DOI: 10.18372/2410-7840.19.11478
УДК 003.26:621.39:530.145(045)

ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МЕТОДУ ГЕНЕРУВАННЯ ТРИТОВИХ ПСЕВДОВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ ДЛЯ КРИПТОГРАФІЧНИХ ЗАСТОСУВАНЬ

Михайло Шелест, Сергій Гнатюк, Тетяна Жмурко, Василь Кінзерявий, Халіча Юбузова

В останні декілька років розвиток квантової криптографії привернув увагу науковців всього світу та перейшов від рівня теоретичних досліджень до впровадження готових комерційних рішень. Основна відмінність методів квантової криптографії від традиційної (симетричної та асиметричної) криптографії полягає у використанні в своїй основі зовсім різних принципів. Квантова криптографія не залежить від обчислювальних чи інших можливостей порушника, використовує специфічні унікальні властивості квантових частинок і ґрунтується на непорушності законів квантової фізики. Серед найрозвиненіших технологій квантової криптографії варто відзначити квантовий прямий безпечний зв'язок, який дозволяє передавати інформацію відкритим каналом без попереднього її шифрування – нівелюючи проблему розподілу ключів, в цих протоколах кожний біт є конфіденційною інформацію і не повинен потрапити до порушника, тому вимоги до стійкості протоколів є значно вищими і потребують методів підсилення безпеки. Авторами був розроблений метод забезпечення стійкості протоколів квантової криптографії, однак для його реалізації необхідно використання надійних тритових псевдовипадкових послідовностей. У зв'язку з цим у роботі представлено експериментальне дослідження розробленого методу генерування тритових послідовностей, для оцінювання ефективності цього методу було розроблено методику проведення експерименту, яка підтвердила можливість використання згенерованих тритових послідовностей для реалізації запропонованого метода забезпечення стійкості кутритових протоколів квантової криптографії до некогерентних атак, а також для інших криптографічних застосувань в сучасних інформаційно-комунікаційних технологіях.

Ключові слова: *квантова криптографія, генератор псевовипадкових послідовностей, квантовий прямий безпечний зв'язок, трит, захист інформації.*

Вступ. В основу найбільш поширених криптографічних систем покладено принцип захисту, який полягає у великій складності дешифрування повідомлень і вимагає великих обчислювальних потужностей, однак повністю не виключає можливості дешифрування криптограми. Одним з найбільш перспективних і досить нових напрямків захисту інформації є квантова криптографія (КК). На відміну від більшості класичних криптосистем, захищеність яких ґрунтується на недоведених математичних припущеннях, захищеність квантових криптографічних систем спирається на фундаментальні закони квантової механіки, що при належній реалізації таких систем робить принципово неможливим перехоплення інформації і порушення її конфіденційності неавторизованими користувачами.

Однією з найбільш розвинених технологій КК, поряд з квантовим розподілом ключів, є квантовий прямий безпечний зв'язок (КПБЗ), який дозволяє передавати інформацію відкритим каналом напряму (без попереднього її шифрування – проблема розподілу ключів нівелюється). На сьогодні запропоновано велику кількість методів КПБЗ [1-3, 6-8], що базуються на різних квантових технологіях і можуть використовуватись як для захищеного передавання інформації (з використанням кубітів або кудитів), так і для розподілу криптографічних ключів.

Проте, в протоколах КПБЗ кожний біт є конфіденційною інформацію і не повинен потрапити до порушника, тому вимоги до стійкості протоколів КПБЗ є значно вищими, ніж до стійкості протоколів квантового розподілу ключів. Недоліком протоколів КПБЗ є те що вони мають лише асимптотичну стійкість до некогерентних атак і, безумовно, потребують методів підсилення безпеки.

Аналіз існуючих досліджень і постановка завдання. У [1, 7] описані методи підсилення безпеки протоколів КПБЗ, а в [6] авторами був запропонований і теоретично обґрунтований метод забезпечення стійкості протоколів КПБЗ.

Застосування розробленого у [6] методу зебезпечення стійкості кутритових протоколів квантової криптографії потребує генерування трійкових (тритових) псевдовипадкових ослідовностей (ПВП), за рахунок чого збільшується інформаційна місткість протоколів передавання. Проведений аналіз дозволив виявити достатню кількість існуючих генераторів ПВП (ГПВП), які можуть використовуватись для різного роду застосувань [4, 5, 7-11, 14, 16]. Стандарт ISO/IEC 18031, який встановлює концептуальні моделі, термінологію і

вимоги, що відносяться до конструктивних елементів і властивостей систем, які використовуються для генерування випадкових бітів в криптографічних застосуваннях, визначає два типи генераторів: недетерміновані (механізм генерування випадкових бітів, який використовує джерело ентропії для генерування випадкового потоку бітів) і детерміновані (механізм генерування бітів, який використовує детерміновані механізми, такі як криптографічні алгоритми, на джерелі ентропії для генерування випадкового потоку бітів) генератори випадкових бітів [5].

За способом отримання ГПВП діляться на три принципово різних класи: табличні – основний недолік є скінченими, фізичні (радіоактивне випромінювання, фізичні генератори шумів, квантові генератори, генератори імпульсних послідовностей тощо) – спільними і найбільш суттєвими недоліками, що утрудняють їх застосування є обмежена швидкодія, низька стабільність основних імовірнісних характеристик, що пояснюється нестабільністю первинних джерел, дрейфом параметрів перетворюючих схем, джерел живлення та вимагає періодичної статистичної перевірки якості; складність апаратурної реалізації, алгоритмічні (ГПВП, наприклад, метод серединних квадратів, метод серединних добутків, метод перемішування, лінійний конгруентний метод) [14].

Також ГПВП можна класифікувати за різними законами розподілу [4], проте найбільш повна класифікація представлена у роботі [7]. Відповідно до цієї класифікації методи формування ПВП поділяються на криптостійкі та не криптостійкі. До яких у свою чергу відносять криптостійкі: на основі потокових шифрів (наприклад, Dragon-128, SEAL, RC4, RC5, RC6, Grain, Yamb, Phelix), на основі блокових шифрів (наприклад, ГОСТ 28147-89, AES, ANSI X9.17, DES), на основі односторонніх функцій (наприклад, генератори BBS, RSA, Dual_EC_DRBG (еліптичні криві), GPSSD (лінійні коди) та ін.) і не криптостійкі: на основі елементарних рекурентів (наприклад, лінійний конгруентний генератор, поліноміальний конгруентний генератор, адитивний генератор Фібоначчі, адитивний генератор Фібоначчі з запізненням, мультиплікативний генератор Фібоначчі з запізненням), на основі операцій в кінцевих полях (наприклад, генератори Галуа, Де Брейна, Фібоначчі, адитивний, Голмана, стискаючий тощо). Проте всі розроблені на сьогодні методи генерування ПВП (генератори) орієнтовані на бінарні послідовності, а отже розробка методу генерування тритових ПВП є актуальним науковим завданням. З огляду на це, в роботі [12] було запропоновано метод генерування тритових ПВП ξ із множиною векторів внутрішніх станів V_p $(V_p = \{0, 1, 2\}^p)$, множиною секретних ключів V_n, множиною векторів ініціалізації V_e та множиною вихідних послідовностей V_m, де $p = 14 \cdot l$, $n = 4 \cdot l$, $e = p - n = 10 \cdot l$, $m = b \cdot n$, $l = d \cdot s$ і b, d, s – натуральні числа. Для верифікації методу необхідно розробити відповідне програмне забезпечення. З огляду на це, *метою статті* є проведення експериментального дослідження методу генерування тритових послідовностей для оцінювання його ефективності в криптографічних застосуваннях.

Теоретичне обґрунтування запропонованого методу. У методі генерування тритових ПВП для генерування вихідних даних виконуються такі етапи [12]:

Етап 1 – Ініціалізація вектора внутрішнього стану. Виконується початкова ініціалізація вектора внутрішнього стану U на основі вектора ініціалізації VI та секретного ключа K, $U \in V_p$, $VI \in V_e$, $K \in V_n$.

