуку одноциклових диференційних характеристик для визначеного класу ARX-шифрів, що дозволяють з низькою обчислювальною складністю отримувати оцінки стійкості циклової функції блокового шифру визначеного класу до диференційного криптоаналізу;

– *отримали подальший розвиток* методи пошуку багатоциклових диференційних характеристик для визначеного класу ARX-шифрів, що відрізняються вдосконаленим механізмом відбору вхідних різниць та формуванням початкової множини одноциклових диференційних характеристик, що дозволяє отримати оцінку щодо стійкості повномасштабного блокового шифру визначеного класу до диференційного криптоаналізу.

Отримано *третій та четвертий практичні результати*:

– розроблено перспективний постквантовий малоресурсний блоковий шифр «Кипарис», що забезпечує високий та надвисокий рівні стійкості та перевершує за швидкодією відомі малоресурсні блокові шифри на 32- та 64-бітових процесорах загального призначення та мобільних платформах;

– обґрунтовано стійкість симетричного блокового шифру «Кипарис-256» до диференційного криптоаналізу згідно з вимогами практичного критерію, а саме показано, що максимальна середня за ключами ймовірність диференційної характеристики є набагато меншою за ймовірність успіху атаки прямого перебирання ключів.

Запропонований блоковий шифр «Кипарис» заснований на мережі Фейстеля, а циклова функція шифру представляє собою ARX-перетворення.

Схема розгортання циклових ключів шифру є неін’єктивною та використовує принципи побудови схеми розгортання ключів шифру «Калина». Алгоритм підтримує довжину блока (ключа) 256 та 512 бітів, що дозволяє забезпечити високий рівень криптографічної стійкості.

Розроблена математична модель оцінки стійкості визначеного класу ARX-шифрів до диференційного криптоаналізу дозволяє отримати вираз для обчислення середньої (за ключами) ймовірності диференційної характеристики шифру та зробити обґрунтоване припущення щодо значень вхідних різниць, які при проходженні через цикл шифрування формують характеристику, що має високу ймовірність. Модель включає припущення про те, що шифр є марковським, про ймовірність перетворення диференційних різниць на модульних суматорах, про зв’язок кількості активних біт у вхідній різниці з ймовірністю диференційної характеристики, а також містить вирази для обчислення ймовірностей одноциклових та багатоциклових диференційних характеристик. На основі представленої моделі розроблено методи пошуку диференційних характеристик класу блокових ARX-шифрів.

Розроблено методи пошуку диференційних характеристик циклової функції визначеного класу ARX-шифрів (прямий метод пошуку, метод пошуку «у двох напрямках» та оптимізований метод пошуку диференційної характеристики з високою ймовірністю), що дозволяють знайти найбільш ймовірні одноциклові диференційні характеристики. Метою всіх трьох підходів є активізація найменшої кількості біт на входах суматорів циклової функції, що, в свою чергу, збільшує ймовірність заданого перетворення. Останній із запропонованих методів дозволив здійснити ефективний пошук диференційної характеристики на один цикл перетворення блокового шифру «Кипарис» з ймовірністю, що дорівнює 1/4.

Застосування запропонованого методу пошуку багатоциклових диференційних характеристик, заснованого на побудові множини високоймовірнісних одноциклових диференційних характеристик, до блокового шифру «Кипарис-256» показало, що побудовані одноциклові диференційні

характеристики, вихідні різниці яких мають малу вагу Гемінга (а значить і достатньо високу ймовірність), не можуть бути продовжені для побудування багатоциклових диференційних характеристик з високою ймовірністю.

Одна зі знайдених найбільш ймовірних багатоциклових диференційних характеристик для блокового шифру «Кипарис-256» може бути побудована лише для шести циклів шифрування з ймовірністю $\mathrm{MEDP}^{(6)}(\Omega) \approx 2^{-223}$. Таким чином, блоковий шифр «Кипарис-256» є стійким до диференційного криптоаналізу з точки зору практичного критерію.

У п’ятому розділі *(Експериментальні дослідження властивостей симетричного блокового шифру «Кипарис»)* досліджені статистичні, лавинні та швидкісні характеристики розробленого перспективного блокового шифру «Кипарис». Дослідження статистичних властивостей шифру показало, що шифруюче перетворення та схема розгортання ключів алгоритмів «Кипарис-256» та «Кипарис-512» задовольняють вимогам зі статистичного тестування випадкових послідовностей NIST STS.

Дослідження лавинних показників блокового шифру «Кипарис» показало, що «Кипарис-256» (число циклів дорівнює 10) та «Кипарис-512» (число циклів дорівнює 14) відповідають вимогам щодо лавинного ефекту починаючи з чотирьох циклів шифрування.

Порівняння швидкодії блокового шифру «Кипарис» зі швидкодією відомих малоресурсних алгоритмів здійснювалося на платформах Windows, Linux та Android (із залученням одного ядра процесора в однопотоковому застосуванні). Блоковий шифр «Кипарис» продемонстрував високу продуктивність на всіх досліджуваних програмно-апаратних платформах. На платформі Windows 10 з 32-бітовою архітектурою найкращий результат показав шифр «Кипарис-256» (порядку 3,5 Гбіт/сек). На платформі Windows 10 з 64-бітовою архітектурою найкращий результат показав шифр «Кипарис-512» (приблизно 5 Гбіт/сек). На платформі Linux з 64-бітовою архітектурою блоковий шифр «Кипарис-256» показав надвисокий результат зі швидкодії (понад 8 Гбіт/сек).

Ключові слова: симетричний блоковий шифр, малоресурсний блоковий шифр, S-блок, нелінійність, алгебраїчний імунітет, циклова функція, схема розгортання ключів, ARX-шифр, диференційний криптоаналіз, диференційна характеристика.

ABSTRACT

Rodinko M. Yu. Methods of construction and research of properties of lightweight block ciphers and their components. – Qualification scholarly paper: a manuscript.

Thesis submitted for obtaining the Doctor of Philosophy degree in Information Technologies, Speciality 122 – Computer Science. – V. N. Karazin Kharkiv National University, Ministry of Education and Science of Ukraine, Kharkiv, 2020.

The dissertation is devoted to the development and improvement of methods on cryptographic properties analysis of block ciphers components and construction of perspective cryptographic transformations.

The aim of the dissertation is to increase performance of symmetric cryptographic transformations and improve methods of analysis of their strength.

The first chapter of the dissertation *(Actual state and development perspectives of analysis and synthesis methods of symmetric block ciphers)* contains the analysis of the actual state of development of block encryption technologies and research problems description. In particular, the basic high-level constructions, as well as the principles of construction of the round function and key schedules of block ciphers are analyzed. The analysis of modern lightweight block ciphers and requirements to symmetric primitives used in devices with constrained energy consumption is presented. The results of the analysis revealed disadvantages and unresolved issues related to the existing lightweight block ciphers, based on which the tasks of the research are formulated.

In the second chapter *(Development of the improved method of optimal S-boxes generation with high nonlinearity and algebraic immunity)*, the problem of increasing the speed (reducing computational complexity) of generating optimal S-boxes using available computational resources is solved. The first scientific result is obtained: it is improved the gradient descent method for generating nonlinear substitution tables, which differs from the known one by a reasonable optimal order of criteria

application when checking substitutions against cryptographic indicators, which significantly reduces the complexity of optimal S-boxes generation.

The first practical result is obtained: the improved gradient descent method allows to reduce the average generation time of the optimal byte S-box with nonlinearity 104, δ-uniformity 8, algebraic immunity 3 and the minimum degree of Boolean functions 7 on a PC from 2.5 hours to 30 minutes (reduction of calculation time by 5 times).

The optimization of the gradient descent method is based on the fact that the criteria for selecting substitutions are partially interdependent. Thus, the time of verification of the substitution for compliance with a set of criteria significantly depends on the order of application of these criteria during the verification process. In this regard, an analytical expression is proposed to determine the order of application of the criteria, in which the time of checking the substitution will be minimal.

During the testing of the proposed approach, for the four selected criteria, the values of the substitution check time (and, accordingly, the generation time) were calculated for all possible combinations of criteria. Calculations have shown that the choice of the optimal order of application of the criteria allows to reduce the generation time of the S-box by almost 5 times. Analytically obtained values were also confirmed experimentally. The average time to generate optimal substitutions using software implementation taking into account the proposed improvement was 30 minutes.

The third chapter *(Development of the method of collision properties evaluation of non-injective key schedules of symmetric block ciphers)* solves the problem of improving the estimates of the probability of coincidence of the cardinalities of the set of sequences of round keys, which are formed by a non-injective key schedule, and the set of encryption keys for symmetric ciphers. The second scientific result is obtained: it is further developed the mathematical method of estimating the collision properties of non-injective key schedules of block ciphers, which differs in the application of an improved mathematical model and more

efficient mathematical apparatus and allows to obtain an accurate estimate of the collision probability of two round key sequences.

The second practical result is obtained: the estimation of the lower bound of strength of the current national standard DSTU 7624:2014 is improved, namely it is proved that the complexity of exhaustive search attacks on non-injective key schedules is almost equal to the complexity of attacks on injective ones.

Using an advanced mathematical method for estimating the probability of matching the cardinalities of a set of round key sequences generated by a non-injective scheme and a set of encryption keys, it is proved that for a full-scale cipher this probability practically equals to 1. Moreover, non-injective key schedules provide additional resistance to side channel attacks and some other methods of cryptanalysis. Thus, when constructing a perspective block cipher, it is advisable to use a non-injective key schedule.

The fourth chapter *(Construction and analysis of properties of the perspective post quantim lightweight block cipher)* solves the following problems:

– development of the lightweight block cipher that meets the requirements for ensuring a high and ultra-high security level (according to the NESSIE classification), as well as has a compact implementation and high performance on various software and hardware platforms;

– development of a mathematical model for estimating a strength of a certain class of ARX-like ciphers to differential cryptanalysis, based on the known statements of the theory of cryptanalysis and reasonable assumptions about the properties of the components of such ciphers;

– development of methods for finding the most probable one-round differential characteristics of a certain class of ARX-ciphers, taking into account the specifics of the round function construction used in ciphers of this class;

– improvement of methods of searching for multi-round differential characteristics of a certain class of ARX-ciphers using the known approaches and results of application of the developed methods of searching for one-round differential characteristics.

The third and fourth scientific results were obtained:

– for the first time two methods of searching for one-round differential characteristics for a certain class of ARX-ciphers are proposed, which allow to obtain estimates of the strength of the round function of a block cipher of the certain class to differential cryptanalysis with low computational complexity;

– methods of searching for multi-round differential characteristics for a certain class of ARX-ciphers have been further developed, which differ by an improved mechanism for selecting input differences and forming an initial set of one-round differential characteristics, which allows to estimate the strength of a full-scale block cipher of the certain class to differential cryptanalysis.

The third and fourth practical results are obtained:

– a perspective post-quantum lightweight block cipher Cypress is developed, which provides high and ultra-high levels of security and exceeds the performance of known lightweight block ciphers on 32- and 64-bit general-purpose processors and mobile platforms;

– the strength of the block cipher Cypress-256 to the differential cryptanalysis in accordance with the requirements of the practical criterion is proven, namely it is shown that the maximum average probability of differential characteristics is much less than the probability of success of the exhaustive search attack.

The proposed block cipher Cypress is based on the Feistel network, and the round function of the cipher is an ARX-transformation. The cipher key schedule is non-injective and uses the principles of construction of Kalyna key schedule. The algorithm supports the block (key) length of 256 and 512 bits, which allows to provide a high level of cryptographic security.

The developed mathematical model for estimating the strength of a certain class of ARX-ciphers to differential cryptanalysis allows to obtain an expression for calculating the average (by keys) probability of the cipher differential characteristic and make a reasonable assumption about the values of input differences which form a high-probable characteristic while propagating through the encryption round. The model includes assumptions that the cipher is a Markov one, on the probability of

transformation of differences on modular adders, on the correlation of the number of active bits in the input difference with the probability of differential characteristics, and contains expressions to calculate the probabilities of one-round and multi-round differential characteristics. On the basis of the presented model the methods of search of differential characteristics of a class of block ARX-ciphers are developed.

The methods of searching for differential characteristics of the round function of a certain class of ARX-ciphers are developed (direct search method, two-way search method and optimized method of searching for a differential characteristic with high probability) to find the most probable one-round differential characteristics. The purpose of all three approaches is to activate the smallest number of bits at the inputs of the adders of the round function, which, in turn, increases the probability of a given transformation. The last of the proposed methods allowed to implement an effective search for the differential characteristic for one round of transformation of the block cipher Cypress with a probability which is equal to 1/4.

The application of the proposed method of searching for multi-round differential characteristics, based on the construction of a set of high-probable one-round differential characteristics, to the block cipher Cypress-256 showed that built one-round differential characteristics, the output differences of which have a small Hamming weight, cannot be extended to construct multi-round differential characteristics with high probability.

One of the most probable multi-round differential characteristics found for the block cipher Cypress-256 can be constructed only for six rounds of encryption with the probability $\mathrm{MEDP}^{(6)}(\Omega) \approx 2^{-223}$. Thus, the block cipher Cypress-256 is secure against the differential cryptanalysis in terms of practical criteria.

The fifth chapter *(Experimental research of Cypress block cipher properties)* researches the statistical, avalanche and performance characteristics of the developed perspective block cipher Cypress. The research of the cipher statistical properties showed that the encryption transformation and the key schedule of the algorithms Cypress-256 and Cypress-512 meet the requirements for statistical testing of random sequences NIST STS.

A research of avalanche indicators of the Cypress block cipher showed that Cypress-256 (number of rounds is 10) and Cypress-512 (number of rounds is 14) meet the requirements for the avalanche effect starting with four encryption rounds.

The performance of the Cypress block cipher was compared with the performance of the known lightweight algorithms on Windows, Linux and Android platforms (involving one processor core in a single threaded application). The block cipher Cypress demonstrated high performance on all investigated software and hardware platforms. On the Windows 10 platform with a 32-bit architecture, the best result was shown by the cipher Cypress-256 (about 3.5 Gbps). On the Windows 10 platform with a 64-bit architecture, the best result was shown by the cipher Cypress-512 (approximately 5 Gbps). On the Linux platform with a 64-bit architecture, the block cipher Cypress-256 showed an extremely high performance (over 8 Gbps).

Key words: symmetric block cipher, lightweight block cipher, S-box, nonlinearity, algebraic immunity, round function, key schedule, ARX-cipher, differential cryptanalysis, differential characteristic.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Розділ монографії, опублікований у співавторстві, що входить до міжнародної наукометричної бази:

1. Andrushkevych A., Gorbenko Y., Kuznetsov O., Oliynykov R., Rodinko M. A Prospective Lightweight Block Cipher for Green IT Engineering. *Green IT Engineering: Social, Business and Industrial Applications. Studies in Systems, Decision and Control*. Springer, Cham. 2019. Vol. 171. P. 95–112. (Scopus). (Особистий внесок здобувача: формування розділів щодо принципів побудови та властивостей перспективного малоресурсного блокового шифру).

Публікації у наукових фахових виданнях України:

2. Родінко М. Ю., Олійников Р. В. Методи пошуку диференційних характеристик циклової функції симетричного блокового шифру «Кипарис». *Радиотехника*. 2017. Вып. 191. С. 47–51.
   (Особистий внесок здобувача: розробка методів пошуку одноциклових диференційних характеристик ARX-шифру та результати застосування до блокового шифру «Кипарис-256»).
3. Родінко М. Ю. Малоресурсний симетричний блоковий шифр «Кипарис» – сутність та основні властивості. *Математичне та комп'ютерне моделювання. Серія: Технічні науки*. 2017. Вип. 15. С. 203–208.
4. Лисицкая И. В., Лисицкий К. Е., Родинко М. Ю., Головко И. А., Жариков И. И., Корниенко М. А., Кулеба М. В. Экспериментальные данные по определению динамических показателей прихода блочных симметричных шифров к состоянию случайной подстановки. *Радіоелектроніка, інформатика, управління*. 2017. № 1. С. 129–141. (Web of Science).

(Особистий внесок здобувача: отримання експериментальних даних щодо кількості активних S-блоків на різних циклах шифрування для низки блокових шифрів).

5. Родінко М. Ю. Оцінка стійкості симетричного блокового шифру «Кипарис» до диференційного криптоаналізу. *Радиотехника*. 2018. Вып. 195. С. 113–124.
6. Елисеев Р. Ю., Родинко М. Ю., Олейников Р. В. Дифференциальный криптоанализ блочного ARX-шифра «Кипарис-256». *Прикладная радиоэлектроника*. 2018. Вып. 17, № 3–4. С. 121–126.

   (Особистий внесок здобувача: постановка задачі, формування методики дослідження диференційних властивостей шифру).

Публікація у періодичному науковому виданні держави-члена ЄС, що входить до міжнародної наукометричної бази:

7. Rodinko M., Oliynykov R., Gorbenko Y. Optimization of the High Nonlinear S-Boxes Generation Method. *Tatra Mountains Mathematical Publications*. 2017. Vol. 70. Is. 1. P. 93–105. (Scopus, Словаччина).

   (Особистий внесок здобувача: розробка удосконаленого методу генерації підстановок та його застосування для генерації оптимальних S-блоків).

Наукові праці, які засвідчують апробацію матеріалів дисертації:

8. Rodinko M., Oliynykov R., Gorbenko Y. Improvement of the high nonlinear S-boxes generation method. *Problems of Infocommunications Science and Technology (PIC S&T)* : Proceedings of 2016 Third International Scientific-Practical Conference, 4–6 October 2016, Kharkiv, 2016. P. 63–66. (Scopus).

   (Особистий внесок здобувача: розробка удосконаленого методу генерації S-блоків з високою нелінійністю).
9. Родінко М. Ю., Олійников Р. В., Горбенко І. Д. Перспективний блоковий шифр «Кипарис». *Комп'ютерне моделювання у наукоємних технологіях*

(КМНТ -2016) : матеріали Міжн. наук.-техн. конф., 26–31 травня 2016 р., Харків, 2016. С. 292–295.

(Особистий внесок здобувача: розробка перспективного блокового шифру).

10. Родінко М. Ю., Олійников Р. В. Аналіз неін'єктивних схем розгортання ключів симетричних блокових шифрів. *Безпека інформації в інформаційно-телекомунікаційних системах* : матеріали Міжн. наук.-практ. конф., 25–26 травня 2016 р., Київ, 2016. Вип. 18. С. 50.

    (Особистий внесок здобувача: аналіз колізійних властивостей неін'єктивних схем розгортання ключів симетричних блокових шифрів).
11. Родінко М. Ю. Оцінка ймовірності існування еквівалентних ключів для неін'єктивних схем розгортання ключів симетричних блокових шифрів. *Радіоелектроніка і молодь у XXI столітті* : матеріали XVIII Міжн. молодіжного форуму, 19–21 квітня 2016 р., Харків : ХНУРЕ, 2016. Т.5. С. 77–78.
12. Родінко М. Ю. Математична модель оцінки властивостей неін'єктивних схем розгортання ключів симетричних блокових шифрів. *Проблеми кібербезпеки інформаційно-телекомунікаційних систем* : матеріали доповідей наук.-практ. конф., 10–11 березня 2016 р., Київ, 2016. С. 71–72.
13. Родінко М. Ю., Олійников Р. В. Постквантовий малоресурсний алгоритм симетричного блокового перетворення «Кипарис». *Проблеми кібербезпеки інформаційно-телекомунікаційних систем* : матеріали доповідей наук.-практ. конф., 23–24 березня 2017 р., Київ, 2017. С. 172–176.

    (Особистий внесок здобувача: розробка принципів побудови алгоритму симетричного блокового перетворення «Кипарис» та дослідження його властивостей).
14. Родінко М. Ю., Олійников Р. В., Руженцев В. І., Єлисєєв Р. Ю. Підхід до оцінки диференційних властивостей перспективного симетричного блокового шифру «Кипарис». *Безпека інформації в інформаційно-телекомунікаційних системах* : матеріали XIX Міжн. наук.-практ. конф., 25–26 травня 2017 р., м. Буча, Київська обл., 2017. С. 105–106.

(Особистий внесок здобувача: розробка методу пошуку диференційних характеристик циклової функції блокового шифру «Кипарис»).

15. Rodinko M., Oliynykov R. Open problems of proving security of ARX-based ciphers to differential cryptanalysis. *Problems of Infocommunications Science and Technology (PIC S&T)* : Proceedings of 2017 4th International Scientific-Practical Conference, 10–13 October 2017, Kharkiv, 2017. P. 228–231. (Scopus).
(Особистий внесок здобувача: аналіз існуючих методів оцінки стійкості ARX-шифрів до диференційного криптоаналізу).
16. Rodinko M., Oliynykov R., Eliseev R. Search for one-round differential characteristics of lighweight block cipher Cypress-256. *Dependable Systems, Services and Technologies (DESSERT)* : Proceedings of 2018 IEEE 9th International Conference, 24–27 May 2018, Kyiv, 2018. P. 312–315. (Scopus).
(Особистий внесок здобувача: розробка методів пошуку одноциклових диференційних характеристик ARX-шифру «Кипарис»).
17. Rodinko M., Oliynykov R. An Approach to Search for Multi-Round Differential Characteristics of Cypress-256. *Problems of Infocommunications Science and Technology (PIC S&T)* : Proceedings of 2018 International Scientific-Practical Conference, 9–12 October 2018, Kharkiv, 2018. P. 659–662. (Scopus).
(Особистий внесок здобувача: розробка принципів методики пошуку багатоциклових диференційних характеристик ARX-шифру «Кипарис»).
18. Rodinko M., Oliynykov R. Comparing Performances of Cypress Block Cipher and Modern Lighweight Block Ciphers on Different Platforms. *Problems of Infocommunications, Science and Technology (PIC S&T)* : Proceedings of 2019 IEEE International Scientific-Practical Conference, 8–11 October 2019, Kyiv, 2019. P. 113–116. (Scopus).
(Особистий внесок здобувача: отримання експериментальних даних щодо швидкодії відомих малоресурсних блокових шифрів та шифру «Кипарис»).
19. Rodinko M., Oliynykov R. The Method of Searching for Differential Trails of ARX-based Block Cipher Cypress. *Dependable Systems, Services and*

Technologies (DESSERT) : Proceedings of 2020 IEEE 11th International Conference, 14–18 May 2020, Kyiv, 2020. P. 157–160. (Scopus).

(Особистий внесок здобувача: розробка методу пошуку багатоциклових диференційних характеристик ARX-шифру).

Наукові публікації, що додатково відображають зміст дисертації:

20. 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.

    (Особистий внесок здобувача: розробка методу оцінки колізійних властивостей неін'єктивних схем розгортання ключів).

21. Родінко М. Ю., Олійников Р. В. Дослідження продуктивності малоресурсного блокового шифру «Кипарис» на різних платформах. *Радіотехніка*. 2020. Вип. 200. С. 51–57. (Index Copernicus).

    (Особистий внесок здобувача: отримання експериментальних даних щодо швидкодії блокового шифру «Кипарис» та низки малоресурсних блокових шифрів).

Патенти:

22. Спосіб криптографічного перетворення двійкових даних: пат. 111448 Україна: № a201503976; заявл. 24.04.2015; опубл. 25.04.2016, Бюл. № 8. 6 с.

    (Особистий внесок здобувача: отримання експериментальних даних щодо кількості активних S-блоків на різних циклах запропонованого перетворення).

23. Спосіб криптографічного перетворення двійкових даних (варіанти): пат. 111547 Україна: № a201500942; заявл. 06.02.2015; опубл. 10.05.2016, Бюл. № 9. 13 с.

    (Особистий внесок здобувача: здійснення розрахунків щодо кількості активних S-блоків на різних циклах запропонованого перетворення).

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ДХ	–	Диференційна характеристика
СРК	–	Схема розгортання ключів
ТЛА	–	Таблиця лінійних апроксимацій
ТРР	–	Таблиця розподілу різниць
AEAD	–	Authenticated Encryption with Associated Data
AES	–	Advanced Encryption Standard
ARX	–	Addition, Rotation, XOR
DES	–	Data Encryption Standard
NESSIE	–	New European Schemes for Signatures, Integrity and Encryption
NIST	–	National Institute of Standards and Technology
NIST STS	–	NIST Statistical Test Suite
SPN	–	Substitution Permutation Network
XOR	–	Exclusive OR (операція додавання за модулем 2)

ВСТУП

Обґрунтування вибору теми дослідження. На сьогоднішній день спостерігається усталена тенденція розповсюдження та впровадження інформаційних технологій у всі сфери людської діяльності. Інтенсифікація використання електронного документообігу та інтернет-банкінгу, створення цифрових реєстрів та впровадження різноманітних електронних послуг [1,2] – все це робить нові виклики інформаційній безпеці. Інформація, що циркулює у вищезгаданих системах, потребує надійного захисту, а саме попередження модифікації та знищення, запобігання несанкціонованого доступу до інформаційних ресурсів тощо.

Окремо необхідно відмітити зростання кількості мобільних та фізичних пристроїв, які постійно підключені до мережі Інтернет та можуть виступати складовими компонентами Інтернету Речей (англ. Internet of Things) [3]. Часто такі пристрої оперують конфіденційною інформацією користувачів (наприклад, даними для доступу до онлайн-банкінгу), при цьому потенційно маючи низку вразливостей. Багато персональних даних передається й через саму мережу Інтернет різними додатками та потребує захисту.

Ефективний захист інформації, що циркулює у інформаційно-комунікаційних системах (ІКС), здійснюється за допомогою криптографічних методів та засобів [4,5]. Впровадження нових ІКС із доступом користувачів до відповідних ресурсів через розподілену недовірену мережу вимагає удосконалення існуючих методів захисту.

Однією з найголовніших проблем є забезпечення конфіденційності та цілісності інформації, що передається через мережу. Розв’язання цієї проблеми полягає у застосуванні криптографічних методів захисту інформації, серед яких окрему групу складають симетричні блокові шифри, які мають такі властивості, як простота реалізації, висока швидкодія та надійність. Окрім забезпечення конфіденційності, блокові шифри використовуються в якості компонентів

функцій гешування та кодів автентифікації повідомлень, а також для генерації псевдовипадкових послідовностей. Згідно з дослідженнями [6,7], блокові шифри залишаться актуальними навіть в умовах появи і поширення квантових комп'ютерів, а для забезпечення високого рівня криптографічної стійкості необхідно буде лише збільшити довжину ключа.

У липні 2015 р. в Україні був введений у дію новий стандарт симетричного блокового перетворення ДСТУ 7624:2014 [8], який визначає блоковий шифр «Калина». Алгоритм побудований з урахуванням останніх тенденцій у проєктуванні симетричних примітивів та має високий рівень криптографічної стійкості. У якості нелінійного перетворення у блоковому шифрі «Калина» використовується чотири S-блока, що мають оптимальні криптографічні показники [9,10]. Один з можливих режимів застосування шифру передбачає використання S-блоків в якості довгострокових ключових елементів. Генерація оптимальних S-блоків [10] за допомогою відомих методів є достатньо повільною при програмній реалізації, тому актуальною задачею є вдосконалення методів генерації підстановок з точки зору підвищення швидкодії.

Схема розгортання ключів (СРК) шифру «Калина» розроблялася з урахуванням необхідності захисту від атак на реалізацію та атаки на зв'язаних ключах [9]. З цією метою була розроблена односпрямована СРК, що забезпечує неможливість відновлення циклового ключа при знанні ключа шифрування або інших підключів. Особливістю односпрямованих СРК є те, що вони є неін'єктивними, тобто теоретично припускається існування еквівалентних ключів (таких, що формують однакову послідовність циклових ключів). Односпрямовані СРК застосовуються і в інших відомих блокових шифрах таких, як FOX [11], Twofish [12] та ін. Оцінка ймовірності співпадіння потужностей множини послідовностей циклових ключів, які формуються неін'єктивною СРК, і множини ключів шифрування дозволила б додатково обґрунтувати стійкість односпрямованих СРК, а отже й більш високий рівень стійкості національного стандарту симетричного блокового перетворення.

Водночас, блоковий шифр «Калина» за вимогами швидкодії та компактності реалізації не підходить для застосування у пристроях з обмеженою кількістю споживання енергії таких, як смарт-картки та ін. Для застосування у подібних пристроях розробляються малоресурсні шифри, які зайняли окрему нішу у сучасній криптографії. Інтерес до розробки малоресурсних примітивів проявляє й Національний інститут стандартів та технологій (англ. National Institute of Standards and Technology, NIST) Сполучених Штатів Америки (США), який проводить конкурс на розробку малоресурсного алгоритму автентифікованого шифрування з приєднаними даними (англ. authenticated encryption with associated data, AEAD) [13,14].

Як правило, малоресурсні алгоритми використовують прості операції, що багатократно повторюються. Широкого застосування набуло так зване ARX-перетворення, що включає послідовне застосування операцій додавання за модулем, XOR та циклічного зсуву n-бітових слів, що перемішуються та поступають на вхід цих самих операцій. Відсутність прозорого математичного підґрунтя ARX-шифрів викликає проблеми при оцінці стійкості таких примітивів до різних видів криптоаналізу, зокрема диференційного. На сьогоднішній день не існує універсальної теорії оцінки стійкості ARX-шифрів до диференційного криптоаналізу, тому кожен окремий клас таких алгоритмів потребує розробки своїх методів пошуку диференційних характеристик (ДХ) [15,16].

Майже всі відомі на сьогодні малоресурсні блокові алгоритми не забезпечують високого рівня криптографічної стійкості, оскільки не підтримують довжину ключа 256 та 512 бітів, а значить не зможуть бути застосованими у постквантовий період. Таким чином, актуальною задачею є розробка малоресурсного блокового шифру, в якому висока криптографічна стійкість поєднується з високою швидкодією.

Зв’язок роботи з науковими програмами, планами, темами. Дисертаційні дослідження проводились в рамках науково-дослідних робіт: № 1-41-16 «Аналіз стану, обґрунтування вимог та напрямів розвитку,

стандартизація, розробка та впровадження криптографічних систем для надання електронних довірчих послуг» (№ ДР 0116U000810), № 1-41-18 «Аналіз, дослідження, розробка та стандартизація криптографічних систем для захисту інформації в пост-квантовому середовищі, в умовах інформаційних і гібридних війн» (№ ДР 0118U002024).

Мета і завдання дослідження. Метою дисертаційної роботи є підвищення продуктивності симетричних криптографічних перетворень і удосконалення методів аналізу їх стійкості.

Для досягнення поставленої мети були сформульовані та вирішені наступні задачі.

1. Підвищення швидкості (зниження обчислювальної складності) генерації оптимальних S-блоків на доступній обчислювальній техніці за допомогою методу градієнтного спуску за рахунок удосконалення алгоритму перевірки підстановок на відповідність криптографічним показникам.

2. Суттєве удосконалення оцінок щодо ймовірності співпадіння потужностей множини послідовностей циклових ключів, які формуються неін’єктивною схемою розгортання циклових ключів, і множини ключів шифрування.

3. Розробка малоресурсного блокового симетричного шифру, що підтримує високий та надвисокий рівні стійкості (за класифікацією NESSIE), а також має компактну реалізацію та високу продуктивність на різних програмно-апаратних платформах.

4. Розробка математичної моделі для оцінки стійкості визначеного класу ARX-подібних шифрів до диференційного криптоаналізу, що базується на відомих положеннях теорії криптоаналізу та обґрунтованих припущеннях щодо властивостей компонентів таких шифрів.

5. Розробка методів пошуку найбільш ймовірних одноциклових диференційних характеристик визначеного класу ARX-шифрів з урахуванням особливостей побудування циклової функції, що використовується у шифрах визначеного класу.

6. Удосконалення методів пошуку багатоциклових диференційних характеристик визначеного класу ARX-шифрів із використанням відомих підходів та результатів застосування розроблених методів пошуку одноциклових диференційних характеристик.

Об’єкт дослідження – процеси захисту інформації з використанням симетричних блокових шифрів.

Предмет дослідження – методи побудови та дослідження малоресурсних симетричних блокових шифрів та їх компонентів.

Методи дослідження. При проведенні дисертаційних досліджень використовувалися методи теорії скінченних груп та полів, теорії булевих функцій, теорії ймовірностей та комбінаторного аналізу, математичного аналізу.

Методи теорії скінченних груп та полів, теорії булевих функцій та комбінаторного аналізу використовувалися при вирішенні задачі удосконалення методу градієнтного спуску для генерації нелінійних вузлів заміни.

Методи теорії ймовірностей, комбінаторного та математичного аналізу використовувалися для оцінки ймовірності колізії двох послідовностей циклових ключів, побудованих за допомогою неїн’єктивної схеми розгортання циклових ключів.

Методи теорії ймовірностей та комбінаторного аналізу використовувалися при розробці математичної моделі оцінки стійкості та методів пошуку високоймовірних диференційних характеристик визначеного класу ARX-шифрів, а також при розробці перспективного малоресурсного симетричного блокового шифру.

Наукова новизна отриманих результатів. У результаті вирішення поставлених задач були отримані наступні нові наукові результати.

1. Вперше запропоновано два методи пошуку одноциклових диференційних характеристик для визначеного класу ARX-шифрів, що дозволяють з низькою обчислювальною складністю отримувати оцінки

стійкості циклової функції блокового шифру визначеного класу до диференційного криптоаналізу.

2. Отримали подальший розвиток методи пошуку багатоциклових диференційних характеристик для визначеного класу ARX-шифрів, що відрізняються вдосконаленим механізмом відбору вхідних різниць та формуванням початкової множини одноциклових диференційних характеристик, що дозволяє отримати оцінку щодо стійкості повномасштабного блокового шифру визначеного класу до диференційного криптоаналізу.

3. Удосконалений метод градієнтного спуску для генерації нелінійних таблиць заміни, що відрізняється від відомого обґрунтованим оптимальним порядком застосування критеріїв при перевірці підстановок на відповідність криптографічним показникам, що дозволяє суттєво знизити складність генерації оптимальних S-блоків.

4. Отримав подальший розвиток математичний метод оцінки колізійних властивостей неін’єктивних схем розгортання ключів блокових шифрів, що відрізняється застосуванням вдосконаленої математичної моделі та більш ефективного математичного апарату та дозволяє отримати уточнену оцінку ймовірності колізії двох послідовностей циклових ключів.

Практичне значення отриманих результатів.

1. Запропонований удосконалений метод градієнтного спуску дозволяє зменшити середній час генерації оптимального байтового S-блоку з нелінійністю 104, δ-рівномірністю 8, алгебраїчним імунітетом 3 та мінімальною степінню булевих функцій 7 на персональному комп’ютері з 2,5 годин до 30 хвилин (скорочення часу розрахунків у 5 разів).

2. Отримано вдосконалене значення оцінки нижньої границі стійкості діючого національного стандарту ДСТУ 7624:2014, а саме доведено, що складність атак переборного типу на неін’єктивні схеми розгортання ключів практично дорівнює складності атак на ін’єктивні схеми.

3. Розроблений перспективний постквантовий малоресурсний симетричний блоковий шифр «Кипарис», що забезпечує високий та надвисокий

рівні стійкості та перевершує за швидкодією відомі малоресурсні блокові шифри на 32- та 64-бітових процесорах загального призначення та мобільних платформах.

4. Обґрунтовано стійкість симетричного блокового шифру «Кипарис-256» до диференційного криптоаналізу згідно з вимогами практичного критерію, а саме показано, що максимальна середня за ключами ймовірність диференційної характеристики є набагато меншою за ймовірність успіху атаки прямого перебирання ключів.

Теоретичні та практичні результати дисертаційних досліджень реалізовані у приватному акціонерному товаристві «Інститут інформаційних технологій» та застосовуються у навчальному процесі Харківського національного університету імені В. Н. Каразіна.

Особистий внесок здобувача. Дисертаційна робота є самостійно виконаною науковою працею, в якій викладено методи аналізу криптографічних властивостей компонентів симетричних блокових шифрів та побудови перспективних криптографічних перетворень. Усі наукові результати, викладені в дисертаційній роботі, одержані автором особисто і відображені у наукових публікаціях. З наукових праць, виданих у співавторстві, у роботі використані лише ті положення, що становлять індивідуальний внесок автора. Конкретний внесок здобувача в цих роботах зазначений у списку наукових публікацій, опублікованих за темою дисертації.

Апробація матеріалів дисертації. Основні результати дисертаційної роботи були представлені, доповідались та обговорювались на: ***9 міжнародних наукових, науково-практичних та науково-технічних конференціях, проведених в Україні***: III-VI-й Міжнародній науково-практичній конференції «Problems of Infocommunications Science and Technology (PIC S&T)» (м. Харків, 2016-2018 рр.; м. Київ, 2019 р.); IX-й та XI-й Міжнародній конференції «Dependable Systems, Services and Technologies (DESSERT)» (м. Київ, 2018 р., 2020 р.); XVIII-XIX-й Міжнародній науково-практичній конференції «Безпека інформації у інформаційно-телекомунікаційних системах» (м. Київ, 2016 р.;

м. Буча, 2017 р.); Міжнародній науково-технічній конференції «Комп'ютерне моделювання у наукоємних технологіях (КМНТ-2016)» (м. Харків, 2016 р.); ***2 науково-практичних конференціях*** «Проблеми кібербезпеки інформаційно-телекомунікаційних систем» (м. Київ, 2016-2017 рр.); ***XVIII Міжнародному молодіжному форумі*** «Радіоелектроніка і молодь у XXI столітті» (м. Харків, 2016 р.). Додатково результати роботи обговорювалися на: ***IX Міжнародній конференції IEEE*** «Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)» (м. Бухарест, Румунія, 2017 р.).

Публікації. Основні результати дисертаційних досліджень опубліковані у 23 наукових працях, серед яких: 1 розділ монографії, опублікований у співавторстві, що входить до міжнародної наукометричної бази (Scopus), 5 статей у наукових фахових виданнях України, 1 стаття у періодичному науковому виданні держави-члена ЄС, що входить до міжнародної наукометричної бази (Scopus), 2 статті, що додатково висвітлюють результати дисертації, 2 патенти України, 12 матеріалів та тез доповідей на конференціях.

Структура та обсяг дисертації. Дисертаційна робота складається зі вступу, 5 розділів, висновків, списку використаних джерел та 5 додатків. Обсяг загального тексту дисертації складає 201 сторінку, з них 113 сторінок основного тексту. Робота ілюстрована 17 рисунками та 26 таблицями. Список використаних джерел містить 142 найменування.

РОЗДІЛ 1
АКТУАЛЬНИЙ СТАН ТА ПЕРСПЕКТИВИ РОЗВИТКУ МЕТОДІВ АНАЛІЗУ ТА СИНТЕЗУ СИМЕТРИЧНИХ БЛОКОВИХ ШИФРІВ

1.1 Розвиток та стандартизація алгоритмів симетричного блокового шифрування

Сьогодні симетричні блокові шифри займають провідне місце серед криптографічних методів, а їх застосування вийшло далеко за рамки забезпечення конфіденційності. На основі компонентів симетричних блокових шифрів будуються геш-функції [17-19], генератори псевдовипадкових послідовностей [20] та коди автентифікації повідомлень [21], а низка режимів блокового шифру дозволяє ефективно застосовувати один примітив у різних цілях, роблячи його універсальним.

Першим стандартизованим симетричним блоковим шифром, введеним у дію у США, став шифр DES (Data Encryption Standard) [22], який пізніше був визнаний вразливим через невелику довжину ключа, існування слабких ключів та появу публікацій щодо диференційного й лінійного криптоаналізу шифру [23, 24]. У зв’язку із необхідністю заміни шифру DES у 1997 році NIST оголосив перший відкритий криптографічний конкурс із вибору алгоритму шифрування [25]. Переможець конкурсу AES (Advanced Encryption Standard) [25] повинен був стати новим стандартом симетричного шифрування США, що призначатиметься для використання не менше, ніж протягом тридцяти років. До алгоритму висувалися дві наступні обов’язкові вимоги [26]:

а) 128-бітний розмір блоку шифрування;

б) алгоритм повинен підтримувати не менше трьох розмірів ключів шифрування – 128, 192 та 256 бітів.

Також NIST висунув перелік рекомендованих вимог, серед яких стійкість алгоритму до всіх відомих на поточний момент криптоаналітичних атак,

простота й обґрунтованість структури алгоритму, відсутність слабких та еквівалентних ключів та ін. [26].

За результатами першого етапу конкурсу з усіх представлених алгоритмів було обрано п’ять фіналістів [27]: MARS [28], RC6 [29], Rijndael [30], Serpent [31] та Twofish [12]. Переможцем конкурсу став алгоритм Rijndael, на основі якого був прийнятий федеральний стандарт США FIPS-197 [32].

За аналогією з конкурсом AES подібні проєкти були виконані у Європі (NESSIE – New European Schemes for Signatures, Integrity and Encryption [33]) та Японії (CRYPTREC – Cryptography Research and Evaluation Committees [34]).

Основним завданням проєкту NESSIE, розпочатого у 2000 р., був відбір десяти кращих криптографічних примітивів. До кандидатів на стандарти симетричного блокового шифрування були висунуті підвищені вимоги. Для блокових шифрів було введено три класи стійкості [35]:

а) високий рівень безпеки – довжина блоку $l_б$ = 128 бітів, довжина ключа $l_к$ = 256 бітів;

б) нормальний рівень безпеки – довжина блоку $l_б$ = 128 бітів, довжина ключа $l_к$ = 128 бітів;

в) успадкований рівень безпеки – довжина блоку $l_б$ = 64 біта, довжина ключа $l_к$ = 128 бітів.

За результатами конкурсу були відібрані й рекомендовані до використання наступні алгоритми [35]: MISTY1 [36] (Японія), Camellia [37] (Японія), SHACAL-2 [38] (Франція) та AES (FIPS-197) [32].

Проєкт CRYPTREC був заснований у Японії у 2000 р. для оцінки та моніторингу електронного уряду. Проєкт рекомендує шифри, а також встановлює критерії оцінки криптографічних модулів. До списку [39], складеного в березні 2013 р., потрапили шифри Triple DES, AES та Camellia.

До недавнього часу чинним стандартом блокового шифрування України був радянський алгоритм ГОСТ 28147-89 [40]. Враховуючи необхідність створення нового вітчизняного стандарту шифрування, Державна служба спеціального зв’язку та захисту інформації (ДССЗІ) в 2006 р. оголосила про

початок відкритого конкурсу по висуванню кандидатів на національний стандарт [41]. За результатами конкурсу був відзначений шифр «Калина», на основі якого в 2014 р. був розроблений, а в липні 2015 р. введений в дію стандарт симетричного блокового перетворення ДСТУ 7624:2014 [8].

На заміну шифру ГОСТ 28147-89 в Росії та Білорусі були введені стандарти шифрування ГОСТ Р 34.12-2015 [42] та СТБ 34.101.31-2011 [43].

Конкурс CAESAR [44] був проведений з метою розробки автентифікованих шифрів (англ. authenticated cipher). Один з напрямів конкурсу стосувався розробки шифрів для застосування у малоресурсних додатках, фіналістами якого стали алгоритми Ascon та ACORN.

Процесс стандартизації та проведення конкурсів продовжується й сьогодні з метою розробки примітивів для більш вузького застосування. Так, NIST розпочав проєкт, присвячений малоресурсній криптографії, метою якого є аналіз та стандартизація алгоритмів, придатних для використання у середовищах з обмеженими можливостями з обробки та зберігання даних [13]. Наразі NIST проводить конкурс на розробку малоресурсного алгоритму AEAD та функції гешування, вимоги до якого наведено в [14]. Наразі опубліковано список кандидатів, що потрапили до другого етапу конкурсу [45].

1.2 Алгебраїчна модель симетричного блокового шифру

Розглянемо узагальнену модель симетричного блокового шифру [46].

Блоковий шифр – це функція E: $\{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$, яка приймає на вхід n-бітовий відкритий текст та k-бітовий ключ і повертає n-бітовий шифртекст [46]. Довжина ключа k та довжина блока n є параметрами блокового шифру.

Нехай для кожного ключа $K \in \{0,1\}^k$ існує функція E_K: $\{0,1\}^n \rightarrow \{0,1\}^n$, що визначається як $E_K(M) = E(K,M)$. Вимагається, щоб для будь-якого блокового шифру та будь-якого ключа K функція E_K була перестановкою на $\{0,1\}^n$. Функція E_K має інверсію, яку ми позначимо як E_K^{-1}.

Нехай E^{-1}: $\{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ визначається як $E^{-1}(K, C) = E_K^{-1}(C)$. Це зворотний блоковий шифр до E [46].

Характерною особливістю блокових шифрів є поділ вхідного повідомлення на блоки фіксованого розміру. Найпростішим режимом блокового шифру, що передбачає незалежне шифрування окремих блоків даних, є режим простої заміни, зображений на рисунку 1.1.

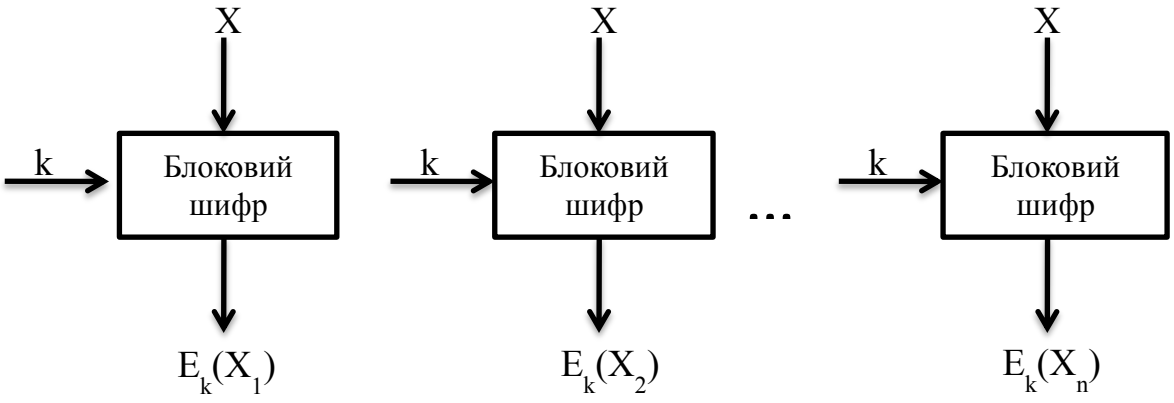

Рис. 1.1 Блоковий шифр у режимі простої заміни

Більшість блокових шифрів є ітеративними алгоритмами, що передбачає послідовне застосування декількох ключезалежних перестановок. Одна ітерація називається циклом, а ітеративна функція – цикловою функцією [47].

Блоковий шифр E називається r-цикловим ітеративним блоковим шифром, якщо для кожного ключа він може бути представлений композицією ключезалежних циклових перестановок таких, що для кожного $K \in F_2^k$ [48]:

$$E(K, \cdot) = f_r(K_r, \cdot) \circ f_{r-1}(K_r - 1, \cdot) \circ \cdots \circ f_2(K_2, \cdot) \circ f_1(K_1, \cdot), \qquad (1.1)$$

де $\circ$ позначає суперпозицію перестановок;

$f_i(K_i, \cdot)$: $F_2^b \rightarrow F_2^b$ – ключезалежні циклові перестановки;

K_i – циклові підключі, отримані з ключа шифрування K із використанням алгоритму розгортання ключів:

$$g : K \rightarrow (K_1, K_2, ..., K_{r-1}, K_r). \qquad (1.2)$$

Графічне представлення ітеративного блокового шифру наведене на рисунку 1.2 [48].

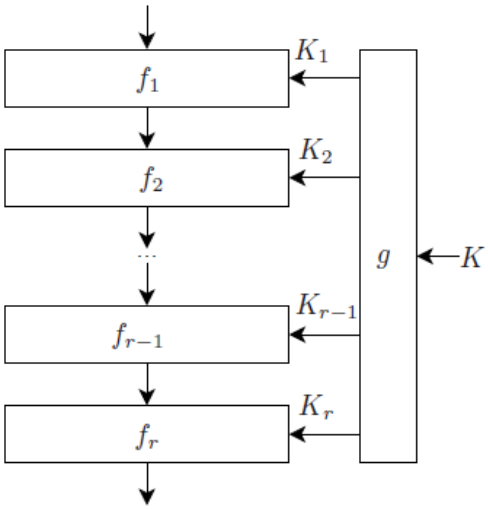

Рис. 1.2 Ітеративний блоковий шифр [48]

1.3 Високорівневі конструкції

В основі більшості блокових шифрів лежить одна з наступних високорівневих конструкцій [49]: SPN-структура (Substitution Permutation Network), мережа Фейстеля або схема Лей-Массі.

SPN-структура (див. рис. 1.3) складається з декількох перетворень, кожне з яких має спеціальне призначення [50].

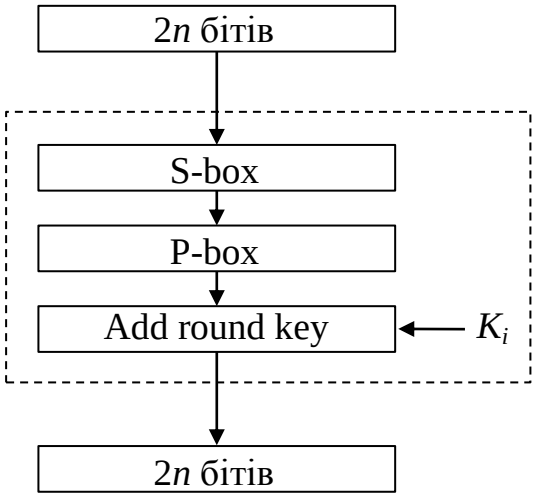

Рис. 1.3 Структура алгоритмів на основі SPN-структури [50]

Перетворення типу «S-box» застосовує нелінійну заміну до блока, що шифрується. Нелінійна функція часто реалізується за допомогою таблиці підстановки, яку називають S-блоком [47,49].

Перетворення типу «P-box» змішує різні частини блока, часто із використанням лінійної функції.

Перетворення типу «Add round key» застосовує ключ (підключ) до блока. Змішення ключа з блоком часто використовується за допомогою операції додавання за модулем 2 (XOR) або іншої групової операції.

На основі SPN-структури побудовані такі шифри як AES [32], PRESENT [51], SAFER [52], «Калина» [8] та ін.

На відміну від SPN-структури, мережа Фейстеля не обробляє весь блок даних за один цикл шифрування. Ця конструкція передбачає розбиття блока на декілька підблоків (частіше за все на два), один з яких обробляється деякою функцією F і накладається на один або декілька інших підблоків.

На рисунку 1.4 наведена найбільш поширена структура алгоритмів на основі мережі Фейстеля.

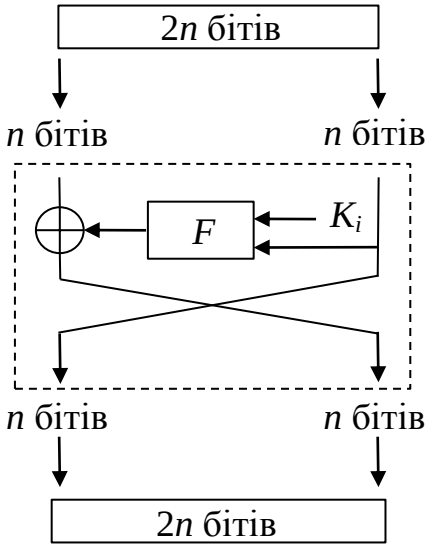

Рис. 1.4 Структура алгоритмів на основі мережі Фейстеля [50]

Накладання обробленого підблока на необроблений частіше за все виконується за допомогою логічної операції XOR або іншої групової операції.

