**УДК 004.056.5** **DOI:10.30837/rt.2024.2.217.02**

*Я.А. ДЕРЕВ'ЯНКО, М.В. ЄСІНА, канд. техн. наук, Д.Ю. ГОРБЕНКО*

# ОБҐРУНТУВАННЯ МЕТОДІВ ОБЧИСЛЕННЯ ТА АНАЛІЗ ВЛАСТИВОСТЕЙ ПСЕВДОВИПАДКОВИХ ТА ВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ НА ОСНОВІ ДНК

## Вступ

Невід'ємною вимогою до сучасних інформаційних систем є надання користувачам послуг конфіденційності, цілісності, доступності та неспростовності. Якість таких послуг напряму залежить від криптографічних перетворень, важливою складовою яких є випадковість. Тому генерування псевдовипадкових (ПВП) та випадкових (ВП) послідовностей є однією з актуальних та важливих задач. Випадкові числа використовуються для генерації сеансових ключів, параметрів підпису, nonce, викликів, засліплення та маскування значень (для запобігання атакам на реалізацію) тощо. ВП генеруються на основі фізичних та нефізичних джерел шуму [1]. ПВП генеруються з використанням генераторів випадкових послідовностей (ГВП), як правило на основі відносно коротких ВП, наприклад, ключів тощо.

Наші попередні дослідження [2] вказують на теоретичну можливість використання у якості джерела шуму (ДШ) та відповідно джерела ВП ДНК. Як показав пошук та аналіз, детальних досліджень в цьому напрямі немає чи вони недоступні. Також згідно [1] ДНК можливо віднести до нефізичних ДШ та відповідно ГВП. Для оцінки та порівняння ПВП та ВП на основі використання ДНК потрібно провести широкі практичні та теоретичні дослідження з використанням ентропійних (стохастичних) методів та методик, та визнаних статистичних методик (бажано стандартизованих). Цим вимогам задовольняють AIS 20 та AIS 31 [1], що визначають стандартизовані удосконалені методи стохастичного та статистичного оцінювання, та порівняння з іншими ДШ та відповідно ГВП, як потрібно оцінювати ГВЧ.

Дана стаття присвячена новим розробленим методам отримання ПВП та ВП на основі послідовностей ДНК. Вони розглядаються у якості не фізичних справжніх ВП, в тому числі з використанням за необхідності екстракторів [1, 2]. Результатами дослідження є згенеровані ПВП та ВП на основі ДНК, а також експериментально отримані значення статистичної оцінки та оцінки подібності послідовностей.

По суті ця стаття є вступом в теорію та практику генерування ПВП та ВП на основі ДНК. У ній подаються результати вирішення наступних проблемних питань:

1) Пропозиції щодо інтерпретації та подання ДНК у якості не фізичного ДШ та подальшої оцінки.

2) Обґрунтування та розробка методів обчислення ПВП та ВП, а також методів порівняння послідовностей.

3) Аналіз статистичних та стохастичних властивостей ВП та ПВП на основі ДНК, а також удосконалення їх властивостей на основі екстракції.

4) Аналіз подібності ПВП, ВП та ДНК послідовностей для різних ДНК.

Вважаємо, що актуальними та необхідними є подальші дослідження, оцінка та порівняння різних ДНК, встановлення подібності ПВП різних ДНК тощо.

## 1. Інтерпретація ДНК для можливості виконання подальшого дослідження та оцінка властивостей таких послідовностей

У ДНК зустрічається чотири види азотистих основ (аденін, гуанін, тимін і цитозин). Зважаючи на це ДНК можна представити у вигляді послідовності азотистих основ наступним чином (рис. 1):

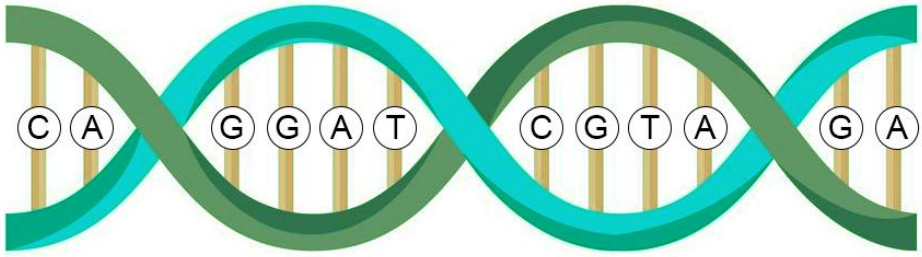


Рис. 1. ДНК (coding strand) у вигляді азотистих основ

Таким чином, маємо послідовність, яка складається з алфавіту, у якому можливі чотири значення – A, C, G та T. Така послідовність може бути представлена у вигляді двійкової послідовності шляхом заміни цих значень на відповідні двійкові комбінації – A=00, C=01, G=10 та T=11. Наприклад, маючи таку послідовність як на рис. 1 – CAGGATCGTAGA – отримаємо двійкову послідовність 010010100011011011001000.

Зразки ДНК різних організмів різної довжини доступні на сайті Національної Бібліотеки Медицини США у банку генів (GenBank) [3] та у ДНК банку Японії (DNA Data Bank of Japan) [4].

Для проведення досліджень було обрано декілька зразків ДНК різних організмів. Обрані послідовності ДНК було представлено у вигляді двійкових як описано вище. Для кожної з них було проведено статистичне та стохастичне тестування. Отримані результати надають інформацію про те чи мають «сирі» послідовності достатньо випадковості і чи потребують вони покращення.

Результати тестування «сирих» послідовностей представлено на прикладі тестування послідовностей з ДНК людини, оскільки результати для інших послідовностей є доволі схожими. Оскільки показники не є задовільними, їх представлення у вигляді статистичного портрету не є доцільним (рис. 2, 3):

