УДК 004.056.5 **DOI:10.30837/rt.2025.4.223.02**

Я.А. ДЕРЕВ'ЯНКО, Д.Ю. ГОРБЕНКО

ТЕОРЕТИЧНЕ ОБҐРУНТУВАННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ ЕКСТРАКТОРІВ ЕНТРОПІЇ НА ОСНОВІ НАЦІОНАЛЬНИХ КРИПТОГРАФІЧНИХ АЛГОРИТМІВ

Вступ

Рекомендації щодо застосування ентропії під час формування випадкових даних наведено у NIST Special Publication 800-90B [1], де визначається порядок збору та оцінки ентропії з відповідних джерел шуму, а також порядок застосування екстракції ентропії з отриманих даних, щоб результат відповідав певним вимогам щодо ентропії на біт, рівномірності та непередбачуваності.

Першим та найважливішим кроком для побудови екстрактора є правильний вибір джерела шуму (ДШ). ДШ – це основа безпеки для джерела ентропії і для RNG в цілому, оскільки воно містить недетермінований процес, що і надає ентропію. Якщо недетермінована активність ДШ виробляє щось інше, ніж двійкові дані, процес відбору зразків включає в себе процес оцифрування.

ДШ поділяються на фізичні – використовують спеціальне обладнання для генерування випадковості та нефізичні – використовують системні дані або введення користувача системи для генерування випадковості.

Для побудови екстрактора необхідним є збір вибірок з певного ДШ, оцінка його ентропії за цими вибірками і подальша обробка з використанням певного криптографічного примітиву. Для подальшого отримання послідовностей довільної довжини на основі виходу з екстрактора можливим є застосування різноманітних криптографічних примітивів для розширення виходу екстрактора ентропії.

У статті проаналізовано існуючі рекомендації щодо побудування екстракторів ентропії на основі криптографічних примітивів, виконано теоретичне обґрунтування можливості застосування національних алгоритмів у якості подібних екстракторів, а також наведено приклади практичної реалізації екстракторів на основі національних криптографічних алгоритмів. Актуальність розробки цих нових засобів отримання випадковості обґрунтована важливістю використання результуючих даних з високими показниками рівномірності та непередбачуваності при обчисленні загальносистемних параметрів та генеруванні ключів квантовостійких стандартизованих криптографічних перетворень.

1. Принципи побудови екстрактора ентропії та криптографічні компоненти, що для цього використовуються

Згідно з [1] побудова екстрактора ентропії має обов'язково включати наступні кроки:

- вибір найбільш відповідного ДШ та збір зразків з нього;
- оцінка ентропії ДШ на основі зібраних «сирих» даних;
- застосування рекомендованого довіреного криптографічного примітиву для виконання екстракції ентропії з зібраних з зразків;
- оцінка ентропії на виході екстрактора.

1.1. Вибір джерела шуму та збір зразків

Саме ДШ в кінцевому рахунку відповідає за непередбачуваність бітових рядків, що виводяться екстрактором ентропії, тому важливим є вибір правильного ДШ з найбільш відповідними показниками ентропії. Важливим кроком при побудові екстрактора ентропії є коректне отримання ентропії з ДШ для виконання подальшої екстракції.

У [1] припускається, що значення вибірки (тобто символи) з ДШ складаються з бітових рядків фіксованої довжини. Рекомендації щодо збору зразків наступні.

Для валідації (тобто оцінки ентропії ДШ) повинен бути зібраний послідовний набір даних, що складається щонайменше з 1 000 000 значень, отриманих з ДШ (тобто необроблені дані). Якщо генерація 1 000 000 послідовних зразків неможлива, допускається конкатенація декількох менших наборів послідовних зразків (згенерованих з використанням того самого ДШ). Менші набори повинні містити щонайменше 1000 зразків. Об'єднаний набір даних повинен містити не менше 1 000 000 зразків [1].

Отже, для виконання наступного кроку – оцінки ентропії ДШ – необхідно отримати достатню кількість «сирих» зразків з ДШ.

1.2. Оцінка ентропії джерела шуму на основі «сирих» даних

Наступним важливим кроком є кількісна оцінка ентропії ДШ шляхом оцінки отриманих вибірок. У [1] передбачено проведення оцінки min-ентропії, яка відповідає найбільш консервативному способу вимірювання непередбачуваності набору і тому використовується як оцінка в найгіршому випадку. Також для більш повної оцінки у даній роботі передбачається обчислення ентропії Шеннона [2] та колізійна ентропія [3].

Методи оцінки min-ентропії засновані на двох підходах. Перший базується на ентропійній статистиці, а другий – на предикторах. У [1] визначаються такі оцінки: найпоширенішого значення, колізій, Маркова, стиснення, t-кортежу, найдовшого повторюваного підрядка, найпоширеніших значень у фіксованому вікні, прогнозування затримки, MultiMMC та LZ78Y.

В свою чергу ентропія Шеннона та колізійна ентропія розраховується для отримання незміщеної ентропії шляхом застосування методів корекції. Основними з них є [2]:

- Miller-Madow correction (MM);
- Jackknife (UnveilJ);
- Best Upper Bound (BUB);
- Chao-Chen (CS);
- James-Stein (SHR);
- Schürmann (SHU).

Таким чином, подальше виконання екстракції потребує впровадження оцінки min-ентропії ДШ, оскільки така оцінка є мірою ентропії для найгіршого випадку. Впровадження оцінок ентропії Шеннона та колізійної ентропії є опціональним і залежить від бажаної швидкості та складності роботи екстрактора

1.3. Криптографічні примітиви для екстракції ентропії з отриманих «сирих» даних

Згідно з [1] екстрактор ентропії отримує вхідні дані від джерела шуму та генерує вихід джерела ентропії. Розмір входу та виходу екстрактора ентропії в бітах, позначених як n_{in} та n_{out} відповідно, є фіксованими. Виходи ДШ об'єднуються для побудови n_{in}-бітового входу для функції екстракції. Ентропія входу, позначена h_{in}, залежить від кількості зразків, необхідних для побудови n_{in}-бітового входу. Якщо потрібно w зразків, то h_{in} оцінюється як $w \times h$ біт. Розмір входу екстрактора має бути кратним розміру виходу джерела шуму.

