УДК 004.056.5 **DOI:10.30837/rt.2020.1.200.06**

О.О. КУЗНЕЦОВ, д-р техн. наук, А.С. КІЯН, А.І. ПУШКАРЬОВ, Т.Ю. КУЗНЕЦОВА

ТЕСТУВАННЯ КОДОВИХ ГЕНЕРАТОРІВ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ ДЛЯ ПОСТКВАНТОВОГО ЗАСТОСУВАННЯ

Вступ

Використання випадкових послідовностей є невід'ємною частиною у сфері інформаційної безпеки та, зокрема, у криптографії [1, 2]: формуванні ключів та паролів користувачів, генерації гамуючих послідовностей при потоковому шифруванні, формуванні векторів ініціалізації для блокових шифрів та інше. З метою отримання подібних послідовностей використовують різноманітні способи генерації, серед яких особливе місце посідають криптографічно стійкі генератори псевдовипадкових послідовностей, що поділяються на блокові, потокові та ті, що базуються на використанні односторонніх функцій. Останні є перспективним напрямком досліджень, з тієї причини, що є доказово стійкими, базуючись на добре вивченій теоретико-складністній задачі [2]. Однак суттєвою проблемою більшості з генераторів псевдовипадкових послідовностей, які використовують односторонню функцію, є поява повномасштабних квантових комп'ютерів, який працює на основі явищ квантової заплутаності та квантової суперпозиції. Подібні пристрої здатні пришвидшити виконання обчислювальних операцій в тисячі разів, порівняно з класичним комп'ютером.

Активні дослідження в області розробки квантових комп'ютерів призвели до реакції зі сторони криптографічної спільноти, що проявляється у вигляді конкурсу постквантової стандартизації, оголошеного Національним інститутом стандартів і технологій у 2017 р., та результатом якого стане стандартизація проектів у сфері шифрування, електронного цифрового підпису та механізмів інкапсуляції ключів [3]. Останній факт впливає на безпечність існуючих криптопримітивів, фактично роблячи їх нестійкими. Звідси набуває актуальності дослідження нових методів побудови генераторів псевдовипадкових послідовностей, які будуть здатні зберігати стійкість в умовах класичного та квантового криптоаналізу, тобто постквантових методів генерації. Перспективним напрямком у цьому контексті є генератори, що базуються на використанні кодів, з декількох причин. По-перше, такі генератори відносяться до доказово стійких, оскільки засновані на відомій проблемі синдромного декодування, що вважається NP-складною. По-друге, є доведеним той факт, що кодові генератори є постквантовими, оскільки, на даний момент, невідомо ефективний алгоритм для вирішення проблеми синдромного декодування ні з застосуванням класичних, ні з застосуванням квантових обчислень [4].

У статті висвітлено принципи побудови класичного кодового генератора псевдовипадкових послідовностей та запропоновано новий підхід до реалізації кодового генератора, що дозволяє подолати недолік у класичній схемі такий, як зменшена практична довжина періоду формуємої послідовності. Надалі проведено евристичне тестування таких генераторів: швидкості генерації послідовності, довжини періоду послідовності та стійкості до класичного та квантового криптоаналізу. Дослідження стійкості здійснено у порівнянні зі стійкістю генераторів, що базуються на складності вирішення поширених теоретико-складнісних задач [5].

Доказово стійкий генератор Фішера – Штерна

Серед доказово стійких генераторів, стійкість яких заснована на проблемі синдромного декодування класичним вважається генератор, запропонований Фішером та Штерном. Саме його структуру покладено в основу розвитку подальших схем. Розглянемо принципи побудови подібного генератора. У його основі лежить використання блокового (n,k,d) коду, який задано перевірочною матрицею H, розміром $n\times k$. Функціонування генератора можна представити за допомогою наступної схеми (рис.1)[6].

