DOI 10.31891/2307-5732-2023-325-5-269-278
УДК 004.056.55

ЩУР НАТАЛІЯ
Державний університет «Житомирська політехніка»
https://orcid.org/0000-0002-1182-4799
e-mail: thalitana@gmail.com
ПОКОТИЛО ОЛЕКСАНДРА
Державний університет «Житомирська політехніка»
https://orcid.org/0000-0002-1587-235X
e-mail: kik_poa@ztu.edu.ua
БАЙЛЮК ЄЛІЗАВЕТА
Державний університет «Житомирська політехніка»
https://orcid.org/0000-0002-4961-7816
e-mail: liza.bailiuk@gmail.com

ОГЛЯД ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ АЛГОРИТМІВ-ФІНАЛІСТІВ КОНКУРСУ NIST ЗІ СТАНДАРТИЗАЦІЇ ЛЕГКОВАГОВОЇ КРИПТОГРАФІЇ

Статтю присвячено процесу стандартизації легковагових криптографічних алгоритмів, включаючи важливі етапи від відбору кандидатів до оголошення остаточного переможця. У статті здійснено огляд та порівняльний аналіз алгоритмів-фіналістів конкурсу NIST зі стандартизації легковагової криптографії. Розглянуто типові криптографічні конструкції та примітиви, на базі яких реалізовані перетворення у алгоритмах-фіналістах. Проведено дослідження специфікацій алгоритмів-фіналістів, проаналізовано їх можливі варіанти та модифікації. Представлено порівняння алгоритмів за різними характеристиками (розміри ключа, блоку, тегу тощо), наведено показники безпеки кожного алгоритму. Стаття має на меті дати загальне уявлення про дизайн та властивості алгоритмів-фіналістів конкурсу NIST, а також допомогти з вибором оптимального алгоритму для конкретного сценарію застосування.

Ключові слова: легковагова криптографія, стандартизація, автентифіковане шифрування, хешування.

SHCHUR NATALIIA, POKOTYLO OLEKSANDRA, BAILIUK YELYZAVETA
Zhytomyr Polytechnic National University

OVERVIEW AND COMPARATIVE ANALYSIS OF NIST COMPETITION FINALIST ALGORITHMS FOR LIGHTWEIGHT CRYPTOGRAPHY STANDARDIZATION

The shift from desktop computers to small devices brings a wide range of new security and privacy concerns. In many conventional cryptographic standards, the trade off between security, performance and resource requirements was optimized for desktop and server environments. As a result, implementing the current cryptography standards in resource-constrained devices becomes challenging due to the inherent limitations of such devices. Lightweight cryptography is an effective tool for the development of Internet of Things technologies, medical equipment, automotive electronics, and various other fields where balancing performance and security is crucial However, choosing the optimal cryptographic algorithm for lightweight cryptography remains a challenging task. That is why the main goal of the NIST competition is to select such lightweight cryptographic algorithms that would have low computational complexity, minimal memory usage, and low energy consumption, while providing a reliable level of protection for various applications.

Typical cryptographic constructions and primitives upon which transformations in finalist algorithms are built are discussed. Specifications of finalist algorithms are examined, and their potential variants and modifications are analyzed. A comparison of algorithms based on various characteristics (key size, block size, tags, etc.) is presented, security indicators of each algorithm are given. The article aims to provide a general understanding of the design and properties of NIST competition finalist algorithms, aiding in the selection of the optimal algorithm for specific usage scenarios.

Keywords: lightweight cryptography, standardization, authenticated encryption, hashing.

Постановка проблеми

Зі збільшенням використання малопотужних пристроїв у різних сферах, таких як охорона здоров'я, бездротові сенсорні мережі, Інтернет речей (Internet of Things, IoT) тощо, потреба в безпеці та належному захисті даних у таких пристроях також зростає. Незважаючи на те, що існує велика кількість криптографічних алгоритмів для безпечного шифрування даних, вони в основному розроблені для більш традиційного використання на ПК, і вони не тільки споживають багато енергії, але також вимагають багато процесорної потужності та фізичного простору.

Легковагова криптографія (lightweight cryptography, LWC) є галуззю криптографії, яка займається розробкою та застосуванням криптографічних алгоритмів для ефективної роботи на пристроях з обмеженими обчислювальними ресурсами. Ці алгоритми зазвичай називають «полегшеними» алгоритмами криптографії. «Легкі» не означає, що вони не є криптостійкими, а швидше те, що вони повинні бути ефективними та вимагати мінімальних ресурсів для своєї реалізації та виконання.

Розробка та стандартизація легковагових криптографічних алгоритмів є постійним процесом, оскільки швидкість розвитку технологій та вимоги безпеки змінюються з часом. Тому багато вчених та дослідників продовжують працювати над вдосконаленням і розвитком нових «полегшених» алгоритмів, які задовольняють сучасним вимогам безпеки та ефективності. Найчастіше, розробники LWC змушені вибирати між трьома, часом взаємовиключними, вимогами до алгоритмів: безпекою, вартістю і продуктивністю.

На практиці не складає труднощів оптимізувати будь-які два параметра: безпеку і вартість, безпеку і

продуктивність або вартість і продуктивність, однак дуже важко поєднати всі три одночасно. Наприклад, для блокових шифрів довжина ключа забезпечує компроміс між безпекою та вартістю, кількість раундів забезпечує компроміс між безпекою та продуктивністю, а апаратна архітектура – між ціною та продуктивністю. Розробка легковагових криптографічних алгоритмів вимагає балансування цих компромісів, щоб забезпечити належний рівень безпеки та ефективності в умовах обмежених ресурсів.

Аналіз останніх джерел

Проектуванням, розробкою та дослідженням алгоритмів легковагової криптографії займається велика кількість науковців і криптографів, серед яких C. Dobraunig, M. Eichlseder, F. Mendel, M. Schläffer, J. Daemen, G. Bertoni, D. Bernstein, T. Lange, A. Bogdanov та багато інших. Серед українських вчених різні питання створення та використання легковагових алгоритмів для захисту систем з обмеженими ресурсами розглядали такі дослідники як Я. Совин, Ю. Наконечний, І. Опірський, М. Стахів, В. Семеренко, В. Дудикевич, І. Собчук, Л. Ракобовчук, М. Родінко та інші.

