НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичних методів захисту інформації

«На правах рукопису»
УДК (004.056.55)

«До захисту допущено»
Завідувач кафедри
_____ Сергій ЯКОВЛЄВ
«__» _____ 2026 р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Математичні методи криптографічного захисту інформації»

зі спеціальності: 113 Прикладна математика
на тему: **«Аналіз та дослідження практичних криптосистем на основі arx-конструкцій»**

Виконав:
студент IV курсу, групи ФІ-23
Баранов Глєб Сергійович _____

Керівник:
к. т. н., доцент кафедри ММЗІ
Яковлєв Сергій Володимирович _____

Рецензент:
посада, степінь, звання
Прізвище Ім'я По-батькові _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ — 2026

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичних методів захисту інформації

Рівень вищої освіти — перший (бакалаврський)
Спеціальність — 113 Прикладна математика,
ОПП «Математичні методи криптографічного захисту інформації»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Сергій ЯКОВЛЄВ
«__» _____ 2026 р.

ЗАВДАННЯ
на дипломну роботу

Студент: Баранов Глєб Сергійович

1. Тема роботи: «*Порівняльний аналіз ARX-перестановок та блокових шифри на основі ARX*
як сучасних ARX-примітивів», науковий керівник дисертації: к. т. н., доцент кафедри ММЗІ Яковлєв Сергій Володимирович,

затверджені наказом по університету №__ від «__» _____ 2026 р.

2. Термін подання студентом роботи: «__» _____ 2026 р.

3. Об'єкт дослідження: симетричні криптографічні примітиви ARX-типу — блокові шифри та перестановки.

4. Предмет дослідження: архітектурні принципи побудови ARX-примітивів, їх структурна класифікація та характеристики.

5. Перелік завдань:

а. Формалізувати базові поняття та означення ARX-дизайну.

б. Систематизувати і описати ARX-перестановки за сімействами і структурними ознаками

в. Провести порівяльний аналіз ARX-перестановок та систем в які вони вбудовані

г. Описати і систематизувати блокові ARX-шифри.

д. Методично проаналізувати характеристики ARX-перестановок і блокових шифрів.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: Презентація доповіді

7. Орієнтовний перелік публікацій: планується доповідь на всеукраїнській конференції

8. Дата видачі завдання: 10 вересня 2025 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником	01-15 вересня 2025 р.	Виконано
2	Огляд опублікованих джерел за тематикою дослідження	Вересень-жовтень 2025 р.	Виконано
3	Аналіз конструкцій ARX-перестановок та їх криптографічних властивостей	Жовтень-січень 2025 р.	Виконано
4	Дослідження блокових шифрів на основі ARX — порівняння раундових функцій, ключових розкладів, швидкості	Січень-Лютий 2026 р.	Виконано
5	Розробка критеріїв порівняльного аналізу	Березень 01-15 2026 р.	Виконано
6	Проведення математичних обрахухвань та програмна реалізація задля отримання відсутніх даних зопублікованих джерел	Березень 15-30 2026 р.	Виконано
7	Оформлення дипломної роботи	Травень 2026 р.	Виконано

Студент _____ Глєб БАРАНОВ

Керівник _____ Сергій ЯКОВЛЄВ

РЕФЕРАТ

Кваліфікаційна робота містить: 76 стор., 17 рисунки, 3 таблиць, 34 джерел.

У межах даної роботи основна увага приділяється огляду та систематизації існуючих криптосистем ARX-типу, аналізу їх архітектурних рішень, принципів побудови та сфер застосування.

Метою даного дослідження є аналіз та систематизація практичних криптосистем, побудованих на основі ARX-конструкцій, з оцінкою їхньої криптографічної стійкості та ефективності. Об‘єктом даного дослідження є симетричні криптографічні примітиви ARX-типу: блокові шифри і перестановки.

Результати дослідження:

– Формалізувано базові поняття та означення ARX-дизайну.

– Систематизувано і описати ARX-перестановки за сімействами і структурними ознаками

– Проведено порівяльний аналіз ARX-перестановок та систем в які вони вбудовані

– Описано і систематизувати блокові ARX-шифри.

– Методично проаналізувано характеристики ARX-перестановок і блокових шифрів.

ARX-КОНСТРУКЦІЇ, СИМЕТРИЧНА КРИПТОГРАФІЯ, АНАЛІЗ КРИПТОСИСТЕМ, СТАТИСТИЧНИЙ АНАЛІЗ

ABSTRACT

The qualification work contains: 76 pages, 17 figures, 3 tables, 34 references.

This work focuses on the review and systematization of existing ARX-type cryptosystems, the analysis of their architectural solutions, design principles, and application domains.

The research objective is the analysis and systematization of practical cryptosystems based on ARX constructions, with an assessment of their cryptographic strength and efficiency. The object of research is symmetric cryptographic primitives of ARX-type: block ciphers and permutations.

Research results:

– the basic concepts and definitions of ARX design have been formalized;

– ARX permutations have been systematized and described by families and structural features;

– a comparative analysis of ARX permutations and the systems in which they are embedded has been carried out;

– ARX block ciphers have been described and systematized;

– the characteristics of ARX permutations and block ciphers have been methodically analyzed.

ARX CONSTRUCTIONS, SYMMETRIC CRYPTOGRAPHY, CRYPTOSYSTEM ANALYSIS, STATISTICAL ANALYSIS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

V_n — множина всіх бітових векторів довжини n.

$x = (x_{n-1}, \ldots, x_1, x_0)$, $y = (y_{n-1}, \ldots, y_1, y_0)$ — n-бітові двійкові вектори, $x_i, y_j \in \{0,1\}$.

$x \lll r$ — циклічний зсув вектора x на r позицій ліворуч.

$x \ggg r$ — циклічний зсув вектора x на r позицій праворуч.

$x + y$ — додавання двох n-бітових слів за модулем 2^n.

$x - y$ — віднімання двох n-бітових слів за модулем 2^n.

$x \vee y$ — операція логічного АБО.

$x \wedge y$ — операція логічного ТА.

$\oplus$ — операція додавання за модулем 2 (XOR).

$\leftarrow$ — операція присвоєння.

$\mathrm{DP}(\alpha \rightarrow \beta)$ — диференціальна ймовірність переходу різниці α у різницю β.

$xdp^+(\alpha, \beta \rightarrow \gamma)$ — диференціальна ймовірність проходження різниць через додавання за модулем 2^n.

ε — зміщення лінійної апроксимації; $c = 2\varepsilon$ — відповідна кореляція.

rp^+ — обертальна ймовірність для операції додавання за модулем 2^n.

$\mathrm{Adv}(\mathcal{D})$ — перевага розрізнювача (дистингвішера) $\mathcal{D}$.

ВСТУП

ARX-криптосистеми будуються виключно на простих операціях, що безпосередньо підтримуються інструкційним набором сучасних обчислювальних процесорів. До таких операцій належать додавання за модулем, побітове додавання, циклічні зсуви. Такий підхід забезпечує простоту програмної реалізації та високу швидкодію, що зумовило широке використання ARX-конструкцій у сучасних криптографічних алгоритмах.

Завдяки низьким вимогам до обчислювальних ресурсів і відсутності складних табличних перетворень ARX-криптосистеми стали важливою складовою напряму «легкої криптографії» (lightweight cryptography), орієнтованого на створення надійних симетричних алгоритмів для середовищ з обмеженими ресурсами, зокрема вбудованих систем, мобільних пристроїв та Інтернету речей(IoT). У цьому контексті ARX-підхід розглядається як один із базових принципів побудови ефективних і портативних криптографічних рішень.

Специфіка алгебраїчної структури ARX-систем визначає особливості їх проєктування та аналізу, а також впливає на вибір параметрів і раундових функцій у практичних реалізаціях.

1 АЛГОРИТМИ З ARX-ДИЗАЙНОМ

У цьому розділі будуть розглянуті принципи побудови ARX-криптосистем. Розглянуті алгоритми охоплюють блокові та потокові шифри,криптографічні геш-функції, губчасті конструкції та великі криптографічні перестановки, в яких ARX-підхід є ключовим елементом проєктування. Окрім цього, розглянуто основні методи криптоаналізу, розроблені спеціально для ARX-конструкцій.

Метою розділу є систематизація ARX-примітивів на дві окремі окреслювальні групи, задля подальшого порівняльного аналізу.

1.1 Загальна модель ARX-дизайну

ARX-дизайн базується на трьох основних операціях:

– **A**ddition (додавання за модулем 2^n);

– **R**otation (циклічний зсув);

– **X**OR (побітове виключне АБО).

Вказані операції виконуються над словами фіксованої розрядності n та комбінуються у раундових перетвореннях з метою досягнення нелінійності, дифузії та повної залежності вихідного стану від кожного біта вхідних даних.

Додавання за модулем 2^n є ключовим джерелом нелінійності в ARX-дизайнах, оскільки ця операція не допускає лінійного подання над полем $GF(2)$. Циклічні зсуви забезпечують перерозподіл бітових впливів у межах машинного слова, тоді як операція XOR використовується для ефективного лінійного змішування інформації.

Типова ARX-конструкція реалізується у вигляді послідовності раундів, у кожному з яких застосовується заздалегідь визначений або параметризований набір операцій додавання, циклічних зсувів та XOR.

Кількість раундів і структура їх поєднання безпосередньо впливають на швидкість наростання дифузії та загальні криптографічні властивості алгоритму.

Однією з визначальних переваг ARX-дизайну є його орієнтація на програмну реалізацію. Усі базові операції напряму підтримуються сучасними процесорними архітектурами та не потребують використання S-блоків, що забезпечує високу портативність і ефективність ARX-криптосистем на різних апаратних платформах.

1.2 ARX-примітиви

Оскільки центральним об'єктом подальшого розгляду є перестановки, та блочні шифри, побудовані з ARX-операцій, спершу формалізуємо базові поняття.

Означення 1.1 (Перестановка)**.** Нехай $n \in \mathbb{N}$. *Перестановкою* множини V_n називається бієктивне відображення

$$\pi \colon V_n \to V_n.$$

У криптографічному контексті інтерес становлять перестановки, реалізовані композицією простих оборотних операцій.

Означення 1.2 (ARX-перестановка)**.** Нехай $n \in \mathbb{N}$. **ARX-перестановкою** називається відображення

$$\pi \colon V_n \to V_n,$$

яке є бієкцією та може бути подане як скінченна композиція елементарних відображень, побудованих виключно з таких операцій:

– побітове додавання за модулем 2 (XOR):

$$x \mapsto x \oplus y, \quad y \in V_n;$$

– додавання за модулем 2^n:

$$x \mapsto x + y \pmod{2^n}, \quad y \in V_n;$$

– циклічний зсув:

$$x \mapsto x \lll r \quad \text{або} \quad x \mapsto x \ggg r, \quad r \in \{0,1,\ldots,n-1\}.$$

Бієктивність відображення π гарантується оборотністю кожної з перелічених операцій.

ARX-перестановки є ключовим компонентом багатьох сучасних криптографічних примітивів, включаючи геш-функції, AEAD-схеми та блокових і потокових шифрів. Детальна систематизація конкретних ARX-перестановок за сімействами є предметом розділу 2.

Було визначенно що для порівняльного аналізу доцільно розглядати блокові шифри як окремий сегмент ARX-перестановок, в зв'язку з тим, що коли всі інші криптографічні системи(такі як *геш-функції*, *губчасті конструкції*, *потокові шифри*, *MAC-алгоритми*) використовують ARX-перестановки як ядро своїх алгоритмів, то блокові шифри є формально сімейством ARX-перестановок, які параметризуються секретним ключем, і мають власну структуру, яка відрізняє їх від інших ARX-примітивів.

Задамо додатково визначення блокових шифрів які базуються на ARX.

Означення 1.3 (Блоковий шифр з ARX-структурою)**.** *Блоковим шифром з ARX-структурою* називається сім'я ARX-перестановок (див. 1.2)

$$\pi \colon V_k \times V_n \to V_n,$$

Де параметр k — довжина ключа, n — розмір блоку.

Де для $\forall m \in \mathcal{P}, c \in \mathcal{C}$,$K \in \mathcal{K}$, $E \in \mathcal{E}$,$D \in \mathcal{D}$, шифрування задається

як

$$E_K : \mathcal{P} \to \mathcal{C}$$

, а дешифрування — обернена функція

$$\mathrm{D}_K(\mathrm{E}_K(m)) = m \qquad \forall\, K \in V_k,\, m \in V_n.$$

.

1.3 Критерії порівняльної оцінки ARX-конструкцій

Після того як ми задали формальні визначення ARX-примітивів, можна перейти до визначення критеріїв, за якими в межах цієї кваліфікаційної роботи будуть оцінюватися ARX-конструкції.

Для порівняльного аналізу критеріями будуть наступні характеристики:

– **Розмір ключа** — кількість бітів, що визначає секретний ключ шифру.

– **Розмір блоку** — кількість бітів, що визначає розмір вхідного та вихідного блоку шифру.

– **Кількість раундів** — кількість повторень раундової функції.

– **Раундова функція** — які і скільки операцій ARX використовується в кожному раунді шифру.

– **Швидкодія/ефективність** — продуктивність на ARM, x64. Зазвичай оцінюється в тактах на байт (clocks/byte), також можемо розглядати FPGA- і ASIC- інфраструктури з відповідними значеннями і позначками якщо такі бенчмарки існують в мережі.

– **Рівень безпеки** — відношення кількості раундів що вдалося зламати за одного набору ключа до загальної кількості раундів.

Також важливо зазначити що **сфера застосування** — важливий критерій, який, попри все, не буде виноситися в окремі таблиці через незручність подання інформації.

1.4 Методи криптоаналізу ARX-конструкцій

Систематизувавши ARX-примітиви та їхні структурні особливості, перейдемо до методів, якими оцінюють їхню стійкість. Повноцінно оцінити безпеку криптосистеми неможливо без розгляду криптоаналізу, тому нижче стисло розглянуто сучасні методи, розроблені спеціально для ARX-конструкцій.

Спершу формалізуємо поняття, якими послуговується подальший аналіз.

Означення 1.4 (Диференціальна ймовірність)**.** Нехай ϕ — відображення (операція). *Диференціальною ймовірністю* переходу вхідної різниці α у вихідну різницю β називається

$$\mathrm{DP}(\alpha \to \beta) = \Pr_x\big[\phi(x \oplus \alpha) \oplus \phi(x) = \beta\big],$$

де ймовірність береться за рівномірно розподіленим входом x. Для додавання за модулем 2^n застосовують позначення $xdp^+(\alpha, \beta \to \gamma)$ — імовірність того, що пара вхідних різниць (α, β) дає вихідну різницю γ.

Означення 1.5 (Лінійна апроксимація та зміщення)**.** *Лінійною апроксимацією* відображення $\phi\colon V_n \to V_n$ називається співвідношення $\langle a, x\rangle \oplus \langle b, \phi(x)\rangle = 0$ з масками $a, b \in V_n$, де $\langle u, v\rangle = \bigoplus_i u_i v_i$ — скалярний добуток над GF(2). *Зміщенням* (bias) апроксимації називають величину

$$\varepsilon = \Pr_x\big[\langle a, x\rangle = \langle b, \phi(x)\rangle\big] - \tfrac{1}{2},$$

а відповідну кореляцію — $c = 2\varepsilon$.

Означення 1.6 (Обертальна пара та обертальна ймовірність)**.** *Обертальною парою* з параметром r називається пара слів $(x, x \lll r)$. *Обертальною ймовірністю* відображення ϕ називають

$$\Pr_x\big[\phi(x \lll r) = \phi(x) \lll r\big].$$

Операції $\oplus$ та $\lll$ зберігають обертальне співвідношення з імовірністю 1, тоді як додавання за модулем 2^n його порушує; відповідну ймовірність позначають rp^+.

Означення 1.7 (Розрізнювач)**.** *Розрізнювачем* (distinguisher) примітиву Π називається ефективний алгоритм $\mathcal{D}$, що з доступом до оракула — яким є або Π, або ідеальний об'єкт $\mathcal{I}$ того самого інтерфейсу — повертає біт-здогадку, а величина

$$\mathrm{Adv}(\mathcal{D}) = \left|\Pr[\mathcal{D}^{\Pi} = 1] - \Pr[\mathcal{D}^{\mathcal{I}} = 1]\right|$$

називається його *перевагою*. Розрізнювач з істотною перевагою доводить відхилення Π від ідеальної поведінки, проте не обов'язково дає змогу відновити ключ.

Основним методом аналізу ARX-систем є обертальний криптоаналіз, запропонований Ховратовичем та Ніколічем на конференції FSE 2010. Метод ґрунтується на тому, що операції XOR та циклічного зсуву комутують з одночасним циклічним зсувом усіх слів стану, тоді як операція модульного додавання цю симетрію порушує [26].

Теорема 1.1 (Ховратович–Ніколіч, FSE 2010 [26])**.** *Імовірність того, що циклічний зсув на r позицій зберігається під час одного додавання за модулем 2^n, тобто $(x + y) \lll r = (x \lll r) + (y \lll r)$ для рівномірно випадкових $x, y \in V_n$, дорівнює*

$$rp^+(r) = \tfrac{1}{4}\big(1 + 2^{r-n} + 2^{-r} + 2^{-n}\big).$$

Зокрема, для $r = 1$ і великих n маємо $rp^+(1) \approx \frac{3}{8} = 2^{-1.415}$. Доведення наведено у [26].

Використовуючи ймовірнісний аналіз цього порушення, можна конструювати атаки на основі обертальних пар – пар вхідних значень, пов'язаних фіксованим циклічним зсувом. Узагальненням обертального криптоаналізу є RX-криптоаналіз, розроблений Ашуром та Лю і

опублікований у виданні *IACR Transactions on Symmetric Cryptology* у 2016 році [27]. Метод враховує присутність адитивних констант у раундових функціях, що в чистому обертальному аналізі суттєво ускладнювало побудову трейлів через порушення обертальної симетрії.

У класичному диференціальному криптоаналізі центральним інструментом є таблиця розподілу різниць (DDT – Difference Distribution Table), яка описує розподіл вихідних різниць для кожної можливої вхідної різниці S-блоку. В ARX-конструкціях S-блоки відсутні, а вся арифметична нелінійність зосереджена в операції модульного додавання за модулем 2^n. Це принципово змінює характер диференціального аналізу: замість скінченної таблиці підстановок аналітик має справу з імовірнісним відображенням, де диференціальна ймовірність одного кроку визначається розподілом переносів при двійковому додаванні.

Перший систематичний підхід до обчислення диференціальних ймовірностей модульного додавання був запропонований Ліпмаа та Моріаї на конференції FSE 2001. Їхній алгоритм дозволяє ефективно обчислювати ймовірність того, що пара вхідних слів з заданою різницею дасть задану вихідну різницю після однієї операції модульного додавання [25].

Теорема 1.2 (Ліпмаа–Моріаї, FSE 2001 [25])**.** *Нехай $\alpha, \beta, \gamma \in V_n$ — XOR-різниці, а* $\mathrm{eq}(x,y,z)$ *— n-бітове слово, що має одиницю в позиції i тоді й лише тоді, коли $x_i = y_i = z_i$. Диференціал $(\alpha, \beta \to \gamma)$ при додаванні за модулем 2^n можливий ($xdp^+ > 0$) тоді й лише тоді, коли*

$$\mathrm{eq}(\alpha \lll 1, \beta \lll 1, \gamma \lll 1) \wedge \big(\alpha \oplus \beta \oplus \gamma \oplus (\beta \lll 1)\big) = 0.$$

У цьому разі

$$xdp^+(\alpha, \beta \to \gamma) = 2^{-\mathrm{wh}(\neg\,\mathrm{eq}(\alpha,\beta,\gamma))},$$

де $\mathrm{wh}(\cdot)$ *— кількість позицій $i \in \{0, \ldots, n-2\}$, у яких біти $\alpha_i, \beta_i, \gamma_i$ не всі однакові. Доведення наведено у [25].*

Наприклад, для $\alpha = 0\ldots01$, $\beta = 0\ldots00$, $\gamma = 0\ldots01$ умова

можливості виконується, а $\neg\,\mathrm{eq}$ має одиницю лише в позиції 0, тож $xdp^{+}(\alpha,\beta \to \gamma) = 2^{-1}$.