Нехай $U = (x_1, x_2, x_3, x_4, x_5, x_6, y_1, y_2, y_3, y_4, k_1, k_2, k_3, k_4)$, де x_i, y_j, k_j – частини вектора внутрішнього стану U ($x_i \in V_l$, $y_j \in V_l$, $k_j \in V_l$, $i \in \overline{1,6}$, $j \in \overline{1,4}$); $VI = (VI_1, VI_2, VI_3, VI_4, VI_5, VI_6, VI_7, VI_8, VI_9, VI_{10})$, де VI_o – частини вектора ініціалізації VI ($VI_o \in V_l$, $o \in \overline{1,10}$); $K = (K_1, K_2, K_3, K_4)$, де K_w – частини секретного ключа K ($K_w \in V_l$, $w \in \overline{1,4}$).

Тоді внутрішній стан вектора U ініціалізується таким чином: $x_i = VI_i$, $y_j = VI_{6+j}$, $k_j = K_j$, $i \in \overline{1,6}$, $j \in \overline{1,4}$.

Етап 2 – Генерування ПВП. На основі поточних значень внутрішнього стану вектора U виконується поступова генерування вихідної послідовності $M = (M_1, ..., M_b)$, $M \in V_m$, M_q – частини вихідної послідовності M, $M_q \in V_n$, $q \in \overline{1,b}$. Зауважимо, що при генерування кожного M_q поточні значення внутрішнього стану вектора U весь час змінюються.

2.1. Для генерування частини вихідної послідовності M_q r-разів ($q \in \overline{1,b}$, r – натуральне число) виконується наступне:

2.1.1. *Розраховуються нові значення векторів* x_1, x_2, x_3. Спочатку визначається x_1: $x_1' = Sbox(x_1 + k_1)$; $x_1 = (x_1' \oplus x_4) <<< k_4$. Далі обраховується x_2: $x_2' = Sbox(x_2 + k_2)$; $x_2 = (x_2' + x_5) >>> k_3$. Наприкінці розраховується x_3: $x_3' = (x_3 + x_6) \oplus y_3$; $x_3 = Mix(x_3') <<< x_1$.

2.1.2. *Обчислюються нові значення векторів* k_1, k_2, y_1, y_2. Спочатку визначаються k_1 та k_2: $k_1' = Sbox(x_1 \oplus k_1) + x_5$; $k_1 = Sbox(k_1' \oplus y_1)$; $k_2' = Mix(x_2 + k_2 + x_6)$; $k_2 = Sbox(k_2' \oplus y_2)$. Далі обраховуються значення y_1 та y_2: $y_1' = (k_1 + y_1) <<< x_2$; $y_1 = Sbox(y_1' \oplus k_3)$; $y_2' = ((k_2 + y_2) >>> x_3) \oplus k_4$; $y_2 = Mix(Sbox(y_2'))$.

2.1.3. *Розраховуються нові значення векторів* x_4, x_5, x_6. Спочатку визначається x_4: $x_4' = Sbox(x_4 + k_3)$; $x_4 = (x_4' \oplus x_1) <<< k_2$. Далі обраховується x_5: $x_5' = Sbox(x_5 + k_4)$; $x_5 = (x_5' + x_2) >>> k_1$. Наприкінці розраховується x_6: $x_6' = (x_6 + x_3) \oplus y_1$; $x_6 = Mix(x_6') <<< x_4$.

2.1.4. *Обчислюються нові значення векторів* k_3, k_4, y_3, y_4. Спочатку визначаються k_3 та k_4: $k_3' = Sbox(x_4 \oplus k_3) + x_2$; $k_3 = Sbox(k_3' \oplus y_3)$; $k_4' = Mix(x_5 + k_4 + x_3)$; $k_4 = Sbox(k_4' \oplus y_4)$. Далі обраховуються значення y_3 та y_4: $y_3' = (k_3 + y_3) <<< x_5$; $y_3 = Sbox(y_3' \oplus k_1)$; $y_4' = ((k_4 + y_4) >>> x_6) \oplus k_2$; $y_4 = Mix(Sbox(y_4'))$.

2.2. За допомогою конкатенації векторів y_i обраховується вихідна послідовність M_q: $M_q = (y_1 \,|\, y_2 \,|\, y_3 \,|\, y_4)$.

У зазначених виразах, символи $\oplus$ та $+$ відповідають операціям покоординатного додавання за модулем 3 та алгебраїчну операцію додавання за модулем 3^l відповідно. Під операцією $X <<< Y$ розуміємо операцію циклічного зсуву вліво числа X на Y позицій, а під $X >>> Y$ – циклічного зсуву вправо числа X на Y позицій. Під операцією $Sbox(X)$ розуміємо операцію в якій X розбивається на d частин довжиною s тритів, над кожною з яких виконується підстановка на множині V_s: $Sbox(X) = (S(X_1), ..., S(X_d))$, де $X = (X_1, ..., X_d)$, $X \in V_l$, $X_i \in V_s$, $i \in \overline{1,d}$, а S – підстановка на зазначеній множині. $Mix(X)$ відповідає операції у якій виконується лінійне розсіювання тритів вектора X (як $Mix(X)$ можуть бути використані МДР-коди).

Алгоритм TriGen. На основі запропоновано метода генерування тритових ПВП ξ розроблено алгоритм TriGen (псевдокод алгоритму зображено на рис. 1). У алгоритмі TriGen використовуються такі параметри $d = 4$, $s = 6$, $l = d \cdot s = 24$, $p = 14 \cdot l = 336$, $n = 4 \cdot l = 96$, $e = p - n = 10 \cdot l = 240$, $m = b \cdot n = 96 \cdot b$, $r = 4$, b – натуральне число.

В операції $Sbox(X)$ виконується нелінійна заміна кожних шести тритів числа X на відповідне їм значення таблиці підстановок. Використовується лише одна таблиця підстановок, що побудована за допомогою обрахунку зворотного елементу поля $(X)^{-1} \in GF(3^6)$ з подальшим виконанням афінного перетворення над полем $GF(3)$: $S(X) = M \cdot (X)^{-1} + V$, де $X, V \in GF(3^6)$, а M – квадратна невироджена матриця над полем $GF(3)$ розміром 6×6. Кінцеве поле GF(3^6) фіксується кільцем многочленів з операціями за модулем незвідного многочлена $m(x) = x^6 + x + 2$.

Для побудови запропонованої таблиці замін були обрані такі значення матриці M та вектора V:

$$M = \begin{pmatrix} 1 & 2 & 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 & 1 & 1 \\ 1 & 2 & 1 & 2 & 0 & 1 \\ 1 & 0 & 2 & 1 & 2 & 0 \\ 0 & 1 & 0 & 2 & 1 & 2 \\ 2 & 0 & 1 & 1 & 2 & 1 \end{pmatrix}, \quad V = \begin{pmatrix} 0 \\ 2 \\ 2 \\ 1 \\ 0 \\ 2 \end{pmatrix}.$$

В операції $Mix(X)$ квадратна не вироджена матриця M' над полем $GF(3)$ розміром 24×24 тритів множиться на X (представлений у вигляді вектора-стовпчика) над полем $GF(3)$. Матриця M' побудована на основі масиву U таким чином: $M'[i][j] = U[(j + 24 - i) \bmod 24]$, де $i, j = \overline{0, 23}$, а масив U приймає значення: U = {1, 0, 2, 2, 1, 0, 2, 0, 1, 1, 1, 2, 0, 1, 2, 1, 0, 2, 0, 0, 1, 2, 0, 2 }.

TriGen

Input: вектор ініціалізації VI, секретний ключ K, $VI \in V_{240}$, $K \in V_{96}$, параметр b.

Output: вихідна послідовність $M = (M_1,...,M_b)$, $M \in V_{96b}$, $M_q \in V_{96}$, $q \in \overline{1,b}$.

1. $x_i = VI_i$, $y_j = VI_{6+j}$, $k_j = K_j$, $i \in \overline{1,6}$, $j \in \overline{1,4}$.