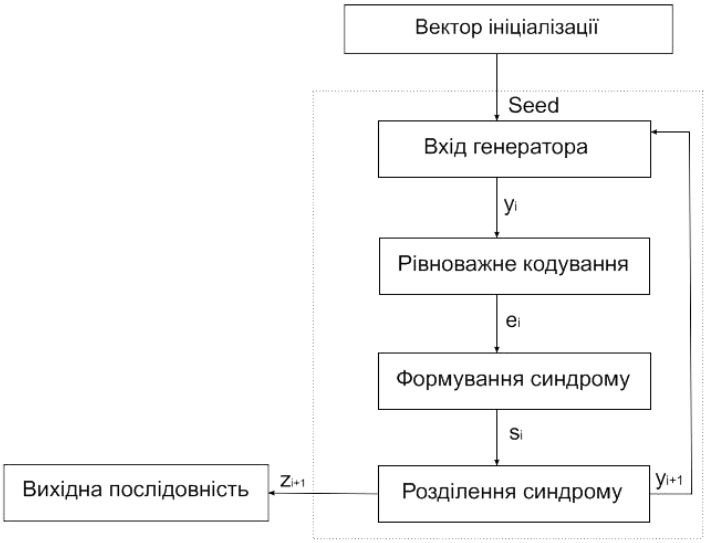

Рис. 1. Структурна схема генератора Фішера – Штерна

Етап 1. Обрати лінійний блоковий (n,k,d) код та необхідну довжину формуємої послідовності-l. На вхід генератора подається вектор ініціалізації($Seed$) довжини $m=\left\lfloor \log_2\binom{n}{t} \right\rfloor$. У цьому випадку $\lfloor x \rfloor$ позначає найбільше ціле число, що не перевищує x .

Етап 2. Вектор ініціалізації $Seed$ перетворюється за допомогою методу рівноважного кодування на вектор e_0, вага Хеммінга якого дорівнює виправляючій здатності коду($w(e_0)=t$). Величина t обчислюється згідно з параметрами обраного коду за правилом $t=\left\lceil \frac{d-1}{2} \right\rceil$.

Етап 3. До зміненого завдяки рівноважному кодуванню вектору e_0 застосовується рекурентне правило:

$$s_i = e_i \cdot H^T$$

де e_i _ двійковий вектор довжини n та $w(e_i)=t$; s_i _ двійковий вектор довжини $n-k$; H _ двійкова перевірочна матриця (n,k,d) коду .

Етап 4. Сформований на попередньому етапі вектор s_i розділяється на дві частини:

$$s_i = y_{i+1} \| z_{i+1}$$

де довжина вектору y_{i+1} дорівнює m біт, а довжина вектору z_{i+1} є $n-k-m$ біт.

Етап 5. Вектор z_{i+1} направляється на вихід генератора у якості частини сформованої послідовності.

Етап 6. Вектор y_{i+1} подається на вхід генератора як вектор ініціалізації для здійснення наступного циклу формування послідовності. Етапи 2-5 повторюються, поки не буде сформована послідовність необхідної довжини. Кількість циклів, що необхідно здійснити для формування послідовності дорівнює:

$$r=\left\lceil \frac{l}{n-k-m} \right\rceil,$$

де $\lceil x \rceil$ є найменшим цілим числом, що перевищує x .

В роботі [11] показано, що розглянутий генератор Фішера – Штерна володіє певними недоліками. Зокрема, проведені дослідження періодичних властивостей формованих послідовностей показали, що можливе раннє зациклювання генератора. В цьому випадку генератор формує послідовності, період яких значно (на декілька порядків) менший за максимальний. Отже, генератор Фішера – Штерна слід застосовувати з обережністю в додатках, які висувають вимоги до періодичних властивостей формованих послідовностей. В цій статті ми

пропонуємо новий генератор, який дозволяє формувати послідовності з максимально можливим періодом [7, 8].