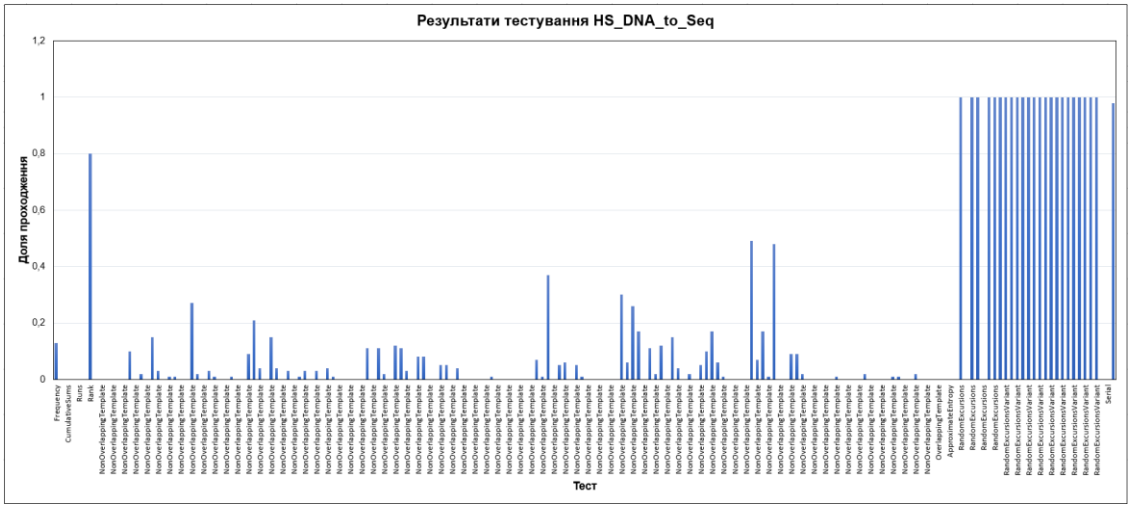


Рис. 2. Результати статистичного тестування «сирої» послідовності з ДНК людини

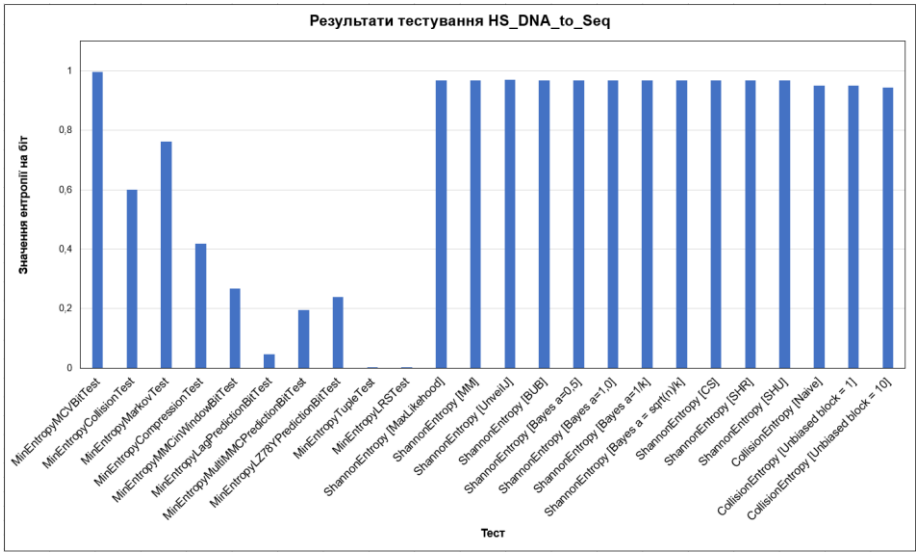

Рис. 3. Результати стохастичного тестування «сирої» послідовності з ДНК людини

Як видно з отриманих результатів, статистичні характеристики «сирих» послідовностей не є задовільними, майже всі статистичні тести провалені. Хоча більшість тестів мінімальної ентропії мають незадовільне значення, показники ентропії Шеннона та колізійної ентропії є хорошими, що вказує на присутність певної кількості ентропії у таких послідовностях, це дає можливість застосування екстракції випадковості з використанням, наприклад, блокових симетричних шифрів (БСШ).

## 2. Обґрунтування методів обчислення ПВП та ВП, а також методів порівняння послідовностей

Для отримання послідовностей було використано екстрактор або ж DRNG на основі ДСТУ 7624:2014 [5] з довжиною ключа 256 біт у режимі гамування (CTR), оскільки саме такий режим пропонується використовувати з блоковим шифром для отримання ПВП та ВП у AIS 20/31 [1].

У стандарті [5] передбачено наступне:

Для генерації псевдовипадкових послідовностей з використанням ДСТУ 7624:2014 у режимі гамування (CTR):

- Задання ключа шифрування
  для $\lambda$=256 Key = Hash256 (b || 1, bBytes+1)
  для $\lambda$>256 Key = Hash512 (b || 1, bBytes+1)
- Задання синхропосилки
  для $\lambda$=256 IV = Hash256 (b || 2, bBytes+1)
  для $\lambda$>256 IV = Hash512 (b || 2, bBytes+1),

де b – рядок октетів, який визначає внутрішній стан генератора, bBytes (октетів) – його довжина.

Для отримання випадкових послідовностей, необхідно застосовувати у якості ключа та синхропосилки випадкові значення. У стандарті [5] вказано наступне.

Для генерації випадкових послідовностей з використанням ДСТУ 7624:2014 в режимі гамування (CTR):

- як ключ шифрування вибираємо випадковий двійковий рядок завдовжки 256, якщо $\lambda$=256 та 512, якщо $\lambda$=384 або $\lambda$ = 512. Цей ключ визначає внутрішній стан генератора;
- як синхропосилку вибираємо випадковий двійковий рядок завдовжки 256, якщо $\lambda$=256 та 512, якщо $\lambda$=384 або $\lambda$=512.

Таким чином, ДСТУ 7624:2014 дозволяє отримати ПВП та ВП в залежності від обраного режиму. Далі будуть представлені алгоритми отримання ПВП та ВП, в яких враховано виконання вказаних вище вимог.

### 2.1. Отримання псевдовипадкових послідовностей з ДНК з використанням ДСТУ 7624:2014 на одному ключі

Для отримання ПВП було використано екстрактор на основі ДСТУ 7624:2014 з довжиною ключа 256 біт у режимі гамування (CTR).

Псевдокод алгоритму отримання псевдовипадкових послідовностей наводиться далі:

```
Алгоритм 1
Вхід:
Файл з необробленими даними
Вихід:
Файл з псевдовипадковою послідовністю на виході
1 Вибір режиму роботи
#define CK_SIZE 32
2 Ініціалізація контексту
kalyna_t* ctx44_e = KalynaInit(256, 256);
3 Ініціалізація буферу, в який буде зчитуватися необроблена послідовність
uint64_t mem[seq_len/8];
4 Зчитування послідовності з бінарного файлу у буфер
mem[i] = байти[i – i+8]
5 Ініціалізація ключа та синхропосилки залежно від режиму
uint64_t initKey …._e[CK_SIZE/8] = {0x.., 0x.., … 0x… };
uint64_t nonce …._e[CK_SIZE/8] = {0x.., 0x.., … 0x… };
6 Набір послідовності необхідної довжини
while (не досягнуто необхідної довжини послідовності)
Зашифрування початкової/вже зашифрованої раніше послідовності
KalynaKeyExpand(key, ctx)
counter = 0;
        for (i < dataLen; i+= CK_SIZE/8){
                uint64_t pt…._e[CK_SIZE/8] = {nonce[i] … nonce[i+…]};
                pt = pt^counter;
                counter++;
                KalynaEncipher(pt…._e, ctx…._e, ct…._e);
                ct … ct[i+…]  = ct^mem;
                mem[i] … mem[i+…] = ct…_e[i] … ct…_e[i+…];
        }
        Запис у файл зашифрованої частини
end while
```

Розроблений алгоритм дозволяє отримувати ПВП на основі вхідних послідовностей та обраних ключа і синхропосилки. Таким методом було отримано послідовності різних довжин з ДНК різних організмів. Результати аналізу статистичних властивостей таких послідовностей будуть надані у наступному пункті.

### 2.2. Отримання випадкових послідовностей з ДНК з використанням ДСТУ 7624:2014 на сеансових ключах, отриманих з NPTRNG ядра Linux

Для відповідності вимогам [5] щодо ключа та синхропосилки для генерації ВП, їх отримання відбувалося з використанням NPTRNG /dev/random [6] на ядрі Linux 5.4, що за AIS 20/31 [1] відповідає класу функціональності NTG.1, проте в більш пізніх версіях ядра 5.5+ цей генератор відносять до DRNG класу DRG.3 [1].

Отже, отримання випадкової послідовності можливе подібним чином:

1. Обираємо з ДНК людини двійкову послідовність невеликого розміру (з метою частішої зміни ключа шифрування та синхропосилки), наприклад 1 МБ.

2. З використанням NPTRNG отримуємо ключ шифрування та синхропосилку для кожної ітерації зашифрування (наприклад, з використанням генератора /dev/random на платформі Linux).

3. Зашифровуємо послідовність ДНК необхідну кількість разів, щоб отримати необхідну довжину вихідної послідовності.

