SYSTEMS AND METHODS OF INFORMATION PROTECTION
СИСТЕМИ І МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ

УДК 004.056.5 **DOI:10.30837/rt.2024.1.216.01**

І.Д. ГОРБЕНКО, д-р техн наук, А.М. ОЛЕКСІЙЧУК, д-р техн. наук,
О.Г.КАЧКО, канд. техн. наук, Я.А. ДЕРЕВ'ЯНКО

ДОСЛІДЖЕННЯ МЕТОДІВ ТА АЛГОРИТМІВ ДЛЯ ГЕНЕРАЦІЇ (ПСЕВДО)ВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ НАД ДОВІЛЬНИМ АЛФАВІТОМ

Вступ

Робота присвячена дослідженню алгоритмів для генерації (псевдо)випадкових послідовностей над довільним алфавітом, оцінці їх складності (часової та ємнісної). Це дослідження важливе в зв'язку з тим, що для сучасних постквантових алгоритмів застосовують саме такі послідовності для генерації ключів та випадкових компонентів електронного підпису та інкапсуляції ключів, наприклад [1 – 3].

1. Генерація двійкової послідовності

Для побудови генераторів, що виробляють випадкові чи псевдовипадкові послідовності над алфавітом потужності $q>1$, в роботі застосовують підхід, згідно з яким спочатку формуються двійкові випадкові чи псевдовипадкові послідовності, які задовольняють відомим статистичним характеристикам, з яких потім певним чином отримуються послідовності q-х символів.

Для генерації таких двійкових послідовностей можна застосовувати різні засоби, наприклад:

- спеціальні пристрої, наприклад серії «Гряда» [24], а також квантові генератори серії QRNG [25];
- кросплатформні засоби, основані на застосуванні команд процесора, наприклад команди *rdrand* для процесорів *Intel*, яка, в залежності від обраного режиму, генерує випадкову послідовність завдовжки 16, 32 або 64 біта;
- спеціальні функції операційних систем, наприклад для генерації випадкових даних в ОС WINDOWS, застосовують функцію криптопровайдера CryptGenRandom, в ОС LINUX – випадкові дані з заданою кількістю байтів зчитуються з пристрою /dev/random.

Для подальшої роботи були згенеровані двійкові послідовності завдовжки 13 мільйонів байтів кожним зі способів і виконано їх тестування за допомогою статистичних тестів NIST (NIST 800-22).

2. Генератори послідовностей з довільним алфавітом

2.1. Означення основних понять

Скінченним автоматом називають впорядкований набір $A=(X,S,Y,h,f)$, де X,S,Y – скінченні множини, $h: X\times S\to S$ та $f: X\times S\to Y$ – відображення. Множини X,Y та S називаються *вхідним алфавітом*, *вихідним алфавітом* та *множиною станів* (або *внутрішнім алфавітом*) автомата A. Відображення h та f називають відповідно *функцією переходів* та *функцією виходів* цього автомата. Автомат називається *автономним*, якщо функції h і f не залежать від першого аргументу. В цьому випадку множину X не згадують та вважають, що $h: S\to S$, $f: S\to Y$.

Згідно із загальновизнаним означенням, *генератором псевдовипадкових послідовностей* (*ПВП*) називають скінченний автономний автомат $A=(S,Y,h,f)$, де S та Y позначають відповідно внутрішній і вихідний алфавіти автомата A, $h: S\to S$ і $f: S\to Y$ є, відповідно,

функціями переходів і виходів цього автомата. Генератор ПВП функціонує в дискретні моменти часу або такти, виробляючи за довільним елементом $s_0 \in S$ (*початковим станом* автомата A) *внутрішню послідовність* $s_0, s_1, \ldots$, де $s_{i+1} = h(s_i)$ та *вихідну послідовність* (*гаму*) $\gamma_i = f(s_i)$, $i = 0, 1, \ldots$. Для будь-якого натурального L відображення $\Gamma_L : s_0 \mapsto \gamma_0, \gamma_1, \ldots, \gamma_{L-1}$, де $\gamma_i = f(h^i(s_0))$, $i \in \overline{0, L-1}$, $s_0 \in S$, називається *гамоутворювальним відображенням генератора* A*, обмеженим на довжині* L (див. [4]).

Генератор випадкових послідовностей визначається як дискретне джерело, що виробляє послідовність випадкових величин $\xi_1, \xi_2, \ldots$, які приймають значення у певній скінченній множині Y. Такий генератор задається набором усіх скіченновимірних розподілів ймовірностей цих випадкових величин, тобто розподілів вигляду $\{\mathbf{P}(\xi_{i_1} = a_1, \ldots, \xi_{i_k} = a_k) : a_1, \ldots, a_k \in Y\}$, де $1 \le i_1 < \ldots < i_k$, $k = 1, 2, \ldots$

Як відомо, основною вимогою до якісного генератора ПВП є псевдовипадковість. Нагадаємо формальне означення цього поняття [5 – 7].

Розглянемо таку гру між Дослідником та Криптоаналітиком.

1. Дослідник генерує випадковий рівноймовірний початковий стан $s_0 \in S$ генератора $A = (S, Y, h, f)$. Після цього він з ймовірністю $1/2$ вибирає або відрізок гами $\gamma = \gamma_0, \gamma_1, \ldots, \gamma_{L-1}$, вироблений генератором за початковим станом s_0, або випадкову рівноймовірну послідовність довжини L над алфавітом Y.

2. Дослідник передає Криптоаналітику послідовність $y = y_0, y_1, \ldots, y_{L-1}$, отриману в результаті вибору, зробленого на кроці 1.

3. Криптоаналітик, використовуючи будь-який статистичний критерій, розв'язує задачу перевірки гіпотези $H_0 : y = \gamma$; проти альтернативи $H_1 : y$ є суто випадковою послідовністю.

Нехай $T > 0$, $\varepsilon \in (0, 1/2)$. За означенням (див. [4]) генератор A називається (T, L, ε)-*псевдовипадковим*, якщо будь-який критерій для розрізнення зазначених вище гіпотез H_0 та H_1, що має середню ймовірність помилки не вище ε, використовує принаймні T (умовних) операцій (нагадаємо, що середня ймовірність помилки критерію визначається за формулою $1/2(\Pr(H_1 \mid H_0) + \Pr(H_0 \mid H_1))$, де $\Pr(H_1 \mid H_0)$ та $\Pr(H_0 \mid H_1)$ – ймовірності помилок першого та другого роду відповідно).

Іншими словами, генератор гами є (T, L, ε)-псевдовипадковим, якщо не існує способу відрізнити його вихідну послідовність довжини L, отриману при випадковому рівноймовірному початковому стані, від суто випадкової послідовності такої ж довжини над алфавітом Y з середньою ймовірністю помилки не вище ε, використовуючи менше ніж T операцій.

Аналогічно визначається поняття стійкового генератора випадкових послідовностей. Припустимо, що Криптоаналітик має доступ до оракула, який з імовірністю $1/2$ є визначеним генератором (гіпотеза H_0) та з такою ж ймовірністю є генератором випадкових рівноймовірних послідовностей над алфавітом Y (гіпотеза H_1). Тоді визначений генератор називається (T, L, ε)-*псевдовипадковим*, якщо будь-який критерій для розрізнення зазначених гіпотез, що має середню ймовірність помилки не вище ε, використовує принаймні T (умовних) операцій.

2.2. Допустимі перетворення двійкових послідовностей

Нехай Q – скінченна множина потужності $q > 1$, A – алгоритм, що переробляє деякі послідовності бітів в окремі символи над алфавітом Q. Назвемо такий алгоритм (та перетворення, яке він реалізує) *допустимим*, якщо випадкові рівноймовірні послідовності бітів перетворюються у випадкові рівноймовірні символи.

2.2.1. Алгоритм A_1 (Метод відбору[1]).

Нехай l – найменше натуральне число таке, що $q \le 2^l$, і алгоритм A_1 визначається наступним чином.

1. Згенерувати двійковий вектор довжини l.

2. Якщо ціле число, двійковим записом якого є згенерований вектор, менше за q, повернути це число як результат і завершити роботу. Інакше – повернутися до кроку 1.

Твердження 1. Алгоритм A_1 є допустимим.

Доведення. Нехай $x = x_1, x_2, \ldots$ – послідовність незалежних випадкових рівноймовірних двійкових векторів довжини l, $y = \mathrm{A}_1(x)$ – випадковий символ, що формується за нею з використанням алгоритму A_1. Для будь-якого $i \in \overline{0, q-1}$ позначимо U_i подію, яка полягає в тому, що $y = i$. Ця подія відбувається тоді й тільки тоді, коли існує невід'ємне ціле число k таке, що послідовність двійкових чисел, з якої отримується символ y, має вигляд $\tilde{x}_1, \ldots, \tilde{x}_k, y$, де $\tilde{x}_1, \ldots, \tilde{x}_k \ge q$. Отже,

$$\mathbf{P}\{U_i\} = 2^{-l} \sum_{k=0}^{\infty} (1 - 2^{-l} q)^k = 2^{-l} \frac{1}{1 - (1 - q2^{-l})} = q^{-1},$$

що й треба було довести.

Зауважимо, що середнє число бітів (випадкової рівноймовірної) вхідної послідовності, потрібних для отримання одного q-го символу за допомогою алгоритму A_1, дорівнює

$$lq2^{-l} \sum_{k=0}^{\infty} (k+1)(1 - 2^{-l} q)^k = 2^l lq^{-1} < 2l = 2\lceil \log q \rceil.$$

2.2.2. Алгоритм A_2. (Метод швидкої гральної кістки [8]).