Розвитком цього підходу стала модель S-функцій, запропонована Моухою та співавторами, яка поширює обчислення диференціальних ймовірностей на складені ARX-конструкції [8]. Сучасним інструментом автоматизованого пошуку диференціальних трейлів є методи на основі MILP (Mixed Integer Linear Programming) та SAT-розв'язувачів, що дозволяють систематично перебирати простір можливих диференціальних шляхів у багатораундових конструкціях і знаходити трейли з оптимальною ймовірністю.

Диференціально-лінійний криптоаналіз є комбінованим підходом, що поєднує диференціальні та лінійні апроксимації на різних сегментах атакованого примітиву. Застосований до ARX-конструкцій, цей метод забезпечив найкращі відомі результати для зменшеної кількості раундів ChaCha та SPARX [9]: диференціальна частина атаки охоплює нелінійні сегменти, де визначальну роль відіграє модульне додавання, тоді як лінійна апроксимація ефективно описує ті сегменти конструкції, де нелінійний внесок є мінімальним. Отримані результати підтверджують, що комбінування класичних методів з урахуванням специфіки ARX-дизайну є потужним інструментом для оцінки криптографічної стійкості сучасних ARX-примітивів і визначення запасу безпеки відносно повнораундових атак.

1.5 Висновки до 1 розділу

У першому розділі було розглянуто теоретичні основи ARX-дизайну та проведено систематизацію криптографічних примітивів, побудованих на операціях додавання за модулем 2^n, циклічного зсуву та XOR. Показано, що поєднання зазначених операцій дозволяє реалізовувати ефективні та портативні криптографічні алгоритми без використання таблиць підстановок, що є однією з характерних особливостей

ARX-конструкцій.

Було формалізовано поняття ARX-перестановки та блокового шифру з ARX-структурою, а також обґрунтовано доцільність поділу ARX-примітивів на дві окремі групи для подальшого порівняльного аналізу: криптографічні перестановки та блокові шифри. Встановлено, що блокові шифри утворюють окремий клас ARX-конструкцій, оскільки є сімействами ключозалежних перестановок і мають власні структурні особливості.

Також було визначено критерії порівняльної оцінки ARX-конструкцій, серед яких розмір ключа, розмір блоку, кількість раундів, структура раундової функції, швидкодія та рівень криптографічної стійкості. Запропонований набір критеріїв буде використано для аналізу конкретних алгоритмів у наступних розділах роботи.

Окрему увагу приділено сучасним методам криптоаналізу ARX-конструкцій. Розглянуто основні підходи, зокрема диференціальний, обертальний та RX-криптоаналіз, а також диференціально-лінійні методи аналізу. Наведено базові поняття та результати, що лежать в основі оцінювання стійкості ARX-примітивів до відомих атак.

Отримані результати створюють необхідну теоретичну основу для подальшого дослідження та порівняльного аналізу конкретних ARX-перестановок і блокових шифрів, що буде виконано в наступних розділах.

2 ARX-ПЕРЕСТАНОВКИ ЯК БАЗОВИЙ КРИПТОГРАФІЧНИЙ ПРИМІТИВ

Якщо перший розділ було присвячено загальним принципам ARX-дизайну та методам його аналізу, то предметом цього розділу є безпосередні об'єкти дослідження – ARX-перестановки як самостійні криптографічні примітиви. У сучасній криптографії перестановка $\pi\colon V_n \to V_n$ відіграє роль універсального будівельного блоку: одна й та сама конструкція може слугувати ядром потокового шифру, компресійною функцією гешу, внутрішньою перестановкою губчастої схеми, примітивом MAC або AEAD. Саме ця універсальність робить вивчення ARX-перестановок у відриві від конкретного режиму принципово важливим: властивості примітиву визначають безпеку будь-якої системи, що на ньому ґрунтується.

Було вирішено систематизувати перестановки у п'ять сімейств за спільною структурою раундової функції, зв'язком «примітив – спадкоємець» та цільовою областю.

2.1 Сімейство Salsa20 – ChaCha – Forró

Це сімейство об'єднує три перестановки з однаковим 512-бітним станом (16 слів по 32 біти) та єдиною ідеєю: двовимірне перемішування (стовпцевий і діагональний раунди) із фінальним пословним додаванням початкового стану. Еволюція визначається пришвидшенням розсіювання: Salsa20 встановив базову конструкцію, ChaCha змінив порядок операцій у раундовій функції та перейшов до діагонального перемішування, Forró вирішив структурне обмеження 4-слівного чвертьраунду, ввівши 5-слівну функцію з перекриваючими групами слів.

Salsa20 – π

Перестановку Salsa20 – π запропонував Деніел Дж. Бернштейн у 2005 році в рамках конкурсу eSTREAM як ядро однойменного потокового шифру [5]. Стан – вектор $X = (x_0, \ldots, x_{15}) \in \mathbb{Z}_{2^{32}}^{16}$, що природно подається матрицею 4×4. Базовим будівельним блоком є чвертьраундова функція $\mathrm{QR}(a,b,c,d)$.

Алгоритм 1. *Чвертьраундова функція Salsa20.*

Вхід: чотири слова $a, b, c, d \in \mathbb{Z}_{2^{32}}$.

Вихід: оновлені значення (a', b', c', d').

1. $b \leftarrow b \oplus ((a + d) \lll 7)$.
2. $c \leftarrow c \oplus ((b + a) \lll 9)$.
3. $d \leftarrow d \oplus ((c + b) \lll 13)$.
4. $a \leftarrow a \oplus ((d + c) \lll 18)$.

Алгоритм 2. *Перестановка Salsa20 (20 раундів).* (Див. схему А.1)

Вхід: стан $X = (x_0, \ldots, x_{15}) \in \mathbb{Z}_{2^{32}}^{16}$.

Вихід: перетворений стан $X' \in \mathbb{Z}_{2^{32}}^{16}$.

1. $X_0 \leftarrow X$.
2. Повторити 10 разів:
   (а) *Стовпцевий раунд:*
   $$\mathrm{QR}(x_0,x_4,x_8,x_{12}),$$
   $$\mathrm{QR}(x_5,x_9,x_{13},x_1),$$
   $$\mathrm{QR}(x_{10},x_{14},x_2,x_6),$$
   $$\mathrm{QR}(x_{15},x_3,x_7,x_{11}).$$
   (б) *Рядковий раунд:*
   $$\mathrm{QR}(x_0,x_1,x_2,x_3),$$
   $$\mathrm{QR}(x_5,x_6,x_7,x_4),$$
   $$\mathrm{QR}(x_{10},x_{11},x_8,x_9),$$
   $$\mathrm{QR}(x_{15},x_{12},x_{13},x_{14}).$$
3. $x'_i \leftarrow x_i + (X_0)_i, \quad i = 0, \ldots, 15.$

Криптоаналіз потокового шифру Salsa20. Алгоритм

Ліпмаа–Моріаї [25] дає змогу обчислювати диференціальну ймовірність одного модульного додавання, що є основним інструментом аналізу Salsa20. Оммасон, Фішер, Хазаї, Майер та Рехбергер у роботі [10] вперше застосували метод *probabilistic neutral bits* до сімейства «латинських танців»: вони виявили, що певні біти початкового стану мають статистично незначущий вплив на обрані вихідні біти впродовж перших кількох раундів. Це дозволяє ефективно перебирати частину ключового простору і зламати Salsa20 зі скороченою кількістю раундів: на 7 раундах складність атаки становить 2^{151} операцій, а Salsa20/8 (8 раундів) зламується зі складністю 2^{251} замість 2^{256}.

Хвратович та Ніколіч [26] показали, що QR-функція Salsa20 не зберігає обертальну симетрію: ймовірність обертального переходу через один QR становить 2^{-2}, через два суміжних QR – 2^{-4}. Для повного 20-раундового Salsa20 обертальних розрізнювачів не існує, проте скорочені варіанти (до 8 раундів) уразливі до комбінованих обертально-диференціальних атак.

S-функції Моухи та співавторів [8] узагальнюють аналіз диференціальних ймовірностей для складених ARX-конструкцій. У Salsa20 лінійна нелінійність QR пов'язана з переносами при модульному додаванні; лінійні апроксимації одного QR мають ймовірність відхилення від $\frac{1}{2}$, що не перевищує 2^{-1}. Для повного 20-раундового Salsa20 відсутні відомі атаки, кращі за перебір 2^{256}.

Застосування та вирішення структурних обмежень. Перестановка Salsa20 є публічною композицією ефективно оборотних операцій, тому обернене відображення π^{-1} обчислюється з тією самою складністю, що й сама перестановка: за будь-яким виходом $y = \pi(x)$ прообраз $x = \pi^{-1}(y)$ відновлюється за час одного обчислення перестановки. Отже, ізольована перестановка не може бути однобічною функцією і не містить секретного параметра, що унеможливлює її безпосереднє використання як примітиву гешування або MAC без зовнішнього механізму руйнування оборотності. У потоковому шифрі Salsa20 ця проблема вирішується інакше: ключ k,

нонс n та лічильник t завантажуються у конкретні позиції початкового стану (клітинки x_0,x_5,x_{10},x_{15} – «рядок» констант; x_1,x_2,x_3, x_4, x_{11},x_{12},x_{13},x_{14} – ключ; x_6,x_7 – нонс; x_8,x_9 – лічильник), а вихід перестановки XOR-ується з відкритим текстом. Фінальне додавання $X' = \text{Rounds}(X) + X$ забезпечує псевдовипадковість виходу навіть за відомого X. Використовується у Salsa20 ,XSalsa20 [32] (192-бітний нонс) та як прообраз BLAKE G-функції.

ChaCha – π

На відміну від Salsa20, де QR розпочинається з XOR, Бернштейн у 2008 році [4] запропонував схему ADD-XOR-ROT: кожне нове додавання відразу залежить від попереднього XOR. Друга принципова зміна — перехід від рядкових до *діагональних* раундів, що формує щільнішу залежність між операціями і прискорює наростання лавинного ефекту та усуває потребу в транспонуванні матриці між раундами і покращує паралелізм на SIMD-архітектурах.

Алгоритм 3. *Чвертьраундова функція ChaCha.*

Вхід: $(a, b, c, d) \in \mathbb{Z}_{2^{32}}^4$.

Вихід: (a', b', c', d').

1. $a \leftarrow a + b; \quad d \leftarrow (d \oplus a) \lll 16.$
2. $c \leftarrow c + d; \quad b \leftarrow (b \oplus c) \lll 12.$
3. $a \leftarrow a + b; \quad d \leftarrow (d \oplus a) \lll 8.$
4. $c \leftarrow c + d; \quad b \leftarrow (b \oplus c) \lll 7.$

Алгоритм 4. *Перестановка ChaCha (20 раундів).* (Див. схему А.2)

Вхід: $X \in \mathbb{Z}_{2^{32}}^{16}$.

Вихід: $X' \in \mathbb{Z}_{2^{32}}^{16}$.

1. $X_0 \leftarrow X$.
2. Повторити 10 разів:
   (a) *Стовпцевий раунд:*

$$\text{QR}(x_0,x_4,x_8,x_{12}),$$
$$\text{QR}(x_1,x_5,x_9,x_{13}),$$

$$\mathrm{QR}(x_2,x_6,x_{10},x_{14}),$$
$$\mathrm{QR}(x_3,x_7,x_{11},x_{15}).$$

(б) *Діагональний раунд:*

$$\mathrm{QR}(x_0,x_5,x_{10},x_{15}),$$
$$\mathrm{QR}(x_1,x_6,x_{11},x_{12}),$$
$$\mathrm{QR}(x_2,x_7,x_8,x_{13}),$$
$$\mathrm{QR}(x_3,x_4,x_9,x_{14}).$$

3. $x'_i \leftarrow x_i + (X_0)_i, \quad i = 0,\ldots,15.$

Застосування та вирішення структурних обмежень.

ChaCha–π застосовується як ядро однойменного потокового шифру, за такою самою механікою як це робить його попередник у лиці Salsa20–π.

Але для того щоб подувати нам повноцінний протокол в нас не вистачає автентифікації: ChaCha дає лише конфіденційність, без гарантій цілісності. Це обмеження знімається на рівні схеми: RFC 8439 поєднує ChaCha20 з автентифікатором Poly1305 у AEAD-конструкцію ChaCha20-Poly1305. Окремо коротка довжина нонса (96 біт, ризик повторення за випадкового вибору) розширюється до 192 біт через HChaCha20 у схемі XChaCha20.

Криптоаналіз потокового шифру ChaCha.

Aumasson et al. [10] показали, що ChaCha є більш стійким за Salsa20: атака на ChaCha7 вимагає 2^{248} операцій. Coutinho та Souza Neto [9] розробили покращений метод лінійних апроксимацій для ARX і отримали диференціально-лінійну атаку на 7-раундовий ChaCha зі складністю $2^{228.51}$. Dey et al. [44] вдосконалили цей метод та знизили складність атаки на ChaCha7 до $2^{221.95}$; подальші роботи 2023 р. досягли складності порядку 2^{207}, проте повний 20-раундовий ChaCha20 залишається криптографічно стійким.

Діагональна структура ChaCha робить його більш стійким до обертального аналізу порівняно з Salsa20: діагональні раунди перемішують слова матриці у різних напрямках, знижуючи ймовірність обертального переходу між двома суміжними станами [26].

RX-криптоаналіз Ашура і Лю [27] враховує вплив адитивних констант раундів на обертальні трейли. ChaCha20 (20 раундів) не має жодних відомих атак, кращих за повний перебір 2^{256}. Окремий клас становлять атаки за побічними каналами: Кумар та співавтори [7] продемонстрували практичну збійну атаку (fault attack) на програмну реалізацію ChaCha20, яка відновлює секретний стан за незначної кількості ін'єкцій збоїв.

Forró – π

Попри покращення ChaCha відносно Salsa20, обидва шифри мають спільне структурне обмеження: QR(a,b,c,d) щоразу охоплює чотири слова з фіксованих груп, і між сусідніми викликами QR у межах одного раунду немає прямої залежності. Coutinho et al. (ASIACRYPT 2022) [14] виявили це як причину уповільненого росіювання та запропонували 5-слівну функцію Q(a,b,c,d,e), де п'яте слово e береться зі суміжної групи: один виклик Q одразу перемішує елементи двох різних стовпців.

Алгоритм 5. *П'ятислівна функція Forró.*

Вхід: $(a, b, c, d, e) \in \mathbb{Z}_{2^{32}}^5$.

Вихід: (a', b', c', d', e').

1. $d \leftarrow d + e$
2. $c \leftarrow c \oplus d$
3. $b \leftarrow b + c$
4. $b \leftarrow b \lll 10$
5. $a \leftarrow a + b$
6. $e \leftarrow e \oplus a$
7. $d \leftarrow d + e$
8. $d \leftarrow d \lll 27$
9. $c \leftarrow c + d$
10. $b \leftarrow b \oplus c$
11. $a \leftarrow a + b$
12. $a \leftarrow a \lll 8$

Алгоритм 6. *Перестановка Forró (7 раундів).* (Див. схему А.3)

Вхід: $X \in \mathbb{Z}_{2^{32}}^{16}$.

Вихід: $X' \in \mathbb{Z}_{2^{32}}^{16}$.

1. $X_0 \leftarrow X$.
2. Повторити 7 разів:
   (а) *Стовпцевий прохід:*
   $$\mathrm{Q}(x_0,x_4,x_8,x_{12},x_3),$$
   $$\mathrm{Q}(x_1,x_5,x_9,x_{13},x_0),$$
   $$\mathrm{Q}(x_2,x_6,x_{10},x_{14},x_1),$$
   $$\mathrm{Q}(x_3,x_7,x_{11},x_{15},x_2).$$
   (б) *Діагональний прохід:*
   $$\mathrm{Q}(x_0,x_5,x_{10},x_{15},x_3),$$
   $$\mathrm{Q}(x_1,x_6,x_{11},x_{12},x_0),$$
   $$\mathrm{Q}(x_2,x_7,x_8,x_{13},x_1),$$
   $$\mathrm{Q}(x_3,x_4,x_9,x_{14},x_2).$$
3. $x'_i \leftarrow x_i + (X_0)_i, \quad i = 0,\ldots,15.$

Застосування. Forró застосовується як ядро однойменного потокового шифру з тією самою схемою ключового завантаження, що й ChaCha20: ключ, нонс та лічильник завантажуються у фіксовані позиції 4×4-матриці, а виходи послідовних перестановок утворюють ключовий потік. 7 раундів замість 10 при рівнозначному запасі стійкості зменшують кількість обчислювальних операцій приблизно на 30%.

Криптоаналіз.

Завдяки 5-слівній функції Q Forró досягає повного розсіювання за менше раундів, що підтверджено в оригінальному дослідженні [14]: найкраща диференціальна атака на Forró сягає 5 раундів, тоді як повна 7-раундова перестановка залишається за межами відомих методів.

Перехресні залежності між групами слів через п'яте слово e ускладнюють побудову обертальних пар: при однаковому обертальному зсуві всіх слів модульне додавання у Q порушує симетрію у двох парах замість однієї (як у ChaCha).

Загальна структура Forró схожа на ChaCha, але більш щільні залежності між словами сприяють швидшому наростанню нелінійності. Відсутність обертальних пар і більш щільне розсіювання роблять Forró більшстійким до лінійного та диференціально-лінійного криптоаналізу порівняно з ChaCha, хоча повний аналіз Forró ще не завершено.

Порівняння потокових шифрів на основі цих перестановок

Нижче наведена таблиця яка структурує ці потокові шифри як семейство.

Критерій	**Salsa20**	**ChaCha20**	**Forró14**
Розмір ключа (біт)	128-256	256	256
Розмір солі (біт)	64	96	64
Кількість раундів (рекомендована)	20	20	14
$\oplus$ за раунд	16	16	16
$\lll$ за раунд	16	16	8
$+$ за раунд	16	16	16
Повне розсіювання (раундів)	4	3	2
Рівень безпеки	8/20 (40%)	7/20 (35%)	5.5/14 (39%)

Таблиця 2.1 – Порівняльна таблиця базових варіацій потокових шифрів Salsa20, ChaCha та Forró

Також окремо порівняємо швидкодію цих шифрів на різних архітектурах — ARMv8 (64 bits), Intel x86-64 and Intel x86-64 використовуючи SIMD.[6]

Архітектура	**Salsa20**	**ChaCha20**	**Forro20**
ARMv8	6,01cpb	8,56cpb	11,4cpb
Intel x86-64	5,01cpb	4,91cpb	8,41cpb
Intel x86-64 v SIMD	1,08cpb	0,96cpb	1,52cpb

Таблиця 2.2 – Порівняльна таблиця базових варіацій потокових шифрів Salsa20, ChaCha та Forró

Як ми можемо побачити, за рахунок додавання 5 слова в функцію перемішування, навіть попри зменшену кількість раундів Forro14 програє в швидкості своїм старішим представникам.

2.2 Сімейство BLAKE

Геш-функцію BLAKE запропонували Оммасон, Хензен, Майер та Фань у 2008 р. [15] у рамках конкурсу NIST SHA-3. Базовим блоком для кожного геша цього сімейства, включно з усіма спадкоємцями, є G-функція, структурно аналогічна чвертьраунду ChaCha з двома додатковими ін'єкціями слів повідомлення та (у першій редакції) констант. Усі п'ять розглянутих перестановок використовують ту саму G-функцію; різняться вони лише розрядністю слова, ротаційними сталими, наявністю XOR-констант та кількістю раундів. Тому опишемо G-функцію один раз у параметризованому вигляді, а конкретні представники задамо як її параметризації.

G-функція

(Див. схему А.4)

Вхід: слова стану a,b,c,$d \in \mathbb{Z}_{2^w}$ (де $w = 32$ або $w = 64$); слова повідомлення $m_i, m_j \in \mathbb{Z}_{2^w}$; для BLAKE-256/512 – додатково константи $c_i, c_j \in \mathbb{Z}_{2^w}$.