Висвітленню процесу стандартизації легковагової криптографії присвячено ряд публікацій [1, 2]. У роботах [3–6] здійснено аналіз та оцінку показників роботи апаратних та програмних реалізацій легковагових алгоритмів на різних платформах за певних сценаріїв використання. Наприклад, у роботі [4] показано, що алгоритми, ASCON, TinyJambu, Xoodyak досягли в 10–25 разів кращої енергоефективності, ніж ISAP, Elephant та Grain-128AEAD при обробці однакової кількості біт.

Робота [7] надає оцінку криптографічного алгоритму-переможця ASCON для реальних IoT-застосувань. Зазначається, що ASCON потребує близько 0.2% RAM та 0.1% CPU для шифрування повідомлення довжиною 10 байт на Raspberry Pi Zero, у той час коли AES потребує близько 0.4% RAM та 0.2% CPU для шифрування такого ж повідомлення на тому самому пристрої.

У роботі [8] особлива увага приділяється аналізу вразливостей 10 алгоритмів-фіналістів конкурсу NIST зі стандартизації легковагової криптографії до атак з використанням помилок.

Разом з тим, варто зазначити, що процес стандартизації легковагової криптографії залишається недостатньо висвітленим в українській науковій літературі, що стало підгрунтям для обрання теми нашого дослідження.

Метою роботи є огляд та порівняльний аналіз алгоритмів-фіналістів конкурсу NIST зі стандартизації легковагової криптографії. В рамках статті планується описати основні конструкції та криптографічні примітиви, що лежать в основі алгоритмів-фіналістів. Стаття також має на меті провести дослідження специфікацій кожного алгоритму, проаналізувати їх можливі варіанти та модифікації, порівняти алгоритми-фіналісти за рядом характеристик (розміри ключа, блоку, стану, тегу, режими роботи тощо), оцінити показники їх безпеки проти відомих типів атак.

Виклад основного матеріалу

Національний інститут стандартів і технологій (National Institute of Standards and Technology, NIST) почав дослідження легковагової криптографії ще у 2013 році. Після декількох воркшопів і обговорень із зацікавленими сторонами в уряді, промисловості та наукових колах, у 2015 році NIST ініціював процес стандартизації алгоритмів, які б забезпечували функції автентифікованого шифрування та хешування для обмежених середовищ, де продуктивність поточних криптографічних стандартів NIST була неприйнятною.

У 2018 році було оголошено конкурс NIST Lightweight Cryptography з метою визначення стандарту легковагової криптографії та опубліковано основні вимоги до учасників конкурсу [9]. Алгоритми шифрування повинні забезпечувати автентифіковане шифрування з приєднаними даними (Authenticated Encryption with Associated Data, AEAD), тобто поєднувати як шифрування, так і автентифікацію в одному криптографічному алгоритмі. AEAD дозволяє не лише забезпечити конфіденційність та цілісність повідомлення, але й автентифікувати додаткові дані, які не підлягають шифруванню. Для перевірки автентичності даних використовується тег автентифікації (tag), що є унікальним значення фіксованого розміру, обчислення якого зазвичай відбувається за допомогою деякої криптографічної функції хешування. Відомі існуючі на той момент «легкі» криптографічні алгоритми такі як ACORN, PRESENT, CLEFIA, GRAIN, SIMON та інші не забезпечували шифрування і автентифікацію одночасно.

Учасникам було дозволено подавати на конкурс сімейство (не більше 10) алгоритмів, де члени сімейства можуть відрізнятися зовнішніми параметрами (наприклад, довжиною ключа) або внутрішніми параметрами (наприклад, кількістю раундів або розміром стану). Сімейство алгоритмів має включати одного основного члена, у якого довжина ключа щонайменше 128 біт, nonce (випадкове значення, що застосовуються для підвищення рівня безпеки) – 96 біт та тег – 64 біти. Обмеження на розміри вхідних даних (відкритий текст, приєднані дані та обсяг даних, які можна обробити з одним ключем) становить 2^{50}-1 байт. Для хеш-функцій обчислена довжина дайджесту (хешу) повинна бути щонайменше 256 біт, а максимальна довжина повідомлення 2^{50}-1 байт. Загальні компоненти алгоритмів хешування мають бути сумісні з AEAD-алгоритмами того самого сімейства. Основні критерії оцінки алгоритмів наступні:

1) *Стійкість до атак.* Алгоритм повинен бути стійким до методів лінійного та диференціального криптоаналізу, а також до таких типів атак як, атака бічними каналами (Side Channel Attack) та атаки на основі помилок (Fault Attack). Криптоаналітичні атаки мають потребувати щонайменше 2^{112} обчислень на класичному комп'ютері.

2) *Швидкодія на різних платформах.* Алгоритм повинен мати хорошу швидкодію на різних типах пристроїв, таких смартфони, смарт-карти, RFID-датчики, сенсори тощо. Швидкодія вимірюється за такими

показниками, як час шифрування, розшифрування, хешування одного блоку даних, пропускна здатність тощо.

3) *Енергоефективність.* Алгоритм повинен мати низьке енергоспоживання на пристроях, які працюють від батарейок або безпровідного живлення. Енергоефективність вимірюється за такими показниками, як кількість циклів процесора на один байт даних, кількість операцій на один байт даних, коефіцієнт енергоефективності тощо.

4) *Розмір коду.* Алгоритм повинен мати малий розмір коду для того, щоб займати менше пам'яті на обмежених пристроях. Розмір коду вимірюється за такими показниками, як кількість байт коду для реалізації алгоритму на C/C++, коефіцієнт стиснення коду тощо.

Загалом алгоритми-учасники конкурсу повинні були мати значно кращу ефективність в обмежених середовищах (апаратні та вбудовані програмні платформи) порівняно з поточними стандартами NIST.