Після накладання підблоки міняються місцями, тобто в наступному циклі алгоритму обробляється вже інший підблок даних.

Перевагою шифрів на основі мережі Фейстеля є можливість побудування інволютивного перетворення, коли для зашифрування та розшифрування застосовується один код алгоритму (різниця між цими операціями складається лише в порядку застосування підключів). Така властивість найбільш корисна при апаратній реалізації або на платформах з обмеженими ресурсами. На основі мережі Фейстеля побудовані алгоритми DES [22], Camellia [37], CAST [53], MARS [28], ГОСТ 28147-89 [40] та ін.

Альтернативною високорівневою конструкцією є схема Лей-Массі. Загальна схема одного циклу перетворень алгоритму на основі схеми Лей-Массі наведена на рисунку 1.5.

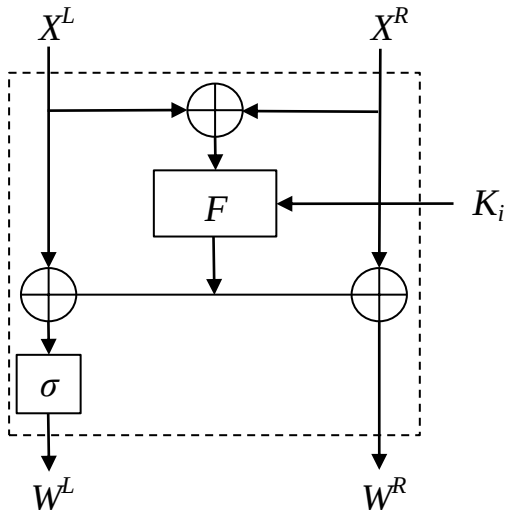

Рис. 1.5 Структура алгоритмів на основі схеми Лей-Массі [54]

Операції шифрування можуть бути представлені наступним чином [54]:

$$L_i = \sigma\left(L_{i-1} \oplus F\left(L_{i-1} \oplus R_{i-1}, K_{i-1}\right)\right), \tag{1.3}$$

$$R_i = R_{i-1} \oplus F\left(L_{i-1} \oplus R_{i-1}, K_{i-1}\right) \text{ при } i \in 1\ldots n-1. \tag{1.4}$$

На останньому циклі перетворення σ відсутнє [54]:

$$L_n = L_{n-1} \oplus F\left(L_{n-1} \oplus R_{n-1}, K_{n-1}\right), \quad (1.5)$$

$$R_n = R_{n-1} \oplus F\left(L_{n-1} \oplus R_{n-1}, K_{n-1}\right), \quad (1.6)$$

де R_i та L_i – права та ліва частини повідомлення на i-ому циклі;

σ – деяке ортоморфне перетворення;

F – функція ускладнення, що залежить від ключа.

Як і мережа Фейстеля, схема Лей-Мессі забезпечує властивість інволютивності перетворення. Окрім того, ця високорівнева конструкція не вимагає наявності властивості бієктивності циклової функції, що спрощує розробку та реалізацію.

На основі схеми Лей-Мессі побудовані такі симетричні блокові шифри як IDEA (International Data Encryption Standard) [55], FOX [11] та ін.

1.4 Циклова функція

Ідея побудови блокових шифрів із використанням послідовності простих шифруючих перетворень була закладена ще К. Шеноном [56]. Одночасно із цим К. Шенон ввів поняття перемішування (англ. confusion) та розсіювання (англ. diffusion), які й сьогодні використовуються при проєктуванні блокових алгоритмів.

Перемішування необхідне для того, щоб зробити зв’язок між ключем та шифртекстом складним для аналізу [56,57]. Це означає, що один символ шифртексту повинен залежати від декількох частин ключа [57]. Простий спосіб зробити це полягає у використанні підстановки, тобто заміни деякого блока бітів іншим, наприклад, за допомогою таблиці підстановок.

Дифузія (розсіювання) забезпечує розповсюдження збитковості структур відкритого тексту на зашифрований текст [56]. Дифузія означає, що зміна одного символа відкритого тексту призводить до зміни декількох символів

шифртексту та навпаки [57]. Одним зі способів досягнення дифузії є використання транспозиції, тобто перестановки порядку слідування частин відкритого тексту. На практиці замість простої перестановки, як правило, застосовується деяка лінійна функція.

Циклова функція будь-якого блокового шифру обов'язково містить як нелінійні, так і лінійні перетворення, які повинні володіти певними криптографічними властивостями.

1.4.1 Принцип побудови лінійного перетворення

Ефективність дифузії лінійного перетворення блокового шифру визначається таким показником, як кількість гілок активізації (англ. branch number). Для лінійного перетворення A розмірністю $n \times n$, визначеного над скінченним полем F^n, кількість гілок активізації $\beta(A)$ обчислюється як [58]

$$\beta(A) = \min\left\{ wt(x) + wt(Ax^T) \middle| x \in F^n, x \neq 0 \right\}, \qquad (1.7)$$

де $wt(x)$ – вага Гемінга від x.

Лінійне перетворення будується таким чином, щоб максимізувати значення $\beta(A)$, тобто зробити дифузію якомога більш ефективною. За допомогою використання більш загальних лінійних або афінних функцій, ніж проста перестановка, потужність дифузійного перетворення може бути збільшена [56]. Часто лінійне перетворення представляється у вигляді матриці над скінченним полем, на яку помножується проміжне значення шифртексту. Так, наприклад, у шифрі AES застосовується матриця розмірністю 4×4, для якої $\beta(A) = 5$. Для підвищення швидкодії обирають й матриці розмірністю 8×8 (наприклад, «Калина»), водночас матриця розмірністю 16×16 є неефективною з точки зору реалізації.

У ARX-шифрах в якості лінійних операцій зазвичай виступають додавання за модулем 2 та (циклічний) зсув бітів [59-61]. Оскільки, лінійне перетворення не базується на математичних функціях з визначеними криптографічними властивостями, а ARX-перетворення, як правило, немає чіткого поділу на лінійну та нелінійну частини, обчислення $\beta(A)$ є складною задачею, вирішення якої зводиться до повного перебирання усіх можливих варіантів входів.

1.4.2 Методи побудови нелінійних таблиць заміни (S-блоків)

Як зазначалось вище, застосування підстановок (S-блоків) забезпечує нелінійний характер зв'язку між відкритим текстом, ключем шифрування та шифртекстом. Властивості S-блоків значно впливають на стійкість (та запас стійкості) шифру до різних методів криптоаналізу [62,63]. Відбір та подальше застосування S-блоків з покращеними криптографічними показниками дозволяє покращити характеристики симетричного перетворення. Це дає змогу зменшити число ітерацій алгоритму, зберігши при цьому рівень його стійкості до криптоаналітичних атак [64].

S-блок називається оптимальним [10], якщо він задовольняє сукупності відомих на поточний момент граничних значень показників, що визначають стійкість симетричного перетворення до диференційного, лінійного та алгебраїчного криптоаналізу. Випадкові S-блоки не є оптимальними, в основному, через низькі показники стійкості до лінійного криптоаналізу [10].

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 [65]. 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.

У 2-му розділі пропонується удосконалення методу генерації підстановок, яке полягає у визначенні найбільш оптимального порядку застосування критеріїв відбору. Наводяться результати застосування запропонованого підходу до відомого методу генерації підстановок з високою нелінійністю (модифікованого методу градієнтного спуску [66]).

1.5 Принципи побудови та атаки на схеми розгортання ключів

У ітеративних блокових шифрах на кожній ітерації перетворення застосовується унікальний ключ (підключ). З точки зору криптографічної стійкості, всі підключі краще генерувати випадково та незалежно від інших, оскільки:

- це збільшує складність криптоаналітичних атак таких, як диференційний та лінійний криптоаналіз, атака на зв'язаних ключах та ін.;
- більшість відомих моделей оцінки стійкості блокових шифрів до криптоаналітичних атак передбачають, що підключі є рівноймовірними і незалежними.

Однак, зазначений підхід є незастосовуваним на практиці через високу складність генерації, зберігання та передавання послідовності (як правило, не менше, ніж з 10-ти підключів) відносно довгих ключів (як правило, від 128 до 512 бітів). У зв'язку з цим, у симетричних примітивах генерується один ключ – ключ шифрування (майстер-ключ), з якого обчислюються (розгортаються) підключі. Такий алгоритм обчислення підключів називається схемою

розгортання ключів (англ. key schedule, ключовий розклад). У [67] під схемою розгортання ключів розуміється алгоритм, що розширює відносно короткий майстер-ключ у відносно великий розширений ключ (декілька сотень чи тисяч бітів) для застосування в алгоритмах зашифрування та розшифрування.

Оскільки СРК є детермінованим алгоритмом, вона може бути вразливою до певних атак. Основними атаками на схеми розгортання ключів, метою яких є отримання циклових ключів та ключа шифрування, є наступні.

Слайд-атака. Атака [68] була вперше описана А. Бірюковим та Д. Вагнером у 1999 р. та є криптографічною атакою на основі підібраного відкритого тексту. Слайд-атака експлуатує степінь самоподоби блокового шифру та в основному застосовується до ітеративних блокових шифрів з періодичною СРК. Ідея атаки полягає у зсуві однієї копії процесу зашифрування відносно іншої копії процесу зашифрування таким чином, що два процеси є зсунутими на один цикл [68]. Це дає можливість легко отримати ключ шифрування після однієї ітерації шифру.

Атака на зв'язаних ключах. Атака припускає [69], що криптоаналітику відоме деяке математичне співвідношення, що зв'язує між собою ключі. Наприклад, співвідношення може бути простим значенням XOR з відомою константою $K_1 = K_2 \oplus C$ або більш складним зв'язком [69]. Атака вперше була запропонована Е. Біхамом та нагадує слайд-атаку.

Слабкі ключі. Слабким вважається ключ *K*, для якого зашифрування є ідентичною функцією до розшифрування [67]. Якщо число слабких ключів відносно мале, вони можуть не представляти загрози для шифру, якщо той використовується для забезпечення конфіденційності. Однак в деяких геш-функціях, що побудовані на основі блокових шифрів, зловмисник може обрати вхідне значення ключа при спробі пошуку колізії. В таких режимах блоковий шифр не повинен мати слабких ключів.

Прості зв'язки та еквівалентні ключі. Простий зв'язок виникає між двома різними ключами і проявляється як співвідношення між відкритими текстами та шифртекстами [67]. Два ключа є еквівалентними, якщо вони

зашифровують усі відкриті тексти ідентично [67]. Це може розглядатися як спеціальний вид простого зв’язку.

Атаки на СРК, що не є односпрямованими. СРК не є односпрямованою, якщо маючи декілька циклових підключів, зловмисник може отримати інформацію про ключ шифрування або інші невідомі підключі [67]. Наприклад, відновлення декількох циклових підключів дозволяє відновити більшу частину майстер-ключа в СРК DES. Е. Біхам та А. Шамір використали це для оптимізації їхньої диференційної атаки на DES. Крім того, це може зробити простішим пошук слабких та зв’язаних ключів для СРК, що не є односпрямованими [67].

Як видно з представленого огляду, як відсутність властивості односпрямованості, так і наявність еквівалентних ключів шифрування може призвести до можливості здійснення ряду серйозних атак, направлених на відновлення ключа шифрування. Однак, односпрямовані СРК, застосування яких видається найбільш доцільним з точки зору захисту від багатьох атак, є неін’єктивними, тобто теоретично потужність множини послідовностей циклових ключів є меншою за потужність множини ключів шифрування. Зауважимо, що саме така СРК реалізована у блоковому шифрі «Калина».

Таким чином, оцінка ймовірності співпадіння двох послідовностей циклових ключів, розгорнутих з різних ключів шифрування, є актуальною задачею, рішення якої дозволить додатково обґрунтувати стійкість неін’єктивних СРК та доцільність їх використання у блокових шифрах. Вирішенню цього питання присвячений 3-й розділ дисертації.

1.6 Методи оцінки стійкості симетричних блокових шифрів до диференційного криптоаналізу

Диференційний криптоаналіз був запропонований Елі Біхамом та Аді Шаміром у 1991 р. на алгоритм DES [23] і став першим відомим методом отримання секретного ключа шифру зі складністю, меншою складності повного

перебору ключів. В основі диференційного криптоаналізу блокових шифрів лежить аналіз проходження різниці між двома відкритими текстами крізь цикли шифрування та оцінка ймовірності перетворення вхідної різниці a у вихідну різницю b [23,70].

Нехай задано ітеративний блоковий шифр $E_k^{(r)}(x)$ з розміром ключа κ, що складається з r ітерацій циклової функції $f(x)$. Пара (a,b) називається одноцикловим диференціалом, ймовірність якого для n-бітового вхідного значення $f(x)$ обчислюється як [71,72]

$$\mathrm{DP}(a,b) = 2^{-n} \#\left\{ x \middle| f(x) + f(x+a) = b \right\}. \tag{1.8}$$

Аналогічно можна побудувати i-цикловий диференціал та r-цикловий диференціал:

$$\mathrm{DP}(a,b) = 2^{-n} \#\left\{ x \middle| E_k(x) + E_k(x+a) = b \right\}. \tag{1.9}$$

Середня за ключами ймовірність диференціалу (a,b) обчислюється як

$$\mathrm{EDP}(a,b) = 2^{-\kappa} \sum_{k \in F_2^\kappa} \mathrm{DP}\left[k\right](a,b). \tag{1.10}$$

Максимальне значення диференційної ймовірності визначається як

$$\mathrm{MEDP}(a,b) = \max_{a \neq 0, b} \mathrm{EDP}(a,b). \tag{1.11}$$

Значення $\mathrm{MEDP}(a,b)$ називається теоретичною стійкістю блокового шифру до диференційного криптоаналізу [73]. Однак, як правило, при аналізі

блокових шифрів, користуються оцінкою практичної стійкості [73], яка визначається максимальною ймовірністю диференційної характеристики.

Середньою за ключами *r*-цикловою диференційною характеристикою Ω називається $(r+1)$-мірний кортеж $\Omega=(\alpha,\beta_1,...,\beta_r)$, де β_i відповідає різниці, отриманій після *i*-го циклу шифрування двох вхідних текстів з різницею α [70,72]. Середня за ключами ймовірність ДХ Ω обчислюється як

$$\mathrm{EDP}(\Omega)=2^{-\kappa}\sum_{k\in F_2^{\kappa}} P(\Delta Y(1)=\beta_1,...,\Delta Y(r)=\beta_r\,|\,\Delta X=\alpha). \qquad (1.12)$$

Максимальне значення ймовірності ДХ визначається як

$$\mathrm{MEDP}(\Omega)=\max_{\Omega\in(a,b)}\mathrm{EDP}(\Omega(a,b)). \qquad (1.13)$$

Як правило, сучасні ітеративні блокові шифри є марковськими. Ітеративний шифр з цикловою функцією $Y=f(X,Z)$ є марковським, якщо ймовірність $P(\Delta Y=\beta\,|\,\Delta X=\alpha,X=\gamma)$ не залежить від γ у випадку, коли циклові ключі є випадково розподіленими [70]. Для марковського шифру ймовірність багатоциклової характеристики може бути апроксимована добутком ймовірностей одноциклових характеристик.

Для успішної атаки на блоковий шифр необхідно знайти $(r-1)$-циклову диференційну характеристику з ймовірністю [70]

$$P(E_k^{(r-1)}(x)+E_k^{(r-1)}(x+a)=b)=p>>2^{-n}. \qquad (1.14)$$

Традиційно, показники стійкості блокових шифрів до атак диференційного криптоаналізу прийнято пов’язувати з диференційними

властивостями S-блоків [10,62]. Основним диференційним показником S-блока S є максимальне значення в таблиці розподілу різниць [74], що визначається як

$$\delta = \max_{\alpha \in F_2^n, \alpha \neq 0, \beta \in F_2^n} \#\{x \mid S(x) \oplus S(x \oplus \alpha) = \beta\}. \quad (1.15)$$

Відповідно, оптимальні характеристики стійкості перетворення до атак диференційного криптоаналізу пов'язані з низькими значеннями максимумів перетворення нетривіальних різниць.

Загалом, для блокових шифрів, що базуються на S-блоках, максимальна ймовірність ДХ для одного циклу перетворення визначається максимумом таблиці розподілу різниць S-блока та мінімальною кількістю гілок активізації лінійного перетворення [50]. Подібний підхід є застосовуваним до шифрів, побудованих згідно зі стратегією широкого сліду, таких як AES, Калина, Camellia та ін. завдяки тому, що нелінійна та лінійна складові цих алгоритмів побудовані із застосуванням прозорого математичного апарату, а значить мають теоретично обґрунтовані криптографічні властивості [75].

Останнім часом все більшої популярності набирають блокові шифри, що базуються на простих операціях – додаванні за модулем, (циклічному) зсуві, логічному «І» і т.п. Так, багато сучасних малоресурсних алгоритмів відносяться до так званих ARX-шифрів (Addition-Rotation-XOR), що складаються з операцій додавання за певним модулем, циклічного зсуву та додавання за модулем 2. Оскільки подібна циклова функція є скоріше інтуїтивно обраною, ніж математично обґрунтованою, традиційний метод оцінки стійкості до диференційного криптоаналізу для ARX-шифру не підходить.

На сьогоднішній день не існує універсальної теорії щодо оцінки стійкості ARX-шифрів до диференційного криптоаналізу, тому оцінювання в основному базується на евристичних методах пошуку найкращої диференційної характеристики. При цьому методи, що застосовуються до різних класів ARX-шифрів, можуть мати певні особливості. Розробка та вдосконалення методів

оцінки стійкості ARX-шифрів до диференційного криптоаналізу, які фактично зводяться до пошуку диференційних характеристик з високою ймовірністю, є однією з пріоритетних задач сучасної криптографії.

1.7 Особливості побудування малоресурсних симетричних блокових шифрів

1.7.1 Аналіз вимог до малоресурсних симетричних примітивів

Малоресурсна криптографія (англ. lightweight cryptography) [76,77] займається розробкою симетричних примітивів, які забезпечують високу швидкодію перетворень та компактну реалізацію на різних платформах (персональних комп'ютерах, мобільних пристроях, серверах). Малоресурсні симетричні шифри знаходять широке застосування у таких технологіях як Інтернет Речей, смарт-лічильники, системи безпеки для автомобілів, бездротові системи моніторингу стану пацієнта та ін. [77]. Для пристроїв із низьким споживанням енергії, наприклад, пристроїв з автономним джерелом живлення, виконання неенергоємних криптографічних операцій має суттєве значення [77].

Попередній звіт NIST, опублікований у рамках проєкту зі стандартизації малоресурсних примітивів [78], містить огляд існуючих малоресурсних алгоритмів і пристроїв, в яких вони застосуються, мету проєкту та вимоги до перспективних малоресурсних примітивів, серед яких:

− криптографічна стійкість, що передбачає забезпечення рівня стійкості не менше, ніж 2^{112} операцій;

− гнучкість, що включає можливість ефективної реалізації алгоритму на різних платформах, а також допускає варіації реалізації на одній платформі;

− схожість різних функцій, тобто операції зашифрування та розшифрування використовують подібні циклові функції, що вимагає меншої кількості ресурсів, необхідних для реалізації різних операцій на одному пристрої;

− відсутність значного розширення шифртексту у порівнянні з відкритим текстом;

− стійкість до атак по побічним каналам та атак на реалізацію, що є необхідним, враховуючи той факт, що зловмисник може отримати фізичний доступ до пристрою;

− стійкість до атак на зв’язаних ключах.

Вимоги до алгоритмів AEAD, які подаються на конкурс, та загальні критерії їх оцінки, є наступними [14]:

− стійкість до атак по побічним каналам та атак на реалізацію;

− витрати з пам’яті та енергоспоживання;

− показники продуктивності;

− наявність результатів аналізу алгоритму, проведеного незалежними експертами, або використання в алгоритмі компонентів існуючих стандартів;

− придатність як для програмної, так і для апаратної реалізації.

1.7.2 Огляд сучасних малоресурсних симетричних блокових шифрів

Так само як і в традиційних, у малоресурсних блокових шифрах найбільш поширеними типами високорівневої конструкції є SPN-структура та мережа Фейстеля. В деяких шифрах застосовуються специфічні різновиди цих конструкцій такі, наприклад, як ARX-подібна SPN-структура у блоковому шифрі SPARX [16] або один з варіантів узагальненої мережі Фейстеля (англ. Generalized Feistel Network, GFN) у шифрах CLEFIA [79] та TWINE [80].

Серед блокових шифрів, заснованих на SPN-структурі, окремо можна виділити AES-подібні алгоритми, до яких відносяться шифри KLEIN [81] та Midori [82]. Причиною, з якої автори спираються на AES при розробці нових алгоритмів, є високий рівень вивченості шифру та його стійкість, що доведена математично та перевірена часом. Алгоритми-наступники, як правило, мають менший розмір блока та оперують напівбайтами замість байтів (KLEIN-64) або підтримують різні варіанти перетворень в залежності від розміру блока

(Midori-64 оперує напівбайтами, а Midori-128 – байтами). Застосування архітектури шифру AES забезпечує стійкість алгоритму, в той час як зменшення розмірів параметрів перетворення дозволяє підвищити швидкодію та зробити реалізацію шифру придатною для використання у апаратних пристроях з обмеженою кількістю споживання енергії.

До малоресурсних SPN-шифрів також відносяться PRESENT [51], PRINCE [83], PRIDE [84], Rectangle [85] та ін. Найпопулярнішою довжиною блока тут залишається 64 біти, довжина ключа, як правило, тримається на рівні 128 бітів з метою забезпечення нормального рівня стійкості.

Багато малоресурсних блокових шифрів засновано і на мережі Фейстеля, серед яких, наприклад, KASUMI [86], SEA (Scalable Encryption Standard) [87], спроєктовані з метою ефективного застосування в апаратних пристроях. Необхідно відмітити й алгоритми, що засновані на модифікаціях класичної мережі Фейстеля. Подібні високорівневі конструкції мають назву узагальнених мереж Фейстеля (GFN) і включають незбалансовані мережі Фейстеля (два напівблока мають різний розмір), мережі Фейстеля, в яких кількість підблоків складає більше двох та ін. До таких шифрів належать CLEFIA [79], TWINE [80].

Також в якості циклової функції мережі Фейстеля у малоресурсних алгоритмах все частіше застосовується не чергування S-блоків та лінійного перетворення, а так зване ARX-перетворення, що складається з операцій модульного додавання, циклічного зсуву та додавання за модулем 2. Операції, як правило, виконуються не над байтами, а над цілим напівблоком, розмір якого, найчастіше складає 32 або 64 біти, що суттєво підвищує швидкодію перетворень при використанні шифру на процесорах з архітектурою аналогічної розрядності. До таких шифрів відносяться Chaskey [88], LEA [61], SPECK [60], TEA [89], XTEA [90]. У блоковому шифрі SIMON [60] замість додавання за модулем застосовується операція побітового «І».

У деяких потокових симетричних шифрах ARX-перетворення застосовується без мережі Фейстеля як повноцінна високорівнева конструкція (ChaCha [59], Salsa20 [91]).

Окремо необхідно відмітити блокові шифри SPARX та LAX [16], принципи побудування яких дозволили цим шифрам стати першими ARX-подібними алгоритмами, які є доказово стійкими до диференційного та лінійного криптоаналізу. Блоковий шифр SPARX побудований згідно зі стратегією довгого сліду (англ. long trail strategy), запропонованою авторами по аналогією зі стратегією широкого сліду [75], відповідно з якою був розроблений блоковий шифр AES. Замість S-блока нова стратегія оперує поняттям ARX-блока, що складається з операцій додавання, циклічного зсуву та XOR і забезпечує нелінійність та дифузію. Відповідно до стратегії довгого сліду, блоковий шифр розглядається як ланцюг з ARX-блоків, які перемежовуються з операцією додавання з ключем [16]. Кожний цикл шифрування SPARX складається з такого ARX-блока та лінійного шару.

Також автори алгоритму SPARX пропонують ще один ARX-подібний блоковий шифр з доказовою стійкістю, що має назву LAX. Цей шифр не наслідує стратегію довгого сліду, а заснований на ідеї мінімізації максимальної диференційної ймовірності, що проходить крізь декілька циклів шифрування.

Як видно з представленого огляду, в основному, малоресурсні блокові шифри мають розмір блока та ключа шифрування не більше 128 біт (в деяких випадках, ключ може дорівнювати 256 бітам). Це пов’язано з тим, що основною метою цих алгоритмів є швидкодія та компактна реалізація, що обумовлює використання достатньо невеликих розмірів параметрів. Між тим, для можливості застосування у постквантовий період блоковий шифр повинен підтримувати розміри блока та ключа, що дорівняють 256 та 512 біт.

Таким чином, актуальною задачею є розробка малоресурсного блокового шифру, який забезпечуватиме як високу криптографічну стійкість, так і високу продуктивність. Слід оцінити стійкість шифру до найбільш розвиненої атаки – диференційного криптоаналізу, а для цього необхідно розробити методи оцінки стійкості визначеного класу ARX-шифрів до диференційного криптоаналізу, оскільки, як було згадано вище, призначених для цього універсальних методів не існує. Розв’язанню цих задач присвячений 4-ий розділ дисертації.

Висновки до розділу 1

1. Основними високорівневими конструкціями, що використовуються при побудуванні блокових шифрів, є SPN-структура, мережа Фейстеля та схема Лей-Мессі. Ідея SPN-структури полягає у послідовному застосуванні перетворень типу підстановка, перестановка та складання з ключем з метою досягнення високої степені перемішування та дифузії. Перевагою мережі Фейстеля та схеми Лей-Мессі є можливість побудування інволютивного перетворення, що корисно при апаратній реалізації або на платформах з обмеженими ресурсами.

2. Циклова функція блокового шифру обов'язково містить як лінійні, так і нелінійні перетворення. В якості нелінійного перетворення зазвичай використовуються S-блоки з визначеними (оптимальними) криптографічними показниками (диференційними, лінійними, алгебраїчними та ін.). На сьогоднішній день існують різні методи генерації оптимальних S-блоків. Широкого застосування набули методи генерації засновані на використанні збалансованих булевих функцій, які дозволяють побудувати S-блоки з високою нелінійністю, однак, такі S-блоки не задовольняють сучасним вимогам зі стійкості до алгебраїчного криптоаналізу. Тому все більш популярними стають евристичні методи, які, як правило, включають перевірку сформованого S-блока на відповідність набору критеріїв. Евристичні методи вимагають використання значних обчислювальних ресурсів для генерації оптимальних S-блоків, що може стати проблемою при застосуванні цих методів для генерації підстановок для використання у якості довгострокових ключових елементів. Таким чином, актуальною є задача вдосконалення існуючих методів генерації оптимальних S-блоків з точки зору підвищення швидкодії. Результати розв'язання цієї задачі представлені у другому розділі дисертації.

3. Важливим компонентом блокового шифру є схема розгортання циклових ключів. З 1-го липня 2015 р. в Україні введений у дію стандарт криптографічного перетворення ДСТУ 7624:2014, що визначає блоковий шифр

«Калина» та режими його роботи. У шифрі використовується нова односпрямована неін’єктивна СРК, для якої теоретично припускається, що ймовірність існування еквівалентних ключів більше нуля. Таким чином, актуальною є задача визначення ймовірності співпадіння потужностей множини послідовностей циклових ключів, які формуються СРК шифру «Калина» або подібною неін’єктивною СРК, і множини ключів шифрування. Результати розв’язання цієї задачі представлені у третьому розділі дисертації.

4. У зв’язку з поширенням мобільних технологій, Інтернету Речей та ін. широкого застосування набуває малоресурсна криптографія. Актуальною задачею є створення малоресурсного симетричного блокового шифру з високим рівнем стійкості, необхідним для використання шифру у постквантовий період, та високими показниками продуктивності. Широкого застосування у малоресурсних алгоритмах набула конструкція, що має назву ARX і передбачає застосування простих операцій таких, як додавання за модулем, циклічний зсув та XOR. Одним з основних етапів при розробці блокового шифру є оцінка його стійкості до відомих методів криптоаналізу, зокрема, диференційного. Водночас, на сьогоднішній день не існує універсальних методів оцінки стійкості ARX-шифрів до диференційного криптоаналізу. Отже, з метою доведення стійкості малоресурсного алгоритму до цієї атаки необхідно розробити методи оцінки стійкості класу ARX-шифрів, до якого входить алгоритм, що оцінюється. Фактично, ця задача зводиться до розробки нових або вдосконалення існуючих методів пошуку найкращих диференційних характеристик шифру. Результати розробки малоресурсного шифру та методів оцінки визначеного класу ARX-шифрів наводяться у четвертому розділі дисертації.

Результати досліджень даного розділу наведено в публікації здобувача: [64].

РОЗДІЛ 2
РОЗРОБКА УДОСКОНАЛЕНОГО МЕТОДУ ГЕНЕРАЦІЇ ОПТИМАЛЬНИХ S-БЛОКІВ З ВИСОКИМИ НЕЛІНІЙНІСТЮ ТА АЛГЕБРАЇЧНИМ ІМУНІТЕТОМ

2.1 Характеристика та основні критерії відбору S-блоків блокових шифрів

Як було зазначено у першому розділі, S-блоки відіграють ключову роль у забезпеченні стійкості блокового шифру до основних методів криптоаналізу: диференційного [23], лінійного [24] та алгебраїчного [92]. З цих причин S-блоки симетричних блокових шифрів повинні мати визначені криптографічні показники.

Основні критерії відбору S-блоків можна розділити на дві групи. Перша група включає критерії, що беруть до уваги стійкість перетворення до методів криптоаналізу. До другої групи входять критерії, засновані на оцінці криптографічних властивостей булевих функцій S-блока [93]. До них відносять нелінійність, максимум автокореляції, критерій розповсюдження та ін. Проте, як було показано в [10], чимало з критеріїв цієї групи є несуттєвими або надлишковими при застосуванні у блокових шифрах.

Максимум таблиці розподілу різниць визначається як [94]

$$\delta = \max_{\alpha \in F_2^n, \alpha \neq 0, \beta \in F_2^n} \#\left\{ x \middle| S(x) \oplus S(x \oplus \alpha) = \beta \right\}. \qquad (2.1)$$

Властивість впливає на стійкість шифру до диференційного криптоаналізу, який є однією з найбільш універсальних та ефективних атак на блокові шифри.

Еквівалентним поняттю максимального значення таблиці розподілу різниць (TPP) є поняття δ-рівномірності [94].

Означення 2.1 [94]. Нехай G_1 та G_2 – скінченні абелеви групи. Відображення F: $G_1 \to G_2$ називається диференційно δ-рівномірним, якщо для всіх $\alpha \in G_1$, $\alpha \neq 0$ та $\beta \in G_2$

$$|\{z \in G_1 | F(z + \alpha) - F(z) = \beta\}| \leq \delta. \tag{2.2}$$

Відповідно до цього визначення оптимальні характеристики стійкості перетворення F до атак диференційного криптоаналізу пов'язані з низькими значеннями δ-рівномірності. Очевидно, що вимога низьких значень δ-рівномірності еквівалентна вимозі низьких значень максимумів перетворення нетривіальних різниць. У зв'язку з цим, для досягнення високої стійкості перетворення необхідно отримати низькі значення δ-рівномірності.

Абсолютний максимум таблиці лінійних апроксимацій (ТЛА) визначається як [95]

$$\lambda = \max_{\alpha \in F_2^n, \alpha \neq 0, \beta \in F_2^n} \left| \#\left\{ x \left| \bigoplus_{s=0}^{N} (x[s] \cdot \alpha[s]) = \bigoplus_{t=0}^{N} (S(x)[t] \cdot \beta[t]) \right. \right\} \right|, \tag{2.3}$$

де $\mu[s]$ – s-ий біт значення μ.

Ця властивість визначає стійкість шифру до лінійного криптоаналізу. У [95] показано, що повний набір лінійних характеристик, названий лінійним корпусом, повинен бути прийнятий до розгляду для точної оцінки стійкості шифру до атак лінійного криптоаналізу.

Значна частина відомих методик оцінки стійкості блокових шифрів до атак диференційного і лінійного криптоаналізу заснована на диференційних та лінійних властивостях S-блоків, використаних при побудові шифру. Так,

наприклад, в [96] показано, що SPN структура з максимальним дифузійним шаром забезпечує доказову стійкість проти диференційного (лінійного) криптоаналізу: ймовірність кожного диференціалу (лінійного корпуса) обмежена значенням p^n (q^n), де p (q) є максимальною диференційною (лінійною) ймовірністю n активних S-блоків.

Алгебраїчний імунітет характеризує стійкість шифру до алгебраїчної атаки, тобто визначає мінімальну степінь перевизначеної системи рівнянь, за допомогою якої можна описати S-блок. Система рівнянь будується із використанням усіх можливих добутків комбінацій вхідних та вихідних змінних над полем $GF(2)$. Такий опис S-блока дозволяє отримати нижчу степінь термів, ніж при представленні у вигляді набора булевих функцій.

У загальному вигляді для S-блока розмірністю $n \times m$ необхідна кількість рівнянь системи степені d дорівнює [10]

$$r = N_c - Rank(A), \tag{2.4}$$

де

$$N_c = \sum_{i=0}^{d} C_{n+m}^{i}, \tag{2.5}$$

а $Rank(A)$ – ранг двійкової матриці A, що містить усі можливі добутки вхідних та вихідних бітів S-блока. Розмірність такої матриці дорівнює $|A| = \left(2^n\right) \times N_c$.

Мінімальна степінь булевих функцій S-блока. Будь-який S-блок може бути представлений у вигляді набору булевих функцій. Нехай $S = (f_0, f_1, ..., f_{m-1})$ – підстановка розмірністю $n \times m$, де f_i – булева функція від n змінних. Мінімальна степінь S-блока визначається як [97]

$$\deg(S) = \min_{0<j<2^m} \left(\deg(g_j)\right), \tag{2.6}$$

де g_j – множина всіх лінійних комбінацій f_i; $\deg(g_j)$ – степінь алгебраїчної нормальної форми булевої функції.

Відсутність нерухомих точок. Згідно цього критерію для підстановки *S* не повинно існувати таких переходів, що *S*(*x*) = *x*.

У більшості шифрів цей критерій використовується для захисту від статистичних атак.

Нелінійність S-блока також вважається одним із основних критеріїв. У термінах апарата булевих функцій [97] нелінійність підстановки *S* дорівнює

$$NL(S) = \min_{0<j<2^m} \left(NL(g_j)\right), \tag{2.7}$$

де $NL(g_j)$ – мінімальна відстань Гемінга між функцією g_j й усіма афінними функціями над полем $GF(2^n)$.

Однак значення нелінійності однозначно визначається зі значення максимуму таблиці лінійних апроксимацій [97] і для підстановки *S* степені *n* дорівнює

$$NL(S) = 2^{n-1} - \frac{1}{2} \max_{\alpha,\beta \in GF(2^n)} \left|LAT(\alpha,\beta)\right|. \tag{2.8}$$

2.2 Сучасні методи генерації S-блоків

Значна частина існуючих методів генерації S-блоків може бути віднесена до одного з двох типів: алгебраїчних [65, 98, 99] та випадкових. Останні прості у реалізації, проте при доступних для практичної реалізації обчислювальних потужностях дозволяють отримати підстановки з нелінійністю не вище 98.

У табл. 2.1 наведені результати дослідження властивостей випадково згенерованих підстановок степені $n = 2^8$. Вибірка експерименту склала 10 млн. підстановок. У процесі експерименту не було знайдено жодної підстановки з нелінійністю 100. При цьому критерію алгебраїчного імунітету задовольнили всі згенеровані підстановки. У [10] були отримані випадкові підстановки з нелінійністю 100, але для цього був використаний кластер з 4096 комп'ютерів.

Таблиця 2.1

Криптографічні властивості випадково згенерованих підстановок

Критерій	Значення	Відсоток S-блоків, що задовольняють критерію, %
Максимум TPP	8	0,004
Максимум TЛА (нелінійність)	32 (96)	11
	30 (98)	0,15
	28 (100)	0
Мінімальна степінь булевих функцій S-блока	7	30
Алгебраїчний імунітет	3	100

На противагу випадковим методам генерації алгебраїчні методи передбачають побудування S-блока на основі збалансованих булевих функцій з нелінійністю 112. Такі підстановки довгий час вважалися найбільш оптимальними. Блоковий шифр Rijndael/AES [32] також використовує S-блок цього типу. Разом з тим, такі байтові підстановки мають небажану властивість: значення їх алгебраїчного імунітету дорівнює 2, що створює потенційну вразливість шифру до алгебраїчних атак.

Метод модифікованого градієнтного спуску, що розглядається в роботі, належить до так званих евристичних методів генерації S-блоків, і дозволяє забезпечити стійкість як до диференційних та лінійних, так і до алгебраїчних атак. Нижче наводяться основні кроки цього алгоритму [66]:

а) генерація псевдовипадкової підстановки:
   – генерація перестановки S на основі векторної булевої функції, що реалізує степеневе перетворення у кінцевому полі;
   – випадковий обмін місцями N пар значень перестановки S і формування перестановки S';
б) перевірка згенерованої перестановки S' на відповідність сукупності критеріїв.

Даний алгоритм об’єднує переваги алгебраїчних та випадкових методів генерації S-блоків і дозволяє отримати підстановку з алгебраїчним імунітетом 3 та граничною нелінійністю 104. Питання щодо існування підстановок з більш високою нелінійністю за умови збереження високого значення алгебраїчного імунітету залишається відкритим.

Інший евристичний метод, що дозволяє отримати S-блоки з нелінійністю 104, був запропонований в [100], і представляє собою комбінацію спеціального генетичного алгоритму та пошуку по дереву. Однак автор не надає ніякої інформації про значення інших показників, отриманих ним підстановок.

У [101] вперше пропонується метод генерації байтових підстановок з нелінійністю 106 й 108 та алгебраїчним імунітетом 3, що базується на використанні напівбайтових підстановок. Максимум TPP цих підстановок дорівнює 6.

Зазначимо, що блоковий симетричний шифр «Калина» [8] та функція гешування «Купина» [102], представлені у нових українських стандартах, використовують S-блоки з кращими відомими на момент публікації криптографічними показниками. У табл. 2.2 представлені показники даних підстановок.

Описаний модифікований метод градієнтного спуску є найбільш ефективним методом генерації оптимальних S-блоків на момент публікації результатів [103]. Тим не менше, метод може бути додатково оптимізований з метою підвищення швидкодії при генерації підстановок на персональному комп’ютері.

Таблиця 2.2

Криптографічні характеристики підстановок з алгоритмів «Калина» та «Купина»

Характеристика	Значення
Максимум TPP	8
Максимум ТЛА	24
Мінімальна степінь булевих функцій S-блока	7
Нелінійність	104
Алгебраїчний імунітет	3
Відсутність нерухомих точок	Так

2.3 Удосконалення методу генерації S-блоків

У якості вхідних параметрів метод генерації S-блоків приймає наступні [66]:

– векторна булева функція $F(x)$ (з нелінійністю 112 та максимумом таблиці розподілу різниць, рівним 4);

– число випадкових пар значень N, що підлягають обміну.

У якості векторної булевої функції пропонується використовувати функцію виду $F(x) = x^d$. Для отримання можливих значень степені d скористаємося наступною формулою [97]:

$$d = (2^n - 1) - 2^i,\ i = 0,\ldots,7. \qquad (2.9)$$

У табл. 2.3 представлені векторні булеві функції, можливі для використання в алгоритмі генерації S-блоків з $n = 2^8$.

Значення $N = 22$, при якому з найбільшою ймовірністю досягаються всі необхідні властивості S-блока, було отримане в [10].

Таблиця 2.3

Перелік можливих для використання векторних булевих функцій

i	$F(x)$
0	x^{127}
1	x^{191}
2	x^{223}
3	x^{239}
4	x^{247}
5	x^{251}
6	x^{253}
7	x^{254}

Для руйнування циклової структури підстановки (усунення нерухомих точок та коротких циклів) застосовувалася наступна ЧРА-еквівалентність (частково розширена афінна еквівалентність):

$$F(x) = M_1 \cdot G(M_2 \cdot x \oplus V_2) \oplus V_1. \qquad (2.10)$$

Основна частина обчислювальних ресурсів витрачається на другому етапі пошуку S-блоків – перевірці підстановок на відповідність критеріям відбору. Оптимізація цього етапу дозволить значно зменшити час генерації S-блоків.

Критерії відбору підстановок є частково взаємозалежними, тому, змінюючи порядок їх застосування у процесі перевірки підстановки, можна суттєво скоротити час генерації S-блока. Розглянемо принцип знаходження найбільш оптимального порядку застосування критеріїв.

Нехай задано k критеріїв відбору підстановок $\xi_0, \xi_1, ..., \xi_{k-1}$. Тоді число можливих комбінацій k критеріїв, що задають порядок їх застосування, дорівнює $k!$.

Нехай F_σ, де $\sigma \in [0; k!)$, є комбінацією критеріїв наступного виду:

$$F_\sigma = \xi_{\theta_{k-1}(\sigma)} \circ \xi_{\theta_{k-2}(\sigma)} \circ \ldots \circ \xi_{\theta_i(\sigma)} \circ \ldots \circ \xi_{\theta_0(\sigma)}, \tag{2.11}$$

де $\theta_i(\sigma) \in [0;k) -$ функція, що задає критерій для *i*-ої позиції у комбінації F_σ.

Введемо деяку функцію $T(F_\sigma)$, значенням якої буде час перевірки однієї підстановки з використанням комбінації критеріїв F_σ. Тоді задача мінімізації часу перевірки підстановки на відповідність *k* критеріям зводиться до знаходження $t_{\min}$:

$$t_{\min} = \min_{0 \le \sigma < 2^k!} (T(F_\sigma)). \tag{2.12}$$

Комбінація критеріїв F_σ, що відповідає значенню $t_{\min}$, і буде найбільш оптимальною.

Визначимо тепер аналітичний вираз для знаходження значень функції $T(F_\sigma)$. Для цього введемо наступні фактори, що впливають на час перевірки підстановки:

– p_i – ймовірність того, що підстановка задовольняє *i*-ому критерію;

– v_i – час перевірки однієї підстановки на відповідність *i*-ому критерію.

Тут індекс *i* позначає порядковий номер критерію в конкретній комбінації F_σ. Застосування критеріїв здійснюється справа наліво.

Значення факторів знаходяться експериментально, оскільки способи їх аналітичного отримання на поточний момент невідомі.

Використовуючи введені фактори, отримуємо наступний вираз для знаходження $T(F_\sigma)$:

$$T(F_\sigma) = \sum_{i=0}^{k-1} (\varphi_i \cdot v_i), \tag{2.13}$$

де $\varphi_0 = 1, \varphi_i = \varphi_{i-1} \cdot p_{i-1}, i = 1 \ldots k-1.$

Таким чином, мінімізація функції $T(F_\sigma)$ дозволяє отримати найбільш оптимальний порядок застосування критеріїв для генерації S-блоків.

2.4 Результати застосування запропонованого підходу

2.4.1 Порівняння теоретичних та емпіричних результатів

Розглянемо застосування запропонованого підходу для генерації S-блоків степені $n = 2^8$ на прикладі наступних чотирьох критеріїв ($k = 4$):

– максимум ТРР $a = 8$;

– максимум ТЛА $b = 26$;

– мінімальна степінь булевих функцій $c = 7$;

– алгебраїчний імунітет $d = 3$.

Генерація підстановок здійснювалася на основі векторної булевої функції $F(x) = x^{254}$.

У табл. 2.4 наведені експериментально отримані значення факторів p та v.

Таблиця 2.4

Значення факторів для чотирьох критеріїв

Критерій	Фактор p	Фактор v, сек
a	0,66	0,0003
b	0,1	0,0017
c	0,3	0,0018
d	0,6	0,0067

Далі, відповідно до формули (2.13), були розраховані значення функції $T(F_\sigma)$ для комбінацій критеріїв $F_\sigma \in [0;24)$. Розраховані значення наведені в табл. 2.5.

Відповідно до табл. 2.5, мінімальне значення функції $t_{\min} = 0{,}0016735$ отримано для комбінації критеріїв $d \circ c \circ b \circ a$.

Таблиця 2.5

Розраховані значення функції $T(F_\sigma)$

Номер комбінації критеріїв	Порядок застосування критеріїв	Значення функції $T(F_\sigma)$	Номер комбінації критеріїв	Порядок застосування критеріїв	Значення функції $T(F_\sigma)$
0	$a \circ b \circ c \circ d$	0,0080914	12	$c \circ a \circ b \circ d$	0,0078093
1	$a \circ b \circ d \circ c$	0,0041214	13	$c \circ a \circ d \circ b$	0,0024593
2	$a \circ c \circ b \circ d$	0,0078334	14	$c \circ b \circ a \circ d$	0,0076245
3	$a \circ c \circ d \circ b$	0,0024834	15	$c \circ b \circ d \circ a$	0,0054665
4	$a \circ d \circ b \circ c$	0,0025164	16	$c \circ d \circ a \circ b$	0,0022435
5	$a \circ d \circ c \circ b$	0,0020864	17	$c \circ d \circ b \circ a$	0,0019355
6	$b \circ a \circ c \circ d$	0,0080360	18	$d \circ a \circ b \circ c$	0,0024517
7	$b \circ a \circ d \circ c$	0,0040660	19	$d \circ a \circ c \circ b$	0,0020217
8	$b \circ c \circ a \circ d$	0,0077948	20	$d \circ b \circ a \circ c$	0,0023593
9	$b \circ c \circ d \circ a$	0,0056368	21	$d \circ b \circ c \circ a$	0,0019573
10	$b \circ d \circ a \circ c$	0,0034186	22	$d \circ c \circ a \circ b$	0,0019815
11	$b \circ d \circ c \circ a$	0,0030166	23	$d \circ c \circ b \circ a$	0,0016735

Представлені в табл. 2.5 значення – це час, що витрачається на перевірку однієї підстановки. Експерименти показали, що для генерації одного S-блока, що задовольняє чотирьом критеріям, в середньому необхідно перевірити 102 підстановки. Таким чином, час генерації підстановки можна розрахувати як $T_{теор}(F_\sigma) = T(F_\sigma) \cdot 102$. Також були отримані експериментальні значення часу генерації S-блока для всіх комбінацій критеріїв. На рисунку 2.1 представлені графіки функцій $T_{теор}(F_\sigma)$ (суцільна крива) та $T_{експ}(F_\sigma)$ (пунктирна крива).

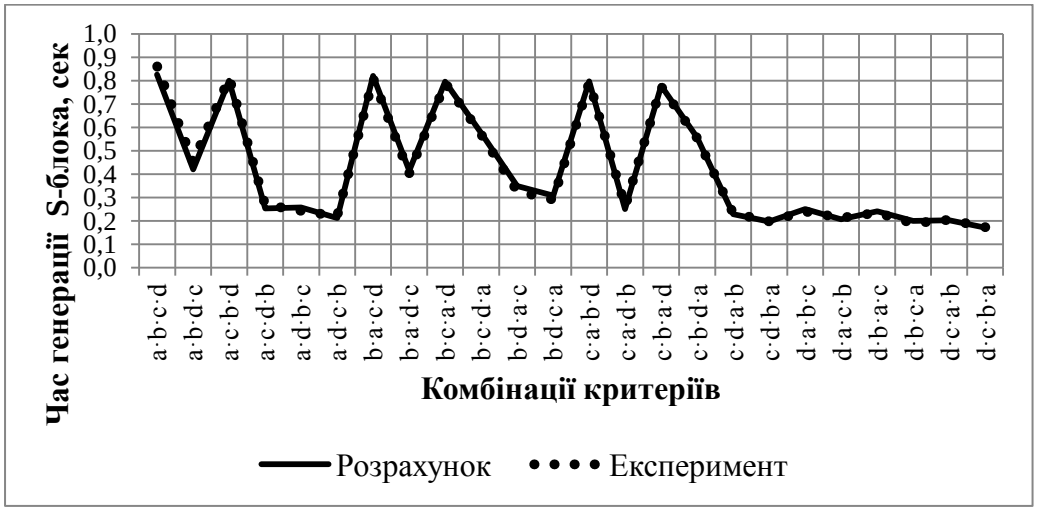

Рис. 2.1 Графіки функцій $T_{теор}(F_\sigma)$ та $T_{експ}(F_\sigma)$

2.4.2 Аналіз ефективності запропонованого методу

Далі найбільш ефективний порядок застосування критеріїв був визначений для генерації оптимальних підстановок з наступними характеристиками:

– максимум ТРР $a = 8$;

– максимум ТЛА $b = 24$ (нелінійність підвищена до 104);

– мінімальна степінь булевих функцій $c = 7$;

– алгебраїчний імунітет $d = 3$;

– відсутність коротких циклів (довжиною 3 та менше).

Експерименти показали, що ймовірність того, що підстановка має нелінійність 104, дорівнює 0,0000007.

Відсутність коротких циклів досягається шляхом застосування ЧРА-еквівалентності до отриманого S-блока, тому немає необхідності включати цей критерій до списку критеріїв при розрахунку мінімального часу перевірки.

В табл. 2.6 представлені розраховані значення функції $T(F_\sigma)$. Мінімальне значення $t_{\min} = 0{,}001422$ при комбінаціях критеріїв $c \circ d \circ b \circ a$ та $d \circ c \circ b \circ a$.

В табл. 2.7 – 2.8 наведені приклади згенерованих оптимальних S-блоків.