Змінні параметри: ротаційні сталі (r_1,r_2,r_3,r_4).

Вихід: (a', b', c', d').

1. $a \leftarrow a + b + (m_i \oplus c_j)$.
2. $d \leftarrow (d \oplus a) \ggg r_1$.
3. $c \leftarrow c + d$.
4. $b \leftarrow (b \oplus c) \ggg r_2$.
5. $a \leftarrow a + b + (m_j \oplus c_i)$.
6. $d \leftarrow (d \oplus a) \ggg r_3$.
7. $c \leftarrow c + d$.
8. $b \leftarrow (b \oplus c) \ggg r_4$.

У BLAKE2 та BLAKE3 XOR-константи відсутні, тож кроки 1 і 5 спрощуються до $a \leftarrow a + b + m_i$ та $a \leftarrow a + b + m_j$ відповідно. Перестановка утворюється стовпцево-діагональним застосуванням G до 16-слівного стану, поданого матрицею 4×4: один раунд складається з чотирьох викликів G над стовпцями та чотирьох над діагоналями, а повна перестановка повторює раунд N_r разів. Конкретні параметри подано в Таблиці **??**.

Перестановка	**Слово** w	**Стан**	N_r	(r_1,r_2,r_3,r_4)	**Конст.**
BLAKE-256	32	512	14	(16,12,8,7)	так
BLAKE-512	64	1024	16	(32,25,16,11)	так
BLAKE2s	32	512	10	(16,12,8,7)	ні
BLAKE2b	64	1024	12	(32,24,16,63)	ні
BLAKE3	32	512	7	(16,12,8,7)	ні

Таблиця 2.3 – Параметри перестановок сімейства BLAKE

Також задамо раундову функцію і так звану процедуру фіналізації для BLAKE-256.

Алгоритм 7. *Раундова функція BLAKE.* (Див. схему А.5)

Вхід: стан $X = (v_0, \ldots, v_{15})$, подбаний матрицею 4×4; слова повідомлення $m_0, \ldots, m_{15}$; перестановка σ_r для раунду r.

Вихід: оновлений стан.

1. *Стовпцевий прохід:*

$$\mathrm{G}_0(v_0,v_4,v_8,v_{12}),$$
$$\mathrm{G}_1(v_1,v_5,v_9,v_{13}),$$
$$\mathrm{G}_2(v_2,v_6,v_{10},v_{14}),$$
$$\mathrm{G}_3(v_3,v_7,v_{11},v_{15}).$$

2. *Діагональний прохід:*

$$\mathrm{G}_4(v_0,v_5,v_{10},v_{15}),$$
$$\mathrm{G}_5(v_1,v_6,v_{11},v_{12}),$$
$$\mathrm{G}_6(v_2,v_7,v_8,v_{13}),$$

$$\mathrm{G}_7(v_3,v_4,v_9,v_{14}).$$

Алгоритм 8. *Фіналізація BLAKE-256.*

Вхід: ланцюгове значення $h_0,\ldots,h_7$; сіль $s_0,\ldots,s_3$; стан $v_0,\ldots,v_{15}$ після раундів.

Вихід: нове ланцюгове значення $h'_0,\ldots,h'_7$.

$$\begin{aligned}
h'_0 &\leftarrow h_0 \oplus s_0 \oplus v_0 \oplus v_8, & h'_4 &\leftarrow h_4 \oplus s_0 \oplus v_4 \oplus v_{12},\\
h'_1 &\leftarrow h_1 \oplus s_1 \oplus v_1 \oplus v_9, & h'_5 &\leftarrow h_5 \oplus s_1 \oplus v_5 \oplus v_{13},\\
h'_2 &\leftarrow h_2 \oplus s_2 \oplus v_2 \oplus v_{10}, & h'_6 &\leftarrow h_6 \oplus s_2 \oplus v_6 \oplus v_{14},\\
h'_3 &\leftarrow h_3 \oplus s_3 \oplus v_3 \oplus v_{11}, & h'_7 &\leftarrow h_7 \oplus s_3 \oplus v_7 \oplus v_{15}.
\end{aligned}$$

BLAKE-256 та BLAKE-512

Це початкова редакція: G містить XOR-константи, а перестановка виконує 14 раундів для 32-бітної версії та 16 – для 64-бітної. BLAKE-512 відрізняється від BLAKE-256 лише розрядністю слова, ротаціями (32,25,16,11) та двома додатковими раундами; структура перестановки тотожна.

Криптоаналіз. Бірюков, Ніколіч і Рой [17] побудували високоймовірнісні диференціальні трейли на 2–3 раунди й на їх основі – бумеранг-атаки: розрізнювач компресійної функції BLAKE-256 на 7 раундів (2^{232}) та ключованої перестановки на 8 раундів (2^{242}); Хао [18] знизив складність 8-раундового результату до 2^{198} і побудував 8,5-раундовий розрізнювач ключованої перестановки BLAKE-512 зі складністю 2^{464}. Окремо Оммасон та співавтори [20] отримали tuple-розрізнювач перестановки на 4 середні раунди зі складністю лише 2^{64}. Повні 14- та 16-раундові перестановки відомих атак не мають.

XOR-ування різних констант у кожному зверненні до G руйнує симетрію між обертальними парами: у RX-формалізмі Ашура і Лю [27] обертальних розрізнювачів для повних перестановок не побудовано. Успадкована від ChaCha структура ADD-XOR-ROT стримує накопичення лінійної кореляції, тож лінійних атак, кращих за загальні межі, не відомо.

BLAKE2s та BLAKE2b

Друга редакція [16] прибирає XOR-константи з G (слова повідомлення додаються безпосередньо), скорочує кількість раундів (10 для BLAKE2s, 12 для BLAKE2b) і переобирає ротації лише для 64-бітної гілки: BLAKE2s зберігає сталі BLAKE-256, а BLAKE2b замінює (25,11) на SIMD-дружні (24,63), де $\ggg 63 \equiv \lll 1$ та байтово вирівняні 16/24/32 покращують векторизацію.

Криптоаналіз. Найкращі відомі результати – бумеранг-розрізнювачі ключованої перестановки: 8,5 раунду для BLAKE2b (2^{474}) та 7,5 раунду для BLAKE2s (2^{184}), причому коректність підтверджено практичними 6,5-раундовими четвірками [18]. Принципово, що бумеранги для BLAKE не переносяться на BLAKE2 автоматично: як показали Гуо та співавтори [19], ініціалізація BLAKE2 жорстко фіксує шість слів стану й зменшує число ступенів вільності атакувальника, через що окремі зміни навіть підвищили стійкість до бумеранг-атак. Повні перестановки атак на зіткнення, прообрази чи відновлення ключа не мають – усі наявні результати є розрізнювачами.

BLAKE3

BLAKE3 [30] використовує ту саму G-функцію, що й BLAKE2s (32-бітні слова, ротації (16,12,8,7)), але скорочену до 7 раундів. Принципова відмінність – не в перестановці, а в режимі обв'язки: замість послідовної HAIFA-конструкції попередників BLAKE3 застосовує *деревоподібний* (Merkle-tree) режим із поділом входу на чанки по 1 КіБ, що обробляються незалежно й паралельно, з лічильником та доменними прапорцями замість окремих констант. Це дає необмежений паралелізм, вбудований режим XOF (довільна довжина виходу) та KDF. Криптоаналіз самої перестановки наслідується від BLAKE2s; новизна BLAKE3 лежить у конструкції, а не в примітиві.

Застосування та вирішення структурних обмежень

Це сімейство широко використовується для перевірки цілісності файлів, у Git-like системах, протоколах Argon2 (RFC 9106), Zcash і Sia.

Порівняльний аналіз BLAKE функцій

Далі надана побудована таблиця з підсумками у вигляді порівнянь геш-функцій цього сімейства

Критерій	**BLAKE-256**	**BLAKE-512**	**BLAKE2s**	**BLAKE2b**	**BLAKE3**
Розмір блоку (біт)	512	1024	512	1024	512
Розмір слова (біт)	32	64	32	64	32
Розмір стану (біт)	512	1024	512	1024	512
Конструкція	HAIFA	HAIFA	HAIFA	HAIFA	дерево Меркла
Кількість раундів	14	16	10	12	7
$\oplus$ за раунд	48	48	32	32	32
$\lll$ за раунд	32	32	32	32	32
$+$ за раунд	48	48	48	48	48
Атака**	8/14	8.5/16	7.5/10	8.5/12	—*

Таблиця 2.4 – Порівняльна таблиця параметрів геш-функцій родини BLAKE

*— — доки не існує атак на BLAKE3.

** — мається на увазі атака на перестановку, а не на всю геш функцію.

Надалі пропонуються дві окремі таблиці порівняння швидкодії даних алгоритмів, бо єдиного бенчмарку всіх п'яти функцій на одній машині в літературі не існує.Тому розглянемо окремо порівняння BLAKE3—BLAKE2S—BLAKE2B і BLAKE-256—BLAKE-512—BLAKE2S—BLAKE2B.

Тести проводились на x-86-64 використовуючи SIMD, і були запозичені з офіційного git-hub аккаунта oconnor663.

Ці дані були взяті зі статті «BLAKE2: simpler, smaller, fast as MD5»[16] і математично обраховані. Вони слугують для теоретичного

Реалізація	**BLAKE2s**	**BLAKE2b**	**BLAKE3**
x86-64, SIMD	4,66 cpb	2,81 cpb	0,95 cpb

Таблиця 2.5 – Порівняння швидкодії геш-функцій BLAKE3—BLAKE2S—BLAKE2B

Архітектура	**BLAKE-256**	**BLAKE-512**	**BLAKE2s**	**BLAKE2b**
Sandy Bridge (Intel)	7,49	5,76	5,34	3,32
Bulldozer (AMD)	11,83	6,88	8,20	5,29

Таблиця 2.6 – Порівняння швидкодії геш-функцій BLAKE-256, BLAKE-512, BLAKE2s, BLAKE2b на довгих повідомленнях

розуміння того що загалом є BLAKE3—найшвидший, а BLAKE першої версії є найповільнішим алгоритмом. Ні в якому разі неможна співставляти дані з таблиць вище напряму, бо бенчмарки останньої проводилися в 2012–2013 р. на залізі того часу, зараз би це було зовсім інші результати, але сама суть у тому що, BLAKE3 при будь яких розкладах швидший за BLAKE2s/BLAKE2b, а вони, в свою чергу швидші за BLAKE256/512.

2.3 SipRound і Chaskey-π: MAC і геш-орієнтовані перестановки

Обидві перестановки об'єднані спільною метою: ефективний MAC для малих і вбудованих пристроїв. Структура схожа: чотири слова стану поділяються на дві пари, кожна пара змішується незалежно через ADD-XOR-ROT, потім виконується перехресне змішування між парами. SipRound оперує 64-бітними словами (стан 256 біт), Chaskey π – 32-бітними (стан 128 біт).

Алгоритм 9. *SipRound.*

Це перестановка є математичною функцією у хеш-функції

SipHash.(Див. схему А.6) розроблена Оммасоном та Бернштейном у 2012 р. [22].

Вхід: $(v_0,v_1,v_2,v_3) \in \mathbb{Z}_{2^{64}}^4$.

Вихід: (v'_0,v'_1,v'_2,v'_3).

1. $v_0 \leftarrow v_0 + v_1$
2. $v_1 \leftarrow (v_1 \lll 13) \oplus v_0$
3. $v_0 \leftarrow v_0 \lll 32$
4. $v_2 \leftarrow v_2 + v_3$
5. $v_3 \leftarrow (v_3 \lll 16) \oplus v_2$
6. *Перехресне змішування:* $v_0 \leftarrow v_0 + v_3$
7. $v_3 \leftarrow (v_3 \lll 21) \oplus v_0$
8. $v_2 \leftarrow v_2 + v_1$
9. $v_1 \leftarrow (v_1 \lll 17) \oplus v_2$
10. $v_2 \leftarrow v_2 \lll 32$

Криптоаналіз. Кожне застосування SipRound має диференціальну ймовірність, що визначається модульними додаваннями; метод Ліпмаа–Моріаї [25] дає верхню межу 2^{-4} для ненульового диференціала через один SipRound. SipHash-2-4 (2 раунди компресії + 4 раунди фіналізації) обрано авторами виходячи з аналізу нейтральних бітів: в 2022 р. опубліковані атаки на SipHash-1-1, але не на SipHash-2-4.

Константи обертання $(13, 16, 21, 17, 32)$ обрані так, щоб максимізувати розсіювання за одне застосування і уникнути обертальних розрізнювачів [26]. Лінійні апроксимації ізольованого SipRound можливі, але SipHash-2-4 приховує їх за рахунок ключових ін'єкцій і достатньої кількості раундів.

Застосування та вирішення структурних обмежень. SipRound як неключована перестановка не надає жодних MAC-гарантій: зловмисник, що знає стан, може довільно продовжити обчислення. У SipHash ця проблема вирішується XOR-ін'єкцією 128-бітного ключа у початковий стан через IV $(v_0 \dots v_3)$, де $v_i = \mathrm{IV}_i \oplus k_i$ (з константами і ключем). Це перетворює перестановку на PRF, безпека якого зводиться до

псевдовипадковості SipRound. SipHash-2-4 застосовується у CPython, Rust, Swift та ядрі Linux для захисту геш-таблиць від HashDoS.

Chaskey-π

Порівняно з SipRound, Chaskey π [23] оперує 32-бітними словами, що оптимізовано для 32-бітних ARM Cortex-M. Структура ідентична: дві незалежні пари + перехресне змішування. Розроблена Мухою, Меннінком та співавторами у 2014 р., стандартизована у ISO/IEC 29192-6. Варто зазначити, що Chaskey є повноцінним MAC-алгоритмом, але в даній роботі розглядається саме його ARX-ядро — Chaskey-π.

Алгоритм 10. *Один раунд Chaskey* π. (Див. схему А.7)

Вхід: $(v_0,v_1,v_2,v_3) \in \mathbb{Z}_{2^{32}}^4$.

Вихід: (v'_0,v'_1,v'_2,v'_3).

1. $v_0 \leftarrow v_0 + v_1$
2. $v_1 \leftarrow (v_1 \lll 5) \oplus v_0$
3. $v_0 \leftarrow v_0 \lll 16$
4. $v_2 \leftarrow v_2 + v_3$
5. $v_3 \leftarrow (v_3 \lll 8) \oplus v_2$
6. *Перехресне змішування:* $v_0 \leftarrow v_0 + v_3$
7. $v_3 \leftarrow (v_3 \lll 13) \oplus v_0$
8. $v_2 \leftarrow v_2 + v_1$
9. $v_1 \leftarrow (v_1 \lll 7) \oplus v_2$
10. $v_2 \leftarrow v_2 \lll 16$

Повна перестановка π виконує 12 послідовних застосувань раундової функції (Алгоритм 10), Chaskey-12.

Криптоаналіз

Ключовий результат по безпеці Chaskey – ключовідновлювальна атака Мавромати [13] на Chaskey-6 (6 раундів): вона експлуатує структуру Even–Mansour і потребує 2^{43} запитів на кожного з 2^{43} користувачів. Атака підтвердила, що 6 раундів недостатньо.

Льоран [12] удосконалив метод диференціально-лінійного аналізу за

допомогою *розбиття* (partitioning): замість одного лінійного апроксиматора використовується набір апроксиматорів, що діють на різних підмножинах пар. Це дало атаку на 7-раундовий Chaskey і продемонструвало, що Chaskey-8 має менший запас стійкості, ніж очікувалося. Саме тому специфікація була оновлена до Chaskey-12 [23].

Застосування та вирішення структурних обмежень. π є чистою бієкцією без ключового матеріалу. Схема Even–Mansour перетворює її на блоковий шифр:

$$E_K(m) = \pi(m \oplus K_1) \oplus K_2, \qquad D_K(c) = \pi^{-1}(c \oplus K_2) \oplus K_1,$$

де підключі K_1, K_2 виводяться зі 128-бітного ключа K. У MAC-конструкції Chaskey цей шифр застосовується у CBC-MAC-подібному режимі, тож обчислюється лише автентифікаційний тег, а обернена перестановка π^{-1} не потрібна. Схему стандартизовано в ISO/IEC 29192-6.

2.4 Alzette ARX-box та шляхи його використання

П'яте сімейство засноване на *64-бітному ARX-боксі* (Alzette) як атомарній одиниці, що масштабується у більші перестановки (SPARKLE) через лінійний шар. На відміну від усіх попередніх сімейств, де розмір стану фіксований, тут розмір задається кількістю пар, що дозволяє будувати SPARKLE256, SPARKLE384, SPARKLE512 зміною одного параметра.Також цьому сімейству належать шифри *TRAX*, *CRAX*, бо вони також будуються на цьому блоці.

Alzette

Розроблена Байєрле та співавторами у 2020 р. [24] як самостійний ARX-бокс. Стан – пара $(x,y) \in \mathbb{Z}_{2^{32}}^2$; параметр – константа $c \in \mathbb{Z}_{2^{32}}$.

Алгоритм 11. *ARX-бокс Alzette.* (Див. схему А.8)

Вхід: $(x,y) \in \mathbb{Z}_{2^{32}}^2$; константа c.

Вихід: (x',y').

1. $x \leftarrow x + (y \ggg 31); \quad y \leftarrow y \oplus (x \ggg 24); \quad x \leftarrow x \oplus c.$
2. $x \leftarrow x + (y \ggg 17); \quad y \leftarrow y \oplus (x \ggg 17); \quad x \leftarrow x \oplus c.$
3. $x \leftarrow x + y; \quad y \leftarrow y \oplus (x \ggg 31); \quad x \leftarrow x \oplus c.$
4. $x \leftarrow x + (y \ggg 24); \quad y \leftarrow y \oplus (x \ggg 16); \quad x \leftarrow x \oplus c.$

Диференціальний криптоаналіз. Байєрле та співавтори [24] провели вичерпний MILP/SAT-пошук трейлів і формалізували стійкість Alzette точними межами.

Теорема 2.1 (Байєрле та ін., CRYPTO 2020 [24])**.** *Для повного 4-крокового ARX-боксу Alzette максимальна очікувана ймовірність диференціального трейла (MEDCP) дорівнює* 2^{-6}*, а максимальна очікувана абсолютна кореляція лінійного трейла (MELCC) дорівнює* 2^{-2}*. Доведення (вичерпний пошук трейлів і експериментальна перевірка над фіксованими ключами) наведено у [24].*

Обертальний криптоаналіз. Ін'єкція константи c після кожного кроку руйнує обертальну симетрію: обертальна пара (x,y) і $(x \lll r, y \lll r)$ не може пройти через $x \leftarrow x \oplus c$ зі збереженням обертального співвідношення, якщо $c \neq c \lll r$. У специфікації SPARKLE для кожного Alzette-боксу використовується різна константа c_i (вибрана з числа π та e), що повністю виключає можливість обертальних атак [24].

Лінійний криптоаналіз. Згідно з Теоремою 2.1, максимальна абсолютна кореляція лінійного трейла через повний Alzette становить 2^{-2}: чотири різні константи обертання та ін'єкція константи на кожному кроці не дають лінійним апроксимаціям накопичувати більшу кореляцію.

Застосування та вирішення структурних обмежень. Alzette є самостійним примітивом, але як ізольована 64-бітна перестановка не придатна для побудови гешу або AEAD через малий стан: 2^{32} пошуків достатньо для пошуку зіткнення у 64-бітному виході. Перехід до SPARKLE вирішує цю проблему масштабуванням стану до 256–512 біт.

SPARKLE

Сімейство перестановок SPARKLE256/384/512 [3] побудоване шляхом паралельного застосування Alzette-боксів до пар слів з подальшим лінійним перемішуванням між парами.

Алгоритм 12. *Один крок SPARKLE256.* (Див. схему А.9)

Вхід: стан $S = (x_0,y_0,x_1,y_1,x_2,y_2,x_3,y_3) \in \mathbb{Z}_{2^{32}}^8$; номер кроку t.