Псевдокод алгоритму отримання випадкових послідовностей наводиться далі.

```
Алгоритм 2
Вхід:
Файл з необробленими даними
Вихід:
Файл з випадковою послідовністю на виході
1 Вибір режиму роботи
#define CK_SIZE 32
2 Ініціалізація контексту
kalyna_t* ctx44_e = KalynaInit(256, 256);
3 Ініціалізація буферу, в який буде зчитуватися послідовність
uint64_t mem[seq_len/8];
4 Зчитування послідовності з бінарного файлу у буфер
mem[i] = байти[i – i+8]
5 Набір послідовності необхідної довжини
while (не досягнуто необхідної довжини послідовності)
        Ініціалізація ключа та синхропосилки залежно від режиму
        fd = open("/dev/random", O_RDONLY);
        read(fd, initKey, CK_SIZE);
        read(fd, nonce, CK_SIZE);
        Зашифрування початкової/вже зашифрованої раніше послідовності
        KalynaKeyExpand(key, ctx)
        counter = 0;
        for (i < dataLen; i+= CK_SIZE/8){
                uint64_t pt…._e[CK_SIZE/8] = {nonce[i] … nonce[i+…]};
                pt = pt^counter;
                counter++;
                KalynaEncipher(pt…._e, ctx…._e, ct…._e);
                ct[i] … ct[i+…]  = ct^mem;
                mem[i] … mem[i+…] = ct…_e[i] … ct…_e[i+…];
        }
        Запис у файл зашифрованої частини
end while
```

Розроблений алгоритм дозволяє отримувати ВП на основі вхідних послідовностей та ключа і синхропосилки, отриманих в використанням відповідного генератора. Таким методом також було отримано послідовності різних довжин з ДНК різних організмів. Результати статистичного аналіз представлено у наступному пункті.

Далі розглянуто методи порівняння послідовностей.

### 2.3. Порівняння послідовностей з використанням підрахунку k-мерів та k-мер відстані

Хорошим методом порівняння без попереднього вирівнювання ДНК послідовностей або будь-яких рядків на подібність є підрахунок k-мерів. У біоінформатиці k-мери – це підрядки довжиною *k*, що містяться в біологічній послідовності. Переважно використовуються в контексті обчислювальної геноміки та аналізу послідовностей.

При застосуванні, наприклад, добітових чи байтових потоків, або простих рядків у якості k-мерів виступають підрядки довжини *k*: для бітового рядка підрядки бітів, для байтового – байтів, а для звичайного тесту – підрядки символів відповідно.

У якості прикладу для демонстрації взято частину послідовності ДНК довжиною 8 нуклеотидів. У табл. 1 показано всі можливі k-мери довжини від 1 до 8 для заданої послідовності.

Таблиця 1

Значення всіх можливих k-мерів (підрядків довжиною *k*)
для демонстраційної послідовності

| CACGATCG | |
|---|---|
| Значення k | k-мери |
| 1 | C, A, C, G, A, T, C, G |
| 2 | CA, AC, CG, GA, AT, TC, CG |
| 3 | CAC, ACG, CGA, GAT, ATC, TCG |
| 4 | CACG, ACGA, CGAT, GATC, ATCG |
| 5 | CACGA, ACGAT, CGATC, GATCG |
| 6 | CACGAT, ACGATC, CGATCG |
| 7 | CACGATC, ACGATCG |
| 8 | CACGATCG |

Розкладання послідовності на k-мери для аналізу дозволяє аналізувати набір фрагментів фіксованого розміру, а не саму послідовність, що може бути більш ефективним. k-мери дуже корисні для зіставлення послідовностей, а операції з множинами швидші, простіші, і для роботи з ними існує багато доступних алгоритмів і технік. По суті, використання k-мерів спрощує біоінформатику до підрахунку та порівняння наявності чи відсутності речей.

Отже, для порівняння двох ДНК послідовностей для кожної з них спочатку підраховується кількість k-мерів у її складі. Створюється таблиця, де кожному з k-мерів ($4^k$ для стандартного алфавіту ДНК і, наприклад, $2^k$ у випадку двійкового алфавіту) відповідає кількість входжень конкретного k-меру у послідовність, що досліджується.

Для такого підрахунку можна також скористатися пакетом kmer для мови програмування R [7], який підраховує кількість k-мерів у послідовностях, проте у пакеті підраховуються всі можливі k-мери простору $4^k$. Такий метод спричиняє серйозне навантаження на апаратні ресурси комп'ютера, найбільшим чином на оперативну пам'ять при великих значеннях *k*. Оскільки для збереження масиву k-мерів для $k$=31 потрібно $4.6^{18}$ елементів, що не є можливим на жодному з сучасних комп'ютерів.

У даному дослідженні пропонується новий метод, який є доволі швидким, не потребує великих затрат пам'яті та має порівняно невисоку алгоритмічну складність.

Запропонований метод полягає в наступному.

Алгоритм 3

1 Пошук всіх можливих k-мерів у обох послідовностях

2 Створення об'єднання цих k-мерів, щоб вилучити зі списку повторні k-мери

3 Представлення k-мерів у двійковому вигляді (A=00, C=01, G=10, T=11). Це в свою чергу дає можливість представлення отриманих k-мерів у вигляді 64-бітних чисел, якщо $k$<33 для ДНК послідовностей і $k$<65 для двійкових послідовностей

4 Створення map для кожної з послідовностей

5 Присвоєння в якості ключа значень з набору різних (distinct) k-мерів

6 Прохід по послідовностях і додавання +1 до індексу з ключем, який відповідає k-меру, що знайдено в послідовності. Такий прохід не потребує подвійного циклу і дозволяє підраховувати кількість входжень кожного з k-мерів під час одного проходження

7 Розрахунок k-мер відстані

Математичну формулу для розрахунку k-мер відстані було представлено у роботі [8]. При реалізації алгоритму було використано формулу, що використовується для розрахунку k-мер відстані у пакеті kmer для мови програмування R [7], а саме:

$$F = \sum_{Distinct.length} \frac{\min(p(s_1), p(s_2))}{\min(len(s_1), len(s_2)) - k + 1}, \quad dist(s_1, s_2) = \frac{\log(0.1 + F) - \log(1.1)}{\log(0.1)} \tag{1}$$

де F – дробове загальне число k-мер, $p(s)$ – відповідний k-мер з простору $4^k$ ($2^k$) кожної з послідовностей, а $len(s)$ – довжина послідовностей.

Для коректного вибору розміру k-мерів необхідно також враховувати розмір послідовностей, що будуть порівнюватися, а також алфавіт цих послідовностей. Інструмент Mash [9] пропонує визначати розмір k-мерів для оцінки, оцінюючи ймовірність випадкового збігу як

$$p = \frac{1}{\frac{\left(\Sigma\right)^k}{g} + 1}, \tag{2}$$

де g – розмір геному (послідовності), а Σ – алфавіт (ACGT або, наприклад, 01). Якщо ця ймовірність перевищує поріг (за замовчуванням 0.01), то розмір k-мерів – $k$ – необхідно збільшувати.

У інших джерелах, пропонується обирати розмір так, щоб загальна кількість доступних k-мерів була достатньо більшою за розмір геному, що досліджується.

### 2.4. Порівняння послідовностей на основі MinHash відстані з використанням геш-функції Купина (ДСТУ 7564:2014)

Ще одним методом, який забезпечує досить високу швидкість обчислення, а також має невисокі вимоги до пам'яті, що буде використовуватися, є алгоритм MinHash. У комп'ютерних науках та аналізі даних MinHash – це техніка для швидкої оцінки того, наскільки схожі два набори. Схема була вперше запропонована Andrei Broder у роботі [10]. Застосування такого методу до порівняння послідовностей ДНК вперше згадується у роботі [11]. У нашому дослідженні пропонується порівнювати як послідовності ДНК, так і двійкові послідовності на основі алгоритму MinHash з використанням у якості геш-функції національного стандарту ДСТУ 7564:2014 [12].

Метод порівняння двох послідовностей передбачає наступний порядок дій.