У березні 2019 року NIST отримав 57 заявок з 25 різних країн, з яких 56 були прийняті до розгляду як кандидати для стандартизації у першому турі. Процес оцінки алгоритмів виконувався за допомогою відкритих методів та був доступним для перевірки й коментарів з боку громадськості та експертів. До другого туру, у серпні 2019 року, пройшли 32 алгоритми. Провівши складну процедуру відбору кандидатів, у 2021 році, було сформовано список 10 фіналістів, а саме ASCON, Elephant, GIFT-COFB, Grain-128AEAD, ISAP, PHOTON-Beetle, Romulus, SPARKLE, TinyJambu та Xoodyak. Зрештою, у лютому 2023 року NIST оголосив про рішення схвалити сімейство алгоритмів ASCON як новий стандарт легковагової криптографії.

Варто зазначити, що переважна частина алгоритмів-фіналістів використовують криптографічну губку як основу для своєї роботи. Криптографічна губка (Sponge) – це конструкція, яка може використовуватись для різноманітних криптографічних операцій, таких як хешування, шифрування, автентифікація та генерація випадкових чисел. Головна ідея губки полягає в тому, що вона може бути використана для обчислення функцій з різною фінальною довжиною виводу (дайджесту, шифру, тегу автентифікації тощо), шляхом зміни лише параметрів конструкції, а не її основного алгоритму. Найбільш відомий стандарт, який у своїй роботі використовує конструкцію губки є алгоритм хешування SHA-3 (Keccak).

Губка будується з трьох основних компонентів: функція доповнення pad, внутрішня функція перетворення f (зазвичай реалізовує псевдовипадкову перестановку за допомогою побітових операцій циклічного зсуву, XOR, AND, NOT) та внутрішній стан S (масив двійкових значень фіксованого розміру b (бітів)). Стан S поділяється на дві частини – бітрейт стану S_r та ємність стану S_c довжинами r та c відповідно, таким чином $b = r + c$. Бітрейт відповідає за швидкість обробки відкритого тексту, в той час як ємність відповідає за рівень безпеки.

Схематично структуру губки зображено на рис. 1 [10]. Спочатку вхідне повідомлення M, за допомогою функції доповнення pad, доповнюється (за необхідності) до довжини кратній бітрейту r та ділиться на блоки P_i. Потім b біт стану ініціалізуються нулем, і подальша робота губки проводиться в 2 етапи.

Поглинання (Absorbing). Вхідні дані подаються в губку блоками фіксованого розміру r. Губка послідовно абсорбує (поглинає) ці блоки даних – блок P_i додається за модулем 2 з бітрейтом стану S_r: $S = (S_r \oplus P_i)$, після чого проводиться перестановка із використанням внутрішньої функції перетворення f: $S = f(S)$. Така послідовність дій виконується з усіма блоками повідомлення.

Вижимання (Squeezing). Після поглинання губка переходить у режим вижимання, де вона «вичавлює» дані фіксованої довжини (або будь-якої потрібної довжини) зі свого внутрішнього стану. Це може бути хеш-значення, зашифрований текст або інші вихідні дані. Тепер на кожній ітерації формується вихідна послідовність бітів Z_i, елементами якої є біти бітрейту стану S_r: $Z_i = S_r$, а до стану знову застосовується перестановка f: $S = f(S)$. І це повторюється стільки разів, скільки блоків Z_i потрібно. У результаті отримується вихідне значення Z потрібної довжини l. У разі перевищення потрібної довжини результату зайві біти відкидають.

Для шифрування розробники передбачили дуплексний режим. Відмінність полягає в тому, що на кожній ітерації сформована послідовність бітів Z залежить від усіх раніше отриманих вхідних даних.

SPONGENT – це сімейство хеш-функцій, яке засноване на конструкції губки та використовує перестановку на зразок блокового шифру PRESENT [11]. SPONGENT має 13 варіантів, які позначають SPONGENT-$n/c/r$, де n – довжина хешу, c – ємність стану та r – бітрейт стану (наприклад, SPONGENT-128/256/128, SPONGENT-160/320/160, SPONGENT-224/448/224). Перестановка Spongent-π використовується як криптографічний примітив в алгоритмі-фіналісті Elephant і кількох інших LWC алгоритмах. Перестановка складається з трьох основних кроків: XOR бітів стану зі значенням, яке оновлюється за допомогою регістра зсуву лінійного зворотного зв'язку (LFSR), порозрядної підстановки з використанням 4-бітового S-боксу і бітової перестановки згідно таблиці.

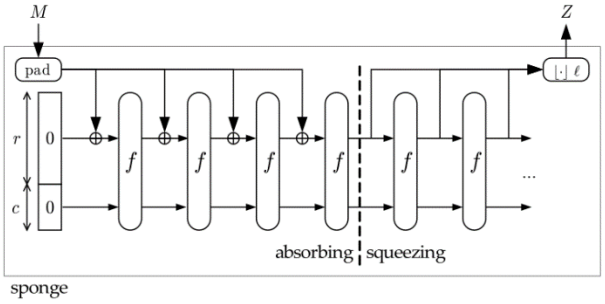

Рис. 1. Конструкція криптографічної губки (звичайний режим роботи)

Keccak – це сімейство універсальних хеш-функцій, яке базується на конструкції губки, має гнучкі параметри й може використовуватися як для хешування, так і для автентифікованого шифрування або генерації псевдовипадкових послідовностей [12]. У 2015 році NIST оголосив Keccak новим примітивом для використання в SHA-3 (Secure Hash Algorithm 3).

У Keccak основним примітивом є функція перестановки Keccak-f, обрана з набору 7 перестановок, позначених Keccak-f[b], де $b = 25 \times 2^{\ell}$ – розмір стану, ℓ – ціле число від 0 до 6. Keccak-f використовує операції XOR, AND і NOT, і розрахована на просту реалізацію як на програмному, так і апаратному рівнях. Кількість раундів може бути $12 + 2 \times \ell$ та відрізнятися в залежності від версії Keccak і необхідного рівня безпеки. Основна реалізація SHA-3 використовує значення $\ell = 6$, тобто має 1600 бітовий внутрішній стан і 24 раунди.