Таблиця 2.6

Значення функції $T(F_\sigma)$ для S-блоків з нелінійністю 104

Номер комбінації критеріїв	Порядок застосування критеріїв	Значення функції $T(F_\sigma)$	Номер комбінації критеріїв	Порядок застосування критеріїв	Значення функції $T(F_\sigma)$
0	$a \circ b \circ c \circ d$	0,0080860	12	$c \circ a \circ b \circ d$	0,0077200
1	$a \circ b \circ d \circ c$	0,0041160	13	$c \circ a \circ d \circ b$	0,0017000
2	$a \circ c \circ b \circ d$	0,0077200	14	$c \circ b \circ a \circ d$	0,0075532
3	$a \circ c \circ d \circ b$	0,0017000	15	$c \circ b \circ d \circ a$	0,0053952
4	$a \circ d \circ b \circ c$	0,0023100	16	$c \circ d \circ a \circ b$	0,0017000
5	$a \circ d \circ c \circ b$	0,0017000	17	$c \circ d \circ b \circ a$	0,0014220
6	$b \circ a \circ c \circ d$	0,0080360	18	$d \circ a \circ b \circ c$	0,0023100
7	$b \circ a \circ d \circ c$	0,0040660	19	$d \circ a \circ c \circ b$	0,0017000
8	$b \circ c \circ a \circ d$	0,0077948	20	$d \circ b \circ a \circ c$	0,0022266
9	$b \circ c \circ d \circ a$	0,0056368	21	$d \circ b \circ c \circ a$	0,0018246
10	$b \circ d \circ a \circ c$	0,0034186	22	$d \circ c \circ a \circ b$	0,0017000
11	$b \circ d \circ c \circ a$	0,0030166	23	$d \circ c \circ b \circ a$	0,0014220

Таблиця 2.7

Приклад оптимального S-блока на основі векторної булевої функції x^{191} (шістнадцятькова нотація)

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
0	5c	06	e1	54	39	4c	9b	08	f4	32	c1	22	7a	0b	81	47
1	79	e2	a5	10	76	e4	86	c0	2a	75	1c	77	f0	1e	3d	a4
2	91	19	34	95	7d	85	b8	c7	a7	3b	e8	cd	4d	b4	fc	bb
3	7c	17	42	98	31	ec	bc	f5	5d	fb	02	4f	4e	78	e6	94
4	e7	30	2c	0d	e0	f3	bf	fa	db	ba	15	1d	40	18	ca	b1
5	f9	03	d0	d8	ad	44	3a	72	a2	73	df	66	01	fe	be	fd
6	ef	e3	a9	cb	28	b2	d5	2b	23	2e	99	5e	2d	5b	c8	48
7	6e	8f	f6	c5	d7	cc	82	65	14	67	c3	1f	26	e9	8c	97
8	a1	71	8d	ae	1b	ee	c6	68	84	b9	60	87	5f	9c	49	6b
9	b6	b0	6f	ff	d9	b7	38	cf	a0	eb	8b	4a	f7	3f	3e	da
a	80	b5	59	0c	6a	1a	96	d2	89	8e	9e	d4	24	25	16	ab
b	a6	9d	33	70	05	74	63	7b	5a	36	6d	4b	ea	dd	f8	ac
c	21	2f	69	53	51	f2	7f	92	9a	6c	43	00	d6	50	a3	46
d	c9	29	90	37	c2	41	7e	09	55	58	20	aa	27	e5	88	64
e	61	f1	d3	af	d1	11	9f	0a	0e	13	12	3c	dc	35	ed	45
f	93	b3	c4	bd	57	62	52	8a	a8	0f	04	ce	de	07	83	56

Таблиця 2.8

Приклад оптимального S-блока на основі векторної булевої функції x^{254} (шістнадцятькова нотація)

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
0	1a	a8	96	a1	a6	97	80	26	c1	f2	32	7f	8b	c9	f0	c3
1	64	79	27	10	43	4c	6c	9b	c4	ac	d8	ea	b2	9e	d5	8e
2	7d	02	c7	0e	17	83	cb	07	61	e0	84	fa	3e	03	7a	24
3	be	8c	19	6f	1d	f7	b8	68	b3	e6	db	78	d1	cd	0a	a7
4	a3	b4	f1	fc	3f	5d	57	4f	42	8d	ca	71	5f	ab	66	d9
5	a0	72	16	ad	9c	2c	49	30	bb	99	31	ce	34	3c	fe	d3
6	18	d0	ef	cf	82	36	cc	6d	d6	b7	c6	5c	58	86	20	e4
7	75	7e	87	41	8a	53	1f	21	63	67	74	37	0c	2d	91	48
8	54	df	38	73	44	b1	ae	40	2a	62	fb	c5	f5	1c	4d	af
9	45	70	dc	95	04	ec	0f	bc	fd	6b	0d	a2	2e	93	3a	eb
a	59	aa	c0	55	06	ed	e1	50	4b	d7	5a	65	4a	e3	25	a9
b	c8	b5	5b	76	47	05	14	22	2f	81	9a	0b	c2	77	09	35
c	90	1e	e9	3d	7b	f4	51	92	29	33	b0	9d	23	d2	12	6a
d	89	2b	d4	28	dd	f6	f8	8f	08	69	39	00	a5	e5	e2	88
e	52	1b	f9	da	bf	b9	f3	60	13	ff	56	7c	de	6e	5e	85
f	3b	9f	e8	11	4e	bd	94	a4	46	ba	ee	15	98	01	b6	e7

Для генерації оптимального S-блока в середньому необхідно перевірити 1 100 000 підстановок. Таким чином, середній час генерації оптимального S-блока дорівнює

$$t_{\min} \cdot 1{,}100{,}000 = 0.001422 \cdot 1{,}100{,}000 = 1564.2 \text{ сек.} \approx 26 \text{ хвилин.}$$

Висновки до розділу 2

1. У результаті виконаних досліджень запропоноване вдосконалення перевірки властивостей підстановок для евристичних методів генерації S-блоків. В основу оптимізації покладено той факт, що критерії відбору підстановок є частково взаємозалежними. Це дозволяє стверджувати, що час перевірки підстановки на відповідність набору критеріїв суттєво залежить від порядку застосування цих критеріїв у процесі перевірки. У зв'язку з цим, запропонований аналітичний вираз для визначення порядку застосування критеріїв, за якого час перевірки підстановки буде мінімальним.

2. Для апробації викладеного підходу в якості методу генерації S-блоків був обраний модифікований алгоритм градієнтного спуску. Для чотирьох обраних критеріїв були розраховані значення часу перевірки підстановки (та, відповідно, часу генерації) для всіх можливих комбінацій критеріїв. Розрахунки показали, що вибір найоптимальнішого порядку застосування критеріїв дозволяє зменшити час генерації S-блока майже в 5 разів (у порівнянні з найгіршим випадком). Аналітично отримані значення були підтверджені й експериментально.

3. Були визначені два найкращих варіанти порядку застосування критеріїв для генерації оптимальних S-блоків степені 2^8 з нелінійністю 104, алгебраїчним імунітетом 3 та δ-рівномірністю 8. Середній час генерації оптимальних підстановок за допомогою програмної реалізації з урахуванням запропонованого вдосконалення склав 30 хвилин (проти 2,5 годин для найгіршого випадку).

4. Представлений підхід дозволяє суттєво оптимізувати перевірку властивостей підстановок, не залежить від самого алгоритму генерації та може бути застосований до вдосконалення будь-якого евристичного методу генерації S-блоків.

Результати досліджень даного розділу наведено в публікаціях здобувача: [103,104].

РОЗДІЛ 3
РОЗРОБКА МЕТОДУ ОЦІНКИ КОЛІЗІЙНИХ ВЛАСТИВОСТЕЙ НЕІН'ЄКТИВНИХ СХЕМ РОЗГОРТАННЯ КЛЮЧІВ СИМЕТРИЧНИХ БЛОКОВИХ ШИФРІВ

3.1 Сучасні вимоги до схем розгортання ключів симетричних блокових шифрів

У процесі розробки блокового шифру, як правило, більше уваги приділяється шифруючому перетворенню, ніж СРК, тому консенсусу щодо необхідних та достатніх умов, яким повинна задовольняти СРК, на сьогоднішній день не досягнуто [105]. Багато з існуючих правил проєктування СРК мають різні недоліки [105]:

- є далекими від практичного застосування;
- є занадто слабкими з точки зору безпеки;
- фокусуються лише на певних типах атак і є однобічними;
- є емпіричними і не мають достатнього обґрунтування.

Загальними принципами побудування СРК є відсутність слабких, напівслабких та еквівалентних ключів. Застосування циклових констант необхідно для попередження симетрії шифру, яка призводить до можливості реалізації слайд-атак [9].

Класичним підходом до проєктування СРК вважається застосування бієктивного перетворення для відображення ключа шифрування у послідовність циклових ключів:

$$f : \left\{K^{(i)}\right\} \leftrightarrow \left\{K_1^{(i)}, K_2^{(i)}, ..., K_t^{(i)}\right\}. \qquad (3.1)$$

Перші СРК були дуже простими і включали, наприклад, просту перестановку бітів ключа шифрування (DES [22], IDEA [55]) або пряме чи рекурсивне лінійне перетворення з ключа шифрування [105]. Застосування простої бієктивної функції у складі СРК забезпечує наступні властивості:

- компактна програмна або апаратна реалізація;
- достатньо висока швидкодія перетворень;
- відсутність еквівалентних ключів шифрування, тобто таких, що породжують ідентичні послідовності циклових ключів.

При очевидних перевагах, подібні СРК мають і суттєві недоліки:

- відсутність властивості односпрямованості (складність відновлення ключа шифрування при знанні одного або декількох підключів є не вищою за поліноміальну), що робить шифр більш уразливим до криптоаналітичних атак таких, як атаки на реалізацію [106] та ін;
- відсутність нелінійних операцій знижує стійкість СРК до атак на зв’язаних ключах [69];
- деякі ітеративні шифри з дуже простими СРК навіть при повному наборі циклів не досягають рівномірного розподілу ймовірностей диференціалів та лінійних корпусів.

У зв’язку з існуванням описаних вразливостей, прості СРК почали ускладнювати. З’явилося поняття «сильної» СРК, метою якої є усунення будь-якої слабкості, яка гіпотетично або практично може бути використана для атаки на блоковий шифр [107]. Показано [108], що добре спроєктовані шифри зі складними СРК є більш стійкими до атак диференційного та лінійного криптоаналізу та досягають рівномірного розподілу швидше, ніж шифри з простими СРК.

У [109] стверджується, що «сильна» СРК повинна мати наступні загальні властивості, які можуть бути досягнуті водночас:

- односпрямована функція, стійка до колізій (функція, яку неможливо інвертувати);

- мінімальна взаємна інформація (між всіма бітами підключа та бітами майстер-ключа);
- ефективна реалізація.

Доцільними можна також вважати вимоги, висунуті до СРК при розробці шифру «Калина» [9]:

- нелінійна залежність кожного біта кожного циклового ключа від кожного біта ключа шифрування;
- циклові ключі суттєво відрізняються і мають складну нелінійну залежність;
- захист від відомих криптоаналітичних атак, що орієнтовані на схему розгортання ключів;
- відсутність слабких ключів, при яких погіршуються криптографічні властивості або знижується стійкість перетворення;
- обчислювальна складність формування всіх циклових ключів не перевищує складності зашифрування трьох блоків;
- простота програмної, програмно-апаратної і апаратної реалізації.

У якості прикладу «сильної» СРК розглянемо СРК блокового шифру «Калина» та принципи її побудування. Для шифру «Калина» була розроблена односпрямована СРК, що забезпечує неможливість відновлення ключа шифрування при знанні одного або декількох підключів. Властивість односпрямованості СРК шифру забезпечується за рахунок застосування двох ключів у алгоритмі формування циклових ключів: ключа шифрування та обчисленого на його основі допоміжного ключа [9].

Циклові ключі з парними індексами формуються на основі ключа шифрування, допоміжного ключа та константи, що залежить від номеру циклу; циклові ключі з непарними індексами формуються шляхом циклічного зсуву парних [9]. Таким чином забезпечується односпрямованість СРК, унікальність кожного сформованого значення і додатково покращуються криптографічні властивості як схеми розгортання, так і всього шифру в цілому [9].

Односпрямовані СРК застосовуються і в інших відомих блокових шифрах таких, як FOX [11], Twofish [12] та ін.

Особливістю односпрямованих СРК є те, що вони є неін'єктивними, тобто теоретично припускається, що потужність множини послідовностей циклових ключів є меншою за потужність множини ключів шифрування:

$$\#\left\{K^{(i)}\right\} \geq \#\left\{K_1^{(i)}, K_2^{(i)}, ..., K_t^{(i)}\right\}. \qquad (3.2)$$

Якщо ймовірність співпадіння потужностей цих двох множин $P_\theta < 1$, це означає, що СРК не задовольняє вимозі відсутності еквівалентних ключів, що створює потенційну вразливість блокового шифру до певних видів атак. *Метою даного розділу* є розробка методу оцінки ймовірності співпадіння потужностей множини послідовностей циклових ключів, які формуються неін'єктивною СРК, і множини ключів шифрування.

3.2 Метод оцінки властивостей неін'єктивних схем розгортання ключів блокових шифрів

Розв'язання задачі оцінки потужності множини послідовностей циклових ключів було започатковано у [50]. Запропонуємо новий підхід до її вирішення, який дозволяє отримати більш точні оцінки та довести, що для неін'єктивних схем розгортання циклових ключів потужність множини циклових ключів не зменшується у порівнянні з ін'єктивними схемами.

Введемо математичну модель оцінки ймовірності співпадіння потужностей множини послідовностей циклових ключів, які формуються неін'єктивною СРК, і множини ключів шифрування. Математична модель використовує припущення, що схема розгортання ключів представляє собою випадкове відображення (див. визначення 3.1), що випливає із конструкції та

підтверджується результатами статистичного тестування. Тоді справедливою є теорема 3.1.

Визначення 3.1 [110]. Випадковим відображенням σ множини $X = \{1, 2, ..., n\}$ в себе є випадкова величина, що приймає значення з множини $\sum_n$ всіх однозначних відображень множини X в себе.

Теорема 3.1 (про ймовірність співпадіння потужностей множини послідовностей циклових ключів, які формуються неін'єктивною СРК, і множини ключів шифрування).

Нехай τ – неін'єктивна схема розгортання ключів, що реалізує випадкове відображення та має наступні параметри:

k – довжина ключа шифрування в бітах;

l – довжина циклового ключа в бітах;

t – кількість циклових ключів;

$K = 2^k$ – потужність множини ключів шифрування;

$L = 2^{tl}$ – потужність множини послідовностей циклових ключів.

Тоді ймовірність співпадіння потужностей множини послідовностей циклових ключів, які формуються неін'єктивною СРК, і множини ключів шифрування для τ обчислюється як

$$P_\theta = \frac{L!}{(L-K)!L^K} \tag{3.3}$$

і практично дорівнює

$$P_\theta = \lim_{\substack{L\to\infty \\ K\to\infty}} \frac{L!}{(L-K)!L^K} = 1. \tag{3.4}$$

Доведення.

Нехай задано множину ключів шифрування $\Psi = \left\{ K^{(i)} \mid i = 0,1,...,2^k - 1 \right\}$ з потужністю $K = |\Psi|$ та множину послідовностей циклових ключів $\Lambda = \left\{ L^{(i)} | i = 0,1,...,2^{tl} - 1 \right\}$ з потужністю $L = |\Lambda|$.

Послідовність циклових ключів, що формується СРК, має вигляд $L^{(i)} = (L_0^{(i)}, L_1^{(i)}, ..., L_{t-1}^{(i)})$.

Генерація кожної послідовності ключів $L^{(i)}$ виконується за допомогою випадкової функції $f : \Psi \to \Lambda$.

Для кожного ключа шифрування $K^{(i)}$ формується послідовність циклових ключів, тобто всього з множини Λ обирається K послідовностей. Першу послідовність $L^{(0)} = (L_0^{(0)}, L_1^{(0)}, ..., L_{t-1}^{(0)})$ можна обрати L способами, другу $L^{(1)} = (L_0^{(1)}, L_1^{(1)}, ..., L_{t-1}^{(1)})$ (так, щоб вона не співпадала з першою) – $(L-1)$ способами, останню $L^{(K)} = (L_0^{(K)}, L_1^{(K)}, ..., L_{t-1}^{(K)})$ – $(L-(K-1))$ способами.

Таким чином, загальна кількість варіантів K послідовностей, що не повторюються, дорівнює $L \cdot (L-1) \cdot ... \cdot (L-(K-1)) = \dfrac{L!}{(L-K)!}$.

Число всіх можливих варіантів послідовностей циклових ключів дорівнює L^K. Тоді ймовірність співпадіння потужностей множини послідовностей циклових ключів, які формуються неін'єктивною СРК, і множини ключів шифрування визначається як

$$P_\theta = \frac{L!}{(L-K)!L^K}. \qquad (3.5)$$

Відмітимо, що в реальних СРК l є кратним k: $l = \dfrac{k}{c}$, де $c = const$ (зазвичай, $c = 1$ або $c = 2$).

Покладемо $x = 2^k$ і перепишемо формулу (3.5) у наступному вигляді:

$$P_\theta = \frac{L!}{(L-K)!L^K} = \frac{2^{\frac{tk}{c}}!}{\left(2^{\frac{tk}{c}} - 2^k\right)!\left(2^{\frac{tk}{c}}\right)^{2^k}} = \frac{x^{\frac{t}{c}}!}{\left(x^{\frac{t}{c}} - x\right)!x^{\frac{tx}{c}}}. \qquad (3.6)$$

Оскільки, значення x є дуже великим, можемо покласти $x \to \infty$ і знайти границю для P_θ:

$$P_\theta = \lim_{x\to\infty} \frac{x^{\frac{t}{c}}!}{\left(x^{\frac{t}{c}} - x\right)!x^{\frac{tx}{c}}} =$$

$$= \lim_{x\to\infty} \frac{x^{\frac{t}{c}}\left(x^{\frac{t}{c}} - 1\right)\left(x^{\frac{t}{c}} - 2\right)\ldots\left(x^{\frac{t}{c}} - x + 1\right)\left(x^{\frac{t}{c}} - x\right)!}{\left(x^{\frac{t}{c}} - x\right)!x^{\frac{tx}{c}}} =$$

$$= \lim_{x\to\infty} \frac{x^{\frac{t}{c}}\left(x^{\frac{t}{c}} - 1\right)\left(x^{\frac{t}{c}} - 2\right)\ldots\left(x^{\frac{t}{c}} - x + 1\right)}{x^{\frac{tx}{c}}} =$$

$$= \lim_{x\to\infty} \frac{x^{\frac{t}{c}}x^{\frac{t}{c}}\left(1 - \frac{1}{x^{\frac{t}{c}}}\right)x^{\frac{t}{c}}\left(1 - \frac{2}{x^{\frac{t}{c}}}\right)\ldots x^{\frac{t}{c}}\left(1 - \frac{1}{x^{\frac{t}{c}-1}} + \frac{1}{x^{\frac{t}{c}}}\right)}{x^{\frac{tx}{c}}} =$$

$$= \lim_{x\to\infty}\left(1 - \frac{1}{x^{\frac{t}{c}}}\right)\left(1 - \frac{2}{x^{\frac{t}{c}}}\right)\ldots\left(1 - \frac{1}{x^{\frac{t}{c}-1}} + \frac{1}{x^{\frac{t}{c}}}\right) = 1.$$

Доведення закінчено.

3.3 Результати оцінки властивостей неін'єктивних схем розгортання ключів блокових шифрів

В таблицях 3.1–3.2 представлені результати розрахунків за формулою (3.3) для $l = k$ та $l = \frac{k}{2}$ відповідно.

Таблиця 3.1

Результати розрахунків за формулою при $l = k$

t	P_θ
k = 4, l = 4	
2	0,6197211
3	0,9710922
4	0,9981705
5	0,9998856
6	0,9999928
7	0,9999996
8	0,99999997
9	0,999999998
10	0,99999999989
k = 8, l = 8	
2	0,6073225
3	0,9980564
4	0,9999924
5	0,99999997
6	0,99999999988
k = 16, l = 16	
2	0,6065337
3	0,9999924

Як видно з таблиці 3.1, вже при $K = 2^4$ та $L = 2^{40}$ ймовірність співпадіння потужностей множини послідовностей циклових ключів, які формуються неін'єктивною СРК, і множини ключів шифрування дорівнює 0,99999999989. Зі зростанням довжин ключа шифрування та циклового ключа значення P_θ зростає. Для повномасштабного шифру, відповідно до теореми 3.1, $P_\theta = 1$. Це означає, що складність атак переборного типу на неін'єктивні СРК практично дорівнює складності атак на ін'єктивні СРК.

Таблиця 3.2

Результати розрахунків за формулою при $l = \frac{k}{2}$

t	P_θ
$k = 4, l = 2$	
3	0,129067534793
4	0,619721146136
5	0,888889811578
6	0,971092240239
7	0,992700248983
8	0,998170476853
9	0,999542332241
10	0,999885565688
11	0,999971390067
12	0,999992847461
13	0,999998211862
14	0,999999076128
15	0,999999769032
16	0,999999942258
17	0,999999985565
18	0,999999996391
19	0,999999999098
20	0,999999999774

Продовження таблиці 3.2

t	P_θ
$k = 8, l = 4$	
3	0,000291692740
4	0,607322476268
5	0,969349104961
6	0,998056365958
7	0,999878413801
8	0,999984771129
9	0,999999048189
10	0,999999940512
11	0,999999996282
12	0,999999999768
13	0,999999999985

Відповідно, обернена величина $Q_\theta = 1 - P_\theta$ визначає ймовірність колізії двох послідовностей циклових ключів. На рис. 3.1 представлено графік, що демонструє експоненційний характер залежності між кількістю циклових ключів та Q_θ.

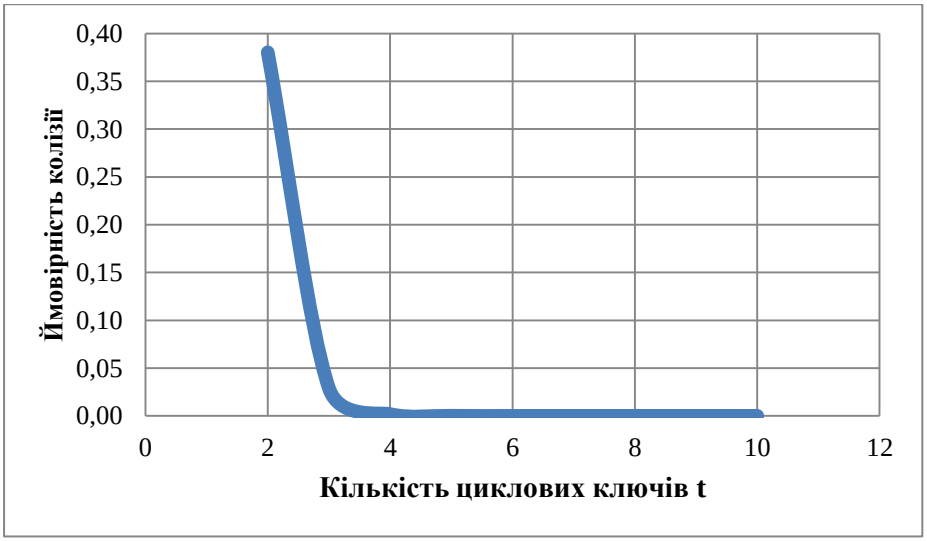

Рис. 3.1 Графік залежності між кількістю циклових ключів та ймовірністю колізії Q_θ

Висновки до розділу 3

1. Застосування «сильних» схем розгортання ключів дозволяє усунути вразливості, які теоретично або практично можуть бути використані для атаки на шифр. Крім того, використання сильних СРК покращує характеристики шифру, зокрема диференційні та лінійні. До основних вимог, яким повинна задовольняти «сильна» СРК відносяться односпрямованість, нелінійна залежність між всіма бітами циклових ключів та ключа шифрування і ефективна реалізація.

2. Більшість атак на схеми розгортання ключів не представляють практичної небезпеки, проте вони показують теоретичну слабкість, яка може бути використана за певних обставин. Найбільш небезпечними можна вважати атаку на зв’язаних ключах та атаки на СРК, що не є односпрямованими.

3. У розділі запропонований метод оцінки ймовірності співпадіння потужностей множини послідовностей циклових ключів, які формуються неін’єктивною СРК, і множини ключів шифрування. Показано, що для повномасштабного шифру ця ймовірність практично дорівнює 1.

4. За допомогою запропонованого методу доведено, що складність атак переборного типу на неін’єктивні схеми розгортання ключів практично дорівнює складності атак на ін’єктивні схеми (складність перебірних атак не знижується). При цьому неін’єктивні СРК забезпечують додаткову стійкість до атак на реалізацію та деяких інших криптоаналітичних атак. Таким чином, при побудуванні перспективного симетричного блокового шифру доцільно використовувати саме неін’єктивну схему розгортання циклових ключів.

Результати досліджень даного розділу наведено в публікаціях здобувача: [111-114].

РОЗДІЛ 4
ПОБУДОВА ТА АНАЛІЗ ВЛАСТИВОСТЕЙ ПЕРСПЕКТИВНОГО ПОСТКВАНТОВОГО МАЛОРЕСУРСНОГО БЛОКОВОГО ШИФРУ

4.1 Вимоги до перспективного малоресурсного симетричного блокового шифру

Як видно з огляду, представленого у розділі 1, достатньо багато малоресурсних блокових шифрів мають меншу довжину блока даних та ключа шифрування у порівнянні з традиційними симетричними примітивами. Зменшення розмірів параметрів здійснюється з метою підвищення швидкодії перетворень та зменшення обсягу необхідної для шифрування пам'яті [77]. Відмітимо, що серед сучасних малоресурсних блокових шифрів багато таких, що мають розмір блока, що дорівнює 64 бітам. Застосування цих шифрів у постквантовий період стане неможливим, оскільки складність алгоритму Гровера дорівнює $\sqrt{N}$ і для 64-бітового блока складатиме всього 2^{32} операцій [115].

Таким чином, для малоресурсного блокового шифру, що пропонується для застосування у постквантовий період, необхідно обирати розмір блока та довжину ключа шифрування, що дорівнюють не менше, ніж 256 бітам. Доцільно також передбачити можливість використання 512-бітових параметрів.

Враховуючи вимоги до малоресурсних алгоритмів, запропонованих NIST [78], та можливу появу квантових комп'ютерів, сформулюємо основні вимоги до перспективного малоресурсного симетричного блокового шифру.

а) Високий рівень криптографічної стійкості проти перебірних атак, що обумовлює необхідність підтримки ключів довжиною 256 та 512 бітів.

б) Застосування «сильної» схеми розгортання ключів (СРК) з метою захисту від криптоаналітичних атак на СРК.

в) Висока швидкодія на різних програмно-апаратних платформах (в тому числі, мобільних).

г) Компактна програмна реалізація як для стаціонарних систем (робочі станції, сервери), так і для мобільних платформ (смартфони, планшети).

д) Наявність варіантів алгоритму, оптимізованих як під 64-бітові, так і під 32-бітові системи.

е) Мінімальний об’єм пам’яті для швидкодіючої реалізації (компактний код та відсутність таблиць передобчислень).

ж) Постійний час шифрування блока на сучасних процесорах незалежно від параметрів, що обробляються – для захисту від атак по побічним каналам.

4.2 Синтез компонентів малоресурсного симетричного блокового шифру

4.2.1 Високорівнева конструкція блокового шифру

Очевидно, що вибір високорівневої конструкції впливає на компактність реалізації та швидкодію блокового шифру. Як видно з аналізу, представленого у розділі 1, найбільш розповсюдженими та добре вивченими високорівневими конструкціями на сьогоднішній день є SPN-структура та мережа Фейстеля. Кожна з цих двох конструкцій має свої переваги та є більш ефективною для конкретних застосувань.

Першою суттєвою перевагою мережі Фейстеля є ідентичність процедур зашифрування та розшифрування, що вдвічі зменшує об’єм програмного коду, який реалізує логіку шифруючого перетворення. У результаті отримуємо більш компактну реалізацію.

По-друге, мережа Фейстеля дозволяє реалізувати обробку блока за допомогою перетворень з розміром входу, меншим за сам розмір блока (вхід циклової функції дорівнює ½ або ¼ блока).

У випадку SPN-структури, обернена функція може відрізнятися від прямої функції, що ускладнює апаратну реалізацію та компактну реалізацію на процесорах смарт-карток [116].

Як зазначено у [116], SPN-структура більш схильна до розпаралелювання і на мікропроцесорі з великою кількістю виконавчих пристроїв може бути обчислена швидше, ніж мережа Фейстеля. Однак, ця перевага не відноситься до смарт-карток і т.п., оскільки більшість із них містять мікропроцесори з малою кількість виконавчих пристроїв. Оскільки, перспективний блоковий шифр орієнтований на застосування у пристроях з обмеженою кількістю споживання енергії, властивий SPN-структурі паралелізм не є перевагою при виборі високорівневої конструкції для даного шифру.

Зазначимо також, що більшість сучасних малоресурсних блокових шифрів засновані на мережі Фейстеля (SPECK [60], SIMON [60], TEA [89], XTEA [90] та ін.).

Таким чином, враховуючи компактність реалізації, швидкодію та особливості реалізації у пристроях з обмеженою кількістю споживання енергії, така високорівнева конструкція як мережа Фейстеля найбільше підходить для застосування у перспективному малоресурсному блоковому шифрі.

4.2.2 Циклова функція

Класичним підходом до побудування циклової функції є послідовне застосування нелінійного та лінійного перетворень. У якості нелінійного перетворення використовуються S-блоки з криптографічними показниками, необхідними для забезпечення стійкості блокового шифру до атак диференційного, лінійного та алгебраїчного криптоаналізу [11, 32, 117]. Лінійне перетворення обирається з міркувань забезпечення активізації якомога більшого числа бітів, що поступають на вхід S-блоків [74]. Безперечною перевагою цього підходу є можливість обчислення верхньої границі ймовірності (r-1)-циклової диференційної (лінійної) характеристики шифру за

рахунок прозорого математичного апарату, на якому базуються методи побудування S-блоків та лінійного перетворення [70].

Недоліками алгоритмів, побудованих за допомогою описаного підходу (таких, як AES), є недостатньо висока швидкодія та складність реалізації для застосування у пристроях з обмеженою кількістю споживання енергії. Крім того, швидкодіюча реалізація цих шифрів передбачає використання таблиць передобчислень, що потребує використання значного обсягу пам’яті, якого смарт-картки та подібні пристрої можуть не мати.

Як було зазначено вище, альтернативним підходом, який знайшов широке застосування у малоресурсних симетричних, стало ARX-перетворення, яке передбачає послідовне застосування операцій модульного додавання, циклічного зсуву та додавання за модулем 2 (XOR). В архітектурі ARX немає чіткого поділу на лінійну та нелінійну частини. Нелінійна операція модульного додавання та лінійні операції – циклічний зсув та XOR повторюються у певному порядку, формуючи циклову функцію.

Популярність ARX-перетворення обумовлена простотою, високою швидкодією та легкістю масштабування використовуваних операцій. ARX-подібні шифри мають компактну реалізацію і є ефективними як на програмних, так і на апаратних платформах. Такі шифри, як правило, оперують не байтами, а 16- або 32-бітними словами, що прискорює виконання операцій на сучасних процесорах.

Однак, на відміну від шифрів, заснованих на S-блоках, для ARX-подібних шифрів не існує загальної методики обґрунтування стійкості до диференційного та лінійного криптоаналізу. Велика кількість повторень простих операцій робить дуже складним представлення математичного обґрунтування навіть для окремого алгоритму. Тим не менше, існуючі евристичні алгоритми пошуку диференційних характеристик [15,118] дозволяють отримати достатньо об’єктивну оцінку щодо стійкості алгоритму до диференційного криптоаналізу.

Забезпечення компактності та швидкодії можливе лише за умови застосування простих операцій, тому, очевидно, що орієнтація розробників малоресурсних примітивів на ARX-архітектуру є цілком обґрунтованою.

В основу циклової функції перспективного постквантового малоресурсного блокового шифру пропонується покласти ARX-перетворення схоже на те, що застосовується у потоковому шифрі ChaCha [59]. Передбачається, що 256-бітовий та 512-бітовий шифри оперуватимуть 32- та 64-бітовими словами відповідно, що дозволить забезпечити високу продуктивність на різних платформах.

Циклова функція складатиметься з багатократного застосування трьох простих операцій до різних 32- або 64-бітових слів: додавання за модулем 2^{32} (2^{64}), побітового додавання та циклічного зсуву. Кожне повторення операцій можна представити у вигляді елементарного перетворення, зображеного на рисунку 4.1.

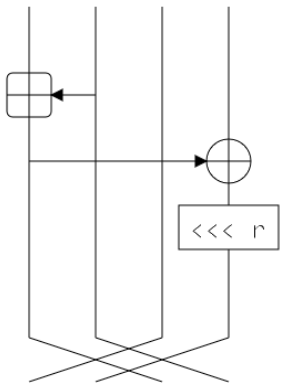

Рис. 4.1 Елементарне перетворення перспективного блокового шифру

Такий вигляд елементарного перетворення (перше слово складається з другим, а результат подається на вхід двійкового суматора разом з четвертим словом і т.д.) забезпечує хороше перемішування та розповсюдження активних бітів по різним словам.

Пропонується, щоб один цикл шифруючого перетворення складався з восьми послідовних застосувань елементарного перетворення, що забезпечить

оптимальне співвідношення кількості взаємозалежних бітів на вході та виході до кількості операцій процесору (що підтверджується експериментальним тестуванням статистичних та швидкісних характеристик циклової функції).

4.2.3 Схема розгортання ключів

Для формування ключового розкладу шифру пропонується використовувати односпрямовану (неін'єктивну) СРК, подібну до тієї, що використовується в алгоритмі «Калина» [8]. Обґрунтування використання односпрямованих СРК у блокових шифрах представлено у розділі 3, де, зокрема, доведено, що складність перебірних атак на неін'єктивні СРК у порівнянні з ін'єктивними схемами не знижується, при цьому неін'єктивні схеми забезпечують додатковий захист від атак на реалізацію та інших криптоаналітичних атак.

Ще однією перевагою обраної схеми є те, що в алгоритмі СРК використовуються функції з шифруючого перетворення, що робить реалізацію більш компактною.

4.3 Постквантовий малоресурсний симетричний блоковий шифр «Кипарис»

4.3.1 Параметри алгоритму

Алгоритм шифрування «Кипарис» виконує перетворення блоків даних розміром l бітів, із використанням ключа шифрування довжиною k бітів, $l,k \in \{256,512\}$, $l = k$. Операції виконуються над s-бітними словами, $s \in \{32,64\}$. Основні загальні параметри шифру наведені в табл. 4.1.

Таблиця 4.1

Загальні параметри блокового шифру «Кипарис»

	Кипарис-256	Кипарис-512
Розмір блока (*l*), бітів	256	512
Довжина ключа (*k*), бітів	256	512
Довжина слова (*s*), бітів	32	64
Кількість ітерацій перетворення (*t*)	10	14

Варто відмітити, що «Кипарис-256» орієнтований на використання на 32-бітних платформах, «Кипарис-512» – на 64-бітних платформах, в т.ч. із вимогами до компактної реалізації та обмеженого енергоспоживання.

4.3.2 Шифруюче перетворення

Загальна схема процедури зашифрування представлена на рисунку 4.2.

На вхід процедури зашифрування подається блок відкритого тексту $P=\left(P_0,P_1,...,P_7\right)$ та циклові ключі $RK^{(0)},RK^{(1)},\ldots,RK^{(t-1)}$. Кожний ключ $RK^{(i)}=\left(RK_0^{(i)},RK_1^{(i)},RK_2^{(i)},RK_3^{(i)}\right)$ складається з 4-х *s*-бітових слів. Циклові ключі формуються за допомогою СРК на основі ключа шифрування $K=\left(K_0,K_1,...,K_7\right)$.

Блок відкритого тексту *P* ділиться на два підблока: $L_0=\left(P_0,P_1,P_2,P_3\right)$, $R_0=\left(P_4,P_5,P_6,P_7\right)$. Вихід *i*-ої ітерації перетворення обчислюється як:

$$\begin{aligned} L_i &= R_{i-1}\oplus F\left(L_{i-1},RK^{(i-1)}\right),\\ R_i &= L_{i-1.} \end{aligned} \qquad (4.1)$$

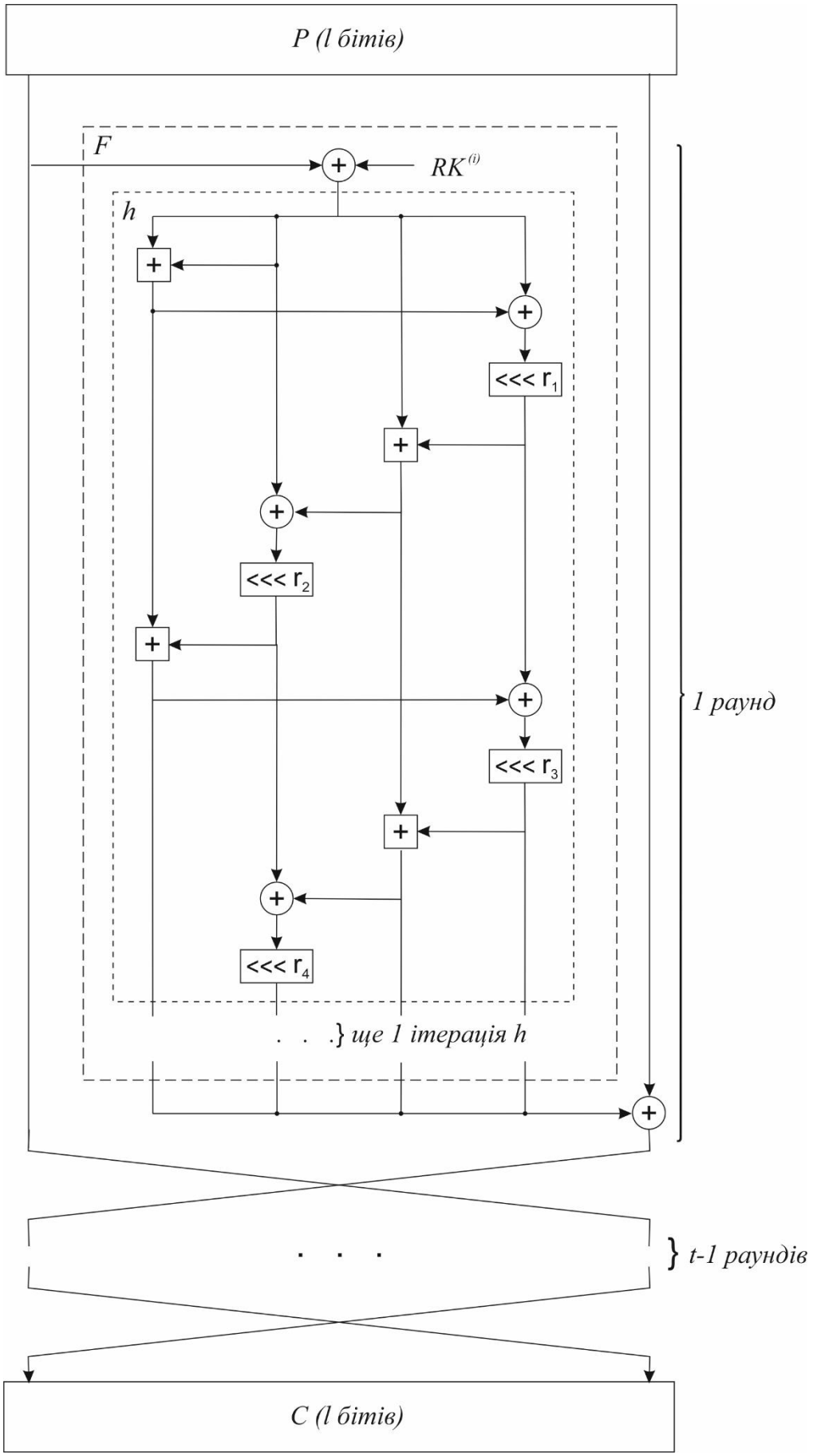

Рис. 4.2 Процедура зашифрування шифру «Кипарис»

Циклова функція F містить додавання підблока L_{i-1} з ключем $RK^{(i-1)}$ за модулем 2 та двократне повторення функції $h\left(P_0', P_1', P_2', P_3'\right)$, на вхід якої подається чотири s-бітних слова. Вихідне значення функції h обчислюється як:

$$\begin{aligned} P_0' &= ADD(P_0', P_1'),\ P_3' = XOR(P_3', P_0'),\ P_3' = ROTL(P_3', r1), \\ P_2' &= ADD(P_2', P_3'),\ P_1' = XOR(P_1', P_2'),\ P_1' = ROTL(P_1', r2), \\ P_0' &= ADD(P_0', P_1'),\ P_3' = XOR(P_3', P_0'),\ P_3' = ROTL(P_3', r3), \\ P_2' &= ADD(P_2', P_3'),\ P_1' = XOR(P_1', P_2'),\ P_1' = ROTL(P_1', r4), \end{aligned} \tag{4.2}$$

де $ADD\left(x, y\right)-$ додавання за модулем s двох s-бітних слів;

$XOR\left(x, y\right)-$ XOR двох s-бітних слів;

$ROTL\left(x, r\right)-$ циклічний зсув s-бітного слова вліво на r біт.

Значення циклічних зсувів $\left(r_0, r_1, r_2, r_3\right)$ залежать від довжини блока і дорівнюють:

- для шифру «Кипарис-256» $\left(r_0, r_1, r_2, r_3\right) = \left(16,12,8,7\right).$
- для шифру «Кипарис-512» $\left(r_0, r_1, r_2, r_3\right) = \left(32,24,16,15\right).$

Процедура розшифрування є ідентичною до процедури зашифрування, лише циклові ключі подаються у зворотному порядку.

4.3.3 Схема розгортання ключів шифру «Кипарис»

Циклові ключі формуються за допомогою неін’єктивної СРК, в основі якої лежить структура СРК шифру «Калина».

При шифруванні використовується t циклових ключів довжиною $4\times s$ бітів.

Спочатку формується проміжний ключ K_σ довжиною $4\times s$ бітів. Формування ключа здійснюється із використанням ключа шифрування $K = \left(K_0, K_1, ..., K_7\right)$ наступним чином:

$$
\begin{aligned}
&K_l = (K_0, K_1, K_2, K_3), K_r = (K_4, K_5, K_6, K_7),\\
&st = XOR(0x1, K_l), st = h(h(st)),\\
&st = ADD(st, K_r), st = h(h(st)),\\
&st = XOR(st, K_l), K_\sigma = st.
\end{aligned}
\tag{4.3}
$$

Циклові ключі формуються на основі ключа шифрування K, проміжного ключа K_σ та константи tmv = (0x000F000F, 0x000F000F, 0x000F000F, 0x000F000F) наступним чином:

$$
\begin{aligned}
&st = (K_0, K_1, K_2, K_3), K_t = ADD(K_\sigma, tmv),\\
&st = ADD(st, K_t), st = h(h(st)),\\
&st = XOR(st, K_t), st = h(h(st)),\\
&st = ADD(st, K_t), RK_{2i} = st,\\
&tmv = ShiftLeft(tmv, 0x1),\\
&st = (K_4, K_5, K_6, K_7), K_t = ADD(K_\sigma, tmv),\\
&st = ADD(st, K_t), st = h(h(st)),\\
&st = XOR(st, K_t), st = h(h(st)),\\
&st = ADD(st, K_t), RK_{2i+1} = st,
\end{aligned}
\tag{4.4}
$$

де функція *ShiftLeft* передбачає зсув кожного s-бітового слова вліво на один біт.

Після формування кожної пари парний/непарний ключ значення tmv та K модифікуються наступним чином:

$$
\begin{aligned}
&tmv = ShiftLeft(tmv, 1),\\
&K = ROTLKey(K, 1),
\end{aligned}
\tag{4.5}
$$

де $ROTLKey(K, 1)$ – циклічний зсув масиву s-бітних слів ключа вліво.

Альтернативні варіанти побудови блокових шифрів наведені в [119,120].

4.4 Аналіз відомих підходів до оцінки стійкості ARX-шифрів до диференційного криптоаналізу

У якості нелінійної операції в ARX-шифрах виступає додавання n-бітових слів за модулем 2^n, а циклові ключі, як правило, вводяться за допомогою операції XOR, тому, диференційна ймовірність шифру визначається ймовірностями перетворення різниць (обчислених за допомогою операції XOR) на модульних суматорах [15]. Розмір модуля є достатньо великим у порівнянні з розмірністю S-блоків (як правило, 32-64 біти), що з точки зору обчислювальної складності унеможливлює побудування повної таблиці розподілу різниць.

Визначення мінімальної кількості гілок активізації також є складною задачею, оскільки перемежування простих (нелінійних та лінійних) операцій, що не підпорядковується чіткому математичному обґрунтуванню, є складним з точки зору аналізу його криптографічних властивостей.

Часто один цикл перетворення ARX-шифру містить декілька нелінійних та лінійних операцій, що чергуються між собою, при цьому ключове забілювання застосовується лише на початку кожного циклу. Згідно класичної теорії [70], такий шифр не є марковським, оскільки вхідні значення нелінійних операцій, починаючи з другої, не рандомізуються ключем. Тим не менше, в сучасних роботах з диференційного криптоаналізу ARX-шифрів робиться припущення, що шифр є марковським, і ймовірність ДХ для одного циклу перетворення обчислюється як добуток ймовірностей перетворення різниць при проходженні крізь нелінійні операції [15]. Таке припущення не матиме суттєвого впливу на результат оцінки, проте значно спростить процес оцінювання (хоча, поодинокі приклади випадків, коли шифр поводить себе як немарковський, також наведені в літературі [118]). В будь-якому разі, на поточний момент невідомо, як розраховувати диференційну ймовірність у разі припущення про «немарковість» шифру. Викладені у цій статті результати також базуватимуться на припущенні, що шифр «Кипарис» є марковським шифром.

Як вже згадувалось, вперше підхід до проєктування ARX-шифрів, які є доказово стійкими до диференційного (лінійного) криптоаналізу, представлений в [16]. Стратегія довгого сліду пропонує використовувати S-блоки разом з простими лінійними операціями [16]. Застосування запропонованого підходу при розробці шифру SPARX дозволило отримати оцінку верхньої границі диференційної ймовірності для цього шифру.

Що стосується оцінки стійкості існуючих ARX-шифрів, на сьогоднішній день не існує універсального теоретичного методу оцінки верхньої границі ймовірності ДХ для ARX-шифрів. Існуючі методи оцінки, як правило, базуються на результатах застосування евристичних алгоритмів пошуку кращих диференційних характеристик [15, 118]. До найбільш відомих таких методів можна віднести наступні:

– модифікований алгоритм Мацуї із застосуванням часткових таблиць розподілу різниць [15], найбільш розвинутий з існуючих методів;

– метод, заснований на пошуку ймовірностних нейтральних бітів (англ. probabilistic neutral bits) [121], наразі застосований до потокових шифрів Salsa та ChaCha;

– метод, заснований на задачі здійснимості булевих формул (англ. SAT solvers) [118], який також запропонований для криптоаналізу потокового шифру Salsa.

Найбільшого застосування набув метод пошуку на основі часткових таблиць розподілу різниць, запропонований А. Бірюковим та В. Вєлічковим [15].

Таблиця розподілу різниць для додавання n-бітових слів за модулем 2^n містить ймовірності перетворення двох вхідних різниць у вихідну після проходження крізь операцію модульного додавання.

Означення 4.1 [15,122]. Нехай α, β та γ – фіксовані n-бітові різниці (відповідно до операції XOR). Диференційна ймовірність додавання за модулем 2^n (xdp^+) – це ймовірність, з якою вхідні різниці α та β переходять у вихідну

різницю γ після проходження через операцію додавання, обчислена для всіх пар n-бітових вхідних текстів (x, y):

$$\mathrm{xdp}^{+}\left(\alpha,\beta\to\gamma\right)=2^{-2n}\cdot\#\left\{\left(x,y\right):\left(\left(x\oplus\alpha\right)+\left(y\oplus\beta\right)\right)\oplus\left(x+y\right)\right\}=\gamma. \qquad (4.6)$$

Як можна помітити з формули (4.6), розрахунок значення ймовірності прямим шляхом перебирання усіх вхідних пар навіть для одного переходу є обчислювально складною задачею. Ефективний алгоритм для практичного обчислення значення xdp^{+} представлений в [123].

У зв'язку з великим розміром модуля n, навіть із застосуванням ефективного алгоритму, побудування повної таблиці розподілу різниць є нездійсненною на практиці задачею. У свою чергу в [15] пропонується будувати так звану часткову TPP, що містить диференціали з ймовірністю, що дорівнює або перевищує заданий поріг [15]:

$$(\alpha,\beta,\gamma)\in D\Leftrightarrow(\alpha,\beta\to\gamma)\geq p_{thres}. \qquad (4.7)$$

Далі пропонується побудувати таку часткову таблицю для усієї циклової функції та, використовуючи модифікований алгоритм Мацуї, здійснити пошук диференційних характеристик. Цей метод був успішно застосований до блокових шифрів заснованих на мережі Фейстеля з ARX-подібною цикловою функцією, а саме таких як SPECK, TEA, XTEA та ін.

Можна відмітити, що описаний метод найбільше підходить до шифрів з достатньо простою цикловою функцією, в якій не передбачається поділу вхідного значення на слова. Це пояснюється тим, що коли операції додавання та зсуву застосовуються до цілого вхідного значення, побудувати часткову TPP для такої циклової функції достатньо просто.

4.5 Математична модель оцінки стійкості визначеного класу ARX-шифрів до диференційного криптоаналізу

Нехай $\oplus$ (XOR) є операцією, що визначає різницю між парою текстів, а $\boxplus$ є операцією додавання за модулем 2^s, для якої диференційна ймовірність $\mathrm{xdp}^+ \le 1$. Опишемо клас ARX-шифрів, що підлягатиме аналізу.

1. Найменше елементарне перетворення містить три послідовні операції: додавання за модулем, XOR та циклічний зсув (див. рис. 4.1).

2. Операції виконуються над *s*-бітовими словами довільної розрядності.

3. Вихід *i*-ої ітерації перетворення обчислюється згідно з формулою (4.1).

4. Кількість слів, що поступають на вхід циклової функції, дорівнює чотирьом: P_0, P_1, P_2, P_3.

5. Композиція операцій над словами циклової функції *F* має наступний вигляд:

$$F(P_0,...,P_3) = \prod_{i=1}^{2} \eta(P_0,P_1) \circ \chi(P_3,P_0) \circ \rho_{r_j^{(1)}}(P_3) \circ \\ \circ \eta(P_2,P_3) \circ \chi(P_1,P_2) \circ \rho_{r_j^{(0)}}(P_1), \tag{4.8}$$

де $\eta(x,y)-$ додавання за модулем *s* двох *s*-бітних слів;

$\chi(x,y)-$ XOR двох *s*-бітних слів;

$\rho_{r_i^{(j)}}(x)-$ циклічний зсув *s*-бітного слова вліво на $r_i^{(j)}$ біт.

Для визначеного класу ARX-шифрів введемо наступні припущення.

Припущення 4.1. ARX-шифр, що належить до визначеного класу, є марковським шифром, тому:

а) середня за ключами ймовірність одноциклової ДХ $\mathrm{EDP}^{(1)}(\Omega)$ дорівнює добутку ймовірностей перетворення вхідних різниць на *m* модульних суматорах:

$$\mathrm{EDP}^{(1)}(\Omega)=\prod_{i=1}^{m}\mathrm{xdp}^{+}(\alpha_i,\beta_i\to\gamma_i), \tag{4.9}$$

де $(\alpha_i,\beta_i)-$ різниці на вході *i*-го суматора, γ_i- різниця на виході *i*-го суматора;

б) середня за ключами ймовірність *r*-циклової ДХ визначається добутком ймовірностей одноциклових ДХ [70].

Нехай $(\Omega_1,\Omega_2,...,\Omega_r)$ – множина одноциклових ДХ таких, що $\Omega_1=(\alpha,\beta_1),\Omega_2=(\beta_1,\beta_2),...,\Omega_r=(\beta_{r-1},\beta_r)$ та $\Omega=(\alpha,\beta_1,...,\beta_r)$. Тоді ймовірність $\mathrm{EDP}^{(r)}(\Omega)$ може бути апроксимована як

$$\mathrm{EDP}^{(r)}(\Omega)=\prod_{i=1}^{r}\mathrm{EDP}^{(1)}(\Omega_i). \tag{4.10}$$

Припущення 4.1 витікає із загальноприйнятих припущень, що робляться з метою спрощення отримання оцінок для ARX-шифрів [15, 70].