Алгоритм 4

1 Розбиття послідовностей на k-мери необхідної довжини

У нашому дослідженні пропонується перетворювати k-мери на бітові послідовності, що дасть можливість зберігати їх у вигляді 64-бітних чисел. Це, в свою чергу, дозволить розбивати послідовності ДНК на k-мери довжини до 32 включно, що є достатнім для будь-якого геному, а двійкові послідовності на k-мери довжини до 64 включно, що також дозволить оцінювати подібність доволі довгих двійкових послідовностей.

2 Гешування кожного з отриманих k-мерів

У якості геш-функції у нашому дослідженні буде використовуватися геш-функція Купина. Найменшим виходом Купини є 32 байтове геш-значення, тому для можливості представлення набору гешів у вигляді 64-бітних чисел взято тільки перші 8 байтів отриманих гешів. Такий підхід є можливим, оскільки криптографічні геш-функції розроблені таким чином, що є можливим зрізання вихідних даних до певного розміру, і зрізана геш-функція залишається безпечною криптографічною геш-функцією [13]. Зрізання також не є критичним з тієї причини, що для алгоритму MinHash не вимагається криптографічно стійка геш-функція.

3 Сортування отриманих гешів

4 Вибір невеликої кількості найменших геш-значень (скетчу), які і будуть представленням послідовності. Чим більш схожими є послідовності, тим більше MinHash вони ділитимуть між собою.

Для оцінки схожості цих двох наборів пропонується використовувати MinHash оцінку Жакарра. Для наборів k-мерів A та B алгоритм MinHash оцінює індекс Жакарра наступним чином:

$$jaccard(A_s, B_s) = \frac{|A_s \cap B_s|}{s}, \tag{3}$$

де $A_s, B_s$ – підмножини такі, що $|A_s \cup B_s|$ дорівнює розміру скетчу, $s$. Розмір скетчу відповідає кількості MinHash, які зберігаються. Більші скетчі краще відображають послідовність, але за рахунок більшого розміру файлів та довшого часу порівняння.

Похибка оцінки MinHash відстані для заданого розміру скетчу $s$ дорівнює $\sqrt{\frac{1}{s}}$ [11].

Після отримання індексу Жакарра MinHash відстань оцінюється за формулою

$$\frac{-\log(2.0 \times jaccard)/(1.0 + jaccard)}{k}, \tag{4}$$

де $k$ – обраний розмір k-мерів. Такий метод дозволяє доволі швидко і з розумними затратами продуктивності оцінити схожість двох послідовностей ДНК з доволі точним наближенням.

Точність описаних методів порівняння послідовностей оцінюється у п. 4 даної роботи.

## 3. Аналіз статистичних властивостей випадкових та псевдовипадкових послідовностей на основі ДНК

### 3.1. Статистичне тестування псевдовипадкових послідовностей

У даному пункті надані результати дослідження послідовностей, отриманих шляхом виконання алгоритму 1 (п. 2.1). Дослідження проводилося з використанням набору тестів NIST STS [14]. Цей набір тестів для тестування генераторів випадкових чи псевдовипадкових чисел дає змогу з високою часткою ймовірності судити про те, наскільки послідовність, що генерується досліджуваним примітивом, є статистично безпечною, оскільки він включає у себе найбільш розповсюджені тести, які охоплюють більшість аспектів випадковості послідовності. Довжина послідовностей 13 МБ.

Таблиця 2

Результати статистичного тестування псевдовипадкових послідовностей на виході ДСТУ 7624:2014

| Довжина ключа | Послідовність | Кількість тестів, у яких тестування пройшли більш 96 % послідовностей | Кількість тестів, у яких тестування пройшли більш 99 % послідовностей | Кількість тестів для яких значення P<0.001 | Успіх |
|---|---|---|---|---|---|
| 128 | HS_Kalyna128Encrypted | 188 (99 %) | 143 (76 %) | 1 | + |
| | EC_Kalyna128Encrypted | 188 (99 %) | 127 (67 %) | 0 | + |
| | FC_Kalyna128Encrypted | 188 (99 %) | 134 (70 %) | 0 | + |
| | VO_Kalyna128Encrypted | 189 (100 %) | 138 (73 %) | 1 | + |
| | MV_Kalyna128Encrypted | 188 (99 %) | 131 (69 %) | 0 | + |
| | AP_Kalyna128Encrypted | 187 (99 %) | 130 (69 %) | 0 | + |

Як видно з результатів дослідження, всі згенеровані послідовності успішно пройшли статистичне тестування. Найкращий результат для більш жорсткого критерію показала послідовність HS_Kalyna128Encrypted (з ДНК людини), для неї успішно пройдено 143 тести (76%). Для більш послабленого критерію найкращою є послідовність

VO_Kalyna128Encrypted (з ДНК калини), оскільки проходить всі 189 тестів (100 %). На рис. 4 представлено статистичні портрети досліджених послідовностей.

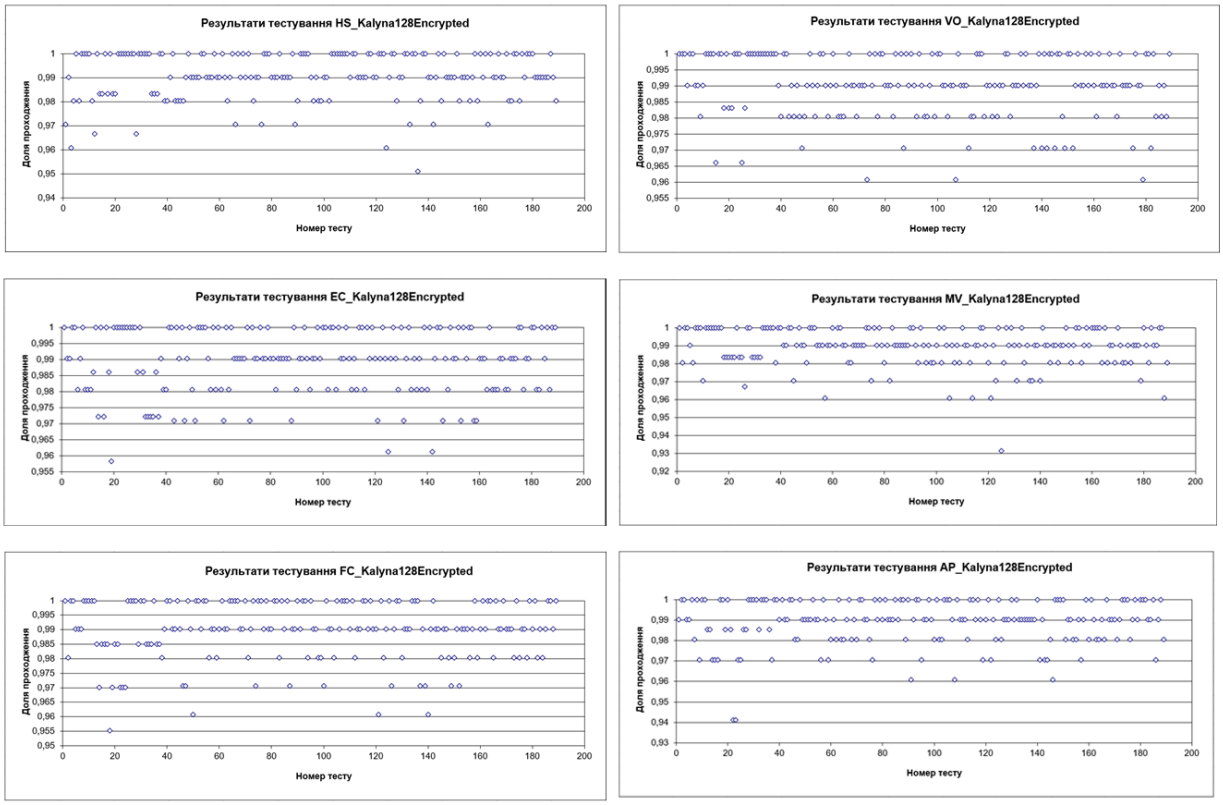


Рис. 4. Статистичні портрети псевдовипадкових послідовностей довжини 13 МБ

Результати тестування з використанням набору NIST STS показують, що псевдовипадкові послідовності з ДНК на виході алгоритму Калина мають гарні статистичні характеристики.