2. *For* $q = 1; q \le b;\ q{+}{+}$ *do*

2.1. *For* $j = 0;\ j < 4;\ j{+}{+}$ *do*

2.1.1. $x_1 = \left(Sbox\left(x_1 + k_1\right) \oplus x_4\right) <<< k_4$; $x_2 = \left(Sbox\left(x_2 + k_2\right) + x_5\right) >>> k_3$;

$x_3 = Mix\left(\left(x_3 + x_6\right) \oplus y_3\right) <<< x_1$;

2.1.2. $k_1 = Sbox\left(\left(Sbox\left(x_1 \oplus k_1\right) + x_5\right) \oplus y_1\right)$; $k_2 = Sbox\left(Mix\left(x_2 + k_2 + x_6\right) \oplus y_2\right)$;

2.1.3. $y_1 = Sbox\left(\left(\left(k_1 + y_1\right) <<< x_2\right) \oplus k_3\right)$; $y_2 = Mix\left(Sbox\left(\left(\left(k_2 + y_2\right) >>> x_3\right) \oplus k_4\right)\right)$;

2.1.4. $x_4 = \left(Sbox\left(x_4 + k_3\right) \oplus x_1\right) <<< k_2$; $x_5 = \left(Sbox\left(x_5 + k_4\right) + x_2\right) >>> k_1$;

$x_6 = Mix\left(\left(x_6 + x_3\right) \oplus y_1\right) <<< x_4$;

2.1.5. $k_3 = Sbox\left(\left(Sbox\left(x_4 \oplus k_3\right) + x_2\right) \oplus y_3\right)$; $k_4 = Sbox\left(Mix\left(x_5 + k_4 + x_3\right) \oplus y_4\right)$;

2.1.6 $y_3 = Sbox\left(\left(\left(k_3 + y_3\right) <<< x_5\right) \oplus k_1\right)$; $y_4 = Mix\left(Sbox\left(\left(\left(k_4 + y_4\right) >>> x_6\right) \oplus k_2\right)\right)$.

2.2. $M_q = \left(y_1 \mid y_2 \mid y_3 \mid y_4\right)$

Рис. 1. Псевдокод алгоритму TriGen

Експериментальне дослідження методу генерування тритових псевдовипадкових послідовностей. *Мета експериментів* – дослідити ефективність розробленого методу у порівнянні з відомим та перевіритийого адекватність.

Для цього необіхно розв'язати такі задачі:

1. Дослідити запропонований та відомий метод генерування тритових ПВП за методикою NIST STS.

2. Дослідити запропонований та відомий метод генерування тритових ПВП за розробленим методом оцінювання якості тритових послідовностей.

Вхідними параметрами є: згенеровані послідовності (послідовності переводяться із трійкового у двійковий вид) генератором TritGen та стандартним генератором ПВП C++ (середовище Microsoft Visual Studio 2012) та згенеровані тритові послідовності генератором TritGen та стандартним генератором ПВП C++ (середовище Microsoft Visual Studio 2012).

Вихідні параметри: результати проходження за методикою NIST STS та результати проходження за розробленим методом оцінювання якості тритових послідовностей, який реалізований у вигляді консольного застосунку TritSTS.

Послідовність дій при проведені експериментів:

1. Згенерувати по 100 послідовностей генераторами (генератором TritGen та стандартним генератором ПВП C++), конвертувати трити у біти, після чого програмним комплексом NIST STS перевірити їх на псевдовипадковість.

2. Згенерувати по 100 послідовностей генераторами (генератором TritGen та стандартним генератором ПВП C++), після чого розробленим консольним застосунком TritSTS перевірити їх на псевдовипадковість.

Дослідження методів генерування ПВП за методикою NIST STS. Спочатку запропонований метод генерування тритових ПВП було вирішено перевірити за методикою NIST STS [2] для того щоб перевірити чи зможуть стандартні бітові тести адекватно оцінити псевдовипадковість тритових послідовностей.

Статистичні тести NIST STS використовуються для визначення якісних та кількісних ознак випадковості послідовності чисел [3]. Для прийняття висновків щодо проходження послідовностями випадкових чисел статистичних тестів використовується три основні критерії [2, 3]:

1. Критерій прийняття рішення на основі установлення деякого граничного рівня.

2. Критерій на основі установлення фіксованого довірчого інтервалу.

3. Критерій визначення для статистики тесту деякого відповідного значення ймовірності P-value (ймовірності випадку, що статистика тесту прийме значення більше за значення, що спостерігається

при випробуванні послідовності, в передбаченні її випадковості).

В основі статистичного тесту лежить перевірка деякої нульової гіпотези H_0 такої, що досліджувана послідовність – випадкова [2, 3, 15]. Також передбачена альтернативна гіпотеза H_1 – досліджувана послідовність не випадкова. Отже, згенерована послідовність досліджується набором тестів, у кожному з яких робиться висновок щодо відхилення або прийняття нульової гіпотези H_0. Для кожного тесту обирається адекватна статистика випадковості, на підставі котрої далі відхиляється або приймається гіпотеза H_0. Така статистика, відповідно припущенню на випадковість, володіє деяким розподілом випадкових значень. Теоретично розподіл статистики для нульової гіпотези вираховується із застосуванням математичних методів. Далі із такого зразкового розподілу визначається критичне значення. При проведенні тесту вираховується значення тестової статистики, яке порівнюється із критичним значенням. При перевищенні тестового критичного значення над еталонним, приймається гіпотеза H_1, в іншому випадку – гіпотеза H_0.

Для проведення експериментів були обрані такі вхідні параметри для застосування статистичних тестів NIST STS (згідно [2, 3, 15]):

1. Довжина послідовності, що тестується n=106 біт.

2. Кількість послідовностей, що тестується m=100.

3. Рівень значущості α=0,01.

4. Кількість тестів q=188, серед яких такі тести: Frequency − 1, Block Frequency − 1, Cumulative Sums − 2, Runs − 1, Longest Run − 1, Rank − 1, FFT − 1, Non Overlapping Template − 148, Overlapping Template − 1, Universal − 1, Approximate Entropy − 1, Random Excursions − 8, Random Excursions Variant − 18, Serial − 2, Linear Complexity − 1.

Таким чином, обсяг вибірки, що тестується, склав $N = 10^6 \times 100 = 10^8$ біт, кількість тестів (q) для різних довжин q = 188, таким чином, статистичний портрет генератора містить 18800 значень ймовірності P.

В ідеальному випадку при m = 100 і α = 0,01 у ході тестування може бути відкинута тільки одна послідовність зі ста, тобто коефіцієнт проходження кожного тесту має складати 99%. Але це занадто жорстке правило, тому аналогічно роботі [15] було застосовано правило на основі довірчого інтервалу, нижня межа дорівнює 0,96015.

Для проведення експериментів був розроблений консольний програмний засіб TriGen на мові програмування C++ в середовищі Microsoft Visual Studio 2012, який дозволяє генерувати тритові послідовності запропонованим алгоритмом TriGen та стандартним генератором ПВП C++.

Експерименти було вирішено проводити наступним чином:

1. Кожним досліджуваним генератором (алгоритмом TriGen та стандартним генератором ПВП C++) виконувалось генерування трьох послідовностей довжиною 60000000 трит.

2. Відбувалось переведення тритової послідовності у двійкову систему числення. Для цього тритова послідовніть ділилася на блоки довжиною 24 трита, кожен блок окремо переводився у 40 бітну послідовність. В результаті чого отримувалась двійкова послідовність довжиною 108 біт.

3. Отримана двійкова послідовність перевірялася статистичними тестами NIST STS. В результаті отримувались статистичні портрети послідовностей.

Генерування тритових ПВП стандартним генератором C++ виконувалась наступний чином:

1. Обиралися довільні значення векторів k і a (для кожної послідовності окремо) та вказувалась потрібна довжина послідовності n.

2. За допомогою функції rand() відбувалось генерування кожного трита послідовності.

Псевдокод процедури генерування наведений на рис. 2.

C++ gen

Input: вектори k і a, $k \in V_{32}$,

$a \in V_{32}$ $(V_n = \{0,1\}^n)$.

Output: вихідна ПВП $M = (M_1, ..., M_n)$,

$M_q \in \{0,1,2\}^n$, $q \in \overline{1,n}$.