Наступний алгоритм A_2 вдосконалює попередній та надає змогу зменшити кількість бітів, потрібних для формування одного q-го символу.

1. Покласти $v = 1$, $c = 0$.

2. Виконувати у циклі такі обчислення:

– згенерувати випадковий рівноймовірний біт b та покласти $v = 2v$; $c = 2c + b$;

– якщо $q \le c < v$, покласти $v = v - q$, $c = c - q$;

– якщо $c < q \le v$, повернути значення c як результат і завершити роботу.

Отже, наведений алгоритм знаходить за послідовністю незалежних випадкових рівноймовірних бітів q-й символ c, який формується в циклі при першому виконанні умови $c < q \le v$.

В [8] доведено таке твердження.

Твердження 2. Алгоритм A_2 є допустимим. При цьому для будь-якого $\alpha > 0$ середнє число бітів, потрібних для формування одного q-го символу за допомогою цього алгоритму, асимптотично дорівнює

$$\log q + \frac{1}{2} + \frac{1 - \gamma}{\ln 2} + F(\log q) + O(q^{-\alpha}),\ q \to \infty,$$

де $\gamma = 0{,}5772\ldots$ – константа Ойлера, F – певний періодичний тригонометричній поліном.

З практичного погляду становлять інтерес такі алгоритми, що не є допустимими, але перетворюють випадкові рівноймовірні послідовності бітів у символи, розподіл яких є близьким до рівномірного на множині Q.

Для будь-якого $\varepsilon \in (0, 1)$ назвемо алгоритм A *ε-допустимим*, якщо для випадкової рівноймовірної двійкової послідовності x символ $y = \mathrm{A}(x)$ задовольняє умову

[1] Застосовують в алгоритмах [1 – 3]

$$|\mathbf{P}(y=i)-q^{-1}|\le\varepsilon,\ i\in\overline{0,q-1}. \quad (1)$$

2.2.3. Алгоритм A_3. (Метод *Уокера* [9, с. 144]).

Розглянемо алгоритм A_3, що визначається наступним чином:

1. Згенерувати вектор $x\in\{0,1\}^l$.

2. Обчислити $y=\lfloor 2^{-l}q\hat{x}\rfloor$, де $\hat{x}$ – ціле число з двійковим записом x.

Твердження 3. Алгоритм A_3 є 2^{-l}-допустимим.

Доведення. Для кожного $i\in\overline{0,q-1}$ мають місце такі співвідношення:

$$\lfloor 2^{-l}q\hat{x}\rfloor=i\Leftrightarrow 2^l qi\le\hat{x}<2^l q(i+1)\Leftrightarrow\lceil 2^l qi\rceil\le\hat{x}<\lceil 2^l q(i+1)\rceil.$$

Отже, якщо $\hat{x}$ є випадковим числом з рівномірним розподілом на множині $\{0,1,...,q-1\}$, то

$$\mathbf{P}(y=i)=2^{-l}\left(\lceil 2^l q(i+1)\rceil-\lceil 2^l qi\rceil\right)=q^{-1}+2^{-l}\theta,$$

де $|\theta|<1$, що й треба було довести.

У зв'язку з твердженням 3 є важливим питання про вибір параметра l або, іншими словами, про те, наскільки малим повинно бути значення ε у формулі (1) для того, щоб вважати розподіл символу $y=\mathrm{A}_3(x)$ "практично рівномірним".

Природна відповідь на це питання полягає в тому, що розподіл ймовірностей на скінченній множині є "практично рівномірним", якщо його неможливо надійно відрізнити від суто рівномірного за допомогою оптимального (байєсівського) критерію, маючи вибірку з цього розподілу певного обмеженого обсягу.

Наступне твердження є наслідком теореми 3 в [10].

Твердження 4. Найменший обсяг вибірки, необхідний для розрізнення із середньою ймовірністю помилки $\delta\in(0,1/2)$ розподілу ймовірностей $P=(P(x):x\in Q)$ на скінченній множині Q потужності $q>1$ та рівномірного розподілу на цій множині, становить

$$t\ge\frac{2(1-h(\delta))\ln 2}{\Delta(P)},$$

де $h(\delta)=-\delta\ln\delta-(1-\delta)\ln(1-\delta)$, $\Delta(P)=q^{-1}\sum_{x\in Q}(qP(x)-1)^2$.

Наслідок. Найменший обсяг вибірки, необхідний для розрізнення 2^{-l}-допустимого та рівномірного розподілів ймовірностей на множині з q елементів, становить не менше ніж $\frac{2(1-h(\delta))\ln 2}{q^2 2^{-2l}}$, тобто є величиною порядку $2^{2l}q^{-2}$.

Зокрема, якщо довжина вихідної послідовності генератора, до якої має доступ криптоаналітик, не перевищує 2^{64} q-х символів, то метод Уокера можна використовувати на практиці при $l=32+\lceil\log q\rceil$.

3. Обґрунтування криптографічних властивостей генераторів над довільним алфавітом, що будуються за допомогою допустимих перетворень

Розглянемо генератор Γ_{A} (псевдо)випадкових послідовностей над алфавітом Q, який визначається за генератором двійкових послідовностей Γ та допустимим алгоритмом A таким чином: якщо $x=x_1,x_2,...$ – вихідна послідовність генератора Γ, то $y=\mathrm{A}(x)$ є вихідною послідовністю генератора Γ_{A} (тут и далі $\mathrm{A}(x)$ позначає послідовність, члени якої отри-

муються шляхом послідовного застосування алгоритму A до відповідних відрізків послідовності x, які не перетинаються).

Твердження 5. Якщо Γ є (T, L, ε)-псевдовипадковим генератором, то Γ_{A} є $(T-t, L', \varepsilon)$-псевдовипадковим, де L' – мінімальна довжина усіх скінченних послідовностей $y = \mathrm{A}(x)$, що отримуються з двійкових послідовностей x довжини L, t – максимальна часова складність обчислення таких послідовностей y за допомогою алгоритму A.

Доведення. Припустимо, що існує критерій D, якій відрізняє вихідну послідовність довжини L' генератора Γ_{A} із середньою ймовірністю помилки не вище ніж ε, використовуючи не більше ніж $T-t$ операцій. Побудуємо критерій $\tilde{D}$, який розв'язує аналогічну задачу для генератора Γ.

Нехай x – двійкова послідовність довжини L, що є або вихідною послідовністю генератора Γ (гіпотеза H_0) або суто випадковою (гіпотеза H_1). Тоді критерій $\tilde{D}$ полягає в обчисленні послідовності $y = \mathrm{A}(x)$ (довжини не менше за L') та застосуванні до неї критерію D. Якщо D приймає висновок про справжність гіпотези H_ν, то такий саме висновок приймає і критерій $\tilde{D}$, $\nu = 1, 2$. Зрозуміло, що $\tilde{D}$ виконує не більше ніж $(T-t)+t = T$ операцій.

Оцінимо середню ймовірність помилки цього критерію.

Нехай справджується гіпотеза H_0 і $\tilde{D}$ припускається помилки. Тоді припускається помилки критерій D, приймаючи послідовність $y = \mathrm{A}(x)$ за суто випадкову. Отже, $\Pr_{\tilde{D}}(H_1 \mid H_0) \le \Pr_D(H_1 \mid H_0)$, тобто ймовірність помилки першого роду критерію $\tilde{D}$ не перевищує ймовірність помилки першого роду критерію D.

Нехай зараз справджується гіпотеза H_1 і $\tilde{D}$ припускається помилки. Тоді послідовність x є суто випадковою, а критерій D оголошує не суто випадковою послідовність, яка обчислюється за x з використанням алгоритму A. Але тоді внаслідок допустимості алгоритму A критерій D припускається помилки. Отже, $\Pr_{\tilde{D}}(H_0 \mid H_1) \le \Pr_D(H_0 \mid H_1)$, тобто ймовірність помилки другого роду критерію $\tilde{D}$ не перевищує ймовірність помилки другого роду критерію D.

Таким чином, на підставі наведених міркувань середня ймовірність помилки критерію $\tilde{D}$ не перевищує ε, що однак суперечить припущенню про (T, L, ε)-псевдовипадковість генератора Γ.

Твердження доведено.

Отримане твердження свідчить про те, що генератор Γ_{A}, отриманий з вхідного генератора Γ за допомогою допустимого алгоритму A, є не гірше за Γ.

4. Експериментальне дослідження алгоритмів $\mathbf{A_1}$, $\mathbf{A_2}$, $\mathbf{A_3}$

Алгоритми відрізняються кількістю бітів (розміром двійкової послідовності), яка може бути постійною для заданого алфавіту або залежати від конкретних бітів послідовності.

Для забезпечення отримання рівноймовірних символів алгоритми можуть виключати застосовування деяких послідовностей, коректувати їх довжину

Реалізації алгоритмів передбачають отримання послідовності зі значеннями в інтервалі [0..q-1], де q – параметр алгоритму, визначає потужність алфавіту. Алгоритми легко перетворити в алгоритми генерації вектору з елементами в діапазоні $\left[-\frac{q}{2}..\frac{q}{2}\right]$, якщо від кожного елементу вектору відняти значення $\frac{q}{2}$, або навпаки, значення $\frac{q}{2}$ віднімати від елементу.