Оскільки результати для різних ДНК послідовностей є доволі схожими, у подальших дослідженнях будуть використовуватися тільки послідовності з ДНК людини.

Також для перевірки можливості генерування ПВП більшої довжини було отримано три послідовності довжини 151739136 байтів (144 МБ) та одну 1517391360 байтів (1,41 ГБ). Частину ключа зашифрування та синхропосилки для кожного з випадків показано у табл. 3.

Таблиця 3

Ключ зашифрування та синхропосилка для генерування ПВП

| Генерування послідовностей довжини 13 МБ, першої послідовності довжини 144 МБ та довжини 1,41 ГБ | |
|---|---|
| Ключ | 0x7E, 0x81, …, 0xD1, 0xB6 |
| Синхропосилка | 0xF3, 0x42, …, 0xD2, 0xFC |
| Генерування другої послідовності довжини 144 МБ | |
| Ключ | 0x92, 0x48, …, 0xBC, 0x74 |
| Синхропосилка | 0x8A, 0x04, …, 0x2C, 0xDE |
| Генерування третьої послідовності довжини 144 МБ | |
| Ключ | 0xFF, 0x06, …, 0x02, 0x4D |
| Синхропосилка | 0x73, 0xA4, …, 0xFE, 0x3F |

Дані значення є частиною послідовності, отриманої з квантового генератора.

Послідовності довжини 151739136 байтів вдалося успішно та доволі швидко протестувати з використанням NIST STS. Результати тестування у вигляді статистичних портретів надаються на рис. 5.

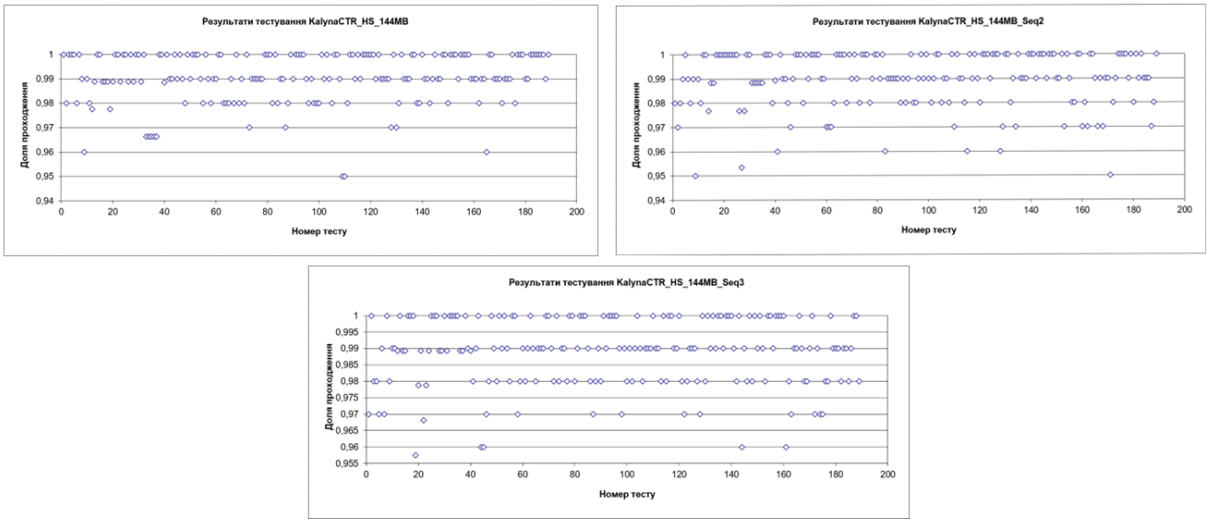


Рис. 5. Статистичні портрети псевдовипадкових послідовностей довжини 144 MB

Отримані з використанням NIST STS статистичні портрети також показують, що послідовності успішно проходять майже всі тести.

Додатково було виконано статистичне тестування послідовності довжини 1,41 ГБ. Тестування такої послідовності відбувалося майже 24 години, тому таке дослідження є доволі часомістким. Зауважте, що у статистичному портреті упущено значення для тесту «Перевірки шаблонів, що перекриваються», оскільки значення тільки для цього тесту є незадовільним (56/100) і псує наочність представлення результатів інших тестів (рис. 6).

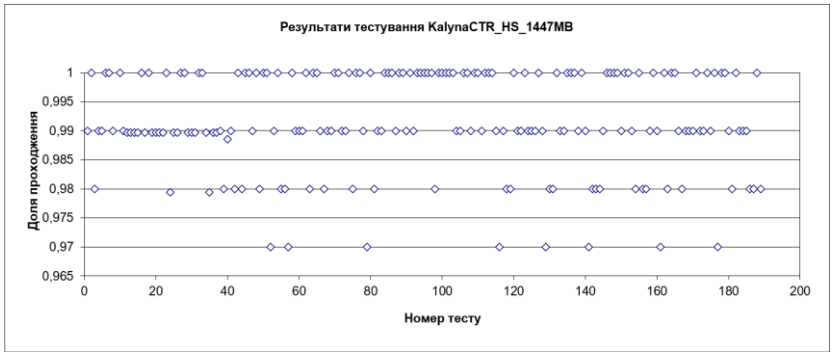


Рис. 6. Статистичний портрет послідовності KalynaCTR_HS_1447MB
(без урахування тесту шаблонів, що перекриваються)

Провал тесту «Перевірки шаблонів, що перекриваються» може свідчити про велику кількість *m*-бітних серій з одиниць у послідовності. Проте, це також може вказувати, що такі тести не розраховані на настільки довгі послідовності. Це перевірено у наступному пункті.

З отриманих результатів можна зробити висновок, що навіть досить довгі послідовності на виході ДСТУ 7624:2014 мають хороші статистичні характеристики. Це вказує на те, що такі послідовності можуть бути використані у якості ПВП для необхідних задач.

### 3.2. Статистичне тестування випадкових послідовностей

У даному пункті надано результати дослідження послідовностей, отриманих з використанням алгоритму 2 (п. 2.2).

За допомогою екстрактора було отримано три послідовності довжини 150994944 байтів (144 МБ) та одну 1509949440 байтів (1,40 ГБ). У якості основи було взято частину двійкової послідовності з ДНК людини. Ключ зашифрування та синхропосилка для кожної ітерації зашифрування отримувалася з використанням NPTRNG /dev/random [6] для ОС Linux. Тобто ключ та nonce при кожному повторному зашифруванні є випадковими даними, що і передбачено стандартом ДСТУ 7624:2014 для генерації випадкових послідовностей.