Стан S є тривимірним масивом розмірністю $5 \times 5 \times w$, де $w = 2^{\ell}$ біт. Складові масиву стану можуть бути представлені у вигляді рядка (raw), стовпчика (column), зрізу (slice), смуги (lane), площини (plane) тощо.

Кожен раунд Keccak-f складається з послідовного застосування кроків описаних нижче.

- θ (theta) – на цьому кроці стан розглядається як двовимірний масив (5×5), де кожен елемент масиву складається з одного слова довжиною w біт. Якщо позначити цей масив через $A(x, y)$, де $x, y = 0,1,\ldots,4$, то на кроці θ виконуються такі операції:

$$C[x] = A[x,0] \oplus A[x,1] \oplus A[x,2] \oplus A[x,3] \oplus A[x,4], \quad x = 0,1,2,3,4;$$

$$D[x] = C[x-1] \oplus rot(C[x+1],1), \quad x = 0,1,2,3,4;$$

$$A[x, y] = A[x, y] \oplus D[x], \quad x, y = 0,1,2,3,4.$$

$C[x]$ і $D[x]$ є одновимірними масивами, які містять п’ять слів довжиною w біт. Символ $\oplus$ позначає побітову операцію XOR двох w -бітових операндів, а $rot(C[x+1],1)$ виконує поворот операнда на один біт (в напрямку осі z). Усі індекси беруться за модулем 5, наприклад, $C[-1]$ означає $C[4]$.

- ρ (rho) – на цьому кроці відбувається обертання кожного біта стану на певне значення, залежно від його координат. В результаті чого отримується новий масив B:

$$B[y, 2x+3y] = rot(A[x, y], r[x, y]), \quad x, y = 0,1,2,3,4.$$

Операція $rot([x, y], i)$ повертає у масиві A одне слово на i бітових позицій праворуч відносно осі z. Кількість обертань становить $r[x, y]$ та визначається таблицею констант обертання, створеною розробниками.

- π (pi) – цей крок виконує перестановку координат кожного біта стану, що полягає у зміні положення смуг.
- χ (chi) – на цьому кроці виконується нелінійне перетворення кожного рядка стану за допомогою операцій XOR, NOT та AND і застосовується до площини за раз. Крок χ маніпулює масивом B, обчисленим на попередньому кроці, і розміщує результат у масиві стану A:

$$A[x, y] = B[x, y] \oplus \left(\left(\overline{B[x+1, y]}\right) \wedge B[x+2, y]\right), \quad x, y = 0,1,2,3,4.$$

$\overline{B[i, j]}$ позначає побітове доповнення смуги за адресою [i, j], а ∧ є порозрядна логічна операція AND двох операндів. Як і на всіх інших етапах, індекси беруться за модулем 5.

- ι (iota) – цей крок, виконує XOR смуги з раундовою константою в точці [0,0] масиву стану A:

$A[0,0] = A[0,0] \oplus RC[i]$.

Розглянемо детальніше 10 алгоритмів-фіналістів конкурсу NIST LWC та основні принципи їх роботи.

ASCON – сімейство алгоритмів, що базується на конструкції криптографічної губки, з однойменною перестановкою ASCON всередині. ASCON включає 7 варіантів криптографічних алгоритмів, що надають широкий спектр функціональних можливостей для вирішення різноманітних завдань безпеки, основні з яких – автентифіковане шифрування з приєднаними даними та хешування.

До AEAD-алгоритмів належать блокові шифри ASCON-128 та ASCON-128a. Також існує ще один варіант шифрування під назвою ASCON-80pq, який володіє стійкістю до квантового пошуку ключів. Ці шифри характеризуються довжиною ключа k, розміром блоку даних r, кількістю раундів a для перестановок p^a на початковій та завершальній стадіях і кількістю раундів b для перестановок p^b на проміжних стадіях.

ASCON-128 та ASCON-128a використовують 128-бітові ключі та 64-бітовий та 128-бітовий блоки вхідних даних відповідно. Також обидві конфігурації використовують 12 раундів перестановок p^a. А от кількість раундів перестановок p^b відрізняються: ASCON-128 передбачає 6 раундів, ASCON-128a – 8.

Для хешування ASCON пропонує набір хеш-функцій ASCON-Hash та ASCON-Hasha, ASCON-Xof та ASCON-Xofa. Хеш-функції ASCON-Hash та ASCON-Hasha обробляють блоки вхідних даних r довжиною 64 біт, а на виході отримують хеш довжиною 256 біт. Основна відмінність полягає в кількості раундів перестановок p^b, зокрема у першої функції – це 12 раундів, у другої – 8. Розробниками рекомендується поєднувати між собою ASCON-128 та ASCON-Hash (працюють з однаковим розміром блоку), або ASCON-128a та ASCON-Hasha (мають однакове значення p^b).

Головні процеси в шифрах ASCON – це дві 320-бітні перестановки p^a та p^b, які здійснюються над внутрішнім станом S. В процесі перестановок послідовно використовується перетворення, засноване на SP-мережі. Воно складається з етапів додавання констант, заміни за допомогою S -box та лінійної дифузії з 64-бітними дифузійними функціями.

Elephant – це також блоковий шифр на основі перестановок, що підтримує AEAD. Для автентифікації використовується процедура Encrypt-then-MAC – відкритий текст спочатку шифрується із застосуванням секретного ключа та алгоритму шифрування, а потім на основі отриманого зашифрованого тексту обчислюється MAC. Elephant був спроектований таким чином, щоб його роботу можна було легко розпаралелити.

Розробники представили три модифікації: дві з них – Dumbo та Jumbo використовують перестановку Spongent на 160 та 176 біт відповідно (забезпечують різний рівень безпеки), а третя – Delirium використовує перестановку Keccak на 200 біт. Усі три варіанти використовують LFSR для маскування. Основне маскування, що використовується в автентифікованому шифруванні, складається з LFSR і перестановки (Spongent або Keccak).