Вихід: оновлений стан S'.

1. *ARX-шар*: для $i = 0,1,2,3$:
   (а) $(x_i, y_i) \leftarrow \text{Alzette}_{c_i}(x_i, y_i)$ (Алгоритм 11 з константою c_i).
   (б) $y_i \leftarrow y_i \oplus \text{RCON}[t]$.
2. *Лінійний шар*: $S \leftarrow \mathcal{M}_4(S)$, де $\mathcal{M}_4$ – фіксоване лінійне відображення над $\mathbb{Z}_{2^{32}}^8$.

Диференціальний криптоаналіз. Формальна межа диференціальної ймовірності Alzette ($\leqslant 2^{-6}$ за 4 кроки) є вхідними даними для MILP-аналізу SPARKLE: автори [3] провели вичерпний пошук диференціальних трейлів і показали, що для SPARKLE256 за 7 кроків логарифмічна ймовірність найкращого трейла не перевищує -128, що забезпечує 128-бітний рівень безпеки відносно диференціального криптоаналізу.

Обертальний криптоаналіз. Різні константи c_i у кожному Alzette-боксі та раундова константа $\text{RCON}[t]$ повністю блокують обертальні атаки, як доведено у [24]. Лінійний шар $\mathcal{M}_4$ додатково перемішує слова між боксами, унеможливлюючи незалежний аналіз окремих пар.

Застосування та вирішення структурних обмежень. SPARKLE як бієкція вбудовується у губчасту конструкцію: SCHWAEMM використовує duplex-режим, ESCH — sponge-режим. У губчастих схемах перестановка застосовується лише у прямому напрямку (фази поглинання та вичавлення), тож оберненість SPARKLE^{-1} не використовується і не потрібна навіть для дешифрування в duplex-режимі. Детальний розгляд цих схем наведено у підрозд. 3.4.

Окремо варто згадати два блокові шифри побудовані на 64-бітному ARX-боксі Alzette (A_c, Алгоритм 11) . Їх запропонували Байєрле, Бірюков та співавтори – та сама команда, що й SPARKLE, – в одній роботі з Alzette [24], застосувавши до нього стратегію довгого сліду. Ця спільність робить їх показовим прикладом того, як один ARX-бокс породжує і криптографічну перестановку (SPARKLE), і блокові шифри різних класів: компактний 64-бітний CRAX та широкий 256-бітний твікований TRAX.

CRAX

CRAX (точніше CRAX-S-10) – 64-бітний блоковий шифр зі 128-бітним ключем, побудований як ключочергувальний (key-alternating) шифр над ARX-боксом Alzette [24]. Завдяки гранично простому розкладу ключа, що не потребує попереднього обчислення, він є академічною альтернативою Speck-64/128 за програмною ефективністю й випереджає його на коротких повідомленнях у мікроконтролерних реалізаціях.

Алгоритм 13. *Раундова функція CRAX.*

Стан – пара $(x, y) \in \mathrm{GF}(2)^{32} \times \mathrm{GF}(2)^{32}$, ключ – $K = (k_0, k_1, k_2, k_3)$. Шифрування складається з 10 кроків; на кроці s $(0 \leqslant s < 10)$ до стану домішується лічильник кроку та пара ключових слів, після чого застосовується Alzette A_c з константою $c_{s \bmod 5}$:

$$(x, y) \leftarrow A_{c_{s \bmod 5}}\big(x \oplus s \oplus \kappa_0^s,\ y \oplus \kappa_1^s\big),$$

$$(\kappa_0^s, \kappa_1^s) = \begin{cases} (k_0, k_1), & s \text{ парне}, \\ (k_2, k_3), & s \text{ непарне}, \end{cases}$$

після чого виконується фінальне додавання ключа $(x, y) \leftarrow (x \oplus k_0,\ y \oplus k_1)$. П'ять констант $c_0, \ldots, c_4$ та структура A_c узяті з Алгоритму 11.

Алгоритм 14. *Дешифрування CRAX.*

Дешифрування проходить кроки у зворотному порядку, замінюючи A_c на обернений ARX-бокс A_c^{-1}: спершу знімається фінальне додавання $(x \oplus k_0, y \oplus k_1)$, далі для s від 9 до 0 застосовується $A^{-1}_{c_{s \bmod 5}}$ з наступним XOR відповідної пари ключових слів і лічильника s.

Алгоритм 15. *Функція генерування ключа CRAX.*

Окремого розкладу ключа немає: чотири слова майстер-ключа (k_0,k_1,k_2,k_3) використовуються безпосередньо, чергуючи пари (k_0,k_1) та (k_2,k_3) за парністю кроку. Саме ця відсутність попередньо обчисленого розкладу й робить CRAX винятково компактним за пам'яттю.

Криптоаналіз CRAX. Стійкість CRAX успадковує доведені межі Alzette (Теорема 2.1): 10 кроків (тобто 40 ARX-раундів) дають значний запас відносно однослідового диференціального та лінійного криптоаналізу за стратегією довгого сліду. Це порівняно недавня академічна конструкція (2020 р.) без відомих атак на повну версію.

TRAX

TRAX (TRAX-L-17) – твікований блоковий шифр із 256-бітним блоком, 256-бітним ключем і 128-бітним твіком [24]. На відміну від Threefish, він використовує 32-бітні слова (зручніші для векторизації) і має SPN-структуру з доведеними межами для всіх лінійних та всіх (пов'язаних за твіком) диференціальних слідів.

Алгоритм 16. *Крок TRAX.*

Стан – чотири 64-бітні гілки (x_b, y_b), $b = 0, \ldots, 3$. Шифрування складається з 17 кроків; на кроці s виконується:

1. *Ін'єкція твіку* (лише для непарних s): твік XOR-иться у дві гілки, $x_0 \oplus T_0, y_0 \oplus T_1, x_1 \oplus T_2, y_1 \oplus T_3$.

2. *ARX-шар*: для кожної гілки b додається підключ і застосовується Alzette, $(x_b, y_b) \leftarrow A_{c_{(4s+b) \bmod 8}}(x_b \oplus sk^s_{2b},\, y_b \oplus sk^s_{2b+1})$.

3. *Лінійний шар* ℓ', ідентичний шару $\mathcal{M}_4$ перестановки SPARKLE256: над $(\mathbb{F}_2^{16})^4$ він діє як $\ell'(z_1,z_2,z_3,z_4) = (z_4,\, z_3 \oplus z_4,\, z_2,\, z_1 \oplus z_2)$, після чого гілки переставляються.

Після 17 кроків виконується фінальне додавання підключа до всіх гілок.

Алгоритм 17. *Дешифрування TRAX.*

Дешифрування виконує кроки у зворотному порядку: спершу знімається фінальний підключ, далі для кожного кроку застосовуються обернений лінійний шар, потім A_c^{-1} із відніманням підключа в кожній гілці й, для непарних кроків, зняття твіку. Розклад підключів і твік збігаються з шифруванням.

Алгоритм 18. *Функція генерування ключа TRAX.*

Розклад ключа породжує 18 наборів по 8 підключових слів із 8-слівного стану ключа $(k_0, \ldots, k_7)$. Перед кожним кроком поточний стан стає черговим набором підключів, після чого оновлюється:

$$k_0 \leftarrow k_0 + k_1 + c_{2s \bmod 8}, \qquad k_2 \leftarrow k_2 \oplus k_3 \oplus s,$$
$$k_4 \leftarrow k_4 + k_5 + c_{(2s+1) \bmod 8}, \qquad k_6 \leftarrow k_6 \oplus k_7 \oplus (s \ll 16),$$

після чого слова стану циклічно зсуваються на одну позицію. Залежність від лічильника s (зокрема $s \ll 16$) запобігає ковзним і пов'язаним за твіком атакам.

Криптоаналіз TRAX. Завдяки SPN-структурі та доведеним межам Alzette автори [24] надають верхні оцінки ймовірності всіх диференціальних (у тому числі пов'язаних за твіком) і лінійних слідів – сильніша гарантія, ніж у Threefish. Заявлено стійкість у моделі пов'язаного твіку доти, доки число запитів (x, T) за фіксованого ключа не перевищує 2^{128}; у моделі пов'язаного ключа гарантій не надається. Як і CRAX, це конструкція 2020 р. без відомих атак на повну версію.

2.5 Висновки до 2 розділу

У другому розділі було проведено систематизацію сучасних ARX-перестановок та досліджено їх як самостійні криптографічні примітиви.

Розглянуті конструкції було об'єднано у сімейства відповідно до спільних принципів побудови, походження раундових функцій та сфер практичного застосування.

Проведений аналіз показав, що розвиток ARX-перестановок відбувається шляхом поступового вдосконалення базових раундових операцій та механізмів перемішування стану. На прикладі сімейств Salsa20–ChaCha–Forró, BLAKE, Threefish та SPARKLE було продемонстровано різні підходи до досягнення повного розсіювання, нелінійності та масштабування внутрішнього стану при збереженні переваг ARX-дизайну.

Окрему увагу приділено результатам сучасного криптоаналізу. Аналіз відомих атак показав, що більшість практичних результатів стосуються лише зменшеної кількості раундів, тоді як для повнораундових версій розглянутих перестановок ефективних атак, що порушують заявлений рівень безпеки, наразі не відомо. Це свідчить про наявність суттєвого запасу стійкості в сучасних ARX-конструкціях.

Також було показано, що окремі ARX-перестановки можуть виступати універсальними будівельними блоками для різних криптографічних систем. На їх основі будуються шифри, геш-функції, AEAD-схеми, губчасті конструкції тощо. Зокрема, ARX-бокс Alzette став основою як перестановки SPARKLE, так і блокових шифрів CRAX та TRAX, що демонструє можливість побудови цілих сімейств криптографічних примітивів навколо однієї базової ARX-конструкції.

Отримані результати дозволяють перейти до безпосереднього порівняльного аналізу ARX-конструкцій за визначеними критеріями та оцінити вплив особливостей їхньої структури на ефективність і криптографічну стійкість.

3 БЛОКОВІ ШИФРИ НА ARX

У цьому розділі ми розглянемо більшість ARX-блокових шифрів, що були запропоновані в науковій літературі. Розглядатимуться виключно щифри, що використовують лише 3 базові операції ARX: додавання за модулем, побітове додавання та циклічні зсуви. Це зумовлено тим, що перелік всіх відомих блокових шифрів, що використовують не чистий ARX-підхід, змусить нас сильно вийти за межі як теми дослідження, так і обсягу дипломної роботи.

3.1 ARX-шифри “До 2000 року”

Пропонується об’єднати шифри, що були винайдені до 2000 року, в окрему групу, і порівняти їх між собою, бо на сьогоднішній день вони не використовуються в практичних реалізаціях, хоча фактично деякі з них не є зламаними, але навіть вони були витіснені більш сучасними конструкціями, що мають кращі характеристики та є застарілими стандартами. Таку групу ми будемо називати *ARX-шифри "До 2000 року"*. Варто зазначити, що вони були засновані ще до того як офіційно був введений термін ARX, але при цьому вони повністю відповідають ARX-підходу.

До даної групи відносяться такі шифри: *FEAL*,*TEA*, *XTEA*, *XXTEA*, *RC5*.Розглянемо їх детальніше і проведемо порівняльний аналіз.

FEAL–Fast data Encipherment ALgorithm

Цей шифр був розробленний в 1987 році японським криптографом Хіроші Кобаяші. Він був розроблений як альтернатива DES, з метою підвищення швидкодії за рахунок меньшої кількості раундів. Цей шифр є першим повноцінним ARX-блоковим шифром.Цей шифр провалив стійкість до багатьох видів криптоаналізу, і на сьогоднішній день він не

використовується в практичних реалізаціях. Тому ми розглянемо його лише для історичної довідки, і не будемо проводити його детальний аналіз.[33]

TEA–Tiny Encryption Algorithm

Алгоритм TEA[34] був описаний у 1994 році Д. Уїлеттом і Р. Нідхамом, як програма що швидко виконується на 32-бітних процесорах, і використовує лише прості операції ARX.

Алгоритм TEA використовує 64-бітний блок, що робиється на 32-бітні слова v_0, v_1 (Фейстель) і 128-бітний ключ(k_0, k_1, k_2, k_3), і складається з 32 раундів.

Раундова функція шифрування

1) $sum+ = \delta$

2) $v_0+ = (v_1 \lll 4) + k_0 \oplus v_1 + sum \oplus (v_1 \ggg 5) + k_1$

3) $v_1+ = (v_0 \lll 4) + k_2 \oplus v_0 + sum \oplus (v_0 \ggg 5) + k_3$

де $\delta = 0x9E3779B9$ — константа, що є частиною золотого перетину.

Декодування виконується у зворотньому порядку, тобто спочатку віднімається константа δ від змінної sum, а потім виконуються віднімання у зворотньому порядку.

Криптоаналіз. TEA має ланку вразливостей що робить можливим поломку цього шифру. Його найбільша проблема полягає в ключі, він має надвзичайно тривіальний розклад як ми бачимо непарні раунди використовують k_0, k_1, а парні k_2, k_3. Вони потрапляють у раунд без жодного нелінійного перемішування що уможливлює так звані related-key атаки. Цим і скористалися Джон Келсі (John Kelsea), Брюс Шнайєра (Bruce Schneier) та Девід Вагнера (David Wagner) у 1997 році у роботі «Related-key cryptanalysis of 3-WAY, Biham-DES,CAST, DES-X, NewDES, RC2, and TEA», побудувавши атаку що ламає ключ з ймовірністю 1 з 2^{23} вибраних відкритих текстів.

Також варто зазначити про вразливість,яка полягає в тому, що в шифрі існують еквівалентні ключи. Зі структури функції шифрування

можна помітити що одночасне перевертання старшого біта в певних парах ключових слів не змінює вихід, бо ці зміни взаємно скасовуються у XOR-комбінації доданків. Звідси стійкість 128-бітного ключа TEA фактично еквівалентна стікості 126 бітного ключа.

XTEA

Невдовзі після виявлення related-key вразливостей у TEA його автори, Д. Уїлер і Р. Нідхем, запропонували розширення[36], відоме як XTEA, що усуває обидві відомі слабкості оригіналу.

XTEA працює з тим самим 64-бітним блоком v_0, v_1 (мережа Фейстеля), 128-бітним ключем (k_0, k_1, k_2, k_3) і 32 раундами, проте вносить дві ключові зміни порівняно з TEA: змінює розклад ключа та повільніше подає ключовий матеріал у раунд.

Раундова функція шифрування

1) $v_0 += ((v_1 \lll 4) \oplus (v_1 \ggg 5)) + v_1 \ \oplus \ sum + k_{sum \,\&\, 0b11}$

2) $sum += \delta$

3) $v_1 += ((v_0 \lll 4) \oplus (v_0 \ggg 5)) + v_0 \ \oplus \ sum + k_{(sum \ggg 11) \,\&\, 0b11}$

де $\delta = 0x9E3779B9$, як і в TEA.

Принципова відмінність — у виборі ключового слова. У XTEA індекс ключового слова обчислюється динамічно з накопичувача sum для v_0, а v_1 додається ще і циклічний зсув на 11, це дозволяє використати усі підключі вже у 2 раундах. Саме ця залежність вибору ключа від sum руйнує тривіальну структуру розкладу, на якій трималися диференціальні related-key атаки на TEA.

Дешифрування, як і в TEA, виконується у зворотному порядку.

Криптоаналіз. Схема з динамічним обчисленням індексу ключового слова вирішила проблему TEA з releated-key атакою, як само описано вище. Натомість було досліджено що дана структура формування функції шифрування стає вразливою до атаки "посередині"(Meet-in-the-Middle)[37] Г. Секар, Н. Моуха, В Велічков та Б. Пренел склали таблицю, який підключ використовується в якому раунді і

виявили, що деякий підключі не використовується протягом великого блоку послідовних раундів. На цій властивості були побудовані атаки на 7, 15 і 23 раунди.

XXTEA—Corrected Block TEA

Слід зазначити, що в тій же самій технічній записці було[36] авторами було запропоновано іншого спадкоємця TEA $\longrightarrow$ **Block TEA**. Основна його ідея була в тому що на відміну від XTEA, цей шифр шифрував блок довільної множини. Тобто фактично весь відкритий текст це був 1 суцільний блок. Ця ідея була зламана вже наступного року, у 1998 він був повністю зламаний, навіть програмно Сааріняном[38], який виявив очевидний недолік у вигляді надзвичайно повільного розсіювання, бо цей шифр циклічно застосовував раунд TEA лише до 1 сусіда ліворуч, і застошувавши атаку на основі відомого шифротексту повністю зломав програмно шифр зі складністю 2^{34}, зазначивши що це не межа, а лише його реалізація, і передбачавши що це можна зробити ще швидше– зі складністю 2^{28} Саме тому вважаю недоречно розглядати його і одразу перейти до спроби виправити всі недоліки у останньому варіанті —XXTEA(Corrected Block TEA). Він був реалізован у тому ж році, що і зламан Block TEA Д. Уїлеттом і Р. Нідхамом[39]. Цей шифр працює с блоком довільної довжини кратної 32-бітам, що складається з n слів v_0, $\ldots$, v_{n-1}, і тим самим 128-бітним ключем (k_0, k_1, k_2, k_3). Блок трактується як циклічний масив, а кількість повних циклів залежить від його розміру й становить $q = 6 + \lfloor 52/n \rfloor$.

Раундова функція шифрування. У межах одного циклу для кожного слова v_p ($p = 0, \ldots, n-1$, з обгортанням індексів за модулем n) виконується:

1) $sum \mathrel{+}= \delta$

2) $e = (sum \ggg 2) \,\&\, 0b11$

3) $v_p \mathrel{+}= \mathrm{MX}(v_{p-1}, v_{p+1})$

де функція змішування $\mathrm{MX}(v_{p-1}, v_{p+1})$ визначається як

$$((v_{p-1} \ggg 5) \oplus (v_{p+1} \lll 2)) + ((v_{p+1} \ggg 3) \oplus (v_{p-1} \lll 4)) \oplus$$

$$\oplus (sum \oplus v_{p+1}) + (k_{(p \,\&\, 0b11) \oplus e} \oplus v_{p-1})$$

.

Дешифрування також виконується у зворотньому порядку.

Криптоаналіз. XXTEA була криптографічно проаналізована Е. Яаррков у 2010 році у його роботі «Cryptanalysis of XXTEA»[40].Його ідея аналізу полагала у його сумнівах в достатньості кількості раундів (всього 6 при щонайменше 53 слів).Ця властивість представляє декілька привабливих можливих підходів для криптоаналітика. Він провів диференціальний криптоаналіз, двома підходами:

– Шукається різниця, яка залишається в одному слові блока через кілька раундів з екстремально малою ймовірністю (2^{-100}) до (2^{-109}), але коли збігається, атака легко розпізнавна через колізії в більшості слів.

– Шукається різниця, яка систематично переміщується в сусідні слова на кожному раунді з вищою ймовірністю ($2^{-56.6}$), що робить пошук правильної пари значно практичнішим — потрібно лише 2^{59} пар текстів у відкритому тексті.

Так Яаррков з першиим підхідом провів вдалу атаку на шифр і відновив ключ з кількістю раундів зменшеною до 3, в іншу чергу він досяг успіху вже для алгоритма який складається з 4 раундів за допомогою 2^{35} пар текстів, коли попередня оцінка була $2^{34.7}$

RC5

Це суттєво інший ARX-шифр від попередніх з сімейства «До 2000 року», який був розроблений Рональдом Рівестом в 1997 році[41]. Цей алгоритм працює з різним розміром блоку і ключа, а також з різною кількістю раундів, тому прийнято позначати як **RC5-W/R/r**, де

– **W** — половина довжини блока(можливі значення 16, 32 і 64).Для

ефективності реалізації прийнято брати довжину рівну машинному слову

– **R** — кількість раундів. Можливі значення від 0 до 255.

– **r** — довжина ключа в байтах, значення від 0 до 255.