Принципи побудови запропонованого генератора

Пропонований метод направлено на подолання недоліку генератора Фішера – Штерна, а саме зменшеної довжини періоду, порівняно з теоретично очікуваною. Головною особливістю структури нового генератора є додавання додаткових структурних елементів таких, як регістри зсуву з лінійним зворотнім зв'язком та суматору, що позначені на рисунку жирним шрифтом (рис.2) [9].

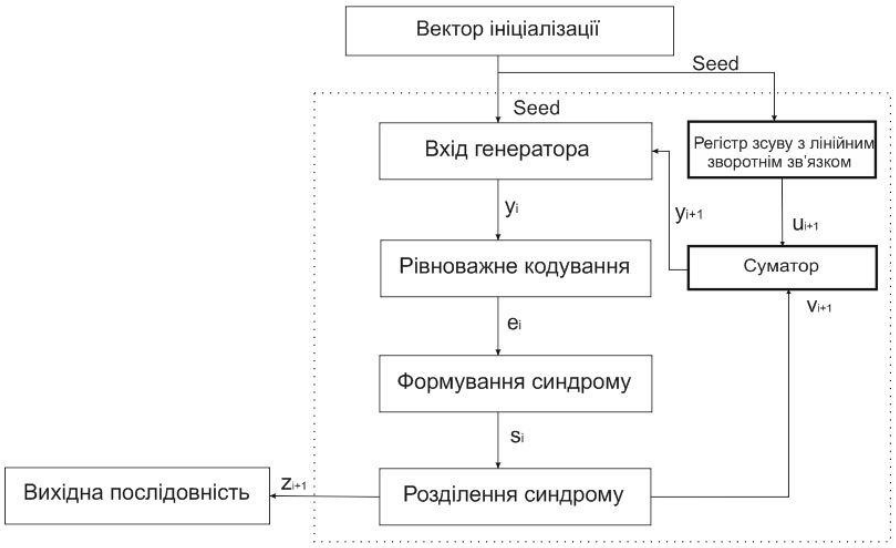

Рис. 2. Структурна схема альтернативного генератора

Аналогічно методу-прототипу в основу безпечності функціонування запропонованого генератора покладено складність вирішення проблеми синдромного декодування. Однак, як можна побачити на схемі, правило формування псевдовипадкових послідовностей змінено. Формалізуємо алгоритм формування псевдовипадкових послідовностей згідно із альтернативною схемою побудови генератора.

Етап 1. Як параметри генератора, необхідно обрати лінійний блоковий (n,k,d) код, потрібну довжину формуємої послідовності l та вектор ініціалізації(*Seed*) довжини $m = \left\lfloor \log_2 \binom{n}{t} \right\rfloor$, що подається на вхід першого циклу алгоритму.

Етап 2. Трансформування вектору ініціалізації *Seed* на вектор e_0, вага Хеммінга якого дорівнює виправляючій здатності коду ($w(e_0) = t$) за допомогою одного з методів рівноважного кодування. Величина t отримана згідно з параметром d обраного коду за правилом $t = \left\lceil \frac{d-1}{2} \right\rceil$.

Етап 3. Трансформований засобами рівноважного кодування вектор e_0 помножується на транспоновану перевірочну матрицю обраного коду:

$$s_i = e_i \cdot H^T$$

де e_i _ бітовий вектор, що має довжину n та вагу t; s_i _ отриманий двійковий вектор довжини $n-k$; H _ перевірочна матриця (n,k,d) коду. Отже, перші три етапи формування послідовності збігаються з етапами методу прототипу.

Етап 4. Вектор s_i, що представляє собою синдромну послідовність, розділяється на дві частини:

$$s_i = v_{i+1} \| z_{i+1}$$

де довжина вектору v_{i+1} дорівнює m біт, а довжина вектору z_{i+1} становить $n-k-m$ біт, що залишилися.

Етап 5. Вектор z_{i+1} стає частиною вихідної формуємої псевдовипадкової послідовності.