Оскільки екстрактор ентропії є детермінованим, ентропія виходу не перевищує h_{in}. Однак екстрактор може зменшити ентропію виходу. Ентропія виходу екстрактора позначається як h_{out}, тобто h_{out} бітів ентропії міститься в n_{out} бітах виходу. Ентропія виходу також залежить від внутрішньої будови екстрактора.

У [1] найвужча внутрішня ширина (Narrowest Internal Width) в межах екстрактора позначається як nw.

Загальну схему роботи екстрактора ентропії згідно з NIST 800-90B наведено на рис. 1.

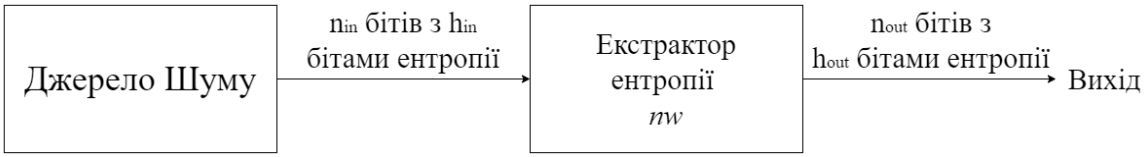

Рис. 1. Загальна схема роботи екстрактора ентропії

Як видно з рисунку, компонент приймає і повертає фіксовану кількість бітів. Важливо також, що ентропія, яку отримує екстрактор, не може перевищувати ентропію отриманих від ДШ даних, які подаються на його вхід.

1.3.1. Рекомендовані NIST криптографічні примітиви

У [1] визначено перевірені та рекомендовані NIST криптографічні примітиви, які необхідно застосовувати з метою екстракції ентропії з отриманих з ДШ даних, щоб вихідні дані задовольняли певним вимогам щодо ентропії на біт, рівномірності та непередбачуваності.

Серед них визначено три алгоритми з ключем [1]:

- HMAC, як зазначено в FIPS 198, з будь-якою затвердженою геш функцією, зазначеною в FIPS 180 або FIPS 202;
- CMAC, як зазначено в SP 800-38B, з блоковим шифром AES (FIPS 197);
- CBC-MAC, з блоковим шифром AES.

Та три алгоритми без ключа [1]:

- будь-яка затверджена геш функція, зазначена в FIPS 180 або FIPS 202;
- Hash_df, як зазначено в SP 800-90A, з використанням будь-якої затвердженої геш функції, зазначеної в FIPS 180 або FIPS 202;
- Block_Cipher_df, як зазначено в SP800-90A з використанням блочного шифру AES (FIPS 197).

Параметри екстракторів на представлених криптографічних примітивах наведені в табл. 1. У ній міститься інформація про найвужчу внутрішню ширину (Narrowest Internal Width) в межах екстрактора та розмір вихідних даних після екстракції.

Таблиця 1

Найвужча внутрішня ширина та довжина виходу перевірених функцій екстракції

Функція (компонент) екстракції	Найвужча внутрішня ширина (Narrowest Internal Width) nw	Довжина виходу екстрактора n_{out}
HMAC	Розмір виходу геш функції	Розмір виходу геш функції
CMAC	Розмір блоку шифру	Розмір блоку шифру
CBC-MAC	Розмір блоку шифру	Розмір блоку шифру
Геш функція	Розмір виходу геш функції	Розмір виходу геш функції
Hash_df	Розмір виходу геш функції	Розмір виходу геш функції
Block_Cipher_df	Розмір ключа шифру	Розмір ключа шифру

1.3.2. Національні криптографічні примітиви, які можливо застосувати у якості екстракторів

Аналіз рекомендованих NIST криптографічних примітивів та їх порівняння зі специфікацією національних стандартів України вказує на можливість застосування для виконання екстракції ентропії наступних алгоритмів та режимів: HMAC та алгоритм Hash_df з використанням геш функції Купина (ДСТУ 7564:2014) [4] та CMAC і алгоритм Block_Cipher_df з використанням блокового шифру Калина (ДСТУ 7624:2014) [5]. Режим CBC-MAC не визначено у [5], тому його застосування не є доцільним. При цьому використання [4] як самостійного екстрактора не рекомендовано, оскільки національна геш функція не належить до універсального геш-сімейства (як, наприклад, SHA), що в свою чергу означає, що її екстракційні властивості не доведено. У табл. 2 надано режими національних алгоритмів, які можливо застосовувати з метою екстракції та їх властивості.

Таблиця 2

Найвужча внутрішня ширина та довжина виходу перевірених функцій екстракції

Функція (компонент) екстракції	Найвужча внутрішня ширина (Narrowest Internal Width) nw	Довжина виходу екстрактора n_{out}
HMAC/Hash_df для Купина-256	Розмір виходу геш функції (256 біт)	Розмір виходу геш функції (256 біт)
HMAC/Hash_df для Купина-512	Розмір виходу геш функції (512 біт)	Розмір виходу геш функції (512 біт)
CMAC для Калина-128/128, Калина-128/256	Розмір блоку шифру (128 біт)	Розмір блоку шифру (128 біт)
CMAC для Калина-256/256, Калина-256/512	Розмір блоку шифру (256 біт)	Розмір блоку шифру (256 біт)
CMAC для Калина-512/512	Розмір блоку шифру (512 біт)	Розмір блоку шифру (512 біт)
Block_Cipher_df для Калина-128/128	Розмір ключа шифру (128 біт)	Розмір ключа шифру (128 біт)
Block_Cipher_df для Калина-128/256, Калина-256/256	Розмір ключа шифру (256 біт)	Розмір ключа шифру (256 біт)
Block_Cipher_df для Калина-Калина-256/512, Калина-512/512	Розмір ключа шифру (512 біт)	Розмір ключа шифру (512 біт)

Як видно з табл. 2, національні криптографічні примітиви надають змогу створення екстракторів з виходами довжини 128, 256 та 512 з хорошими показниками рівномірності та непередбачуваності. Далі виходи екстракторів можна використовувати для формування довших послідовностей з використанням експандерів (блокових шифрів у режимі CTR, потокових шифрів тощо).