4.1. Алгоритм A_1. (*Метод відбору*)

Нехай l – найменше натуральне число таке, що $q \le 2^l$, і алгоритм A_1 визначається наступним чином. Довжина двійкової послідовності для генерації одного значення завжди дорівнює l, але, якщо ціле число, двійковим записом якого є ця послідовність, не менше ніж q, то послідовність не застосовується.

Алгоритм генерації заданої кількості чисел для довільного алфавіту:

Вхід

q – визначає потужність алфавіту, зазвичай, $q \neq 2^k$
n – кількість чисел, які треба сформувати
Вихід
R – вектор, який складається з n чисел, значення яких в діапазоні [0..q – 1].
1 Визначення значення $l = \lceil log_2 q \rceil$, $m = 2 * l$
2 Цикл для формування компонентів вектору
j := 0 Номер елементу вектору
parts := 0 Кількість порції завдовжки l в наявній бітовій послідовності
k:= 0 Номер поточної порції
while j < n do
if k = parts then Бітової послідовності не було або вона вичерпана
$bytes = \left\lceil \frac{m*(n-j)}{8l} \right\rceil * l$
b := create_b(bytes)
k:= 0 Номер поточної порції
$parts = \frac{bytes*8}{l}$ *Кількість порцій завдовжки l*
end if
*part:= subst(b, k*l, l)* *Виділення наступної порції*
value:= number (part) *Формування значення для part*
if value < q then *Значення підходить?*
R[j]:=value;
j:=j + 1
end if
k:=k + 1
end while

Для оптимізації алгоритму в разі, коли значення l фіксоване, крок 1 алгоритму можна не виконувати, а задати значення l як константу.

Для зменшення розміру необхідної бітової послідовності можна враховувати імовірність «не відхилення» значення, тоді m обчислюється за формулою: $m = \left\lceil \frac{2^l}{q} \right\rceil$. Експериментальні результати (див. п. 4.4), отримані з урахуванням цього кроку.

Для оптимізації кроку виділення бітів послідовності замість виділення окремих бітів виділяється уся порція, в якості розміру порції вибирають число, кратне розміру байту або напівбайту, а потім зайві біти, які відповідають старшим розрядам числа, якщо вони є, встановлюють в 0. Такий прийом застосовують в алгоритмах 1, 3.

4.2. Алгоритм A_2. (*Метод швидкої грального кістки)*

Згідно цьому методу для числа обирається змінна кількість бітів від $\lceil log_2 q \rceil$. Середня кількість бітів дорівнює приблизно $\lceil log_2 q \rceil + 1$.

Вхід

q – визначає потужність алфавіту, зазвичай, $q \neq 2^k$
n – кількість чисел, які треба сформувати
Вихід
R – вектор, який складається з n чисел, значення яких в діапазоні [0..q – 1].
1 Визначення значення $l = \lceil log_2 q \rceil$, $m = l + 1$
2 Поточна кількість сформованих бітів len_bit = 0
3 Поточна номер біту k = 0
4 Цикл для формування компонентів вектору
j := 0 Номер елементу вектору

```
while j < n do
        v = 1, c = 0;                   Ініціалізація змінних
        Цикл генерації поточного елементу вектору
        R[j] = q;
        while (R[j] >=q)
                Генерація послідовності, якщо вона не була сформована або вичерпана
                        if   len_bit = k then
                                 len_bit = (m * (n – j) + 7) / 8 * 8;  Бітів
                                len_byte = len_bit / 8;                          Байтів
                                b := create_b(bytes)
                                k:= 0     Номер поточного біту послідовності
                        end if
                        v = 2 * v;
                        c = 2 * c + GET_BIT(b, k);
                        ++k;
                        if v >= q then
                                if c < q then        Знайшли значення
                                        R[j] = c;
                                else
                                        v -= q;
                                        c -= q;
                                end if
                        end if
                end while
                j:=j + 1
        end while
        part:= subst(b, k*l, l)      Виділення наступної порції
        value:= number (part)     Формування значення для part
        if value < q then             Значення підходить?
                R[j]:=value;
                j:=j + 1
        end if
        k:=k + 1
end while
```

4.3. Алгоритм A_3. (*Метод Уокера*)

Цей метод передбачає застосування бітової послідовності фіксованої довжини l, для відповідного числа обчислюється значення компоненту вектору за формулою

$$R\,[i] = \frac{q*Number\,(b)}{2^l}, \qquad (2)$$

де b – бітова послідовність завдовжки l бітів , а $Number\,(b)$ – відповідне ціле число.

Значення l обирається за формулою $l = \lceil log_2 q \rceil + 16$.

Вхід

q – визначає потужність алфавіту, зазвичай, $q \neq 2^k$

n – кількість чисел, які треба сформувати

Вихід

R – вектор, який складається з n чисел, значення яких в діапазоні [0..q – 1].

1 Визначення значення $l = \lceil log_2 q \rceil + 16$

2 Генерація бітової послідовності b завдовжки l * n

$$bytes = \left\lceil \frac{n * l}{8} \right\rceil$$

```
        b := create_b(bytes)
```

3 Цикл для формування компонентів вектору

```
j := 0                          Номер елементу вектору
k:= 0                           Номер поточної порції
while j < n do
        part:= subst(b, k*l, l)          Виділення наступної порції
        value:= number (part)     Формування значення для part
```

$R\,[i] = \frac{q*value}{2^l}$

```
        j:=j + 1
end while
```

4.4. Часові та просторові характеристики алгоритмів

Експериментальне дослідження виконувалося для q = 3329 [1] та q = 8380417 [2, 3], які застосовуються для генерації відкритих параметрів та для q = 5, 9 [2, 3] для генерації особистих ключів. Для кожного значення q для трьох алгоритмів формувалося n = 100000 чисел. Отримані числові послідовності тестувалися відповідно до статистичних алгоритмів. Усі тести пройдено.

Вимірювався час генерації послідовності (часові характеристики) та кількість потрібних байтів двійкової послідовності. Для збільшення точності виміру залежності часу від алгоритму для усіх алгоритмів двійкова послідовність генерувалася завчасно і час її генерації не враховувався.

Для першого та третього алгоритму розглянуто два варіанти.

1. Виділення окремих бітів ланцюжка;

2. Виділення цілого ланцюжка

Далі розглянуто окремо результати для генерації відкритих даних, для яких суттєвим є не тільки час генерації, але і обсяг потрібних даних, а потім для генерації приватних даних, для яких найбільш суттєвим є незалежність часу генерації

4.4.1. Результати для генерації відкритих даних.

Результати виміру для варіанту 1 представлені в табл. 1, а для варіанту 2 – в табл. 2. В табл. 2 наведено також результати для Алгоритму A_2 для полегшення порівняння.

Таблиця 1

Генерація відкритих даних: Варіант 1

Алгоритм	Bytes	Tacts	bytes	Tacts
	q = 3329		q = 8380417	
Алгоритм A_1	187500	22829550	300012	36210220
Алгоритм A_2	162500	4894893	300000	6340177
Алгоритм A_3 (l = 16)	350000	42039398	300000	57110407

Таблиця 2

Генерація відкритих даних: Варіант 2

Алгоритм	Bytes	Tacts	Bytes	Tacts
	q = 3329		q = 8380417	
Алгоритм A_1	187500	586519	495742	139929
Алгоритм A_2	162500	4894893	300000	6340177
Алгоритм A_3 (l = 16)	350000	124457	400000	104412

Для кожного значення q в таблиці перша колонка задає кількість байтів в двійковій послідовності, а колонка 2 – кількість тактів для генерування цієї послідовності та перетворення її в відповідну послідовність з заданим алгоритмом.

Для варіанту 1, в якому перетворення виконується побітно, і за часом, і за довжиною послідовності, кращім є Алгоритм A_2.

Для варіанту 2, в якому виділення виконується порціями, а також розмір порції може збільшуватись для більш ефективного виділення, для алгоритмів 1 та 3 отримали суттєве покращення не менше ніж в 39 разів. Для обох значень q = 3329 та q = 8380417 найбільш ефективним є алгоритм 3. Алгоритм A_3 є найкращим і з боку константного часу, тому що не треба відбирати дані.

Необхідно замітити, що автори алгоритмів [1 – 3] застосовували варіант 2, але Алгоритм A_1, а не Алгоритм A_3.

4.4.2. Результати для генерації приватних даних.

Далі наведено результати на прикладі параметрів для генерації векторів s1, s2 для алгоритмів [2, 3]. В експериментах застосовувались:

Алгоритм A_1 для варіанту 2, саме такий метод застосовували автори відповідних стандартів;

Алгоритм A_3 для варіанту 2, саме такий метод забезпечує незалежність часу генерації від конкретної послідовності, що є особливо важливо для компонентів приватних ключів. Для алгоритму 3 застосовують значення l = 16 та l = 32.

Як і в попередніх експериментах генерується послідовність розміром 10000 елементів. Результати наведені в табл. 3.

Таблиця 3

Генерація приватних даних: Варіант 2

Алгоритм	Bytes	Tacts	Bytes	Tacts
	q = 15 (η = 2)		q = 9 (η = 4)	
Алгоритм A_1	79988	5082205	87500	4120856
Алгоритм A_3, l = 16	250000	368615	250000	368320
Алгоритм A_3, l = 32	450000	365166	450000	342589

Алгоритм A_1 потребує суттєво більшого часу, ніж Алгоритм 3 але вимагає послідовність значно меншого розміру, тобто більшого часу для її генерації, але цей алгоритм не забезпечує константність часу.