Результати аналізу властивостей у вигляді статистичних портретів надаються далі (рис. 7):

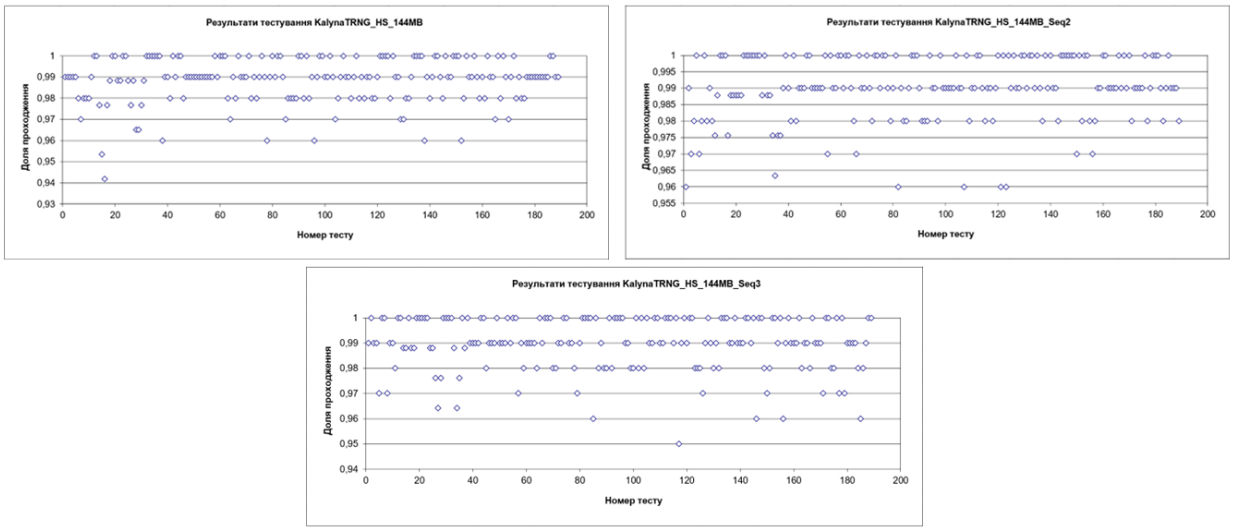


Рис. 7. Статистичні портрети випадкових послідовностей довжини 144 МВ

Отримані статистичні портрети підтверджують, що послідовності успішно проходять тестування.

Для послідовності довжини 1.4 ГБ тест «Перевірки шаблонів, що перекриваються» також показав незадовільний результат (55/100), проте всі інші тести було успішно пройдено. Для представлення у вигляді статистичного портрету значення проваленого тесту також буде упущено (рис. 8):

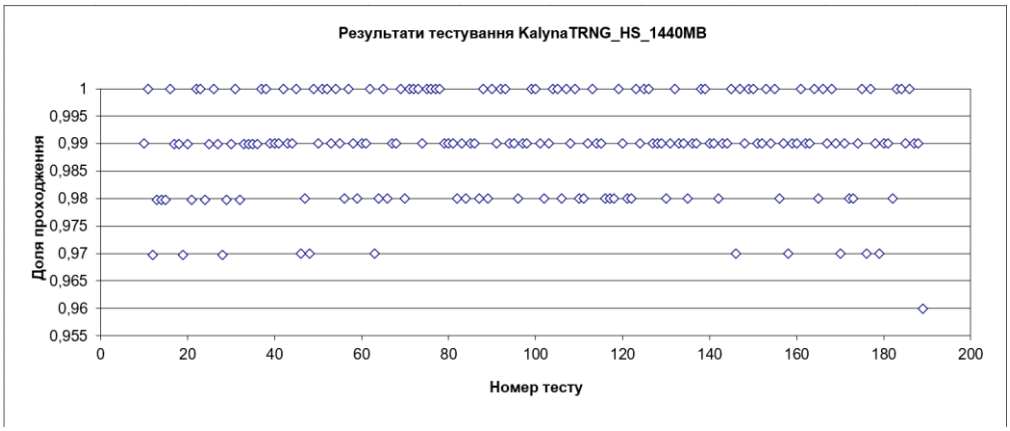


Рис. 8. Статистичний портрет послідовності KalynaTRNG_HS_1447 МВ
(без урахування тесту шаблонів, що перекриваються)

Для перевірки припущення про помилковість цього тесту для дуже довгих послідовностей було додатково перевірено послідовності великої довжини з обґрунтованих генераторів, а саме з /dev/random та CryptGenRandom [15]. Дослідження показало результати тесту «Перевірки шаблонів, що перекриваються» 59/100 та 54/100 відповідно. Хоча, як і для послідовності з екстрактора, всі інші тести було пройдено успішно. Це може означати, що саме даний тест не підходить для перевірки послідовностей таких довжин, а отже його результат не слід враховувати при оцінці.

Випадкові послідовності на виході екстрактора також майже повністю проходять всі тести з набору NIST STS. Це вказує на те, що послідовності з екстрактора на основі ДСТУ 7624:2014 можуть бути використані у якості ВП для необхідних задач.

У наступному пункті оцінено подібність послідовностей: як ДНК, так і двійкових.

### 4. Аналіз подібності ПВП, ВП та ДНК послідовностей для різних організмів

Для оцінки подібності пропонуються два розроблених методи на основі вимірювання k-мер відстані та MinHash відстані (пп. 2.3 та 2.4). Ці методи підходять як для оцінки ДНК послідовностей, які у даному дослідженні виступають у якості ДШ, так і двійкових послідовностей, на виході екстракторів.

#### 4.1. Метод підрахунку k-мерів та k-мер відстані

##### 4.1.1. Порівняння послідовностей ДНК

Для демонстрації методу виконано порівняння послідовностей ДНК для різних організмів з використанням алгоритму 3 (п. 2.3). Також перевірено двійкові послідовності, отримані на виході ДСТУ 7624:2014 на схожість з використанням цього алгоритму. Для перевірки обрано наступні параметри: $k$=5, як визначено за замовчуванням у пакеті R, також обрано $k$=7, 9 та 11, оскільки для підрахунку краще використовувати k-мери непарної довжини. Додатково було виконано перевірку на великій довжині k-мерів – $k$=31. Зверніть увагу, що схожість послідовностей визначається як (1–k-мер відстань) (табл. 4).

Таблиця 4

Результат обчислення k-мер відстані для послідовностей ДНК

| **Ген GNRHR2** | | | | | | |
|---|---|---|---|---|---|---|
| Послідовність 1 | Послідовність 2 | k-мер відстань | | | | |
| | | $k$=5 | $k$=7 | $k$=9 | $k$=11 | $k$=31 |
| Послідовність з ДНК людини | Послідовність з ДНК шимпанзе | 0.008139 | 0.020754 | 0.0288010 | 0.0362676 | 0.1056564 |
| Послідовність з ДНК людини | Послідовність з ДНК макаки резус | 0.019566 | 0.104332 | 0.1654499 | 0.2033711 | 0.4711847 |

Як видно з отриманих результатів, метод справді дозволяє з доволі точним наближенням оцінити схожість ДНК послідовностей. Найкращі результати метод показує саме у проміжку між розрахованими розмірами k-мерів 7 – 11. Отже, розрахунок розміру k-мерів відповідно до розміру послідовності і алфавіту є необхідним, чим більш точним є цей розрахунок, тим більш точною буде оцінка відстані.

Оскільки такий алгоритм можливо застосувати до послідовностей з двійковим алфавітом, далі перевірено послідовності різної довжини, отримані на виході ДСТУ 7624:2014 на подібність.

##### 4.1.2. Порівняння ВП з екстрактора

Для оцінки подібності ВП на виході екстрактора згенеровано по дві послідовності різних довжин: 2, 5 та 13 МБ. Для генерування кожної з двох послідовностей у парі було використано різні частини послідовності ДНК людини. Довжини обрано таким чином, щоб довжиною k-мерів для їх оцінки було 31, 33 та 35 відповідно.

Кожну з пар послідовностей буде перевірено з використанням такої довжини, а також з довжинами, що є на крок меншими та більшими – 29 та 37.

У табл. 5 наведено результати оцінки k-мер відстані між згенерованими послідовностями.