1.4. Оцінка ентропії на виході екстрактора

Зважаючи на те, що вихідна ентропія після виконання екстракції, як вказано у [1] не може зрости, проте може зменшитися, NIST рекомендує оцінювати ентропію на виході:

При використанні перевірених (довірених) NIST компонентів:

Для оцінки передбачена формула

$$h_{out} = Output\_Entropy(n_{in}, n_{out}, nw, h_{in}), \quad (1)$$

де алгоритм обчислення $Output\_Entropy(n_{in}, n_{out}, nw, h_{in})$ наступний:

Вхід:

n_{in} – розмір входу екстрактора ентропії в бітах;

n_{out} – розмір виходу екстрактора ентропії в бітах;

nw – найвужча внутрішня ширина екстрактора ентропії в бітах;

h_{in} – ентропія входу.

Вихід:

h_{out} – значення оцінки ентропії після виконання екстракції.

1 Нехай $P_{high} = 2^{-h_{in}}$ та $P_{low} = \dfrac{(1-P_{high})}{2^{n_{in}}-1}$

2 $n = \min(n_{out}, nw)$

3 $\psi = 2^{n_{in}-n} P_{low} + P_{high}$

4 $U = 2^{n_{in}-n} + \sqrt{2n(2^{n_{in}-n})\ln(2)}$

5 $\omega = U \times P_{low}$

6 Повернення $-\log_2(\max(\psi, \omega))$

Алгоритм 1. Обчислення ентропії на виході екстрактора

Отже, джерело ентропії оцінюється за мінімальною ентропією на вихід екстрактора, h_{out}. Перевірені NIST екстрактори можуть претендувати на повну ентропію на виході.

Для наочного прикладу оцінки нехай теоретичний екстрактор матиме наступні значення: n_{in} = 256, n_{out} = 128, h_{in} = 128, nw = 128.

Тобто маємо певний криптографічний примітив з найвужчою внутрішньою шириною 128 бітів, вхідну послідовність довжиною 256 бітів з 128 бітами ентропії (0,5 на біт) і розмір виходу екстрактора 128 бітів. За алгоритмом 1 отримаємо h_{out}: $\psi = 5.88 \times 10^{-39}$, $\omega = 2.94 \times 10^{-39}$, $-\log_2(\max(\psi, \omega)) = 127.00000000000...92$.

Таким чином, ентропія на виході такого екстрактора буде майже повною, тобто 127 бітів ентропії на 128 бітів виходу.

Необхідно зауважити, що усічення виходу такого екстрактора є допустимим. В такому випадку оцінка ентропії зменшується до пропорції на виході (наприклад, при усіченні 8-бітного виходу з 6 бітами ентропії до 6 біт ентропія зменшується до 3/4 x 6 = 4.5 біт).

При використанні неперевірених (недовірених) NIST компонентів:

Для неперевірених NIST екстракторів ентропія на виході залежить від ентропії та розміру входу (h_{in} і n_{in}), розміру виходу (n_{out}), розміру найвужчої внутрішньої ширини (nw) та ентропії послідовного набору даних на виході екстрактора. Екстракція для збирання такого набору повинна виконуватися екстрактором, що оцінюється.

Отримання набору даних відбувається наступним чином:

Для валідації екстрактора повинен бути зібраний послідовний набір даних, що складається щонайменше з 1 000 000 послідовних виходів. Вихідні дані екстрактора повинні бути об'єднані в тому порядку, в якому вони були згенеровані, і розглядатися як двійковий рядок для цілей тестування. Ентропія такого набору даних повинна оцінюватися з використанням IID тестів для IID даних та non-IID тестів для non-IID даних. Проте, оскільки non-IID тести є доволі консервативними, вони можуть занижувати реальне значення min-ентропії [1]. Отримана оцінка ентропії на біт такого набору буде h'.

Отже, ентропія виходу екстрактора на неперевіреному криптографічному примітиві оцінюється за формулою

$$h_{out} = \min(Output\_Entropy(n_{in}, n_{out}, nw, h_{in}), 0.999 n_{out}, h' \times n_{out}) \qquad (2)$$

1.5. Криптографічні примітиви для розширення виходу з екстрактора

У більш нових версіях ядра Linux застосовується довірена NIST конструкція DRBG [6, 7] у якості генератора – за замовчуванням використовується CTR DRBG з AES-256. Тобто у якості компонента розширення використовується блоковий шифр у потоковому режимі. Для заповнення генератора (seeding) використовується seed, отриманий за рахунок екстракції з пулу ентропії. На основі цього генератор має змогу обчислювати псевдовипадкові значення з хорошими показниками рівномірності.

Зважаючи на це, можливим є застосування національних алгоритмів блокового шифрування Калина (ДСТУ 7624:2014) [5] та потокового шифрування Струмок (ДСТУ 8845:2019) [9] до виходу з екстрактора, для його розширення та формування випадкової послідовності довільної довжини.

Значення максимальної довжини генерації за запит, а також максимальної кількості таких запитів слід обмежити рекомендованими у [8] параметрами – кількість байт на запит «згенерувати» DRBG становить 2^{19} біт або 2^{16} байт, а кількість дозволених запитів до примусового повторного заповнення – 2^{48}.

У наступному розділі представлено практичні реалізації екстракторів на основі криптографічних примітивів та алгоритмів, що описані у п. 1.3.

2. Практична реалізація екстракторів на основі національних стандартів

Для практичної реалізації екстракторів необхідно повторити кроки, описані у розд. 1, використовуючи реальні ДШ та криптографічні примітиви.

2.1. Отримання даних з джерел шуму та оцінка їх ентропії за рекомендаціями NIST 800-90B

Згідно з вимогами [1] оцінка ентропії ДШ потребує послідовного набору даних зі щонайменше з 1 000 000 зразків. У якості ДШ були обрані наступні:

- нефізичне ДШ у вигляді запитів на переривання (IRQ) – у якості зразків використовується часова відмінність дельт часу між мітками часу переривань, що викликані пристроями введення (взаємодії з користувачем)/мережевими пристроями;
- нефізичне ДШ у вигляді дельт часу між мітками часу апаратних подій, отриманих з використанням спеціальних системних функцій: add_input_randomness (обробка подій введення), add_disk_randomness (обробка подій роботи з дисками) із застосуванням модулю kprobe.