Припущення 4.2. При обчисленні вихідної різниці γ, в яку перетворюються вхідні різниці α та β після проходження крізь операцію модульного додавання, обирається вихідна різниця, що має максимальну ймовірність:

$$\gamma=\boxplus(\alpha,\beta),\mathrm{xdp}^{+}(\alpha,\beta\to\gamma)=\max\Gamma, \tag{4.11}$$

де $\Gamma-$ множина усіх можливих вихідних різниць для (α,β).

У багатьох випадках, для пари вхідних різниць (α,β) існує декілька вихідних різниць з максимальною ймовірністю. Якщо таких різниць небагато ($\approx$5-10), тоді обчислюються диференційні шляхи для усіх можливих варіантів. У разі, коли значення $\max\Gamma$ є достатньо малим, кількість вихідних різниць з максимальною ймовірністю може бути дуже великою (тисячі та десятки тисяч).

Тоді робиться випадкова вибірка (англ. random sampling) з множини різниць, що мають максимальну імовірність, та будуються диференційні шляхи лише для них.

Припущення 4.3. У високоймовірнісних одноциклових ДХ ARX-шифру, що належить до визначеного класу, вхідні різниці мають малу вагу Гемінга, а саме 3-7 активних бітів (при цьому, активні біти рознесені по різним словам). Таке припущення пояснюється тим, що вхідні різниці найбільш ймовірних переходів в таблиці розподілу різниць для модульного додавання мають малу кількість активних бітів. Обґрунтуємо припущення 4.3 на прикладі ARX-шифру «Кипарис».

Означення 4.2. Кількістю активних біт b у різниці (α,β), яка поступає на вхід суматора, називається число одиниць, що міститься у доданку $\alpha \oplus \beta$.

Розглянемо часткову ТРР для додавання за модулем 2^{32}, що містить переходи з ймовірністю $\mathrm{xdp}^{+}(\alpha,\beta \to \gamma) \geq 1/2$. Для переходів з ймовірністю $\mathrm{xdp}^{+}(\alpha,\beta \to \gamma) = 1$, яких у ТРР всього чотири, кількість активних біт у вхідній різниці дорівнює $b \leq 1$. Зазначимо, що це справедливо для будь-якого значення n (див. табл. 4.2).

Для переходів з ймовірністю $\mathrm{xdp}^{+}(\alpha,\beta \to \gamma) = 1/2$ (для $n = 32$ таких 744), кількість активних біт у вхідній різниці обмежується двома, $b \leq 2$.

У [122] наводиться вираз, що описує зв'язок між позиціями бітів вхідної та вихідної різниць й імовірністю, а саме верхня границя диференційної ймовірності операції модульного додавання визначається як $\Pr[\alpha,\beta \to \gamma] \leq 2^{-k}$, де $k = \#\{i : \neg(\alpha[i] = \beta[i] = \gamma[i]), 0 \leq i \leq n-2\}$, тобто кількість бітових позицій, за виключенням найбільш значущого біта, на яких біти різниць α,β,γ не є рівними.

Таблиця 4.2

Переходи в ТРР для додавання за модулем 2^n з ймовірністю

$$\mathbf{xdp}^{+}(\alpha,\beta\to\gamma)=\mathbf{1}$$

№	α	β	γ
1	0	0	0
2	$10\ldots0_{n-1}$	0	$10\ldots0_{n-1}$
3	0	$10\ldots0_{n-1}$	$10\ldots0_{n-1}$
4	$10\ldots0_{n-1}$	$10\ldots0_{n-1}$	0

Таким чином, мінімізація кількості активних бітів у різниці на вході модульних суматорів підвищує загальну ймовірність ДХ. У шифрі «Кипарис» перші три слова різниці, що подається на вхід циклової функції, попадають на вхід модульного суматора одразу, а четверте – після застосування операцій побітового додавання та циклічного зсуву, тому припущення про малу кількість активних біт на вході циклової функції є цілком обґрунтованим. Враховуючи вплив лінійних операцій на процес розповсюдження активних бітів, припускається, що 1-2 активних біти на вході циклової функції добре розповсюдяться по різним словам на виході. Експерименти показали, що 1 активний біт на вході циклової функції переходить щонайменше у 7 активних бітів на виході (див. табл. 4.3). У свою чергу, декілька активних бітів у різних словах можуть знищитись за рахунок застосування лінійних операцій. Таким чином, приблизно 3-7 активних бітів на вході циклової функції, які розподілені між різними словами, дозволять отримати високоймовірну ДХ, оскільки забезпечать оптимальне розповсюдження активних бітів для максимізації ймовірності перетворення різниць на суматорах.

Таблиця 4.3

Результати щодо розповсюдження активних бітів для циклової функції блокового шифру «Кипарис»

128-бітова вхідна різниця (1 позначає слово, в якому є активні біти)	Кількість активних бітів на виході циклової функції	
	Нижня границя	Верхня границя
1000	14	14
0100	19	19
0010	7	7
0001	10	10
1100	5	33
0110	12	26
0011	3	17
0101	9	29
1010	7	21
1001	4	24
1110	2	40
1101	5	43
1011	3	21
0111	2	36

4.6 Методи пошуку високоймовірнісних одноциклових диференційних характеристик визначеного класу ARX-шифрів

Як зазначалось вище, найбільш відомим методом пошуку диференційних характеристик є модифікований метод Мацуї із застосуванням часткових ТРР [15], який добре підходить до шифрів з простими цикловими функціями з невеликим розміром входу, що послідовно обробляється декількома операціями додавання та зсуву. У випадку шифру визначеного класу, де 128/256-бітове вхідне значення циклової функції ділиться на 32/64-бітові слова, які проходять крізь велику кількість операцій додавання, часткова ТРР повинна містити

диференціали з достатньо низькою ймовірністю. Крім того, кількість диференційних шляхів зростає з кожним суматором. Все це призводить до того, що обсяг часткової ТРР стає значно великим для обчислення за прийнятний час. Тому вважатимемо, що при побудуванні диференційних характеристик для ARX-шифрів визначеного класу, краще обчислювати значення ймовірностей на суматорах «на льоту», користуючись швидким алгоритмом, запропонованим Ліпмою та Моріарі [123].

Оцінку диференційних властивостей розпочнемо з пошуку найбільш ймовірних ДХ циклової функції (одноциклових ДХ). Згідно з припущенням 4.3, найбільш ймовірними є одноциклові ДХ, вхідні різниці яких мають вагу Гемінга, що дорівнює 3-7 активним бітам. Оскільки побудування одноциклових ДХ для усіх можливих варіантів таких вхідних різниць на персональному комп'ютері займає багато часу, запропонуємо більш ефективні методи пошуку ДХ циклової функції.

4.6.1 Прямий метод пошуку диференційних характеристик циклової функції

У цьому методі пропонується побудувати одноциклові ДХ для вхідних різниць із вагою Гемінга 1-3 бітів (тобто, такі, що можливо швидко обчислити). У загальному випадку, для ARX-шифру, що належить до визначеного класу, з довжиною блока l бітів метод прямого пошуку складається з наступних кроків.

а) Сформувати множину вхідних різниць Ξ, що містить усі можливі комбінації $l/2$ – бітових рядків із вагою Гемінга 1-3 біти ($l/2$ – довжина напівблока, що подається на вхід циклової функції).

б) Для кожної вхідної різниці $\xi_i \in \Xi$ побудувати одноциклову ДХ. Вихідні різниці після проходження крізь операцію модульного додавання обчислювати згідно з припущенням 4.2.

в) Ймовірність отриманої ДХ $\mathrm{EDP}^{(1)}(\Omega)$ обчислити згідно з пунктом а) припущення 4.1. Зазначимо, що для однієї вхідної різниці, як правило, буде існувати декілька ДХ.

У результаті, за допомогою запропонованого методу для 256-бітової циклової функції шифру «Кипарис» була знайдена ДХ з імовірністю $\mathrm{EDP}^{(1)}(\Omega) = 2^{-12}$ (див. табл. 4.4).

Таблиця 4.4

Параметри ДХ циклової функції шифру «Кипарис»

Вхідна різниця (hex)	80000000 80000000 80000000 0
Вихідна різниця (hex)	40844 26260262 C4484400 44084400
Ймовірність ($\log_2 p$)	-12

4.6.2 Метод пошуку ДХ циклової функції «у двох напрямках»

Завдяки дифузії навіть один активний біт на вході циклової функції вже в середині перетворення активує достатньо велику кількість бітів. Як було вказано вище, циклова функція F представляє собою дві ітерації функції h. З метою активізації якомога меншого числа бітів на входах суматорів пропонується оптимізація прямого методу пошуку ДХ, що полягає в наступному.

Циклова функція представляється як дві ітерації функції h. Метод складається з наступних кроків.

а) Сформувати множину вхідних різниць Ξ, що містить усі можливі комбінації $l/2$ – бітових рядків із вагою Гемінга 1-3 біти.

б) Кожну різницю $\xi_i \in \Xi$ розглядати як вихід першої та вхід другої ітерації функції h. Для кожної вхідної різниці $\xi_i \in \Xi$ побудувати диференційні характеристики окремо для функції h та її інверсії h^{-1}. Вихідні різниці після

проходження крізь операцію модульного додавання обчислювати згідно з припущенням 4.2.

в) ДХ для циклової функції F буде представляти об'єднання характеристик для h^{-1} та h. Ймовірність об'єднаної ДХ $\mathrm{EDP}^{(1)}(\Omega)$ обчислити згідно з пунктом а) припущення 4.1.

Цей метод дозволив значно покращити попередній результат: для 256-бітової циклової функції шифру була знайдена ДХ з імовірністю $p(\xi \to \psi) = 2^{-3}$ (див. табл. 4.5).

Таблиця 4.5

Параметри ДХ циклової функції, отриманої за допомогою методу пошуку «у двох напрямках»

Вхідна різниця (hex)	80000 80080000 80000000 80000000
Вихідна різниця (hex)	800 4040040 80080000 80000
Ймовірність ($\log_2 p$)	-3

4.6.3 Оптимізований метод пошуку найкращої ДХ циклової функції

Відмітимо, що для знайденої вище ДХ, ймовірність перетворення на суматорі $p(\alpha_j, \beta_j \to \gamma_j) \geq 1/2$. Цей факт дозволяє стверджувати, що якщо існує якась одноциклова ДХ, краща за знайдену, то вона також буде містити переходи на суматорах з імовірністю, не менше ½. Таким чином, множину вхідних різниць Ξ можна суттєво обмежити, вибравши вхідні різниці таким чином, щоб ймовірність перетворення на перших двох суматорах була не менше ½. Це дозволить знайти ДХ з високою ймовірністю, яка існує для циклової функції. Оптимізований метод пошуку ДХ складається з наступних кроків.

а) Побудувати ТРР для операції додавання за модулем 2^n, що містить переходи з імовірністю $p(\alpha_j, \beta_j \to \gamma_j) \geq 1/2$.

б) Враховуючи, що кожна вхідна різниця $\xi_i \in \Xi$ складається з чотирьох s-бітових слів $\xi_i = \left\{\xi_i^{(0)}, \xi_i^{(1)}, \xi_i^{(2)}, \xi_i^{(3)}\right\}$, множину вхідних різниць Ξ сформувати за допомогою алгоритму, що наведений на рисунку 4.3.

$$for\ each\ (\alpha_j, \beta_j \to \gamma_j)\ from\ pDDT$$

$$\xi_i^{(0)} = \alpha_j; \xi_i^{(1)} = \beta_j;$$

$$for\ each\ (\alpha_k, \beta_k \to \gamma_k)\ from\ pDDT$$

$$\xi_i^{(2)} = \alpha_k; \xi_i^{(3)} = XOR(\gamma_j, ROTR(\beta_k, r));$$

Рис. 4.3 Метод формування множини вхідних різниць

в) Для кожної вхідної різниці $\xi_i \in \Xi$ побудувати одноциклову ДХ. Вихідні різниці після проходження крізь операцію модульного додавання обчислювати згідно з припущенням 4.2.

г) Ймовірність об'єднаної ДХ $EDP^{(1)}(\Omega)$ обчислити згідно з пунктом а) припущення 4.1.

Цей метод дозволив ще покращити попередній результат: для 256-бітової циклової функції шифру була знайдена ДХ з імовірністю $p(\xi \to \psi) = 2^{-2}$ (див. табл. 4.6).

Таблиця 4.6

Параметри ДХ циклової функції шифру «Кипарис», отриманої за допомогою оптимізованого методу

Вхідна різниця (hex)	0 80000000 800000 80008080
Вихідна різниця (hex)	80000000 4000 80 80
Ймовірність ($\log_2 p$)	-2

4.7 Методи пошуку багатоциклових диференційних характеристик блокового шифру «Кипарис»

4.7.1 Метод пошуку багатоциклових ДХ, заснований на побудуванні множини високоймовірнісних одноциклових ДХ

Нагадаємо, що пошук диференційних характеристик проводиться з метою знаходження високоймовірнісних диференційних шляхів та підтвердження, що ймовірність найкращої знайденої (r-1)-циклової ДХ $\mathrm{EDP}^{(r-1)}(\Omega) < 2^k$, де k – довжина ключа шифрування. Для виконання цієї задачі, по-перше, пропонується побудувати достатньо велику множину одноциклових ДХ та виконати пошук можливих комбінацій одноциклових ДХ у двоциклові (багатоциклові) ДХ. Оскільки до множини необхідно включити не лише найбільш ймовірні ДХ, то тут вже прийдеться побудувати одноциклові ДХ для вхідних різниць з вагою Гемінга 1-6 активних бітів (звісно, застосовуючи певний поріг для відсіювання ДХ і отримання прийнятного часу обчислень). В загальному вигляді метод складається з наступних кроків.

а) Згідно з припущенням 4.3, сформувати множину вхідних різниць Ξ, для яких буде побудовано одноциклові ДХ. До множини Ξ включити усі можливі комбінації $l / 2$ – бітових рядків з вагою Гемінга 1-6 бітів.

б) Побудувати одноциклові ДХ для вхідних різниць з множини Ξ. Вихідні різниці після проходження крізь операцію модульного додавання обчислювати згідно з припущенням 2, а ймовірність одноциклової ДХ $\mathrm{EDP}^{(1)}(\Omega)$ згідно з пунктом а) припущення 4.1.

в) Враховуючи, що довжина ключа дорівнює k, а кількість циклів шифрування дорівнює t, з усіх обчислених ДХ до множини Ψ включити ДХ, що мають ймовірність $\mathrm{EDP}^{(1)}_{thres}(\Omega) \geq 2^{-k/t}$.

г) Якщо для вхідних різниць з деякою вагою Гемінга обчислення всіх ДХ потребує значних обчислювальних ресурсів, зменшити значення $\mathrm{EDP}^{(1)}_{thres}(\Omega)$ для ДХ, побудованих для вхідних різниць з цією вагою Гемінга.

д) Здійснити пошук комбінацій одноциклових ДХ з множини Ψ у двоциклові (багатоциклові) ДХ.

Запропонований метод був застосований до шифру «Кипарис-256». У зв'язку з обмеженням обчислювальних ресурсів, до множини Ψ були додані ДХ:

– з ймовірністю $\mathrm{EDP}^{(1)}_{thres}(\Omega) \geq 2^{-26}$ для вхідних різниць з вагою Гемінга 1-4 бітів;

– з ймовірністю $\mathrm{EDP}^{(1)}_{thres}(\Omega) \geq 2^{-18}$ для вхідних різниць з вагою Гемінга 5 бітів;

– з ймовірністю $\mathrm{EDP}^{(1)}_{thres}(\Omega) \geq 2^{-10}$ для вхідних різниць з вагою Гемінга 6 бітів.

Результати побудування множини високоймовірнісних одноциклових ДХ наведені в табл. 4.7.

Таблиця 4.7

Характеристики множини високоймовірнісних одноциклових ДХ

Вага Гемінга вхідної різниці	$\mathrm{EDP}^{(1)}_{thres}(\Omega), \log_2 n$	$\mathrm{MEDP}^{(1)}(\Omega), \log_2 n$	$\lvert\Psi\rvert$
1	-26	> -26	0
2	-26	-14	2986
3	-26	-12	10357
4	-26	-6	28392
5	-18	-2	1446
6	-10	-3	343

Як і було припущено у математичній моделі (див. припущення 4.3), ДХ, побудовані для вхідних різниць з вагою Гемінга 4-6 бітів мають високу ймовірність. Деякі з отриманих ДХ представлені у табл. 4.8.

Таблиця 4.8

Найбільш ймовірні одноциклові диференційні характеристики блокового шифру «Кипарис»

Вхідна різниця у 32-бітових словах, hex	Вихідна різниця у 32-бітових словах, hex	$\mathrm{EDP}^{(1)}(\Omega), \log_2 n$
0 80000000 800000 80008080	80000000 4000 80 80	-2
80000 80080000 80000000 80000000	800 4040040 80080000 80000	-3
0 80000000 1800000 80008080	80000000 4000 80 80	-3
180000 80080000 80000000 80000000	800 4040040 80080000 80000	-4
80000 80000 80800000 8080	80000800 4044040 80080080 80080	-5
80000000 0 80000000 80008000	88000000 40404404 808088 800088	-6
80000000 80000000 80800000 80	8000000 40400404 808008 800008	-6
80 80 80000080 8000	8 40040440 80800800 80000800	-7
8000 8000 8080 800000	800 4044040 80080080 80080	-7
80000000 80000800 800 800	800000 40040040 80000800 80000000	-7
0 80 80000000 808080	80 400000 8000 8000	-7
0 800000 8000 80800080	800000 40 80000000 80000000	-7
80000000 80000000 81800000 80	8000000 40400404 808008 800008	-7
0 100 1 1010100	100 800000 10000 10000	-8
0 200 2 2020200	200 1000000 20000 20000	-8
0 800 8 8080800	800 4000000 80000 80000	-8
0 1000 10 10101000	1000 8000000 100000 100000	-8
0 2000 20 20202000	2000 10000000 200000 200000	-8
0 4000 40 40404000	4000 20000000 400000 400000	-8
0 8000 80 80808000	8000 40000000 800000 800000	-8
180 80 80000080 8000	8 40040440 80800800 80000800	-8
80 80 80000180 8000	8 40040440 80800800 80000800	-8
100 80 80000000 808080	80 400000 8000 8000	-8
80001000 80000800 800 800	800000 40040040 80000800 80000000	-8
8000 8000 8180 800000	800 4044040 80080080 80080	-8
80000000 40000000 400000 40004040	40000000 2000 40 40	-8
0 40 c0000000 404040	40 200000 4000 4000	-8
0 800000 18000 80800080	800000 40 80000000 80000000	-8
0 40000000 80400000 40004040	40000000 2000 40 40	-8

Зазначимо, що деякі ДХ мають вихідну різницю, яка співпадає з вхідною різницею інших ДХ, проте жодних комбінацій одноциклових ДХ у двоциклові виявлено не було (див. пункт д) запропонованого методу). Це означає, що отримані ДХ, вихідні різниці яких мають малу вагу Гемінга, не можуть бути продовжені для побудування багатоциклових ДХ з високою ймовірністю.

4.7.2 Пошук існуючих найбільш ймовірних багатоциклових ДХ та оцінка стійкості блокового шифру «Кипарис-256»

Наступний крок на шляху пошуку багатоциклових ДХ полягає у продовженні одноциклових ДХ з множини Ψ на декілька циклів. Відмітимо, що особливість архітектури мережі Фейстеля дозволяє підбирати вхідну різницю таким чином, щоб «пропустити» один цикл шифрування, тобто створити таку ситуацію, коли на певному циклі на вхід циклової функції подається значення $\Delta X = 0$, ймовірність перетворення якого дорівнює 1. З метою максимізації ймовірності ДХ для перших трьох циклів шифрування, в якості лівої половини різниці пропонується подати значення ΔX, а в якості правої – $\Delta Y = F(\Delta X)$. Завдяки цьому ймовірність ДХ для першого та третього циклів буде однаковою, а для другого дорівнюватиме 1. Шлях проходження вхідної різниці для чотирьох циклів шифрування зображено на рис. 4.4.

Пошук найбільш ймовірних багатоциклових ДХ складається з наступних кроків.

а) Визначити *l*-бітову вхідну різницю як таку, що складається з двох $l/2-$бітових половин ΔX та ΔY.

б) Сформувати множину Z *l*-бітових вхідних різниць для пошуку багатоциклових ДХ наступним чином. Визначити вхідну різницю ζ_i з множини Z як $\zeta_i = (\Delta X_i \,|\, \Delta Y_i)$, де ΔX_i та $\Delta Y_i -$ значення вхідної та вихідної різниць *i*-ої ДХ з множини Ψ відповідно.

в) Для кожної вхідної різниці ζ_i з множини Z побудувати ДХ для j циклів за умови, що $\mathrm{EDP}^{(j)}(\Omega) > 2^{-256}$. ДХ для кожного циклу будувати згідно пунктів (б)-(г) методу, представленого раніше.

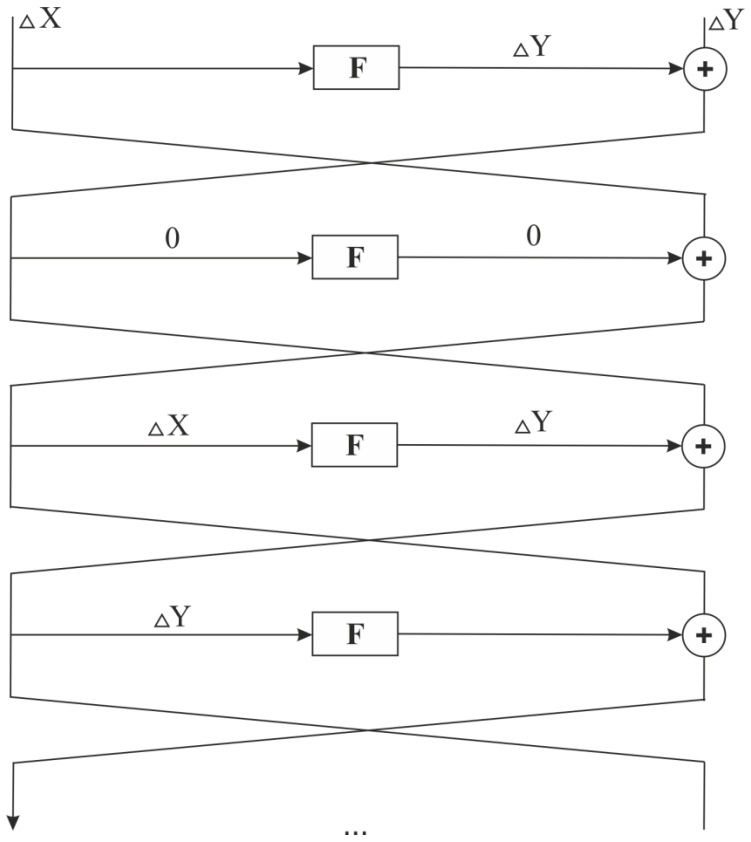

Рис. 4.4 Шлях проходження вхідної різниці для 4-х циклів шифрування

У табл. 4.9 представлені параметри однієї з найбільш ймовірних ДХ, знайденої за допомогою описаного вище методу. Зазначимо, що за рахунок застосування механізму випадкової вибірки а) при обчисленні значень вихідних різниць для операції модульного додавання та б) при обранні виходу з циклової функції між циклами шифрування, значення $\mathrm{EDP}^{(j)}(\Omega)$ є апроксимованим (обчислення всіх існуючих диференційних шляхів навіть для одного значення вхідної різниці є обчислювально складною задачею).

Реалізація методів пошуку диференційних характеристик блокового шифру «Кипарис-256» мовою програмування C++ наведена у Додатку Г.

Таблиця 4.9

Параметри однієї з найбільш ймовірних знайдених багатоциклових ДХ для блокового шифру «Кипарис»

№ циклу, j	ДХ $\Omega(a,b)$ для j-го циклу, hex	$\mathrm{EDP}^{(1)}(\Omega)$, $\log_2 n$	$\mathrm{EDP}^{(j)}(\Omega)$, $\log_2 n$
1	a = (00000000 80008000 00800080 00800080 80008000 40004000 00800080 00800080), b = (00000000 80008000 00800080 00800080 00000000 00000000 00000000 00000000)	-10	-10
2	a = (00000000 00000000 00000000 00000000 00000000 80008000 00800080 00800080), b = (00000000 00000000 00000000 00000000 00000000 80008000 00800080 00800080)	0	-10
3	a = (00000000 80008000 00800080 00800080 00000000 00000000 00000000 00000000), b = (00000000 80008000 00800080 00800080 80008000 40004000 00800080 00800080)	-10	-20
4	a = (80008000 40004000 00800080 00800080 00000000 80008000 00800080 00800080), b = (80008000 40004000 00800080 00800080 c0204020 90009000 00a000a0 00800080)	-27	-47
5	a = (c0204020 90009000 00a000a0 00800080 80008000 40004000 00800080 00800080), b = (c0204020 90009000 00a000a0 00800080 5208d204 40444044 08820882 0a800a80)	-74	-121
6	a = (5208d204 40444044 08820882 0a800a80 c0204020 90009000 00a000a0 00800080), b = (5208d204 40444044 08820882 0a800a80 266e7071 a74313f2 0088e7e0 10fa6fd2)	-102	-223

Таким чином, знайдена найбільш ймовірна ДХ для 6 циклів шифрування має ймовірність

$$\mathrm{MEDP}^{(6)}(\Omega) \approx 2^{-223}. \qquad (4.12)$$

Отримане значення $\mathrm{MEDP}^{(6)}(\Omega)$ будемо називати практичною стійкістю блокового шифру «Кипарис» до диференційного криптоаналізу. Через

застосування механізму випадкової вибірки, значення $\mathrm{MEDP}^{(6)}(\Omega)$ може дещо відрізнятися у різних експериментах, проте це суттєво не впливає на загальний результат оцінки, оскільки

$$\mathrm{MEDP}^{(7)}(\Omega) << 2^{-256}, 7 < (r-1). \qquad (4.13)$$

Таким чином, блоковий шифр «Кипарис-256» є стійким до диференційного криптоаналізу відповідно до вимог практичного критерію.

Висновки до розділу 4

1. У розділі запропонований блоковий шифр, який задовольняє сучасним вимогам до симетричних примітивів. Шифр заснований на мережі Фейстеля, а циклова функція шифру представляє собою ARX-перетворення. Схема розгортання циклових ключів шифру є неін'єктивною та використовує принципи побудови СРК шифру «Калина». Алгоритм підтримує довжину блока (ключа) 256 та 512 бітів, що дозволяє забезпечити високий рівень криптографічної стійкості. «Кипарис-256» орієнтований на використання на 32-бітових платформах, «Кипарис-512» – на 64-бітових платформах.

2. Розроблена математична модель оцінки стійкості визначеного класу ARX-шифрів до диференційного криптоаналізу дозволяє отримати вираз для обчислення середньої за ключами ймовірності диференційної характеристики шифру та зробити обґрунтоване припущення щодо значень вхідних різниць, які при проходженні через цикл шифрування формують характеристику, що має високу ймовірність. Модель включає припущення про те, що шифр є марковським, про ймовірність перетворення диференційних різниць на модульних суматорах, про зв'язок кількості активних біт у вхідній різниці з ймовірністю диференційної характеристики, а також містить вирази для обчислення ймовірностей одноциклових та багатоциклових диференційних

характеристик. Представлена модель лежить в основу розроблених методів пошуку диференційних характеристик класу блокових ARX-шифрів.

3. Розроблено три методи пошуку диференційних характеристик циклової функції визначеного класу ARX-шифрів (прямий метод пошуку, метод пошуку «у двох напрямках» та оптимізований метод пошуку диференційної характеристики з високою ймовірністю), що дозволяють знайти найбільш ймовірні одноциклові диференційні характеристики. Метою всіх трьох підходів є активізація найменшої кількості біт на входах суматорів циклової функції, що, в свою чергу, збільшує ймовірність заданого перетворення. Останній із запропонованих методів дозволив знайти диференційну характеристику на циклову функцію блокового шифру «Кипарис» з ймовірністю, що дорівнює ¼.

4. Найбільшу ймовірність мають одноциклові диференційні характеристики блокового шифру «Кипарис», вхідна різниця яких містить приблизно 3-7 активних бітів, які розподілені між різними словами. Це пояснюється оптимальним розповсюдження активних бітів, що призводить до максимізації ймовірності перетворення різниць на суматорах.

5. Удосконалені методи пошуку багатоциклових диференційних характеристик визначеного класу ARX-шифрів дозволяють знайти високоймовірні *r*-циклові диференційні характеристики в умовах використання обмежених обчислювальних ресурсів. Перший метод базується на формуванні множини одноциклових диференційних характеристик та їх подальшому комбінуванні у багатоциклові характеристики. Другий метод передбачає послідовне «нарощування» характеристики.

6. Застосування запропонованого методу пошуку багатоциклових диференційних характеристик, заснованого на побудуванні множини високоймовірнісних одноциклових диференційних характеристик, до блокового шифру «Кипарис-256» показало, що побудовані одноциклові ДХ, вихідні різниці яких мають малу вагу Гемінга (а значить і достатньо високу ймовірність), не можуть бути продовжені для побудування багатоциклових ДХ з високою ймовірністю.

7. Одна зі знайдених найбільш ймовірних багатоциклових диференційних характеристик для блокового шифру «Кипарис-256» може бути побудована лише для шести циклів шифрування з ймовірністю $\mathrm{MEDP}^{(6)}(\Omega) \approx 2^{-223}$, що дає підстави стверджувати, що блоковий шифр «Кипарис-256» є стійким до диференційного криптоаналізу з точки зору практичного критерію.

Результати досліджень даного розділу наведено в публікаціях здобувача: [124-135].

РОЗДІЛ 5
ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ВЛАСТИВОСТЕЙ СИМЕТРИЧНОГО БЛОКОВОГО ШИФРУ «КИПАРИС»

У цьому розділі наводяться результати експериментальних досліджень наступних властивостей симетричних блокових шифрів «Кипарис-256» та «Кипарис-512»:

– статистичних властивостей шифруючих перетворень та схем розгортання ключів, оцінених згідно з методикою NIST STS;

– лавинних властивостей, а саме таких показників лавинного ефекту, як мінімуми та максимуми математичного сподівання та середньоквадратичного відхилення кількості змінених вихідних бітів при зміні одного вхідного біта для випадкової вибірки вхідних блоків;

– характеристик продуктивності, а саме такого показника, як швидкість зашифрування в режимі простої заміни (у Мбіт/сек); також наводиться порівняння швидкостей зашифрування шифрів «Кипарис-256» та «Кипарис-512» та низки відомих блокових шифрів таких як, AES, SPECK, PRESENT, SPARX та ГОСТ 28147-89.

5.1 Статистичні властивості шифруючого перетворення та схеми розгортання ключів блокового шифру «Кипарис»

Статистичні властивості шифруючого перетворення та схеми розгортання ключів оцінювалися згідно з методикою NIST STS [136]. Для оцінки статистичних властивостей були обрані вхідні послідовності, що мають максимальну збитковість.

Для тестування шифруючого перетворення була обрана послідовність відкритих текстів $m_0 = 0$, $m_1 = 1$, …, $m_i = i$, $m_n = 100000$, (з максимальною

збитковістю). Вихідна послідовність шифртекстів (отримана в режимі електронної кодової книги) тестувалася згідно з NIST STS.

Результати статистичного тестування сформованих вихідних послідовностей для розміру блока 256 та 512 бітів наведені у додатку Б.

Статистичні профілі вихідних послідовностей для розміру блока 256 та 512 бітів для повного набору циклів шифрування наведені на рисунку 5.1 та рисунку 5.2 відповідно.

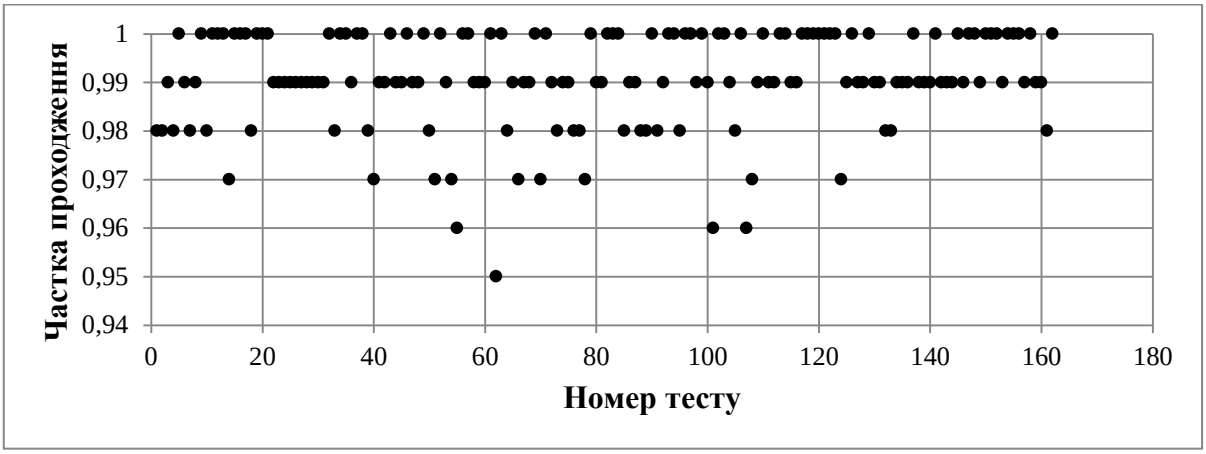

Рис. 5.1 Статистичний профіль вихідної послідовності шифруючого перетворення для розміру блока 256 бітів

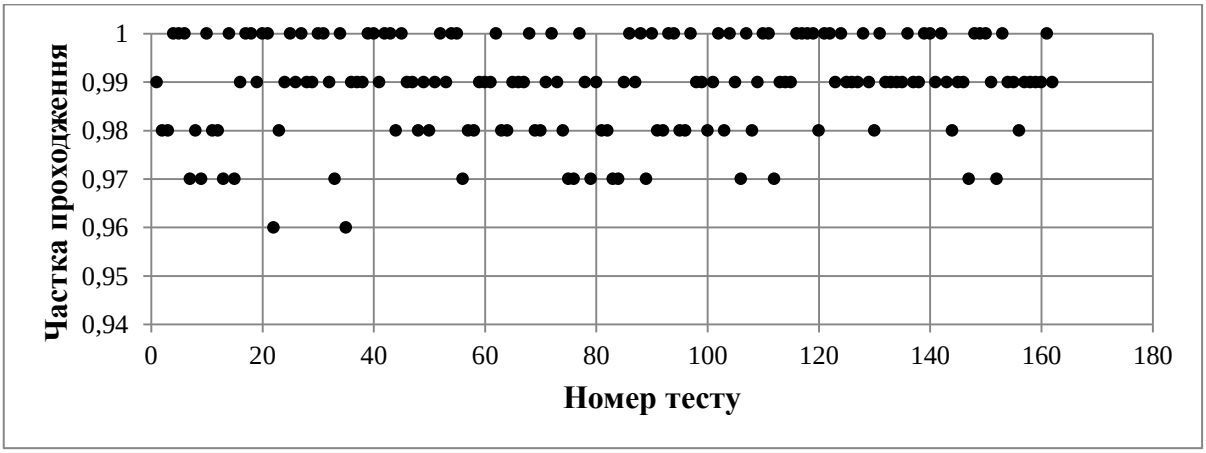

Рис. 5.2 Статистичний профіль вихідної послідовності шифруючого перетворення для розміру блока 512 бітів

Відмітимо, що статистичні властивості було досліджено і для неповної кількості циклів шифрування. Експерименти показали, що вже після четвертого циклу шифрування вихідна послідовність з шифртекстів проходить усі тести на випадковість як для 256-бітового, так і для 512-бітового шифру. На рис. 5.3 представлений статистичний профіль вихідної послідовності шифруючого перетворення після 4-го циклу для шифру «Кипарис-512».

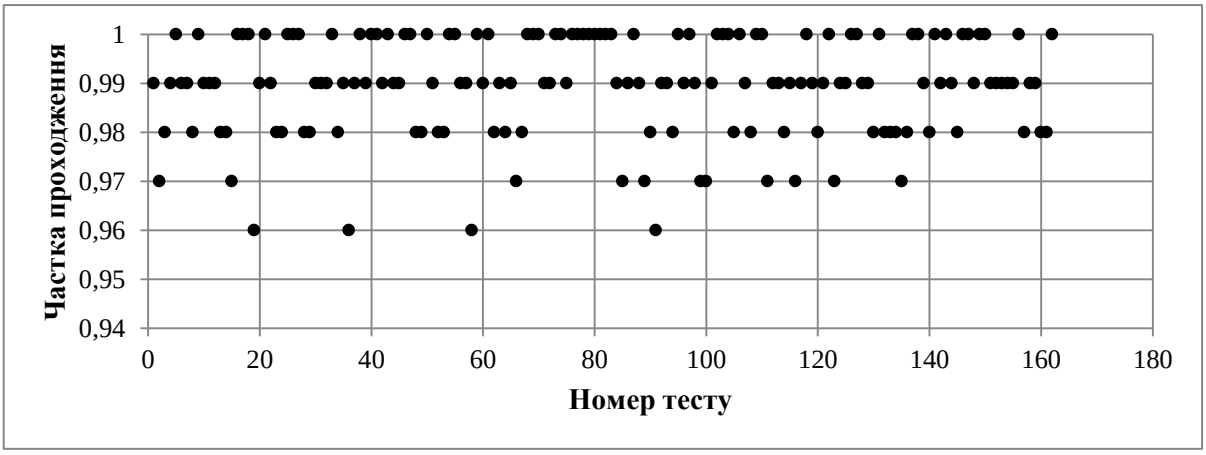

Рис. 5.3 Статистичний профіль вихідної послідовності чотирициклового шифруючого перетворення для розміру блока 512 бітів

Для тестування схеми розгортання ключів була обрана послідовність ключів шифрування $K_0 = 0$, $K_1 = 1$, …, $K_i = i$, $K_n = 100000$ (з максимальною збитковістю). Отримані циклові ключі формували вихідну послідовність, що тестувалася згідно з NIST STS.

Результати статистичного тестування сформованих вихідних послідовностей для довжини ключа 256 та 512 бітів наведені у додатку Б.

Статистичні профілі вихідних послідовностей циклових ключів для довжини ключа 256 та 512 бітів наведені на рисунку 5.4 та рисунку 5.5 відповідно.

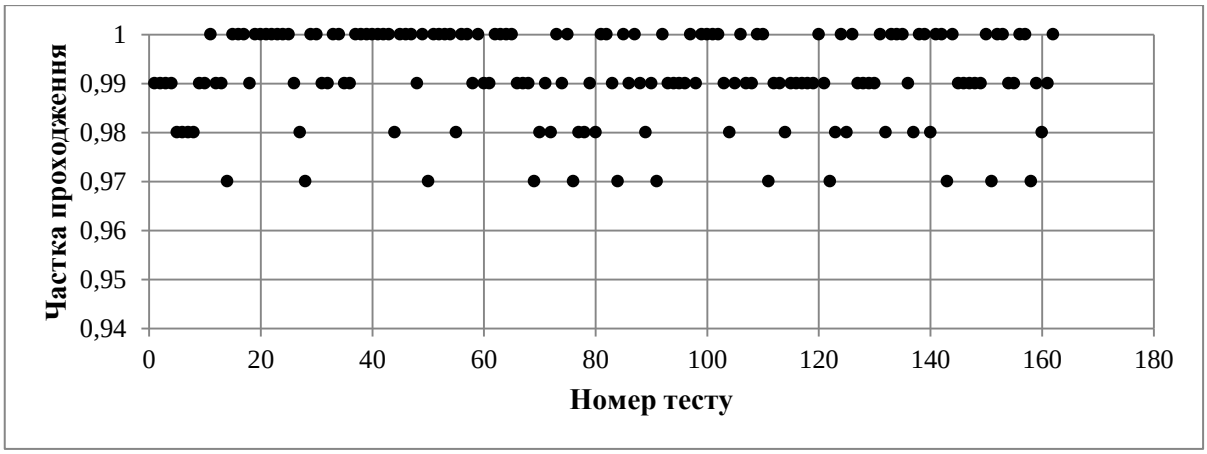

Рис. 5.4 Статистичний профіль вихідної послідовності циклових ключів для довжини ключа 256 бітів

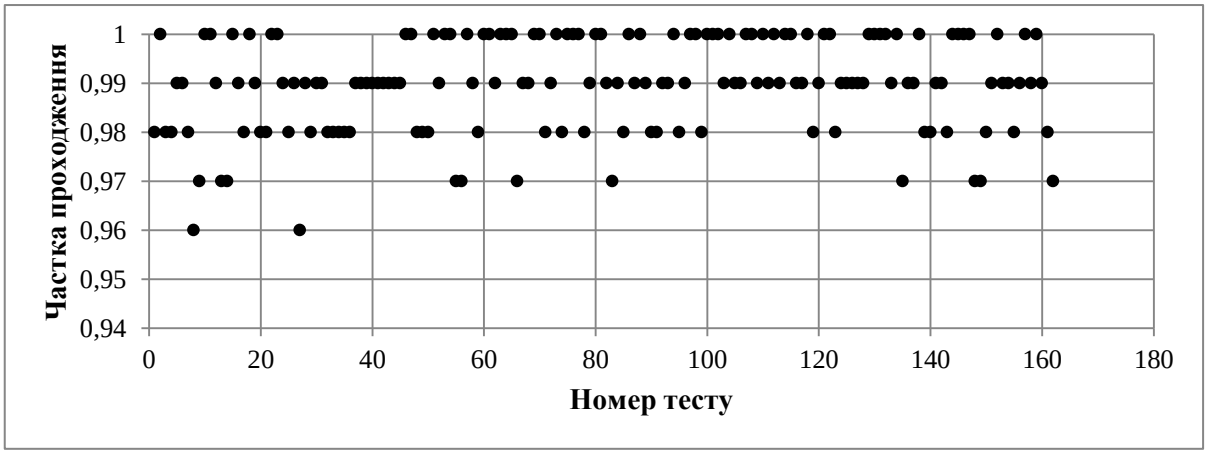

Рис. 5.5 Статистичний профіль вихідної послідовності циклових ключів для довжини ключа 512 бітів

5.2 Лавинні показники блокового шифру «Кипарис»

Лавинний ефект [137] є важливою властивістю шифру, яка означає, що зміна малої кількості бітів у відкритому тексті призведе до «лавинної» зміни значень бітів шифртексту. В ітеративних алгоритмах лавинний ефект досягається завдяки тому, що на кожному циклі зміна одного вхідного біта призводить до зміни декількох вихідних бітів. Якщо блоковий шифр не володіє достатнім лавинним ефектом, криптоаналітик може зробити припущення щодо

вхідної інформації, спираючись на вихідну інформацію, тому досягнення лавинного ефекту є важливою метою при розробці блокового шифру.

Вважається, що алгоритм задовольняє лавинному критерію, якщо зміна одного біта відкритого тексту призводить до зміни не менше половини бітів шифртексту.

Для оцінки лавинного ефекту шифру «Кипарис» були обчислені наступні показники:

- мінімум математичного сподівання кількості вихідних бітів, що змінилися при зміні одного вхідного біта для N блоків даних у результаті зашифрування k циклів;
- максимум математичного сподівання кількості вихідних бітів, що змінилися при зміні одного вхідного біта для N блоків даних у результаті зашифрування k циклів;
- мінімум середньоквадратичного відхилення кількості вихідних бітів, що змінилися при зміні одного вхідного біта для N блоків даних у результаті зашифрування k циклів;
- максимум середньоквадратичного відхилення кількості вихідних бітів, що змінилися при зміні одного вхідного біта для N блоків даних у результаті зашифрування k циклів.

Викладемо методику обчислення лавинних показників шифру для k циклів зашифрування.

а) Для дослідження лавинного ефекту випадковим чином обирається n = 100000 відкритих текстів (блоків даних) $B^{(1)}$, $B^{(2)}$,…, $B^{(n)}$.

б) У кожному $B^{(i)}$ блоку змінюється перший біт $B_1^{(i)}$, і запам’ятовується кількість змінених бітів шифртексту $\mu(B_1^{(i)})$. Далі знаходиться математичне сподівання $\bar{\mu}_1$ по всім значенням $\mu(B_1^{(i)})$, де $i = 1..n$.

в) Крок 2 повторюється для другого біту блока даних і т.д. Таким чином знаходяться значення математичного сподівання кількості змінених біт на виході для кожного біту: $\bar{\mu}_1, \bar{\mu}_2, ..., \bar{\mu}_k$, де k = 256 або 512 в залежності від варіанту шифру.

г) Серед значень $\bar{\mu}_1, \bar{\mu}_2, ..., \bar{\mu}_k$ знаходиться мінімум та максимум.

д) Аналогічним чином відбувається обчислення мінімуму та максимуму середньоквадратичного відхилення.

У таблиці 5.1 наведені результати обчислення лавинних показників для шифру «Кипарис-256».

Таблиця 5.1

Лавинні показники шифру «Кипарис-256»

Кількість циклів шифрування	Показник	Значення
1	Мінімум мат. сподівання	1
	Максимум мат. сподівання	65,0254
	Мінімум середньокв. відхилення	0
	Максимум середньокв. відхилення	49,8347
2	Мінімум мат. сподівання	62,3417
	Максимум мат. сподівання	128,016
	Мінімум середньокв. відхилення	32,1742
	Максимум середньокв. відхилення	81,6093
3	Мінімум мат. сподівання	125,375
	Максимум мат. сподівання	128,06
	Мінімум середньокв. відхилення	63,3573
	Максимум середньокв. відхилення	82,1095
4	Мінімум мат. сподівання	127,929
	Максимум мат. сподівання	128,079
	Мінімум середньокв. відхилення	63,2875
	Максимум середньокв. відхилення	64,6699
5	Мінімум мат. сподівання	127,926
	Максимум мат. сподівання	128,09
	Мінімум середньокв. відхилення	63,2186
	Максимум середньокв. відхилення	64,8311
6	Мінімум мат. сподівання	127,941
	Максимум мат. сподівання	128,078
	Мінімум середньокв. відхилення	63,2596
	Максимум середньокв. відхилення	64,7305
7	Мінімум мат. сподівання	127,94
	Максимум мат. сподівання	128,066
	Мінімум середньокв. відхилення	63,1499
	Максимум середньокв. відхилення	64,7323
8	Мінімум мат. сподівання	127,927
	Максимум мат. сподівання	128,064
	Мінімум середньокв. відхилення	63,2817
	Максимум середньокв. відхилення	64,7861

Продовження таблиці 5.1

Кількість циклів шифрування	Показник	Значення
9	Мінімум мат. сподівання	127,924
	Максимум мат. сподівання	128,072
	Мінімум середньокв. відхилення	63,2531
	Максимум середньокв. відхилення	64,7662
10	Мінімум мат. сподівання	127,941
	Максимум мат. сподівання	128,075
	Мінімум середньокв. відхилення	63,2797
	Максимум середньокв. відхилення	64,778

У таблиці 5.2 наведені результати обчислення лавинних показників для шифру «Кипарис-512».

Таблиця 5.2

Лавинні показники шифру «Кипарис-512»

Кількість циклів шифрування	Показник	Значення
1	Мінімум мат. сподівання	1
	Максимум мат. сподівання	161,034
	Мінімум середньокв. відхилення	0
	Максимум середньокв. відхилення	417,279
2	Мінімум мат. сподівання	98,0896
	Максимум мат. сподівання	256,052
	Мінімум середньокв. відхилення	63,6531
	Максимум середньокв. відхилення	481,193
3	Мінімум мат. сподівання	225,032
	Максимум мат. сподівання	256,081
	Мінімум середньокв. відхилення	126,813
	Максимум середньокв. відхилення	482,083
4	Мінімум мат. сподівання	255,911
	Максимум мат. сподівання	256,084
	Мінімум середньокв. відхилення	126,4
	Максимум середньокв. відхилення	129,707
5	Мінімум мат. сподівання	255,915
	Максимум мат. сподівання	256,1
	Мінімум середньокв. відхилення	125,956
	Максимум середньокв. відхилення	129,338
6	Мінімум мат. сподівання	255,862
	Максимум мат. сподівання	256,096
	Мінімум середньокв. відхилення	126,708
	Максимум середньокв. відхилення	129,394

Продовження таблиці 5.2

Кількість циклів шифрування	Показник	Значення
7	Мінімум мат. сподівання	255,915
	Максимум мат. сподівання	256,102
	Мінімум середньокв. відхилення	126,838
	Максимум середньокв. відхилення	129,639
8	Мінімум мат. сподівання	255,889
	Максимум мат. сподівання	256,096
	Мінімум середньокв. відхилення	126,625
	Максимум середньокв. відхилення	129,672
9	Мінімум мат. сподівання	255,9
	Максимум мат. сподівання	256,078
	Мінімум середньокв. відхилення	126,646
	Максимум середньокв. відхилення	129,51
10	Мінімум мат. сподівання	255,911
	Максимум мат. сподівання	256,109
	Мінімум середньокв. відхилення	126,321
	Максимум середньокв. відхилення	129,6
11	Мінімум мат. сподівання	255,912
	Максимум мат. сподівання	256,1
	Мінімум середньокв. відхилення	126,691
	Максимум середньокв. відхилення	129,297
12	Мінімум мат. сподівання	255,909
	Максимум мат. сподівання	256,138
	Мінімум середньокв. відхилення	126,219
	Максимум середньокв. відхилення	129,697
13	Мінімум мат. сподівання	255,895
	Максимум мат. сподівання	256,103
	Мінімум середньокв. відхилення	126,53
	Максимум середньокв. відхилення	129,524
14	Мінімум мат. сподівання	255,89
	Максимум мат. сподівання	256,108
	Мінімум середньокв. відхилення	126,607
	Максимум середньокв. відхилення	129,597

Як видно з таблиць 5.1-5.2, алгоритми блокового шифрування «Кипарис-256» та «Кипарис-512» задовольняють лавинному критерію вже після 4-го циклу.

5.3 Оцінка швидкодії блокового шифру «Кипарис» та порівняння з відомими малоресурсними алгоритмами

У ході досліджень на різних програмно-апаратних платформах була оцінена швидкодія алгоритмів «Кипарис-256» та «Кипарис-512» та порівняна зі швидкодією таких блокових шифрів як AES, SPECK, SPARX та ДСТУ ГОСТ 28147:2009.

5.3.1 Методика вимірювання швидкодії блокових шифрів

Для вимірювання швидкодії блокових шифрів використовувалась методика та програмний код, написаний мовою програмування C++, що застосовувались для дослідження продуктивності блокового шифру «Калина» [138]. З метою отримання точних та достовірних результатів, методика передбачає багатократне (наприклад, восьмикратне) зашифрування блоку пам’яті фіксованого розміру (наприклад, 1 ГБ) у режимі простої заміни. Блок пам’яті складається з N блоків даних, де N залежить від розміру вхідного блока алгоритму шифрування. Кожен з N блоків представляється у вигляді масиву 64-бітових беззнакових цілих чисел (розмірність масиву залежить від розміру блока, яким оперує шифр).