Таблиця 5

Результат обчислення k-мер відстані для двійкових ВП з екстрактора

| П1 | П2 | k-мер відстань | | | | |
|---|---|---|---|---|---|---|
| | | $k$=29 | $k$=31 | $k$=33 | $k$=35 | $k$=37 |
| Kalyna_2MB_Seq1 | Kalyna_2MB_Seq2 | 0.884817 | **0.967398** | 0.991574 | 0.997882 | 0.999470 |
| Kalyna_5MB_Seq1 | Kalyna_5MB_Seq1 | 0.763548 | 0.923910 | **0.979371** | 0.994756 | 0.998689 |
| Kalyna_13MB_Seq1 | Kalyna_13MB_Seq1 | 0.572017 | 0.828964 | 0.948761 | **0.986419** | 0.996545 |

Як видно з отриманих результатів, при використанні найбільш відповідної для довжини послідовностей довжини k-мерів (виділено жирним) – результати подібності двійкових

послідовностей є близькими до 1 – 3 %. Такий показник вказує на те, що послідовності на виході алгоритму Калина (ДСТУ 7624:2014) у режимі лічильника (гамування) є рівномірними та не схожими одна на одну (рис. 9).

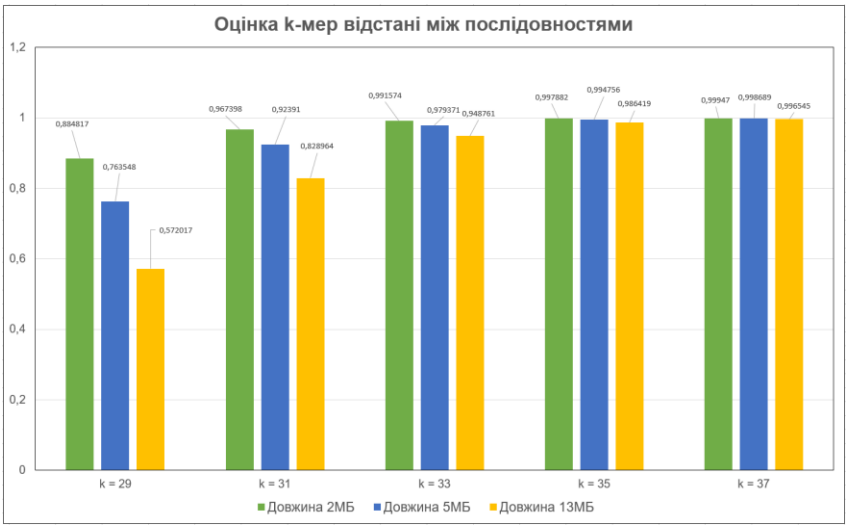


Рис. 9. Оцінка k-мер відстані між двійковими послідовностями на виході ДСТУ 7624:2014

У наступному пункті представлено покращений метод на основі MinHash з використанням у якості геш-функції алгоритму Купина (ДСТУ 7564:2014) [11].

### 4.2. Метод на основі MinHash відстані з використанням ДСТУ 7564:2014

#### 4.2.1. Порівняння послідовностей ДНК

В табл. 6 наведено результати обчислення MinHash відстані для послідовностей ДНК різних організмів з алгоритмом 4 (п. 2.4).

Таблиця 6

Результат обчислення MinHash відстані для послідовностей ДНК

| Ген GNRHR2 | | | | | |
|---|---|---|---|---|---|
| Послідовність 1 | Послідовність 2 | MinHash відстань | | | |
| | | *k*=7 | *k*=9 | *k*=11 | *k*=31 |
| Послідовність з ДНК людини | Послідовність з ДНК шимпанзе | 0.0159405 | 0.0166228 | 0.0155702 | 0.0105054 |
| Послідовність з ДНК людини | Послідовність з ДНК макаки резус | 0.0583056 | 0.0650214 | 0.0612689 | 0.0473965 |

З отриманих результатів можна бачити, що оцінка, починаючи з k=11, є доволі точною і відповідає результатам, які можна отримати іншими інструментами порівняння ДНК, в тому числі і з вирівнюванням. Отримані результати вказують на те, що алгоритм дозволяє з достатньо хорошим наближенням оцінити подібність двох ДНК послідовностей. У наступному пункті виконано оцінку двійкових ВП з екстрактора таким самим методом.

#### 4.2.2. Порівняння ВП з екстрактора

Для оцінки подібності ВП на виході екстрактора були взяті ті самі послідовності, що і у підп. 4.1.2. У табл. 7 наведено результати оцінки MinHash відстані між згенерованими послідовностями.

Як видно з отриманих результатів, при досягненні необхідного розміру k-мерів для відповідної довжини послідовності, відстань між послідовностями оцінюється у приблизно 15 – 16 %, при цьому показник подібності за індексом Жаккара є майже нульовим (0,001 – 0,005). Це означає, що такий метод для двійкового алфавіту може бути недостатньо чутливим. Проте при перевищенні розміру k-мерів на 1 – відстань оцінюється уже в 100 %, що означає повну відмінність між скетчами послідовностей (набором їхніх мінімальних гешів).

Таблиця 7

Результат обчислення MinHash відстані для двійкових послідовностей

| П1 | П2 | MinHash відстань (попередньо порахований індекс Жакарра) | | | | |
|---|---|---|---|---|---|---|
| | | *k*=29 | *k*=31 | *k*=33 | *k*=35 | *k*=37 |
| Kalyna_2MB_Seq1 | Kalyna_2MB_Seq2 | 0.129022 (0.012) | **0.152819 (0.0044)** | 0.195108 (0.0008) | 1.00 (0.00) | 1.00 (0.00) |
| Kalyna_5MB_Seq1 | Kalyna_5MB_Seq1 | 0.0980334 (0.03) | 0.123473 (0.011) | **0.155121 (0.003)** | 0.177589 (0.001) | 1.00 (0.00) |
| Kalyna_13MB_Seq1 | Kalyna_13MB_Seq1 | 0.0674871 (0.076) | 0.0974331 (0.025) | 0.146434 (0.004) | **0.177589 (0.001)** | 1.00 (0.00) |

Отже, представлені методи порівняння є значно простішими у реалізації, ніж попередньо досліджені методи з використанням вирівнювання, і здатні забезпечувати оцінку схожості послідовностей, як ДНК, так і двійкових, з достатньо точним наближенням, майже ідентичним до методів на основі вирівнювання. Такі методи мають значну перевагу у продуктивності, забезпечують кращий час виконання. Також завдяки покращенню розрахунку k-мер відстані з використанням алгоритму MinHash та геш-функції Купина вдалося досягти додаткового прискорення швидкодії та значної економії апаратних ресурсів комп'ютера, зокрема, оперативної пам'яті. Проте такий метод може мати дещо низьку чутливість при дослідженні двійкових послідовностей, які завідома мають бути несхожими, наприклад, походячи з довірених PTRNG чи NPTRNG, а отже потребує подальшого дослідження та більш точного налаштування.

**Висновки**

1. Попередні дослідження вказують на теоретичну можливість використання у якості нефізичного джерела шуму та відповідно ГВП – ДНК. Як показав аналіз наукових джерел, детальних досліджень в цьому напрямі немає чи вони недоступні. Для оцінки та порівняння ПВП та ВП на основі використання ДНК потрібно провести широкі практичні та теоретичні дослідження з використанням ентропійних (стохастичних) методів та методик, а також визнаних статистичних методик (бажано стандартизованих).

2. ДНК можливо подати у вигляді послідовності, яка складається з алфавіту, у якому можливі чотири значення – A, C, G та T. Така послідовність може бути представлена у вигляді двійкової послідовності шляхом заміни цих значень на відповідні двійкові комбінації – A=00, C=01, G=10 та T=11. Наприклад, маючи таку послідовність як на рис. 1 – CAGGATCGTAGA – отримаємо двійкову послідовність 010010100011011011001000.