1. $x = k$, $y = a$;

2. $std : srand(std :: time(0))$;

2. *For* $q = 1; q \le n;$ $q++$ *do*

2.1. *For* $j = 0;$ $j < 4;$ $j++$ *do*

2.1.1. $b = (\text{rand() +x}) \oplus y$;

2.1.2. $c = (\text{rand() +y}) \oplus x$;

2.1.3. $y = (\text{rand() +b}) \oplus c$

2.1.4. $x = (\text{rand() +y}) \oplus b$

2.2. $M[q] = x\%3$;

Рис. 2. Псевдокод процедури генерування тритових ПВП стандартним генератором C++

Генерування тритових ПВП запропонованим алгоритмом TriGen виконувалась згідно опису, що наведений у [12]. Для кожної послідовності окремо задавались початкові значення 24 тритних векторів k_i, $i \in \overline{1,4}$ після чого виконувалось генерування послідовностей.

Спочатку опишемо результати дослідження стандартного генератора ПВП C++. Для наглядності продемонструємо декілька результатів.

Послідовність 1 (C++ gen). Для генерування було обрані наступні початкові параметри: $k = 314342312$, $a = 403242341$. На рис. 3 зображено статистичний портрет проходження тестів NIST STS.

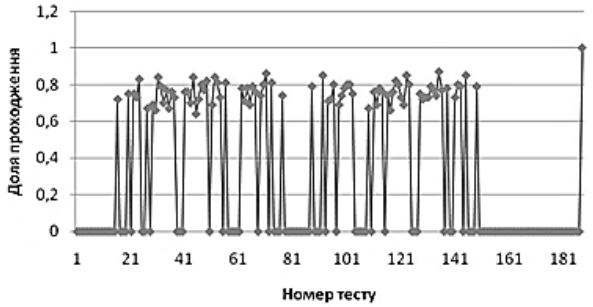

Рис. 3. Статистичний портрет C++ gen послідовності 1, згідно NIST STS

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 1, пройшло 96% − 1. Такі результати свідчать, що згенерована послідовність, що була переведела у двійковий вид тестування не пройшла (приймається гіпотеза H_1).

Послідовність 2 (C++ gen). Для генерування було обрані наступні початкові параметри: $k = 3834425654$, $a = 234525320$. На рис. 4 зображено статистичний портрет проходження тестів NIST STS.

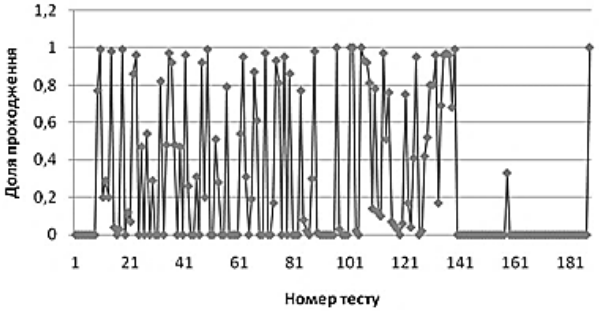

Рис. 4. Статистичний портрет C++ gen послідовності 2, згідно NIST STS

Кількість тестів, що пройшло 99% досліджуваних послідовностей − 9, пройшло 96% − 20. Аналогічно попереднім результатам бачимо, що згенерована послідовність також не пройшла тестування (приймається гіпотеза H_1).

Послідовність 3 (C++ gen). Для генерування було обрані наступні початкові параметри: $k = 2577261391$, $a = 980904215$. На рис. 5 зображено статистичний портрет проходження тестів NIST STS.

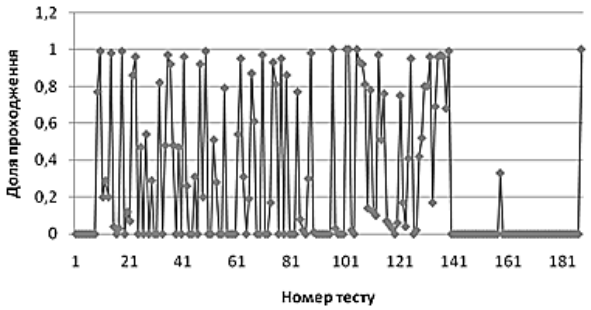

Рис. 5. Статистичний портрет C++ gen послідовності 3, згідно NIST STS

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 6, пройшло 96% − 33. Такі результати свідчать, що згенерована послідовність також не пройшла тестування (приймається гіпотеза H_1).

Тепер опишемо результати дослідження алгоритму TriGen.

Послідовність 1 (TriGen). Для генерування було обрані наступні початкові параметри:

$k_1 = 000000\ 000000\ 000000\ 000000$,

$k_2 = 000000\ 000000\ 000000\ 000001$,

$k_3 = 000000\ 000000\ 000200\ 000000$,

$k_4 = 000000\ 100000\ 000000\ 000000$.

На рис. 6 зображено статистичний портрет проходження тестів NIST STS.

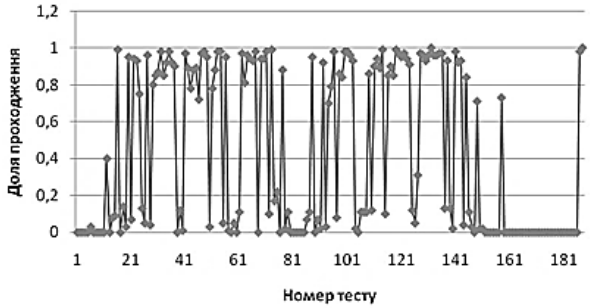

Рис. 6. Статистичний портрет TriGen послідовності 1, згідно NIST STS

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 32, пройшло 96% − 41. Такі результати свідчать, що згенерована послідовність алгоритмом TriGen також не пройшла тестування (приймається гіпотеза H_1).

Послідовність 2 (TriGen). Для генерування було обрані наступні початкові параметри:

$k_1 = 000001\ 000000\ 000001\ 011000$,

$k_2 = 120000\ 000000\ 000210\ 201001$,

$k_3 = 200100\ 200100\ 010200\ 100001$,

$k_4 = 201010\ 111111\ 211000\ 002222$.

На рис. 7 зображено статистичний портрет проходження тестів NIST STS.

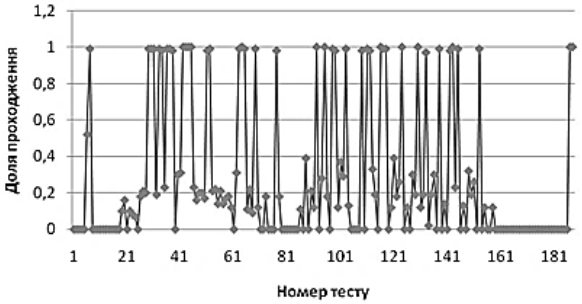

Рис. 7. Статистичний портрет TriGen послідовності 2, згідно NIST STS

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей – 24, пройшло 96% – 37. Такі результати свідчать, що згенерована послідовність алгоритмом TriGen також не пройшла тестування (приймається гіпотеза H_1).

Послідовність 3 (TriGen). Для генерування було обрані наступні початкові параметри:

$k_1 = 102210\ 210120\ 001022\ 010110$,

$k_2 = 201001\ 020210\ 121210\ 121011$,

$k_3 = 102021\ 022011\ 120101\ 122102$,

$k_4 = 120220\ 110200\ 102112\ 011012$.

На рис. 8 зображено статистичний портрет проходження тестів NIST STS.

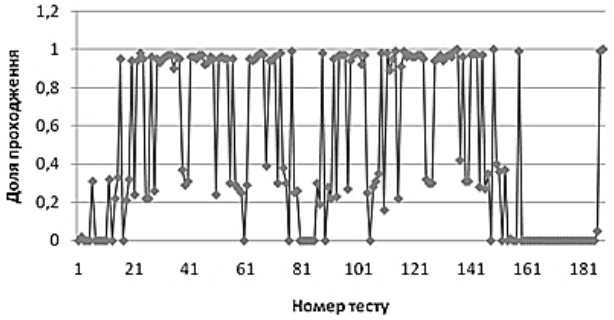

Рис. 8. Статистичний портрет TriGen послідовності 3, згідно NIST STS

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей – 9, пройшло 96% – 51. Такі результати свідчать, що згенерована послідовність алгоритмом TriGen також не пройшла тестування (приймається гіпотеза H_1).

Отже можна зробити висновок, що стандарті бітові тести не можуть корректно оцінити псевдовипадковість тритових послідовностей. Це можна пояснити переведення тритової послідовності у двійкову систему числення. При такому переведенні тритова блок (в даному випадку 24 трита) не може покрити повністю утворений бітовий блок (в даному випадку 40 біта). Оскільки $3^{24} = 282429536481$, а $2^{40} = 1099511627776$ то 817082091295 значення бітового блоку ніколи не появляться у ньому, тому послідовність є передбачуваною.