Dumbo і Jumbo мають розмір тегу 64 біти, тоді як Delirium – 128 біт. Розмір ключа становить 128 біт, а тегу – 96 біт для всіх трьох варіантів. Рівні безпеки, які вони забезпечують – 112 біт, 127 біт і 127 біт відповідно.

Алгоритм шифрування отримує на вхід ключ, nonce, приєднані дані та повідомлення, і повертає зашифрований текст та тег для автентифікації, розшифрування отримує на вхід ті самі параметри, але повертає розшифроване повідомлення, якщо тег коректний.

GIFT-COFB – легковаговий криптографічний алгоритм, який базується на блоковому шифрі GIFT-128 та використовує режим COFB (COmbined FeedBack). Його головною особливістю є ефективне поєднання GIFT-128 із зовнішнім режимом COFB, який дозволяє забезпечити автентифікацію шифрування без необхідності виконання складних операцій інверсії при розшифруванні. Режим COFB обрано для мінімізації розміру апаратної реалізації, що має значення для простоти та ефективності криптографічних алгоритмів.

Алгоритм GIFT-COFB використовує 128-бітний блок та 128-бітний тег. Його криптографічний примітив, GIFT-128, є 40-раундовим SPN-шифром зі структурою, подібною до AES, та ключем довжиною 128 біт. Кожен раунд складається з наступних операцій: ініціалізація, перетворення комірок, перестановка бітів та додавання раундового ключа. Існує альтернативна реалізація другого етапу – з використанням look-up table (LUT). В стандартному підході використовується послідовність бітових операцій для здійснення необхідних перетворень, а в LUT-варіанті – таблиця заздалегідь обчислених значень. Застосування другого способу може покращити ефективність, але автори наголошують, що він є повільнішим, ніж виконання бітових операцій.

Алгоритм шифрування приймає на вхід ключ шифрування K, блок асоційованих даних A та блок відкритого тексту M і генерує шифротекст C та тег автентифікації T таким чином, щоб $|C| = |M|$ і

$|\mathrm{T}| = n$, де n – розмір блоку шифрування в бітах. GIFT-COFB використовує значення n =128, функції доповнення (Padding Function) та зворотнього зв'язку (Feedback Function). Алгоритм дешифрування повертає вихідні дані M або $\perp$, якщо автентифікація не пройшла.

Grain-128AEAD – один із небагатьох біт-орієнтованих потокових алгоритмів з автентифікацією, представлених на конкурсі, оптимізований для апаратної реалізації. Він базується на шифрі Grain, в основі якого є регістри зсуву із зворотнім зв'язком. Версія Grainv1 була обрана фіналістом у 2008 році у профілі апаратних реалізацій в eSTREAM portfolio. Версія Grain-128a включена в стандарт ISO/IEC 29167-13:2015 для систем ідентифікації RFID. Алгоритм Grain-128AEADv2 схожий на Grain-128a, але був допрацьований для підтримки тегів більшого розміру та підтримки AEAD.

Grain-128AEADv2 складається з двох основних блоків: генератора гами, який побудовано з використанням лінійного (LFSR) та нелінійного (NFSR) регістрів зсуву і спеціальної функції для створення гами, та генератора тегу, що містить в собі регістр зсуву та акумулятор, біти якого залежать від гами та бітів відкритого тексту одночасно. Тег формується з усіх бітів акумулятора, які утворилися після оброблення вхідного тексту, та має розмірність 64 біти. Розміри регістрів LFSR та NFSR – по 128 біт кожен. Перед початком роботи алгоритм ініціалізує всі компоненти, використовуючи ключ, довжиною 128 біт, та nonce, довжиною 96 біт. Кількість раундів, зазначена авторами для фази ініціалізації – 512.

В режимі AEAD асоційовані дані не шифруються, але вони приймають участь у формуванні тегу. Алгоритм передбачає наявність бітової маски для виокремлення інформації, яка не потребує шифрування, але має бути автентифікованою. Її використання забезпечує гнучкість алгоритму, так як вона дозволяє включати незашифровані дані не тільки на початку, а і в будь-яких позиціях переданого набору бітів.

ISAP – це сімейство автентифікованих шифрів на основі перестановок, представлене чотирма варіантами: ISAP-A-128A, ISAP-K-128A, ISAP-A-128 і ISAP-K-128. Всі вони спроектовані для забезпечення 128-бітної безпеки від криптоаналітичних атак і відповідний рівень стійкості до атак бічними каналами, але два з них використовують 320-бітну перестановку Ascon-p, а інші – 400-бітну перестановку Keccak-f.

Розміри ключа, nonce та тегу для всіх згаданих варіантів – 128 біт. Значення кількості раундів представлене 4 числами. Перше – кількість раундів для фази автентифікації, друге – для обробки значення nonce у функції зміни ключа, третє – для фаз шифрування та дешифрування, і останнє – для генерації сесійних ключів у функції зміни ключа. Для ISAP-A-128A це значення виглядає як 16/1/8/8, для ISAP-K-128A – 12/1/6/12, для ISAP-A-128 – 20/12/12/12, для ISAP-K-128 – 12/12/12/12.

ISAP використовує підхід Encrypt-then-MAC, де шифрування виконується шляхом XOR-повідомлення та згенерованого потоку ключів, а автентифікація здійснюється на основі схеми Hash-then-MAC. Шифрування побудоване на базі конструкції губки в потоковому режимі з однією особливістю: спочатку викликається функція зміни ключа IsapRk для генерації підключа K_e, після чого все виконується в звичайному режимі. Функція IsapEnc отримує на вхід ключ K, nonce N та довільне велике повідомлення M і генерує шифротекст C розміром $|M|$. IsapEnc працює в режимі потокового шифрування, тому розшифрування ідентичне, за винятком того, що значення M і C міняються місцями.

Конструкція має можливість зміни ключа перед кожним застосуванням губки, що робить її стійкою до пасивних атак бічними каналами, зокрема атак типу DPA і SPA. Всі варіанти ISAP забезпечують конфіденційність повідомлення та цілісність шифротексу, включаючи асоційовані дані.