Блок даних складається з двох W-бітних слів, нехай A і B. Перед першим раундом виконується розширення ключа. Вона состоїть з 4 етапів: *генерування констант*, *розбиття ключа на слова*, *побудова таблиці розширених ключів*, *перемішування*. Простіше кажучи цей етап генерує масив S з $2(R+1)$ слів на основі вихідного ключа.

Раундова функція шифрування. Для кожного раунду $i = 0, 1, \ldots, R-1$ виконується:

1) $A \mathrel{+}= S_{2i}$

2) $B \mathrel{+}= S_{2i+1}$

3) $A = ((A \oplus B) \lll B) + S_{2i}$

4) $B = ((B \oplus A) \lll A) + S_{2i+1}$

Дешифрування проводиться в зворотньому порядку.

Криптоаналіз. Компанія RSA витратила багато часу на аналіз його роботи з 64-бітним блоком. Так, у період з 1995 по 1998 рр. вони опублікували низку звітів, у яких детально проаналізували криптостійкість алгоритму RC5. Оцінка для лінійного криптоаналізу показує, що алгоритм є безпечним після 6 раундів. Диференційний криптоаналіз має складність: 2^{24} для 5 раундів; 2^{45} для 10 раундів; 2^{53} для 12; 2^{68}, для 15 раундів. Для порівняльного аналіза ми будемо разглядати аналог до параметрів DES, так як він був створений ще й як конкурент стандарту того часу — RC5-32/16/7.

Порівняльний аналіз даного кластеру блокових шифрів

Підсумуючи наш аналіз перших з блокових шифрів отримаємо таблицю, де за заданими критеріями порівнюємо дані алгоритми:

Критерій	**TEA**	**XTEA**	**XXTEA**	**RC5-32/16/7**
Розмір ключа	128	128	128	56
Розмір блоку	64	64	$n \cdot 32$	64
Кількість раундів	32	32	q	16
$\lll$ $\ggg$ в раунді	5	5	5	2
$+$ в раунді	6	4	4	3
$\oplus$ в раунді	4	4	6	2
Найкраща атака/раунд	$2^2 3$ /64	18 / 23	2^{35} / 4	2^{68} /15

Таблиця 3.1 – Порівняння ARX-шифрів сімейства «До 2000 року»

$^*q = 6 + \lfloor 52/n \rfloor$.

3.2 Сімейство Speck і SPARK

Speck є найвідомішим і найдослідженішим сучасним легковаговим сімейством шифрів у світі що був розроблений у 2013 році командою вчених що працюють в NSA(Агенство національної безпеки США) Він був представлений поруч з Simon, що не є ARX-шифром в глобальному розумінні, тому розгляжаємо саме сімейство шифрів Speck. Цей шифр буде повністю описаний з офіційної документації «Simon and Speck FAmilies of Lightweight Block Ciphers»[1].

Існує 10 конфігурацій даного шифру, які задаються як Speck $2n/mn$,де $2n$ — довжина блоку, mn — довжина ключа. Нижче наведена таблиця можливих параметрів:

В таблиці було згадано також константи зсуву α та β відповідно до специфікації для подальшого зручного подання раундової функції.

Алгоритм 19. *Раундова функція Speck.*

розмір блоку $2n$	**розмір ключа** mn	**розмір слова** n	**слів ключа** m	**конс. зсуву** α	β	**раундів** T
32	64	16	4	7	2	22
48	72	24	3	8	3	22
	96		4			23
64	96	32	3	8	3	26
	128		4			27
96	96	48	2	8	3	28
	144		3			29
128	128	64	2	8	3	32
	192		3			33
	256		4			34

Таблиця 3.2 – Параметри Speck

Для $k \in \mathrm{GF}(2)^n$ ключозалежна раундова функція Speck$2n$ є відображенням $R_k : \mathrm{GF}(2)^n \times \mathrm{GF}(2)^n \to \mathrm{GF}(2)^n \times \mathrm{GF}(2)^n$, що задається як

$$R_k(x, y) = \big(((x \ggg \alpha) \boxplus y) \oplus k,\ (y \lll \beta) \oplus ((x \ggg \alpha) \boxplus y) \oplus k \big),$$

Блок-схема раундової функція зображена у А.10.

Алгоритм 20. *Дешифрування.*

Дешифрування в Speck робиться за допомогою оберненої раундової функції

$$R_k^{-1}(x, y) = (((x \oplus k) \lll \alpha - (x \oplus y) \ggg \beta),\ (x \oplus y) \ggg \beta).$$

Алгоритм 21. *Функція генерування ключа.*

Початковий ключ записується як $K = (\ell_{m-2}, ..., \ell_0, k_0)$, тоді

$$\ell_{i+m-1} = (k_i + (\ell_i \ggg \alpha)) \oplus i \quad \text{i}$$

$$k_{i+1} = (k_i \lll \beta) \oplus \ell_{i+m-1}.$$

значення k_i це i-тий раундовий ключ, для $0 \leqslant i < T$, T —кількість раундів. **Криптоаналіз Speck.** Розробники стверджують що Speck, хоч і є легковаговим шифром, але має високий рівень криптосійкості від всих видів атак. Це підтверджується більше ніж 70 опублікованих робіт по криптоаналізу даного шифру.

Найкращими атаками на даний шифр є атаки диференційного криптоаналізу, які, в залежності від варіації ломають приблизно 70процентів раундів, і є незначно швидшими ніж звичайний бруд-форс. Нижче подано таблиця складності атак.

Варіант	**Взломано раундів**	**Складність за часом**	**Необхідні дані**
Speck128/256	25/34 (74%)	$2^{253.35}$	$2^{125.35}$
Speck128/192	24/33 (73%)	$2^{189.35}$	$2^{125.35}$
Speck128/128	23/32 (72%)	$2^{125.35}$	$2^{125.35}$
Speck96/144	21/29 (72%)	$2^{143.94}$	$2^{95.94}$
Speck96/96	20/28 (71%)	$2^{95.94}$	$2^{95.94}$
Speck64/128	20/27 (74%)	$2^{125.56}$	$2^{61.56}$
Speck64/96	19/26 (73%)	$2^{93.56}$	$2^{61.56}$
Speck48/96	17/23 (74%)	$2^{95.8}$	$2^{47.8}$
Speck48/72	16/22 (73%)	$2^{71.8}$	$2^{47.8}$
Speck32/64	15/22 (68%)	$2^{63.39}$	$2^{31.39}$

Таблиця 3.3 – Результати криптоаналізу Speck

Говорячи про Speck важко не згадати про блоковий легковаговий шифр **SPARX**.[42] Це спадкоємець нашого шифру Speck який використовує його перестановку Speckey(це фактично раундова функція Speck32/64 без додавання ключа),як ARX-блок(як CRAX,TRAX і Alzette-блок). Вважаю недоречним розглядати повноцінно цей шифр, бо він не є криптографічним ARX-примітивом, але про SPARX варто сказати головне: на відміну від Speck, чия стійкість спирається лише на емпіричну відсутність ефективних атак попри численні спроби

криптоаналізу, він був спроєктований за стратегією довгого сліду (Long Trail Strategy), яка вперше для ARX-сімейства дозволяє математично довести верхні межі ймовірності диференціальних і лінійних слідів. Саме це робить його показовим орієнтиром для порівняння: він демонструє, як той самий примітив, що лежить в основі Speck, можна вбудувати в конструкцію з формальними гарантіями стійкості.Нижче сформована таблиця з результатами анатізу SPARX.

Інстанс	**Взломано раундів**	**Складність за часом**	**Необхідні дані**	**Пам'ять**
Sparx-64/128	15/24 (62%)	2^{101}	2^{37}	2^{64}
Sparx-128/128	22/32 (69%)	2^{105}	2^{102}	2^{72}
Sparx-128/256	24/40 (60%)	2^{233}	2^{104}	2^{202}

Таблиця 3.4 – Результати криптоаналізу SPARX

3.3 HIGHT

HIGHT (High Security and Lightweight) запропоновано Хонгом та співавторами на CHES 2006 [2].Наразі цей корейский шифр є легковаговим стандартом(ISO/IEC 18033-3:2010), призначений для апаратних реалізацій з жорсткими обмеженнями ресурсів (RFID, смарт-картки, IoT-вузли).

Шифр має 64-бітний блок, 128-бітний ключ і 32 раунди. Стан шифру — масив з восьми байтів $(X_0,X_1,\ldots,X_7) \in (\mathbb{Z}_{2^8})^8$. Нелінійність забезпечується двома байтовими функціями, побудованими виключно з циклічних зсувів та XOR [2]:

$$F_0(x) = (x \lll 1) \oplus (x \lll 2) \oplus (x \lll 7), \tag{3.1}$$

$$F_1(x) = (x \lll 3) \oplus (x \lll 4) \oplus (x \lll 6), \tag{3.2}$$

де $x \in \mathbb{Z}_{2^8}$. Обидві функції є лінійними відображеннями над $\mathrm{GF}(2)^8$;

нелінійність шифру забезпечується чергуванням F_0, F_1 з модульним додаванням.

Ключовий матеріал поділяється на ключі відбілювання та раундові підключі:

Алгоритм 22. *Ключі відбілювання HIGHT.*

Вхід: майстер-ключ $MK = (MK_0,MK_1,\ldots,MK_{15}) \in (\mathbb{Z}_{2^8})^{16}$.

Вихід: вектор відбілювання $(WK_0,\ldots,WK_7)$.

1. Для $0 \leqslant i \leqslant 3$: $WK_i \leftarrow MK_{i+12}$.
2. Для $4 \leqslant i \leqslant 7$: $WK_i \leftarrow MK_{i-4}$.

Алгоритм 23. *Ключовий розклад HIGHT.*

Вхід: $MK \in (\mathbb{Z}_{2^8})^{16}$.

Вихід: 128 раундових підключів $(SK_0,\ldots,SK_{127})$, кожен по 8 біт.

1. $\delta_0 \leftarrow (1011010)_2$ (7-бітна константа).
2. Для $1 \leqslant i \leqslant 127$:

$$\delta_i \leftarrow \big((\delta_{i-1} \oplus (\delta_{i-1} \ggg 3)) \lll 6\big) \oplus (\delta_{i-1} \ggg 1).$$

3. Для $0 \leqslant i \leqslant 7$, $0 \leqslant j \leqslant 7$:

$$SK_{16i+j} \leftarrow MK_{(j-i) \bmod 8} + \delta_{16i+j}, \quad SK_{16i+j+8} \leftarrow MK_{((j-i) \bmod 8)+8} + \delta_{16i+j+8}.$$

Алгоритм 24. *Шифрування HIGHT.*

Вхід: відкритий текст P; підключі WK, SK.

Вихід: шифротекст $C = (C_0,\ldots,C_7)$.

1) $P = P_7\|P_6\|P_5\|P_4\|P_3\|P_2\|P_1\|P_0$ (P_i байти відкритого тексту)

2)

$$\begin{aligned}
X_{0,0} &\leftarrow P_0 + WK_0, \quad X_{0,1} \leftarrow P_1,\\
X_{0,2} &\leftarrow P_2 \oplus WK_1, \quad X_{0,3} \leftarrow P_3,\\
X_{0,4} &\leftarrow P_4 + WK_2, \quad X_{0,5} \leftarrow P_5,\\
X_{0,6} &\leftarrow P_6 \oplus WK_3, \quad X_{0,7} \leftarrow P_7.
\end{aligned}$$

3) for $i = 0$ to 30:

$$\begin{aligned}
X_{i+1,0} &\leftarrow X_{i,7} \oplus (F_0(X_{i,6}) + SK_{4i+3}), \quad X_{i+1,1} \leftarrow X_{i,0},\\
X_{i+1,2} &\leftarrow X_{i,1} + (F_1(X_{i,0}) \oplus SK_{4i}), \qquad X_{i+1,3} \leftarrow X_{i,2},\\
X_{i+1,4} &\leftarrow X_{i,3} \oplus (F_0(X_{i,2}) + SK_{4i+1}), \quad X_{i+1,5} \leftarrow X_{i,4},\\
X_{i+1,6} &\leftarrow X_{i,5} + (F_1(X_{i,4}) \oplus SK_{4i+2}), \quad X_{i+1,7} \leftarrow X_{i,6}.
\end{aligned}$$

for $i = 31$:

$$\begin{aligned}
X_{i+1,0} &\leftarrow X_{i,0}, \quad X_{i+1,1} \leftarrow X_{i,1} + (F_1(X_{i,0}) \oplus SK_{124}),\\
X_{i+1,2} &\leftarrow X_{i,2}, \quad X_{i+1,3} \leftarrow X_{i,3} \oplus (F_0(X_{i,2}) + SK_{125}),\\
X_{i+1,4} &\leftarrow X_{i,4}, \quad X_{i+1,5} \leftarrow X_{i,5} + (F_1(X_{i,4}) \oplus SK_{126}),\\
X_{i+1,6} &\leftarrow X_{i,6}, \quad X_{i+1,7} \leftarrow X_{i,7} \oplus (F_0(X_{i,6}) + SK_{127}).
\end{aligned}$$

4)

$$\begin{aligned}
C_0 &\leftarrow X_{32,0} + WK_4, \quad C_1 \leftarrow X_{32,1},\\
C_2 &\leftarrow X_{32,2} \oplus WK_5, \quad C_3 \leftarrow X_{32,3},\\
C_4 &\leftarrow X_{32,4} + WK_6, \quad C_5 \leftarrow X_{32,5},\\
C_6 &\leftarrow X_{32,6} \oplus WK_7, \quad C_7 \leftarrow X_{32,7}.
\end{aligned}$$

5) $C \leftarrow C_7 \| C_6 \| C_5 \| C_4 \| C_3 \| C_2 \| C_1 \| C_0$ (C_i байти шифротексту)

Алгоритм 25. *Дешифрування HIGHT.*

Операція дешифрування ідентична операції шифрування, за винятком двох наступних модифікацій.

1) Всі операції + замінюються операціями −, крім операцій +, що з'єднують SK_i з виходами F_0.

2) Порядок застосування ключів WK_i та SK_i обернений.

Криптоаналіз HIGHT. Автори [2] довели стійкість повних 32 раундів до диференціального та лінійного криптоаналізу. Подальші дослідження встановили: неможливі диференціали охоплюють до 27 раундів у сценарії

з одним ключем; атаки зі зв'язаним ключем — до 31 раунду; лінійний криптоаналіз обмежений 16 раундами. Жодної практично реалізованої атаки на повний 32-раундовий шифр не відомо [2].

3.4 Український легковаговий шифр Кипарис

«Кипарис» запропоновано Родінко та Олійниковим у 2017 році [45] як постквантовий малоресурсний симетричний блоковий шифр фейстель-подібної структури з раундовою ARX-функцією. Специфікація передбачає дві реалізації: «Кипарис-256» для 32-бітної архітектури та «Кипарис-512» для 64-бітної; останнє число в назві дорівнює довжині блоку відкритого тексту й довжині ключа в бітах.

Параметри шифру задаються довжиною слова s та кількістю раундів t. Для «Кипарис-256» маємо $s = 32$, $t = 10$; для «Кипарис-512» — $s = 64$, $t = 14$. Розмір блоку й ключа складає $8s$ біт. Константи циклічного зсуву (r_1,r_2,r_3,r_4) дорівнюють (16,12,8,7) для «Кипарис-256» та (32,24,16,15) для «Кипарис-512» [45].

Стан шифру — блок із восьми s-бітних слів $x = x_0\|x_1\|\cdots\|x_7 \in (\mathbb{Z}_{2^s})^8$. Нелінійність забезпечується модульним додаванням у складі ARX-функції h, що оперує над чотирма s-бітними словами:

Алгоритм 26. *ARX-функція h «Кипарис».*

Вхід: чотири слова $(y_0,y_1,y_2,y_3) \in (\mathbb{Z}_{2^s})^4$; константи зсуву r_1,r_2,r_3,r_4.

Вихід: перетворені слова (y_0,y_1,y_2,y_3).

1. $y_0 \leftarrow y_0 + y_1, \quad y_3 \leftarrow (y_3 \oplus y_0) \lll r_1$;
2. $y_2 \leftarrow y_2 + y_3, \quad y_1 \leftarrow (y_1 \oplus y_2) \lll r_2$;
3. $y_0 \leftarrow y_0 + y_1, \quad y_3 \leftarrow (y_3 \oplus y_0) \lll r_3$;
4. $y_2 \leftarrow y_2 + y_3, \quad y_1 \leftarrow (y_1 \oplus y_2) \lll r_4$.

Раундова функція F є дворазовим застосуванням h до результату

побітового додавання входу з раундовим ключем:

$$F(L, RK) = h\big(h(L \oplus RK)\big). \tag{3.3}$$

Шифрування використовує класичну схему Фейстеля: вхідний блок розбивається на дві половини по $4s$ біт, що оновлюються протягом t раундів.

Алгоритм 27. *Шифрування «Кипарис».*

Вхід: блок відкритого тексту $x = x_0\|\cdots\|x_7$; раундові ключі $RK^{(0)},\ldots,RK^{(t-1)}$, кожен довжиною $4s$ біт.

Вихід: шифротекст $C = L_t\|R_t$.

1. $L_0 \leftarrow x_0\|x_1\|x_2\|x_3, \quad R_0 \leftarrow x_4\|x_5\|x_6\|x_7$.
2. for $i = 1$ to t:

$$L_i \leftarrow R_{i-1} \oplus F\big(L_{i-1}, RK^{(i-1)}\big), \qquad R_i \leftarrow L_{i-1}.$$

3. $C \leftarrow L_t\|R_t$.

Дешифрування. Завдяки фейстелівській структурі дешифрування Cypress-256 використовує ту саму ARX-функцію без її обернення: достатньо застосувати ті самі раундові ключі в зворотному порядку.

Криптоаналіз Кипарис.

Родінко та співавтори [46] застосували метод побудови багатораундових диференціальних характеристик Кипарису-256. Пошук показав, що найбільш імовірна диференціальна характеристика, що спирається на одноraундові характеристики з 4–6 активними бітами, розподіленими між різними словами, охоплює щонайбільше **6 раундів**. Жодна характеристика не вдалася для 7 чи більше раундів, що задовольняє практичний критерій стійкості до диференціального криптоаналізу для 10-раундового шифру [46].

3.5 Великий блоковий шифр для побудови геш-функції Shein

Threefish

Threefish значно виділяється серед усіх розглянутих блокових шифрів. Це єдиний алгоритм який має просто велетенські розміри блоків і ключів, і не вважається легковаговим. Його задача полягає в тому щоб бути надійною основою для геш-функції Shein, через це пріоритом була висока продуктивність на потужних платформах.Він був розроблений Фергюсоном, Луксом, Шнайером та співавторами у 2010 р. [21].

Цей алгоритм являється *Твікованим блоковим шифром.*

Означення 3.1 (Твік)**.** *Твікованим блоковим шифром* називається відображення $\widetilde{E}\colon V_k \times V_\theta \times V_n \to V_n$, для якого при кожному фіксованому $(K,T) \in V_k \times V_\theta$ відображення $\widetilde{E}(K,T,\cdot)$ є бієкцією V_n. Параметр T називається *твіком*; на відміну від ключа, він відкритий і може вільно змінюватися, задаючи незалежні перестановки для одного ключа.

Далі представлені всі конфігурації даного шифру. Зазначимо що T(твік) приймає 128 біт.

Довжина блоку	**Довжина ключа**	**Кількість слів N_w**	**Кількість раундів N_r**
256	256	4	72
512	512	8	72
1024	1024	16	80

Таблиця 3.5 – Параметри Threefish

Розклад ключа (описаний нижче) перетворює ключ і твік на послідовність з $N_r/4+1$ підключів, кожен з яких складається з N_w слів. Слова підключа s позначаємо через $(k_{s,0},\ldots,k_{s,N_w-1})$.