Значення l не впливає на розмір послідовності і практично не впливає на час генерації.

Р е к о м е н д а ц і я . Для генерації приватних даних застосовувати Алгоритм 3 замість алгоритму 1 при l = *32*.

5. Статистичне тестування послідовностей з довільним алфавітом

5.1. Методи тестування

Нехай задано алфавіт $\Sigma = \{\sigma_0, ..., \sigma_{k-1}\}$ і деяку послідовність $\gamma = \gamma_0, ..., \gamma_{n-1}$, де $\gamma_i \in \Sigma$. Тести призначені для перевірки гіпотези про рівномірний розподіл послідовності γ. Без втрати загальності надалі будемо вважати, що $\Sigma = \{0,1,...,\mathrm{k}-1\}$ для деякого k. Математичне очікування для рівномірного розподілу над Σ визначено як $\mu(\Sigma) = (k-1)/2$. Середньоквадратичне відхилення для рівномірного розподілу над Σ визначена як $\sigma(\Sigma) = \sqrt{(k^2-1)/12}$.

5.1.1. Монобітний тест.

Вхідні дані: послідовність $\gamma = \gamma_0, ..., \gamma_{n-1}$ з алфавіту $\Sigma = \{0,1,...,\mathrm{k}-1\}$, рівень значимості α.

Вихідні дані: 1, якщо послідовність має рівномірний розподіл, 0 інакше.

Алгоритм:

Крок 1: Обчислити $\mu(\Sigma) = (k-1)/2$ та $\sigma^2(\Sigma) = (k^2-1)/12$.

Крок 2: Обчислити $S_n = \sum_{i=0}^{n-1} \gamma_i$.

Крок 3: Обчислити $s_{obs} = \dfrac{|S_n - \mu(\Sigma)n|}{\sigma(\Sigma)\sqrt{n}}$.

Крок 4: Обчислити P-значення $p = erfc\left(\dfrac{s_{osb}}{\sqrt{2}}\right)$.

Крок 5: Повернути 1, якщо $p > \alpha$, інакше повернути 0.

Тест було реалізовано на мові програмування C++, код алгоритму монобітного тесту надано у лістингу 1:

```
double monobitTest(vector<int> seq, int q, double alpha) {
    cout << "\nMonobit testing in progress...\n";
    double mu = (q – 1) / 2;
    double sigma = (pow(q, 2) – 1) / 12;
    uint64_t sumSeq = 0;
    ofstream myfile, myFile2;
    myfile.open("HRNGIDQRandomSeq2_toQ.txt");
    for (int i = 0; i < seq.size(); i++) {
        sumSeq += seq[i];
        myfile << seq[i] << "\n";
    }
    myfile.close();
    double obsSeq = (fabs((sumSeq)-(mu * seq.size()))) / (sqrt(sigma) * sqrt(seq.size()));
    double beforeERFC = obsSeq / (sqrt(2));
    double p = erfc(beforeERFC);
    cout << "\n" << "p = " << p;
    return p;
}
```

Лістинг 1 – Код алгоритму монобітного тесту

5.1.2. Критерій узгодженості Колмогорова – Смирнова та Хі-квадрат.

Для статистичного тестування рівномірності розподілу у різних джерелах пропонується використовувати критерій узгодженості Колмогорова – Смирнова [11] чи критерій Хі-квадрат [12].

У NIST Engineering Statistics Book [13] зазначено, що хоча K-S тест зазвичай розробляється в контексті неперервних розподілів для нецензурованих і незгрупованих даних, його було поширено на дискретні розподіли, а також на цензуровані та згруповані дані. Приклад адаптації тесту на дискретні розподіли надано у роботі [14]. Там пропонується модернізована реалізація функції stats::ks.test() для мови програмування R [15]. Пакет містить запропоновану нову функцію ks.test(). Він не змінює існуючу поведінку ks.test(), за одним незначним винятком – додає можливості, необхідні для виконання тестів з гіпотетичними дискретними розподілами.

Для використання модернізованої версії функції необхідно підключити пакет, виконавши у коді команду: install.packages("ks.test", repos="http://R-Forge.R-project.org").

Для перевірки послідовностей довільного алфавіту з використанням критерію Колмогорова – Смирнова було вирішено зробити декілька тестів:

- перший: подібний до того, який показано у [14], тобто однозразковий тест Колмогорова – Смирнова;
- другий [16]: двозразковий тест, який перевіряє, чи походять два набори даних з одного розподілу.

У першому випадку на вхід тесту подається числовий вектор, який містить значення вибірки з одного з розподілів, та символьний рядок, який визначає функцію, що генерує p-значення для гіпотетичного розподілу, яка може бути однією з pnorm, pbeta, pcauchy, pchisq, pexp, pf, pgamma, plnorm, plogis, pt, punif, pweibull, pbinom, pgeom, phyper, pnbinom, ppois, pwilcox. Також можливим є явне задання розподілу, наприклад, для рівномірного дискретного розподілу в межах значень 0-14, параметр, який передаватиметься у функцію, матиме вигляд: ecdf(0:14).

У другому випадку на вхід подаються два числових вектори, для яких треба перевірити факт походження з одного розподілу.

Для того щоб перевірити, чи належать послідовності за заданим алфавітом рівномірному розподілу, генерується випадковий набір даних з дискретного рівномірного розподілу, що відповідає алфавіту послідовності з використанням функції rdunif з бібліотеки purrr [17]. Генерація відбувається наступним чином:

x <- rdunif(25557884, 0, 14).

У прикладі генерується набір з 25557884 чисел в межах 0-14, які відповідають рівномірному дискретному розподілу.

Для визначення успішності проходження тесту даними застосовується критичне значення для D [13] ($\max(D^+, D^-)$), де D^+ – статистика проходження тесту для alternative = "g", а D^- – статистика проходження тесту для alternative = "l". Критичне значення для D обчислюється за наступною формулою [18]:

$$D = \frac{\text{Константне значення для обраного } \alpha}{\sqrt{n}}$$

де n – довжина вибірки(вибірок), що перевіряється, а константне значення для вибірок понад 50 значень для різних рівнів значущості надається у [19]. У табл. 4 показано критичне значення для D для обраних рівнів значущості.

Таблиця 4

Критичні значення для D для різних рівнів значущості

Рівень значущості	Формула обчислення критичного значення
$\alpha = 0.01$	$\frac{1.62762}{\sqrt{n}}$
$\alpha = 0.05$	$\frac{1.35810}{\sqrt{n}}$
$\alpha = 0.1$	$\frac{1.22385}{\sqrt{n}}$

У якості альтернативи NIST пропонує використовувати тест хі-квадрат [20]. Тест хі-квадрат є альтернативою тестам Андерсона – Дарлінга та Колмогорова – Смірнова. Основною перевагою такого тесту в нашому випадку є те, що тест хі-квадрат може бути застосований до дискретних розподілів, а отже підійде для тестування послідовності довільного алфавіту.

Тест хі-квадрат визначено для перевірки гіпотез:

H_0: дані підпорядковуються заданому розподілу.

H_a: дані не відповідають вказаному розподілу.

Статистика тесту: для обчислення критерію хі-квадрат дані розбиваються на k груп, а тестова статистика визначається як

$$\chi^2 = \sum_{i=1}^{k} (O_i - E_i)^2 / E_i ,$$

де O_i – спостережувана частота для групи i, а E_i – очікувана частота для групи i.

Тестова статистика приблизно відповідає розподілу хі-квадрат з $(k-c)$ ступенями свободи, де k – кількість непорожніх груп, а c – кількість оцінюваних параметрів для розподілу + 1. Для рівномірного розподілу $c = 1$. Тобто, кількість ступенів свободи для рівномірного розподілу буде визначатися за формулою

$$df = k - 1$$

Отже, гіпотеза про те, що дані походять з генеральної сукупності із зазначеним розподілом, відхиляється, якщо $\chi^2 > \chi^2_{1-\alpha,k-c}$, де $\chi^2_{1-\alpha,k-c}$ – критичне значення критерію хі-квадрат з $(k-c)$ ступенями свободи і рівнем значущості α.

Для проведення статистичного тестування з використанням хі-квадрат було обрано двосторонній тест, який передбачає наступну реалізацію [21]:

- Для двостороннього тесту знайдіть стовпчик, що відповідає $1-\frac{\alpha}{2}$ у таблиці критичних значень верхньої межі, і відхиліть нульову гіпотезу, якщо тестова статистика більша за наведене в таблиці значення. Аналогічно, знайдіть у таблиці стовпчик, що відповідає $\frac{\alpha}{2}$, для нижніх критичних значень і відхиліть нульову гіпотезу, якщо тестова статистика менша за наведене в таблиці значення.

У табл. 5 наведено критичні значення нижньої та верхньої межі для обраних ступенів свободи – 14 та 2022 – та обраних рівнів значущості. Значення для ступенів свободи 2022 пораховані з використанням онлайн калькулятора – Critical Chi-Square Value Calculator [22] – оскільки таблиці, які присутні на сайті NIST, містять детальні дані тільки для ступенів свободи до 100 і стислі для ступенів до 1000.