Дослідження методів генерування ПВП запропонованим методом оцінювання їх якості

У роботі [13] детально описаний метод оцінювання якості ПВП. Опишемо вхідні параметри для його застосування (параметри обрано аналогічно тестам NIST STS):

1. Довжина послідовності тритів n = 150000 трит.

2. Кількість послідовностей, що тестується m = 100.

3. Рівень значущості $\alpha = 0{,}01$.

4. Кількість тестів q = 152, серед яких такі тести: Freqeuncy monotrit test – 1, Frequency Block Trit Test – 1, Trit Runs Test – 1, Test for the Longest Run in a Block – 1, Trit Non-Overlapping Template Matching Test – 148, Trit Overlapping Template Matching Test – 1.

Таким чином, обсяг вибірки, що тестується, склав $N = 1{,}5 \cdot 10^7$ трит, кількість тестів (q) для різних довжин q = 152, таким чином, статистичний портрет генератора містить 152 значень ймовірності P. Як і для тестів NIST STS будемо застосовувати правило довірчих інтервалів, тобто коефіцієнт проходження кожного тесту має складати 0,96015%.

На основі запропонованого методу оцінки якості ПВП та наведених вище вхідних параметрах було розроблене консольне програмне забезпечення TritSTS (на мові програмування C++ в середовищі Microsoft Visual Studio 2012), що дозволяє оцінювати тритові послідованості на псевдовипадковість. Даний засіб використовувався при проведенні експериментів.

Експерименти було вирішено проводити наступним чином:

1. Кожним досліджуваним генератором (алгоритмом Trigen та стандартним генератором ПВП C++) виконувалось генерування п'ятьох послідовностей довжиною $N = 1{,}5 \cdot 10^7$ трит.

2. Отримана тритова послідовність перевірялася програмою TritSTS. В результаті отримувались статистичні портрети послідовностей.

Генерування тритових послідовностей стандартним генератором C++ та запропонованим алгоритом Trigen розписана вище.

Перевірку на псевдовипадковість програмою TritSTS почнемо із результатів дослідження алгоритму TriGen.

Послідовність 1 (TriGen). Для генерування було обрані наступні початкові параметри (які вже використовувались при перевірці тестами NIST STS):

$$k_1 = 102210\ 210120\ 001022\ 010110,$$
$$k_2 = 201001\ 020210\ 121210\ 121011,$$
$$k_3 = 102021\ 022011\ 120101\ 122102,$$
$$k_4 = 120220\ 110200\ 102112\ 011012.$$

На рис. 9 зображено статистичний портрет проходження тестів TritSTS.

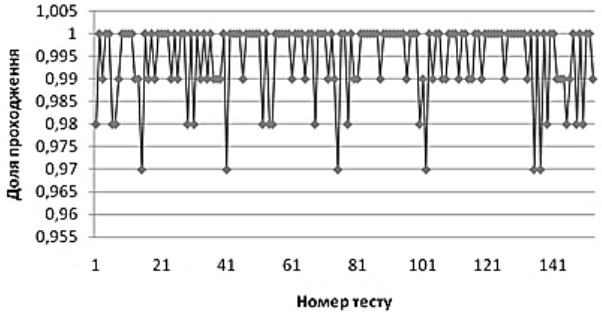

Рис. 9. Статистичний портрет TriGen послідовності 1, згідно TritSTS

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 132, пройшло 96% − 153. Кількість $P_{value_{01}} \geq 0,01$ − 153, $P_{value_{02}} \geq 0,01$ − 153, $P_{value_{12}} \geq 0,01$ − 153. Такі результати свідчать, що згенерована послідовність пройшла тестування.

Послідовність 2 (TriGen). Для генерування було обрані наступні початкові параметри (які вже використовувались при перевірці тестами NIST STS):

$$k_1 = 000000\ 000000\ 000000\ 000000,$$
$$k_2 = 000000\ 000000\ 000000\ 000001,$$
$$k_3 = 000000\ 000000\ 000200\ 000000,$$
$$k_4 = 000000\ 100000\ 000000\ 000000.$$

На рис. 10 зображено статистичний портрет проходження тестів TritSTS.

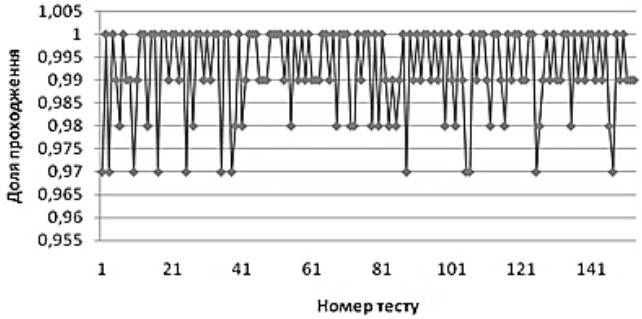

Рис. 10. Статистичний портрет TriGen послідовності 2, згідно TritSTS

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 121, пройшло 96% − 153. Кількість $P_{value_{01}} \geq 0,01$ − 152, $P_{value_{02}} \geq 0,01$ − 151, $P_{value_{12}} \geq 0,01$ − 153. Такі результати свідчать, що згенерована послідовність пройшла тестування (не зважаючи на $P_{value_{XY}} < 0,01$). Отже, приймається гіпотеза H_0.

Послідовність 3 (TriGen). Для генерування було обрані наступні початкові параметри (які вже використовувались при перевірці тестами NIST STS):

$$k_1 = 000000\ 000000\ 000000\ 000000,$$
$$k_2 = 000000\ 000000\ 000000\ 000001,$$
$$k_3 = 000000\ 000000\ 000200\ 000000,$$
$$k_4 = 000000\ 100000\ 000000\ 000000.$$

На рис. 11 зображено статистичний портрет проходження тестів TritSTS.

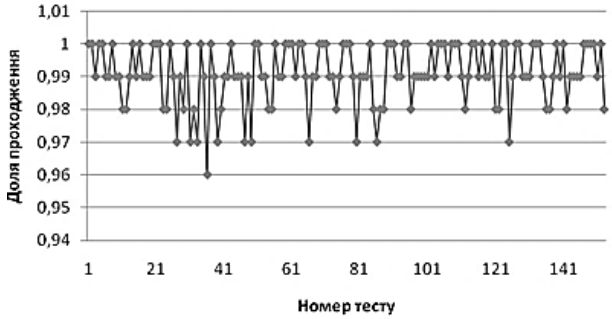

Рис. 11. Статистичний портрет TriGen послідовності 3, згідно TritSTS

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 121, пройшло 96% − 152. Кількість $P_{value_{01}} \geq 0,01$ − 151, $P_{value_{02}} \geq 0,01$ − 153, $P_{value_{12}} \geq 0,01$ − 152. Такі результати свідчать, що згенерована послідовність не погана, проте вона не пройшла тестування.

Послідовність 4 (TriGen). Для генерування було обрані наступні початкові параметри:

$$k_1 = 100020\ 102000\ 100022\ 011102,$$
$$k_2 = 221100\ 010101\ 020202\ 212121,$$
$$k_3 = 111002\ 020100\ 210200\ 110202,$$
$$k_4 = 120022\ 102021\ 022001\ 100202.$$

На рис. 12 зображено статистичний портрет проходження тестів TritSTS.

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 114, пройшло 96% − 153. Кількість $P_{value_{01}} \geq 0,01$ − 152, $P_{value_{02}} \geq 0,01$ − 152, $P_{value_{12}} \geq 0,01$ − 151. Такі результати свідчать, що згенерована послідовність

пройшла тестування (не зважаючи на $P_{value_{XY}} < 0,01$).