Етап 6. Вектор v_{i+1} поступає до суматору.

Етап 7. Початковий вектор ініціалізації $Seed$ поступає до регістру зсуву з лінійним зворотнім зв'язком, тобто задає початковий стан u_0 рекурентного перетворення. Кожний наступний стан регістру u_{i+1} обчислюється згідно з правилом [10]:

$$u_{i+1} = \varphi(u_i)$$

Поточний стан регістру далі поступає на суматор.

Етап 8. Обчислюється сума за модулем стану регістру зсуву з лінійним зворотнім зв'язком та вектору v_{i+1}, що отримано після розділення синдрому: $y_{i+1} = u_{i+1} + v_{i+1}$

Етап 9. Отриманий вектор y_{i+1} поступає на вхід генератора як вектор ініціалізації на наступному циклі перетворення. Кількість циклів, що необхідно здійснити для генерації послідовності необхідної довжини визначається за правилом:

$$r = \left\lceil \frac{l}{n-k-m} \right\rceil$$

де l – задана довжина послідовності, $n-k-m$ – довжина вектору, що поступає на вихід в кінці кожного циклу роботи.

Евристичне тестування кодових генераторів

Тестування псевдовипадкових послідовностей прийнято умовно поділяти на три групи: статистичне, евристичне та графічне. Статистичні тести використовують для встановлення чисельної оцінки якості послідовності. До евристичного тестування належать тести на криптостійкість, швидкість формування послідовностей, дослідження періоду, тест на точність визначення деяких констант методом Монте-Карло [11]. Статистичне тестування представлених генераторів було проведено у роботах [12]. У межах цієї статті проведемо дослідження генераторів з точки зору їх швидкодії, періоду послідовностей та стійкості до криптоаналізу.

Дослідження стійкості до класичного та квантового криптоаналізу

Як згадувалося вище, стійкість кодових генераторів псевдовипадкових чисел базується на складності вирішення задачі декодування синдрому. Декодування (n,k,d) коду заключається у знаходженні згідно з матрицями G та H і кодового слову з помилками $c*$ іншого кодового слова c. Отже, формулювання задачі синдромного декодування виглядає наступним чином: знайти вектор помилок з використанням синдромної послідовності.

Для загального випадку використання випадкового лінійного коду, тобто коду, у якому рядки перевірочної матриці обрано випадково та рівноймовірно, складність обчислення кореляційним способом, тобто за допомогою зіставлення кодового слова з помилками всім кодовим словам c (n,k,d) коду буде зростати експоненційно в залежності від параметрів використовуваного коду. Вирішення такої задачі для невповноваженого користувача, тобто того, кому невідомо матрицю, є NP-складною задачею. Подібні задачі вважаються стійкими навіть в умовах використання квантових обчислень.

Відповідно до досліджень, зокрема згідно з описаним у розділі 2 алгоритмом Шора, складність вирішення задач факторизації та дискретного логарифмування в умовах використання квантового комп'ютера зводиться до поліноміальної складності. З генераторами, заснованими на синдромному декодуванні, ситуація протилежна, оскільки таку задачу ототожнюють з NP-складною, звідки можемо зробити висновок, що генератори, які базуються на ній, здатні зберігати свою безпечність в постквантовому середовищі.

Складність класичного криптоаналізу щодо генераторів Фішера – Штерна та альтернативного генератора була обчислена відповідно до оцінки стійкості до перестановочного кодування, а саме оцінки мінімальної кількості покриваючих множин, що обчислюється за допомогою формули [13]:

$$N \geq \frac{C_n^t}{C_{n-k}^t} = \frac{\frac{n!}{t!(n-t)!}}{\frac{(n-k)!}{t!(n-k-t)!}} = \frac{n!(n-k-t)!}{(n-t)!(n-k)!}. \quad (1)$$

Причому $C_n^t = \frac{n!}{t!(n-t)!}$ є загальною кількістю комбінацій помилок, а $C_{n-k}^t = \frac{(n-k)!}{t!(n-k-t)!}$ – максимальною кількістю комбінацій помилок, які можуть бути покриті даною множиною.