Для кожного з алгоритмів N задається константою, наприклад, *number_of_blocks_in_memory_128* для шифру з 128-бітовим розміром блока. Для ініціалізації блоку пам’яті певного розміру використовується відповідна функція, наприклад, *InitMemoryEncryptionBlock128()*.

Для отримання поточного значення числа тактів процесору призначена функція *DetermineTime()*. Для отримання числа тактів в операційній системі Linux використовувалася функція *gettimeofday()* з бібліотеки <sys/time.h>. Для операційної системи Windows була написана власна функція *Mygettimeofday()*.

На основі значень системного часу до початку та після закінчення виконання програмного блоку, що реалізує зашифрування, було обчислено швидкість зашифрування у Мбіт/с.

Вимірювання швидкодії блокових шифрів здійснювалося на наступних платформах:

– процесор Intel Core i7-7500U з тактовою частотою 2,7-2,9 GHz, операційна система Windows 10 x32;

– процесор Intel Core i7-7500U з тактовою частотою 2,7-2,9 GHz, операційна система Windows 10 x64;

– процесор Intel Core i5-4670U з тактовою частотою 3,4 GHz, операційна система Linux (64-bits);

– процесор Mediatek MT6582 з тактовою частотою 1,3 GHz, 4 ядра ARM Cortex-A7, операційна система Android 4.2.2 Jelly Bean (32-bits);

– процесор Exynos 7880 з тактовою частотою 1,9 GHz, 8 ядер ARM Cortex-A53, операційна система Android 8.0.0 (64-bits).

У зв’язку із тим, що процесори, використовувані у мобільних пристроях, мають набагато нижчу продуктивність, для вимірювання швидкодії алгоритмів на ОС Android замість 8 ГБ пам’яті шифрувалося 8МБ.

Окрім блокових шифрів «Кипарис-256» та «Кипарис-512» для отримання оцінок щодо швидкості зашифрування було обрано наступні блокові шифри:

- AES-256;
- SPECK-64/128;
- SPECK-128/128;
- SPARX-128/128;
- ДСТУ ГОСТ 28147: 2009.

AES та ГОСТ-28147-89 було обрано як найбільш відомі та перевірені часом алгоритми, а SPECK – з міркувань того, що цей шифр, подібно до шифру «Кипарис», заснований на мережі Фейстеля з ARX-перетворенням у якості циклової функції. SPARX був обраний для порівняння як перший (разом із LAX) доказово стійкий малоресурсний блоковий шифр.

Реалізація шифрів «Кипарис-256» та «Кипарис-512» мовою програмування C++ наведена у Додатку В.

Обчислення швидкодії блокового шифру AES-256 здійснювалось для оптимізованої реалізації з використанням таблиць передобчислень, представленої в [138]. Реалізацію шифру ДСТУ ГОСТ 28147:2009 було обрано з того ж джерела [138].

Також були використані реалізації блокових шифрів SPECK-64/128 та SPARX-128/128 з бібліотеки FELICS [139], що містить оптимізовані реалізації найбільш відомих малоресурсних алгоритмів. Зазначимо, що не всі реалізації з цієї бібліотеки видаються достатньо оптимізованими з точки зору швидкодії (принаймні ті, що орієнтовані на застосування на процесорах загального призначення). Так, наприклад, внесення незначних змін у програмний код, що реалізує блоковий шифр SPECK-64/128 (заміна викликів функцій у процедурі зашифрування простою підстановкою коду, який вони містять), дозволило підвищити швидкодію у декілька разів.

Для SPECK-128/128 було обрано реалізацію, запропоновану авторами [140], перевагою якої є дуже компактний програмний код (функція зашифрування містить менше десяти рядків коду).

5.3.2 Результати вимірювання швидкодії блокових шифрів

Результати вимірювання швидкодії шифрів на 32-бітовій платформі Windows 10 наведені в табл. 5.3.

На платформі Windows 10 x32 шифри SPECK-128/128, SPARX-128/128 та ДСТУ ГОСТ 28147:2009 показали близький результат у межах 600-750 Мбіт/сек. Далі йдуть блокові шифри AES-256 та «Кипарис-512» зі швидкістю порядку 1,5 Гбіт/сек. Реалізація блокового шифру SPECK-64/128 [119] забезпечує швидкість шифрування порядку 3 Гбіт/сек.

Таблиця 5.3

Порівняння продуктивності малоресурсних блокових шифрів на платформі Windows 10 x32, процесор Intel Core i7-7500U

Блоковий шифр	Розмір блока, бітів	Довжина ключа, бітів	Швидкість зашифрування, Мбіт/сек	Реалізація
«Кипарис-256»	256	256	3472,04	Додаток В
«Кипарис-512»	512	512	1555,54	Додаток В
AES-256	128	256	1441,85	[138]
SPECK	64	128	3059,16	[139]
SPECK	128	128	748,96	[140]
SPARX	128	128	661,83	[139]
ДСТУ ГОСТ 28147:2009	64	128	603,4	[138]

Значна різниця у швидкості зашифрування між реалізаціями шифрів SPECK-64/128 та SPECK-128/128 пояснюється тим, що SPECK-64/128 оперує 32-бітовим блоком, а SPECK-128/128 – 64-бітовим блоком, тому на 32-бітовій платформі SPECK-64/128 значно виграє у швидкодії. Теж саме стосується і шифрів «Кипарис-256» та «Кипарис-512».

Найкращий результат на 32-бітовій платформі Windows 10 показав блоковий шифр «Кипарис-512», його швидкодія склала майже 3,5 Гбіт/сек.

Результати вимірювання швидкодії шифрів на 64-бітових платформах Windows 10 та Linux наведені в таблицях 5.4-5.5 відповідно.

На 64-бітовій платформі Windows 10 найкращий результат очікувано показали блокові шифри «Кипарис-512» (порядку 5 Гбіт/сек) та SPECK-128/128 (порядку 4,8 Гбіт/сек), які обробляють 64-бітові блоки даних. Крім того, перевагою блоковий шифру «Кипарис-512» є надвисокий рівень стійкості шифру, що дозволить йому застосовуватися у пост квантовий період. Результати для інших алгоритмів значно не змінилися у порівнянні з

результатами, отриманими на 32-бітовій платформі, лише швидкість зашифрування алгоритму SPARX-128/128 помітно зросла з 749 до 937 Мбіт/сек.

Таблиця 5.4

Порівняння продуктивності малоресурсних блокових шифрів на платформі Windows 10 x64, процесор Intel Core i7-7500U

Блоковий шифр	Розмір блока, бітів	Довжина ключа, бітів	Швидкість зашифрування, Мбіт/сек	Реалізація
«Кипарис-256»	256	256	3502,46	Додаток В
«Кипарис-512»	512	512	4942,77	Додаток В
AES-256	128	256	1653,79	[138]
SPECK	64	128	3038,03	[139]
SPECK	128	128	4786,81	[140]
SPARX	128	128	936,373	[139]
ДСТУ ГОСТ 28147:2009	64	128	526,583	[138]

Таблиця 5.5

Порівняння продуктивності малоресурсних блокових шифрів на платформі Linux (64 bit), процесор Intel Core i5-4670U

Блоковий шифр	Розмір блока, бітів	Довжина ключа, бітів	Швидкість зашифрування, Мбіт/сек	Реалізація
«Кипарис-256»	256	256	8418,3	Додаток В
«Кипарис-512»	512	512	5356,9	Додаток В
AES-256	128	256	1920,75	[138]
SPECK	64	128	3179,49	[139]
SPECK	128	128	5276,39	[140]
SPARX	128	128	1049,53	[139]
ДСТУ ГОСТ 28147:2009	64	128	640,469	[138]

Згідно з результатами, представленими у табл. 5.5, співвідношення між швидкостями досліджуваних алгоритмів на 64-бітовій платформі Linux приблизно таке саме, як і на Windows 10 x64. Головною відмінністю є значне покращення результату для шифру «Кипарис-256», швидкість якого перевершила 8 Гбіт/сек, що може бути пов’язано з використанням нової версії компілятора gcc version 5.4.0, який виконує певну оптимізацію. Далі йдуть шифри «Кипарис-512» та SPECK-128/128 з приблизно однаковим результатом у більш ніж 5 Гбіт/сек.

У таблицях 5.6-5.7 представлено результати вимірювання швидкодії обраних алгоритмів на мобільній платформі Android.

На мобільних платформах «Кипарис» також продемонстрував високі результати. Так, наприклад, на процесорі Exynos 7880 «Кипарис-256» та «Кипарис-512» мають майже 1,3 Гбіт/сек та 1 Гбіт/сек відповідно, в той час, коли найближчий конкурент SPECK-64/128 має лише 640 Мбіт/сек.

Таблиця 5.6.

Порівняння продуктивності малоресурсних блокових шифрів на платформі Android 4.2.2 (32-bits), процесор Mediatek MT6582

Блоковий шифр	Розмір блока, бітів	Довжина ключа, бітів	Швидкість зашифрування, Мбіт/сек	Реалізація
«Кипарис-256»	256	256	592	Додаток В
«Кипарис-512»	512	512	467	Додаток В
AES-256	128	256	109	[138]
SPECK	64	128	599	[139]
SPECK	128	128	205	[140]
SPARX	128	128	71	[139]

Таблиця 5.7

Порівняння продуктивності малоресурсних блокових шифрів на платформі Android 8.0.0 (64-bits), процесор Exynos 7880

Блоковий шифр	Розмір блока, бітів	Довжина ключа, бітів	Швидкість зашифрування, Мбіт/сек	Реалізація
«Кипарис-256»	256	256	1263	Додаток В
«Кипарис-512»	512	512	999	Додаток В
AES-256	128	256	183	[138]
SPECK	64	128	639	[139]
SPECK	128	128	422	[140]
SPARX	128	128	145	[139]

На рисунку 5.6 у графічному вигляді подано результати, представлені в таблицях 5.3-5.5.

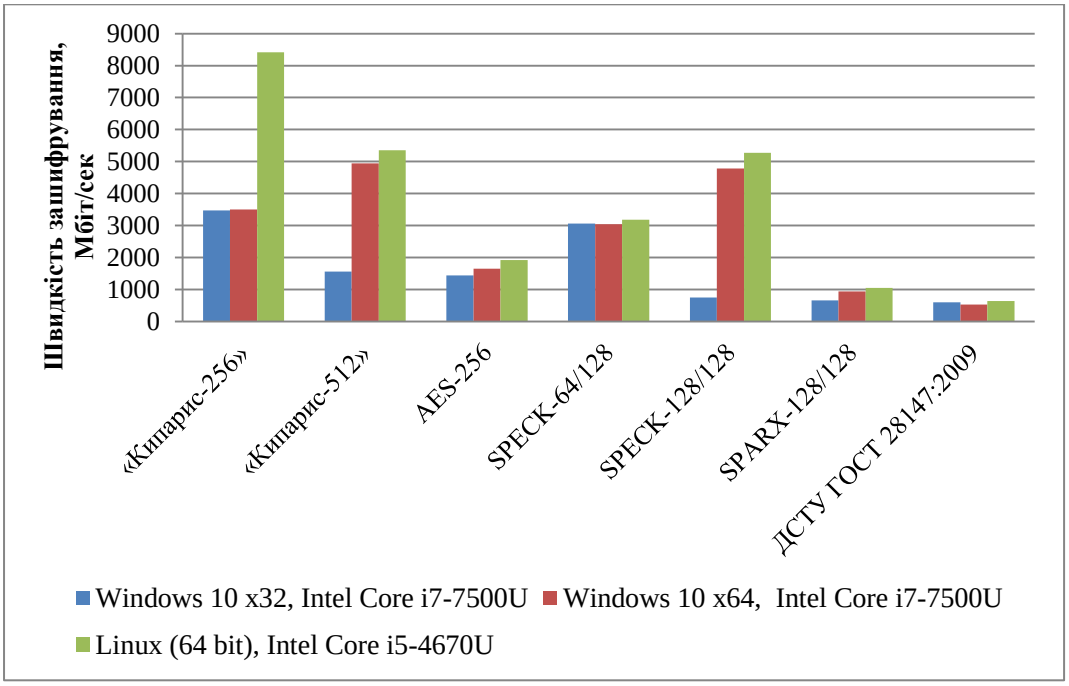

Рис. 5.6 Порівняння швидкодії блокового шифру «Кипарис» з відомими блоковими шифрами на платформах загального призначення

На рисунку 5.7 у графічному вигляді подано результати, представлені в таблицях 5.6-5.7.

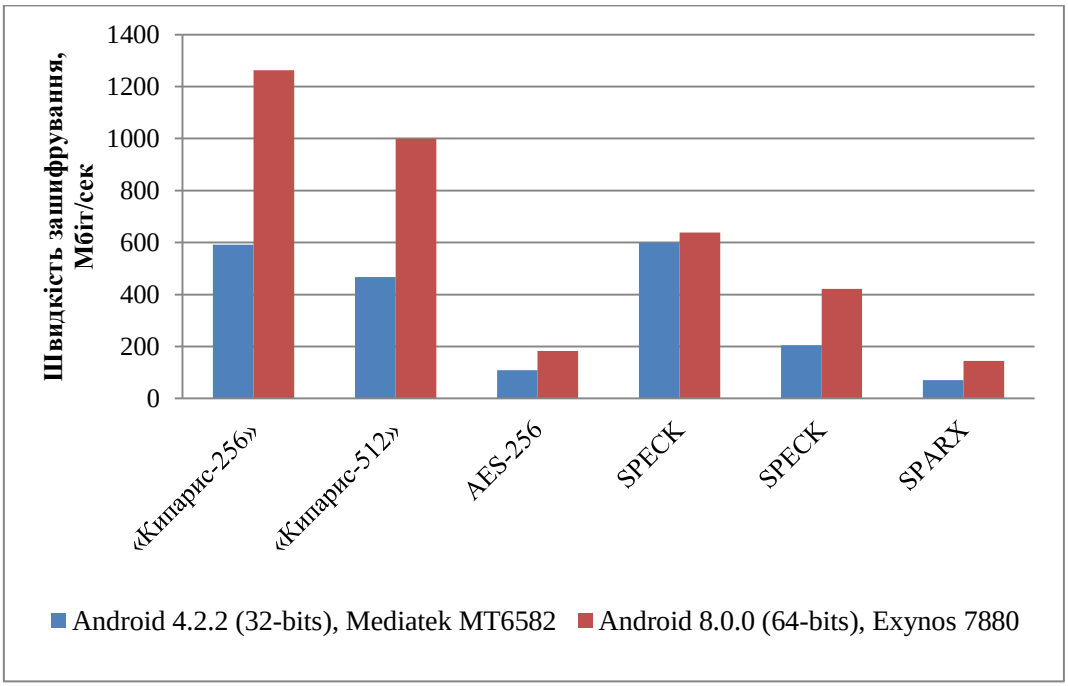

Рис. 5.7 Порівняння швидкодії блокового шифру «Кипарис» з відомими блоковими шифрами на платформі Android

Висновки до розділу 5

1. Дослідження статистичних властивостей блокового шифру «Кипарис» показало, що шифруюче перетворення та схема розгортання ключів алгоритмів «Кипарис-256» та «Кипарис-512» задовольняють вимогам зі статистичного тестування випадкових послідовностей NIST STS.

2. Дослідження лавинних показників блокового шифру «Кипарис» показало, що «Кипарис-256» (число циклів дорівнює 10) та «Кипарис-512» (число циклів дорівнює 14) відповідають вимогам щодо лавинного ефекту починаючи з чотирьох циклів шифрування.

3. Порівняння швидкодії блокового шифру «Кипарис» зі швидкодією відомих малоресурсних алгоритмів здійснювалося із використанням програмної реалізації, розробленої мовою програмування C++, що дозволяє отримати

високу продуктивність (на базі нативного коду) за рахунок використання машинно-незалежної мови програмування. Вимірювання швидкодії блокових шифрів здійснювалося на платформах Windows, Linux та Android.

4. Блоковий шифр «Кипарис» продемонстрував високу продуктивність на всіх досліджуваних програмно-апаратних платформах:

а) на платформі Windows 10 з 32-бітовою архітектурою найкращий результат показав шифр «Кипарис-256» (трохи менше 3,5 Гбіт/сек), за ним слідує SPECK-128/128 (3 Гбіт/сек), а шифр AES-256 відстає майже у 2,5 рази (1,5 Гбіт/сек);

б) на платформі Windows 10 з 64-бітовою архітектурою блоковий найкращий результат показав шифр «Кипарис-512» (майже 5 Гбіт/сек), якому незначно поступився за швидкістю шифр SPECK-128/128 (4,8 Гбіт/сек); при цьому блоковий шифр «Кипарис-512» забезпечує надвисокий рівень стійкості;

в) на платформі Linux з 64-бітовою архітектурою блоковий шифр «Кипарис-256» показав надвисокий результат зі швидкодії (понад 8 Гбіт/сек), далі з приблизно однаковим результатом слідують шифри «Кипарис-512» та SPECK-128/128 (понад 5 Гбіт/сек);

г) на платформі Android 8.0.0 найкращими також були блокові шифри «Кипарис-256» та «Кипарис-512» (1,3 Гбіт/сек та 1 Гбіт/сек відповідно), за якими слідує SPECK-64/128 з результатом у 0,6 Гбіт/сек.

5. Загалом, з точки зору продуктивності та зручності реалізації на різних програмно-апаратних платформах алгоритм «Кипарис» має наступні переваги:

а) два варіанти шифру («Кипарис-256» та «Кипарис-512») орієнтовані на 32-бітову та 64-бітову архітектури відповідно;

б) висока швидкодія перетворень незалежно від платформи, що використовується;

в) компактна реалізація незалежно від платформи, що використовується (сервер, робоча станція або мобільний пристрій);

г) мінімальний необхідний об’єм пам’яті для швидкодіючої реалізації, відсутність необхідності у таблицях передобчислень;

д) можливість організації ефективних захищених високошвидкісних каналів зв’язку між мобільними системами та серверами, у тому числі тими, що використовують апаратні прискорювачі.

Результати досліджень даного розділу наведено в публікаціях здобувача: [141-142].

ВИСНОВКИ

У ході дисертаційних досліджень отримано наступні наукові та практичні результи.

1. Протягом останніх двадцяти років проведено ряд конкурсів та прийнято ряд стандартів з симетричного блокового шифрування. Так, наприклад, у Сполучених Штатах Америки був стандартизований блоковий шифр AES, а в Україні – «Калина», що забезпечують високий рівень криптографічної стійкості. В останній час багато уваги приділяється розробці малоресурсних примітивів для застосування у пристроях з обмеженою кількістю споживання енергії. Наразі NIST проводить конкурс на розробку малоресурсного алгоритму AEAD, на який вже була подана низка кандидатів. Серед відомих існуючих малоресурсних блокових шифрів можна виділити SPECK, SPARX, TEA, XTEA, LEA, PRESENT, CLEFIA та ін. Більшість з цих шифрів мають високу швидкодію, проте підтримують довжину ключа, недостатню для забезпечення стійкості у постквантовий період. Окремим питанням є обґрунтування стійкості малоресурсних примітивів, зокрема, заснованих на ARX-перетворенні. Ці та інші питання частково вирішено у дисертаційній роботі.

2. Удосконалений метод градієнтного спуску, що застосовується для генерації S-блоків, шляхом оптимізації з точки зору швидкодії алгоритму перевірки сформованої підстановки на відповідність ряду криптографічних показників. Можливість оптимізації обумовлюється тим, що критерії відбору підстановок є частково взаємозалежними, тому час перевірки підстановки на відповідність критеріям суттєво залежить від порядку їх застосування. Запропонований аналітичний вираз, що дозволяє визначити порядок застосування критеріїв, за якого час перевірки, а значить, і час генерації підстановки, буде мінімальним.

3. Застосування удосконаленого методу градієнтного спуску дозволяє в п’ять разів скоротити середній час генерації оптимальних байтових S-блоків з нелінійністю 104, алгебраїчним імунітетом 3 та δ-рівномірністю 8. Середній час генерації оптимальних підстановок за допомогою програмної реалізації з урахуванням запропонованого вдосконалення склав 30 хвилин. Крім того, представлений підхід не залежить від самого алгоритму генерації та може бути застосований до будь-якого евристичного методу генерації S-блоків.

4. Розроблено модель оцінки колізійних властивостей неін’єктивних схем розгортання циклових ключів, в рамках якої сформульована та доведена теорема про те, що для повномасштабного шифру ймовірність співпадіння потужностей множини послідовностей циклових ключів, які формуються неін’єктивною схемою розгортання ключів, і множини ключів шифрування практично дорівнює 1. Практично це означає, що складність атак переборного типу на неін’єктивні схеми розгортання ключів у порівнянні з ін’єктивними схемами не знижується. При цьому неін’єктивні схеми розгортання ключів забезпечують додаткову стійкість до атак на реалізацію та деяких інших криптоаналітичних атак.

Отриманий результат дозволяє обґрунтувати більш високий рівень стійкості діючого національного стандарту симетричного блокового перетворення ДСТУ 7624:2014, який також застосовує неін’єктивну схему розгортання циклових ключів.

5. Розроблена математичну модель оцінки стійкості визначеного класу ARX-шифрів до диференційного криптоаналізу дозволяє отримати вираз для обчислення середньої (за ключами) ймовірності диференційної характеристики шифру та зробити обґрунтоване припущення щодо значень вхідних різниць, які при проходженні через цикл шифрування формують характеристику, що має високу ймовірність. Модель включає припущення про те, що шифр є марковським, про ймовірність перетворення диференційних різниць на модульних суматорах, про зв’язок кількості активних біт у вхідній різниці з ймовірністю диференційної характеристики, а також містить вирази для

обчислення ймовірностей одноциклових та багатоциклових диференційних характеристик.

6. Розроблений постквантовий малоресурсний блоковий шифр «Кипарис», що заснований на ARX-перетворенні. Алгоритм підтримує довжину блока (ключа) 256 та 512 бітів, що дозволяє забезпечити високий та надвисокий рівні криптографічної стійкості. Блоковий шифр «Кипарис-256» орієнтований на використання на 32-бітових платформах, в той час як «Кипарис-512» – на 64-бітових платформах.

7. Розроблено методи пошуку диференційних характеристик циклової функції визначеного класу ARX-шифрів (метод пошуку «у двох напрямках» та оптимізований метод пошуку диференційної характеристики з високою ймовірністю), що дозволяють знайти найбільш ймовірні одноциклові диференційні характеристики. Метою всіх трьох підходів є активізація найменшої кількості біт на входах суматорів циклової функції, що, в свою чергу, збільшує ймовірність заданого перетворення. Останній із запропонованих методів дозволив знайти диференційну характеристику на циклову функцію блокового шифру «Кипарис» з ймовірністю, що дорівнює ¼.

8. Удосконалені методи пошуку багатоциклових диференційних характеристик визначеного класу ARX-шифрів дозволяють знайти високоймовірні *r*-циклові диференційні характеристики в умовах використання обмежених обчислювальних ресурсів. Застосування методу пошуку багатоциклових диференційних характеристик, заснованого на побудуванні множини високоймовірнісних одноциклових диференційних характеристик, до блокового шифру «Кипарис-256» показало, що побудовані одноциклові ДХ, вихідні різниці яких мають малу вагу Гемінга (а значить і достатньо високу ймовірність), не можуть бути продовжені для побудування багатоциклових ДХ з високою ймовірністю.

9. В рамках запропонованої математичної моделі доведено стійкість блокового шифру «Кипарис-256» до диференційного криптоаналізу відповідно до вимог практичного критерію, для якого знайдена 6-циклова диференційна

характеристика з ймовірністю, значно меншою за ймовірність атаки повного перебору.

10. Дослідження статистичних властивостей блокового шифру «Кипарис» показало, що шифруюче перетворення та схема розгортання ключів алгоритмів «Кипарис-256» та «Кипарис-512» задовольняють вимогам зі статистичного тестування випадкових послідовностей NIST STS. Дослідження лавинних показників блокового шифру «Кипарис» показало, що «Кипарис-256» (число циклів дорівнює 10) та «Кипарис-512» (число циклів дорівнює 14) відповідають вимогам щодо лавинного ефекту починаючи з чотирьох циклів шифрування.

11. Блоковий шифр «Кипарис» забезпечує високу продуктивність на таких платформах як Windows, Linux та Android, перевершуючи такі відомі алгоритми як AES, SPECK, SPARX та ін. На платформі Windows 10 з 32-бітовою архітектурою швидкість блокового шифру «Кипарис-256» складає 3,5 Гбіт/сек, а блокового шифру «Кипарис-512» – 1,5 Гбіт/сек. На платформі Windows 10 з 64-бітовою архітектурою швидкість блокового шифру «Кипарис-256» складає 3,5 Гбіт/сек, а блокового шифру «Кипарис-512» – 5 Гбіт/сек. На платформі Linux з 64-бітовою архітектурою блоковий шифр «Кипарис-256» має понад 8 Гбіт/сек. На платформі Android найкращими блокові шифри «Кипарис-256» та «Кипарис-512» мають 1,3 Гбіт/сек та 1 Гбіт/сек відповідно.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про встановлення Вимог до засобів електронної ідентифікації, рівнів довіри до засобів електронної ідентифікації для їх використання у сфері електронного урядування : наказ Державного агентства з питань електронного урядування України від 27.11.2018 р. № 86. URL: https://zakon.rada.gov.ua/laws/show/z1462-18#Text (дата звернення 15.08.2020).

2. Про забезпечення реалізації деяких питань цифрового розвитку : наказ Державного агентства з питань електронного урядування України від 09.04.2019 р. № 24. URL: https://zakononline.com.ua/documents/show/110656___110656 (дата звернення 15.08.2020).

3. Atzori L., Iera A., Morabito G. The internet of things: A survey. *Computer networks*. 2010. V. 54.15. P. 2787–2805.

4. Про захист інформації в інформаційно-телекомунікаційних системах : Закон України від 05.07.1994 р. № 80/94-ВР. *Відомості Верховної Ради України.* 1994. № 31. Ст. 286.

5. Загальні положення щодо захисту інформації в комп'ютерних системах від несанкціонованого доступу : НД ТЗІ 1.1-002-99 від 28.04.1999 р. № 22.

6. Post-Quantum Cryptography / ed. by Daniel J. Bernstein, Johannes Buchmann, Erik Dahmen. Berlin, Heidelberg : Springer-Verlag, 2009. 245 p.

7. Bernstein Daniel J. Grover vs. McEliece. *Post-Quantum Cryptography* : Proceedings of the Third International Workshop, 25–28 May 2010, Darmstadt, Germany. Berlin, Heidelberg : Springer, 2010. P. 73–80.

8. ДСТУ 7624:2014. Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення [Чинний від 01–07–2015]. Вид. офіц. Київ : Мінекономрозвитку України, 2015. 119 с.

9. Принципи побудови і основні властивості нового національного стандарту блокового шифрування України / Р. Олійников та ін. *Захист інформації*. 2015. Т. 17, №2. С. 142–157.

10. Казимиров О. В. Методи та засоби генерації нелінійних вузлів заміни для симетричних криптоалгоритмів : дис. … канд. техн. наук : 05.13.21 / Харківський національний університет радіоелектроніки. Харків, 2014. 190 с.

11. Junod P., Vaudenay S. FOX: a new family of block ciphers. *Selected Areas in Cryptography* : International Workshop, 9–10 August 2004, Waterloo, Canada. Berlin, Heidelberg : Springer, 2005. P. 114–129.

12. Twofish: A 128-Bit Block Cipher / B. Schneier et al. *AES algorithm submission*. June 15, 1998. 68 p.

13. Lightweight cryptography. Project overview. *NIST* : веб-сайт. URL: https://csrc.nist.gov/projects/lightweight-cryptography (дата звернення: 15.08.2020).

14. Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process. URL: https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf (дата звернення: 15.08.2020).

15. Biryukov A., Velichkov V. Automatic Search for Differential Trails in ARX Ciphers (Extended Version). IACR Cryptology ePrint Archive: Report 2013/853. 2013. 32 p. URL: eprint.iacr.org/2013/853.pdf.

16. Design strategies for ARX with provable bounds: SPARX and LAX / Daniel Dinu et al. *International Conference on the Theory and Application of Cryptology and Information Security* : Springer, Berlin, Heidelberg, December 2016. P. 484–513.

17. Secure Hash Standard (SHS) : FIPS PUB 180-4. *National Institute of Standards and Technology*, August 1995.

18. BLAKE2: simpler, smaller, fast as MD5 / Jean-Philippe Aumasson et al. *International Conference on Applied Cryptography and Network Security* : Springer, Berlin, Heidelberg, 2013. P. 119–135.

19. The Skein hash function family / Niels Ferguson et al. *Submission to NIST (round 3)*, 2010. 7.7.5:3.

20. Recommendation for the entropy sources used for random bit generation / M. Sönmez Turan et al. *National Institute of Standards and Technology*, 2016. 76 p.

21. Dworkin M. Recommendation for block cipher modes of operation: The CCM mode for authentication and confidentiality : No. NIST Special Publication (SP) 800-38C (Withdrawn). *National Institute of Standards and Technology*, 2004. 28 p.

22. Standart, Data Encryption. Federal Information Processing Standards Publication 46 : National Bureau of Standards, US Department of Commerce. 1977. 17 p.

23. Biham E., Shamir A. Differential Cryptanalysis of DES-like Cryptosystem. *Journal of Cryptology*. 1991. Vol. 4. P. 3–72.

24. Matsui M. Linear cryptanalysis method for DES cipher. *Advances in Cryptology – Eurocrypt'93* : Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, 23–27 May 1993, Lofthus, Norway. Berlin, Heidelberg : Springer-Verlag, 1994. P. 386–397.

25. Cryptographic standards and guidelines. AES development : веб-сайт. URL: http://www.nist.gov/aes (дата звернення 15.08.2020).

26. Announcing request for candidate algorithm nominations for the Advanced Encryption Standard (AES) : Department of commerce, National Institute of Standards and Technology. URL: http://csrc.nist.gov/archive/aes/pre-round1/aes_9709.htm (дата звернення 15.08.2020).

27. Nechvatal, J., Barker, E., Bassham, L., et al. Report on the development of the Advanced Encryption Standard (AES). *Journal of Research of the National Institute of Standards and Technology*. 2001. Vol. 106, No. 3. C. 511–577.

28. MARS – A Candidate Cipher for AES / C. Burwick et al. *AES algorithm submission*. August 20, 1999. 63 p.

29. The RC6™ Block Cipher / R. Rivest et al. *AES algorithm submission*. June 1998. 21 p.

30. Daemen J., Rijmen V. AES Proposal: Rijndael. *AES algorithm submission*. September 3, 1999. 45 p.

31. Anderson R., Biham E., Knudsen L. Serpent: A Proposal for the Advanced Encryption Standard. *First Advanced Encryption Standard (AES) Conference*, Ventura, CA. 1998. 23 p.

32. Standard, Advanced Encryption. Federal Information Processing Standards Publication 197 / FIPS PUB, 46-3 : National Bureau of Standards, US Department of Commerce. 2001. 51 p.

33. NESSIE. New European Schemes for Signatures, Integrity and Encryption : NESSIE home page. URL: https://www.cosic.esat.kuleuven.be/nessie/ (дата звернення 15.08.2020).

34. CRYPTREC Cryptography Research and Evaluation Committees : CRYPTREC home page. URL: http://www.cryptrec.go.jp/english/ (дата звернення 15.08.2020).

35. Final report of European project number IST-1999-12324. New European Schemes for Signatures, Integrity, and Encryption : Springer-Verlag, 2004. 126 p.

36. Matsui M. Specification of MISTY1 – a 64-bit block cipher. Primitive submitted to NESSIE by E. Takeda, Mitsubishi. Sept. 2000. 14 p.

37. Camellia: A 128-bit Block Cipher Suitable for Multiple Platforms – Design and Analysis / K. Aoki et al. *Selected Areas in Cryptography*. Springer, Berlin, Heidelberg, 2000. P. 39–56.

38. Handschuh, H. SHACAL. Primitive submitted to NESSIE by Gemplus. Sept. 2000. 17 p.

39. Specifications of e-Government Recommended Ciphers: веб-сайт. URL: http://www.cryptrec.go.jp/english/method.html (дата звернення: 15.08.2020).

40. ГОСТ 28147–89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования [Введ. 01–07–1990]. Москва: Изд-во стандартов, 1989. 28 с.

41. Положення про проведення відкритого конкурсу криптографічних алгоритмів : Державна служба спеціального зв'язку та захисту інформації

України, Інститут кібернетики імені В.М. Глушкова Національної академії наук України. URL: http://www.dstszi.gov.ua/dstszi/control/uk/publish/printable_article ?art_id=48383 (дата звернення: 15.08.2020).

42. Проект национального стандарта Российской Федерации. Информационная технология. Криптографическая защита информации. Блочные шифры. Москва: Стандартинформ, 2015. 25 с.

43. СТБ 34.101.31–2011. Информационные технологии и безопасность. Защита информации. Криптографические алгоритмы шифрования и контроля целостности [Введ. 31–01–2011] . Минск, 2011. 35 с.

44. Cryptographic competitions. URL: https://competitions.cr.yp.to/index.html (дата звернення: 15.08.2020).

45. Lightweight cryptography. Round 2 Candidates. *NIST* : веб-сайт. URL: https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates (дата звернення: 15.08.2020).

46. Bellare M., Rogaway P. Introduction to modern cryptography. *UCSD CSE*. 2005. 283 p.

47. Menezes A. J., Oorschot P. C. Van, Vanstone S. A. Handbook of applied cryptography: *Handbook of Applied Cryptography*. CRC Press, 1996. 780 p.

48. Bogdanov A. Analysis and Design of Block Cipher Constructions: Dissertation. Bochum, 2009. 202 p.

49. Горбенко І. Д., Горбенко Ю. І. Прикладна криптологія. Теорія. Практика. Застосування : підручник для вищих навчальних закладів. Харків : Форт, 2013. 880 с.

50. Олейников Р. В. Методы анализа и синтеза перспективных симметричных криптографических преобразований : дис. ... д-ра техн. наук : 05.13.05. Харьков, 2013. 423 с.

51. PRESENT: An Ultra-Lightweight Block Cipher / A. Bogdanov et al. Springer, Berlin, Heidelberg, 2007. P. 450–466.

52. Massey J. SAFER K-64: A byte-oriented block-ciphering algorithm. *Fast Software Encryption* : Springer, Berlin, Heidelberg, 1993. P. 1–17.

53. Adams C. The CAST-128 Encryption Algorithm. URL: https://tools.ietf.org/html/rfc2144 (дата звернення: 15.08.2020).

54. Олейников Р.В., Кайдалов Д.С. Оценка сложности различения схемы Лей-Месси и случайной перестановки. *Прикладная радиоэлектроника*. 2012. Т. 11, № 2. С. 152–159.

55. Lai X., Massey J. L. A proposal for a new block encryption standard (IDEA): *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 1991. P. 389–404.

56. Shannon C.E. Communication Theory of Secrecy Systems. *Bell System Technical Journal*. 1949. Vol. 28. P. 656–715.

57. Wade Trappe and Lawrence C. Washington, Introduction to Cryptography with Coding Theory. Second edition. Pearson Prentice Hall, 2006.

58. Saarinen M. J. O. Cryptographic analysis of all 4×4-bit S-boxes. *International Workshop on Selected Areas in Cryptography*. Springer, Berlin, Heidelberg , 2011. P. 118–133.

59. Bernstein D. J. ChaCha, a variant of Salsa20. *Workshop Record of SASC*. 2008. Vol. 8. P. 3–5.

60. Beaulieu R., Treatman-Clark S., Shors D., et al. The SIMON and SPECK lightweight block ciphers: *Proceedings - Design Automation Conference*, Institute of Electrical and Electronics Engineers Inc., 2015. P. 1–6.

61. Hong D., Lee J. K., Kim D. C., Kwon D., Ryu K. H., Lee D. G. LEA: A 128-bit block cipher for fast encryption on common processors // International Workshop on Information Security Applications, August 2013. Springer, Cham. P. 3–27.

62. Сорока Л. С., Кузнецов А. А., Московченко В. И., Исаев С. А. Исследование дифференциальных свойств блочно-симметричных шифров. *Системи обробки інформації*. 2010. № 6(87). С. 286–294.

63. Oliynykov R., Kazymyrov O. An Impact Of S-box Boolean Function Properties To Strength Of Modern Symmetric Block Ciphers. *Радиотехника*. 2011. Вып. 166. С. 11–17.

64. Лисицкая И.В., Лисицкий К.Е., Родинко М.Ю., Головко И.А., Жариков И.И., Корниенко М.А., Кулеба М.В. Экспериментальные данные по определению динамических показателей прихода блочных симметричных шифров к состоянию случайной подстановки. *Радіоелектроніка, інформатика, управління*. 2017. № 1. С. 129–141.

65. Nyberg K. Perfect nonlinear S-boxes. *EUROCRYPT'91*: proceedings of the Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8–11, 1991. Berlin; Heidelberg: Springer, 1991. P. 378–386.

66. Kazymyrov O., Kazymyrova V., Oliynykov R. A Method For Generation Of High-Nonlinear S-boxes Based On Gradient Descent. IACR Cryptology ePrint Archive. Report 2013/578. 2013. 9 p.

67. Kelsey J., Schneier B., Wagner D. Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES. *Advances in Cryptology – CRYPTO'96* : Proceedings of Annual International Cryptology Conference, Aug., 1996. Vol. 1109 of Lecture Notes in Computer Science, N. Koblitz, Ed. Springer-Verlag, 1996. P. 237–252.

68. Biryukov A., Wagner D. Slide attacks. *Fast Software Encryption*. Berlin, Heidelberg : Springer, 1999. P. 245–259.

69. Biham E. New types of cryptanalytic attacks using related keys. *Journal of Cryptology*. Springer-Verlag, 1994. V. 7. No. 4. P. 229–246.

70. Lai X., Massey J. L., Murphy S. Markov ciphers and differential cryptanalysis // Advances in Cryptology–EUROCRYPT'91: Proceedings of Workshop on the Theory and Application of of Cryptographic Techniques, 8–11 Apr., 1991, Brighton, UK. Vol. 547. Berlin, Heidelberg: Springer, 1991. P. 17–38.

71. Daemen J., Rijmen V. Probability distributions of correlation and differentials in block ciphers. *Journal of Mathematical Cryptology*. 2007. 1(3). P. 221–242.

72. Canteaut A., Roué J. Differential Attacks Against SPN: A Thorough Analysis / International Conference on Codes, Cryptology, and Information Security, C2SI 2015, May 2015, Rabat, Morocco. Springer, Cham, 2015. P. 45–62.

73. Kanda M., Takashima Y., Matsumoto T., Aoki K., Ohta K. A strategy for constructing fast round functions with practical security against differential and linear cryptanalysis. *International Workshop on Selected Areas in Cryptography*. Berlin, Heidelberg : Springer, 1998. P. 264–279.

74. Nyberg K., Knudsen L. Provable security against differential cryptanalysis. *Journal of Cryptology*. 1995. 8.1. P. 27–37.

75. Daemen J., Rijmen V. The Wide Trail Design Strategy // Honary, B. (ed.) Cryptography and Coding, 2001. Vol. 2260 of Lecture Notes in Computer Science. Springer, Heidelberg, 2001. P. 222–238.

76. Katagi M., Moriai S. Lightweight Cryptography for the Internet of Things. Sony Corporation. 2008. P. 7–10.

77. Mouha N. The Design Space of Lightweight Cryptography. *NIST Lightweight Cryptography Workshop*. 2015. 19 p.

78. Kerry A. McKay, Larry Bassham, Meltem Sönmez Turan, Nicky Mouha. Report on Lightweight Cryptography. URL: https://nvlpubs.nist.gov/nistpubs/ir/2017/NIST.IR.8114.pdf.

79. The 128-bit block cipher CLEFIA / Taizo Shirai et al. *International workshop on fast software encryption*. Springer, Berlin, Heidelberg, 2007. P. 181–195.

80. Suzaki T., Minematsu K., Morioka S., et al. Twine: A lightweight, versatile block cipher. *ECRYPT Workshop on Lightweight Cryptography*. LC11. 2011. P. 146–169.

81. Gong Z., Nikova S., Law Y. W. KLEIN: a new family of lightweight block ciphers. *International Workshop on Radio Frequency Identification: Security and Privacy Issues*. Springer, Berlin, Heidelberg, 2011. P. 1–18.

82. Banik S., Bogdanov A., Isobe T., Shibutani K., Hiwatari H., Akishita T., Regazzoni F. Midori: A block cipher for low energy. *Advances in Cryptology* –

ASIACRYPT 2015 : Proceedings of 21st International Conference on the Theory and Application of Cryptology and Information Security, November 29 – December 3, 2015, Auckland, New Zealand. Part II, Vol. 9453 of Lecture Notes in Computer Science. Berlin, Heidelberg : Springer, 2015. P. 411–436.

83. Borghoff J., et al. PRINCE – A Low-Latency Block Cipher for Pervasive Computing Applications – Extended Abstract. *Advances in Cryptology –ASIACRYPT 2012* : Proceedings of 18th International Conference on the Theory and Application of Cryptology and Information Security, 2–6 Dec., 2012, Beijing, China. Vol. 7658 of Lecture Notes in Computer Science. Berlin, Heidelberg : Springer, 2012. P. 208–225.

84. Albrecht M. R., Driessen B., Kavun E. B., Leander G., Paar C., Yalcın T. Block ciphers – focus on the linear layer (feat. PRIDE). *Advances in Cryptology – CRYPTO 2014* : Proceedings of Annual Cryptology Conference. Part I., Vol. 8616 of Lecture Notes in Computer Science. Berlin, Heidelberg : Springer, 2014. P. 57–76.

85. Zhang W., Bao Z., Lin D., Rijmen V., Yang B., Verbauwhede I. RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. *Science China Information Sciences*. 2015. 58(12), P. 1–15.

86. Kitsos P., Galanis M. D., Koufopavlou O. High-speed hardware implementations of the KASUMI block cipher // Proceedings of 2004 IEEE International Symposium on Circuits and Systems, May, 2004. Vol. 2. P. 549–552.

87. Standaert F. X., Piret G., Gershenfeld N., Quisquater J. J. SEA: A scalable encryption algorithm for small embedded applications // Proceedings of International Conference on Smart Card Research and Advanced Applications, April, 2006. Berlin, Heidelberg: Springer. P. 222–236.

88. Mouha N., Mennink B., Van Herrewege A., Watanabe D., Preneel B., Verbauwhede I. Chaskey: an efficient MAC algorithm for 32-bit microcontrollers // Proceedings of International Conference on Selected Areas in Cryptography, August, 2014. Springer, Cham. P. 306–323.

89. Wheeler D. J., Needham R. M. TEA, a tiny encryption algorithm // Preneel, B. (ed.) Proceedings of FSE ’94, the Second Fast Software Encryption Workshop,

14–16 Dec., 1994, Belgium, December. Lecture Notes in Computer Science, vol. 1008. Berlin, Heidelberg: Springer, 1995. P. 363–366.

90. Needham R. M., Wheeler D. J. TEA extensions // Technical report, the Computer Laboratory, University of Cambridge, 1997. Archive available at: http://www.cl.cam.ac.uk/ftp/users/djw3/xtea.ps.

91. Bernstein D. J. The Salsa20 family of stream ciphers. *New stream cipher designs*. 2008. Berlin, Heidelberg: Springer. P. 84–97.

92. Courtois N. T., Pieprzyk J. Cryptanalysis of block ciphers with overdefined systems of equations // Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security, 1–5 Dec., 2002, Queenstown, New Zealand. Berlin, Heidelberg: Springer, 2002. P. 267–287.

93. Y. Crama and P. L. Hammer. Boolean Models and Methods in Mathematics, Computer Science and Engineering / Encyclopedia of Mathematics and its Applications. V. 2, Cambridge University Press, 2010. 759 p.

94. Nyberg K. Differentially uniform mapping for cryptography. *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, Berlin, Heidelberg, 1993. P. 55-64.

95. Nyberg K. Linear approximation of block ciphers. *Advances in Cryptology – EUROCRYPT'94* : Proceedings of Workshop on the Theory and Application of Cryptographic Techniques, 9–12 May, 1994, Perugia, Italy. Vol. 13. Springer Science & Business Media, 1995. P. 439–444.

96. Hong S., Lee S., Lim J., Sung J., Cheon D., Cho I. Provable security against differential and linear cryptanalysis for the SPN structure // Proceedings of International Workshop on Fast Software Encryption, 10–12 Apr., 2000, New York, NY, USA. Berlin, Heidelberg: Springer. P. 273–283.

97. Carlet C. Vectorial Boolean Functions for Cryptography / Boolean Models and Methods in Mathematics, Computer Science, and Engineering (ed. Y. Crama, P. Hammer) : Cambridge, Cambridge University Press, 2010. P. 398–470.

98. Казимиров А. В., Олейников Р. В. Использование векторных функций при генерации подстановок для симметричных криптографических преобразований. *Системи обробки інформації*. 2012. № 6 (104). С. 97–102.

99. Кузнецов А. А., Избенко Ю. А., Московченко И. В. Метод построения криптографически стойких булевых функций на основе градиентного спуска. *Зб. наук. пр. Харк. ун-ту Повітр. Сил*. Харків: ХУПС, 2007. Вип. 1. С. 63–66.

100. TESAŘ P. A New Method for Generating High Non-linearity S-Boxes. *Radioengineering*. 2010. Vol. 19, № 1. P. 23–26.

101. de la Cruz Jiménez R. A. Generation of 8-bit s-boxes having almost optimal cryptographic properties using smaller 4-bit s-boxes and finite field multiplication. *International Conference on Cryptology and Information Security in Latin America*, Sept. 2017. Springer, Cham, 2017. P. 191–206.

102. ДСТУ 7564:2014. Інформаційні технології. Криптографічний захист інформації. Функція ґешування [Чинний від 01–04–2015]. Вид. офіц. Київ : Мінекономрозвитку України, 2015. 39 с.

103. Rodinko M., Oliynykov R., Gorbenko Y. Optimization of the High Nonlinear S-Boxes Generation Method. *Tatra Mountains Mathematical Publications*. 2017. Vol. 70. Is. 1. P. 93–105.

104. Rodinko M., Oliynykov R., Gorbenko Y. Improvement of the high nonlinear S-boxes generation method. *Problems of Infocommunications Science and Technology (PIC S&T)* : Proceedings of 2016 Third International Scientific-Practical Conference, 4–6 October 2016, Kharkiv, 2016. P. 63–66.

105. Huang J., Lai X. Revisiting key schedule’s diffusion in relation with round function’s diffusion. *Designs, codes and cryptography*. 2014. 73.1. P. 85–103.

106. Kocher Paul C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems // Proceedings of Annual International Cryptology Conference, Aug. 1996. Berlin, Heidelberg: Springer, 1996. P. 104–113.

107. May L., Henricksen M., Millan W., Carter G., Dawson E. Strengthening the Key Schedule of the AES // Proceedings of Australasian Conference on

Information Security and Privacy, 3–5 Jul., 2002, Melbourne, Australia. Berlin, Heidelberg: Springer, 2002. P. 226–240.

108. Knudsen L. R., Mathiassen J. E. On the Role of Key Schedules in Attacks on Iterated Ciphers // Proceedings of European Symposium on Research in Computer Security , Sept., 2004. Berlin, Heidelberg: Springer, 2004. P. 322–334.

109. Knudsen L. R. Practically secure Feistel ciphers // Proceedings of International Workshop on Fast Software Encryption, Dec., 1993. Berlin, Heidelberg: Springer, 1993. P. 211– 221.

110. Колчин В. Ф. Случайные отображения : Наука, глав. ред. физико-математической литературы, 1984.

111. 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.

112. Родінко М. Ю., Олійников Р. В. Аналіз неін'єктивних схем розгортання ключів симетричних блокових шифрів. *Безпека інформації в інформаційно-телекомунікаційних системах* : матеріали Міжн. наук.-практ. конф., 25–26 травня 2016 р., Київ, 2016. Вип. 18. С. 50.

113. Родінко М. Ю. Оцінка ймовірності існування еквівалентних ключів для неін'єктивних схем розгортання ключів симетричних блокових шифрів. *Радіоелектроніка і молодь у XXI столітті* : матеріали XVIII Міжн. молодіжного форуму, 19–21 квітня 2016 р., Харків: ХНУРЕ, 2016. Т.5. С. 77–78.

114. Родінко М. Ю. Математична модель оцінки властивостей неін'єктивних схем розгортання ключів симетричних блокових шифрів. *Проблеми кібербезпеки інформаційно-телекомунікаційних систем* : матеріали доповідей наук.-практ. конф., 10–11 березня 2016 р., Київ, 2016. С. 71–72.

115. Горбенко Ю. І., Ганзя Р. С. Аналіз стійкості популярних криптоситем проти квантового криптоаналізу на основі алгоритму Гровера. *Захист інформації*. 2014. 16.2. С. 106–113.

116. Preneel B., Rijmen V. and Bosselaers A. Recent developments in the design of conventional cryptographic algorithms. *State of the Art in Applied Cryptography*. Berlin, Heidelberg: Springer, 1998. P. 105-130.

117. Schneier B. Description of a new variable-length key, 64-bit block cipher (Blowfish) // Proceedings of International Workshop on Fast Software Encryption, Dec., 1993. Berlin, Heidelberg: Springer, 1993. P. 191–204.

118. Mouha, Nicky, and Bart Preneel. Towards finding optimal differential characteristics for ARX: Application to Salsa20. Cryptology ePrint Archive. Report 2013/328, 2013.

119. Спосіб криптографічного перетворення двійкових даних: пат. 111448 Україна: № a201503976; заявл. 24.04.2015; опубл. 25.04.2016, Бюл. № 8. 6 с.

120. Спосіб криптографічного перетворення двійкових даних (варіанти): пат. 111547 Україна: № a201500942; заявл. 06.02.2015; опубл. 10.05.2016, Бюл. № 9. 13 с.

121. Aumasson J. P. et al. New features of Latin dances: analysis of Salsa ChaCha and Rumba. *Fast Software Encryption*: Proceedings of 15th International Workshop, FSE 2008, 10–13 Febr., 2008, Lausanne, Switzerland. Berlin, Heidelberg: Springer, 2008. P. 470–488.

122. Lipmaa H., Wallén J., Dumas P. On the additive differential probability of exclusive-or // Proceedings of International Workshop on Fast Software Encryption, Feb., 2004. Berlin, Heidelberg: Springer, 2004. P. 317–331.

123. Lipmaa H., Moriai S. Efficient algorithms for computing differential properties of addition // Proceedings of International Workshop on Fast Software Encryption, Apr., 2001. Berlin, Heidelberg: Springer, 2001, P. 336–350.

124. Andrushkevych A., Gorbenko Y., Kuznetsov O., Oliynykov R., Rodinko M. A Prospective Lightweight Block Cipher for Green IT Engineering. *Green IT Engineering: Social, Business and Industrial Applications. Studies in Systems, Decision and Control*. Springer, Cham. 2019. Vol. 171. P. 95–112.