Таблиця 5

Критичні значення розподілу хі-квадрат
для заданих ступенів свободи та рівнів значущості

Ступені свободи	Рівень значущості	Критичні значення (нижня межа – верхня межа)
14	0.1	6.571 – 23.685
	0.05	5.629 – 26.119
	0.01	4.074 – 31.319
2022	0.1	1918.549 – 2127.724
	0.05	1899.266 – 2148.522
	0.01	1861.954 – 2189.557

Далі буде виконане статистичне тестування з використанням наведених критичних значень для послідовностей довільного алфавіту, отриманих з використанням методів, описаних раніше з двійкових послідовностей генератора /dev/random. Буде наведено отримане значення хі-квадрат, а також зроблено висновок про рівномірність розподілу послідовності на основі отриманого значення.

Тест було реалізовано на мові програмування C++, код алгоритму тесту хі-квадрат надано у лістингу 2:

```
double chiSquareTest(vector<int> seq, int q) {
   cout << "\nChi-square testing in progress...\n";
   int* values = new int[q];
   for (int k = 0; k < q; k++) {
      values[k] = 0;
   }
   for (int i = 0; i < seq.size(); i++) {
      values[seq[i]]++;
   }
   double e_i = (double)seq.size() / BASE;
   double part = 0.0;
   double chi = 0.0;
   for (int k = 0; k < q; k++) {
      part = 0.0;
      part = pow((double)values[k] – e_i, 2) / e_i;
      chi += part;
   }
   cout << "\nchi = " << fixed << setprecision(3) << chi;
   return chi;
}
```

Лістинг 2 – Код алгоритму тесту хі-квадрат

Тестування з використанням критерію Колмогорова – Смирнова проводилося мовою програмування R з використанням IDE PyCharm 2023.3.3 [23]. Графіки розподілів послідовностей створено вбудованими засобами IDE.

5.2. Результати тестування

Результати тестування наводяться для значень $q = 15$ та $q = 2023$.

5.2.1. Дослідження послідовностей, отриманих методом відбору.

Послідовності до взяття за основою проходять статистичне тестування, а отже мають проходити тестування і при взятті за обраною основою. Отримані результати підтверджують це припущення.

5.2.1.1. Послідовність за основою 15.

- Дослідження з використанням монобітного тесту.

Таблиця 6

Результати тестування для dev_random_Seq_C1 за основою 15

Розмір вибірки після представлення за основою	239000000
p-значення	0.359225
$\alpha = 0.1$	Тестування пройдено успішно
$\alpha = 0.05$	Тестування пройдено успішно
$\alpha = 0.01$	Тестування пройдено успішно

Отже, як видно з отриманих результатів, послідовність за обраною основою успішно проходить статистичний тест.

- Дослідження з використанням тесту хі-квадрат

Таблиця 7

Результати тестування хі-квадрат для dev_random_Seq_C1 за основою 15

Значення хі-квадрат	16.259	
$\alpha = 0.1$	6.571 < <u>16.259</u> < 23.685	Тестування пройдено успішно
$\alpha = 0.05$	5.629 < <u>16.259</u> < 26.119	Тестування пройдено успішно
$\alpha = 0.01$	4.074 < <u>16.259</u> < 31.319	Тестування пройдено успішно

Послідовність успішно проходить тест хі-квадрат, що підтверджує дані минулих статистичних тестів.

- Дослідження з використанням критерію Колмогорова – Смирнова.

Однозразковий тест.

Таблиця 8

Результати однозразкового тесту Колмогорова-Смирнова для dev_random_Seq_C1

Довжина послідовності, що тестується (n)	23900000
D^+	0.000033529
D^-	0.00014358
D	0.00014358
Критичне значення для $\alpha = 0.01$	$Crit_{0.01} = \frac{1.62762}{\sqrt{n}} = 0.0003329$
Критичне значення для $\alpha = 0.05$	$Crit_{0.05} = \frac{1.35810}{\sqrt{n}} = 0.0002778$
Критичне значення для $\alpha = 0.1$	$Crit_{0.1} \frac{1.22385}{\sqrt{n}} = 0.0002503$
Оскільки $D < (Crit_{0.01}, Crit_{0.05}, Crit_{0.1})$ – послідовності походять з одного розподілу	

Однозразковий тест для послідовності за основою 15 показує, що послідовність походить з рівномірного дискретного розподілу для меж 0:14, а отже послідовність є рівномірною.

Двозразковий тест.

Таблиця 9

Результати двозразкового тесту Колмогорова – Смирнова
для dev_random_Seq_C1

Довжина послідовності, що тестується (n)	23900000
D^+	0.00024615
D^-	0.000060167
D	0.00024615
Критичне значення для $\alpha = 0.01$	$Crit_{0.01} = \frac{1.62762}{\sqrt{n}} = 0.0003329$
Критичне значення для $\alpha = 0.05$	$Crit_{0.05} = \frac{1.35810}{\sqrt{n}} = 0.0002778$
Критичне значення для $\alpha = 0.1$	$Crit_{0.1} \frac{1.22385}{\sqrt{n}} = 0.0002503$
Оскільки $D < (Crit_{0.01}, Crit_{0.05}, Crit_{0.1})$ – послідовності походять з одного розподілу	

Двозразковий тест також показує рівномірність послідовності за основою 15, що можна спостерігати на рис. 1.

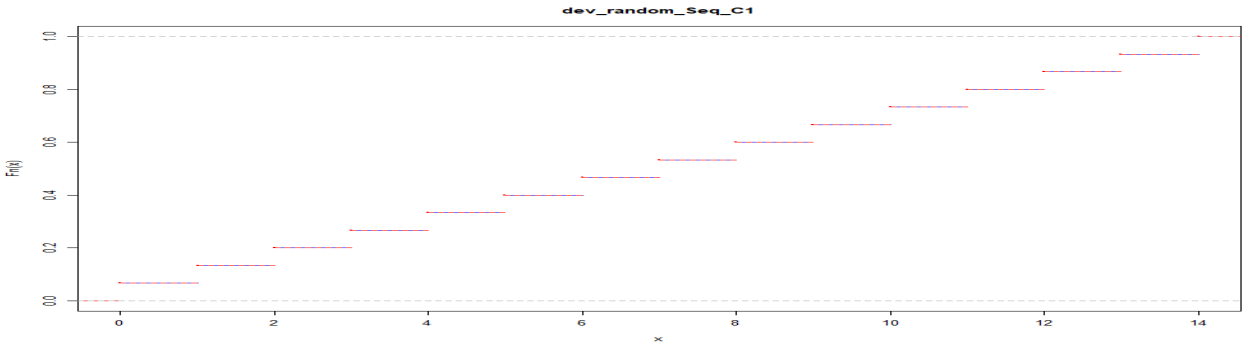

Рис. 1. Порівняння розподілів для послідовності dev_random_Seq_C1 за основою 15
та послідовності з рівномірного дискретного розподілу

Як видно з отриманих графіків, розподіл послідовності майже повністю накладається на рівномірний розподіл для значень з такими ж межами, що означає рівномірність послідовностей довільного алфавіту.

5.2.1.2. Послідовність за основою 2023.

- Дослідження з використанням монобітного тесту

Таблиця 10

Результати тестування для dev_random_Seq_C1 за основою 2023

Розмір вибірки після представлення за основою	90000000
p-значення	0.85315
$\alpha = 0.1$	Тестування пройдено успішно
$\alpha = 0.05$	Тестування пройдено успішно
$\alpha = 0.01$	Тестування пройдено успішно

Послідовність за обраною основою успішно проходить статистичний тест.

- Дослідження з використанням тесту хі-квадрат.

Таблиця 11

Результати тестування хі-квадрат для dev_random_Seq_C1 за основою 2023

Значення хі-квадрат	2032.124	
$\alpha = 0.1$	1918.549 < 2032.124 < 2127.724	Тестування пройдено успішно
$\alpha = 0.05$	1899.266 < 2032.124 < 2148.522	Тестування пройдено успішно
$\alpha = 0.01$	1861.954 < 2032.124 < 2189.557	Тестування пройдено успішно

Послідовність успішно проходить тест хі-квадрат.

- Дослідження з використанням критерію Колмогорова – Смирнова.

Однозразковий тест.

Таблиця 12

Результати однозразкового тесту Колмогорова – Смирнова для dev_random_Seq_C1

Довжина послідовності, що тестується (n)	9000000
D^+	0.00016463
D^-	0.00018614
D	0.00018614
Критичне значення для $\alpha = 0.01$	$Crit_{0.01} = \frac{1.62762}{\sqrt{n}} = 0.0005425$
Критичне значення для $\alpha = 0.05$	$Crit_{0.05} = \frac{1.35810}{\sqrt{n}} = 0.0004527$
Критичне значення для $\alpha = 0.1$	$Crit_{0.1} \frac{1.22385}{\sqrt{n}} = 0.00040795$
Оскільки $D < (Crit_{0.01}, Crit_{0.05}, Crit_{0.1})$ – послідовності походять з одного розподілу	

Однозразковий тест для послідовності за основою 2023 показує, що послідовність походить з рівномірного дискретного розподілу для меж 0:2022, а отже послідовність є рівномірною.

Двозразковий тест.