Складність квантового криптоаналізу оцінена як кількість ітерацій, яких потребує алгоритм Гровера для вирішення задачі [14]:

$$C^{\frac{n}{2\log n}},\ C = \frac{1}{(1-R)^{1-R}}\ . \quad (2)$$

У такому випадку R є відносною швидкістю, яку забезпечує використовуваний код. Практичні результати стійкості до класичного та квантового криптоаналізу за формулами (1) та (2) наведені на рис. 3,4 (тут і далі у логарифмічному масштабі за основою 2). Оцінки були проведені щодо різних наборів параметрів (n,k,t) коду. Параметр n обрано у діапазоні від 1024 до 8192.

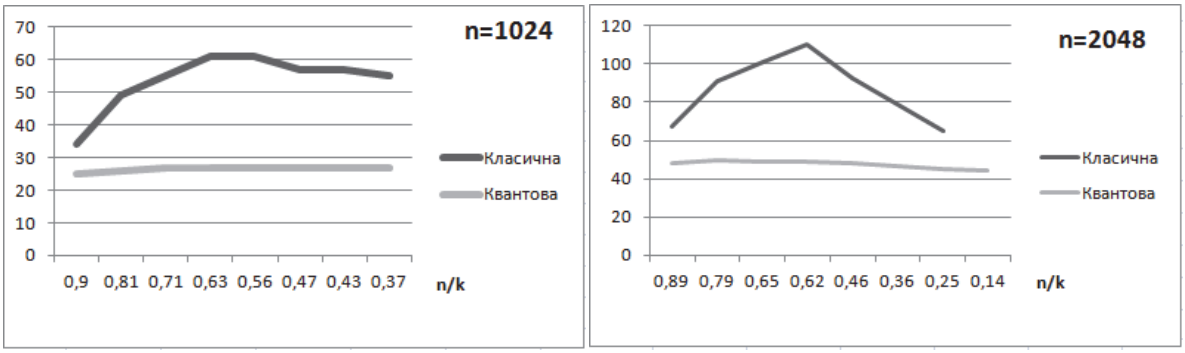

Рис. 3. Стійкість генераторів при n=1024 та при n=2048 відповідно

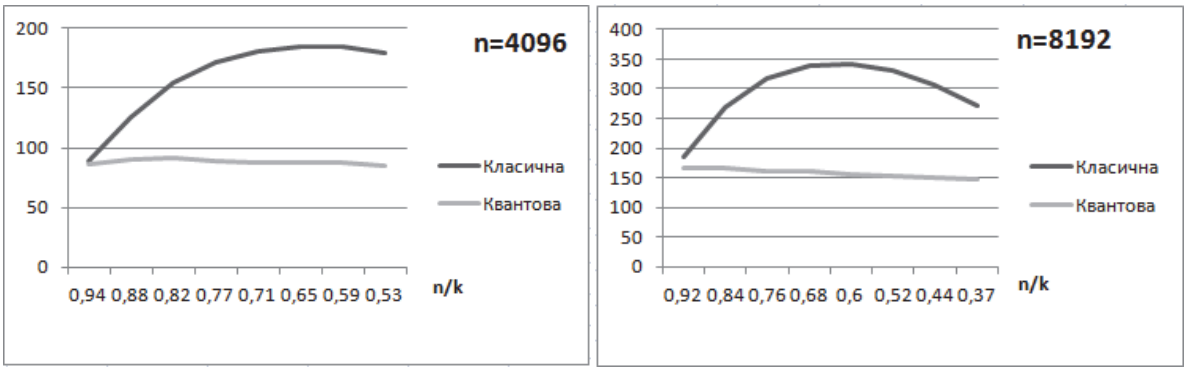

Рис. 4. Стійкість генераторів при n=4096 та n=8192 відповідно

Інформація щодо стійкості задач факторизації та дискретного логарифмування доведена у ряді англомовних видань, а також освітлена у [45].