125. Родінко М. Ю., Олійников Р. В. Методи пошуку диференційних характеристик циклової функції симетричного блокового шифру «Кипарис». *Радиотехника*. 2017. Вып. 191. С. 47–51.

126. Родінко М.Ю. Малоресурсний симетричний блоковий шифр «Кипарис» – сутність та основні властивості. *Математичне та комп'ютерне моделювання. Серія: Технічні науки*. 2017. Вип. 15. С. 203–208.

127. Родінко М.Ю. Оцінка стійкості симетричного блокового шифру «Кипарис» до диференційного криптоаналізу. *Радиотехника*. 2018. Вып. 195. С. 113–124.

128. Елисеев Р.Ю., Родинко М.Ю., Олейников Р.В. Дифференциальный криптоанализ блочного ARX-шифра «Кипарис-256». *Прикладная радиоэлектроника*. 2018. Вып. 17, № 3-4. С. 121–126.

129. Родінко М. Ю., Олійников Р. В., Горбенко І. Д. Перспективний блоковий шифр «Кипарис». *Комп'ютерне моделювання у наукоємних технологіях (КМНТ -2016)* : матеріали Міжн. наук.-техн. конф., 26–31 травня 2016 р., Харків, 2016. С. 292–295.

130. Родінко М. Ю., Олійников Р. В. Постквантовий малоресурсний алгоритм симетричного блокового перетворення «Кипарис». *Проблеми кібербезпеки інформаційно-телекомунікаційних систем*: матеріали доповідей наук.-практ. конф., 23–24 березня 2017 р., Київ, 2017. С. 172–176.

131. Родінко М. Ю., Олійников Р. В., Руженцев В. І., Єлисєєв Р. Ю. Підхід до оцінки диференційних властивостей перспективного симетричного блокового шифру «Кипарис». *Безпека інформації в інформаційно-телекомунікаційних системах*: матеріали XIX Міжн. наук.-практ. конф., 25–26 травня 2017 р., м. Буча, Київська обл., 2017. С. 105–106.

132. Rodinko M., Oliynykov R. Open problems of proving security of ARX-based ciphers to differential cryptanalysis. *Problems of Infocommunications Science and Technology (PIC S&T)* : Proceedings of 2017 4th International Scientific-Practical Conference, 10–13 October 2017, Kharkiv, 2017. P. 228–231.

133. Rodinko M., Oliynykov R., Eliseev R. Search for one-round differential characteristics of lighweight block cipher Cypress-256. *Dependable Systems, Services and Technologies (DESSERT)* : Proceedings of 2018 IEEE 9th International Conference, 24–27 May 2018, Kyiv, 2018. P. 312–315.

134. Rodinko M., Oliynykov R. An Approach to Search for Multi-Round Differential Characteristics of Cypress-256. *Problems of Infocommunications Science and Technology (PIC S&T)* : Proceedings of 2018 International Scientific-Practical Conference, 9–12 October 2018, Kharkiv, 2018. P. 659–662.

135. Rodinko M., Oliynykov R. The Method of Searching for Differential Trails of ARX-based Block Cipher Cypress. *Dependable Systems, Services and Technologies (DESSERT)* : Proceedings of 2020 IEEE 11th International Conference, 14–18 May 2020, Kyiv, 2020. P. 157–160.

136. Random Bit Generation. NIST SP 800-22: Download Documentation and Software. URL: https://csrc.nist.gov/Projects/Random-Bit-Generation/Documentation-and-Software (дата звернення: 15.08.2020).

137. Nadu S. T. A block cipher algorithm to enhance the avalanche effect using dynamic key-dependent S-box and genetic operations. *International Journal of Pure and Applied Mathematics*. 2018. Vol. 119 (10). P. 399–418.

138. Roman-Oliynykov/ciphers-speed. GitHub. URL: https://github.com/Roman-Oliynykov/ciphers-speed (дата звернення: 15.08.2020).

139. FELICS (Fair Evaluation of Lightweight Cryptographic Systems). URL: https://www.cryptolux.org/index.php/FELICS (дата звернення: 15.08.2020).

140. Beaulieu R., Shors D., Smith J., Treatman-Clark S., Weeks B., Wingers L. The SIMON and SPECK Families of Lightweight Block Ciphers. IACR Cryptology ePrint Archive: Report 2013/404. 2013. P. 1–42. URL: https://eprint.iacr.org/2013/404.pdf.

141. Родінко М. Ю., Олійников Р. В. Дослідження продуктивності малоресурсного блокового шифру «Кипарис» на різних платформах. *Радіотехніка*. 2020. Вип. 200. С. 51–57.

142. Rodinko M., Oliynykov R. Comparing Performances of Cypress Block Cipher and Modern Lighweight Block Ciphers on Different Platforms. *Problems of Infocommunications, Science and Technology (PIC S&T)* : Proceedings of 2019 IEEE International Scientific-Practical Conference, 8–11 October 2019, Kyiv, 2019. P. 113–116.

ДОДАТОК А. СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Розділ монографії, опублікований у співавторстві, що входить до міжнародної наукометричної бази:

1. Andrushkevych A., Gorbenko Y., Kuznetsov O., Oliynykov R., Rodinko M. A Prospective Lightweight Block Cipher for Green IT Engineering. *Green IT Engineering: Social, Business and Industrial Applications. Studies in Systems, Decision and Control*. Springer, Cham. 2019. Vol. 171. P. 95–112. (Scopus). (Особистий внесок здобувача: формування розділів щодо принципів побудови та властивостей перспективного малоресурсного блокового шифру).

Публікації у наукових фахових виданнях України:

2. Родінко М. Ю., Олійников Р. В. Методи пошуку диференційних характеристик циклової функції симетричного блокового шифру «Кипарис». *Радиотехника*. 2017. Вып. 191. С. 47–51.
   (Особистий внесок здобувача: розробка методів пошуку одноциклових диференційних характеристик ARX-шифру та результати застосування до блокового шифру «Кипарис-256»).
3. Родінко М. Ю. Малоресурсний симетричний блоковий шифр «Кипарис» – сутність та основні властивості. *Математичне та комп'ютерне моделювання. Серія: Технічні науки*. 2017. Вип. 15. С. 203–208.
4. Лисицкая И. В., Лисицкий К. Е., Родинко М. Ю., Головко И. А., Жариков И. И., Корниенко М. А., Кулеба М. В. Экспериментальные данные по определению динамических показателей прихода блочных симметричных шифров к состоянию случайной подстановки. *Радіоелектроніка, інформатика, управління*. 2017. № 1. С. 129–141. (Web of Science).

(Особистий внесок здобувача: отримання експериментальних даних щодо кількості активних S-блоків на різних циклах шифрування для низки блокових шифрів).

5. Родінко М. Ю. Оцінка стійкості симетричного блокового шифру «Кипарис» до диференційного криптоаналізу. *Радиотехника*. 2018. Вып. 195. С. 113–124.
6. Елисеев Р. Ю., Родинко М. Ю., Олейников Р. В. Дифференциальный криптоанализ блочного ARX-шифра «Кипарис-256». *Прикладная радиоэлектроника*. 2018. Вып. 17, № 3–4. С. 121–126.

   (Особистий внесок здобувача: постановка задачі, формування методики дослідження диференційних властивостей шифру).

Публікація у періодичному науковому виданні держави-члена ЄС, що входить до міжнародної наукометричної бази:

7. Rodinko M., Oliynykov R., Gorbenko Y. Optimization of the High Nonlinear S-Boxes Generation Method. *Tatra Mountains Mathematical Publications*. 2017. Vol. 70. Is. 1. P. 93–105. (Scopus, Словаччина).

   (Особистий внесок здобувача: розробка удосконаленого методу генерації підстановок та його застосування для генерації оптимальних S-блоків).

Наукові праці, які засвідчують апробацію матеріалів дисертації:

8. Rodinko M., Oliynykov R., Gorbenko Y. Improvement of the high nonlinear S-boxes generation method. *Problems of Infocommunications Science and Technology (PIC S&T)* : Proceedings of 2016 Third International Scientific-Practical Conference, 4–6 October 2016, Kharkiv, 2016. P. 63–66. (Scopus).

   (Особистий внесок здобувача: розробка удосконаленого методу генерації S-блоків з високою нелінійністю).
9. Родінко М. Ю., Олійников Р. В., Горбенко І. Д. Перспективний блоковий шифр «Кипарис». *Комп'ютерне моделювання у наукоємних технологіях*

(КМНТ -2016) : матеріали Міжн. наук.-техн. конф., 26–31 травня 2016 р., Харків, 2016. С. 292–295.

(Особистий внесок здобувача: розробка перспективного блокового шифру).

10. Родінко М. Ю., Олійников Р. В. Аналіз неін'єктивних схем розгортання ключів симетричних блокових шифрів. *Безпека інформації в інформаційно-телекомунікаційних системах* : матеріали Міжн. наук.-практ. конф., 25–26 травня 2016 р., Київ, 2016. Вип. 18. С. 50.

    (Особистий внесок здобувача: аналіз колізійних властивостей неін'єктивних схем розгортання ключів симетричних блокових шифрів).
11. Родінко М. Ю. Оцінка ймовірності існування еквівалентних ключів для неін'єктивних схем розгортання ключів симетричних блокових шифрів. *Радіоелектроніка і молодь у XXI столітті* : матеріали XVIII Міжн. молодіжного форуму, 19–21 квітня 2016 р., Харків : ХНУРЕ, 2016. Т.5. С. 77–78.
12. Родінко М. Ю. Математична модель оцінки властивостей неін'єктивних схем розгортання ключів симетричних блокових шифрів. *Проблеми кібербезпеки інформаційно-телекомунікаційних систем* : матеріали доповідей наук.-практ. конф., 10–11 березня 2016 р., Київ, 2016. С. 71–72.
13. Родінко М. Ю., Олійников Р. В. Постквантовий малоресурсний алгоритм симетричного блокового перетворення «Кипарис». *Проблеми кібербезпеки інформаційно-телекомунікаційних систем* : матеріали доповідей наук.-практ. конф., 23–24 березня 2017 р., Київ, 2017. С. 172–176.

    (Особистий внесок здобувача: розробка принципів побудови алгоритму симетричного блокового перетворення «Кипарис» та дослідження його властивостей).
14. Родінко М. Ю., Олійников Р. В., Руженцев В. І., Єлисєєв Р. Ю. Підхід до оцінки диференційних властивостей перспективного симетричного блокового шифру «Кипарис». *Безпека інформації в інформаційно-телекомунікаційних системах* : матеріали XIX Міжн. наук.-практ. конф., 25–26 травня 2017 р., м. Буча, Київська обл., 2017. С. 105–106.

(Особистий внесок здобувача: розробка методу пошуку диференційних характеристик циклової функції блокового шифру «Кипарис»).

15. Rodinko M., Oliynykov R. Open problems of proving security of ARX-based ciphers to differential cryptanalysis. *Problems of Infocommunications Science and Technology (PIC S&T)* : Proceedings of 2017 4th International Scientific-Practical Conference, 10–13 October 2017, Kharkiv, 2017. P. 228–231. (Scopus).
(Особистий внесок здобувача: аналіз існуючих методів оцінки стійкості ARX-шифрів до диференційного криптоаналізу).
16. Rodinko M., Oliynykov R., Eliseev R. Search for one-round differential characteristics of lighweight block cipher Cypress-256. *Dependable Systems, Services and Technologies (DESSERT)* : Proceedings of 2018 IEEE 9th International Conference, 24–27 May 2018, Kyiv, 2018. P. 312–315. (Scopus).
(Особистий внесок здобувача: розробка методів пошуку одноциклових диференційних характеристик ARX-шифру «Кипарис»).
17. Rodinko M., Oliynykov R. An Approach to Search for Multi-Round Differential Characteristics of Cypress-256. *Problems of Infocommunications Science and Technology (PIC S&T)* : Proceedings of 2018 International Scientific-Practical Conference, 9–12 October 2018, Kharkiv, 2018. P. 659–662. (Scopus).
(Особистий внесок здобувача: розробка принципів методики пошуку багатоциклових диференційних характеристик ARX-шифру «Кипарис»).
18. Rodinko M., Oliynykov R. Comparing Performances of Cypress Block Cipher and Modern Lighweight Block Ciphers on Different Platforms. *Problems of Infocommunications, Science and Technology (PIC S&T)* : Proceedings of 2019 IEEE International Scientific-Practical Conference, 8–11 October 2019, Kyiv, 2019. P. 113–116. (Scopus).
(Особистий внесок здобувача: отримання експериментальних даних щодо швидкодії відомих малоресурсних блокових шифрів та шифру «Кипарис»).
19. Rodinko M., Oliynykov R. The Method of Searching for Differential Trails of ARX-based Block Cipher Cypress. *Dependable Systems, Services and*

Technologies (DESSERT) : Proceedings of 2020 IEEE 11th International Conference, 14–18 May 2020, Kyiv, 2020. P. 157–160. (Scopus).

(Особистий внесок здобувача: розробка методу пошуку багатоциклових диференційних характеристик ARX-шифру).

Наукові публікації, що додатково відображають зміст дисертації:

20. 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.

    (Особистий внесок здобувача: розробка методу оцінки колізійних властивостей неін'єктивних схем розгортання ключів).

21. Родінко М. Ю., Олійников Р. В. Дослідження продуктивності малоресурсного блокового шифру «Кипарис» на різних платформах. *Радіотехніка*. 2020. Вип. 200. С. 51–57. (Index Copernicus).

    (Особистий внесок здобувача: отримання експериментальних даних щодо швидкодії блокового шифру «Кипарис» та низки малоресурсних блокових шифрів).

Патенти:

22. Спосіб криптографічного перетворення двійкових даних: пат. 111448 Україна: № a201503976; заявл. 24.04.2015; опубл. 25.04.2016, Бюл. № 8. 6 с.

    (Особистий внесок здобувача: отримання експериментальних даних щодо кількості активних S-блоків на різних циклах запропонованого перетворення).

23. Спосіб криптографічного перетворення двійкових даних (варіанти): пат. 111547 Україна: № a201500942; заявл. 06.02.2015; опубл. 10.05.2016, Бюл. № 9. 13 с.

    (Особистий внесок здобувача: здійснення розрахунків щодо кількості активних S-блоків на різних циклах запропонованого перетворення).

ДОДАТОК Б. РЕЗУЛЬТАТИ СТАТИСТИЧНОГО ТЕСТУВАННЯ БЛОКОВОГО ШИФРУ «КИПАРИС»

Результати тестування вихідної послідовності шифруючого перетворення шифру «Кипарис-256»

```
------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
   generator is <Cypress256.dat>
------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
  7  11   7  16   8  12  13   9   4  13  0.224821     98/100     Frequency
 20   8  10   7   7   9  13   4  16   6  0.008879     98/100     BlockFrequency
 12   8   9  11  12  14   9   8   5  12  0.699313     99/100     CumulativeSums
 10  11   6  12  11  10  11   8  11  10  0.971699     98/100     CumulativeSums
  3  16   5  14   9  11  14   7  11  10  0.080519    100/100     Runs
  9  10  12   7   9   9  11  10   9  14  0.946308     99/100     LongestRun
 14   5   6   9  11  13  12   8   9  13  0.474986     98/100     Rank
 13  10   9  11   8   9  15   9   9   7  0.816537     99/100     FFT
 10   6   9   7  14   6   8  11  10  19  0.108791    100/100     NonOverlappingTemplate
  9  11  12   5   9   9   9  12  10  14  0.798139     98/100     NonOverlappingTemplate
  6   8   7  14  15   9   9  11  11  10  0.595549    100/100     NonOverlappingTemplate
  8  10   7   8  14  10  13  12   9   9  0.851383    100/100     NonOverlappingTemplate
 12   7  10  12  14  10  10  12   5   8  0.678686    100/100     NonOverlappingTemplate
 11  11   9   9  10   9  11   7  10  13  0.983453     97/100     NonOverlappingTemplate
 10   8  11  12  15   8  11   6  12   7  0.657933    100/100     NonOverlappingTemplate
  9  11  13   9   8   7  11   5  18   9  0.236810    100/100     NonOverlappingTemplate
  9   5   8  16  18   7  13   9   7   8  0.062821    100/100     NonOverlappingTemplate
  6   8  16   8  11   8  18   6  13   6  0.048716     98/100     NonOverlappingTemplate
  8  10  10   9  15  12   9   8   5  14  0.534146    100/100     NonOverlappingTemplate
 14  13   9  13   5  11   8  11   7   9  0.574903    100/100     NonOverlappingTemplate
  9   7   9  10  12  14  12  10   8   9  0.911413    100/100     NonOverlappingTemplate
 10  11   6  19   9   6  11  10   6  12  0.137282     99/100     NonOverlappingTemplate
 15   6  14  11   5   9  11  15   7   7  0.171867     99/100     NonOverlappingTemplate
  9   8   8   7  12  10  12  11  14   9  0.883171     99/100     NonOverlappingTemplate
 10   9  11  10   8  15   9  11  12   5  0.719747     99/100     NonOverlappingTemplate
  9   7   6   7  17  13  13  11   8   9  0.289667     99/100     NonOverlappingTemplate
 14   9   7  10  13  11  10   5  11  10  0.719747     99/100     NonOverlappingTemplate
 12  10  15   9   6   9  11  16   4   8  0.191687     99/100     NonOverlappingTemplate
 13  16  10  10  12   8   9   4   7  11  0.350485     99/100     NonOverlappingTemplate
  8   8  14   7   9  13   7  10   6  18  0.153763     99/100     NonOverlappingTemplate
  6  10  10  12  11  10  12  10   7  12  0.924076     99/100     NonOverlappingTemplate
  4  11  12  13   9   8   5  13  14  11  0.304126    100/100     NonOverlappingTemplate
  8  11  12  10   9  15  12  10   7   6  0.699313     98/100     NonOverlappingTemplate
  7  10  13   9  12   9   9   9  10  12  0.964295    100/100     NonOverlappingTemplate
 10   7   9   9  15   8  11   9   8  14  0.719747    100/100     NonOverlappingTemplate
 13  12  10  14   9   6   8  11   7  10  0.739918     99/100     NonOverlappingTemplate
 12  14  11  11  10   8   6  11   6  11  0.739918    100/100     NonOverlappingTemplate
  9  10  12   9   9  10   8  15   9   9  0.924076    100/100     NonOverlappingTemplate
 13  10  10   8   7   9   9  13   7  14  0.759756     98/100     NonOverlappingTemplate
 13  13   8   8  10  10   7  14  10   7  0.739918     97/100     NonOverlappingTemplate
 11   9   6  12  11   4  11  12  11  13  0.595549     99/100     NonOverlappingTemplate
 14  10  10   8   7  11   4   9  12  15  0.383827     99/100     NonOverlappingTemplate
  8  10  13  16   6   6   6   9  13  13  0.236810    100/100     NonOverlappingTemplate
  9  11  12   7   7  19   9  14   4   8  0.062821     99/100     NonOverlappingTemplate
 14   8  12  12  10   3   7  12   9  13  0.350485     99/100     NonOverlappingTemplate
```

5	7	10	15	7	8	12	13	13	10	0.401199	100/100		NonOverlappingTemplate
14	9	12	10	10	11	6	7	5	16	0.289667	99/100		NonOverlappingTemplate
11	11	11	9	11	12	8	8	10	9	0.994250	99/100		NonOverlappingTemplate
17	8	13	7	8	8	6	9	9	15	0.202268	100/100		NonOverlappingTemplate
11	10	15	8	13	12	8	5	6	12	0.419021	98/100		NonOverlappingTemplate
8	11	12	11	7	9	14	10	6	12	0.779188	97/100		NonOverlappingTemplate
11	6	14	5	11	9	10	12	13	9	0.595549	100/100		NonOverlappingTemplate
9	10	4	13	11	8	10	20	11	4	0.026948	99/100		NonOverlappingTemplate
14	11	9	9	7	11	10	5	13	11	0.699313	97/100		NonOverlappingTemplate
9	12	8	11	11	6	15	11	10	7	0.719747	96/100		NonOverlappingTemplate
9	7	16	10	16	6	8	10	9	9	0.319084	100/100		NonOverlappingTemplate
14	7	9	10	12	18	10	9	5	6	0.137282	100/100		NonOverlappingTemplate
10	10	6	12	5	11	14	11	13	8	0.574903	99/100		NonOverlappingTemplate
7	12	12	8	8	9	12	10	11	11	0.955835	99/100		NonOverlappingTemplate
9	8	13	13	4	9	12	7	11	14	0.437274	99/100		NonOverlappingTemplate
9	8	12	10	12	10	7	11	9	12	0.971699	100/100		NonOverlappingTemplate
14	10	8	7	11	11	11	8	14	6	0.657933	95/100	*	NonOverlappingTemplate
4	16	11	8	5	12	16	12	8	8	0.080519	100/100		NonOverlappingTemplate
9	9	14	8	8	13	10	10	10	9	0.935716	98/100		NonOverlappingTemplate
7	16	9	9	9	11	10	8	12	9	0.759756	99/100		NonOverlappingTemplate
15	9	10	10	6	10	13	9	7	11	0.719747	97/100		NonOverlappingTemplate
9	8	12	10	11	7	14	11	7	11	0.867692	99/100		NonOverlappingTemplate
6	16	8	13	9	11	12	7	8	10	0.494392	99/100		NonOverlappingTemplate
9	6	7	5	10	15	13	12	11	12	0.401199	100/100		NonOverlappingTemplate
12	11	6	11	7	6	10	11	13	13	0.678686	97/100		NonOverlappingTemplate
8	7	8	10	12	17	10	8	6	14	0.304126	100/100		NonOverlappingTemplate
9	10	6	14	9	12	11	9	11	9	0.897763	99/100		NonOverlappingTemplate
9	8	11	7	9	9	13	12	11	11	0.955835	98/100		NonOverlappingTemplate
5	8	9	11	10	11	10	9	13	14	0.759756	99/100		NonOverlappingTemplate
10	10	7	8	9	13	10	14	8	11	0.883171	99/100		NonOverlappingTemplate
10	9	17	7	8	9	13	11	4	12	0.249284	98/100		NonOverlappingTemplate
6	12	16	9	8	6	8	11	18	6	0.062821	98/100		NonOverlappingTemplate
10	8	5	9	14	8	12	16	5	13	0.191687	97/100		NonOverlappingTemplate
8	8	8	14	16	7	10	16	8	5	0.129620	100/100		NonOverlappingTemplate
10	8	9	9	11	9	6	17	10	11	0.595549	99/100		NonOverlappingTemplate
15	11	14	7	13	5	8	4	14	9	0.115387	99/100		NonOverlappingTemplate
14	11	11	7	15	3	5	11	10	13	0.137282	100/100		NonOverlappingTemplate
10	6	9	8	13	6	8	11	10	19	0.153763	100/100		NonOverlappingTemplate
10	10	10	5	17	6	9	11	14	8	0.262249	100/100		NonOverlappingTemplate
12	9	15	7	9	7	16	9	9	7	0.383827	98/100		NonOverlappingTemplate
8	8	16	7	7	5	10	14	14	11	0.213309	99/100		NonOverlappingTemplate
12	11	12	8	13	10	9	7	8	10	0.935716	99/100		NonOverlappingTemplate
9	16	14	7	11	11	9	9	5	9	0.419021	98/100		NonOverlappingTemplate
10	6	10	7	13	14	9	9	10	12	0.779188	98/100		NonOverlappingTemplate
9	12	13	11	11	11	5	9	9	10	0.883171	100/100		NonOverlappingTemplate
14	8	11	5	7	15	9	11	7	13	0.350485	98/100		NonOverlappingTemplate
18	5	10	10	5	11	10	9	8	14	0.137282	99/100		NonOverlappingTemplate
8	12	10	10	7	10	13	8	13	9	0.911413	100/100		NonOverlappingTemplate
7	11	10	12	14	7	10	9	11	9	0.897763	100/100		NonOverlappingTemplate
6	7	11	13	15	11	11	9	8	9	0.657933	98/100		NonOverlappingTemplate
16	6	5	11	8	6	13	13	11	11	0.224821	100/100		NonOverlappingTemplate
16	10	12	11	8	10	9	6	11	7	0.616305	100/100		NonOverlappingTemplate
8	8	8	12	8	13	13	11	5	14	0.534146	99/100		NonOverlappingTemplate
8	7	9	21	2	12	10	13	10	8	0.010237	100/100		NonOverlappingTemplate
10	14	10	7	12	9	12	5	10	11	0.739918	99/100		NonOverlappingTemplate
17	6	8	8	8	9	11	12	10	11	0.494392	96/100		NonOverlappingTemplate
10	12	11	11	7	15	8	8	12	6	0.657933	100/100		NonOverlappingTemplate
12	8	9	7	9	9	8	8	11	19	0.275709	100/100		NonOverlappingTemplate
12	5	10	17	5	7	8	11	13	12	0.162606	99/100		NonOverlappingTemplate
9	10	5	17	5	7	19	9	9	10	0.023545	98/100		NonOverlappingTemplate
12	13	8	8	13	6	8	12	8	12	0.719747	100/100		NonOverlappingTemplate
14	12	11	12	7	9	10	7	11	7	0.798139	96/100		NonOverlappingTemplate
9	7	10	10	10	7	6	15	13	13	0.554420	97/100		NonOverlappingTemplate
16	8	9	14	7	9	8	19	3	7	0.012650	99/100		NonOverlappingTemplate
5	13	17	6	8	12	10	10	15	4	0.051942	100/100		NonOverlappingTemplate
12	10	13	10	6	8	10	13	10	8	0.867692	99/100		NonOverlappingTemplate

11	9	5	15	9	9	13	12	11	6	0.494392	99/100	NonOverlappingTemplate
7	4	11	14	13	14	10	12	5	10	0.236810	100/100	NonOverlappingTemplate
6	8	10	11	11	7	9	8	16	14	0.455937	100/100	NonOverlappingTemplate
11	6	14	11	8	11	12	6	6	15	0.350485	99/100	NonOverlappingTemplate
6	10	8	9	19	8	12	6	8	14	0.102526	99/100	NonOverlappingTemplate
9	12	6	11	10	19	8	8	10	7	0.213309	100/100	NonOverlappingTemplate
2	7	13	9	8	12	8	15	18	8	0.026948	100/100	NonOverlappingTemplate
10	9	7	8	9	10	8	15	8	16	0.494392	100/100	NonOverlappingTemplate
6	8	11	8	11	10	9	10	14	13	0.816537	100/100	NonOverlappingTemplate
7	15	12	7	15	11	9	10	8	6	0.401199	100/100	NonOverlappingTemplate
6	11	11	7	10	10	8	9	12	16	0.616305	100/100	NonOverlappingTemplate
8	8	11	8	14	10	10	10	11	10	0.964295	100/100	NonOverlappingTemplate
15	7	9	8	11	12	8	6	12	12	0.616305	97/100	NonOverlappingTemplate
15	11	8	7	10	9	11	11	12	6	0.719747	99/100	NonOverlappingTemplate
7	7	6	21	11	15	6	14	5	8	0.003996	100/100	NonOverlappingTemplate
7	11	9	6	14	14	9	5	11	14	0.334538	99/100	NonOverlappingTemplate
7	14	6	13	15	13	9	10	7	6	0.275709	99/100	NonOverlappingTemplate
12	10	8	7	8	9	11	10	13	12	0.935716	100/100	NonOverlappingTemplate
10	11	8	10	8	7	9	14	8	15	0.699313	99/100	NonOverlappingTemplate
9	12	7	6	9	13	15	7	11	11	0.574903	99/100	NonOverlappingTemplate
13	8	6	14	10	12	11	6	9	11	0.657933	98/100	NonOverlappingTemplate
11	11	12	12	12	7	8	7	12	8	0.883171	98/100	NonOverlappingTemplate
10	4	10	7	14	11	7	7	16	14	0.153763	99/100	NonOverlappingTemplate
12	11	8	5	10	8	6	10	13	17	0.262249	99/100	NonOverlappingTemplate
9	7	10	10	8	9	12	7	16	12	0.657933	99/100	NonOverlappingTemplate
7	17	8	5	9	10	9	10	14	11	0.304126	100/100	NonOverlappingTemplate
14	8	7	8	9	14	18	8	7	7	0.137282	99/100	NonOverlappingTemplate
12	3	7	6	14	7	13	14	14	10	0.108791	99/100	NonOverlappingTemplate
7	15	12	16	4	7	8	12	9	10	0.171867	99/100	NonOverlappingTemplate
10	11	10	8	10	7	12	14	8	10	0.924076	100/100	NonOverlappingTemplate
8	10	14	7	11	12	8	9	13	8	0.816537	99/100	NonOverlappingTemplate
17	13	11	9	3	4	12	9	9	13	0.066882	99/100	NonOverlappingTemplate
9	10	10	14	11	6	10	6	9	15	0.574903	99/100	NonOverlappingTemplate
7	7	14	5	12	13	7	11	12	12	0.437274	100/100	NonOverlappingTemplate
8	10	8	6	8	18	4	14	8	16	0.030806	99/100	NonOverlappingTemplate
7	12	7	11	8	13	9	12	7	14	0.678686	100/100	NonOverlappingTemplate
9	7	8	7	8	16	10	6	13	16	0.191687	100/100	NonOverlappingTemplate
10	13	4	13	12	11	11	5	9	12	0.437274	99/100	NonOverlappingTemplate
9	13	14	8	11	11	9	5	9	11	0.739918	100/100	NonOverlappingTemplate
8	9	6	8	12	9	13	11	10	14	0.779188	100/100	NonOverlappingTemplate
8	10	12	8	11	11	14	7	8	11	0.883171	100/100	NonOverlappingTemplate
9	10	6	9	12	8	12	14	12	8	0.798139	99/100	NonOverlappingTemplate
9	12	13	9	8	5	6	7	15	16	0.162606	100/100	NonOverlappingTemplate
12	10	4	8	10	14	11	5	18	8	0.080519	100/100	NonOverlappingTemplate
14	11	11	7	15	3	5	11	10	13	0.137282	100/100	NonOverlappingTemplate
9	16	9	6	7	12	9	17	8	7	0.162606	99/100	OverlappingTemplate
17	10	11	3	10	8	7	12	13	9	0.181557	100/100	Universal
11	14	9	6	15	10	8	5	12	10	0.419021	99/100	ApproximateEntropy
4	2	10	6	10	3	5	11	5	7	0.095617	63/63	RandomExcursions
0	7	5	10	6	5	6	5	14	5	0.016911	63/63	RandomExcursions
9	6	10	5	1	3	7	10	11	1	0.011931	61/63	RandomExcursions
7	7	7	6	1	4	5	7	12	7	0.222869	63/63	RandomExcursions
7	13	7	5	8	8	3	4	3	5	0.128379	60/63	RandomExcursions
9	5	5	7	6	6	3	10	5	7	0.689019	62/63	RandomExcursions
9	4	4	7	3	6	12	12	2	4	0.018969	62/63	RandomExcursions
9	5	5	6	8	4	5	4	9	8	0.723129	61/63	RandomExcursions
8	3	6	4	4	5	7	4	13	9	0.116519	61/63	RandomExcursionsVariant
7	7	3	4	7	11	5	5	6	8	0.551026	62/63	RandomExcursionsVariant
8	6	2	8	5	9	9	8	1	7	0.204076	63/63	RandomExcursionsVariant
7	5	9	5	6	9	5	6	5	6	0.922036	63/63	RandomExcursionsVariant
6	5	9	6	5	7	11	4	7	3	0.484646	63/63	RandomExcursionsVariant
5	10	3	6	8	7	9	9	3	3	0.264458	62/63	RandomExcursionsVariant
6	3	7	9	7	6	5	4	6	10	0.654467	62/63	RandomExcursionsVariant
8	2	8	5	7	8	9	8	4	4	0.484646	63/63	RandomExcursionsVariant
7	9	4	5	6	11	7	5	5	4	0.551026	62/63	RandomExcursionsVariant
8	5	6	5	7	4	8	2	7	11	0.392456	62/63	RandomExcursionsVariant

```
  9   5   5   6   6   7   4   6   8   7  0.941144     60/63      RandomExcursionsVariant
  9   9   5   6   6   5   7   8   3   5  0.756476     60/63      RandomExcursionsVariant
  9   7  12   8   4   3   3   6   9   2  0.063482     60/63      RandomExcursionsVariant
  7   9  10   5   9   1   8   3   6   5  0.186566     60/63      RandomExcursionsVariant
  6   9   6   5   7  13   7   4   4   2  0.116519     61/63      RandomExcursionsVariant
  5   9   4   9   6   7   5   8   5   5  0.819544     60/63      RandomExcursionsVariant
  8   6   3   8   6   6   4   8   8   6  0.848588     61/63      RandomExcursionsVariant
  6   6   7   9   4   4   7   8   5   7  0.900104     61/63      RandomExcursionsVariant
  6  11  14   7  11  10   7  13  12   9  0.678686     99/100     Serial
  7   6  12   8  11  10  11  12  13  10  0.851383     98/100     Serial
  7   8  10  13   3  17  10  12   7  13  0.115387    100/100     LinearComplexity


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 96 for a
sample size = 100 binary sequences.

The minimum pass rate for the random excursion (variant) test
is approximately = 60 for a sample size = 63 binary sequences.

For further guidelines construct a probability table using the MAPLE program
provided in the addendum section of the documentation.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Результати тестування вихідної послідовності шифруючого перетворення шифру «Кипарис-512»

```
------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
   generator is <Cypress512.dat>
------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
 11   9   9   9  14  10   9   8  13   8  0.924076     99/100     Frequency
 14   8   8  11  12   9   9   9  10  10  0.955835     98/100     BlockFrequency
 10   9   8  10   8  10  11  14  11   9  0.971699     98/100     CumulativeSums
 11  10   9  13  10   6  14   5   9  13  0.554420    100/100     CumulativeSums
  9  11  11  10   5  12   8  11  10  13  0.867692    100/100     Runs
  8   9  11  12  15  15   6  11   8   5  0.304126    100/100     LongestRun
 14   8  10   9   8  12  11   7   8  13  0.816537     97/100     Rank
 12   8   9  11  12  15   8   9   8   8  0.816537     98/100     FFT
 14   8   7   5   8   7  14  15  13   9  0.224821     97/100     NonOverlappingTemplate
 11  11  13   9  11   8  13   8   5  11  0.779188    100/100     NonOverlappingTemplate
 10   8  12   8  10   8  11   9  14  10  0.946308     98/100     NonOverlappingTemplate
 11  16   9   8   7  10   7  15   7  10  0.401199     98/100     NonOverlappingTemplate
 11   9   9  10   9  12  11   7  14   8  0.924076     97/100     NonOverlappingTemplate
 10   6  11  11  11  11  14  13   8   5  0.595549    100/100     NonOverlappingTemplate
 15  11   7   9   6  14   7  13   9   9  0.455937     97/100     NonOverlappingTemplate
 10  14   9  14  12   5  15   5  10   6  0.171867     99/100     NonOverlappingTemplate
 14  10   5  13   2  17  11   9  10   9  0.055361    100/100     NonOverlappingTemplate
  7   7   6  16  11   9   8   9  12  15  0.304126    100/100     NonOverlappingTemplate
  6  10  13  10  15  10   9  10  10   7  0.739918     99/100     NonOverlappingTemplate
  6  13   7   5   7  14   9  16  14   9  0.129620    100/100     NonOverlappingTemplate
 11  10  12   6  13   8  13   9   7  11  0.798139    100/100     NonOverlappingTemplate
 17   7   8   9  11   9   8   6  18   7  0.071177     96/100     NonOverlappingTemplate
  7  16   8   8  12  11   9  10   7  12  0.616305     98/100     NonOverlappingTemplate
 12  11   7   8   9  10  16   8  12   7  0.616305     99/100     NonOverlappingTemplate
  6  11  11  14  12  12  10  10   7   7  0.739918    100/100     NonOverlappingTemplate
  8  15   8  10  10  13   6  11  12   7  0.616305     99/100     NonOverlappingTemplate
  7   7  10   7  12  23   5  10   6  13  0.002971    100/100     NonOverlappingTemplate
  9   9   6  13   7  10  12   7  13  14  0.595549     99/100     NonOverlappingTemplate
```

13	9	9	12	13	12	6	6	14	6	0.419021	99/100	NonOverlappingTemplate
4	11	13	14	11	7	11	8	11	10	0.554420	100/100	NonOverlappingTemplate
13	12	13	4	11	15	13	4	5	10	0.080519	100/100	NonOverlappingTemplate
9	13	11	11	3	12	9	13	9	10	0.574903	99/100	NonOverlappingTemplate
12	2	11	10	7	11	13	5	13	16	0.071177	97/100	NonOverlappingTemplate
8	9	8	11	15	8	9	7	10	15	0.595549	100/100	NonOverlappingTemplate
18	8	7	5	13	10	6	9	11	13	0.129620	96/100	NonOverlappingTemplate
13	13	10	10	7	9	7	9	14	8	0.759756	99/100	NonOverlappingTemplate
8	8	10	15	10	12	11	7	8	11	0.816537	99/100	NonOverlappingTemplate
9	10	6	10	10	14	12	6	9	14	0.637119	99/100	NonOverlappingTemplate
5	17	11	12	9	8	8	12	8	10	0.383827	100/100	NonOverlappingTemplate
11	13	9	13	6	6	15	10	8	9	0.514124	100/100	NonOverlappingTemplate
14	11	10	14	9	11	9	9	8	5	0.678686	99/100	NonOverlappingTemplate
6	11	9	9	8	11	11	10	14	11	0.897763	100/100	NonOverlappingTemplate
7	14	6	12	11	9	4	15	10	12	0.262249	100/100	NonOverlappingTemplate
10	11	6	12	9	10	14	8	13	7	0.739918	98/100	NonOverlappingTemplate
5	13	11	7	12	8	14	10	9	11	0.637119	100/100	NonOverlappingTemplate
9	14	5	12	5	14	8	11	9	13	0.334538	99/100	NonOverlappingTemplate
8	16	13	14	8	8	7	5	7	14	0.153763	99/100	NonOverlappingTemplate
9	8	6	13	10	12	14	7	11	10	0.739918	98/100	NonOverlappingTemplate
7	11	7	9	14	8	14	6	13	11	0.514124	99/100	NonOverlappingTemplate
13	8	10	14	8	10	9	7	11	10	0.883171	98/100	NonOverlappingTemplate
6	9	9	7	10	10	14	10	13	12	0.779188	99/100	NonOverlappingTemplate
9	9	14	4	13	12	11	9	8	11	0.595549	100/100	NonOverlappingTemplate
5	9	14	12	12	10	7	10	12	9	0.699313	99/100	NonOverlappingTemplate
7	11	8	6	7	13	14	15	12	7	0.334538	100/100	NonOverlappingTemplate
7	13	9	7	8	15	6	13	8	14	0.334538	100/100	NonOverlappingTemplate
11	7	9	11	14	5	13	13	9	8	0.574903	97/100	NonOverlappingTemplate
17	12	4	12	11	8	7	8	9	12	0.236810	98/100	NonOverlappingTemplate
11	13	7	13	11	4	14	12	9	6	0.334538	98/100	NonOverlappingTemplate
10	9	9	13	11	8	14	8	7	11	0.867692	99/100	NonOverlappingTemplate
7	5	6	7	18	13	13	14	8	9	0.062821	99/100	NonOverlappingTemplate
11	8	6	11	6	6	17	13	13	9	0.202268	99/100	NonOverlappingTemplate
12	8	3	9	10	11	5	16	16	10	0.075719	100/100	NonOverlappingTemplate
5	11	15	6	11	12	7	10	11	12	0.474986	98/100	NonOverlappingTemplate
12	11	7	15	11	6	8	9	9	12	0.678686	98/100	NonOverlappingTemplate
8	8	10	13	7	12	13	14	8	7	0.657933	99/100	NonOverlappingTemplate
11	14	8	12	10	6	7	8	11	13	0.699313	99/100	NonOverlappingTemplate
7	12	9	11	13	10	13	10	9	6	0.834308	99/100	NonOverlappingTemplate
15	7	10	9	13	11	5	11	5	14	0.262249	100/100	NonOverlappingTemplate
12	5	11	10	9	13	5	12	9	14	0.474986	98/100	NonOverlappingTemplate
9	11	11	6	16	11	11	8	7	10	0.637119	98/100	NonOverlappingTemplate
7	7	12	10	14	9	10	11	9	11	0.897763	99/100	NonOverlappingTemplate
11	8	9	7	10	8	11	6	20	10	0.137282	100/100	NonOverlappingTemplate
11	14	8	10	7	9	6	13	10	12	0.739918	99/100	NonOverlappingTemplate
13	6	9	6	7	8	11	15	17	8	0.145326	98/100	NonOverlappingTemplate
15	10	9	7	6	11	15	6	10	11	0.401199	97/100	NonOverlappingTemplate
11	12	11	8	12	12	7	9	9	9	0.964295	97/100	NonOverlappingTemplate
8	10	8	9	10	16	13	9	5	12	0.494392	100/100	NonOverlappingTemplate
6	12	8	8	11	11	9	9	12	14	0.816537	99/100	NonOverlappingTemplate
9	1	13	12	13	14	11	5	15	7	0.035174	97/100	NonOverlappingTemplate
11	10	10	12	11	7	5	9	11	14	0.759756	99/100	NonOverlappingTemplate
11	12	11	14	11	9	8	4	13	7	0.514124	98/100	NonOverlappingTemplate
12	8	12	10	11	12	7	7	7	14	0.739918	98/100	NonOverlappingTemplate
13	9	7	5	8	7	14	15	13	9	0.289667	97/100	NonOverlappingTemplate
15	9	4	12	7	10	16	10	7	10	0.213309	97/100	NonOverlappingTemplate
10	14	11	6	7	3	13	13	14	9	0.181557	99/100	NonOverlappingTemplate
8	10	15	9	9	10	11	7	11	10	0.897763	100/100	NonOverlappingTemplate
14	12	10	10	6	12	10	8	9	9	0.867692	99/100	NonOverlappingTemplate
10	7	12	12	14	10	7	11	11	6	0.739918	100/100	NonOverlappingTemplate
12	7	9	10	15	13	8	8	11	7	0.678686	97/100	NonOverlappingTemplate
11	8	15	7	12	9	10	11	10	7	0.798139	100/100	NonOverlappingTemplate
6	12	12	10	11	8	5	11	16	9	0.419021	98/100	NonOverlappingTemplate
9	9	13	11	6	11	17	6	11	7	0.319084	98/100	NonOverlappingTemplate
11	7	5	13	5	9	14	10	12	14	0.304126	100/100	NonOverlappingTemplate
11	7	9	13	12	10	8	9	13	8	0.897763	100/100	NonOverlappingTemplate

14	13	6	11	6	7	7	11	8	17	0.162606	98/100	NonOverlappingTemplate
13	8	14	7	6	12	10	10	11	9	0.739918	98/100	NonOverlappingTemplate
14	10	11	10	11	8	9	4	17	6	0.191687	100/100	NonOverlappingTemplate
8	3	10	7	13	9	14	8	16	12	0.153763	99/100	NonOverlappingTemplate
13	7	9	10	8	5	8	8	9	23	0.007160	99/100	NonOverlappingTemplate
10	13	8	11	9	6	8	10	12	13	0.851383	98/100	NonOverlappingTemplate
11	9	6	10	10	17	13	13	8	3	0.129620	99/100	NonOverlappingTemplate
15	11	10	8	12	8	7	10	10	9	0.851383	100/100	NonOverlappingTemplate
14	7	7	16	9	9	11	10	6	11	0.437274	98/100	NonOverlappingTemplate
13	8	10	6	10	16	9	9	10	9	0.657933	100/100	NonOverlappingTemplate
12	8	7	8	14	9	13	9	11	9	0.834308	99/100	NonOverlappingTemplate
10	11	9	12	3	9	11	15	4	16	0.080519	97/100	NonOverlappingTemplate
3	8	12	14	10	10	14	11	11	7	0.350485	100/100	NonOverlappingTemplate
13	4	13	9	12	14	2	12	9	12	0.096578	98/100	NonOverlappingTemplate
14	8	7	9	6	12	7	15	11	11	0.474986	99/100	NonOverlappingTemplate
6	11	17	5	6	10	8	8	17	12	0.051942	100/100	NonOverlappingTemplate
15	9	11	11	12	6	14	6	11	5	0.304126	100/100	NonOverlappingTemplate
12	16	8	9	9	13	7	11	8	7	0.554420	97/100	NonOverlappingTemplate
9	10	17	6	11	14	7	3	15	8	0.048716	99/100	NonOverlappingTemplate
9	17	11	10	8	9	9	6	11	10	0.595549	99/100	NonOverlappingTemplate
6	9	10	9	11	14	8	16	6	11	0.419021	99/100	NonOverlappingTemplate
5	10	16	13	9	7	12	12	10	6	0.319084	100/100	NonOverlappingTemplate
11	12	10	11	7	11	10	9	9	10	0.994250	100/100	NonOverlappingTemplate
13	13	9	10	7	7	13	9	11	8	0.816537	100/100	NonOverlappingTemplate
8	11	10	10	8	9	11	12	8	13	0.971699	100/100	NonOverlappingTemplate
14	8	8	9	12	10	9	9	12	9	0.935716	98/100	NonOverlappingTemplate
15	12	8	8	10	15	4	7	14	7	0.153763	100/100	NonOverlappingTemplate
9	7	10	12	10	8	13	15	8	8	0.739918	100/100	NonOverlappingTemplate
11	8	10	9	16	11	8	12	7	8	0.699313	99/100	NonOverlappingTemplate
16	8	7	13	12	11	9	11	9	4	0.334538	100/100	NonOverlappingTemplate
11	8	11	10	10	9	10	14	13	4	0.657933	99/100	NonOverlappingTemplate
7	10	16	13	9	7	8	14	8	8	0.419021	99/100	NonOverlappingTemplate
9	12	8	17	17	12	5	6	5	9	0.037566	99/100	NonOverlappingTemplate
10	11	11	12	10	8	12	8	8	10	0.987896	100/100	NonOverlappingTemplate
7	9	6	14	15	8	9	11	17	4	0.071177	99/100	NonOverlappingTemplate
9	11	7	12	13	4	8	11	15	10	0.437274	98/100	NonOverlappingTemplate
5	12	11	14	14	7	8	13	5	11	0.275709	100/100	NonOverlappingTemplate
11	18	16	8	9	8	9	5	7	9	0.102526	99/100	NonOverlappingTemplate
12	14	14	5	16	6	8	8	13	4	0.055361	99/100	NonOverlappingTemplate
13	10	7	10	11	12	11	8	9	9	0.964295	99/100	NonOverlappingTemplate
8	12	14	12	7	13	14	9	4	7	0.289667	99/100	NonOverlappingTemplate
4	7	13	13	6	13	10	13	14	7	0.202268	100/100	NonOverlappingTemplate
10	13	12	10	11	15	12	4	5	8	0.289667	99/100	NonOverlappingTemplate
23	8	10	11	6	9	11	6	8	8	0.010237	99/100	NonOverlappingTemplate
12	13	7	10	11	13	10	6	9	9	0.834308	100/100	NonOverlappingTemplate
7	14	7	9	15	14	12	6	8	8	0.319084	100/100	NonOverlappingTemplate
10	12	11	11	11	7	3	14	13	8	0.401199	99/100	NonOverlappingTemplate
7	12	11	5	10	13	13	12	8	9	0.678686	100/100	NonOverlappingTemplate
7	11	12	9	8	12	11	11	10	9	0.978072	99/100	NonOverlappingTemplate
9	13	6	12	11	9	9	8	11	12	0.897763	98/100	NonOverlappingTemplate
8	16	10	9	9	10	9	10	8	11	0.851383	99/100	NonOverlappingTemplate
5	7	13	9	12	10	11	11	13	9	0.739918	99/100	NonOverlappingTemplate
11	12	10	14	10	10	8	11	6	8	0.867692	97/100	NonOverlappingTemplate
8	14	11	15	8	13	8	9	6	8	0.494392	100/100	NonOverlappingTemplate
13	11	7	7	8	9	13	10	15	7	0.574903	100/100	NonOverlappingTemplate
13	11	11	11	5	9	10	12	12	6	0.719747	100/100	NonOverlappingTemplate
14	15	10	10	10	6	6	9	8	12	0.514124	99/100	NonOverlappingTemplate
14	14	8	8	17	8	8	6	8	9	0.224821	97/100	NonOverlappingTemplate
10	6	11	14	10	10	8	15	8	8	0.637119	100/100	NonOverlappingTemplate
10	11	11	6	11	12	10	8	13	8	0.911413	99/100	NonOverlappingTemplate
12	9	8	8	8	18	10	9	9	9	0.494392	99/100	NonOverlappingTemplate
12	8	12	10	11	12	7	7	7	14	0.739918	98/100	NonOverlappingTemplate
11	13	9	12	9	5	8	12	14	7	0.595549	99/100	OverlappingTemplate
6	14	13	10	8	9	13	7	13	7	0.514124	99/100	Universal
8	6	3	14	16	7	8	9	13	16	0.035174	99/100	ApproximateEntropy
7	6	7	9	5	6	5	3	10	5	0.689019	62/63	RandomExcursions

14	4	5	7	5	9	4	5	4	6	0.095617	61/63	RandomExcursions
5	4	13	1	6	5	9	9	7	4	0.046169	63/63	RandomExcursions
6	7	6	4	6	6	4	10	5	9	0.756476	63/63	RandomExcursions
5	8	7	11	3	6	6	5	7	5	0.619772	62/63	RandomExcursions
7	10	8	7	6	6	8	3	2	6	0.484646	63/63	RandomExcursions
9	5	8	7	2	4	5	10	7	6	0.452799	63/63	RandomExcursions
7	4	9	6	9	7	5	4	5	7	0.819544	61/63	RandomExcursions
8	6	3	6	10	5	4	9	5	7	0.585209	62/63	RandomExcursionsVariant
8	8	3	8	2	9	6	5	6	8	0.484646	63/63	RandomExcursionsVariant
7	13	6	2	2	7	2	11	6	7	0.013411	63/63	RandomExcursionsVariant
7	11	9	5	1	8	5	9	5	3	0.116519	63/63	RandomExcursionsVariant
7	14	5	4	4	11	6	6	4	2	0.018969	62/63	RandomExcursionsVariant
9	9	3	6	10	4	9	3	4	6	0.242986	62/63	RandomExcursionsVariant
8	10	2	7	3	6	8	8	6	5	0.422034	62/63	RandomExcursionsVariant
9	5	4	8	6	7	7	6	4	7	0.900104	62/63	RandomExcursionsVariant
5	9	10	4	3	7	8	6	7	4	0.517442	63/63	RandomExcursionsVariant
6	13	4	5	4	6	5	7	8	5	0.287306	63/63	RandomExcursionsVariant
5	11	7	1	5	4	7	2	11	10	0.023812	63/63	RandomExcursionsVariant
7	3	4	6	7	6	4	11	5	10	0.337162	63/63	RandomExcursionsVariant
7	6	1	5	7	6	6	5	12	8	0.242986	63/63	RandomExcursionsVariant
6	3	5	5	5	9	8	5	10	7	0.619772	61/63	RandomExcursionsVariant
4	5	6	8	5	5	6	9	6	9	0.848588	61/63	RandomExcursionsVariant
4	7	8	5	4	9	10	6	5	5	0.654467	62/63	RandomExcursionsVariant
5	7	9	4	8	6	7	5	6	6	0.941144	63/63	RandomExcursionsVariant
6	8	4	8	7	5	8	7	4	6	0.922036	63/63	RandomExcursionsVariant
6	6	12	21	6	14	10	11	5	9	0.010237	99/100	Serial
8	8	12	8	11	10	9	14	11	9	0.935716	100/100	Serial
9	10	15	7	8	17	8	10	6	10	0.289667	99/100	LinearComplexity

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 96 for a
sample size = 100 binary sequences.

The minimum pass rate for the random excursion (variant) test
is approximately = 60 for a sample size = 63 binary sequences.

For further guidelines construct a probability table using the MAPLE program
provided in the addendum section of the documentation.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Результати тестування вихідної послідовності циклових ключів СРК шифру «Кипарис-256»