3. Для проведення досліджень було обрано декілька зразків ДНК різних організмів. Обрані послідовності ДНК було представлено у вигляді двійкових, як описано вище. Для кожної з них було проведено статистичне та стохастичне тестування. Отримані результати надають інформацію про те чи мають «сирі» послідовності достатньо випадковості і чи потребують вони покращення.

4. Статистичні характеристики «сирих» послідовностей не є задовільним, майже всі статистичні тести провалені. Хоча більшість тестів мінімальної ентропії мають незадовільне значення, а показники ентропії Шеннона та колізійної ентропії є хорошими, що вказує на присутність певної кількості ентропії у таких послідовностях, це дає можливість застосування екстракції випадковості з використанням, наприклад, БСШ.

5. Для отримання гарних ПВП та ВП на основі ДНК можливо використати екстрактор на основі ДСТУ 7624:2014 з довжиною ключа 256 біт у режимі гамування (CTR), оскільки саме такий режим пропонується використовувати з блоковим шифром для отримання ПВП та ВП у AIS 20/31.

6. Дослідження проводилося з використанням набору тестів NIST STS. Цей набір тестів для тестування генераторів випадкових чи псевдовипадкових послідовностей дає можливість з високою часткою ймовірності судити про те, наскільки послідовність, що генерується досліджуваним примітивом, є статистично безпечною, оскільки він включає у себе найбільш

розповсюджені тести, які охоплюють більшість аспектів випадковості послідовності. Довжина таких послідовностей може бути 13 МБ.

7. Як видно з рис. 4, всі згенеровані послідовності успішно пройшли статистичне тестування. Найкращий результат для більш жорсткого критерію показала послідовність HS_Kalyna128Encrypted (з ДНК людини), для неї успішно пройдено 143 тести (76 %). Для більш послабленого критерію найкращою є послідовність VO_Kalyna128Encrypted (з ДНК калини), оскільки проходить всі 189 тестів (100 %).

8. За допомогою екстрактора було отримано три послідовності довжини 150994944 байтів (144 МБ) та одну 1509949440 байтів (1,40 ГБ). У якості основи було взято частину двійкової послідовності з ДНК людини. Ключ зашифрування та синхропосилка для кожної ітерації зашифрування отримувалася з використанням NPTRNG /dev/random [6] для ОС Linux.

9. Для оцінки подібності ВП на виході екстрактора згенеровано по 2 послідовності різних довжин: 2, 5 та 13 МБ. Для генерування кожної з двох послідовностей у парі було використано різні частини послідовності ДНК людини. Довжини обрано таким чином, щоб довжиною k-мерів для їх оцінки було 31, 33 та 35 відповідно.

10. Для порівняння двох ДНК послідовностей для кожної з них спочатку підраховується кількість k-мерів (підрядків) у її складі. Створюється таблиця, де кожному з k-мерів ($4^k$ для стандартного алфавіту ДНК і, наприклад, $2^k$ у випадку двійкового алфавіту) відповідає кількість входжень конкретного k-меру у послідовність, що досліджується.

11. При досягненні необхідного розміру k-мерів для відповідної довжини послідовності, відстань між послідовностями оцінюється у приблизно 15 – 16 %, при цьому показник подібності за індексом Жаккара є майже нульовим (0,001 – 0,005). Це означає, що такий метод для двійкового алфавіту може бути недостатньо чутливим. Проте, при перевищенні розміру k-мерів на 1 сходинку вище – відстань оцінюється уже в 100.

12. Представлені методи порівняння ПВП та ВП ДНК значно простіші у реалізації, ніж попередньо досліджені методи з використанням вирівнювання, і здатні забезпечувати оцінку схожості послідовностей як ДНК, так і двійкових, з достатньо точним наближенням, майже ідентичним до методів на основі вирівнювання. Такі методи також мають значну перевагу у продуктивності, забезпечують кращий час виконання.

13. Таким чином, оцінка подібності послідовностей з використанням k-мер відстані показує, що при відповідних до довжини послідовності значеннях *k* подібність між послідовностями на виході екстрактора є мінімальною.

14. У цілому вважаємо, що актуальними та необхідними є подальші дослідження, оцінка та порівняння різних ДНК, встановлення подібності ПВП різних ДНК тощо.

**Список літератури:**


1. Matthias Peter, Werner Schindler. A Proposal for Functionality Classes for Random Number Generators. Version 2.36 – Current intermediate document for the AIS 20/31 workshop. 2023. URL: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Certification/Interpretations/AIS_31_Functionality_classes_for_random_number_generators_e_2023.html?nn=910324
2. Горбенко І. Д., Дерев'янко Я. А., Горбенко Д. Ю. ДНК – джерело шуму та нефізичних випадкових послідовностей. Основні положення практичних досліджень. 2024. URL: https://cyberwarfare.viti.edu.ua/assets/files/Cyberwarfare_2024.pdf
3. Національна Бібліотека Медицини США. URL: https://ftp.ncbi.nlm.nih.gov/genbank/
4. ДНК банк Японії. URL: https://ddbj.nig.ac.jp/arsa/quick_search?lang=en;jsessionid=5FC7A34BD7FA0826284A6619E85BCA97
5. ДСТУ 7624:2014. Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення. 2015. URLhttps://online.budstandart.com/ua/catalog/doc-page.html?id_doc=109736
6. Linux manual page. random. URL: https://man7.org/linux/man-pages/man4/random.4.html
7. Shaun Wilkinson. Introduction to the kmer R package. 2019. URL: https://cran.r-project.org/web/packages/kmer/vignettes/kmer-vignette.html.
8. Edgar, R. C. Local homology recognition and distance measures in linear time using compressed amino acid alphabets. 2004. URL: https://europepmc.org/backend/ptpmcrender.fcgi?accid=PMC373290&blobtype=pdf
9. Mash. Fast genome and metagenome distance estimation using MinHash. Sketches. URL: https://mash.readthedocs.io/en/latest/sketches.html

10. Andrei Z. Broder. On the resemblance and containment of documents. 1998. URL: https://web.archive.org/web/20150131043133/http://gatekeeper.dec.com/ftp/pub/dec/SRC/publications/broder/positano-final-wpnums.pdf

11. Brian D. Ondov, Todd J. Treangen, Páll Melsted, Adam B. Mallonee, Nicholas H. Bergman, Sergey Koren & Adam M. Phillippy. Mash: fast genome and metagenome distance estimation using MinHash. 2016. URL: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-016-0997-x

12. ДСТУ 7564:2014. Інформаційні технології. Криптографічний захист інформації. Функція гешування. 2015. URL: https://usts.kiev.ua/wp-content/uploads/2020/07/dstu-7564-2014.pdf.

13. John Kelsey. Truncation Mode for SHA. 2005. URL: https://csrc.nist.gov/csrc/media/events/first-cryptographic-hash-workshop/documents/kelsey_truncation.pdf

14. NIST SP 800-22 Rev. 1. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. 2010, URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf

15. Microsoft Documentation. CryptGenRandom function (wincrypt.h). 2021. URL: https://learn.microsoft.com/en-us/windows/win32/api/wincrypt/nf-wincrypt-cryptgenrandom

*Надійшла до редколегії 10.06.2024*

*Відомості про авторів:*

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

**Єсіна Марина Віталіївна** – канд. техн. наук, доцент, Харківський національний університет імені В. Н. Каразіна, доцент кафедри безпеки інформаційних систем і технологій, факультет комп'ютерних наук, АТ «Інститут Інформаційних Технологій», науковий співробітник-консультант; Україна; e-mail: m.v.yesina@karazin.ua; ORCID: https://orcid.org/0000-0002-1252-7606

**Горбенко Дмитро Юрійович** – Харківського національного університету імені В. Н. Каразіна, студент факультету комп'ютерних наук, АТ "Інститут Інформаційних Технологій", молодший інженер-програміст; Україна; e-mail: jsciitua@gmail.com