Нехай $v_{d,i}$ — значення i-го слова стану шифрування після d раундів. Починаємо з

$$v_{0,i} := p_i \quad \text{для } i = 0, \ldots, N_w - 1,$$

після чого застосовуємо N_r раундів, пронумерованих $d = 0, \ldots, N_r - 1$. У кожному раунді ми додаємо підключ, якщо $d \bmod 4 = 0$. Для $i = 0, \ldots, N_w - 1$ маємо

$$e_{d,i} := \begin{cases} (v_{d,i} + k_{d/4,\,i}) \bmod 2^{64} & \text{якщо } d \bmod 4 = 0, \\ v_{d,i} & \text{інакше.} \end{cases}$$

Перестановки змішування та слів означені так:

$$(f_{d,2j}, f_{d,2j+1}) := \mathrm{MIX}_{d,j}(e_{d,2j}, e_{d,2j+1}) \quad \text{для } j = 0, \ldots, N_w/2 - 1,$$

$$v_{d+1,i} := f_{d,\pi(i)} \quad \text{для } i = 0, \ldots, N_w - 1.$$

До речі, саме з нього взяли структуру раунду з шифру Speck 19, точніше з базової функції MIX.

Алгоритм 28. *MIX-функція Threefish.*

Вхід: $(x_0, x_1) \in \mathbb{Z}_{2^{64}}^2$; константа обертання $R \in \{0, \ldots, 63\}$.

Вихід: (y_0, y_1).

1. $y_0 \leftarrow x_0 + x_1$.
2. $y_1 \leftarrow (x_1 \lll R) \oplus y_0$.

Коротко ознайомитися з блок-схемою раунду Threefish-512 можна на рисунку А.16

Дешифрування. Дешифрування виконує ті самі кроки у зворотному порядку: віднімання відповідного підключа, обернену перестановку слів Π_8^{-1} та обернену MIX-функцію; ключовий розклад і твік збігаються з шифруванням.

Диференціальний криптоаналіз. MIX-функція є простою ARX-операцією над парою слів; її диференціальна ймовірність аналізується методом Ліпмаа–Моріаї [25]. Константи обертання $R_{d,j}$

обрані так, щоб мінімізувати диференціальну імовірність за 8 раундів без ключових ін'єкцій. Через регулярну структуру Π_8 побудова ефективних диференціальних трейлів для повного Threefish ускладнена: будь-який трейл повинен «переживати» чотири незалежних перестановки і ключові ін'єкції.

Обертальний криптоаналіз. Хвратович, Ніколіч та Рехбергер [11] розробили *обертально-рикошетну атаку* (*rotational rebound*) на Skein: вона поєднує Обертальний криптоаналіз [26] з технікою рикошету і дозволяє будувати обертальні розрізнювачі для 53 з 72 раундів Threefish-256 і 57 з 72 раундів Threefish-512. Ці атаки є розрізнювачами, а не атаками на відновлення ключа: вони доводять відхилення поведінки від ідеальної перестановки, але не надають практичного способу відновити вхід за виходом. Повна 72-раундова перестановка з ключами і твіком вважається практично стійкою: жодних ефективних ключовідновлювальних атак не опубліковано.

Застосування та вирішення структурних обмежень. Threefish сам по собі не може прямо використовуватися як геш-функції, через відсутність зв'язку з конкретним повідомленням. У геш-функції Skein ця проблема вирішується конструкцією UBI (*Unique Block Iteration*): $H_i = E^{T_i}_{H_{i-1}}(m_i) \oplus m_i$, де E^T_K – Threefish з ключем K і твіком T_i, що кодує метадані (номер блоку, прапори типу даних).

Варто також зазначити, що порівняння з іншими представниками даного розділу не є доцільним, зовсім інша ідея у цього алгоритму. Але не описати цей ARX-примітив було би помилку через його історичну важність і сучасне використання.

3.6 Порівняльний аналіз

Переходимо до фактичного порівняльного аналізу за обраними критеріями в пункті 1.3. Для початку, вважаю недоречно порівнювати Threefish між іншими блоковими шифрами бо він вже був описаний у

порівняння криптографічних геш-функціях, побудованих на ARX-конструкціях.

Структурне порівняння ARX-шифрів Speck, SPARX, HIGHT та Кипарис можна подивитися в таблиці **??**.

Завершення порівняння варто додати емпіричний вимір — програмну ефективність. Для цього скористаємося результатами проєкту FELICS (Fair Evaluation of Lightweight Cryptographic Systems) [40], який є відкритим фреймворком об'єктивного вимірювання легковагових блокових шифрів на трьох вбудованих платформах: 8-бітовій AVR ATmega128, 16-бітовій TI MSP430 та 32-бітовій ARM Cortex-M3. Для кожного шифру вимірюються три метрики — розмір коду, споживання оперативної пам'яті та час виконання (у тактах), — які зводяться в єдиний показник Figure-of-Merit (FOM); менше значення FOM відповідає кращій реалізації. Нижче наведено результати для Сценарію 1 (шифрування й дешифрування 128 байтів у режимі CBC). Окрім нейтрального зважування метрик, FELICS подає й зважений профіль, у якому час виконання має вагу 2, а розмір коду й пам'ять — по 1, тобто пріоритет надається швидкодії.

Слід наголосити на межах цього порівняння. По-перше, FELICS охоплює лише легковагові блокові шифри, тож із розглянутих у цьому розділі конструкцій бенчмарк-дані наявні тільки для Speck, SPARX, HIGHT та RC5; шифри родини TEA (TEA, XTEA, XXTEA), а також Threefish (як шифр зовсім іншого класу) у фреймворку відсутні. По-друге, для еталона до порівняння додано не-ARX-шифр AES, оскільки саме з ним традиційно зіставляють програмну швидкодію ARX-конструкцій. По-третє, у FELICS використано RC5 з повними 20 раундами (RC5-32/20/16, далі RC5-20) — це розширена, рекомендована авторами версія, відмінна від редукованого RC5-32/16/7, що аналізувався в підрозд. 3.1; саме тому її показники ефективності помітно гірші.

Подивитися результати сценарія можна у таблиці **??**

Деталізовані показники за нейтрального зважування метрик (розмір

коду, оперативна пам'ять і час виконання на кожній з трьох платформ) наведено в Таблиці **??**.

Окремо зазначимо, що для шифру «Кипарис» зіставних із FELICS даних не існує: у самому фреймворку він відсутній, а єдине опубліковане дослідження його швидкодії [42] виконане за іншою методологією (пропускна здатність у режимі ECB на платформах Windows, Linux та Android, виміряна в Мбіт/с) і тому непорівнянне з вбудованими тактовими оцінками FELICS. Через це було прийнято рішення порівняти швидкодію даного шифру через існуюче порівняння з шифром AES на 3 платформах: x86 на 32-бітній архітектурі, x86 на 64-бітній архітектурі та на ARMCortex-A7, задля приблизного розуміння його швидкодії на практиці. Перевірка обчислюється на Мбіт/сек. в таблиці:

Платформа	**Блоковий шифр**		
	«Кипарис-256»	**«Кипарис-512»**	**AES-256**
Intel Core i3 / Windows 7 x32	1796,86	786,24	711,13
Intel Core i3 / Windows 7 x64	1878,5	2617,74	858,77
Intel Core i5 / Linux (64 bit)	3954,55	5395,81	1969,65
ARM Cortex-A7 / Android 4.2.2 (32 bit)	122	136	43

Таблиця 3.6 – Швидкодія шифрів «Кипарис» та AES, Мбіт/с

Отримані результати дозволяють кількісно підтвердити та уточнити якісні спостереження попередніх підрозділів. Speck демонструє найкращий показник FOM у групі (5.2 за нейтрального зважування), що узгоджується з його репутацією одного з найшвидших ARX-шифрів у програмній реалізації: мала кількість операцій і компактний код забезпечують перевагу на всіх трьох платформах. SPARX поступається Speck приблизно у 1,7 раза (FOM 8.8), і ця різниця є вимірюваною ціною доведених меж стійкості: додатковий лінійний шар та інтенсивніше використання ключового матеріалу збільшують і код, і час. Водночас SPARX усе ще випереджає AES, що ілюструє тезу підрозд. 3.2 — формальні гарантії стійкості коштують помірних, а не катастрофічних

втрат ефективності.

HIGHT із показником 14.8 виявляється повільнішим: його байтоорієнтована структура оптимізована під 8-бітові пристрої, тож на 16- та 32-бітових платформах численні байтові операції програють словноорієнтованим шифрам. AES як не-ARX-еталон (FOM 15.8) поступається і Speck, і SPARX у програмній реалізації без апаратного прискорення, що підтверджує програмну перевагу ARX-підходу на платформах без інструкцій AES-NI. Найгірший показник у групі має RC5-20 (FOM 20.8): його залежні від даних циклічні зсуви дорогі на мікроконтролерах без інструкцій змінного зсуву, а повні 20 раундів додають накладні витрати.

Перехід до зваженого профілю (час $\times 2$) зберігає загальний порядок ранжування, але посилює розрив: повільні RC5-20 (37.0) та HIGHT (25.1) просідають найсильніше, тоді як Speck і SPARX утримують лідерство. Це закономірно, адже ARX-шифри проєктувалися саме під високу програмну швидкодію, тож пріоритезація часу виконання працює на їхню користь.

Висновки до розділу 3

У цьому розділі було розглянуто блокові шифри, побудовані на чистому ARX-підході, тобто такі, що використовують виключно модульне додавання, побітове додавання та циклічні зсуви. Аналіз охопив як історичні конструкції, об'єднані в групу «До 2000 року» (FEAL, TEA, XTEA, XXTEA, RC5), так і сучасні легковагові сімейства (Speck, SPARX, HIGHT, «Кипарис») та шифр Threefish, що стоїть осібно як основа геш-функції Skein.

Зіставлення цих шифрів демонструє, що мінімалізм ARX є водночас його перевагою та джерелом ризику. Сама по собі трійка операцій забезпечує високу програмну швидкодію й компактність коду, проте стійкість визначається не цими операціями, а якістю розсіювання та ключового розкладу. Саме тут зосереджені вразливості ранніх

конструкцій: тривіальний розклад ключа TEA уможливив related-key атаки, а перенесення цієї проблеми в динамічний вибір підключа XTEA, хоч і усунуло початкову слабкість, відкрило шифр для атаки «посередині»; надповільне розсіювання Block TEA призвело до повного злому, лише частково виправленого в XXTEA. Кожен наступник у цьому сімействі латав конкретний недолік попередника, не змінюючи самої парадигми, що й пояснює витіснення всієї групи сучаснішими конструкціями.

Перехід до легковагових шифрів засвідчив зрілість ARX-підходу. Speck показав, що мала кількість операцій сумісна з високим запасом міцності: попри понад сімдесят опублікованих робіт, найкращі диференціальні атаки долають близько 70 % раундів і лише незначно випереджають повний перебір. Водночас стійкість Speck спирається на емпіричну відсутність ефективних атак, тоді як SPARX, використовуючи ту саму перестановку Speckey у межах стратегії довгого сліду, вперше для ARX-сімейства дозволяє математично довести верхні межі ймовірності диференціальних і лінійних слідів. HIGHT і «Кипарис» доповнюють картину прикладами стандартизованого байтоорієнтованого шифру з апаратною спрямованістю та національної постквантової фейстель-подібної конструкції відповідно. Threefish же ілюструє інший клас задач: твікований шифр із велетенськими блоками, підпорядкований не легковаговості, а ролі надійного будівельного блоку геш-функції, до того ж саме його MIX-функція стала структурним джерелом раунду Speck.

Емпіричний вимір на основі фреймворку FELICS кількісно підтвердив якісні спостереження. Найкращий показник Figure-of-Merit має Speck (5,2), що відповідає його репутації одного з найшвидших ARX-шифрів; SPARX поступається приблизно у 1,7 раза (8,8), і ця різниця є вимірюваною ціною доведених меж стійкості, а не втратою конкурентоспроможності, адже SPARX усе ще випереджає AES. Гірші показники HIGHT (14,8) та RC5-20 (20,8) закономірно пояснюються

байтовою орієнтацією першого й дорогими на мікроконтролерах залежними від даних зсувами другого. Для «Кипарису» зіставних із FELICS даних немає, тож його швидкодію оцінено непрямо через порівняння пропускної здатності з AES, де він демонструє суттєву перевагу.

Узагальнюючи, розглянуті шифри окреслюють еволюцію ARX-блокових шифрів від зламаних історичних зразків до стандартизованих легковагових примітивів. Наскрізним висновком є те, що безпека ARX-конструкції визначається передусім дизайном розсіювання та ключового розкладу, а не набором базових операцій, і що галузь поступово перейшла від суто емпіричних аргументів стійкості до конструкцій із формально доведеними межами, не пожертвувавши при цьому характерною для ARX програмною ефективністю.

ВИСНОВКИ

У межах кваліфікаційного дослідження виконано систематичний аналіз практичних криптосистем, побудованих на основі ARX–дизайну, з охопленням як архітектурних принципів побудови, так і криптоаналітичних характеристик розглянутих конструкцій. Поставлену мету — аналіз і систематизацію практичних ARX–криптосистем з оцінкою їхньої криптографічної стійкості та ефективності — досягнуто, а всі сформульовані завдання роботи виконано.

Сформульовано узагальнену модель ARX–дизайну та визначено роль трьох базових операцій — модульного додавання, циклічного зсуву і виключного АБО. Показано, що модульне додавання є єдиним джерелом нелінійності, тоді як циклічні зсуви й XOR забезпечують дифузію та лінійне змішування. ARX–підхід є програмно–орієнтованим і не потребує таблиць підстановок або операцій над скінченними полями, що забезпечує ефективність і портативність реалізацій на різних апаратних платформах.

Формалізовано поняття перестановки, ARX–перестановки та блокового шифру з ARX–структурою й обґрунтовано доцільність поділу ARX–примітивів на дві групи: криптографічні перестановки як універсальні будівельні блоки та блокові шифри як сімейства ключозалежних перестановок із власною структурою. Визначено єдиний набір критеріїв порівняльної оцінки — розмір ключа й блоку, кількість раундів, склад раундової функції, швидкодія та рівень криптографічної стійкості, — застосований надалі до всіх розглянутих конструкцій.

ARX–перестановки досліджено як самостійні примітиви та об'єднано у сімейства за спільною раундовою функцією і цільовою областю: Salsa20–ChaCha–Forro, BLAKE, SipRound і Chaskey–π, а також Alzette–SPARKLE. Продемонстровано, що та сама перестановка може слугувати ядром потокового шифру, компресійною функцією гешу, примітивом MAC чи внутрішньою перестановкою губчастої схеми.

Окрему увагу приділено механізмам, що усувають структурні обмеження ізольованої перестановки: фінальному пословному додаванню стану в Salsa20/ChaCha, розширенню нонса до 192 біт через HChaCha20 у XChaCha20, поєднанню ChaCha20 з автентифікатором Poly1305 у AEAD–схему ChaCha20-Poly1305 (RFC 8439), HAIFA–режиму BLAKE2b, деревоподібній схемі BLAKE3 та дуплексному й губчастому режимам SPARKLE, що лежать в основі SCHWAEMM і ESCH. Показано також, що один ARX–бокс Alzette породжує як перестановку SPARKLE, так і блокові шифри CRAX і TRAX, що ілюструє можливість побудови цілого сімейства примітивів навколо однієї базової ARX–конструкції.

Окремо описано та систематизовано блокові ARX–шифри. Історичні конструкції до 2000 року (FEAL, TEA, XTEA, XXTEA, RC5) розглянуто як еволюцію від зламаних схем до усвідомлення вимог до розкладу ключа і швидкості розсіювання. Сучасний сегмент представлено сімейством Speck і SPARX, корейським легковаговим стандартом HIGHT, українським постквантовим шифром «Кипарис» та великоблоковим Threefish, що є ядром геш–функції Skein. Порівняльний аналіз за визначеними критеріями доповнено емпіричним виміром програмної ефективності за фреймворком FELICS: найкращий показник демонструє Speck, тоді як SPARX ціною помірних втрат продуктивності отримує доведені межі стійкості за стратегією довгого сліду, а «Кипарис» за пропускною здатністю випереджає AES на розглянутих платформах.

Для всіх розглянутих конструкцій проведено криптоаналітичне дослідження. Для Salsa20 та ChaCha розібрано атаки на скорочені версії методами *probabilistic neutral bits* (PNB) і диференціально–лінійного аналізу: найкраща відома атака на ChaCha7 має складність $2^{221.95}$ [44] (а подальші роботи знизили її до порядку 2^{207}), тоді як повна 20–раундова версія залишається криптографічно стійкою. Для BLAKE2 та Skein/Threefish розглянуто ротаційний і диференціальний аналіз, для Alzette — доведені точні межі диференціального та лінійного трейлів. Серед блокових шифрів продемонстровано related–key слабкість TEA,

атаку «посередині» на XTEA, повний злам XXTEA, диференціальні атаки приблизно на 70% раундів Speck та повнораундову стійкість HIGHT і «Кипарис». Спільним для всіх повнораундових конструкцій є те, що відомі практичні результати стосуються лише зменшеної кількості раундів, що свідчить про суттєвий запас стійкості сучасних ARX–примітивів.

У результаті дослідження продемонстровано, що ARX–дизайн є універсальним підходом у сучасній симетричній криптографії, який поєднує доведену практичну ефективність із достатнім рівнем криптографічної стійкості до відомих методів аналізу. Здатність однієї базової ARX–конструкції породжувати ціле сімейство примітивів робить цей напрям одним із ключових у проєктуванні легковагових симетричних алгоритмів.

ПЕРЕЛІК ПОСИЛАНЬ

1. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, L. Wingers, *Simon and Speck FAmilies of Lightweight Block Ciphers*, National Security Agency, 2013 [Електронний ресурс]. — https://eprint.iacr.org/2013/404.pdf.

2. D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B.-S. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, S. Chee, *HIGHT: A New Block Cipher Suitable for Low-Resource Device*, Cryptographic Hardware and Embedded Systems (CHES 2006), Lecture Notes in Computer Science, vol. 4249, pp. 46–59, Springer, 2006. [Електронний ресурс]. — https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2014-2015/Paper%20Block%20Cipher%201.pdf.

3. C. Beierle, A. Biryukov, L. Cardoso dos Santos, J. Großschädl, L. Perrin, A. Udovenko, V. Velichkov, Q. Wang, *Lightweight AEAD and Hashing using the Sparkle Permutation Family*, IACR Transactions on Symmetric Cryptology, vol. 2020, no. S1, pp. 208–261, 2020. [Електронний ресурс]. — https://eprint.iacr.org/2020/412.pdf.

4. D. J. Bernstein, *ChaCha, a Variant of Salsa20*, Workshop Record of SASC, 2008. [Електронний ресурс]. — https://cr.yp.to/chacha/chacha-20080128.pdf.

5. D. J. Bernstein, *The Salsa20 Family of Stream Ciphers*, Lecture Notes in Computer Science, vol.4986, Springer, 2008. [Електронний ресурс]. — https://cr.yp.to/snuffle/salsafamily-20071225.pdf.

6. M. Coutinho, I. Passos, J. C. Grados Vásquez, F. L. L. de Mendonça, R. Timteo de Sousa Jr., F. Borges *Latin Dances Reloaded: Improved Cryptanalysis Against Salsa and ChaCha, and the Proposal of Forró*, Part of the book series: Lecture Notes in Computer Science ((LNCS,volume 13791)) [Електронний ресурс]. — https://link.springer.com/chapter/10.1007/978-3-031-22963-3_9#Sec22x.

7. D. Kumar, S. Patranabis, J. Breier, D. Mukhopadhyay, S. Bhasin, A. Chattopadhyay, A. Bakshi, *A Practical Fault Attack on ARX-like Ciphers with a Case Study on ChaCha20*, Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), 2017. [Електронний ресурс]. — https://dr.ntu.edu.sg/server/api/core/bitstreams/452d2e1a-9b26-4630-b5cf-de6cb23f5872/content.

8. N. Mouha, V. Velichkov, C. De Cannière, B. Preneel, *The Differential Analysis of S-Functions*, Selected Areas in Cryptography (SAC 2010), Lecture Notes in Computer Science, vol. 6544, pp. 36–56, Springer, 2010. [Електронний ресурс]. — https://mouha.be/wp-content/uploads/sfunction.pdf.