```
------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
   generator is <KeySchedule256.dat>
------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
 11  16   6  13   8  12  10   9  10   5  0.383827     99/100     Frequency
 12  11   8  12   7  10  13  11   8   8  0.911413     99/100     BlockFrequency
 10  12  14  10   7  14   6   7   8  12  0.554420     99/100     CumulativeSums
  9  12  14  14  10   7   4  15   7   8  0.213309     99/100     CumulativeSums
 12  13  10  12   9   5   7  12  12   8  0.699313     98/100     Runs
 11   7  12   7  10  12  13   4  12  12  0.534146     98/100     LongestRun
 14   8   8  10   8  10  12   7   8  15  0.637119     98/100     Rank
  8   6  12   9  16  10   9  11  11   8  0.657933     98/100     FFT
 12   5  12  10  12   9   9   9  12  10  0.883171     99/100     NonOverlappingTemplate
 12   9  13  12   8  10   7   6  16   7  0.419021     99/100     NonOverlappingTemplate
  9   8   8  13   9  13   9  10  13   8  0.897763    100/100     NonOverlappingTemplate
```

11	11	7	11	13	10	10	10	9	8	0.978072	99/100	NonOverlappingTemplate
13	7	6	7	15	12	10	10	13	7	0.437274	99/100	NonOverlappingTemplate
14	8	16	6	12	9	9	6	13	7	0.262249	97/100	NonOverlappingTemplate
9	11	11	5	10	15	7	12	11	9	0.657933	100/100	NonOverlappingTemplate
6	16	11	6	11	10	12	11	10	7	0.494392	100/100	NonOverlappingTemplate
9	15	13	9	7	12	6	6	12	11	0.474986	100/100	NonOverlappingTemplate
10	8	15	17	9	11	6	9	7	8	0.275709	99/100	NonOverlappingTemplate
11	14	5	11	14	4	15	8	7	11	0.145326	100/100	NonOverlappingTemplate
10	9	13	7	11	11	11	7	17	4	0.236810	100/100	NonOverlappingTemplate
7	13	8	16	8	9	7	6	9	17	0.129620	100/100	NonOverlappingTemplate
10	11	9	7	17	6	13	9	9	9	0.455937	100/100	NonOverlappingTemplate
11	7	10	9	10	13	11	12	10	7	0.946308	100/100	NonOverlappingTemplate
6	14	13	11	13	7	10	4	10	12	0.350485	100/100	NonOverlappingTemplate
14	9	15	10	8	12	8	6	8	10	0.595549	100/100	NonOverlappingTemplate
7	9	12	11	8	11	14	10	10	8	0.911413	99/100	NonOverlappingTemplate
14	11	8	8	5	12	8	10	12	12	0.678686	98/100	NonOverlappingTemplate
12	11	13	9	11	12	8	7	5	12	0.719747	97/100	NonOverlappingTemplate
13	6	9	9	20	11	5	9	9	9	0.075719	100/100	NonOverlappingTemplate
11	7	9	9	9	7	12	10	12	14	0.867692	100/100	NonOverlappingTemplate
9	12	10	12	10	10	9	4	11	13	0.779188	99/100	NonOverlappingTemplate
10	15	9	11	6	10	8	7	12	12	0.699313	99/100	NonOverlappingTemplate
7	8	11	12	12	8	10	13	11	8	0.911413	100/100	NonOverlappingTemplate
7	6	11	7	14	8	7	18	9	13	0.129620	100/100	NonOverlappingTemplate
9	10	7	13	10	13	8	11	6	13	0.759756	99/100	NonOverlappingTemplate
8	10	13	7	13	11	9	8	6	15	0.554420	99/100	NonOverlappingTemplate
13	10	6	7	8	12	13	13	7	11	0.637119	100/100	NonOverlappingTemplate
9	16	8	10	8	9	9	13	8	10	0.739918	100/100	NonOverlappingTemplate
10	13	12	8	9	13	6	8	9	12	0.816537	100/100	NonOverlappingTemplate
9	13	19	6	7	11	7	13	7	8	0.096578	100/100	NonOverlappingTemplate
10	10	8	9	10	14	12	9	10	8	0.964295	100/100	NonOverlappingTemplate
14	9	12	8	6	11	6	12	9	13	0.616305	100/100	NonOverlappingTemplate
6	12	11	6	11	7	10	12	12	13	0.699313	100/100	NonOverlappingTemplate
17	13	6	12	8	8	12	11	7	6	0.236810	98/100	NonOverlappingTemplate
12	10	14	10	5	7	14	12	6	10	0.437274	100/100	NonOverlappingTemplate
6	10	16	9	8	15	14	9	4	9	0.137282	100/100	NonOverlappingTemplate
12	9	10	10	9	11	5	11	15	8	0.719747	100/100	NonOverlappingTemplate
5	11	7	11	9	11	11	13	10	12	0.816537	99/100	NonOverlappingTemplate
7	9	11	14	13	13	7	10	9	7	0.699313	100/100	NonOverlappingTemplate
12	6	8	9	11	9	9	14	15	7	0.554420	97/100	NonOverlappingTemplate
4	11	13	10	9	12	11	12	7	11	0.678686	100/100	NonOverlappingTemplate
8	10	7	7	18	12	13	7	5	13	0.115387	100/100	NonOverlappingTemplate
5	6	4	12	15	12	11	8	13	14	0.122325	100/100	NonOverlappingTemplate
7	15	10	11	8	7	12	10	7	13	0.637119	100/100	NonOverlappingTemplate
8	7	10	10	10	11	12	12	8	12	0.964295	98/100	NonOverlappingTemplate
10	11	8	10	6	8	11	9	10	17	0.574903	100/100	NonOverlappingTemplate
9	12	10	7	8	13	9	8	16	8	0.616305	100/100	NonOverlappingTemplate
14	9	8	11	10	10	14	4	9	11	0.574903	99/100	NonOverlappingTemplate
7	6	9	12	15	13	8	10	9	11	0.637119	100/100	NonOverlappingTemplate
11	13	16	4	12	10	8	12	10	4	0.162606	99/100	NonOverlappingTemplate
14	8	11	14	6	8	13	10	7	9	0.574903	99/100	NonOverlappingTemplate
4	15	11	9	12	7	11	8	8	15	0.275709	100/100	NonOverlappingTemplate
5	12	13	13	8	20	10	5	8	6	0.020548	100/100	NonOverlappingTemplate
5	9	11	13	14	14	3	9	11	11	0.213309	100/100	NonOverlappingTemplate
7	9	8	16	9	9	10	14	13	5	0.334538	100/100	NonOverlappingTemplate
9	7	16	9	13	9	9	13	7	8	0.534146	99/100	NonOverlappingTemplate
13	11	7	7	11	4	10	14	10	13	0.437274	99/100	NonOverlappingTemplate
11	12	12	11	10	13	10	8	5	8	0.816537	99/100	NonOverlappingTemplate
13	8	9	13	10	15	7	9	9	7	0.657933	97/100	NonOverlappingTemplate
16	9	13	9	10	8	8	12	6	9	0.574903	98/100	NonOverlappingTemplate
11	11	12	9	12	11	5	9	8	12	0.867692	99/100	NonOverlappingTemplate
7	10	3	14	9	11	12	11	10	13	0.437274	98/100	NonOverlappingTemplate
9	9	8	10	11	5	13	8	14	13	0.637119	100/100	NonOverlappingTemplate
10	9	6	8	12	15	13	10	7	10	0.657933	99/100	NonOverlappingTemplate
10	10	6	6	6	10	8	20	12	12	0.066882	100/100	NonOverlappingTemplate
16	12	10	11	9	11	8	11	7	5	0.514124	97/100	NonOverlappingTemplate
10	11	10	13	11	11	8	5	14	7	0.678686	98/100	NonOverlappingTemplate

9	10	9	5	9	10	12	10	13	13	0.834308	98/100	NonOverlappingTemplate
10	10	12	6	12	14	7	12	7	10	0.719747	99/100	NonOverlappingTemplate
15	7	14	7	7	11	8	13	11	7	0.419021	98/100	NonOverlappingTemplate
7	11	12	5	9	7	11	15	10	13	0.494392	100/100	NonOverlappingTemplate
10	3	8	12	10	16	9	10	12	10	0.366918	100/100	NonOverlappingTemplate
12	5	12	10	12	9	10	8	12	10	0.867692	99/100	NonOverlappingTemplate
9	9	12	11	9	13	15	5	10	7	0.574903	97/100	NonOverlappingTemplate
6	7	9	9	16	6	13	12	15	7	0.181557	100/100	NonOverlappingTemplate
8	5	13	9	7	15	15	8	6	14	0.145326	99/100	NonOverlappingTemplate
13	9	8	14	13	7	7	12	9	8	0.678686	100/100	NonOverlappingTemplate
12	7	7	10	14	13	7	13	7	10	0.595549	99/100	NonOverlappingTemplate
5	14	7	18	14	16	7	7	7	5	0.009535	98/100	NonOverlappingTemplate
8	9	13	14	9	9	8	8	12	10	0.883171	99/100	NonOverlappingTemplate
8	11	4	10	6	9	15	13	10	14	0.289667	97/100	NonOverlappingTemplate
7	11	9	5	13	15	9	10	13	8	0.494392	100/100	NonOverlappingTemplate
13	7	12	10	11	8	10	9	13	7	0.867692	99/100	NonOverlappingTemplate
11	5	13	9	11	15	7	11	8	10	0.574903	99/100	NonOverlappingTemplate
16	10	14	13	5	8	11	9	7	7	0.275709	99/100	NonOverlappingTemplate
14	5	8	11	17	8	11	6	8	12	0.191687	99/100	NonOverlappingTemplate
14	15	8	10	13	3	8	8	13	8	0.191687	100/100	NonOverlappingTemplate
11	7	10	16	12	6	17	9	6	6	0.096578	99/100	NonOverlappingTemplate
4	9	14	9	4	15	13	14	13	5	0.042808	100/100	NonOverlappingTemplate
15	9	12	8	7	9	12	11	9	8	0.798139	100/100	NonOverlappingTemplate
8	14	13	7	8	16	7	13	9	5	0.202268	100/100	NonOverlappingTemplate
16	11	5	12	8	8	10	9	13	8	0.455937	100/100	NonOverlappingTemplate
5	13	9	9	10	3	8	12	16	15	0.080519	99/100	NonOverlappingTemplate
13	3	9	7	7	9	15	10	17	10	0.085587	98/100	NonOverlappingTemplate
11	11	6	12	10	13	8	10	8	11	0.911413	99/100	NonOverlappingTemplate
8	14	7	6	6	11	13	12	13	10	0.494392	100/100	NonOverlappingTemplate
7	7	16	11	8	12	7	9	11	12	0.554420	99/100	NonOverlappingTemplate
10	11	6	10	9	14	7	15	8	10	0.616305	99/100	NonOverlappingTemplate
12	13	9	7	14	5	11	8	9	12	0.595549	100/100	NonOverlappingTemplate
8	10	18	8	3	10	9	9	13	12	0.137282	100/100	NonOverlappingTemplate
15	10	10	9	9	7	14	7	9	10	0.719747	97/100	NonOverlappingTemplate
15	8	7	16	12	8	7	10	9	8	0.383827	99/100	NonOverlappingTemplate
12	4	14	12	11	12	10	10	7	8	0.554420	99/100	NonOverlappingTemplate
9	10	9	12	17	8	8	10	7	10	0.616305	98/100	NonOverlappingTemplate
10	10	7	15	12	9	5	10	12	10	0.657933	99/100	NonOverlappingTemplate
9	9	8	10	10	9	11	14	13	7	0.897763	99/100	NonOverlappingTemplate
8	9	13	9	8	11	8	9	14	11	0.897763	99/100	NonOverlappingTemplate
7	5	12	14	15	11	7	10	10	9	0.437274	99/100	NonOverlappingTemplate
7	10	8	7	9	7	16	15	13	8	0.304126	99/100	NonOverlappingTemplate
17	6	6	13	8	9	16	7	12	6	0.066882	100/100	NonOverlappingTemplate
8	11	6	7	18	8	13	10	9	10	0.289667	99/100	NonOverlappingTemplate
12	6	15	9	6	6	14	13	6	13	0.171867	97/100	NonOverlappingTemplate
8	6	9	15	4	11	15	12	11	9	0.249284	98/100	NonOverlappingTemplate
11	12	10	6	8	9	8	13	17	6	0.319084	100/100	NonOverlappingTemplate
10	11	10	8	6	14	9	14	11	7	0.699313	98/100	NonOverlappingTemplate
8	6	13	6	15	10	10	6	17	9	0.137282	100/100	NonOverlappingTemplate
5	9	12	7	18	12	13	5	7	12	0.080519	99/100	NonOverlappingTemplate
12	11	10	12	6	9	9	14	9	8	0.851383	99/100	NonOverlappingTemplate
9	9	10	12	11	12	6	9	12	10	0.955835	99/100	NonOverlappingTemplate
10	12	12	10	9	7	7	9	10	14	0.883171	99/100	NonOverlappingTemplate
6	14	7	13	8	10	6	13	13	10	0.455937	100/100	NonOverlappingTemplate
12	10	6	9	8	9	20	11	6	9	0.108791	98/100	NonOverlappingTemplate
12	12	8	10	12	8	10	6	9	13	0.867692	100/100	NonOverlappingTemplate
16	9	15	10	8	7	8	14	6	7	0.213309	100/100	NonOverlappingTemplate
18	7	8	10	8	6	12	16	4	11	0.042808	100/100	NonOverlappingTemplate
10	13	6	6	14	11	12	8	8	12	0.595549	99/100	NonOverlappingTemplate
8	13	9	9	13	8	8	14	13	5	0.514124	98/100	NonOverlappingTemplate
10	9	9	11	7	9	15	8	8	14	0.719747	100/100	NonOverlappingTemplate
12	6	7	11	10	7	13	14	10	10	0.699313	100/100	NonOverlappingTemplate
8	12	10	10	14	14	8	8	8	8	0.779188	98/100	NonOverlappingTemplate
3	16	11	8	11	12	9	10	13	7	0.249284	100/100	NonOverlappingTemplate
13	6	6	12	9	11	8	14	10	11	0.657933	100/100	NonOverlappingTemplate
10	8	11	11	8	17	9	3	13	10	0.224821	97/100	NonOverlappingTemplate

```
 9   9  10   6  10   9  10  13  15   9  0.798139    100/100     NonOverlappingTemplate
12  14  15  11   5   9   7  11   6  10  0.366918     99/100     NonOverlappingTemplate
13   9  10  12  11   5  10  10  13   7  0.759756     99/100     NonOverlappingTemplate
12  16   8   9  10   7  11   7   8  12  0.616305     99/100     NonOverlappingTemplate
 9  10   7  10   6  17   7  15   9  10  0.275709     99/100     NonOverlappingTemplate
 8   3   9   8  15  11  12  11  13  10  0.366918     99/100     NonOverlappingTemplate
11   9  14   8   8  10   9  14   7  10  0.816537    100/100     NonOverlappingTemplate
19  11   9   9   7   7   8  11  10   9  0.289667     97/100     NonOverlappingTemplate
 9   5  13   9  11  14  10   7   8  14  0.514124    100/100     NonOverlappingTemplate
 5  13   9  10   7  10   6  10  16  14  0.262249    100/100     NonOverlappingTemplate
11  13   9   9  10  14  13   3  13   5  0.213309     99/100     NonOverlappingTemplate
 9  13  11  12   4  14   9   8  12   8  0.534146     99/100     NonOverlappingTemplate
10   3   8  12  10  16   9  10  12  10  0.366918    100/100     NonOverlappingTemplate
 9  11  13   8   9   8   7  12  14   9  0.834308    100/100     OverlappingTemplate
14  10   7   6   8  12   8  13  10  12  0.678686     97/100     Universal
11   5   7  11   8  17  11  10  12   8  0.366918     99/100     ApproximateEntropy
 9   5   2   6  11   5   5   3   8   2  0.051942     54/56      RandomExcursions
 9   5   2   3   5  10   2   7   8   5  0.085587     55/56      RandomExcursions
 6   5  12   2   3   8   5   4   4   7  0.075719     54/56      RandomExcursions
 5   5  10   7   7   5   6   3   3   5  0.494392     56/56      RandomExcursions
 5   4   8   7   8   5   9   6   2   2  0.236810     56/56      RandomExcursions
 3   8   8   9   5   4   4   6   3   6  0.419021     55/56      RandomExcursions
 4   4   9   6   7   7   3   7   3   6  0.534146     56/56      RandomExcursions
 4   3   5   6   9   6   3   5   7   8  0.534146     56/56      RandomExcursions
 5   6   2   5   4   6   8   8   7   5  0.657933     56/56      RandomExcursionsVariant
 6   4   5   2   3   5  10   9   8   4  0.153763     55/56      RandomExcursionsVariant
 7   4   5   2   2  11   6   8   5   6  0.122325     55/56      RandomExcursionsVariant
 7   7   1   6   6   4   7   5   7   6  0.616305     55/56      RandomExcursionsVariant
 6   4   7   6   7   5   3   5   7   6  0.911413     54/56      RandomExcursionsVariant
 5   5   8   6   6   7   8   8   0   3  0.191687     55/56      RandomExcursionsVariant
 6   4  11   6   3   6   9   8   1   2  0.026948     55/56      RandomExcursionsVariant
 5   8   8   8   5   3   5   5   4   5  0.699313     56/56      RandomExcursionsVariant
 8   8   6   3   8   6   5   2   5   5  0.494392     56/56      RandomExcursionsVariant
 6   5   5  10   6   4   7   6   5   2  0.494392     56/56      RandomExcursionsVariant
 4   5   7   9   7   3   5   4   8   4  0.534146     55/56      RandomExcursionsVariant
 4   8   4   9   4   6   5   6   5   5  0.739918     55/56      RandomExcursionsVariant
 5   3   9   3   6   8   8   3   6   5  0.383827     55/56      RandomExcursionsVariant
 5   2   5  10   6   6   6   5   7   4  0.494392     54/56      RandomExcursionsVariant
 4   2   6   6   8   6   4  12   3   5  0.085587     55/56      RandomExcursionsVariant
 4   3   4   5   8  10   8   4   6   4  0.319084     56/56      RandomExcursionsVariant
 3   6   4   3   8  12   5   7   5   3  0.085587     56/56      RandomExcursionsVariant
 5   5   4   5   9   3   6   7   6   6  0.779188     56/56      RandomExcursionsVariant
 7   8  12  15   8   7   9  13   9  12  0.637119     98/100     Serial
10  10  14   9  10   9  12  10   8   8  0.964295     99/100     Serial
 8  15   7   9   7  11  12   8   7  16  0.334538    100/100     LinearComplexity


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 96 for a
sample size = 100 binary sequences.

The minimum pass rate for the random excursion (variant) test
is approximately = 53 for a sample size = 56 binary sequences.

For further guidelines construct a probability table using the MAPLE program
provided in the addendum section of the documentation.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Результати тестування вихідної послідовності циклових ключів СРК шифру «Кипарис-512»

```
------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
   generator is <KeySchedule512.dat>
------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
 13   8   9   5  10  12  15   8  10  10  0.616305     98/100     Frequency
 10  14   6   6  11  10  13  15  10   5  0.289667    100/100     BlockFrequency
 12  10  11  10  10  13  11   7   5  11  0.834308     98/100     CumulativeSums
 15  10   7   9   9   6   8  16   9  11  0.401199     98/100     CumulativeSums
  8  10  14   6  10  10  13  10  10   9  0.867692     99/100     Runs
 12  10  15   7  11   8  12  11   9   5  0.595549     99/100     LongestRun
  4   8  11  14   9  10   8  11  13  12  0.574903     98/100     Rank
 13  10  14   5  11   9  13   5  11   9  0.455937     96/100     FFT
 16   6  10   7   6  11   9  11  17   7  0.129620     97/100     NonOverlappingTemplate
  8  11  10  10  13   7   9  10  12  10  0.971699    100/100     NonOverlappingTemplate
 14  12  17   7   8   8  11   9   6   8  0.289667    100/100     NonOverlappingTemplate
 10   8   8   7   8  12  17  14  11   5  0.236810     99/100     NonOverlappingTemplate
 17  12   9  17   8   7   7   5  10   8  0.080519     97/100     NonOverlappingTemplate
  9  11   9   6   9  11  10  17   7  11  0.534146     97/100     NonOverlappingTemplate
  7  12  11  14   7  12  10   7  11   9  0.798139    100/100     NonOverlappingTemplate
  4  12  11  12   7  14  12   8  10  10  0.554420     99/100     NonOverlappingTemplate
 11  10   7  12   8   8  11   8  14  11  0.883171     98/100     NonOverlappingTemplate
  9  10  10  11  10  18   6  10   9   7  0.419021    100/100     NonOverlappingTemplate
  7  12   7  12  12  13  11   7   5  14  0.437274     99/100     NonOverlappingTemplate
 14   6  13  12   7   8  14  11   7   8  0.455937     98/100     NonOverlappingTemplate
  9  10  12   9  10  19   8   9   9   5  0.224821     98/100     NonOverlappingTemplate
  9   7  14   5   8  14   5  14  10  14  0.171867    100/100     NonOverlappingTemplate
  8   7  10  10  14   9  12  10  13   7  0.816537    100/100     NonOverlappingTemplate
 13  13   5   8   7   9  11  11  16   7  0.319084     99/100     NonOverlappingTemplate
 12  12  12   7  12  10   8   7   8  12  0.867692     98/100     NonOverlappingTemplate
  9  10   8   7  10  12  13  10  10  11  0.971699     99/100     NonOverlappingTemplate
 11  10  12  16   4   6  10   8  11  12  0.334538     96/100     NonOverlappingTemplate
  8   9  10  12   9   7  11  16   9   9  0.759756     99/100     NonOverlappingTemplate
 11  10  10  11  12  11  11  12   6   6  0.883171     98/100     NonOverlappingTemplate
  5   9  15  11   9  12  12  12   9   6  0.514124     99/100     NonOverlappingTemplate
  9   9  13  12  16  17   7   4   9   4  0.032923     99/100     NonOverlappingTemplate
 11  15  11  12   9   5  14   6   6  11  0.304126     98/100     NonOverlappingTemplate
  9  10   7   9  17   8   5  15   6  14  0.102526     98/100     NonOverlappingTemplate
 10   9   7  12  10   6  13  11  11  11  0.897763     98/100     NonOverlappingTemplate
 15   9   9   7   8  10   9  13   8  12  0.759756     98/100     NonOverlappingTemplate
  8  13   7  19   9   9  10  10  11   4  0.115387     98/100     NonOverlappingTemplate
  9  11  12   5   9   9   9   9  14  13  0.739918     99/100     NonOverlappingTemplate
  8  10  10  17   8  17  10   4   6  10  0.071177     99/100     NonOverlappingTemplate
 11  13  12   6  12   3  14  11   9   9  0.334538     99/100     NonOverlappingTemplate
 11  13   8  10   9   6  13  10  13   7  0.759756     99/100     NonOverlappingTemplate
 10   9   9  11  18   8   9  10   8   8  0.534146     99/100     NonOverlappingTemplate
 11  10  11   9   7   8   8  19   9   8  0.304126     99/100     NonOverlappingTemplate
 13  12  15   5  13   9   8   9   7   9  0.455937     99/100     NonOverlappingTemplate
  9  10  11   6  11  14  10   9   9  11  0.924076     99/100     NonOverlappingTemplate
 15   7   9   8  10   9   9  12   9  12  0.834308     99/100     NonOverlappingTemplate
  9  10   8   8  11  13   9   5  11  16  0.514124    100/100     NonOverlappingTemplate
  8   9  18   8  10  14   8   4  10  11  0.162606    100/100     NonOverlappingTemplate
 10  12   9  11  11   9   9  10  13   6  0.946308     98/100     NonOverlappingTemplate
 11  10  20   8   8  13   4  11   3  12  0.013569     98/100     NonOverlappingTemplate
 11  17  14  10  10   7   6   6  11   8  0.262249     98/100     NonOverlappingTemplate
 14   9  11  10  11  10   8  10  11   6  0.911413    100/100     NonOverlappingTemplate
  8  13  11  10   4  10  13   9   9  13  0.637119     99/100     NonOverlappingTemplate
 13   5  16  12   8   8   6  10  13   9  0.289667    100/100     NonOverlappingTemplate
  8  10   9  12  12  12   8  12   8   9  0.964295    100/100     NonOverlappingTemplate
```

10	8	6	9	20	10	12	8	7	10	0.129620	97/100	NonOverlappingTemplate
13	11	5	12	10	8	6	7	12	16	0.289667	97/100	NonOverlappingTemplate
10	12	9	7	6	18	14	5	10	9	0.137282	100/100	NonOverlappingTemplate
6	11	12	13	12	6	10	15	5	10	0.350485	99/100	NonOverlappingTemplate
13	6	11	12	9	10	10	13	8	8	0.851383	98/100	NonOverlappingTemplate
6	8	9	11	9	14	9	11	8	15	0.637119	100/100	NonOverlappingTemplate
8	15	11	11	10	8	7	11	10	9	0.867692	100/100	NonOverlappingTemplate
7	11	11	6	10	9	12	12	7	15	0.637119	99/100	NonOverlappingTemplate
13	12	13	8	9	11	7	8	8	11	0.867692	100/100	NonOverlappingTemplate
4	7	11	10	13	8	8	12	16	11	0.319084	100/100	NonOverlappingTemplate
11	11	5	10	12	13	13	7	10	8	0.719747	100/100	NonOverlappingTemplate
10	8	10	12	11	14	13	7	4	11	0.534146	97/100	NonOverlappingTemplate
8	11	9	14	7	6	15	10	11	9	0.595549	99/100	NonOverlappingTemplate
8	7	7	12	11	11	11	6	13	14	0.637119	99/100	NonOverlappingTemplate
4	13	12	18	5	12	10	6	12	8	0.055361	100/100	NonOverlappingTemplate
7	12	5	12	14	7	11	11	14	7	0.401199	100/100	NonOverlappingTemplate
14	10	4	11	12	14	8	8	13	6	0.304126	98/100	NonOverlappingTemplate
10	7	11	9	13	11	14	8	8	9	0.867692	99/100	NonOverlappingTemplate
7	10	7	15	13	6	11	10	7	14	0.401199	100/100	NonOverlappingTemplate
5	8	15	15	15	6	13	6	9	8	0.090936	98/100	NonOverlappingTemplate
16	5	9	5	11	12	8	12	13	9	0.275709	100/100	NonOverlappingTemplate
6	12	9	11	12	10	5	14	10	11	0.657933	100/100	NonOverlappingTemplate
9	6	13	10	6	11	13	8	12	12	0.699313	100/100	NonOverlappingTemplate
9	15	11	10	10	13	9	6	7	10	0.719747	98/100	NonOverlappingTemplate
9	8	10	11	8	9	14	13	7	11	0.867692	99/100	NonOverlappingTemplate
12	10	7	11	13	9	13	9	9	7	0.883171	100/100	NonOverlappingTemplate
10	10	15	12	13	5	10	4	12	9	0.319084	100/100	NonOverlappingTemplate
15	11	11	8	8	13	10	8	3	13	0.304126	99/100	NonOverlappingTemplate
16	6	10	7	6	10	10	11	17	7	0.137282	97/100	NonOverlappingTemplate
8	8	8	9	6	12	6	12	14	17	0.224821	99/100	NonOverlappingTemplate
8	9	15	9	8	14	12	6	11	8	0.574903	98/100	NonOverlappingTemplate
6	8	9	13	5	13	12	12	14	8	0.419021	100/100	NonOverlappingTemplate
9	13	11	5	6	8	6	6	13	23	0.001628	99/100	NonOverlappingTemplate
7	11	7	9	6	9	16	9	11	15	0.350485	100/100	NonOverlappingTemplate
11	8	11	20	8	8	7	6	11	10	0.122325	99/100	NonOverlappingTemplate
13	10	12	11	4	11	10	2	10	17	0.058984	98/100	NonOverlappingTemplate
13	11	7	6	9	7	14	5	15	13	0.213309	98/100	NonOverlappingTemplate
11	12	15	3	12	8	4	9	12	14	0.108791	99/100	NonOverlappingTemplate
8	11	9	16	12	6	6	7	8	17	0.122325	99/100	NonOverlappingTemplate
13	4	9	8	8	14	12	11	12	9	0.534146	100/100	NonOverlappingTemplate
15	6	9	16	13	6	11	6	7	11	0.162606	98/100	NonOverlappingTemplate
9	12	11	11	6	14	8	10	9	10	0.883171	99/100	NonOverlappingTemplate
8	8	13	13	15	8	7	7	12	9	0.554420	100/100	NonOverlappingTemplate
12	13	9	5	11	9	10	10	10	11	0.897763	100/100	NonOverlappingTemplate
7	11	12	8	7	12	12	10	8	13	0.851383	98/100	NonOverlappingTemplate
5	12	12	10	15	14	7	6	12	7	0.262249	100/100	NonOverlappingTemplate
8	16	10	13	12	9	8	9	9	6	0.574903	100/100	NonOverlappingTemplate
5	15	3	11	9	19	11	9	9	9	0.028817	100/100	NonOverlappingTemplate
8	8	11	12	9	9	11	12	8	12	0.971699	99/100	NonOverlappingTemplate
5	12	14	12	10	9	9	8	10	11	0.779188	100/100	NonOverlappingTemplate
12	11	6	8	13	14	8	14	10	4	0.304126	99/100	NonOverlappingTemplate
9	16	7	12	5	11	6	12	11	11	0.366918	99/100	NonOverlappingTemplate
14	10	9	10	9	13	11	9	6	9	0.867692	100/100	NonOverlappingTemplate
7	4	9	10	16	10	9	17	5	13	0.055361	100/100	NonOverlappingTemplate
13	10	8	18	6	5	10	13	6	11	0.108791	99/100	NonOverlappingTemplate
7	10	14	6	8	5	15	6	14	15	0.085587	100/100	NonOverlappingTemplate
11	13	13	8	12	8	7	9	12	7	0.798139	99/100	NonOverlappingTemplate
11	7	8	11	7	8	12	9	13	14	0.759756	100/100	NonOverlappingTemplate
5	8	11	14	16	8	10	8	11	9	0.419021	99/100	NonOverlappingTemplate
13	9	10	7	14	13	12	6	6	10	0.534146	100/100	NonOverlappingTemplate
5	10	10	12	13	12	5	13	9	11	0.554420	100/100	NonOverlappingTemplate
7	2	9	12	8	15	8	11	13	15	0.102526	99/100	NonOverlappingTemplate
10	6	10	9	14	14	10	10	10	7	0.759756	99/100	NonOverlappingTemplate
8	12	12	9	12	7	14	10	10	6	0.759756	100/100	NonOverlappingTemplate
6	10	12	12	10	9	11	8	10	12	0.946308	98/100	NonOverlappingTemplate
9	14	9	8	7	11	10	8	13	11	0.867692	99/100	NonOverlappingTemplate

7	12	7	15	12	9	12	8	11	7	0.637119	100/100	NonOverlappingTemplate
4	8	7	11	7	11	9	18	16	9	0.062821	100/100	NonOverlappingTemplate
11	5	11	15	9	7	7	8	16	11	0.262249	98/100	NonOverlappingTemplate
12	7	9	18	8	5	7	16	11	7	0.062821	99/100	NonOverlappingTemplate
7	13	6	6	10	12	15	11	12	8	0.455937	99/100	NonOverlappingTemplate
7	11	9	12	14	14	7	10	12	4	0.383827	99/100	NonOverlappingTemplate
13	5	11	13	7	6	13	11	8	13	0.419021	99/100	NonOverlappingTemplate
16	6	10	11	9	9	12	11	10	6	0.574903	99/100	NonOverlappingTemplate
9	9	16	10	13	9	11	6	9	8	0.637119	100/100	NonOverlappingTemplate
8	7	9	6	10	3	13	13	14	17	0.062821	100/100	NonOverlappingTemplate
8	6	13	6	14	14	8	8	10	13	0.401199	100/100	NonOverlappingTemplate
9	8	7	8	11	13	6	11	10	17	0.401199	100/100	NonOverlappingTemplate
13	8	13	5	4	9	14	9	13	12	0.249284	99/100	NonOverlappingTemplate
6	14	7	8	12	9	12	10	10	12	0.759756	100/100	NonOverlappingTemplate
14	7	11	7	12	7	16	13	7	6	0.224821	97/100	NonOverlappingTemplate
9	9	7	4	11	13	10	19	9	9	0.122325	99/100	NonOverlappingTemplate
8	7	11	9	12	8	13	9	18	5	0.202268	99/100	NonOverlappingTemplate
7	9	13	11	7	6	12	16	12	7	0.366918	100/100	NonOverlappingTemplate
14	8	13	13	7	8	13	8	11	5	0.437274	98/100	NonOverlappingTemplate
9	11	10	3	14	8	14	6	12	13	0.236810	98/100	NonOverlappingTemplate
7	10	14	12	9	8	7	10	8	15	0.616305	99/100	NonOverlappingTemplate
12	10	9	10	8	15	9	12	9	6	0.779188	99/100	NonOverlappingTemplate
13	9	12	15	4	3	14	12	11	7	0.080519	98/100	NonOverlappingTemplate
9	13	18	11	8	7	6	15	10	3	0.037566	100/100	NonOverlappingTemplate
11	8	8	11	10	9	6	12	12	13	0.883171	100/100	NonOverlappingTemplate
10	10	9	12	13	9	4	10	15	8	0.534146	100/100	NonOverlappingTemplate
6	12	11	10	9	8	10	12	8	14	0.834308	100/100	NonOverlappingTemplate
15	9	11	7	8	7	12	8	11	12	0.719747	97/100	NonOverlappingTemplate
15	11	6	6	9	11	9	12	6	15	0.304126	97/100	NonOverlappingTemplate
9	10	12	14	9	14	9	11	7	5	0.595549	98/100	NonOverlappingTemplate
10	11	9	7	13	9	13	8	13	7	0.816537	99/100	NonOverlappingTemplate
6	10	10	6	9	13	16	10	10	10	0.554420	100/100	NonOverlappingTemplate
10	6	8	14	5	10	11	10	14	12	0.514124	99/100	NonOverlappingTemplate
9	13	16	8	8	6	10	8	9	13	0.494392	99/100	NonOverlappingTemplate
13	12	7	18	8	5	6	13	9	9	0.115387	98/100	NonOverlappingTemplate
15	11	9	10	8	13	10	8	3	13	0.334538	99/100	NonOverlappingTemplate
10	11	10	11	9	6	18	5	9	11	0.275709	100/100	OverlappingTemplate
11	8	13	16	7	7	8	10	11	9	0.595549	99/100	Universal
9	9	13	8	10	9	12	8	13	9	0.946308	100/100	ApproximateEntropy
9	6	7	5	10	7	1	7	5	4	0.364146	60/61	RandomExcursions
4	10	6	3	6	9	7	9	3	4	0.337162	59/61	RandomExcursions
3	10	7	8	7	6	5	8	4	3	0.517442	61/61	RandomExcursions
6	5	3	5	5	5	10	9	8	5	0.619772	61/61	RandomExcursions
7	8	5	7	3	5	8	6	9	3	0.689019	60/61	RandomExcursions
9	2	12	5	2	11	9	4	1	6	0.005166	59/61	RandomExcursions
9	3	11	6	5	8	8	4	5	2	0.204076	58/61	RandomExcursions
6	4	8	6	5	3	6	10	1	12	0.070445	61/61	RandomExcursions
7	7	8	3	5	6	3	11	3	8	0.311542	61/61	RandomExcursionsVariant
9	5	2	7	4	11	6	7	8	2	0.170294	61/61	RandomExcursionsVariant
7	7	4	6	5	9	7	2	5	9	0.619772	61/61	RandomExcursionsVariant
5	10	2	8	5	5	5	8	8	5	0.517442	60/61	RandomExcursionsVariant
5	9	5	3	6	11	7	4	4	7	0.422034	60/61	RandomExcursionsVariant
6	4	9	2	10	8	3	6	6	7	0.364146	61/61	RandomExcursionsVariant
6	6	5	2	7	9	8	6	9	3	0.517442	61/61	RandomExcursionsVariant
6	8	5	3	0	13	10	7	4	5	0.016911	60/61	RandomExcursionsVariant
5	6	9	2	5	8	6	8	6	6	0.756476	60/61	RandomExcursionsVariant
6	8	7	4	4	7	8	5	7	5	0.941144	61/61	RandomExcursionsVariant
10	3	7	7	5	7	7	5	4	6	0.756476	60/61	RandomExcursionsVariant
9	4	9	7	6	5	9	2	5	5	0.484646	60/61	RandomExcursionsVariant
6	4	7	9	6	5	9	3	5	7	0.756476	60/61	RandomExcursionsVariant
3	11	4	5	4	6	6	9	9	4	0.287306	61/61	RandomExcursionsVariant
9	7	2	3	7	9	5	8	5	6	0.484646	61/61	RandomExcursionsVariant
11	5	3	5	7	2	9	8	4	7	0.222869	60/61	RandomExcursionsVariant
8	8	5	4	4	9	6	4	5	8	0.756476	58/61	RandomExcursionsVariant
10	6	3	5	5	5	8	7	8	4	0.654467	59/61	RandomExcursionsVariant
8	12	11	10	14	9	10	9	9	8	0.955835	99/100	Serial

```
 17   7  10   9  10  12   5  11   8  11  0.401199     98/100     Serial
 11  10   8   8   6  13  12   8  14  10  0.759756     97/100     LinearComplexity

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 96 for a
sample size = 100 binary sequences.

The minimum pass rate for the random excursion (variant) test
is approximately = 58 for a sample size = 61 binary sequences.

For further guidelines construct a probability table using the MAPLE program
provided in the addendum section of the documentation.
```