PHOTON-Beetle – це сімейство схем автентифікованого шифрування та хеш-функцій, яке використовує перестановку PHOTON256. Рекомендованими варіантами шифрування є PHOTON-Beetle[32] та PHOTON-Beetle [128], які відрізняються значеннями рейту. Також запропоновано хеш-функцію – PHOTON-Beetle-Hash [32]. PHOTON-Beetle-AEAD ґрунтується на «губкоподібному» режимі AEAD Beetle з комбінованим зворотним зв'язком, тоді як PHOTON-Beetle-Hash базується на структурі губки.

Перестановка PHOTON256 складається з 12 раундів, у кожному з яких виконуються операції AddConstant, SubCells, ShiftRows та MixColumnSerial. AddConstant забезпечує додавання фіксованих констант до комірок внутрішнього стану, SubCells застосовує 4-бітний S-Box до кожної з комірок, ShiftRow змінює позиції комірок в кожному рядку, а MixColumnSerial відповідає за перемішування стовпців шляхом множення визначеної матриці на матрицю стану.

Функція шифрування приймає на вхід ключ шифрування K, nonce N, асоційовані дані A та повідомлення M і повертає шифротекст C та тег T. Відповідно алгоритм дешифрування як вхідні дані приймає значення K, N, A, C і T та повертає відповідне відкрите повідомлення M, якщо тег перевірено.

PHOTON-Beetle-Hash приймає повідомлення будь-якої довжини і генерує хеш-значення довжиною 256 біт. Повідомлення розділяється на блоки, які послідовно обробляються за допомогою перестановки PHOTON256, а значення хешу скорочується до двох частин по 128 біт кожна.

Romulus – це сімейство алгоритмів автентифікованого шифрування з асоційованими даними (AEAD) і хеш-функції, яке базується на твіковому блочному шифрі Skinny-128-384+ (варіант Skinny-128-384, але замість 56 раундів використовуються 40).

Romulus має кілька варіантів: Romulus-N (з підтримкою унікального nonce), Romulus-M (стійкий до неправильного використання nonce), Romulus-T (стійкий до витоків) і Romulus-H (хеш-функція). Romulus-N використовує режим зі зворотнім зв'язком із комбінованим множником на основі змінного блочного шифру, Romulus-M – режим MAC-then-Encrypt, Romulus-T – режим Encrypt-then-MAC.

AEAD варіанти Romulus мають наступні параметри: довжина ключа, розміри nonce та тегу дорівнюють 128 біт. Для Romulus-H характерними ознаками є розмір блоку 256 біт та розмір дайджесту 256 біт.

Базова структура алгоритму включає в себе ініціалізацію з початковим налаштуванням внутрішнього стану, 40 раундів шифрування та формування шифротексту на базі остаточного внутрішнього стану блоку. Кожен раунд включає в себе операції, які ускладнюють структуру шифру і роблять його стійким до криптоатак. До них відносять SubCells (заміна значень кожної комірки стану блоку згідно з таблицею S-Box), AddConstants (генерація та додавання раундових констант до матриці стану), AddRoundTweakey (об'єднання перших двох рядків твікового ключа і стану блоку), ShiftRows (зсув рядків), та MixColumns (перемішування стовбців шляхом множення на визначену матрицю).

SPARKLE – це сімейство шифрів перестановки, яке включає схеми автентифікованого шифрування SCHWAEMM (Sponge-based cipher for Hardened but Weightless Authenticated Encryption on Many Microcontroller) та хеш-функції ESCH (Efficient Sponge-based and Cheap Hashing). Існують наступні варіанти SCHWAEMM: SCHWAEMM256-128, SCHWAEMM128-128, SCHWAEMM192-192 та SCHWAEMM256-256. Перше число в назві означає розмір nonce, друге – розмір ключа (в бітах). Розмір тегу для кожного з варіантів співпадає з розміром ключа, значення рейту – з розміром nonce. Рівень безпеки, який вони забезпечують, відповідно 120, 120, 184 і 248 біт. Проте він є дійсним тільки при унікальному використанні nonce.

Стосовно хеш-функцій, є два варіанти: ESCH256 (розмір дайджесту – 256 біт, рейт – 128 біт) та ESCH384 (розмір дайджесту – 384 біт, рейт – 128 біт). Перший забезпечує рівень безпеки на рівні 128 біт, другий – 192 біт.

AEAD-шифри SCHWAEMM побудовані на основі схеми подвійної губки з комбінованим зворотним зв'язком, хеш-функції – на основі губки. Перестановку SPARKLE побудовано за принципом ARX (Addition-Rotation-XOR) структури. Вона має 3 різновиди, залежно від розміру блоку – SPARKLE256, SPARKLE384 та SPARKLE512. Кожен з них має дві версії – slim та big, які відрізняються кількістю кроків, що використовуються. Slim версії мають меншу кількість кроків для оптимізації продуктивності на платформах з обмеженими ресурсами, тоді як big версії забезпечують більший рівень безпеки.

TinyJAMBU – це схема автентифікованого шифрування, розроблена на основі кандидата JAMBU третього раунду конкурсу CAESAR. Основною складовою режиму є ключова перестановка (без розкладу ключа), яка базується на 128-бітовому нелінійному регістрі зворотного зв'язку. Кожен раунд використовує операцію NAND для забезпечення нелінійності.

Є три варіанти AEAD для даної схеми з різною кількістю раундів (представлена у вигляді двох чисел, де перше – кількість раундів, що використовується для налаштування ключа, обробки тексту та генерації перших 32 біт тегу, а друге – кількість раундів, які застосовуються для обробки nonce, асоційованих даних та генерації останніх 32 біт тегу): TinyJAMBU-128 (128-бітовий ключ, 1024/640 раундів), TinyJAMBU-192 (192-бітовий ключ, 1152/640 раундів), TinyJAMBU-256 (256-бітовий ключ, 1280/640 раундів). Розмір nonce дорівнює 96 біт, тегу – 64, стану – 128 біт для всіх вище зазначених варіантів.