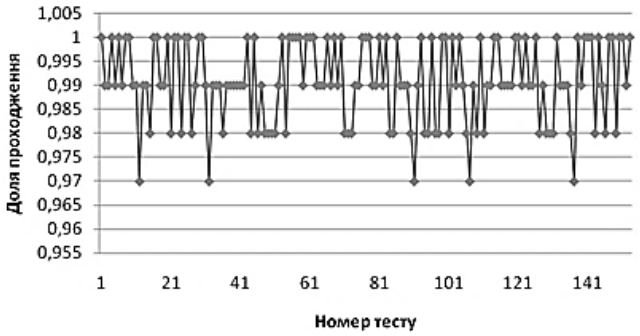

Рис. 12. Статистичний портрет TriGen послідовності 4, згідно TritSTS

Послідовність 5 (TriGen). Для генерування було обрані наступні початкові параметри:

$k_1 = 221020\,102001\,100022\,010112$,

$k_2 = 021222\,110201\,120001\,202021$,

$k_3 = 121110\,022001\,112202\,210201$,

$k_4 = 102212\,001021\,122011\,201202$.

На рис. 13 зображено статистичний портрет проходження тестів TritSTS.

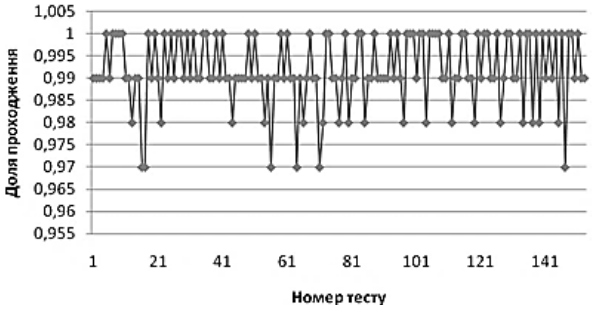

Рис. 13. Статистичний портрет TriGen послідовності 5, згідно TritSTS

Як видно із результатів кількість тестів, що пройшло 99% досліджуваних послідовностей − 129, пройшло 96% − 153. Кількість $P_{value_{01}} \geq 0,01$ − 153, $P_{value_{02}} \geq 0,01$ − 153, $P_{value_{12}} \geq 0,01$ − 153. Такі результати свідчать, що згенерована послідовність пройшла тестування.

Тепер опишемо результати дослідження стандартного генератора ПВП C++ програмою TritSTS.

Послідовність 1 (C++ gen). Для генерування було обрані наступні початкові параметри: $k = 314342312$, $a = 403242341$ (дані параметри вже використовувались при перевірці тестами NIST STS). На рис. 14 зображено статистичний портрет проходження тестів TritSTS.

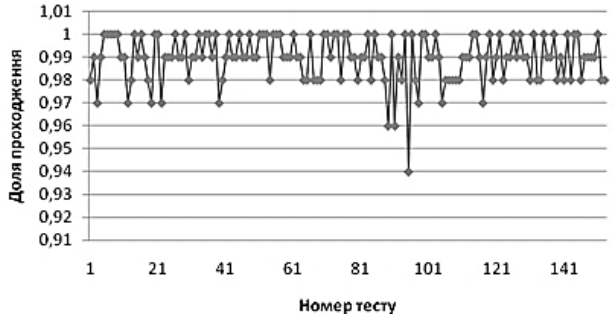

Рис. 14. Статистичний портрет C++ gen послідовності 1, згідно TritSTS

Кількість тестів, що пройшло 99% досліджуваних послідовностей − 109, пройшло 96% − 150. Кількість $P_{value_{01}} \geq 0,01$ − 143, $P_{value_{02}} \geq 0,01$ − 146, $P_{value_{12}} \geq 0,01$ − 141. Такі результати свідчать, що згенерована послідовність не погана, проте вона не пройшла тестування.

Послідовність 2 (C++ gen). Для генерування було обрані наступні початкові параметри: $k = 3834425654$, $a = 234525320$ (дані параметри вже використовувались при перевірці тестами NIST STS). На рис. 15 зображено статистичний портрет проходження тестів TritSTS.

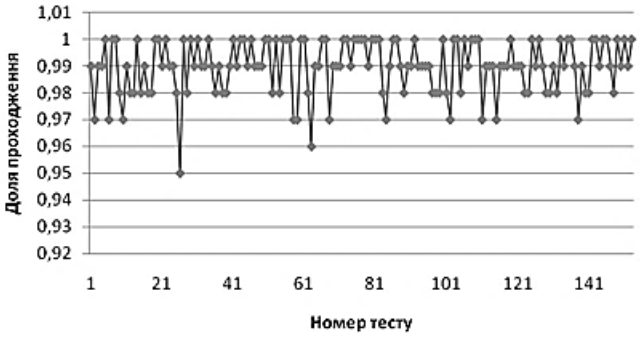

Рис. 15. Статистичний портрет C++ gen послідовності 2, згідно TritSTS

Кількість тестів, що пройшло 99% досліджуваних послідовностей − 111, пройшло 96% − 151. Кількість $P_{value_{01}} \geq 0,01$ − 145, $P_{value_{02}} \geq 0,01$ − 142, $P_{value_{12}} \geq 0,01$ − 143. Такі результати свідчать, що згенерована послідовність не погана, проте вона не пройшла тестування.

Послідовність 3 (C++ gen). Для генерування було обрані наступні початкові параметри: $k = 2577261391$, $a = 980904215$ (дані параметри вже використовувались при перевірці тестами NIST STS). На рис. 16 зображено статистичний портрет проходження тестів TritSTS.

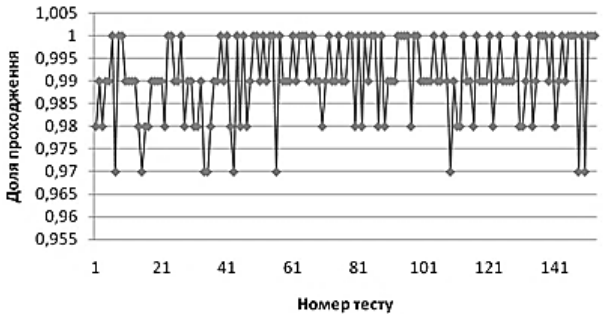

Рис. 16. Статистичний портрет C++ gen послідовності 3, згідно TritSTS

Кількість тестів, що пройшло 99% досліджуваних послідовностей − 117, пройшло 96% − 153. Кількість $P_{value_{01}} \geq 0,01$ − 153, $P_{value_{02}} \geq 0,01$ − 153, $P_{value_{12}} \geq 0,01$ − 153. Такі результати свідчать, що згенерована послідовність пройшла тестування.

Послідовність 4 (C++ gen). Для генерування було обрані наступні початкові параметри: $k = 5674312$, $a = 8476892$. На рис. 17 зображено статистичний портрет проходження тестів TritSTS.

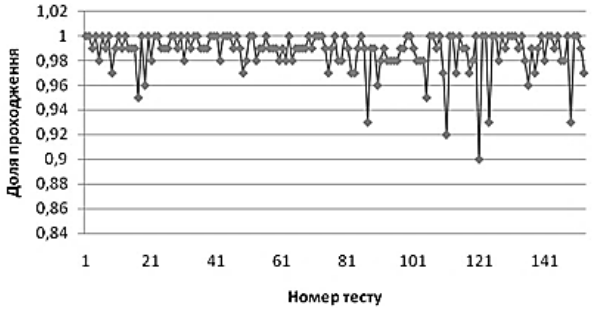

Рис. 17. Статистичний портрет C++ gen послідовності 4, згідно TritSTS

Послідовність 5 (C++ gen). Для генерування було обрані наступні початкові параметри: $k = 7890212$, $a = 34095422$. На рис. 18 зображено статистичний портрет проходження тестів TritSTS.

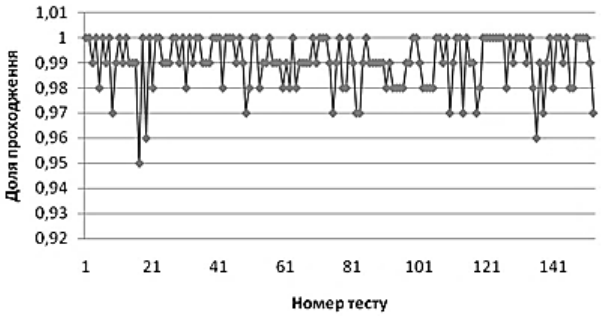

Рис. 18. Статистичний портрет C++ gen послідовності 5, згідно TritSTS

Кількість тестів, що пройшло 99% досліджуваних послідовностей − 114, пройшло 96% − 150. Кількість $P_{value_{01}} \geq 0,01$ − 139, $P_{value_{02}} \geq 0,01$ − 145, $P_{value_{12}} \geq 0,01$ − 141. Такі результати свідчать, що згенерована послідовність не погана, проте вона не пройшла тестування.