9. M. Coutinho, T. C. Souza Neto, *Improved Linear Approximations to ARX Ciphers and Attacks on ChaCha*, Advances in Cryptology – EUROCRYPT 2021, Lecture Notes in Computer Science, vol. 12696, pp. 711–740, Springer, 2021. [Електронний ресурс]. — https://eprint.iacr.org/2020/1350.pdf.

10. J.-P. Aumasson, S. Fischer, S. Khazaei, W. Meier, C. Rechberger, *New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba*, Fast Software Encryption (FSE 2008), Lecture Notes in Computer Science, vol. 5086, pp. 470–488, Springer, 2008. [Електронний ресурс]. — https://eprint.iacr.org/2007/472.pdf.

11. D. Khovratovich, I. Nikolić, C. Rechberger, *Rotational Rebound Attacks on Reduced Skein*, Advances in Cryptology – ASIACRYPT 2010, Lecture Notes in Computer Science, vol. 6477, pp. 1–19, Springer, 2010. [Електронний ресурс]. — https://eprint.iacr.org/2010/538.pdf.

12. G. Leurent, *Improved Differential-Linear Cryptanalysis of 7-round Chaskey with Partitioning*, Advances in Cryptology – EUROCRYPT 2016, Lecture Notes in Computer Science, vol. 9665, pp. 344–371, Springer, 2016. [Електронний ресурс]. — https://eprint.iacr.org/2015/968.pdf.

13. C. Mavromati, *Key-recovery Attacks against the MAC Algorithm Chaskey*, IACR Cryptology ePrint Archive, Report 2015/811, 2015. [Електронний ресурс]. — https://eprint.iacr.org/2015/811.pdf.

14. M. Coutinho, I. Passos, J. C. Grados Vásquez, F. L. L. de Mendonça,

R. T. de Sousa Jr., F. Borges, *Latin Dances Reloaded: Improved Cryptanalysis Against Salsa and ChaCha, and the Proposal of Forró*, Advances in Cryptology – ASIACRYPT 2022, Lecture Notes in Computer Science, vol. 13792, pp. 256–286, Springer, 2022. [Електронний ресурс]. — https://eprint.iacr.org/2022/185.pdf.

15. J.-P. Aumasson, L. Henzen, W. Meier, R. C.-W. Phan, *SHA-3 Proposal BLAKE*, Submission to the NIST SHA-3 Competition, Version 2.0, 2012. [Електронний ресурс]. — https://www.aumasson.jp/blake/blake.pdf.

16. J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, C. Winnerlein, *BLAKE2: Simpler, Smaller, Fast as MD5*, Applied Cryptography and Network Security (ACNS 2013), Lecture Notes in Computer Science, vol. 7954, pp. 119–135, Springer, 2013. [Електронний ресурс]. — https://eprint.iacr.org/2013/322.pdf.

17. A. Biryukov, I. Nikolić, A. Roy, *Boomerang Attacks on BLAKE-32*, Fast Software Encryption (FSE 2011), Lecture Notes in Computer Science, vol. 6733, pp. 218–237, Springer, 2011. [Електронний ресурс]. — https://www.researchgate.net/publication/220942355_Boomerang_Attacks_on_BLAKE-32.

18. Y. Hao, *The Boomerang Attacks on BLAKE and BLAKE2*, Information Security and Cryptology (Inscrypt 2014), Lecture Notes in Computer Science, vol. 8957, pp. 286–310, Springer, 2015. [Електронний ресурс]. — https://eprint.iacr.org/2014/1012.pdf.

19. J. Guo, P. Karpman, I. Nikolić, L. Wang, S. Wu, *Analysis of BLAKE2*, Topics in Cryptology – CT-RSA 2014, Lecture Notes in Computer Science, vol. 8366, pp. 402–423, Springer, 2014. [Електронний ресурс]. — https://eprint.iacr.org/2013/467.pdf.

20. J.-P. Aumasson, G. Leurent, W. Meier, F. Mendel, N. Mouha, R. C.-W. Phan, Y. Sasaki, P. Susil, *Tuple Cryptanalysis of ARX with Application to BLAKE and Skein*, ECRYPT II Hash Workshop, 2011. [Електронний ресурс]. — https://www.aumasson.jp/blake/.

21. N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno,

J. Callas, J. Walker, *The Skein Hash Function Family*, Submission to the NIST SHA-3 Competition, Version 1.3, 2010. [Електронний ресурс]. — https://www.schneier.com/wp-content/uploads/2015/01/skein.pdf.

22. J.-P. Aumasson, D. J. Bernstein, *SipHash: A Fast Short-Input PRF*, Progress in Cryptology – INDOCRYPT 2012, Lecture Notes in Computer Science, vol. 7668, pp. 489–508, Springer, 2012. [Електронний ресурс]. — https://eprint.iacr.org/2012/351.pdf.

23. N. Mouha, B. Mennink, A. Van Herrewege, D. Watanabe, B. Preneel, I. Verbauwhede, *Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers*, Selected Areas in Cryptography (SAC 2014), Lecture Notes in Computer Science, vol. 8781, pp. 306–323, Springer, 2014. [Електронний ресурс]. — https://eprint.iacr.org/2014/386.pdf.

24. C. Beierle, A. Biryukov, L. C. dos Santos, J. Großschädl, L. Perrin, A. Udovenko, V. Velichkov, Q. Wang, *Alzette: A 64-Bit ARX-box*, Advances in Cryptology – CRYPTO 2020, Lecture Notes in Computer Science, vol. 12172, pp. 419–448, Springer, 2020. [Електронний ресурс]. — https://eprint.iacr.org/2019/1378.pdf.

25. H. Lipmaa, S. Moriai, *Efficient Algorithms for Computing Differential Properties of Addition*, Fast Software Encryption (FSE 2001), Lecture Notes in Computer Science, vol. 2355, pp. 336–350, Springer, 2001. [Електронний ресурс]. — https://eprint.iacr.org/2001/001.pdf.

26. D. Khovratovich, I. Nikolić, *Rotational Cryptanalysis of ARX*, Fast Software Encryption (FSE 2010), Lecture Notes in Computer Science, vol. 6147, pp. 333–346, Springer, 2010. [Електронний ресурс]. — https://www.researchgate.net/publication/225958853_Rotational_Cryptanalysis_of_ARX.

27. T. Ashur, Y. Liu, *Rotational Cryptanalysis in the Presence of Constants*, IACR Transactions on Symmetric Cryptology, vol. 2016, no. 1, pp. 57–77, 2016. [Електронний ресурс]. — https://eprint.iacr.org/2016/826.pdf.

28. Y. Nir, A. Langley, *ChaCha20 and Poly1305 for IETF Protocols*,

RFC 8439, Internet Engineering Task Force, 2018. [Електронний ресурс]. — https://www.rfc-editor.org/rfc/rfc8439.

29. M.-J. Saarinen, J.-P. Aumasson, *The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC)*, RFC 7693, Internet Engineering Task Force, 2015. [Електронний ресурс]. — https://www.rfc-editor.org/rfc/rfc7693.

30. J. O'Connor, J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, *BLAKE3: One Function, Fast Everywhere*, BLAKE3 Specification, 2020. [Електронний ресурс]. — https://github.com/BLAKE3-team/BLAKE3-specs/blob/master/blake3.pdf.

31. S. Arciszewski, *XChaCha: eXtended-nonce ChaCha and AEAD_XChaCha20_Poly1305*, IETF Internet-Draft draft-irtf-cfrg-xchacha-03, 2020 (expired). [Електронний ресурс]. — https://datatracker.ietf.org/doc/draft-irtf-cfrg-xchacha/.

32. D. J. Bernstein, *Extending the Salsa20 Nonce*, Workshop on Symmetric Key Encryption (SKEW 2011), 2011. [Електронний ресурс]. — https://cr.yp.to/snuffle/xsalsa-20110204.pdf.

33. Akihiro Shimizu, Shoj i. Miyaguchi *FEAL: Fast Data Encipherment Algorithm*, Proceedings of the 1987 Symposium on Security and Privacy, pp. 107–114, 1987. [Електронний ресурс]. — https://link.springer.com/chapter/10.1007/3-540-39118-5_24.

34. D. Wheeler, R. A. Nidham, *TEA: Tiny Encryption Algorithm*, IACR Cryptology ePrint Archive, Report 1994/033, 1994. [Електронний ресурс]. — https://www.cl.cam.ac.uk/ftp/papers/djw-rmn/djw-rmn-tea.html.

35. John Kelsey, Bruce Schneir,David Wagner *Related-key cryptanalysis of 3-WAY, Biham-DES,CAST, DES-X, NewDES, RC2, and TEA* Information and Communications Security (ICICS 1997),2005 [Електронний ресурс]. — https://link.springer.com/chapter/10.1007/BFb0028479.

36. D. Wheeler, R. A. Nidham, *Tea extensions*, [Електронний ресурс]. — https://www.cl.cam.ac.uk/ftp/papers/djw-rmn/djw-rmn-tea.html.

37. Gautham Sekar, Nicky Mouha, Vesselin Velichkov, Bart Preneel *Meet-*

in-the-Middle Attacks on Reduced-Round XTEA, [Електронний ресурс]. — https://mouha.be/wp-content/uploads/mitm-xtea.pdf.

38. Markku-Juhani Saarinen *Cryptanalysis of block tea*, Newsgroup sci.crypt.research [Електронний ресурс]. — https://www.researchgate.net/publication/228548968_Cryptanalysis_of_block_tea.

39. D. Wheeler, R. A. Nidham, *Correction to xtea*, [Електронний ресурс]. — https://www.movable-type.co.uk/scripts/xxtea.pdf.

40. E. Yarrkov *Cryptanalysis of XXTEA*, IACR Cryptology ePrint Archive, Report 2010/254, 2010. [Електронний ресурс]. — https://eprint.iacr.org/2010/254.pdf.

41. R. Rivest *The RC5 Encryption Algorithm*, MIT Laboratory for Computer Science. 1997 [Електронний ресурс]. — https://web.archive.org/web/20070417135716/http://theory.lcs.mit.edu/~rivest/Rivest-rc5rev.pdf.

42. D. Dinu1, L. Perrin1, A. Udovenko1, V. Velichkov1,J. Großschädl1, A. Biryukov *Design Strategies for ARX with Provable Bounds: Sparx and LAX*, IACR Cryptology ePrint Archive, Report 2016/984, 2016. [Електронний ресурс]. — https://eprint.iacr.org/2016/984.pdf.

43. D. Dinu, Y. Le Corre, D. Khovratovich, L. Perrin, J. Großschädl, A. Biryukov, *Triathlon of Lightweight Block Ciphers for the Internet of Things*, IACR Cryptology ePrint Archive, Report 2015/209, 2015. [Електронний ресурс]. — https://eprint.iacr.org/2015/209.pdf.

44. S. Dey, H. K. Garai, S. Sarkar, N. K. Sharma, *Revamped Differential-Linear Cryptanalysis on Reduced Round ChaCha*, Advances in Cryptology – EUROCRYPT 2022, Lecture Notes in Computer Science, vol. 13277, pp. 351–380, Springer, 2022. [Електронний ресурс]. — https://eprint.iacr.org/2022/536.pdf.

45. Родінко М., Олійников Р. Постквантовий малоресурсний симетричний блоковий шифр «Кипарис» [Електронний ресурс] // Радіотехніка: Всеукр. міжвід. наук.-техн. зб. — 2017. — Вип. 189. — С. 100–107. — Режим доступу: http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?I21DBN=LINK&P21DBN=UJRN&Z21ID=&S21REF=10&

S21CNR=20&S21STN=1&S21FMT=ASP_meta&C21COM=S&2_S21P03=FILA=&2_S21STR=rvmnts_2017_189_13

46. M. Rodinko, R. Oliynykov, K. Yubuzova, *Differential Cryptanalysis of the Lightweight Block Cipher Cypress-256*, International Journal of Computing, vol. 19, no. 2, pp. 273–281, 2020. [Електронний ресурс]. — https://doi.org/10.47839/ijc.19.2.1771.

47. G. Procter, *A Security Analysis of the Composition of ChaCha20 and Poly1305*, IACR Cryptology ePrint Archive, Report 2014/613, 2014. [Електронний ресурс]. — https://eprint.iacr.org/2014/613.pdf.

ДОДАТОК А ВЕЛИКІ РИСУНКИ ТА ТАБЛИЦІ

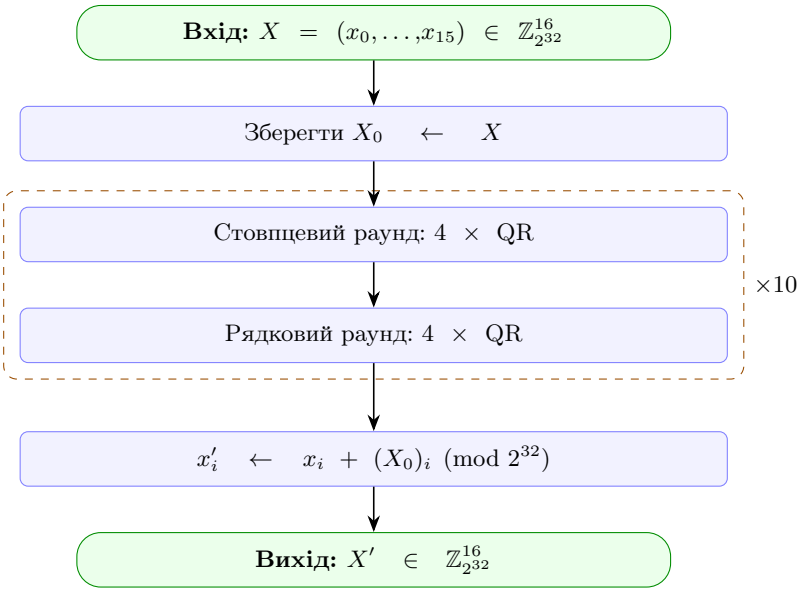

Рисунок А.1 – Блок-схема перестановки Salsa20

Шифр	**Блок** [біт]	**Ключ** [біт]	**Раундів**	$\oplus$ / **раунд**	$\lll,\ggg$ / **раунд**	$+$ / **раунд**	**Рівень безпеки**
Speck32/64	32	64	22	2	2	1	15/22 (68%)
Speck64/96	64	96	26	2	2	1	19/26 (73%)
Speck64/128	64	128	27	2	2	1	20/27 (74%)
SPARX-64/128	64	128	24	6	2	2	15/24 (62%)
SPARX-128/128	128	128	32	6	2	2	22/32 (69%)
SPARX-128/256	128	256	40	6	2	2	24/40 (60%)
HIGHT	64	128	32	12	12	4	27/32 (84%)
Кипарис-256	256	256	10	16	8	8	6/10 (60%)
Кипарис-512	512	512	14	16	8	8	6/10 (60%)

Таблиця А.1 – Структурне порівняння ARX-шифрів Speck, SPARX, HIGHT та Кипарис

Тут «*» позначає реалізації мовою асемблера; решта даних отримана з реалізацій мовою C.

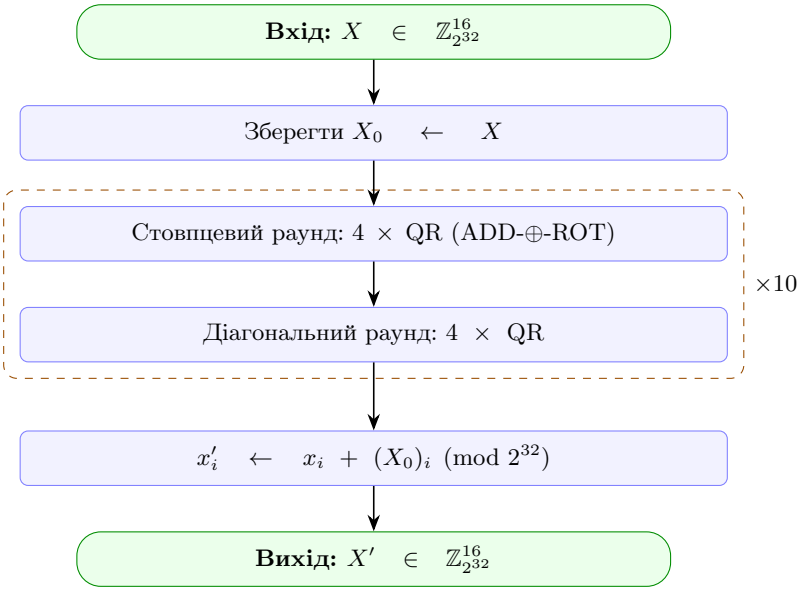

Рисунок А.2 – Блок-схема перестановки ChaCha

Шифр	**Блок**	**Ключ**	**FOM (рівні ваги)**	**FOM (час ×2)**
Speck-64/128	64	128	5.2	7.8
SPARX-64/128	64	128	8.8	13.4
HIGHT-64/128	64	128	14.8	25.1
AES-128/128	128	128	15.8	23.6
RC5-20	64	128	20.8	37.0

Таблиця А.2 – Таблиця 3.6 – Підсумковий показник FOM (Сценарій 1, менше – краще)