У TinyJAMBU використовується 128-бітний ключова перестановка P_n, що складається з n раундів. У кожному і-му раунді використовується 128-бітний нелінійний регістр зворотного зв'язку для оновлення стану. На 32-бітному процесорі можна обчислити 32 раунди перестановки паралельно.

Процес шифрування в алгоритмі TinyJAMBU складається з наступних кроків: ініціалізація початкового стану на основі ключа та nonce, обробка асоційованих даних шляхом застосування фрагментів FrameBits та 32-бітних блоків AD до стану S за допомогою перестановки P 640, шифрування з використанням перестановок P 1024 (TinyJAMBU-128), P 1152 (TinyJAMBU-192) або P 1280 (TinyJAMBU-256), та завершення і отримання 64-бітного тегу, який представляє фінальні 64 біта стану S.

Xoodyak – це універсальний криптографічний примітив, який функціонує як схема автентифікованого шифрування на основі перестановки і як алгоритм хешування. Він побудований на основі фіксованої перестановки з 384-бітовим станом, яка має назву Xoodoo, і працює в режимі Cyclist. Вона використовує тривимірний масив з розмірами 3x4x32 біт, забезпечуючи нелінійність за допомогою простих операцій над 3-бітовими стовпчиками та лінійного змішування між рівнями.

Xoodyak пропонує два режими роботи: хешування та шифрування з ключем, один з яких обирається при ініціалізації. У режимі хешування, він може абсорбувати вхідні рядки даних і отримувати з них дайджести за потреби. Функція Absorb(X) абсорбує вхідний рядок X, тоді як функція Squeeze(ℓ) видає дайджест довжиною ℓ байтів в залежності від вже абсорбованих даних. Режим з ключем забезпечує можливість потокового шифрування, обчислення MAC та автентифікованого шифрування.

Функція Ratchet() в алгоритмі використовується для динамічного оновлення підключів, що ускладнює задачу відновлення ключів або аналізу залежності між різними комунікаційними сесіями.

Для AEAD варіанту Xoodyak визначено наступні параметри: розміри ключа, nonce та тегу – по 128 біт, значення рейту – 192 біта, кількість раундів – 12. Хеш-варіант алгоритму має розмірність дайджесту 256 біт та значення рейту – 130 біт.

Загальні характеристики алгоритмів-фіналістів наведено у таблицях 1 та 2. Як показано в таблиці 1, більшість (сім) фіналістів є алгоритмами на основі криптографічної губки, два фіналісти є представниками блокових шифрів та один потоковий шифр.

Таблиця 1

Загальні характеристики AEAD-алгоритмів NIST LWC

Алгоритм-фіналіст	Варіант	Основний примітив	Режим роботи	Ключ (біт)	Nonce (біт)	Тег (біт)
ASCON	ASCON-128	ASCON Permutation	MonkeyDuplex	128	128	128
	ASCON-128a			128	128	128
	ASCON-80pq			160	128	128
Elephant	Dumbo	Spongent-π[160]	Encrypt-then-MAC	128	96	64
	Jumbo	Spongent-π[176]		128	96	64
	Delirium	Keccak-*f*[200]		128	96	128
GIFT-COFB	GIFT-COFB	GIFT-128	Combined Feedback	128	128	128
Grain-128 AEAD	Grain-128AEAD	Feedback shift register	Encrypt-and-MAC	128	96	64
ISAP	ISAP-A-128a	ASCON Permutation	Encrypt-then-MAC	128	128	128
	ISAP-K-128a	Keccak-*f* [400]		128	128	128
	ISAP-A-128	ASCON Permutation		128	128	128
	ISAP-K-128	Keccak-*f* [400]		128	128	128
PHOTON-Beetle	PHOTON-Beetle-AEAD[128]	$PHOTON_{256}$ Permutation	Sponge with Combined Feedback	128	128	128
	PHOTON-Beetle-AEAD[32]			128	128	128
Romulus	Romulus-N	Skinny-128-384+ Tweakable Block Chipher	Combined Feedback	128	128	128
	Romulus-M		MAC-then-Encrypt	128	128	128
	Romulus-T		Encrypt-then-MAC	128	128	128
SPARKLE	SCHWAEMM256-128	$SPARKLE_{384}$	Sponge with Combined Feedback	128	256	128
	SCHWAEMM128-128	$SPARKLE_{256}$		128	128	128
	SCHWAEMM192-192	$SPARKLE_{384}$		192	192	192
	SCHWAEMM256-256	$SPARKLE_{512}$		256	256	256
TinyJAMBU	TinyJAMBU-128	Keyed Permutation	Sponge	128	96	64
	TinyJAMBU-192			192	96	64
	TinyJAMBU-256			256	96	64
Xoodyak	Xoodyak	Xoodoo Permutation	Cyclist	128	128	128

Таблиця 2

Загальні характеристики алгоритмів хешування NIST LWC

Алгоритм-фіналіст	Варіант	Основний примітив	Режим роботи	Дайджест (біт)
ASCON	ASCON-Hash	ASCON Permutation	Sponge	256
	ASCON-Hasha			256
PHOTON-Beetle	PHOTON-Beetle-Hash [32]	$PHOTON_{256}$ Permutation	Sponge	256
Romulus	Romulus-H	Skinny-128-384+	MDPH[1]	256
SPARKLE	$ESCH_{256}$	$SPARKLE_{384}$	Sponge	256
	$ESCH_{384}$	$SPARKLE_{512}$		384
Xoodyak	Xoodyak	Xoodoo Permutation	Sponge	256

[1]MDPH означає Merkle-Damgard з перестановкою та використанням функції стиснення Hirose DBL

В таблиці 3 наведено результати комплексної оцінки криптографами NIST безпеки кожного з десяти алгоритмів-фіналістів відповідно до кількох ключових аспектів. Кількість доступних реалізацій дає загальну картину наявності та різноманітності варіантів для кожного алгоритму.