Зведені результати експериментів представлені у табл. 1.

Згідно отриманих результатів послідовності згенеровані запропонованим алгоритмом TriGen показали кращі результати ніж ПВП згенеровані стандартним генератором C++.

Зауважимо, що із п'яти представлених послідовностей згенерованим стандартним генератором C++ тільки одна пройшла тестування програмою TritSTS, а за допомогою генератора TriGen успішно пройшли вже чотири із п'яти послідовностей. Якщо брати середні результати, то послідовності згенеровані алгоритмом TriGen на 11,6% частіше успішно проходять 99% послідовностей і на 3,4% частіше успішно проходять 96% послідовностей.

Таблиця 1

Узагальнені результати проходження тестів TritSTS

Послідовність	$P_{value_{01}} \geq 0,01$	$P_{value_{02}} \geq 0,01$	$P_{value_{12}} \geq 0,01$	Кількість тестів у яких тестування пройшло	
				99%	96%
TriGen-1	153	153	153	132	153
TriGen-2	152	151	153	121	153
TriGen-3	151	153	152	121	152
TriGen-4	152	152	151	114	153
TriGen-5	153	153	153	129	153
C++ gen-1	143	146	141	109	150
C++ gen-2	145	142	143	111	151
C++ gen-3	153	153	153	117	153
C++ gen-4	133	131	128	108	143
C++ gen-5	139	145	141	114	150
TriGen сред.	152,2	152,4	152,4	123,4	152,8
C++ gen сред.	142,6	143,4	141,2	111,8	149,4

Отже проаналізувавши результати дослідження запропонованого та відомого методів генерування тритових ПВП за методикою NIST STS можна зробити висновок що дану методику не можливо використовувати для оцінки якості тритових послідовностей, а розроблений метод оцінки якості тритових послідовностей та запропонований метод генерування формує якісні ПВП придатні для використання на практиці.

Висновки. Розроблено метод генерування псевдовипадкових послідовностей, який, за рахунок виконання нової послідовності операцій (підстановок $Sbox(X)$, лінійного розсіювання $Mix(X)$, динамічного циклічного зсуву і додавання за модулем $3\oplus$ та 3^l+) над вектором внутрішніх станів V_p ($V_p=\{0,1,2\}^p$, $p=14\cdot l$) за $r\cdot b$ циклів, дозволяє формувати трійкові незбалансовані («0», «1», «2») псевдовипадкові послідовності $V_{m\cdot b}, m=4\cdot l$, що можуть використовуватись для реалізації запропонованого метода забезпечення стійкості кутритових протоколів квантової криптографії до некогерентних атак, а також для інших криптографічних застосувань в сучасних інформаційно-комунікаційних технологіях.

ЛІТЕРАТУРА

[1]. A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications. NIST Special Publication 800-22. – May 15, 2001. – 164 p.

[2]. NIST STS [Electronic resource] : Download documentation and software. – Electronic data. – NIST, 2014. – Mode of access: http://csrc.nist.gov/groups/ST/toolkit/rng/ documentation_software.html. – [Accessed Jul. 5, 2015].

[3]. On the Interpretation of Results from the NIST Statistical Test Suite / Marek SYS, Zdenek RIHA, Vashek MATYAS, Kinga MARTON, Alin SUCIU // Romanian jurnal of information science and technology. – Vol. 18. – № 1. – 2015. – P. 18-32.

[4]. Гарасимчук О.І. Генератори псевдовипадкових чисел, їх застосування, класифікація, основні методи побудови і оцінка якості / Гарасимчук О.І., Максимович В.М. // Захист інформації. – 2003. – №3. – С. 29-36.

[5]. Горбенко І.Д. Обґрунтування вимог до генераторів випадкових бітів згідно ISO/IEC 18031 / Горбенко І.Д., Шапочка Н.В., Козулін О.О. // Радіоелектронні і комп'ютерні системи. – 2009. – № 6 (40). – С. 94-97.

[6]. Експериментальне дослідження методу забезпечення стійкості кутритових протоколів квантової криптографії / С.О. Гнатюк, Т.О. Жмурко, В.М. Кінзерявий, Х.І. Юбузова // Захист інформації. – 2016. – № 3. – Т. 18. – С. 218-228.

[7]. Євсеєв С.П. Аналіз сучасних методів формування псевдовипадкових послідовностей / С.П. Євсеєв, Р.В. Корольов, М.В. Краснянська // Восточно-Европейский журнал передовых технологий. – 2010. – 3/4 (45). – С. 11-15.

[8]. Иванов М.А. Теория, применение и оценка качества генераторов псевдослучайных последовательностей / М.А. Иванов, И.В. Чугунков. – М.: КУДИЦ-ОБРАЗ, 2003. – 240 с.

[9]. Калугин А.Н. Модификация многомерных псевдослучайных последовательностей с использованием двойственных LFSR-CNS генераторов / А. Н. Калугин // Компьютерная оптика. – 2005. – № 28. – С. 112-118.

[10]. Корольов Р.В. Дослідження періодичних властивостей генераторів псевдовипадкових чисел, заснованих на використанні надмірних блокових кодів / Р.В. Корольов // Системи озброєння і військова техніка. – 2008. – № 3(15). – С.126-128.

[11]. Кренгель Е. И. Исследование и разработка новых классов псевдослучайных последовательностей и устройств их генерации для систем с кодовым разделением каналов: дис. канд. техн. наук: 05.12.13 / Кренгель Евгений Ильич. – М., 2002. – 214 с.

[12]. Метод генерування тритових псевдовипадкових послідовностей для систем квантової криптографії / С.О. Гнатюк, Т.О. Жмурко, В.М. Кінзерявий, Н.А. Сейлова // Безпека інформації. – 2015. – № 2. – Т. 22. – С. 140-147.

[13]. Метод оцінювання якості тритових псевдовипадкових послідовностей для криптографічних застосувань / С.О. Гнатюк, Т.О. Жмурко, В.М. Кінзерявий, Н.А. Сейлова // Information Technology and Security. – 2015. – Vol. 3. – № 2(5). – С. 108-116.

[14]. Назаров Є.О. Генератори псведовипадкових послідовностей для криптографічних систем / Назаров Є.О., Чернишова А.В., Губенко Н.Є.// збірник наукових праць міжнародної науково-технічної конференції «Інформатика і компютерні технології - 2012». – ДонНТУ: 2012. – С. 139-144.

[15]. Потий А.В. Статистическое тестирование генераторов случайных и псевдослучайных чисел с использованием набора статистических тестов NISTSTS / А.В. Потий, С.Ю. Орлова, Т.А. Гриненко // Правове, нормативне та метрологічне забезпечення систем захисту інформації в Україні. – 2001. – Вип. 2. – С. 206-214.

[16]. Рисований О.М. Генератор псевдовипадкових послідовностей по модулю 3 з різною частотою формування псевдовипадкових послідовностей / О.М. Рисований, В.В. Гоготов // Системи обробки інформації – 2010 р. – № 2 (83). – С. 141-143.

REFERENCES

[1]. A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number

Generators for Cryptographic Applications. NIST Special Publication 800-22, May 15, 2001, 164 p.

[2]. NIST STS [Electronic resource] : Download documentation and software. – Electronic data. – NIST, 2014. – Mode of access: http://csrc.nist.gov/ groups/ ST/ toolkit/ rng/ documentation_software.html. – [Accessed Jul. 5, 2015].

[3]. On the Interpretation of Results from the NIST Statistical Test Suite / Marek SYS, Zdenek RIHA, Vashek MATYAS, Kinga MARTON, Alin SUCIU // Romanian jurnal of information science and technology, Vol. 18, № 1, 2015, P. 18-32.

[4]. Harasymchuk O., Maksimovic V. Pseudorandom number generator, their use, classification, basic methods of construction and quality assessment, *Ukrainian information security research journal*, 2003, №3, P. 29-36.

[5]. Horbenko I.D., Shapochka N.V., Kozulin O.O. Justification of requirements for generators of the random bits according to ISO/IEC 18031, *Radio electronic and computer systems*, 2009, № 6 (40), P. 94-97.