2.1.1. Процес отримання даних з ДШ на основі IRQ

Для отримання ентропії з ДШ на основі джитеру дельт часу між запитами на переривання (IRQ) використовується ОС Linux та принцип загальної обробки IRQ в Linux. В основі лежить функція API драйвера високого рівня request_irq(), основне призначення якої виклик заданої функції (обробника) кожного разу при виникненні заданого типу переривання.

Для збору ентропії було створено модулі ядра. Ці модулі виконують перехоплення IRQ для пристроїв введення (миші) та мережевого адаптера. Як вказано вище, вони використовують request_irq(), до якої передається функція обробник. Ця функція обчислює дельту часу між перериваннями, а також відмінність цих часових дельт. З результуючої відмінності дельт часу беруться вісім наймолодших бітів (LSB) та поміщаються до буфера ентропії.

Для реалізації можливості зчитування даних з буфера ентропії у просторі користувача всередині модулів створюються відповідні символьні псевдопристрої. Під час використання користувач з програмного коду викликає ці пристрої і зчитує необхідну/доступну кількість ентропії у бінарний файл доти, доки не буде зібрано необхідну кількість. У випадку реалізації екстрактора необхідно зчитати більше ніж 1,048,576 зразків з ДШ, бо вибірка з джерела шуму відбувається по 8 біт, тобто побайтово. Така кількість обрана, щоб задовольняти умові з [1] щодо розміру вибірки і додатково, щоб зібрати повний 1 МБ «сирих» випадкових даних. У алгоритмах 2 та 3 наведено модулі у вигляді псевдокоду.

1 Ініціалізація буфера ентропії
2 Вибір необхідних пристроїв для обробки переривань (шляхом аналізу /proc/interrupts)
3 Визначення спінлоку та черги очікування (Спінлок (entropy_lock) використовується для захисту одночасного доступу до буфера між контекстами IRQ і процеса. Черга очікування (entropy_wait_queue) дозволяє блокувати читання, коли ще немає достатньої кількості даних)
4 Ініціалізація стану для обчислення джитеру (стан містить значення часу у мілісекундах, коли відбулося останнє переривання і значення останньої дельти між перериваннями)
5 Реалізація функції оцінки доступної ентропії у буфері
6 Реалізація функції обробки переривань – my_interrupt_handler (обчислення нової дельти часу та джитера з попередньою дельтою)
7 Реалізація функції ініціалізації модуля
8 Реалізація функції вивільнення модуля
9 Реалізація функції зчитування з буфера до простору користувача з використанням створеного символьного псевдопристрою
10 Визначення структури, яка відповідатиме за функції, що викликатимуться через стандартний узагальнений інтерфейс псевдопристрою

Алгоритм 2. Модуль ядра Linux для збору ентропії з використанням обробки IRQ

2.1.2. Процес отримання даних з ДШ на основі системних функцій з використанням модулю kprobe

Для отримання ентропії з ДШ на основі системних функцій add_input_randomness (введення), add_disk_randomness (диски) також було створено модулі ядра. У них використовується kprobe, завдяки якому при виклику потрібних системних функцій викликається і необхідний обробник. Як і у випадку IRQ обробник обчислює дельту часу між викликами системних функцій і відмінність цих часових дельт. І далі з результату також отримуються вісім наймолодших бітів (LSB), які записуються до буфера ентропії.

Всередині модулів також створюються символьні псевдопристрої для можливості використання даних, отриманих модулем у користувацькому просторі.

1 Ініціалізація буфера ентропії
2 Визначення kprobe для перехоплення викликів функцій
3 Визначення спінлоку та черги очікування
4 Ініціалізація стану для обчислення джитеру (стан містить значення часу у мілісекундах, коли відбулося останнє переривання і значення останньої дельти між перериваннями)
5 Реалізація функції оцінки доступної ентропії у буфері
6 Реалізація функції обробки
7 Реалізація функції ініціалізації модуля
8 Реалізація функції вивільнення модуля – __exit irq_entropy_exit
9 Реалізація функції зчитування з буфера до простору користувача
10 Визначення структури, яка відповідатиме за функції, що викликатимуться через стандартний узагальнений інтерфейс псевдопристрою

Алгоритм 3. Модуль ядра Linux для збору ентропії з використанням kprobe

З використанням кожного з описаних модулів було зібрано набір зразків (байтів) довжини > 1 000 000, згідно з рекомендаціями NIST 800-90B [1]. Для подальшого використання представлених ДШ у процесі екстракції необхідно оцінити значення ентропії отриманих наборів даних шляхом застосування тестів min-ентропії, визначених у [1].

2.1.3. Оцінка ентропії обраних ДШ на основі отриманих вибірок за рекомендаціями NIST 800-90B

Для отриманих з ДШ вибірок було виконано оцінку показників мінімальної ентропії згідно з рекомендаціями NIST, показників ентропії Шеннона та колізійної ентропії. Результати для кожного з використаних ДШ наведені на рис. 2–4.

Результати вказують на те, що вибірки з ДШ на основі мережевих подій та подій роботи з дисками мають дещо вищі значення ентропії, ніж ДШ на основі введення користувача. Проте показники є доволі високими і на достатньому рівні.

Нижче наведено псевдокод реалізацій екстракторів ентропії на основі національних стандартів.