Важливо відзначити, що реалізації алгоритмів-фіналістів пройшли детальне дослідження з боку незалежних експертів, що підтверджує їх високий рівень безпеки. Наприклад, ASCON відрізняється великою кількістю аналізів і високим рівнем резервів безпеки, а результати атак на його раунди підкреслюють його надійність. Деякі алгоритми, такі як Elephant, хоча і мають високий рівень безпеки, показали вразливості до певних видів атак, зокрема на основі підстановки. Водночас, такі варіанти, як Romulus, здатні зменшувати кількість раундів для забезпечення більшої ефективності, але при цьому зберігають високий рівень стійкості.

Таблиця 3

Оцінка безпеки фіналістів

Алгоритм-фіналіст	Рівень безпеки	Атаки	Кількість раундів для атаки	Кількість AEAD реалізацій	Кількість Hash реалізацій	Кількість комбінованих реалізацій	Загальна кількість реалізацій
ASCON	Високий	Key-recovery, Distinguishers	7 (з 12) раундів ініціалізації	120	110	52	282
Elephant	Високий	Distinguishers	40 (з 80) раундів Spongent	6	-	-	6
GIFT-COFB	Високий	Key-recovery	27 (з 40) раундів GIFT-128	11	-	-	11

Алгоритм-фіналіст	Рівень безпеки	Атаки	Кількість раундів для атаки	Кількість AEAD реалізацій	Кількість Hash реалізацій	Кількість комбінованих реалізацій	Загальна кількість реалізацій
Grain-128 AEAD	Високий	Key-recovery	192 (з 512) раундів ініціалізації	6	-	-	6
ISAP	Високий	Forgery	4 (з 12) раундів	37	1	4	42
PHOTON-Beetle	Обмежений	Distinguishing	10 (з 12) раундів перестановки	20	10	16	46
Romulus	Високий	Key-recovery	32 (з 40) раундів Skinny, 23 (з 40) раунди (для хеш-варіанту)	32	11	27	70
SPARKLE	Високий	Key-recovery, Distinguishers	4,5 (з 11) етапів 384-бітної перестановки	25	13	3	41
TinyJAMBU	Середній	Weak-key distinguishing, Forgery	476 (з 1024) раундів, повний раунд TinyJambu-192 і TinyJambu-256 (для хеш-варіанту)	9	-	-	9
Xoodyak	Високий	Key-recovery	6 (з 12) раундів	9	8	1	18

Висновки

Легковагова криптографія є областю досліджень, яка активно розвивається та адаптується до нових викликів та потреб. З огляду на постійний розвиток криптографічних методів та збільшення обсягу цифрового обміну даними, стандартизація легковагової криптографії стає дедалі важливішою. Саме стандартизація сприяє покращенню якості та безпеки легковагових криптографічних алгоритмів, оскільки вона стимулює їх дослідження та криптоаналіз.

Конкурс NIST вивів на передній план різноманітність легковагових алгоритмів, що пропонують різні підходи щодо безпеки та ефективності. Кожен з цих алгоритмів має свої унікальні характеристики, які визначають його придатність для конкретних сценаріїв використання. Багато алгоритмів показали хороші результати роботи при обмежених обчислювальних ресурсах, що дозволяє їх використовувати на платформах з низькою обчислювальною потужністю, зберігаючи при цьому надійний рівень захисту. З огляду на це, подальші дослідження зосереджуватимуться на можливостях застосування легковагових криптографічних алгоритмів у реальних системах з обмеженими ресурсами.

References

1. Gookyi N., Agyemanh D., Kanda G., Ryoo K. NIST Lightweight Cryptography Standardization Process: Classification of Second Round Candidates, Open Challenges and Recommendations. Journal of Information Processing Systems, Vol. 17, No. 2, pp. 253-270, Apr. 2021. doi.org/10.3745/JIPS.03.0156

2. Turan M. S. Status Report on the Final Round of the NIST Lightweight Cryptography Standardization Process. Gaithersburg, MD: National Institute of Standards and Technology, 2022. doi.org/10.6028/nist.ir.8454

3. Hira R. Software Evaluation for Second Round Candidates in NIST Lightweight Cryptography. Journal of Information Processing. 2023. Vol. 31. P. 205–219. doi.org/10.2197/ipsjjip.31.205

4. Elsadek I. Hardware and Energy Efficiency Evaluation of NIST Lightweight Cryptography Standardization Finalists. 2022 IEEE International Symposium on Circuits and Systems (ISCAS), Austin, TX, USA, 27 May – 1 June 2022. doi.org/10.1109/iscas48785.2022.9937643

5. Birleanu F. G., Bizon N. Quick Analysis of the NIST Lightweight Cryptography Standardization Process Finalists. 2022 14th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Ploiesti, Romania, 30 June – 1 July 2022. doi.org/10.1109/ecai54874.2022.9847450

6. Buchanan W.J., Leandros M. Review of the NIST Light-weight Cryptography Finalist. 2023. doi.org/10.48550/arXiv.2303.14785

7. Avery J. Analysis of Practical Application of Lightweight Cryptographic Algorithm ASCON. Available at: https://csrc.nist.gov/csrc/media/Events/2022/lightweight-cryptography-workshop-2022/documents/papers/analysis-of-practical-application-of-lwc-cryptographic-algorithm-ascon.pdf (Accessed 06 August 2023).

8. Madushan H., Salam I., Alawatugoda J. A Review of the NIST Lightweight Cryptography Finalists and Their Fault Analyses. Electronics. 2022. Vol. 11, no. 24. P. 4199. doi.org/10.3390/electronics11244199

9. NIST. Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process. (2018). Available at: https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdfhttps://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf (Accessed 06 August 2023).

10. Bertoni G. Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. Selected Areas in Cryptography. Berlin, Heidelberg, 2012. P. 320–337. doi.org/10.1007/978-3-642-28496-0_19

11. Bogdanov A. SPONGENT: The Design Space of Lightweight Cryptographic Hashing. IEEE Transactions on Computers. 2013. Vol. 62, no. 10. P. 2041–2053. doi.org/10.1109/tc.2012.196

12. Paar C., Pelzl J. Understanding cryptography: A textbook for students and practitioners. Heidelberg: Springer, 2010. 372 p. Available at: https://www.cryptotextbook.com/download/Understanding-Cryptography-Keccak.pdf.