Таблиця 13

Результати двозразкового тесту Колмогорова – Смирнова для dev_random_Seq_C1

Довжина послідовності, що тестується (n)	9000000
D^+	0.00035444
D^-	0.000109
D	0.00035444
Критичне значення для $\alpha = 0.01$	$Crit_{0.01} = \frac{1.62762}{\sqrt{n}} = 0.0005425$
Критичне значення для $\alpha = 0.05$	$Crit_{0.05} = \frac{1.35810}{\sqrt{n}} = 0.0004527$
Критичне значення для $\alpha = 0.1$	$Crit_{0.1} \frac{1.22385}{\sqrt{n}} = 0.00040795$
Оскільки $D < (Crit_{0.01}, Crit_{0.05}, Crit_{0.1})$ – послідовності походять з одного розподілу	

Двозразковий тест також показує рівномірність послідовності за основою 2023.

Для більшої наочності і зрозумілості отриманих графіків було проведено додаткове тестування для 5000 значень. Графік такого тестування більш ясно показує розбіжності між розподілами, яких не видно на графіках для дуже великої кількості значень (рис. 2).

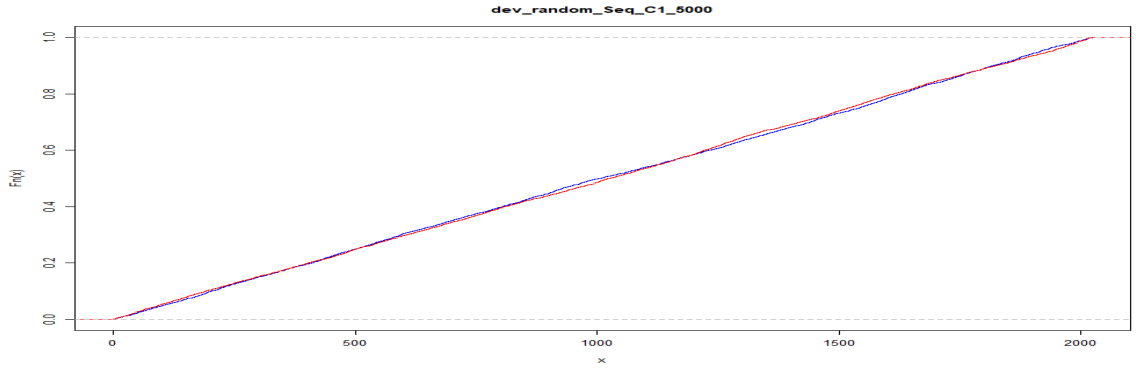

Рис. 2. Порівняння розподілів для послідовності dev_random_Seq_C1 та послідовності з рівномірного дискретного розподілу для 5000 значень

5.2.2. Дослідження послідовностей, отриманих методом «швидкої гральної кістки».

5.2.2.1. Послідовність за основою 15.

- Дослідження з використанням монобітного тесту.

Таблиця 14

Результати тестування для dev_random_Seq_C2 за основою 15

Розмір вибірки після представлення за основою	239000000
p-значення	0.143497
$\alpha = 0.1$	Тестування пройдено успішно
$\alpha = 0.05$	Тестування пройдено успішно
$\alpha = 0.01$	Тестування пройдено успішно

- Дослідження з використанням тесту хі-квадрат.

Таблиця 15

Результати тестування хі-квадрат для dev_random_Seq_C2 за основою 15

Значення хі-квадрат	16.277	
$\alpha = 0.1$	6.571 < 16.277 < 23.685	Тестування пройдено успішно
$\alpha = 0.05$	5.629 < 16.277 < 26.119	Тестування пройдено успішно
$\alpha = 0.01$	4.074 < 16.277 < 31.319	Тестування пройдено успішно

- Дослідження з використанням критерію Колмогорова – Смирнова

Однозразковий тест.

Таблиця 16

Результати однозразкового тесту Колмогорова – Смирнова для dev_random_Seq_C2

Довжина послідовності, що тестується (n)	23900000
D^+	0.000005509
D^-	0.00024991
D	0.00024991
Критичне значення для $\alpha = 0.01$	$Crit_{0.01} = \frac{1.62762}{\sqrt{n}} = 0.0003329$
Критичне значення для $\alpha = 0.05$	$Crit_{0.05} = \frac{1.35810}{\sqrt{n}} = 0.0002778$
Критичне значення для $\alpha = 0.1$	$Crit_{0.1} \frac{1.22385}{\sqrt{n}} = 0.0002503$
Оскільки $D < (Crit_{0.01}, Crit_{0.05}, Crit_{0.1})$ – послідовності походять з одного розподілу	

Однозразковий тест для послідовності за основою 15 показує, що послідовність походить з рівномірного дискретного розподілу для меж 0:14, а отже послідовність є рівномірною.

Двозразковий тест.

Таблиця 17

Результати двозразкового тесту Колмогорова – Смирнова
для dev_random_Seq_C2

Довжина послідовності, що тестується (n)	23900000
D^+	0.00020715
D^-	0.00013912
D	0.00020715
Критичне значення для $\alpha = 0.01$	$Crit_{0.01} = \frac{1.62762}{\sqrt{n}} = 0.0003329$
Критичне значення для $\alpha = 0.05$	$Crit_{0.05} = \frac{1.35810}{\sqrt{n}} = 0.0002778$
Критичне значення для $\alpha = 0.1$	$Crit_{0.1} \frac{1.22385}{\sqrt{n}} = 0.0002503$
Оскільки $D < (Crit_{0.01}, Crit_{0.05}, Crit_{0.1})$ – послідовності походять з одного розподілу	

Двозразковий тест також показує рівномірність послідовності за основою 15, що можна спостерігати на графіку (рис. 3).

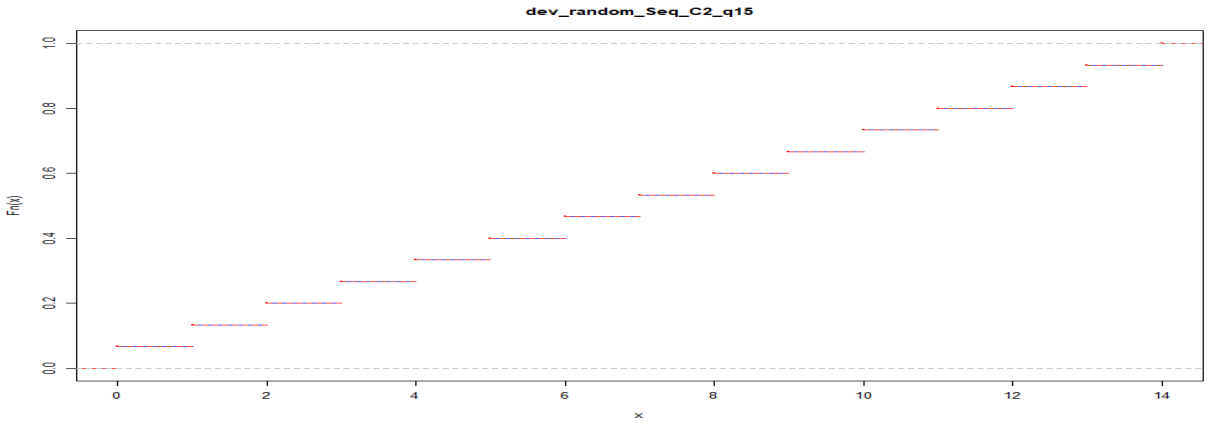

Рис. 3. Порівняння розподілів для послідовності dev_random_Seq_C2 за основою 15
та послідовності з рівномірного дискретного розподілу

5.2.2.2. Послідовність за основою 2023.

- Дослідження з використанням монобітного тесту.

Таблиця 18

Результати тестування для dev_random_Seq_C2 за основою 2023

Розмір вибірки після представлення за основою	90000000
p-значення	0.297884
$\alpha = 0.1$	Тестування пройдено успішно
$\alpha = 0.05$	Тестування пройдено успішно
$\alpha = 0.01$	Тестування пройдено успішно

- Дослідження з використанням тесту хі-квадрат.

Таблиця 19

Результати тестування хі-квадрат для dev_random_Seq_C2 за основою 2023

Значення хі-квадрат	2098.735	
$\alpha = 0.1$	1918.549 < 2098.735 < 2127.724	Тестування пройдено успішно
$\alpha = 0.05$	1899.266 < 2098.735 < 2148.522	Тестування пройдено успішно
$\alpha = 0.01$	1861.954 < 2098.735 < 2189.557	Тестування пройдено успішно

- Дослідження з використанням критерію Колмогорова – Смирнова

Однозразковий тест.

Таблиця 20

Результати однозразкового тесту Колмогорова – Смирнова для dev_random_Seq_C2

Довжина послідовності, що тестується (n)	9000000
D^+	0.000028918
D^-	0.00025623
D	0.00025623
Критичне значення для $\alpha = 0.01$	$Crit_{0.01} = \frac{1.62762}{\sqrt{n}} = 0.0005425$
Критичне значення для $\alpha = 0.05$	$Crit_{0.05} = \frac{1.35810}{\sqrt{n}} = 0.0004527$
Критичне значення для $\alpha = 0.1$	$Crit_{0.1} \frac{1.22385}{\sqrt{n}} = 0.00040795$
Оскільки $D < (Crit_{0.01}, Crit_{0.05}, Crit_{0.1})$ – послідовності походять з одного розподілу	

Однозразковий тест для послідовності за основою 2023 показує, що послідовність походить з рівномірного дискретного розподілу для меж 0:2022, а отже послідовність є рівномірною.

Двозразковий тест.