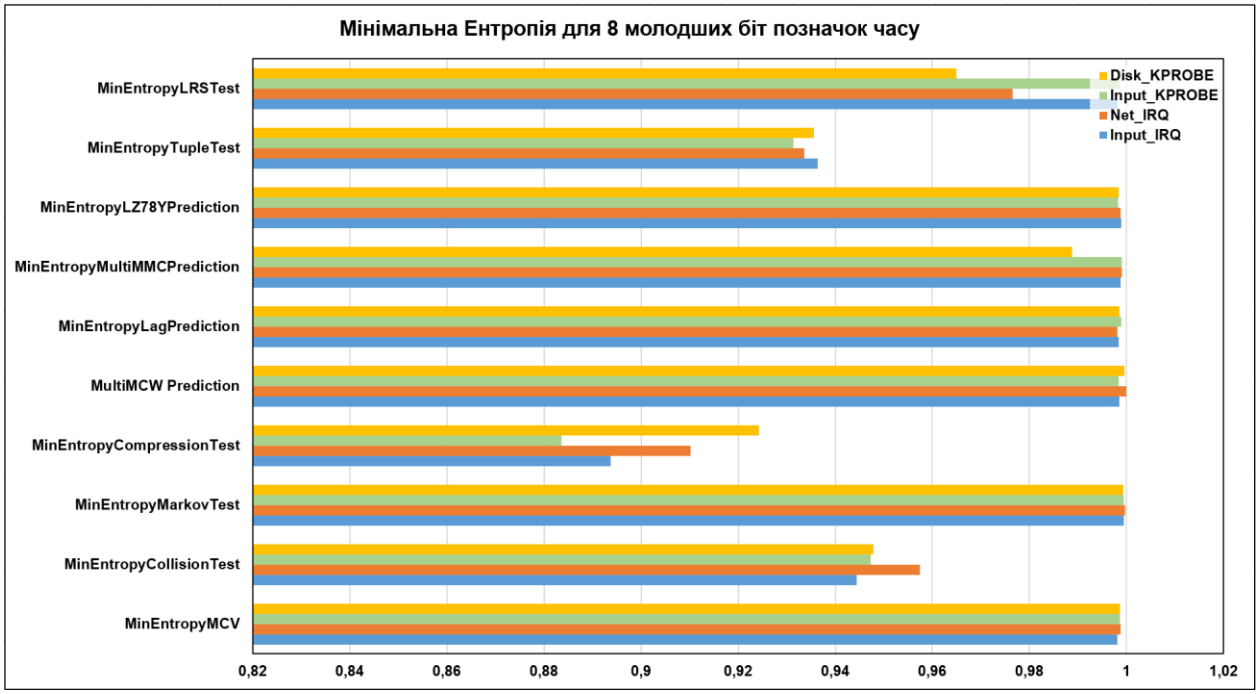

Рис. 2. Оцінки мінімальної ентропії обраних ДШ

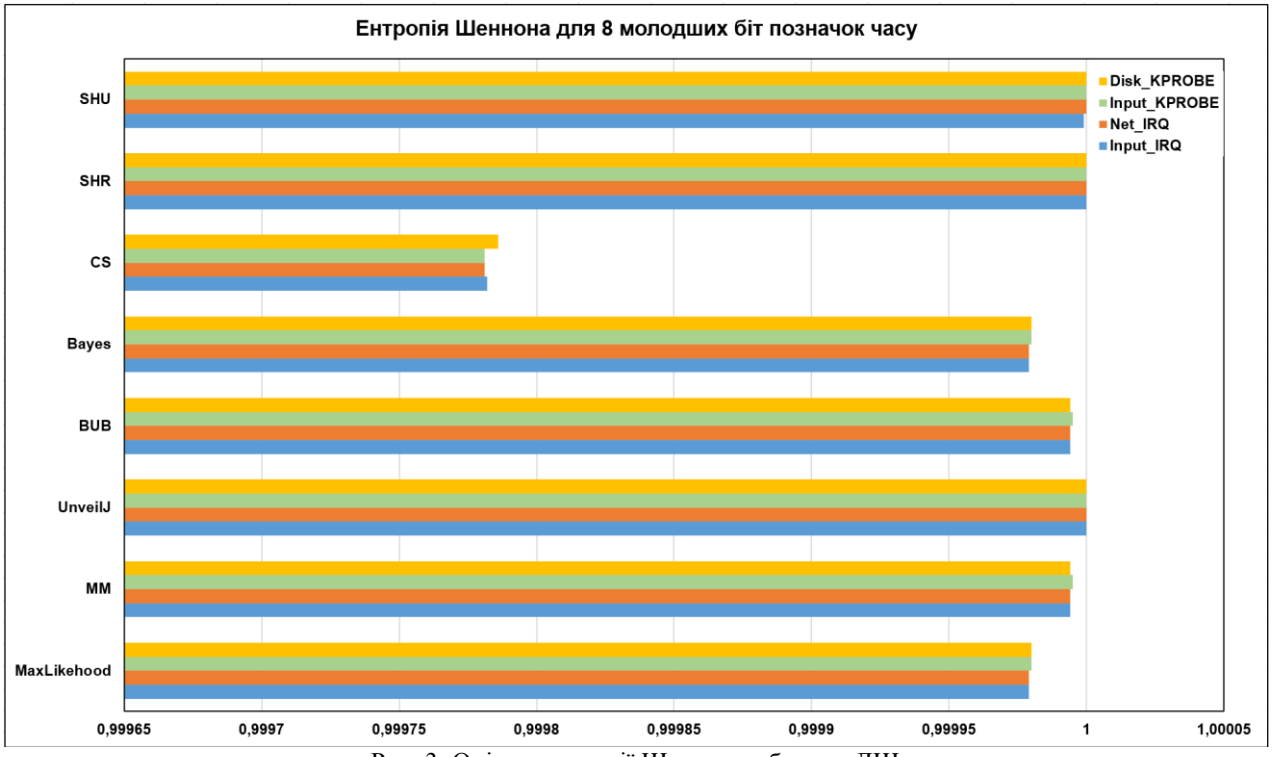

Рис. 3. Оцінки ентропії Шеннона обраних ДШ

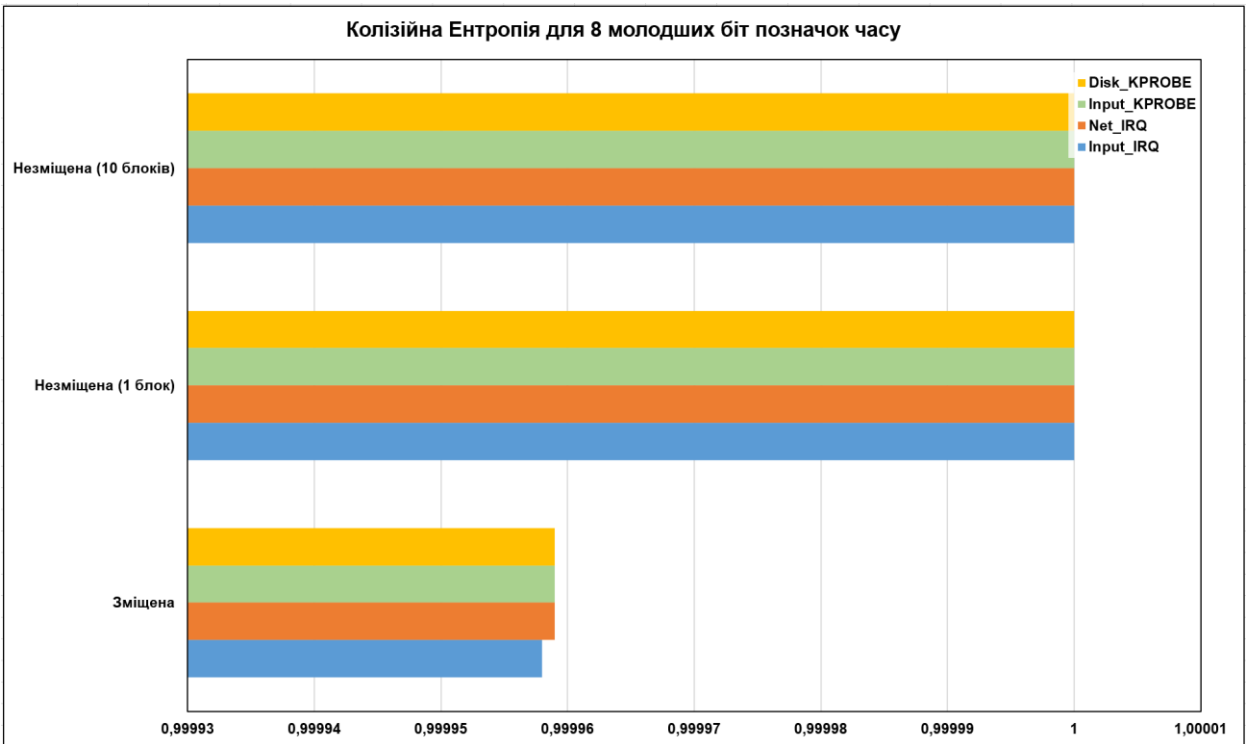

Рис. 4. Оцінки колізійної ентропії обраних ДШ

2.2. Реалізація екстрактора з використанням HMAC режиму геш функції Купина

Для реалізації екстрактора ентропії можливим є застосування криптопримітиву HMAC. Оскільки функція гешування Купина [4] не є перевіреною (довіреною) NIST, вихід екстрактора, побудованого з її використанням, потребуватиме додаткової оцінки за принципом, наведеним раніше.

Далі наведено реалізацію HMAC з використанням ДСТУ 7564:2014 «Купина» у вигляді псевдокоду. Для коректної реалізації HMAC важливо враховувати рекомендації [10, 11].

Вхід:
sk – ключ, відповідного рівня безпеки;
key_len – довжина ключа;
data – дані, для яких обчислюється MAC;
len – довжина даних.
Вихід:
mac – значення результуючого MAC необхідної довжини (в залежності від обраного режиму).
1 Виділення пам'яті для всіх даних, необхідних для обчислення MAC
buf[BLOCKSIZE + BLOCKSIZE / 2]
2 Виділення пам'яті для обчислення конкатенації ($K_0 \oplus ipad$) та data
temp[(BLOCKSIZE + len)]
3 if довжина ключа > BLOCKSIZE
Гешування ключа для отримання рядка та доповнення нулями до BLOCKSIZE
else if довжина ключа < BLOCKSIZE