ДОДАТОК В. ВИХІДНИЙ КОД ПРОГРАМИ, ЩО МІСТИТЬ РЕАЛІЗАЦІЮ БЛОКОВОГО ШИФРУ «КИПАРИС» ТА ДОСЛІДЖЕННЯ ЙОГО ВЛАСТИВОСТЕЙ

//Реалізація алгоритму «Кипарис-256» та дослідження його властивостей (cypress256.cpp)

```
#include <iostream>
#include <fstream>

#include "cypress256.h"
#include "time_measure.h"

using namespace std;

#define ROTL32(v, n) ((v) << (n)) | ((v) >> (32 - (n)))

//Циклова функція шифру «Кипарис-256»
#define HALFROUND32(a, b, c, d) { \
	a += b; d ^= a; d = ROTL32(d, 16); \
	c += d; b ^= c; b = ROTL32(b, 12); \
	a += b; d ^= a; d = ROTL32(d, 8); \
	c += d; b ^= c; b = ROTL32(b, 7); \
}

//Формування проміжного ключа для шифру «Кипарис-256»
void Ksigma256(ui32 masterKey[8], ui32 ksigma[4])
{
	ui32 state[4], k0[4], k1[4];

	state[0] = 0;
	state[1] = 0;
	state[2] = 0;
	state[3] = 1;

	k0[0] = masterKey[0];
	k0[1] = masterKey[1];
	k0[2] = masterKey[2];
	k0[3] = masterKey[3];
	k1[0] = masterKey[4];
	k1[1] = masterKey[5];
	k1[2] = masterKey[6];
	k1[3] = masterKey[7];

	state[0] += k0[0];
	state[1] += k0[1];
	state[2] += k0[2];
	state[3] += k0[3];

	HALFROUND32(state[0], state[1], state[2], state[3]);
	HALFROUND32(state[0], state[1], state[2], state[3]);

	state[0] ^= k1[0];
	state[1] ^= k1[1];
	state[2] ^= k1[2];
	state[3] ^= k1[3];
```

```
	HALFROUND32(state[0], state[1], state[2], state[3]);
	HALFROUND32(state[0], state[1], state[2], state[3]);

	state[0] += k0[0];
	state[1] += k0[1];
	state[2] += k0[2];
	state[3] += k0[3];

	HALFROUND32(state[0], state[1], state[2], state[3]);
	HALFROUND32(state[0], state[1], state[2], state[3]);

	ksigma[0] = state[0];
	ksigma[1] = state[1];
	ksigma[2] = state[2];
	ksigma[3] = state[3];
}

//Формування циклових ключів шифру «Кипарис-256»
void MakeRoundKeys256(ui32 ksigma[4], ui32 masterKey[8], ui32 roundKeys[10][4])
{
	ui32 state[4], k[4], initData[8], tmp = 0;

	ui32 tmv[4] = {0x000F000F, 0x000F000F, 0x000F000F, 0x000F000F};

	initData[0] = masterKey[0];
	initData[1] = masterKey[1];
	initData[2] = masterKey[2];
	initData[3] = masterKey[3];
	initData[4] = masterKey[4];
	initData[5] = masterKey[5];
	initData[6] = masterKey[6];
	initData[7] = masterKey[7];

	for(int i=0; i<=8; i+=2)
	{
		state[0] = initData[0];
		state[1] = initData[1];
		state[2] = initData[2];
		state[3] = initData[3];

		k[0] = ksigma[0];
		k[1] = ksigma[1];
		k[2] = ksigma[2];
		k[3] = ksigma[3];

		k[0] += tmv[0];
		k[1] += tmv[1];
		k[2] += tmv[2];
		k[3] += tmv[3];

		state[0] += k[0];
		state[1] += k[1];
		state[2] += k[2];
		state[3] += k[3];

		HALFROUND32(state[0], state[1], state[2], state[3]);
		HALFROUND32(state[0], state[1], state[2], state[3]);

		state[0] ^= k[0];
		state[1] ^= k[1];
		state[2] ^= k[2];
		state[3] ^= k[3];

		HALFROUND32(state[0], state[1], state[2], state[3]);
```

```
HALFROUND32(state[0], state[1], state[2], state[3]);

state[0] += k[0];
state[1] += k[1];
state[2] += k[2];
state[3] += k[3];

roundKeys[i][0] = state[0];
roundKeys[i][1] = state[1];
roundKeys[i][2] = state[2];
roundKeys[i][3] = state[3];

//next key

tmv[0] <<= 1;
tmv[1] <<= 1;
tmv[2] <<= 1;
tmv[3] <<= 1;

state[0] = initData[4];
state[1] = initData[5];
state[2] = initData[6];
state[3] = initData[7];

k[0] = ksigma[0];
k[1] = ksigma[1];
k[2] = ksigma[2];
k[3] = ksigma[3];

k[0] += tmv[0];
k[1] += tmv[1];
k[2] += tmv[2];
k[3] += tmv[3];

state[0] += k[0];
state[1] += k[1];
state[2] += k[2];
state[3] += k[3];

HALFROUND32(state[0], state[1], state[2], state[3]);
HALFROUND32(state[0], state[1], state[2], state[3]);

state[0] ^= k[0];
state[1] ^= k[1];
state[2] ^= k[2];
state[3] ^= k[3];

HALFROUND32(state[0], state[1], state[2], state[3]);
HALFROUND32(state[0], state[1], state[2], state[3]);

state[0] += k[0];
state[1] += k[1];
state[2] += k[2];
state[3] += k[3];

roundKeys[i+1][0] = state[0];
roundKeys[i+1][1] = state[1];
roundKeys[i+1][2] = state[2];
roundKeys[i+1][3] = state[3];

tmv[0] <<= 1;
tmv[1] <<= 1;
tmv[2] <<= 1;
tmv[3] <<= 1;
```

```
            tmp = initData[0];
            initData[0] = initData[1];
            initData[1] = initData[2];
            initData[2] = initData[3];
            initData[3] = initData[4];
            initData[4] = initData[5];
            initData[5] = initData[6];
            initData[6] = initData[7];
            initData[7] = tmp;
      }
}

//Зашифрування шифром «Кипарис-256»
void CypressEncryption256(ui32 plaintext[8], ui32 ciphertext[8], ui32
roundKeys[10][4])
{
      ui32 L[4], R[4], tmp[4];

      L[0] = plaintext[0];
      L[1] = plaintext[1];
      L[2] = plaintext[2];
      L[3] = plaintext[3];
      R[0] = plaintext[4];
      R[1] = plaintext[5];
      R[2] = plaintext[6];
      R[3] = plaintext[7];

      tmp[0] = L[0];
      tmp[1] = L[1];
      tmp[2] = L[2];
      tmp[3] = L[3];
      L[0] ^= roundKeys[0][0];
      L[1] ^= roundKeys[0][1];
      L[2] ^= roundKeys[0][2];
      L[3] ^= roundKeys[0][3];
      HALFROUND32(L[0], L[1], L[2], L[3]);
      HALFROUND32(L[0], L[1], L[2], L[3]);
      L[0] ^= R[0];
      L[1] ^= R[1];
      L[2] ^= R[2];
      L[3] ^= R[3];
      R[0] = tmp[0];
      R[1] = tmp[1];
      R[2] = tmp[2];
      R[3] = tmp[3];

      tmp[0] = L[0];
      tmp[1] = L[1];
      tmp[2] = L[2];
      tmp[3] = L[3];
      L[0] ^= roundKeys[1][0];
      L[1] ^= roundKeys[1][1];
      L[2] ^= roundKeys[1][2];
      L[3] ^= roundKeys[1][3];
      HALFROUND32(L[0], L[1], L[2], L[3]);
      HALFROUND32(L[0], L[1], L[2], L[3]);
      L[0] ^= R[0];
      L[1] ^= R[1];
      L[2] ^= R[2];
      L[3] ^= R[3];
      R[0] = tmp[0];
      R[1] = tmp[1];
      R[2] = tmp[2];
      R[3] = tmp[3];
```

```
tmp[0] = L[0];
tmp[1] = L[1];
tmp[2] = L[2];
tmp[3] = L[3];
L[0] ^= roundKeys[2][0];
L[1] ^= roundKeys[2][1];
L[2] ^= roundKeys[2][2];
L[3] ^= roundKeys[2][3];
HALFROUND32(L[0], L[1], L[2], L[3]);
HALFROUND32(L[0], L[1], L[2], L[3]);
L[0] ^= R[0];
L[1] ^= R[1];
L[2] ^= R[2];
L[3] ^= R[3];
R[0] = tmp[0];
R[1] = tmp[1];
R[2] = tmp[2];
R[3] = tmp[3];

tmp[0] = L[0];
tmp[1] = L[1];
tmp[2] = L[2];
tmp[3] = L[3];
L[0] ^= roundKeys[3][0];
L[1] ^= roundKeys[3][1];
L[2] ^= roundKeys[3][2];
L[3] ^= roundKeys[3][3];
HALFROUND32(L[0], L[1], L[2], L[3]);
HALFROUND32(L[0], L[1], L[2], L[3]);
L[0] ^= R[0];
L[1] ^= R[1];
L[2] ^= R[2];
L[3] ^= R[3];
R[0] = tmp[0];
R[1] = tmp[1];
R[2] = tmp[2];
R[3] = tmp[3];

tmp[0] = L[0];
tmp[1] = L[1];
tmp[2] = L[2];
tmp[3] = L[3];
L[0] ^= roundKeys[4][0];
L[1] ^= roundKeys[4][1];
L[2] ^= roundKeys[4][2];
L[3] ^= roundKeys[4][3];
HALFROUND32(L[0], L[1], L[2], L[3]);
HALFROUND32(L[0], L[1], L[2], L[3]);
L[0] ^= R[0];
L[1] ^= R[1];
L[2] ^= R[2];
L[3] ^= R[3];
R[0] = tmp[0];
R[1] = tmp[1];
R[2] = tmp[2];
R[3] = tmp[3];

tmp[0] = L[0];
tmp[1] = L[1];
tmp[2] = L[2];
tmp[3] = L[3];
L[0] ^= roundKeys[5][0];
L[1] ^= roundKeys[5][1];
L[2] ^= roundKeys[5][2];
L[3] ^= roundKeys[5][3];
```

```
HALFROUND32(L[0], L[1], L[2], L[3]);
HALFROUND32(L[0], L[1], L[2], L[3]);
L[0] ^= R[0];
L[1] ^= R[1];
L[2] ^= R[2];
L[3] ^= R[3];
R[0] = tmp[0];
R[1] = tmp[1];
R[2] = tmp[2];
R[3] = tmp[3];

tmp[0] = L[0];
tmp[1] = L[1];
tmp[2] = L[2];
tmp[3] = L[3];
L[0] ^= roundKeys[6][0];
L[1] ^= roundKeys[6][1];
L[2] ^= roundKeys[6][2];
L[3] ^= roundKeys[6][3];
HALFROUND32(L[0], L[1], L[2], L[3]);
HALFROUND32(L[0], L[1], L[2], L[3]);
L[0] ^= R[0];
L[1] ^= R[1];
L[2] ^= R[2];
L[3] ^= R[3];
R[0] = tmp[0];
R[1] = tmp[1];
R[2] = tmp[2];
R[3] = tmp[3];

tmp[0] = L[0];
tmp[1] = L[1];
tmp[2] = L[2];
tmp[3] = L[3];
L[0] ^= roundKeys[7][0];
L[1] ^= roundKeys[7][1];
L[2] ^= roundKeys[7][2];
L[3] ^= roundKeys[7][3];
HALFROUND32(L[0], L[1], L[2], L[3]);
HALFROUND32(L[0], L[1], L[2], L[3]);
L[0] ^= R[0];
L[1] ^= R[1];
L[2] ^= R[2];
L[3] ^= R[3];
R[0] = tmp[0];
R[1] = tmp[1];
R[2] = tmp[2];
R[3] = tmp[3];

tmp[0] = L[0];
tmp[1] = L[1];
tmp[2] = L[2];
tmp[3] = L[3];
L[0] ^= roundKeys[8][0];
L[1] ^= roundKeys[8][1];
L[2] ^= roundKeys[8][2];
L[3] ^= roundKeys[8][3];
HALFROUND32(L[0], L[1], L[2], L[3]);
HALFROUND32(L[0], L[1], L[2], L[3]);
L[0] ^= R[0];
L[1] ^= R[1];
L[2] ^= R[2];
L[3] ^= R[3];
R[0] = tmp[0];
R[1] = tmp[1];
```

```
	R[2] = tmp[2];
	R[3] = tmp[3];

	tmp[0] = L[0];
	tmp[1] = L[1];
	tmp[2] = L[2];
	tmp[3] = L[3];
	L[0] ^= roundKeys[9][0];
	L[1] ^= roundKeys[9][1];
	L[2] ^= roundKeys[9][2];
	L[3] ^= roundKeys[9][3];

	HALFROUND32(L[0], L[1], L[2], L[3]);
	HALFROUND32(L[0], L[1], L[2], L[3]);
	L[0] ^= R[0];
	L[1] ^= R[1];
	L[2] ^= R[2];
	L[3] ^= R[3];
	R[0] = tmp[0];
	R[1] = tmp[1];
	R[2] = tmp[2];
	R[3] = tmp[3];

	ciphertext[0] = L[0];
	ciphertext[1] = L[1];
	ciphertext[2] = L[2];
	ciphertext[3] = L[3];
	ciphertext[4] = R[0];
	ciphertext[5] = R[1];
	ciphertext[6] = R[2];
	ciphertext[7] = R[3];
}

//Розшифрування шифром «Кипарис-256»
void CypressDecryption256(ui32 ciphertext[8], ui32 plaintext[8], ui32
roundKeys[10][4])
{
	ui32 L[4], R[4], tmp[4];

	R[0] = ciphertext[0];
	R[1] = ciphertext[1];
	R[2] = ciphertext[2];
	R[3] = ciphertext[3];
	L[0] = ciphertext[4];
	L[1] = ciphertext[5];
	L[2] = ciphertext[6];
	L[3] = ciphertext[7];

	tmp[0] = L[0];
	tmp[1] = L[1];
	tmp[2] = L[2];
	tmp[3] = L[3];
	L[0] ^= roundKeys[9][0];
	L[1] ^= roundKeys[9][1];
	L[2] ^= roundKeys[9][2];
	L[3] ^= roundKeys[9][3];

	HALFROUND32(L[0], L[1], L[2], L[3]);
	HALFROUND32(L[0], L[1], L[2], L[3]);
	L[0] ^= R[0];
	L[1] ^= R[1];
	L[2] ^= R[2];
	L[3] ^= R[3];
	R[0] = tmp[0];
	R[1] = tmp[1];
```

```
R[2] = tmp[2];
R[3] = tmp[3];

tmp[0] = L[0];
tmp[1] = L[1];
tmp[2] = L[2];
tmp[3] = L[3];
L[0] ^= roundKeys[8][0];
L[1] ^= roundKeys[8][1];
L[2] ^= roundKeys[8][2];
L[3] ^= roundKeys[8][3];
HALFROUND32(L[0], L[1], L[2], L[3]);
HALFROUND32(L[0], L[1], L[2], L[3]);
L[0] ^= R[0];
L[1] ^= R[1];
L[2] ^= R[2];
L[3] ^= R[3];
R[0] = tmp[0];
R[1] = tmp[1];
R[2] = tmp[2];
R[3] = tmp[3];

tmp[0] = L[0];
tmp[1] = L[1];
tmp[2] = L[2];
tmp[3] = L[3];
L[0] ^= roundKeys[7][0];
L[1] ^= roundKeys[7][1];
L[2] ^= roundKeys[7][2];
L[3] ^= roundKeys[7][3];
HALFROUND32(L[0], L[1], L[2], L[3]);
HALFROUND32(L[0], L[1], L[2], L[3]);
L[0] ^= R[0];
L[1] ^= R[1];
L[2] ^= R[2];
L[3] ^= R[3];
R[0] = tmp[0];
R[1] = tmp[1];
R[2] = tmp[2];
R[3] = tmp[3];

tmp[0] = L[0];
tmp[1] = L[1];
tmp[2] = L[2];
tmp[3] = L[3];
L[0] ^= roundKeys[6][0];
L[1] ^= roundKeys[6][1];
L[2] ^= roundKeys[6][2];
L[3] ^= roundKeys[6][3];
HALFROUND32(L[0], L[1], L[2], L[3]);
HALFROUND32(L[0], L[1], L[2], L[3]);
L[0] ^= R[0];
L[1] ^= R[1];
L[2] ^= R[2];
L[3] ^= R[3];
R[0] = tmp[0];
R[1] = tmp[1];
R[2] = tmp[2];
R[3] = tmp[3];

tmp[0] = L[0];
tmp[1] = L[1];
tmp[2] = L[2];
tmp[3] = L[3];
L[0] ^= roundKeys[5][0];
```

```
L[1] ^= roundKeys[5][1];
L[2] ^= roundKeys[5][2];
L[3] ^= roundKeys[5][3];
HALFROUND32(L[0], L[1], L[2], L[3]);
HALFROUND32(L[0], L[1], L[2], L[3]);
L[0] ^= R[0];
L[1] ^= R[1];
L[2] ^= R[2];
L[3] ^= R[3];
R[0] = tmp[0];
R[1] = tmp[1];
R[2] = tmp[2];
R[3] = tmp[3];

tmp[0] = L[0];
tmp[1] = L[1];
tmp[2] = L[2];
tmp[3] = L[3];
L[0] ^= roundKeys[4][0];
L[1] ^= roundKeys[4][1];
L[2] ^= roundKeys[4][2];
L[3] ^= roundKeys[4][3];
HALFROUND32(L[0], L[1], L[2], L[3]);
HALFROUND32(L[0], L[1], L[2], L[3]);
L[0] ^= R[0];
L[1] ^= R[1];
L[2] ^= R[2];
L[3] ^= R[3];
R[0] = tmp[0];
R[1] = tmp[1];
R[2] = tmp[2];
R[3] = tmp[3];

tmp[0] = L[0];
tmp[1] = L[1];
tmp[2] = L[2];
tmp[3] = L[3];
L[0] ^= roundKeys[3][0];
L[1] ^= roundKeys[3][1];
L[2] ^= roundKeys[3][2];
L[3] ^= roundKeys[3][3];
HALFROUND32(L[0], L[1], L[2], L[3]);
HALFROUND32(L[0], L[1], L[2], L[3]);
L[0] ^= R[0];
L[1] ^= R[1];
L[2] ^= R[2];
L[3] ^= R[3];
R[0] = tmp[0];
R[1] = tmp[1];
R[2] = tmp[2];
R[3] = tmp[3];

tmp[0] = L[0];
tmp[1] = L[1];
tmp[2] = L[2];
tmp[3] = L[3];
L[0] ^= roundKeys[2][0];
L[1] ^= roundKeys[2][1];
L[2] ^= roundKeys[2][2];
L[3] ^= roundKeys[2][3];
HALFROUND32(L[0], L[1], L[2], L[3]);
HALFROUND32(L[0], L[1], L[2], L[3]);
L[0] ^= R[0];
L[1] ^= R[1];
L[2] ^= R[2];
```

```
	L[3] ^= R[3];
	R[0] = tmp[0];
	R[1] = tmp[1];
	R[2] = tmp[2];
	R[3] = tmp[3];

	tmp[0] = L[0];
	tmp[1] = L[1];
	tmp[2] = L[2];
	tmp[3] = L[3];
	L[0] ^= roundKeys[1][0];
	L[1] ^= roundKeys[1][1];
	L[2] ^= roundKeys[1][2];
	L[3] ^= roundKeys[1][3];
	HALFROUND32(L[0], L[1], L[2], L[3]);
	HALFROUND32(L[0], L[1], L[2], L[3]);
	L[0] ^= R[0];
	L[1] ^= R[1];
	L[2] ^= R[2];
	L[3] ^= R[3];
	R[0] = tmp[0];
	R[1] = tmp[1];
	R[2] = tmp[2];
	R[3] = tmp[3];

	tmp[0] = L[0];
	tmp[1] = L[1];
	tmp[2] = L[2];
	tmp[3] = L[3];
	L[0] ^= roundKeys[0][0];
	L[1] ^= roundKeys[0][1];
	L[2] ^= roundKeys[0][2];
	L[3] ^= roundKeys[0][3];
	HALFROUND32(L[0], L[1], L[2], L[3]);
	HALFROUND32(L[0], L[1], L[2], L[3]);
	L[0] ^= R[0];
	L[1] ^= R[1];
	L[2] ^= R[2];
	L[3] ^= R[3];
	R[0] = tmp[0];
	R[1] = tmp[1];
	R[2] = tmp[2];
	R[3] = tmp[3];

	plaintext[0] = R[0];
	plaintext[1] = R[1];
	plaintext[2] = R[2];
	plaintext[3] = R[3];
	plaintext[4] = L[0];
	plaintext[5] = L[1];
	plaintext[6] = L[2];
	plaintext[7] = L[3];
}

//Зашифрування блока даних шифром «Кипарис-256»
void CypressEncryptOneBlock256(ui32 plaintext[8], ui32 roundKeys[10][4])
{
	CypressEncryption256(plaintext, plaintext, roundKeys);
}

extern ENCRYPTED_MEMORY encrypted_memory;

//Зашифрування блока пам’яті шифром «Кипарис-256» для вимірювання швидкодії
void TestSpeedCypress256()
{
```

```
	ui32 roundKeys[10][4], masterkey[8] = {0x1245A2A1, 0xC2BB7723, 0x2331A4A3,
0xB2CCAB34, 0x2A123A30, 0x2CCAA34C, 0xA124A368, 0xC2BB7723}, ksigma[4];

	Ksigma256(masterkey, ksigma);
	MakeRoundKeys256(ksigma, masterkey, roundKeys);

	for (unsigned int jj = 0; jj < number_of_reencryptions_in_memory; jj++)
		for(unsigned int ii = 0; ii < number_of_blocks_in_memory_256; ii++)
			CypressEncryptOneBlock256((unsigned int *)
&encrypted_memory.block256[ii], roundKeys);
}

//Зашифрування шифром «Кипарис-256» зі змінною кількістю циклів
void CypressEncryptionRound256(ui32 plaintext[8], ui32 ciphertext[8], ui32
roundKeys[10][4], int n)
{
	ui32 L[4], R[4], tmp[4];

	L[0] = plaintext[0];
	L[1] = plaintext[1];
	L[2] = plaintext[2];
	L[3] = plaintext[3];
	R[0] = plaintext[4];
	R[1] = plaintext[5];
	R[2] = plaintext[6];
	R[3] = plaintext[7];

	for(int i=0; i<n; i++)
	{
		tmp[0] = L[0];
		tmp[1] = L[1];
		tmp[2] = L[2];
		tmp[3] = L[3];
		L[0] ^= roundKeys[0][0];
		L[1] ^= roundKeys[0][1];
		L[2] ^= roundKeys[0][2];
		L[3] ^= roundKeys[0][3];
		HALFROUND32(L[0], L[1], L[2], L[3]);
		HALFROUND32(L[0], L[1], L[2], L[3]);
		L[0] ^= R[0];
		L[1] ^= R[1];
		L[2] ^= R[2];
		L[3] ^= R[3];
		R[0] = tmp[0];
		R[1] = tmp[1];
		R[2] = tmp[2];
		R[3] = tmp[3];
	}

	ciphertext[0] = L[0];
	ciphertext[1] = L[1];
	ciphertext[2] = L[2];
	ciphertext[3] = L[3];
	ciphertext[4] = R[0];
	ciphertext[5] = R[1];
	ciphertext[6] = R[2];
	ciphertext[7] = R[3];
}

#define BLOCK_NUMBER 100000

//Обчислення показників лавинного ефекту шифру «Кипарис-256»
void LavinEffect256()
{
	srand(time(NULL));
```

```
      ofstream fout;
      fout.open("LavinEffect256.txt");

      ui32 newPlaintext[8] = {0}, newCiphertext[8] = {0};

      double ExpectedValue[256] = {0}, StandartDeviation[256] = {0};

      ui32 **plaintext = new ui32* [BLOCK_NUMBER];
    for (int count=0; count<BLOCK_NUMBER; count++)
        plaintext[count] = new ui32 [8];

      ui32 **ciphertext = new ui32* [BLOCK_NUMBER];
    for (int count=0; count<BLOCK_NUMBER; count++)
        ciphertext[count] = new ui32 [8];

      double **arrayVal = new double* [BLOCK_NUMBER];
    for (int count=0; count<BLOCK_NUMBER; count++)
        arrayVal[count] = new double [256];

      int index = 0, bits = 0;

      ui32 roundKeys[10][4], masterkey[8] = {0x1245A2A1, 0xC2BB7723, 0x2331A4A3,
0xB2CCAB34, 0x2A123A30, 0x2CCAA34C, 0xA124A368, 0xC2BB7723}, ksigma[4];

      Ksigma256(masterkey, ksigma);
      MakeRoundKeys256(ksigma, masterkey, roundKeys);

      ifstream rand;
      rand.open("rnd.dat", ios::binary|ios::in);
      if(!rand.is_open())
            cout << "ERROR!" << endl;

      for(int rounds=1; rounds<=10; rounds++)
      {
            fout << "After round " << rounds << ":" << endl;

            for(int i=0; i<256; i++)
            {
                  ExpectedValue[i] = 0;
                  StandartDeviation[i] = 0;
            }

            for(int i=0; i<BLOCK_NUMBER; i++)
                  for(int j=0; j<256;j++)
                        arrayVal[i][j] = 0;

            for(int block=0; block<BLOCK_NUMBER; block++)
            {
                  for(int i=0; i<8; i++)
                  {
                        rand.read((char*)&plaintext[block][i], sizeof(ui32));
                  }

                  CypressEncryptionRound256(plaintext[block], ciphertext[block],
roundKeys, rounds);
            }

            for(int block=0; block<BLOCK_NUMBER; block++)
            {
                  index = 0;
                  for(int i=7; i>=0; i--)
                  {
                        for(int j=0; j<32; j++)
                        {
```

```
                              memcpy(newPlaintext, plaintext[block],
sizeof(newPlaintext));

                              newPlaintext[i] = newPlaintext[i] ^ (1<<j);

                              CypressEncryptionRound256(newPlaintext,
newCiphertext, roundKeys, rounds);

                              bits = 0;
                              for(int k=0; k<8; k++)
                                    for(int l=0; l<32; l++)
                                          if(((newCiphertext[k]>>l)&1) !=
((ciphertext[block][k]>>l)&1))
                                                bits++;

                              arrayVal[block][index] = bits;
                              index++;
                        }
                  }
            }
            printf("%d\n", rounds);
            for(int i=0; i<256; i++)
                  for(int j=0; j<BLOCK_NUMBER; j++)
                        ExpectedValue[i] += arrayVal[j][i];

            for(int i=0; i<256; i++)
                  ExpectedValue[i] /= BLOCK_NUMBER;

            for(int i=0; i<256; i++)
                  for(int j=0; j<BLOCK_NUMBER; j++)
                        StandartDeviation[i] += pow((arrayVal[j][i] -
ExpectedValue[i]), 2);

            for(int i=0; i<256; i++)
                  StandartDeviation[i] /= BLOCK_NUMBER;

            double minEx = 0, maxEx = 0, minDev = 0, maxDev = 0;

            for(int i=0; i<256; i++)
            {
                  if(i == 0)
                        minEx = ExpectedValue[i];

                  else if(ExpectedValue[i] < minEx)
                        minEx = ExpectedValue[i];
            }

            for(int i=0; i<256; i++)
                  if(ExpectedValue[i] > maxEx)
                        maxEx = ExpectedValue[i];

            for(int i=0; i<256; i++)
            {
                  if(i == 0)
                        minDev = StandartDeviation[i];

                  else if(StandartDeviation[i] < minDev)
                        minDev = StandartDeviation[i];
            }

            for(int i=0; i<256; i++)
                  if(StandartDeviation[i] > maxDev)
                        maxDev = StandartDeviation[i];

            fout << "Minimum of Expected Value: " << minEx << endl;
```

```
            fout << "Maximum of Expected Value: " << maxEx << endl;
            fout << "Minimum of Standart Deviation: " << minDev << endl;
            fout << "Maximum of Standart Deviation: " << maxDev << endl << endl;
      }
}

//Формування вихідної послідовності шифруючого перетворення шифру «Кипарис-256»
void FormFilesNIST256()
{
      ui32 roundKeys[10][4], masterkey[8] = {0}, ksigma[4];

      Ksigma256(masterkey, ksigma);
      MakeRoundKeys256(ksigma, masterkey, roundKeys);

      ui32 **blocks = new ui32* [1000000];
    for (int count=0; count<1000000; count++)
        blocks[count] = new ui32 [8];

      for(unsigned int i = 0; i < 1000000; i++)
      {
            for(unsigned int j = 0; j < 7; j++)
            {
                  blocks[i][j] = 0;
            }
            blocks[i][7] = i;
      }

      ofstream file;
      file.open("Cypress256.dat", ios::binary|ios::out);

      for(unsigned int ii = 0; ii < 1000000; ii++)
      {
            CypressEncryptOneBlock256(blocks[ii], roundKeys);

            for(int j=0; j<8; j++)
                  file.write((char*)&blocks[ii][j], sizeof(blocks[ii][j]));
      }
}




//Реалізація алгоритму «Кипарис-512» та дослідження його властивостей»
(cypress512.cpp)

#define ROTL64(v, n) ((v) << (n)) | ((v) >> (64 - (n)))

//Циклова функція шифру «Кипарис-512»
#define HALFROUND64(a, b, c, d) { \
      a += b; d ^= a; d = ROTL64(d, 32); \
      c += d; b ^= c; b = ROTL64(b, 24); \
      a += b; d ^= a; d = ROTL64(d, 16); \
      c += d; b ^= c; b = ROTL64(b, 15); \
}

//Вироблення проміжного ключа для шифру «Кипарис-512»
void Ksigma512(ui64 masterKey[8], ui64 ksigma[4])
{
      ui64 state[4], k0[4], k1[4];

      state[0] = 0;
      state[1] = 0;
      state[2] = 0;
      state[3] = 1;
```

```
	k0[0] = masterKey[0];
	k0[1] = masterKey[1];
	k0[2] = masterKey[2];
	k0[3] = masterKey[3];
	k1[0] = masterKey[4];
	k1[1] = masterKey[5];
	k1[2] = masterKey[6];
	k1[3] = masterKey[7];

	state[0] += k0[0];
	state[1] += k0[1];
	state[2] += k0[2];
	state[3] += k0[3];

	HALFROUND64(state[0], state[1], state[2], state[3]);
	HALFROUND64(state[0], state[1], state[2], state[3]);

	state[0] ^= k1[0];
	state[1] ^= k1[1];
	state[2] ^= k1[2];
	state[3] ^= k1[3];

	HALFROUND64(state[0], state[1], state[2], state[3]);
	HALFROUND64(state[0], state[1], state[2], state[3]);

	state[0] += k0[0];
	state[1] += k0[1];
	state[2] += k0[2];
	state[3] += k0[3];

	HALFROUND64(state[0], state[1], state[2], state[3]);
	HALFROUND64(state[0], state[1], state[2], state[3]);

	ksigma[0] = state[0];
	ksigma[1] = state[1];
	ksigma[2] = state[2];
	ksigma[3] = state[3];
}



//Формування циклових ключів шифру «Кипарис-512»
void MakeRoundKeys512(ui64 ksigma[4], ui64 masterKey[8], ui64 roundKeys[14][4])
{
	ui64 state[4], k[4], initData[8], tmp = 0;

	ui64 tmv[4] = {0x000F000F000F000F, 0x000F000F000F000F, 0x000F000F000F000F,
0x000F000F000F000F};

	initData[0] = masterKey[0];
	initData[1] = masterKey[1];
	initData[2] = masterKey[2];
	initData[3] = masterKey[3];
	initData[4] = masterKey[4];
	initData[5] = masterKey[5];
	initData[6] = masterKey[6];
	initData[7] = masterKey[7];

	for(int i=0; i<=12; i+=2)
	{
	    state[0] = initData[0];
	    state[1] = initData[1];
	    state[2] = initData[2];
	    state[3] = initData[3];
```

```
k[0] = ksigma[0];
k[1] = ksigma[1];
k[2] = ksigma[2];
k[3] = ksigma[3];

k[0] += tmv[0];
k[1] += tmv[1];
k[2] += tmv[2];
k[3] += tmv[3];

state[0] += k[0];
state[1] += k[1];
state[2] += k[2];
state[3] += k[3];

HALFROUND64(state[0], state[1], state[2], state[3]);
HALFROUND64(state[0], state[1], state[2], state[3]);

state[0] ^= k[0];
state[1] ^= k[1];
state[2] ^= k[2];
state[3] ^= k[3];

HALFROUND64(state[0], state[1], state[2], state[3]);
HALFROUND64(state[0], state[1], state[2], state[3]);

state[0] += k[0];
state[1] += k[1];
state[2] += k[2];
state[3] += k[3];

roundKeys[i][0] = state[0];
roundKeys[i][1] = state[1];
roundKeys[i][2] = state[2];
roundKeys[i][3] = state[3];

//next key

tmv[0] <<= 1;
tmv[1] <<= 1;
tmv[2] <<= 1;
tmv[3] <<= 1;

state[0] = initData[4];
state[1] = initData[5];
state[2] = initData[6];
state[3] = initData[7];

k[0] = ksigma[0];
k[1] = ksigma[1];
k[2] = ksigma[2];
k[3] = ksigma[3];

k[0] += tmv[0];
k[1] += tmv[1];
k[2] += tmv[2];
k[3] += tmv[3];

state[0] += k[0];
state[1] += k[1];
state[2] += k[2];
state[3] += k[3];

HALFROUND64(state[0], state[1], state[2], state[3]);
```

```
            HALFROUND64(state[0], state[1], state[2], state[3]);

            state[0] ^= k[0];
            state[1] ^= k[1];
            state[2] ^= k[2];
            state[3] ^= k[3];

            HALFROUND64(state[0], state[1], state[2], state[3]);
            HALFROUND64(state[0], state[1], state[2], state[3]);

            state[0] += k[0];
            state[1] += k[1];
            state[2] += k[2];
            state[3] += k[3];

            roundKeys[i+1][0] = state[0];
            roundKeys[i+1][1] = state[1];
            roundKeys[i+1][2] = state[2];
            roundKeys[i+1][3] = state[3];

            tmv[0] <<= 1;
            tmv[1] <<= 1;
            tmv[2] <<= 1;
            tmv[3] <<= 1;

            tmp = initData[0];
            initData[0] = initData[1];
            initData[1] = initData[2];
            initData[2] = initData[3];
            initData[3] = initData[4];
            initData[4] = initData[5];
            initData[5] = initData[6];
            initData[6] = initData[7];
            initData[7] = tmp;
      }
}


//Зашифрування шифром «Кипарис-512»
void CypressEncryption512(ui64 plaintext[8], ui64 ciphertext[8], ui64
roundKeys[14][4])
{
      ui64 L[4], R[4], tmp[4];

      L[0] = plaintext[0];
      L[1] = plaintext[1];
      L[2] = plaintext[2];
      L[3] = plaintext[3];
      R[0] = plaintext[4];
      R[1] = plaintext[5];
      R[2] = plaintext[6];
      R[3] = plaintext[7];

      //Наступний блок повторюється 14 разів (змінюється індекс циклового ключа)
      tmp[0] = L[0];
      tmp[1] = L[1];
      tmp[2] = L[2];
      tmp[3] = L[3];
      L[0] ^= roundKeys[0][0];
      L[1] ^= roundKeys[0][1];
      L[2] ^= roundKeys[0][2];
      L[3] ^= roundKeys[0][3];
      HALFROUND64(L[0], L[1], L[2], L[3]);
      HALFROUND64(L[0], L[1], L[2], L[3]);
      L[0] ^= R[0];
```

```
	L[1] ^= R[1];
	L[2] ^= R[2];
	L[3] ^= R[3];
	R[0] = tmp[0];
	R[1] = tmp[1];
	R[2] = tmp[2];
	R[3] = tmp[3];

	ciphertext[0] = L[0];
	ciphertext[1] = L[1];
	ciphertext[2] = L[2];
	ciphertext[3] = L[3];
	ciphertext[4] = R[0];
	ciphertext[5] = R[1];
	ciphertext[6] = R[2];
	ciphertext[7] = R[3];
}

//Зашифрування блока пам’яті шифром «Кипарис-512» для вимірювання швидкодії
void TestSpeedCypress512()
{
	ui64 roundKeys[14][4], masterkey[8] = {0x1245A2A12331A4A3,
0xB2CCAA34C2BB7723, 0x1245A2A12331A4A3, 0xB2CCAB34C2BB7723, 0x1245A2A12331A4A3,
0xB2CCAA34C2BB7723, 0x1245A2A12331A4A3, 0xB2CCAB34C2BB7723}, ksigma[4];

	Ksigma512(masterkey, ksigma);
	MakeRoundKeys512(ksigma, masterkey, roundKeys);

	for (unsigned int jj = 0; jj < number_of_reencryptions_in_memory; jj++)
		for(unsigned int ii = 0; ii < number_of_blocks_in_memory_512; ii++)
			CypressEncryptOneBlock512((unsigned long long *)
&encrypted_memory.block512[ii], roundKeys);
}


//Функції для виконання коду на C++ під ОС Android (myapp.cpp)
JNIEXPORT jfloat JNICALL
Java_com_rodinko_myapp_MainActivity_stringFromJNI(JNIEnv* env, jclass clazz)
{
	struct timeval start_ticks, finish_ticks;

	print_aes_test();
	InitMemoryEncryptionBlock128();
	DetermineTime(start_ticks);
	Test_AES_128_256_Speed_Expanded_Memory();
	DetermineTime(finish_ticks);

	return (float) CalculateEncryptionSpeedMemory(start_ticks,
finish_ticks);
}

JNIEXPORT jfloat JNICALL
Java_com_rodinko_myapp_MainActivity_stringFromJNI2(JNIEnv* env, jclass clazz)
{
	struct timeval start_ticks, finish_ticks;

	InitMemoryEncryptionBlock256();
	DetermineTime(start_ticks);
	TestSpeedCypress256();
	DetermineTime(finish_ticks);

	return (float) CalculateEncryptionSpeedMemory(start_ticks,
finish_ticks);
}
```

```
JNIEXPORT jfloat JNICALL
Java_com_rodinko_myapp_MainActivity_stringFromJNI3(JNIEnv* env, jclass clazz)
{
        struct timeval start_ticks, finish_ticks;

        InitMemoryEncryptionBlock512();
        DetermineTime(start_ticks);
        TestSpeedCypress512();
        DetermineTime(finish_ticks);

        return (float) CalculateEncryptionSpeedMemory(start_ticks,
finish_ticks);
}
```

ДОДАТОК Г. ВИХІДНИЙ КОД ПРОГРАМИ, ЩО МІСТИТЬ РЕАЛІЗАЦІЮ МЕТОДІВ ПОШУКУ ДИФЕРЕНЦІЙНИХ ХАРАКТЕРИСТИК БЛОКОВОГО ШИФРУ «КИПАРИС-256»

```
//diff.cpp
#define _CRT_RAND_S
#include <iostream>
#include <fstream>
#include <math.h>
#include <iterator>
#include <vector>
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include <limits.h>

#include "operations.h"

#define ROTL32(v, n) ((v) << (n)) | ((v) >> (32 - (n)))
#define ROTR32(v, n) ((v) >> (n)) | ((v) << (32 - (n)))

using namespace std;

struct out_diff{
	unsigned int output_def;
	double p;
};

struct oneRoundDiff {
	unsigned int in[4];
	unsigned int out[4];
	double prob;
};

struct fulloneRoundDiff {
	unsigned int in[8];
	unsigned int out[8];
	double prob;
};

void getProbableDiffs(unsigned int alpha, unsigned int beta, vector <out_diff>
&new_diff)
{
	struct out_diff tmp;
	vector<unsigned int> v;
	errno_t e;
	unsigned int number, res;

	double p = max_xdp_add_lm(alpha, beta, v);

	if (v.size() > 5)
	{
		for (int i = 0; i < 5; i++)
		{
			e = rand_s(&number);
			res = (unsigned int)((double)number / ((double)UINT_MAX + 1) *
v.size());
			tmp.output_def = v[res];
```

```
                tmp.p = p_xdp_add_lm(alpha, beta, v[res]);
                new_diff.push_back(tmp);
                v.erase(v.begin() + res);
          }
          v.clear();
    }
    else {
          for (int i = 0; i < v.size(); i++)
          {
                tmp.output_def = v[i];
                tmp.p = p_xdp_add_lm(alpha, beta, v[i]);
                new_diff.push_back(tmp);
          }
          v.clear();
    }
}

const double thres_round = 0.0000000000000001; // тут задається поріг
ймовірності
const double thres_round_two = 0.0000000000000002;// тут задається поріг
ймовірності

FILE* fm;
errno_t err = fopen_s(&fm, "result.txt", "w");
FILE* daf;
errno_t er = fopen_s(&daf, "result_2.txt", "w");

bool f1(unsigned int def[4], unsigned int def2[4], unsigned int out_diff, double
p1)
{
    //fprintf(fm, "0. %x %x %x %x\n", def[0], def[1], def[2], def[3]);
    def2[0] = out_diff;
    def2[1] = def[1];
    def2[2] = def[2];
    def2[3] = def[3];

    def2[3] = ROTL32(def2[3] ^ def2[0], 16);

    return true;
}

bool f2(unsigned int def[4], unsigned int def2[4], unsigned int out_diff, double
p1, double p2)
{
    if( (p1 - thres_round) < 0)
    {
          return false;
    }

    def2[0] = def[0];
    def2[1] = def[1];
    def2[2] = out_diff;
    def2[3] = def[3];

    def2[1] = ROTL32(def2[1] ^ def2[2], 12);

    return true;
}

bool f3(unsigned int def[4], unsigned int def2[4], unsigned int out_diff, double
p1, double p2)
{
    if( (p1 - thres_round) < 0)
    {
          return false;
```

```
	}

	def2[0] = out_diff;
	def2[1] = def[1];
	def2[2] = def[2];
	def2[3] = def[3];

	def2[3] = ROTL32(def2[3] ^ def2[0], 8);

	return true;
}

bool f4(unsigned int def[4], unsigned int def2[4], unsigned int out_diff, double
p1, double p2)
{
	if( (p1 - thres_round) < 0)
	{
		return false;
	}

	def2[0] = def[0];
	def2[1] = def[1];
	def2[2] = out_diff;
	def2[3] = def[3];

	def2[1] = ROTL32(def2[1] ^ def2[2], 7);

	return true;
}

bool f5(unsigned int def[4], unsigned int def2[4], unsigned int out_diff, double
p1, double p2)
{
	if( (p1 - thres_round) < 0)
	{
		return false;
	}

	def2[0] = out_diff;
	def2[1] = def[1];
	def2[2] = def[2];
	def2[3] = def[3];

	def2[3] = ROTL32(def2[3] ^ def2[0], 16);

	return true;
}

bool f6(unsigned int def[4], unsigned int def2[4], unsigned int out_diff, double
p1, double p2)
{
	if( (p1 - thres_round) < 0)
	{
		return false;
	}

	def2[0] = def[0];
	def2[1] = def[1];
	def2[2] = out_diff;
	def2[3] = def[3];

	def2[1] = ROTL32(def2[1] ^ def2[2], 12);

	return true;
}
```

```
bool f7(unsigned int def[4], unsigned int def2[4], unsigned int out_diff, double
p1, double p2)
{
	if( (p1 - thres_round) < 0)
	{
		return false;
	}

	def2[0] = out_diff;
	def2[1] = def[1];
	def2[2] = def[2];
	def2[3] = def[3];

	def2[3] = ROTL32(def2[3] ^ def2[0], 8);

	return true;
}

bool f8(unsigned int def[4], unsigned int def2[4], unsigned int out_diff, double
p1, double p2)
{
	if( (p1 - thres_round) < 0)
	{
		return false;
	}

	def2[0] = def[0];
	def2[1] = def[1];
	def2[2] = out_diff;
	def2[3] = def[3];

	def2[1] = ROTL32(def2[1] ^ def2[2], 7);

	return true;
}

int newSearch(unsigned int def[4], vector <oneRoundDiff> &structDiff, vector
<out_diff> &diff_res)
{
	double p[8] = { 1.0 }, p_tmp = 0.0;
	unsigned int def2[4] = {0}, def3[4] = {0}, def4[4] = {0}, def5[4] = {0},
		def6[4] = {0}, def7[4] = {0}, def8[4] = {0}, def9[4] = {0},
orig[4] = {0}, a = 0, sum = 0;

	//vector <out_diff> diff_res;
	vector <out_diff> diff, diff2, diff3, diff4, diff5, diff6, diff7, diff8,
diff9;
	oneRoundDiff str;
	//str.prob = 0.0;
	int carry = 0;

	orig[0] = def[0];
	orig[1] = def[1];
	orig[2] = def[2];
	orig[3] = def[3];

	struct out_diff tmp;
	tmp.output_def = 0;
	tmp.p = 0.0;
	diff_res.push_back(tmp);
	diff_res.push_back(tmp);
	diff_res.push_back(tmp);
	diff_res.push_back(tmp);
	diff_res.push_back(tmp);
	diff_res.push_back(tmp);
```

```
	diff_res.push_back(tmp);
	diff_res.push_back(tmp);

	unsigned int first_in[8] = { 0 }, second_in[8] = { 0 };
	diff.clear();
	getProbableDiffs(def[0], def[1], diff);
	for(int i = 0; i < diff.size(); i++)
	{
		first_in[0] = def[0];
		second_in[0] = def[1];
		diff_res[0].output_def = diff[i].output_def;
		diff_res[0].p = diff[i].p;
		p[0] = diff[i].p;
		f1(def, def2, diff[i].output_def, p[0]);
		diff2.clear();
		getProbableDiffs(def2[2], def2[3], diff2);
		for(int j = 0; j < diff2.size(); j++)
		{
			first_in[1] = def2[2];
			second_in[1] = def2[3];
			diff_res[1].output_def = diff2[j].output_def;
			diff_res[1].p = diff2[j].p;
			p[1] = p[0] * diff2[j].p;
			if( f2(def2, def3, diff2[j].output_def, p[1], p[0]) == false )
				continue;
			diff3.clear();
			getProbableDiffs(def3[0], def3[1], diff3);
			for(int k = 0; k < diff3.size(); k++)
			{
				first_in[2] = def3[0];
				second_in[2] = def3[1];
				diff_res[2].output_def = diff3[k].output_def;
				diff_res[2].p = diff3[k].p;
				p[2] = p[1] * diff3[k].p;
				if( f3(def3, def4, diff3[k].output_def, p[2], p[1]) ==
false )
					continue;
				diff4.clear();
				getProbableDiffs(def4[2], def4[3], diff4);
				for(int l = 0; l < diff4.size(); l++)
				{
					first_in[3] = def4[2];
					second_in[3] = def4[3];
					diff_res[3].output_def = diff4[l].output_def;
					diff_res[3].p = diff4[l].p;
					p[3] = p[2] * diff4[l].p;
					if( f4(def4, def5, diff4[l].output_def, p[3],
p[2]) == false )
						continue;
					diff5.clear();
					getProbableDiffs(def5[0], def5[1], diff5);
					for(int m = 0; m < diff5.size(); m++)
					{
						first_in[4] = def5[0];
						second_in[4] = def5[1];
						diff_res[4].output_def =
diff5[m].output_def;
						diff_res[4].p = diff5[m].p;
						p[4] = p[3] * diff5[m].p;
						if( f5(def5, def6, diff5[m].output_def,
p[4], p[3]) == false )
							continue;
						diff6.clear();
						getProbableDiffs(def6[2], def6[3], diff6);
						for(int n = 0; n < diff6.size(); n++)
```

```
                                    {
                                          first_in[5] = def6[2];
                                          second_in[5] = def6[3];
                                          diff_res[5].output_def =
diff6[n].output_def;
                                          diff_res[5].p = diff6[n].p;
                                          p[5] = p[4] * diff6[n].p;
                                          if( f6(def6, def7,
diff6[n].output_def, p[5], p[4]) == false )
                                                continue;
                                          diff7.clear();
                                          getProbableDiffs(def7[0], def7[1],
diff7);
                                          for(int o = 0; o < diff7.size(); o++)
                                          {
                                                first_in[6] = def7[0];
                                                second_in[6] = def7[1];
                                                diff_res[6].output_def =
diff7[o].output_def;
                                                diff_res[6].p = diff7[o].p;
                                                p[6] = p[5] * diff7[o].p;
                                                if( f7(def7, def8,
diff7[o].output_def, p[6], p[5]) == false )
                                                      continue;
                                                diff8.clear();
                                                getProbableDiffs(def8[2],
def8[3], diff8);
                                                for(int q = 0; q < diff8.size();
q++)
                                                {
                                                      first_in[7] = def8[2];
                                                      second_in[7] = def8[3];
                                                      diff_res[7].output_def =
diff8[q].output_def;
                                                      diff_res[7].p = diff8[q].p;
                                                      //fprintf(fm, "%d\t%x\n",
diff8.size(), diff8[q].output_def);
                                                      p[7] = p[6] * diff8[q].p;
                                                      if( f8(def8, def9,
diff8[q].output_def, p[7], p[6]) == false )
                                                            continue;

                                                      str.in[0] = orig[0];
                                                      str.in[1] = orig[1];
                                                      str.in[2] = orig[2];
                                                      str.in[3] = orig[3];

                                                      str.out[0] = def9[0];
                                                      str.out[1] = def9[1];
                                                      str.out[2] = def9[2];
                                                      str.out[3] = def9[3];

                                                      str.prob = p[7];

                                                      structDiff.push_back(str);

                                                      fprintf(fm, "\nIn. %x %x %x
%x\n", orig[0], orig[1], orig[2], orig[3]);
                                                      fflush(fm);
                                                      fprintf(fm, "Out. %x %x %x
%x \t%f\n", def9[0], def9[1], def9[2], def9[3], (log(p[7])/log(2.0)));
                                                      fflush(fm);
                                                      for (int cc = 0; cc <
diff_res.size(); cc++)
                                                      {
```

```
                                                                  fprintf(fm, "(%x, %x)
-> %x, %f\n", first_in[cc], second_in[cc], diff_res[cc].output_def,
diff_res[cc].p);
                                                            }
                                                            fflush(fm);
                                                      }
                                                }
                                          }
                                    }
                              }
                        }
                  }
            }

      return 0;
}


int multiRound(unsigned int defl[4], unsigned int defr[4], int round) {
      vector <oneRoundDiff> diff;
      vector <out_diff> diffm;

      unsigned int tmp[4] = { 0 }, number, res;
      unsigned int mas[4] = { 0 }, mas2[4] = { 0 };
      double max = 0.0;
      errno_t e;

      for (int i = 0; i < 4; i++)
      {
            mas[i] = defl[i];
            mas2[i] = defr[i];
      }

      tmp[0] = defl[0];
      tmp[1] = defl[1];
      tmp[2] = defl[2];
      tmp[3] = defl[3];

      newSearch(defl, diff, diffm);

      if (diff.size() == 0 )
      {
            return 0;
      }

      else {
            for (int i = 0; i < diff.size(); i++)
            {
                  if (diff.at(i).prob - max > 0)
                  {
                        max = diff.at(i).prob;
                  }
            }

            vector <oneRoundDiff> newStr;
            for (int i = 0; i < diff.size(); i++)
            {
                  //printf("Out: %x %x %x %x ", diff.at(i).out[0],
diff.at(i).out[1], diff.at(i).out[2], diff.at(i).out[3]);
                  //printf("\n");
                  if (abs(diff.at(i).prob - max) <
0.00000000000000000000000000000001)
                        newStr.push_back(diff.at(i));
            }
```

```
            if (newStr.size() > 10)
            {
                  for (int i = 0; i < 10; i++)
                  {
                        e = rand_s(&number);
                        res = (unsigned int)((double)number / ((double)UINT_MAX
+ 1) * newStr.size());

                        defr[0] = mas2[0];
                        defr[1] = mas2[1];
                        defr[2] = mas2[2];
                        defr[3] = mas2[3];

                        defr[0] = defr[0] ^ newStr.at(res).out[0];
                        defr[1] = defr[1] ^ newStr.at(res).out[1];
                        defr[2] = defr[2] ^ newStr.at(res).out[2];
                        defr[3] = defr[3] ^ newStr.at(res).out[3];

                        defl[0] = defr[0];
                        defl[1] = defr[1];
                        defl[2] = defr[2];
                        defl[3] = defr[3];

                        defr[0] = tmp[0];
                        defr[1] = tmp[1];
                        defr[2] = tmp[2];
                        defr[3] = tmp[3];

                        fprintf(daf, "Input: %x %x %x %x ", mas[0], mas[1],
mas[2], mas[3]);
                        fprintf(daf, "%x %x %x %x\n", mas2[0], mas2[1], mas2[2],
mas2[3]);
                        fprintf(daf, "Output: %x %x %x %x ", defl[0], defl[1],
defl[2], defl[3]);
                        fprintf(daf, "%x %x %x %x\n", defr[0], defr[1], defr[2],
defr[3]);
                        fprintf(daf, "%f\n", (log(newStr.at(res).prob) /
log(2.0)));
                  }
            }
            else {
                  fprintf(daf, "%d\n", newStr.size());
                  for (int i = 0; i < newStr.size(); i++)
                  {
                        defr[0] = mas2[0];
                        defr[1] = mas2[1];
                        defr[2] = mas2[2];
                        defr[3] = mas2[3];

                        defr[0] = defr[0] ^ newStr.at(i).out[0];
                        defr[1] = defr[1] ^ newStr.at(i).out[1];
                        defr[2] = defr[2] ^ newStr.at(i).out[2];
                        defr[3] = defr[3] ^ newStr.at(i).out[3];

                        defl[0] = defr[0];
                        defl[1] = defr[1];
                        defl[2] = defr[2];
                        defl[3] = defr[3];

                        defr[0] = tmp[0];
                        defr[1] = tmp[1];
                        defr[2] = tmp[2];
                        defr[3] = tmp[3];
```

```
                        fprintf(daf, "Input: %x %x %x %x ", mas[0], mas[1],
mas[2], mas[3]);
                        fprintf(daf, "%x %x %x %x\n", mas2[0], mas2[1], mas2[2],
mas2[3]);
                        fprintf(daf, "Output: %x %x %x %x ", defl[0], defl[1],
defl[2], defl[3]);
                        fprintf(daf, "%x %x %x %x\n", defr[0], defr[1], defr[2],
defr[3]);
                        fprintf(daf, "%f\n", (log(newStr.at(i).prob) /
log(2.0)));
                }
            }
      }

      return 0;
}

//operations.cpp
unsigned int aop(unsigned int x, int n)
{
      unsigned int logn = log((double)n) / log((double)2);
      unsigned int *xmas = new unsigned int[logn];
      unsigned int *ymas = new unsigned int[logn];

      xmas[0] = x & (x >> 1);
      for (int i = 1; i < logn - 1; i++)
            xmas[i] = xmas[i - 1] & (xmas[i - 1] >> (1 << i));

      ymas[0] = x & (~xmas[0]);
      for (int i = 1; i < logn; i++)
            ymas[i] = ymas[i - 1] | ((ymas[i - 1] >> (1 << i)) & xmas[i - 1]);

      unsigned int tmp = ymas[logn - 1];
      delete[] xmas;
      delete[] ymas;

      return tmp;
}

unsigned int C(unsigned int x, unsigned int y, unsigned int n)
{
      unsigned int a = ~(x ^ y);
      if (((a >> (n - 1)) & 1))
            a &= ~(1 << (n - 1));

      unsigned int b = a & (a >> 1) & (x ^ (x >> 1));
      return aop(b, n);
}

uint32_t hw32(const uint32_t w)
{
      uint32_t res = w - ((w >> 1) & 0x55555555);
      res = (res & 0x33333333) + ((res >> 2) & 0x33333333);
      res = (res + (res >> 4)) & 0x0F0F0F0F;
      res = res + (res >> 8);
      return (res + (res >> 16)) & 0x000000FF;
}

bool is_eq(const unsigned int x, const unsigned int y, const unsigned int z)
{
      return ((x == y) && (x == z));
}

unsigned int eq(unsigned int x, unsigned int y, unsigned int z)
{
```

```
	unsigned int e = (~x ^ y) & (~x ^ z);
	return e;
}

unsigned int mask(unsigned int n)
{
	return pow((double)2, (double)(n - 1)) - 1;
}

double p_xdp_add_lm(unsigned int da, unsigned int db, unsigned int dc)
{
	if ((eq(da << 1, db << 1, dc << 1) & (da ^ db ^ dc ^ (da << 1))) != 0)
		return 0;
	return 1 / pow(2, (hw32(~eq(da, db, dc) & mask(32))));
}

double max_xdp_add_lm(unsigned int da, unsigned int db, std::vector<unsigned
int> &dc_ret)
{
	//srand(time(NULL));
	unsigned int n = 32, number = 0, res;
	double p_max = 0.0;
	unsigned int dc = 0;
	errno_t err;

	std::vector<unsigned int> v;
	std::vector<unsigned int> v_new;

	dc |= (da & 1) ^ (db & 1);
	v.push_back(dc);

	unsigned int Cc = C(da, db, n);

	for (uint32_t i = 1; i < n; i++) {
		uint32_t C_this = (Cc >> i) & 1;
		uint32_t da_prev = (da >> (i - 1)) & 1;
		uint32_t db_prev = (db >> (i - 1)) & 1;

		uint32_t da_this = (da >> i) & 1;
		uint32_t db_this = (db >> i) & 1;
		uint32_t dc_this = 0;
		for (uint32_t j = 0; j < v.size(); j++) {
			uint32_t dc_prev = (v[j] >> (i - 1)) & 1;
			if (is_eq(da_prev, db_prev, dc_prev)) {
				dc_this = (da_this ^ db_this ^ da_prev);
				v_new.push_back(v[j] | (dc_this << i));
			}
			else {
				if ((i == (n - 1)) || (da_this != db_this) || (C_this ==
1)) {
					dc_this = 0;
					v_new.push_back(v[j] | (dc_this << i));

					dc_this = 1;
					v_new.push_back(v[j] | (dc_this << i));
				}
				else {
					dc_this = da_this;
					v_new.push_back(v[j] | (dc_this << i));
				}
			}
		}
		//CHECK v_new size
		//v.clear();
		unsigned int s = v_new.size();
```

```
            double dd = (double)s / 2.0;
            if (s > 50000ULL)
            {
                  for (int k = 0; k < dd; k++)
                  {
                        err = rand_s(&number);
                        res = (unsigned int)((double)number / ((double)UINT_MAX
+ 1) * v_new.size());
                        v_new.erase(v_new.begin() + res);
                  }
            }
            v = v_new;
            v_new.clear();
      }

      dc_ret = v;
      p_max = p_xdp_add_lm(da, db, v[0]);

      return p_max;
}
```

ДОДАТОК Д. АКТИ ВПРОВАДЖЕННЯ

"ЗАТВЕРДЖУЮ"

Проректор з наукової роботи
Харківського національного університету
імені В.Н. Каразіна
проф. _____ В.О. Катрич
« 10 » 09 2020 р.

АКТ
впровадження результатів дисертаційної роботи
на здобуття ступеня доктора філософії
Родінко Марії Юріївни
у наукову роботу кафедри безпеки інформаційних систем і технологій
Харківського національного університету імені В.Н. Каразіна

Комісія у складі голови комісії, доктора технічних наук, доцента Рассомахіна С.Г., та членів комісії, доктора технічних наук, професора Горбенко І.Д., доктора технічних наук, професора Кузнецова О.О. встановила, що у Харківському національному університеті імені В.Н. Каразіна впроваджені результати дисертаційних досліджень, що одержані Родінко Марією Юріївною, при виконанні наступних науково-дослідних робіт:

1. № 1-41-16 «Аналіз стану, обгрунтування вимог та напрямів розвитку, стандартизація, розробка та впровадження криптографічних систем для надання електронних довірчих послуг» (№ ДР 0116U000810) в частині розробки удосконаленого методу генерації оптимальних S-блоків.
2. № 1-41-18 «Аналіз, дослідження, розробка та стандартизація криптографічних систем для захисту інформації в пост-квантовому середовищі, в умовах інформаційних і гібридних війн» (№ ДР 0118U002024) в частині розробки методів дослідження властивостей шифрів.

Голова комісії, д.т.н., доцент С.Г. Рассомахін

Члени комісії:
д.т.н., професор І.Д. Горбенко

д.т.н. професор О.О. Кузнецов

"ЗАТВЕРДЖУЮ"
Проректор з науково-педагогічної роботи
Харківського національного університету
імені В.Н. Каразіна
Академік НАН України
проф. _____ М.О. Азарєнков

«09» 09 2020 р.

АКТ
впровадження результатів дисертаційної роботи
на здобуття ступеня доктора філософії
Родінко Марії Юріївни
у навчальний процес Харківського національного університету імені В.Н. Каразіна

Комісія у складі голови комісії, доктора технічних наук, доцента Рассомахіна С.Г., та членів комісії, доктора технічних наук, професора Горбенко І.Д., доктора технічних наук, професора Кузнецова О.О. встановила, що у Харківському національному університеті імені В.Н. Каразіна впроваджені наступні результати, що одержані Родінко Марією Юріївною в процесі виконання дисертаційних досліджень.

1. По дисципліні "Прикладна криптологія" для спеціальності "Кібербезпека" при підготовці та читанні лекцій за темами 8-го розділу "Криптографічний аналіз симетричних криптосистем", зокрема використані результати по розробці методів диференційного криптоаналізу ARX-шифрів та оцінки стійкості перспективного малоресурсного блокового шифру «Кипарис».

2. Розробки по дослідженню диференційних властивостей ARX-шифрів використані при виконанні подальших досліджень у магістерських атестаційних роботах.

Голова комісії, д.т.н., доцент С.Г. Рассомахін

Члени комісії:
д.т.н., професор І.Д. Горбенко

д.т.н., професор О.О. Кузнецов

"ЗАТВЕРДЖУЮ"

Виконавчий директор

АТ «ІІТ»

В.Д. Кравченко

«__» ____ 2020 р.

АКТ
впровадження результатів дисертаційної роботи
на здобуття ступеня доктора філософії
Родінко Марії Юріївни
у Приватному акціонерному товаристві «Інститут інформаційних технологій»

Комісія у складі голови комісії, члена Наглядової ради, першого заступника головного конструктора АТ "ІІТ", кандидата технічних наук Горбенко Ю.І. та членів комісії, начальника відділу АЗЗІ АТ "ІІТ" Бобуха В.А., технічного директора АТ "ІІТ" Шумова О.І. встановила, що у Приватному акціонерному товаристві «Інститут інформаційних технологій» впроваджені наступні результати, що одержані Родінко Марією Юріївною в процесі виконання дисертаційних досліджень.

1. Удосконалений метод градієнтного спуску для генерації S-блоків, що дозволив зменшити середній час генерації оптимального байтового S-блоку на персональному комп'ютері з 2,5 годин до 30 хвилин.
2. Математичний метод оцінки колізійних властивостей неін'єктивних схем розгортання ключів блокових шифрів, що дозволив отримати вдосконалене значення оцінки стійкості діючого національного стандарту ДСТУ 7624:2014.
3. Перспективний малоресурсний блоковий шифр «Кипарис», що забезпечує високий та надвисокий рівні стійкості та від 2,5 разів перевершує за швидкодією блоковий шифр AES на 32- та 64-бітових процесорах загального призначення та мобільних платформах.

Голова комісії, член Наглядової ради,
перший заступник головного
конструктора АТ "ІІТ, к.т.н. Ю.І. Горбенко

Члени комісії:
начальник відділу АЗЗІ АТ "ІІТ" В.А. Бобух

технічний директор АТ "ІІТ" О.І. Шумов