Таблиця 21

Результати двозразкового тесту Колмогорова – Смирнова для dev_random_Seq_C2

Довжина послідовності, що тестується (n)	9000000
D^+	0.00039344
D^-	0.000086889
D	0.00039344
Критичне значення для $\alpha = 0.01$	$Crit_{0.01} = \frac{1.62762}{\sqrt{n}} = 0.0005425$
Критичне значення для $\alpha = 0.05$	$Crit_{0.05} = \frac{1.35810}{\sqrt{n}} = 0.0004527$
Критичне значення для $\alpha = 0.1$	$Crit_{0.1} \frac{1.22385}{\sqrt{n}} = 0.00040795$
Оскільки $D < (Crit_{0.01}, Crit_{0.05}, Crit_{0.1})$ – послідовності походять з одного розподілу	

Двозразковий тест також показує рівномірність послідовності за основою 2023, що можна спостерігати на рис. 4.

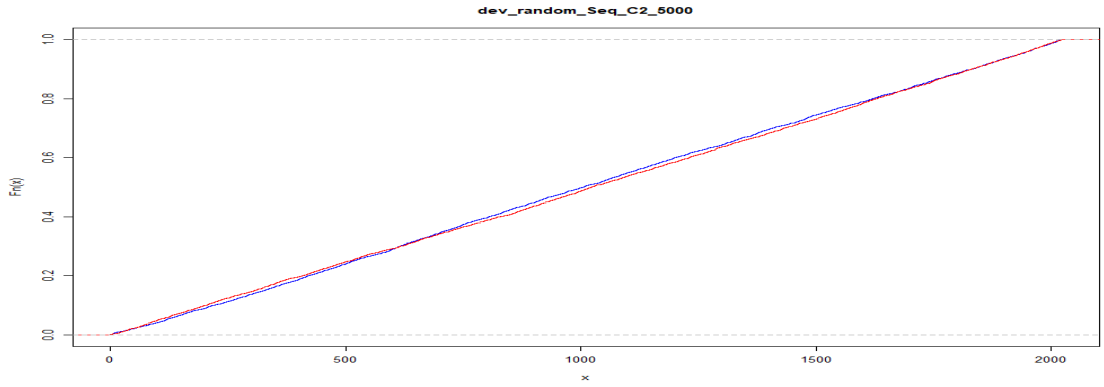

Рис. 4. Порівняння розподілів для послідовності dev_random_Seq_C2
та послідовності з рівномірного дискретного розподілу для 5000 значень

5.2.3. Дослідження послідовностей, отриманих методом Уокера.

5.2.3.1. Послідовність за основою 15.

- Дослідження з використанням монобітного тесту.

Таблиця 22

Результати тестування для dev_random_Seq_C3 за основою 15

Розмір вибірки після представлення за основою	239000000
p-значення	0.706594
$\alpha = 0.1$	Тестування пройдено успішно
$\alpha = 0.05$	Тестування пройдено успішно
$\alpha = 0.01$	Тестування пройдено успішно

- Дослідження з використанням тесту хі-квадрат

Таблиця 23

Результати тестування хі-квадрат для dev_random_Seq_C3 за основою 15

Значення хі-квадрат	10.969	
$\alpha = 0.1$	6.571 < 10.969 < 23.685	Тестування пройдено успішно
$\alpha = 0.05$	5.629 < 10.969 < 26.119	Тестування пройдено успішно
$\alpha = 0.01$	4.074 < 10.969 < 31.319	Тестування пройдено успішно

- Дослідження з використанням критерію Колмогорова – Смирнова.

Однозразковий тест.

Таблиця 24

Результати однозразкового тесту Колмогорова – Смирнова
для dev_random_Seq_C3

Довжина послідовності, що тестується (n)	23900000
D^+	0.000060767
D^-	0.00010215
D	0.00010215
Критичне значення для $\alpha = 0.01$	$Crit_{0.01} = \frac{1.62762}{\sqrt{n}} = 0.0003329$
Критичне значення для $\alpha = 0.05$	$Crit_{0.05} = \frac{1.35810}{\sqrt{n}} = 0.0002778$
Критичне значення для $\alpha = 0.1$	$Crit_{0.1} \frac{1.22385}{\sqrt{n}} = 0.0002503$
Оскільки $D < (Crit_{0.01}, Crit_{0.05}, Crit_{0.1})$ – послідовності походять з одного розподілу	

Однозразковий тест для послідовності за основою 15 показує, що послідовність походить з рівномірного дискретного розподілу для меж 0:14, а отже послідовність є рівномірною.

Двозразковий тест.

Таблиця 25

Результати двозразкового тесту Колмогорова – Смирнова
для dev_random_Seq_C3

Довжина послідовності, що тестується (n)	23900000
D^+	0.00018473
D^-	0.000006736
D	0.00018473
Критичне значення для $\alpha = 0.01$	$Crit_{0.01} = \frac{1.62762}{\sqrt{n}} = 0.0003329$
Критичне значення для $\alpha = 0.05$	$Crit_{0.05} = \frac{1.35810}{\sqrt{n}} = 0.0002778$
Критичне значення для $\alpha = 0.1$	$Crit_{0.1} \frac{1.22385}{\sqrt{n}} = 0.0002503$
Оскільки $D < (Crit_{0.01}, Crit_{0.05}, Crit_{0.1})$ – послідовності походять з одного розподілу	

Двозразковий тест також показує рівномірність послідовності за основою 15, що можна спостерігати на рис. 5.

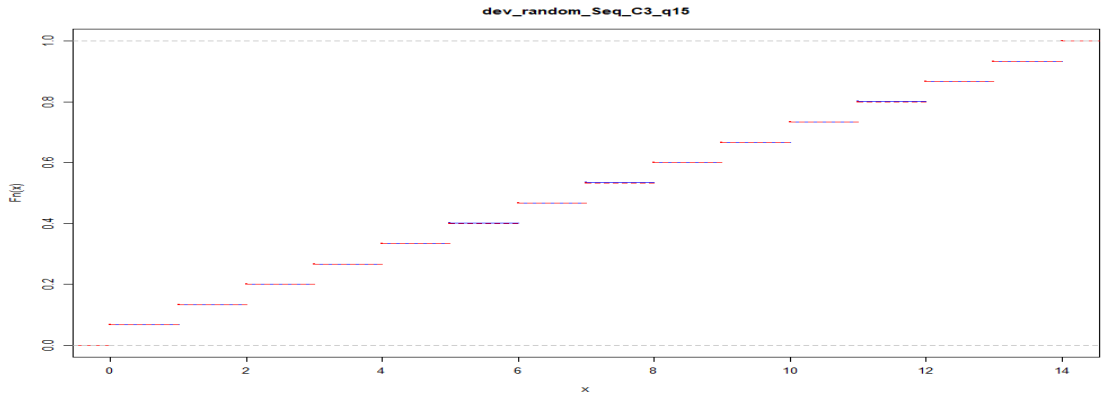

Рис. 5. Порівняння розподілів для послідовності dev_random_Seq_C3 за основою 15 та послідовності з рівномірного дискретного розподілу

5.2.3.2. Послідовність за основою 2023.

- Дослідження з використанням монобітного тесту.

Таблиця 26

Результати тестування для dev_random_Seq_C3 за основою 2023

Розмір вибірки після представлення за основою	90000000
p-значення	0.638141
$\alpha = 0.1$	Тестування пройдено успішно
$\alpha = 0.05$	Тестування пройдено успішно
$\alpha = 0.01$	Тестування пройдено успішно

- Дослідження з використанням тесту хі-квадрат.

Таблиця 27

Результати тестування хі-квадрат для dev_random_Seq_C3 за основою 2023

Значення хі-квадрат	2067.994	
$\alpha = 0.1$	1918.549 < 2067.994 < 2127.724	Тестування пройдено успішно
$\alpha = 0.05$	1899.266 < 2067.994 < 2148.522	Тестування пройдено успішно
$\alpha = 0.01$	1861.954 < 2067.994 < 2189.557	Тестування пройдено успішно

- Дослідження з використанням критерію Колмогорова – Смирнова.

Однозразковий тест.

Таблиця 28

Результати однозразкового тесту Колмогорова – Смирнова для dev_random_Seq_C3

Довжина послідовності, що тестується (n)	9000000
D^+	0.00020297
D^-	0.00024453
D	0.00024453
Критичне значення для $\alpha = 0.01$	$Crit_{0.01} = \frac{1.62762}{\sqrt{n}} = 0.0005425$
Критичне значення для $\alpha = 0.05$	$Crit_{0.05} = \frac{1.35810}{\sqrt{n}} = 0.0004527$
Критичне значення для $\alpha = 0.1$	$Crit_{0.1} \frac{1.22385}{\sqrt{n}} = 0.00040795$
Оскільки $D < (Crit_{0.01}, Crit_{0.05}, Crit_{0.1})$ – послідовності походять з одного розподілу	

Однозразковий тест для послідовності за основою 2023 показує, що послідовність походить з рівномірного дискретного розподілу.

Двозразковий тест.

Таблиця 29

Результати двозразкового тесту Колмогорова – Смирнова для dev_random_Seq_C3

Довжина послідовності, що тестується (n)	9000000
D^+	0.00018144
D^-	0.00038344
D	0.00038344
Критичне значення для $\alpha = 0.01$	$Crit_{0.01} = \frac{1.62762}{\sqrt{n}} = 0.0005425$
Критичне значення для $\alpha = 0.05$	$Crit_{0.05} = \frac{1.35810}{\sqrt{n}} = 0.0004527$
Критичне значення для $\alpha = 0.1$	$Crit_{0.1} \frac{1.22385}{\sqrt{n}} = 0.00040795$
Оскільки $D < (Crit_{0.01}, Crit_{0.05}, Crit_{0.1})$ – послідовності походять з одного розподілу	

Двозразковий тест також показує рівномірність послідовності за основою 2023 (рис. 6).