Доповнення нулями до BLOCKSIZE
4 Виконання XOR для ключа та ipad
$K_0 \oplus ipad$ (ipad = байт 0x36 повторений BLOCKSIZE разів)
5 Конкатенація з вхідними даними
$K_0 \oplus ipad \,||\, data$

6 Застосування Kupyna256 або Kupyna512 для обчислення гешу в залежності від обраного режиму

Kupyna256_(512)(buf + BLOCKSIZE, temp, BLOCKSIZE + len);

7 Виконання XOR для ключа та opad

$K_0 \oplus opad$ (opad = байт 0x5C повторений BLOCKSIZE разів)

8 Об'днання з попередньо обчисленим гешем

9 Застосування Kupyna256 або Kupyna512 для обчислення гешу для $K_0 \oplus opad \,||\, H((K_0 \oplus ipad) \,||\, data)$

Kupyna512(buf, buf, BLOCKSIZE + BLOCKSIZE / 2);

Алгоритм 4. Kupyna-HMAC

Вихідними даними алгоритму є сформований над вхідними даними MAC з використанням криптографічного алгоритму HMAC.

Загальний принцип використання екстрактора на основі режиму HMAC геш функції Калина наступний:

- отримання з розглянутих у п. 2.1 ДШ такої кількості даних, щоб ентропія вхідного рядка була максимально можливою для виходу обраного екстрактора (ентропія = nw – найвужчій внутрішній ширині екстрактора);

- застосування екстрактора для екстракції з вхідних даних рівномірного виходу довжиною nw з ентропією близькою до або рівною nw (256 бітний рядок з ентропією ≈ 256).

Схематичне зображення процесу виконання екстракції показано на рис. 5.