[6]. Gnatyuk S., Zhmurko T., Kinzeryavyy V., Yubuzova H. Experimental research of efficiency increasing method for qutrits quantum cryptography protocols, *Ukrainian information security research journal*, 2016, № 3, Iss. 18, P. 218-228.

[7]. Yevseyev S.P., Korol′ov R.V., Krasnyans′ka M.V. Analysis of current methods of generating pseudorandom sequences, Eastern European Journal of Advanced Technology., 2010, 3/4 (45), P. 11-15.Ivanov M.A., Chugunkov I.V. Theory, application and quality assessment of pseudo-random sequence generators, Moscow: Kudits-Obraz, 2003, 240 p.

[8]. Kalugin A.N. Modification of multidimensional pseudo-random sequences using dual LFSR-CNS generators, *Computer optics*, 2005, No. 28. P. 112-118.

[9]. Korol′ov R.V. Research of periodic properties of the pseudorandom number generator based on excessive use of block codes, *Weapons systems and military equipment*., 2008, № 3 (15), P. 126-128.

[10]. Krengel' Ye. Research and development of new classes of pseudo-random sequences and devices for their generation for systems with code division channels: dis. Cand. Tech. Sciences: 05.12.13, M., 2002, 214 p.

[11]. Gnatyuk S., Zhmurko T., Kinzeryavyy V., Seilova N. Method of trit pseudorandom sequences generating for quantum cryptography systems // Ukrainian Scientific Journal of Information Security, 2015, vol. 22, issue 2, P. 140-147.

[12]. Gnatyuk S., Zhmurko T., Kinzeryavyy V., Seilova N. The method of evaluating the quality trit pseudorandom sequences for cryptographic applications // Information Technology and Security, 2015, Vol. 3, № 2(5), P. 108-116.

[13]. Nazarov Y., Chernyshova A., Hubenko N. Generators of pseudorandom sequences for cryptographic systems, *Scientific papers of international scientific conference "Information and Computer Technology 2012" DonNTU*: 2012, P. 139-144.

[14]. Potiy A., Orlova S., Grinenko T. Statistical testing of generators of random and pseudo-random numbers using a set of statistical tests NIST STS, *Legal, regulatory and metrological support of information security in Ukraine*., 2001, Vol. 2, P. 206-214.

[15]. Rysovanyy O.M., Hohotov V. Generator pseudorandom sequence modulo 3 different frequencies of pseudorandom sequences // Information processing systems, 2010, № 2 (83). P. 141-143.

ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ МЕТОДА ГЕНЕРИРОВАНИЯ ТРИТОВЫХ ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ ДЛЯ КРИПТОГРАФИЧЕСКИХ ПРИЛОЖЕНИЙ

В последние несколько лет развитие квантовой криптографии привлекло внимание ученых всего мира и перешло от уровня теоретических исследований к внедрению готовых коммерческих решений. Основное отличие методов квантовой криптографии от традиционной(симметричной и асимметричной) криптографии заключается в использовании в своей основе совсем разных принципов. Квантовая криптография не зависит от вычислительных или других возможностей нарушителя, использует специфические уникальные свойства квантовых частиц и основывается на нерушимости законов квантовой физики. Среди самых развитых технологий квантовой криптографии стоит отметить квантовую прямую безопасную связь, которая позволяет передавать информацию открытым каналом без предыдущей ее шифровки, - нивелируя проблему распределения ключей, в этих протоколах каждый бит есть конфиденциальной информацией и не должен попасть к нарушителю, по этому требования к стойкости протоколов есть более высокими и требует методов усиления безопасности. Авторами был разработанный метод обеспечения стойкости протоколов квантовой криптографии, однако для его реализации необходимо использование надежных тритовых псевдослучайных последовательностей. В связи с этим в работе представлено экспериментальное исследование разработанного метода генерирования тритовых последовательностей, для оценивания эффективности этого метода была разработана методика проведения эксперимента, которая подтвердила возможность использования сгенерированных тритовых последовательностей для реализации предложенного метода обеспечения стойкости кутритовых протоколов квантовой криптографии к некогерентным атакам, а также для других криптографических приложений в современных информационно-коммуникационных технологиях.

Ключевые слова: квантовая криптография, генератор псевдослучайных последовательностей, квантовая прямая безопасная связь, трит, защита информации.

EXPERIMENTAL RESEARCH OF TRIT PSEUDORANDOM SEQUENCE GENERATING METHOD FOR CRYPTOGRAPHIC APPLICATIONS

In the last few years, development of quantum cryptography has attracted the scientist's attention. Researches passed from the theoretical level to introduction of ready commercial decisions. The main difference between the methods of quantum cryptography and traditional (symmetric and asymmetric) cryptography consists in use of absolutely different principles in the basis. The quantum cryptography does not depend on computing or other intruder opportunities, uses specific unique quantum particles properties and based on inviolability of quantum physics laws. Among the most developed technologies of quantum cryptography is worth noting quantum secure direct communication, which can transmit information by open channel without encryption - removing key distribution problem, but in these protocols each bit of information is confidential and intruder shouldn't get it, that's why requirements for protocols stability is much higher and require security amplification methods. Such method was previously developed by authors, but its implementation requires the use of reliable trit pseudorandom sequences. In this context, this study presents an experimental research of trit pseudorandom sequence generating method for cryptographic applications. For evaluating the effectiveness of this method was developed experimental methodology, which confirms the possibility of using trit sequences generated by the method for ensuring the stability of quantum cryptography qutrit protocols to non-coherent attacks and for other cryptographic applications in modern information and communication technologies.

Keywords: quantum cryptography, pseudorandom sequences generator, quantum secure direct communication, trit, information security.

Шелест Михайло Євгенович, доктор технічних наук, професор, завідувач кафедри комп'ютерної техніки та програмування, Міжнародний технологічний університет "Миколаївська політехніка".
E-mail: mishel3141@gmail.com

Шелест Михаил Евгеньевич, доктор кандидат технических наук, професор, заведующий кафедрой компютерной техники и програмирования, Международный технологический университет "Николаевская политехника".

Shelest Mykhaylo, Dr.Sc., Professor, Head of Academic Department of computer engineering and programming, International Technological University "Mykolayivs′ka politekhnika".

Гнатюк Сергій Олександрович, кандидат технічних наук, доцент, доцент кафедри безпеки іфнормаційних технологій Національного авіаційного університету.
E-mail: s.gnatyuk@nau.edu.ua.

Гнатюк Сергей Александрович, кандидат технических наук, доцент, доцент кафедры безопасности информационных технологий Национального авиационного университета.

Gnatyuk Sergiy, PhD in Eng, Associate Professor of IT-Security Academic Department, National Aviation University.

Жмурко Тетяна Олександрівна, кандидат технічних наук, доцент кафедри безпеки інформаційних технологій Національного авіаційного університету.
E-mail: t.zhmurko@nau.edu.ua.

Жмурко Татьяна Александровна, кандидат технических наук, доцент кафедры безопасности информационных технологий Национального авиационного университета.

Zhmurko Tetiana, PhD in Eng., Associate Professor of IT-Security Academic Department, National Aviation University.

Кінзерявий Василь Миколайович, кандидат технічних наук, доцент кафедри безпеки іфнормаційних технологій Національного авіаційного університету.
E-mail: v.kinzeryavyy@gmail.com.

Кинзерявый Василий Николаевич, кандидат технических наук, доцент кафедры безопасности информационных технологий, Национальный авиационный университет.

Kinzeryavyy Vasyl, PhD in Eng., Associate Professor of IT-Security Academic Department, National Aviation University.

Юбузова Халіча Ібрагімівна, старший викладач кафедри інформаційної безпеки Казахського національного дослідницького технічного університету ім. К.І. Сатпаева (Алмати, Казахстан).
E-mail: hali4a@mail.ru.

Юбузова Халича Ибрагимовна, старший преподаватель кафедры информационная безопасность Казахского национального исследовательского технического университета им. К.И. Сатпаева (Алматы, Казахстан).

Yubuzova Khalicha, Senior Lecture of Information Security Academic Department, Kazakh National Research Technical University named after K.I. Satpayev (Almaty, Republic of Kazakhstan).