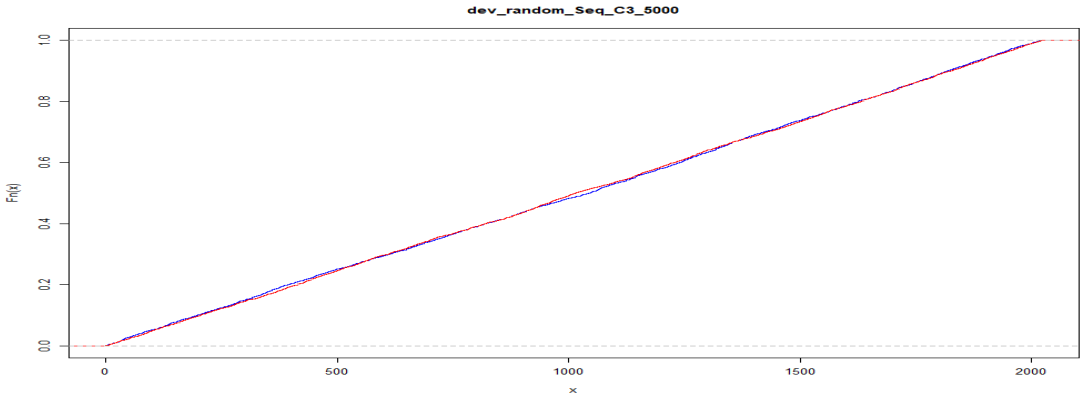

Рис. 6. Порівняння розподілів для послідовності dev_random_Seq_C3 та послідовності з рівномірного дискретного розподілу для 5000 значень

На рис. 7 показано значення хі-квадрат для послідовностей з різних методів отримання послідовностей довільного алфавіту для різних основ.

Рис. 7. Входження послідовностей за основою 15 у діапазон критичних значень хі-квадрат для рівня значущості 0.1

Рис. 8. Входження послідовностей за основою 2023 у діапазон критичних значень хі-квадрат для рівня значущості 0.1

Як видно з отриманих результатів, послідовності довільного алфавіту, отримані кожним з методів успішно проходять статистичне тестування з використанням різних критеріїв.

Висновки

Твердження 5 надає можливість будувати обґрунтовано стійки (у загальновизнаному сенсі цього слова [5 – 7]) генератори (псевдо)випадкових послідовностей над довільним алфавітом, виходячи з аналогічних генераторів двійкових послідовностей та допустимих алгоритмів, визначених в п. 3.

Виконано порівняння наведених методів генерації послідовностей за швидкодією та обсягом необхідних даних для параметрів, які застосовують в алгоритмах [1 – 3]. Для практичного застосування за швидкодією їхніх програмних реалізацій і обсягом вхідних даних, що ними застосовуються, доцільно застосовувати алгоритм A_3 замість алгоритму A_1

Послідовності довільного алфавіту, отримані кожним з методів, успішно проходять статистичне тестування з використанням різних критеріїв.

Список літератури:
1. FIPS.203. https://csrc.nist.gov/pubs/fips/203/ipd
2. FIPS.204. https://csrc.nist.gov/pubs/fips/204/ipd
3. ДСТУ 9212-2023. ДСТУ 9212:2023 Інформаційні технології. Криптографічний захист інформації. Алгоритм електронного підпису на модульних решітках
4. Олексійчук А.М., Курінний О.В. Методи криптоаналізу потокових шифрів : навч. вид. Київ : КПІ ім. Ігоря Сікорського, 2023. 172 с.
5. Blum M., Micali S. How to generate cryptographically strong sequences of pseudo-random bits // SIAM Journal of Computing. 1984. Vol. 13(4). P. 850–864.
6. Yao A.C. Theory and application of trapdoor functions // The 23-th. Annual Symposium on Foundations of Computer Science. IEEE, 1982. P. 80–91.
7. Katz J., Lindell Y. Introduction to modern cryptography. CRC Press, 2015. 598 p.
8. Lumbroso J. Optimal discrete uniform generation from coin flips, and application // arXiv.1304.1916v1 [cs.DS] 11 Apr 2013.
9. Кнут Д. Искусство программирования. Т. 2. Получисленные алгоритмы. 3-е изд. ; пер. с англ. Москва : Изд. дом “Вильямс”, 2000. 832 с.
10. Алексейчук А.Н. Неасимптотические нижние границы информационной сложности статистических атак на симметричные криптосистемы // Кибернетика и системный анализ. 2018. №. 1. С. 93–104.
11. Zvi Drezner,Ofir Turel & Dawit Zerom. A Modified Kolmogorov – Smirnov Test for Normality. 2010. URL: https://www.tandfonline.com/doi/citedby/10.1080/03610911003615816?scroll=top&needAccess=true (дата звернення: 12.03.2024).
12. William G. Cochran. The χ2 Test of Goodness of Fit. 1952. URL: https://www.jstor.org/stable/2236678 (дата звернення: 12.03.2024).
13. NIST Engineering Statistics Handbook. Kolmogorov – Smirnov Goodness-of-Fit Test. URL: https://www.itl.nist.gov/div898/handbook/eda/section3/eda35g.htm (дата звернення: 12.03.2024).
14. Taylor B. Arnold and John W. Emerson. Nonparametric Goodness-of-Fit Tests for Discrete Null Distributions. 2011. URL: http://www.stat.yale.edu/~jay/EmersonMaterials/DiscreteGOF.pdf (дата звернення: 12.03.2024).
15. R Development Page. Proposed Revision to stats::ks.test(). URL: https://r-forge.r-project.org/R/?group_id=802 (дата звернення: 12.03.2024).
16. Docs Tibco. Kolmogorov-Smirnov Tests. URL: https://docs.tibco.com/pub/enterprise-runtime-for-R/6.0.0/doc/html/Language_Reference/stats/ks.test.html (дата звернення: 12.03.2024).
17. Purrr: Functional Programming Tools. URL: https://uribo.github.io/rpkg_showcase/programming/purrr.html (дата звернення: 12.03.2024).
18. S. Massa. Lecture 13: Kolmogorov Smirnov Test & Power of Tests. Department of Statistics, University of Oxford. 2016. URL: https://www.ime.unicamp.br/~dias/Lecture%2013.pdf (дата звернення: 12.03.2024).
19. Kolmogorov-Smirnov Test Critical Values. URL: https://people.cs.pitt.edu/~lipschultz/cs1538/prob-table_KS.pdf (дата звернення: 12.03.2024).
20. NIST Engineering Statistics Handbook. Chi-Square Goodness-of-Fit Test. URL: https://www.itl.nist.gov/div898/handbook/eda/section3/eda35f.htm (дата звернення: 12.03.2024).
21. NIST Engineering Statistics Handbook. Critical Values of the Chi-Square Distribution. URL: https://www.itl.nist.gov/div898/handbook/eda/section3/eda3674.htm (дата звернення: 12.03.2024).
22. Soper D.S. Critical Chi-Square Value Calculator. 2024. URL: https://www.danielsoper.com/statcalc (дата звернення: 12.03.2024).
23. Jetbrains. PyCharm 2023.3.3. URL: https://www.jetbrains.com/pycharm/whatsnew/ (дата звернення: 12.03.2024).
24. Генератор рівномірно розподілених ймовірних чисел: пат. 33361 Україна : G06F7/58, 07C15/00. №99020855; заявл. 16.02.1999; опубл. 15.02.2001 Бюл. №1
25. Офіційний сайт КГВЧ Quantis RNG від швейцарської компанії ID Quantique (IDQ) [Електроний ресурс]. Режим доступу: https://www.idquantique.com/random-number-generation/overview/.

Надійшла до редколегії 14.01.2024

Відомості про авторів:

Горбенко Іван Дмитрович – д-р техн. наук, професор, Харківський національний університет імені В. Н. Каразіна, професор кафедри безпеки інформаційних систем і технологій, факультет комп’ютерних наук, АТ “Інститут Інформаційних Технологій”, головний конструктор, Україна; e-mail: gorbenkoi@iit.kharkov.ua; ORCID: https://orcid.org/0000-0003-4616-3449

Олексійчук Антон Миколайович – д-р техн. наук, доцент, Інститут спеціального зв'язку та захисту інформації Національного технічного університету України “Київський політехнічний інститут імені Ігоря Сікорського”, професор спеціальної кафедри № 1; Україна; e-mail: alex-dtn@ukr.net

Качко Олена Григорівна – канд. техн. наук, Харківський національний університет радіоелектроніки, професор кафедри програмної інженерії, факультет комп’ютерних наук, АТ «Інститут інформаційних технологій», начальник відділу програмування; Україна; e-mail: iit@iit.kharkov.ua; ORCID: https://orcid.org/0000-0001-9249-0497

Дерев’янко Ярослав Андрійович – науковий співробітник-консультант АТ «Інститут інформаційних технологій», Україна; e-mail: yarik0009258@gmail.com; ORCID: https://orcid.org/0000-0002-3290-3373