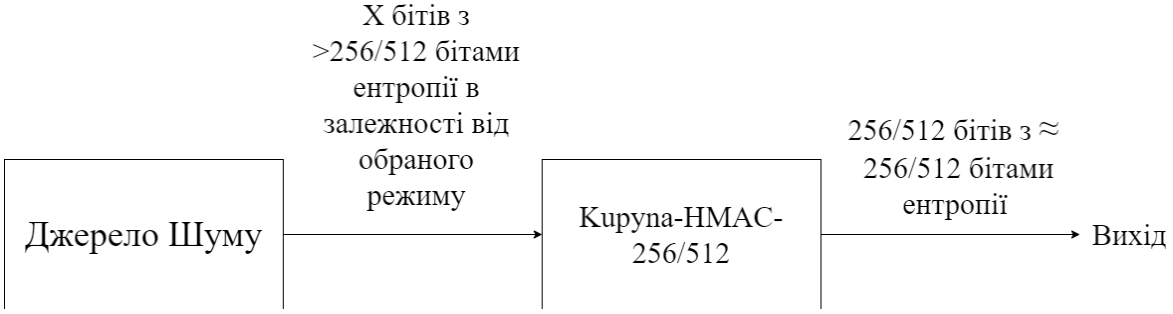

Рис. 5. Схематичне зображення екстрактору на основі Купина–HMAC

На виході такого екстрактора, як показано на рисунку, очікується рядок довжини, яка відповідає розміру виходу геш функції з показниками ентропії, що ≈ ентропії вхідних даних.

2.3. Реалізація екстрактора з використанням алгоритму Hash_df за вимогами NIST 800-90A та NIST 800-90B на основі геш функції Купина

Для реалізації екстрактора ентропії також можливим є використання алгоритму Hash_df (hash derivation function) – отримання похідної величини. Такий алгоритм схвалено для застосування у якості екстрактора ентропії у [1]. Сам алгоритм описано у [8]. У даному пункті наведено реалізацію Hash_df з використанням Купини. Для коректної реалізації Hash_df важливо враховувати рекомендації [8].

Перевагою даного алгоритму над HMAC є відсутність необхідності у випадковому ключі, оскільки Hash_df є безключовою функцією.

Вхід:
input – дані, для яких обчислюється геш;
input_len – довжина даних;
no_of_bits_to_return – кількість бітів, які повинен повернути Hash_df (Для Hash_df довжина виводу, зазначена в табл. 1, 2 (Розмір виходу геш функції), використовується як вхідний параметр no_of_bits_to_return для виклику Hash_df [8])
Вихід:
result – результат виконання Hash_df.
1 Обчислення кількості ітерацій як no_of_bits_to_return/outlen (Оскільки згідно з [1, 8] для екстрактора ентропії no_of_bits_to_return=outlen цей крок може бути пропущений)
len = ceil(no_of_bits_to_return/KUPYNA_OUTLEN);
2 Виділення пам'яті для результату
temp = malloc(len*KUPYNA_OUTLEN);
3 Встановлення лічильника рівним 1
counter = 1;
4 Обчислення довжини конкатенації $counter \| no\_of\_bits\_to\_return \| input$
concat_len = len + 4 + input_len;
5 Виділення пам'яті для конкатенації
concat = malloc(concat_len);
6 Виконання конкатенації потрібних елементів
concat[0] = counter;
bits_to_return[4];
u32_to_bytes(bits_to_return, no_of_bits_to_return);
memcpy(concat + 1, bits_to_return, 4);
memcpy(concat + 5, input_string, input_len);
7 Застосування гешування до конкатенації
Kupyna256/512(concat, concat_len, digest);
8 Повернення отриманого гешу
result = digest

Алгоритм 5. Kupyna-Hash_df

Загальний принцип використання екстрактора на основі алгоритму Hash_df з геш функцією Купина такий самий як і для режиму HMAC.

Схематичне зображення процесу виконання екстракції показано на рис. 6.

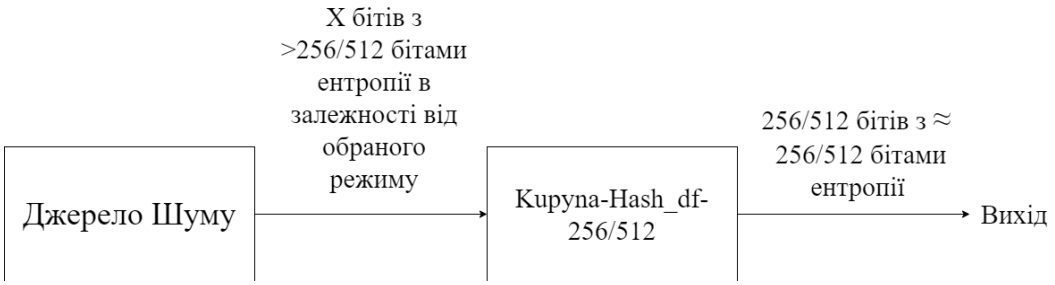

Рис. 6. Схематичне зображення екстрактору на основі алгоритму Hash_df з геш функцією Купина

На виході такого екстрактора, як показано на рисунку, очікується рядок довжини, яка відповідає розміру виходу геш функції з показниками ентропії, що ≈ ентропії вхідних даних.

2.4. Реалізація екстрактора з використанням CMAC режиму блокового шифру Калина

Ще одним можливим для використання у екстракторі криптографічним алгоритмом є CMAC. У стандарті Калина визначено використання алгоритму у такому режимі.У даному пункті наведено реалізацію CMAC з використанням Калини у вигляді псевдокоду.

Вхід:
input – дані, для яких обчислюється геш;
input_len – довжина даних;
q – довжина імітовставки (рекоменд. q = block_size)
ctx – контекст алгоритму Калина або (Раундові ключі (rnd_key), Розмір блоку (nb), Розмір ключа (nk), Кількість раундів (nr))
Вихід:
h – обчислена імітовставка.
1 Доповнення повідомлення нулями, допоки input_len mod block_size $\neq$ 0
2 if повідомлення було доповнене,
$K_\delta = 1$
else
$K_\delta = 0$
3 Встановлення бітів імітовставки C в нуль
4 Зашифрування K_δ
KalynaEncipher(((uint64_t*)K_sigma, ctx, (uint64_t*)K_sigma);
5 Розбиття повідомлення на n блоків довжини block_size
6 Цикл від 1 до n-1
Виконання XOR для C та оригінального повідомлення
Зашифрування отриманого XOR та присвоєння результату імітовставці
7 Для останнього блоку
Виконання XOR для C, оригінального повідомлення та K_δ
Зашифрування отриманого XOR та присвоєння результату імітовставці

Алгоритм 6. Kalyna-CMAC

Загальний принцип використання екстрактора на основі CMAC режиму алгоритму Калина такий самий як і для попередніх екстракторів – отримання необхідної кількості даних з ДШ і виконання екстракції.