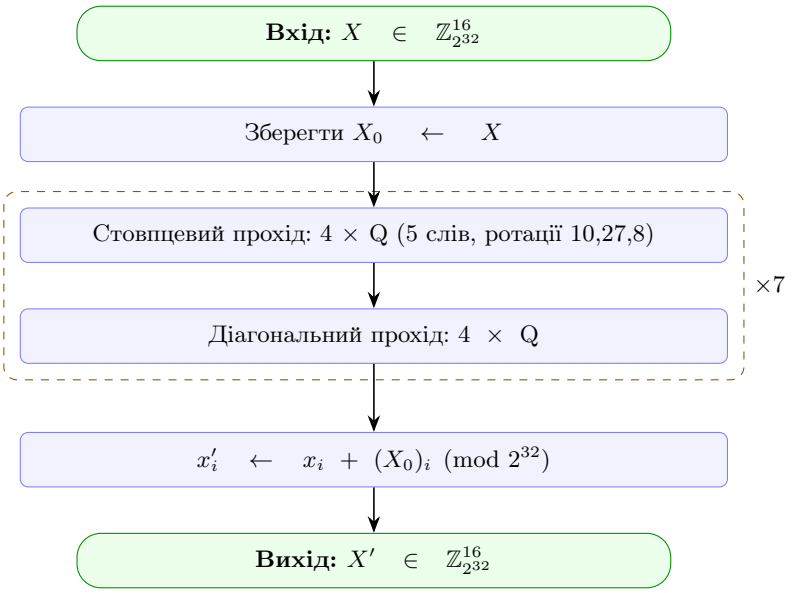

Рисунок А.3 – Блок-схема перестановки Forró

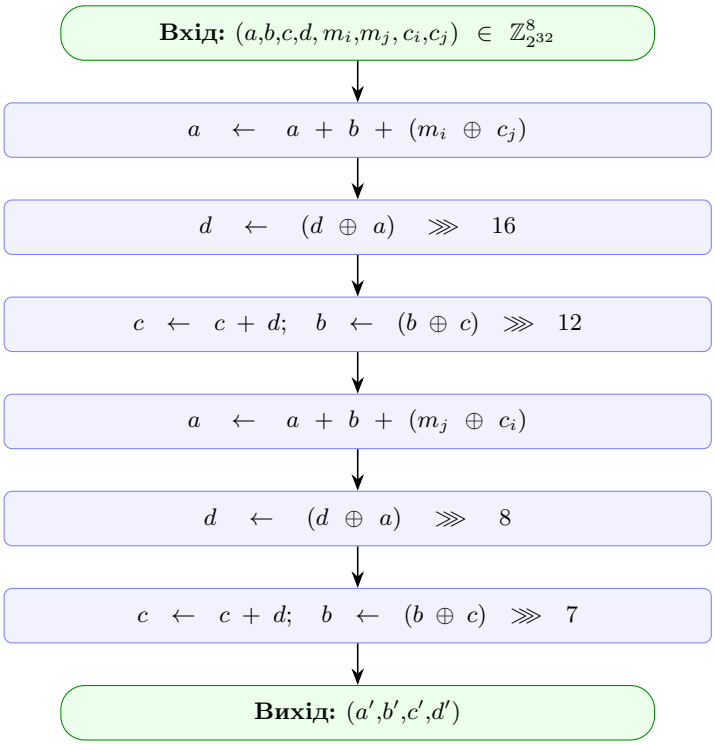

Рисунок А.4 – Блок-схема G-функції BLAKE-256

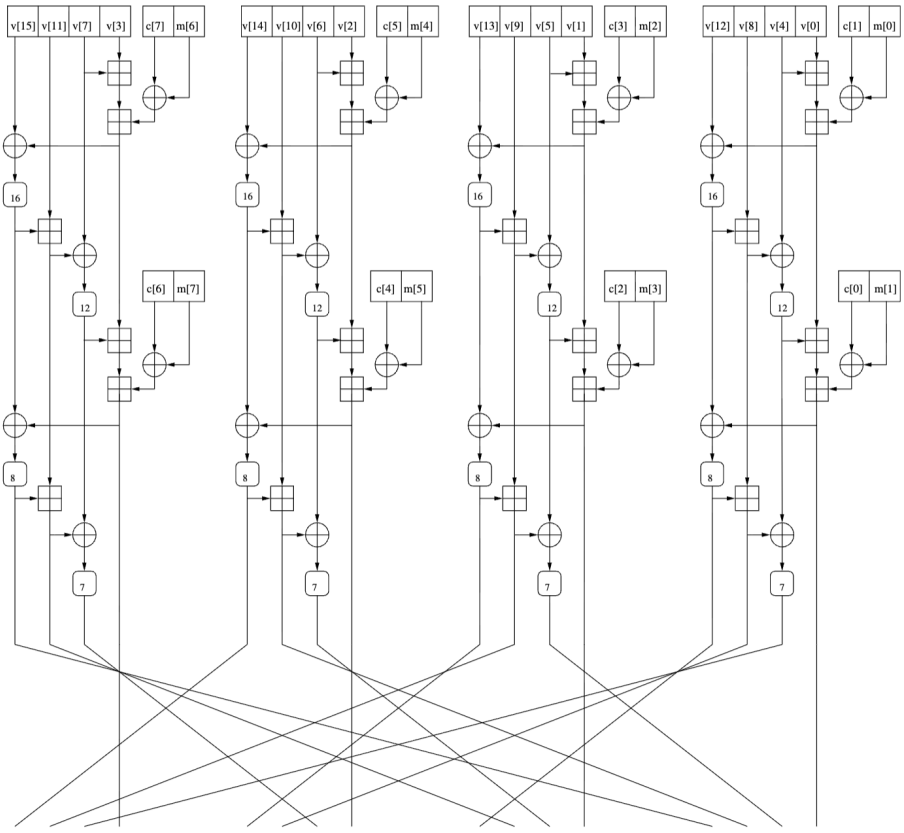

Рисунок А.5 – Раунд BLAKE-256

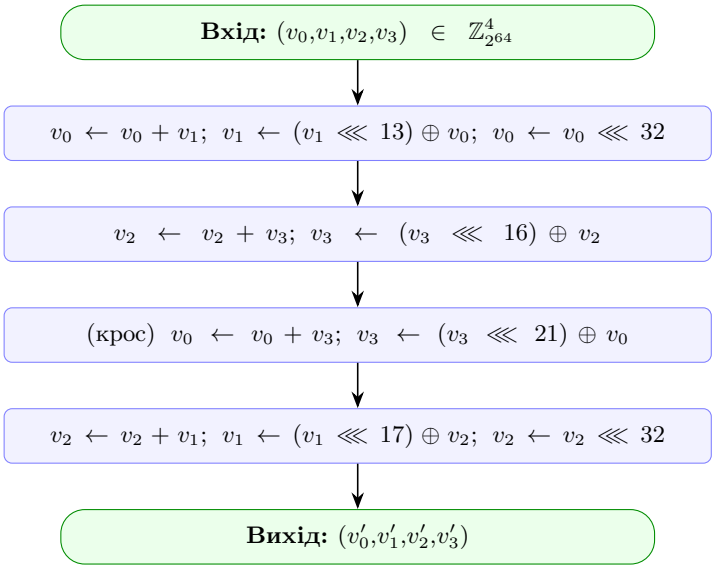

Рисунок А.6 – Блок-схема SipRound

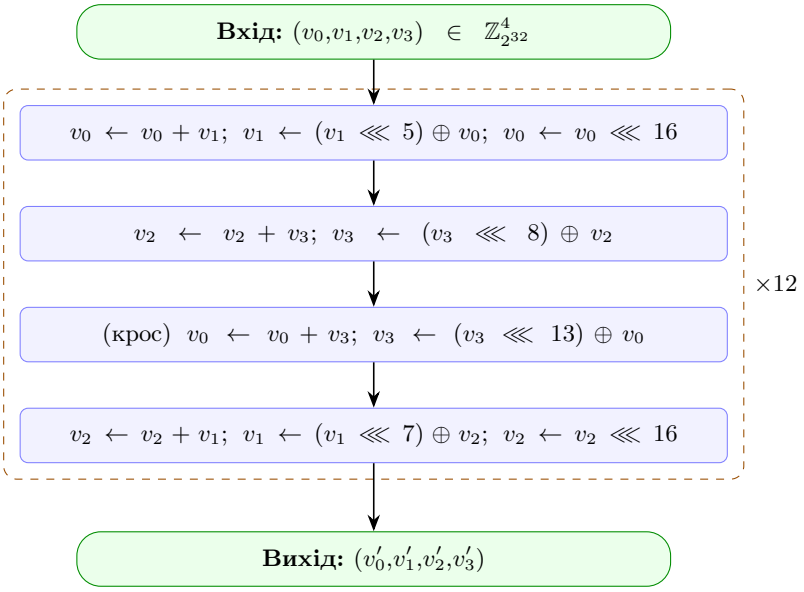

Рисунок А.7 – Блок-схема Chaskey π (12 раундів)

Шифр	**Блок** [б]	**Ключ** [б]	**AVR** Код [Б]	**AVR** RAM [Б]	**AVR** Час [такти]	**MSP** Код [Б]	**MSP** RAM [Б]	**MSP** Час [такти]	**ARM** Код [Б]	**ARM** RAM [Б]	**ARM** Час [такти]	**FOM**
Speck-64/128	64	128	874*	302*	44895*	572*	296*	32333*	444	308	16505	5.2
SPARX-64/128	64	128	1198*	392*	65539*	966*	392*	36766*	1200*	424*	40887*	8.8
HIGHT-64/128	64	128	1414*	333*	94557*	1238*	328*	120716*	1444*	380*	90385*	14.8
AES-128/128	128	128	3010*	408*	58246*	2684*	408*	86506*	3050*	452*	73868*	15.8
RC5-20	64	128	3706	368	252368	1240	378	386026	624	376	36473	20.8

Таблиця А.3 – Таблиця 3.7 – Деталізовані результати FELICS (Сценарій 1, нейтральне зважування)

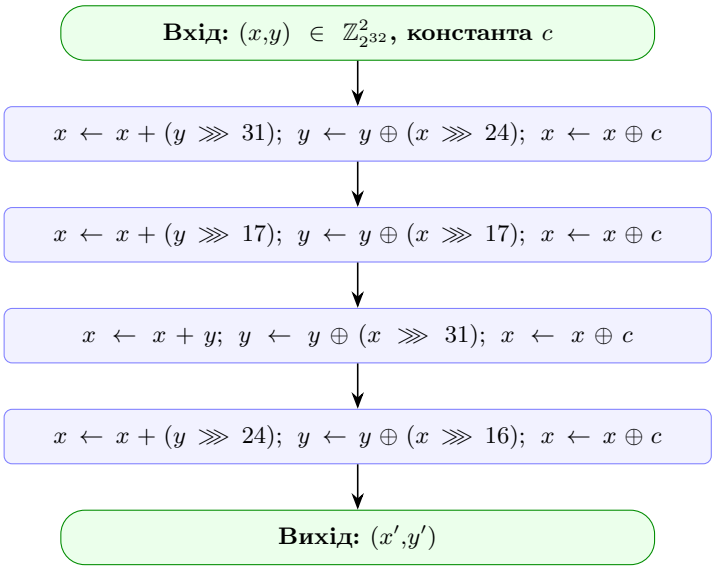

Рисунок А.8 – Блок-схема ARX-боксу Alzette

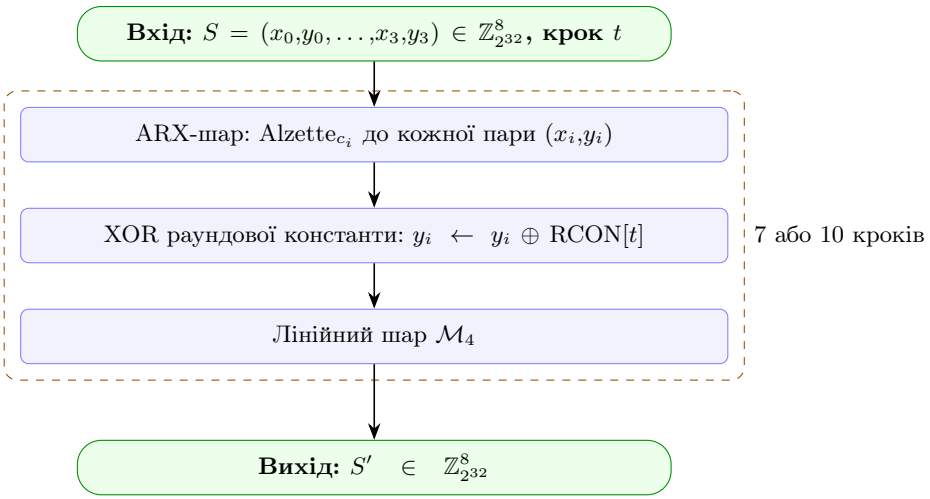

Рисунок А.9 – Блок-схема одного кроку SPARKLE256

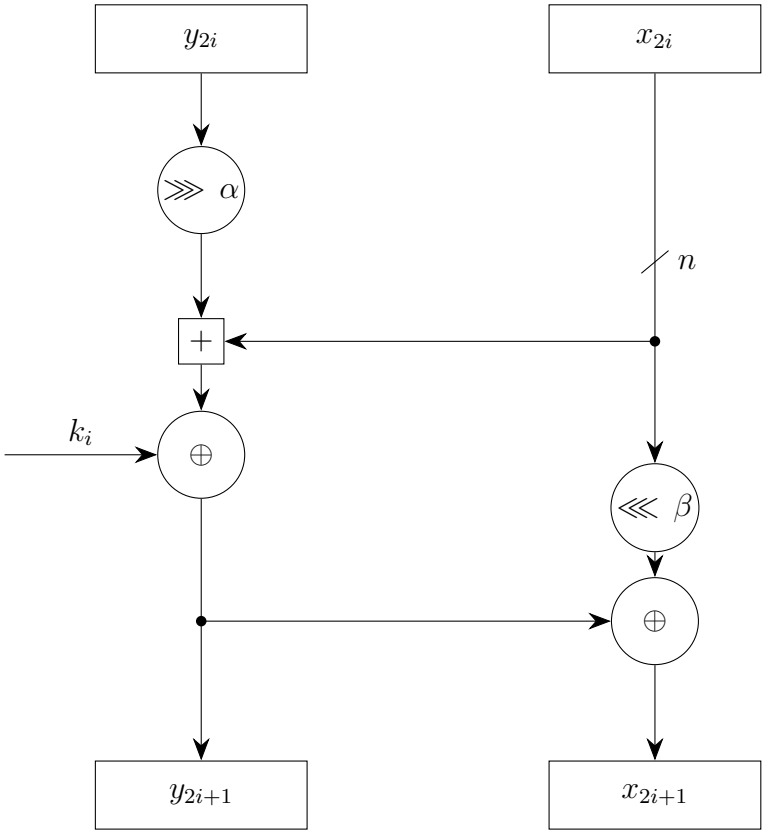

Рисунок А.10 – Раундова функція SPECK; (y_{2i}, x_{2i}) позначає підшифр після i кроків шифрування.

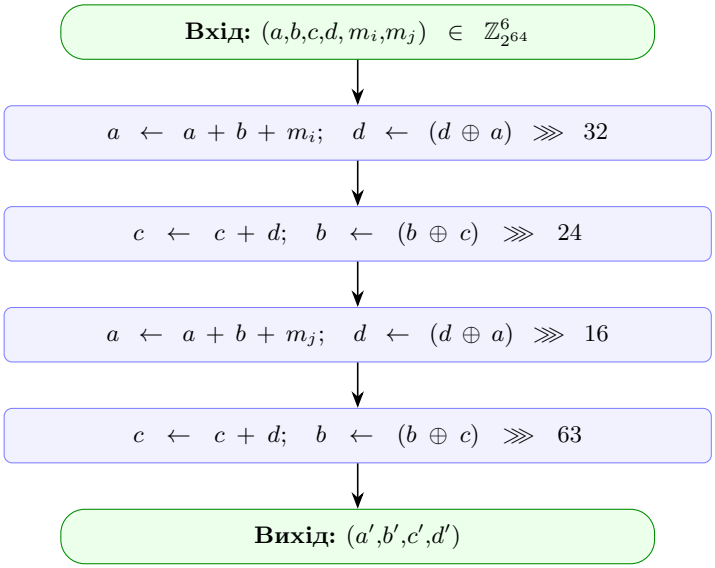

Рисунок А.11 – Блок-схема G-функції BLAKE2b

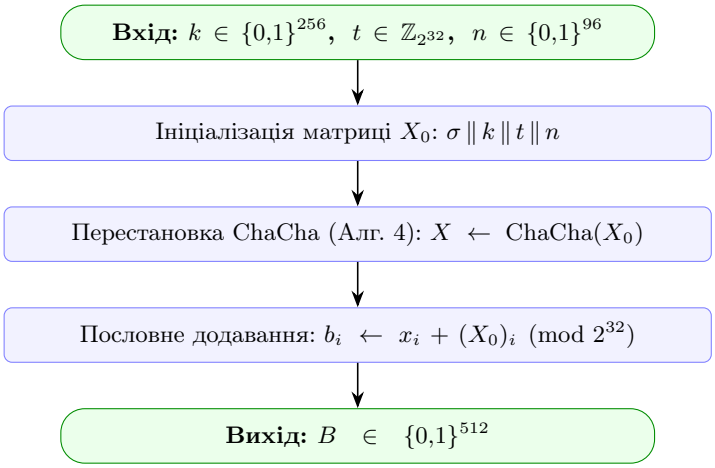

Рисунок А.12 – Блок-схема генерації блоку ChaCha20

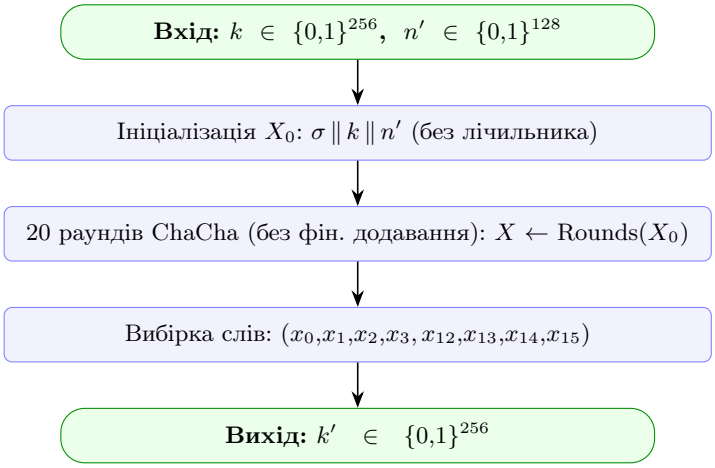

Рисунок А.13 – Блок-схема HChaCha20

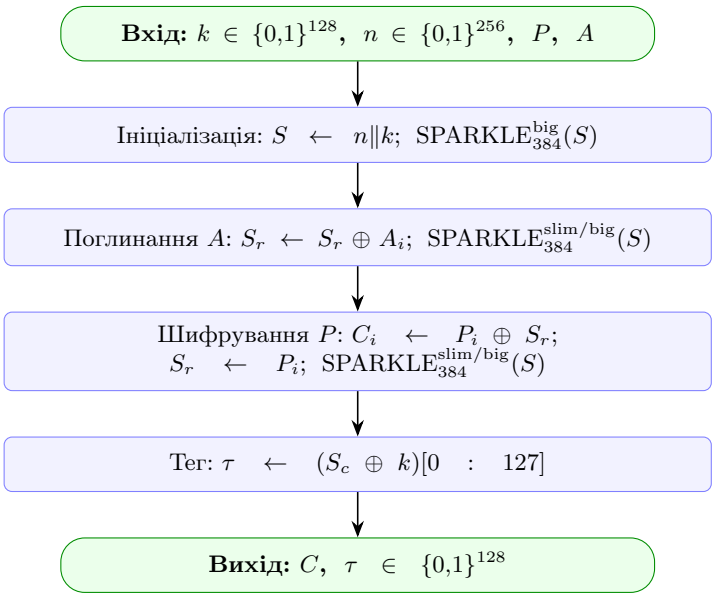

Рисунок А.14 – Блок-схема SCHWAEMM256-128_Encrypt

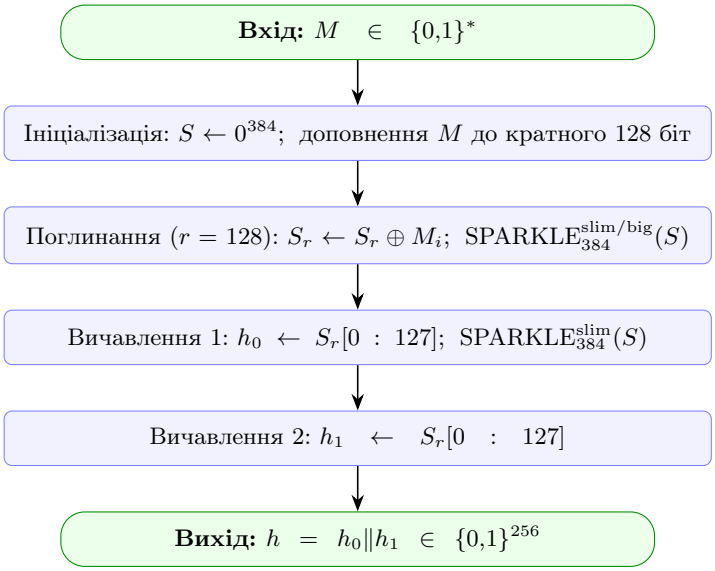

Рисунок А.15 – Блок-схема ESCH256_Hash

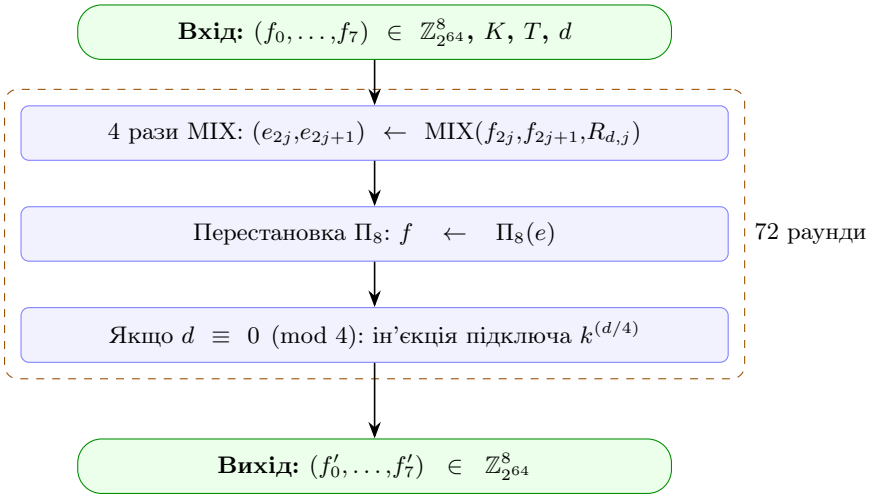

Рисунок А.16 – Блок-схема раунду Threefish-512