Схематичне зображення процесу виконання екстракції показано на рис. 7.

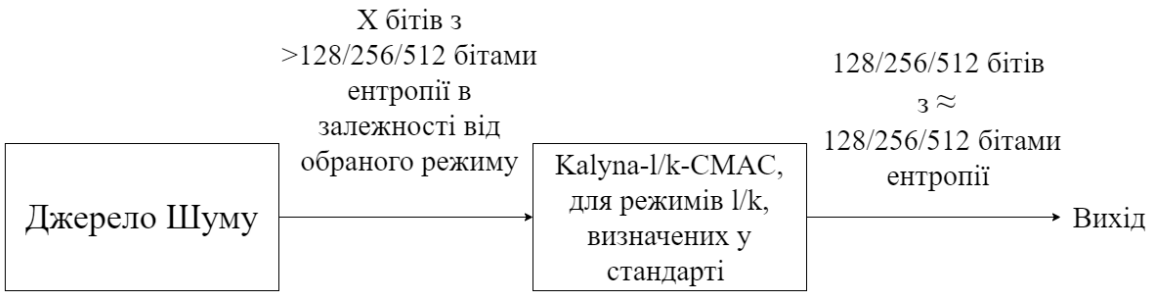

Рис. 7. Схематичне зображення екстрактору на основі Калина-CMAC

На виході такого екстрактора очікується рядок довжини, яка відповідає розміру блоку шифру з показниками ентропії, що ≈ ентропії вхідних даних.

Висновки

1. Проведено аналіз існуючих рекомендацій щодо побудування екстракторів ентропії на основі криптографічних примітивів. Цей процес потребує вибору відповідного джерела шуму, збору достатньо великого набору зразків з даного ДШ, для можливості подальшої

оцінки та виконання екстракції ентропії з використанням довіреного криптографічного компонента. Для усіх функцій і усіх режимів отримано прискорення не менше ніж в два рази, для більшості функцій та режимів прискорення більше ніж в три рази.

2. Виконано теоретичне обґрунтування можливості застосування національних криптографічних алгоритмів для виконання екстракції на основі подібності криптографічних примітивів та режимів, що застосовуються, до довірених NIST.

3. Наведено приклади практичної реалізації екстракторів на основі національних криптографічних алгоритмів з детальним описом та реалізацією всіх необхідних кроків. Зокрема показано практичну реалізацію процесу збирання зразків з нефізичних ДШ та оцінки показників ентропії цих ДШ за існуючими міжнародними рекомендаціями. Представлено практичну реалізацію кроку екстракції на основі національних алгоритмів гешування та блокового шифрування.

4. У подальших дослідженнях планується розробка алгоритму екстракції block_cipher_df на основі алгоритму блокового шифрування Калина, перевірка відповідності виходів екстракторів на основі національних криптографічних алгоритмів необхідним вимогам щодо ентропії, а також застосування компонентів розширення виходу екстракторів на основі національних алгоритмів для отримання послідовностей довільної довжини.

Список літератури:

1. Turan M.S., Barker E., Kelsey J., McKay K., Baish M., Boyle M. (2018). NIST SP 800-90B. Recommendation for the Entropy Sources Used for Random Bit Generation. Available at: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf
2. Rodríguez L., Madarro-Capó E., Legón-Pérez C., Rojas O., Sosa-Gómez G. (2021). Selecting an Effective Entropy Estimator for Short Sequences of Bits and Bytes with Maximum Entropy. Available at: https://doi.org/10.3390/e23050561
3. Skorski M. (2016). Improved Estimation of Collision Entropy in High and Low-Entropy Regimes and Applications to Anomaly Detection. Available at: https://eprint.iacr.org/2016/1035.pdf
4. ДСТУ 7564:2014. Інформаційні технології. Криптографічний захист інформації. Функція гешування. (2015).
5. ДСТУ 7624:2014. Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення. (2016).
6. Müller S., Mayer S., Dr. Holz C., Dr. Hohenegger A. (2022). Documentation and Analysis of the Linux Random Number Generator. Version 5.0. Available at: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Studies/LinuxRNG/LinuxRNG_EN_V5_0.pdf?__blob=publicationFile&v=3
7. Müller S. (2025). Linux/dev/random–ANewApproach. Available at: https://www.chronox.de/lrng/releases/v59/lrng-v59.pdf
8. Barker E., Kelsey J. (2015). NIST SP 800-90A Rev. 1. Recommendation for Random Number Generation Using Deterministic Random Bit Generators. NIST. Available at: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf
9. ДСТУ 8845:2019. Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного потокового перетворення. (2019).
10. FIPS PUB 198-1. (2008). The Keyed-Hash Message Authentication Code (HMAC). Available at: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.198-1.pdf
11. Krawczyk H., Bellare M., Canetti R. (1997). RFC 2104. HMAC: Keyed-Hashing for Message Authentication. Available at: https://datatracker.ietf.org/doc/html/rfc2104

Надійшла до редколегії 23.10.2025

Відомості про авторів:

Дерев'янко Ярослав Андрійович – Харківський національний університет імені В. Н. Каразіна, аспірант кафедри кібербезпеки інформаційних систем, мереж і технологій, навчально-наукового інституту комп'ютерних наук та штучного інтелекту, АТ «Інститут Інформаційних технологій», науковий співробітник-консультант; Україна; e-mail: yarik0009258@gmail.com; ORCID: https://orcid.org/0000-0002-3290-3373

Горбенко Дмитро Юрійович – Харківський національний університет імені В.Н. Каразіна, студент кафедри кібербезпеки інформаційних систем, мереж і технологій, навчально-наукового інституту комп'ютерних наук та штучного інтелекту; АТ «Інститут інформаційних технологій», молодший інженер-програміст; Україна; e-mail: jsciitua@gmail.com