Аналізуючи отримані результати, можна зробити висновок, що найбільш ефективним з точки зору безпеки є параметри коду, при якому відношення параметру k до параметру n

приблизно дорівнює 0,66. З іншого боку, при збільшенні параметру n стійкість як до класичного, так і до квантового криптоаналізу стрімко зростає (рис. 5).

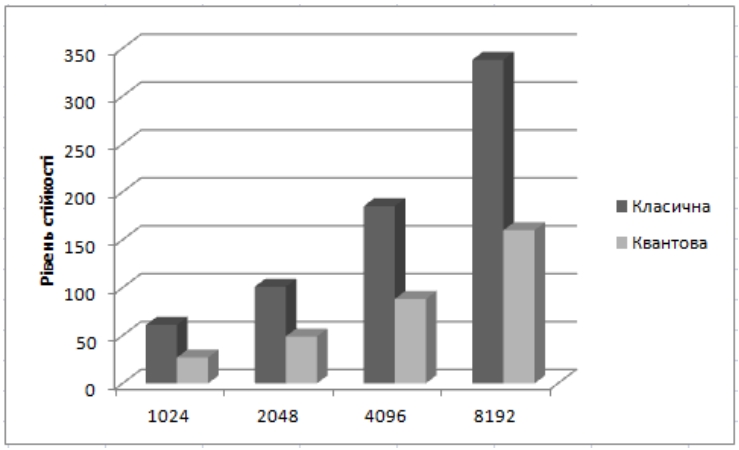

Рис. 5. Стійкість генераторів при співвідношенні $k/n \approx 0,66$ та різних n

Очевидно, що у постквантовому середовищі використовувані параметри повинні зростати і рекомендованими параметрами у цьому випадку є n від 4096.

Як згадувалося раніше, в основі доказово стійких генераторів лежить використання теоретико-складнісних задач. На сьогодні стійкість більшості криптографічних схем базується на складності вирішення задач факторизації (генератор Blum-Blum-Shub) та дискретного логарифмування (Dual_EC_DRBG). Оцінка цих задач описані у ряді джерел, зокрема у [15]. Порівняння стійкості до квантового криптоаналізу задач факторизації, дискретного логарифмування та синдромного декодування продемонстрована на рис. 6 відносно класичного рівня стійкості у 112, 192 та 256 біт.

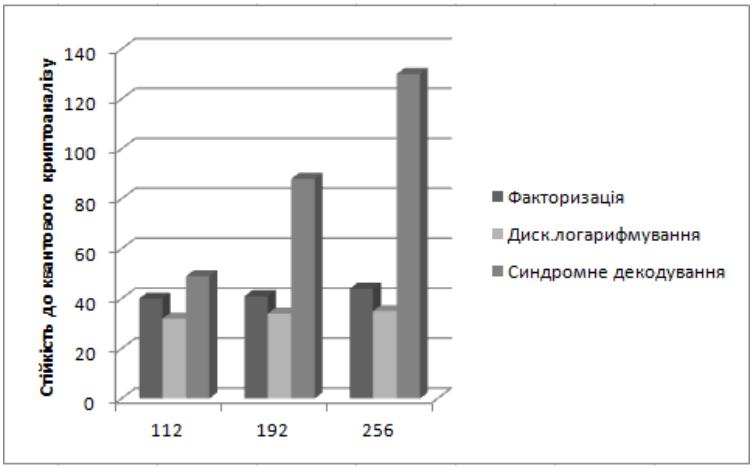

Рис. 6. Стійкість до квантового криптоаналізу теоретико-складністних задач

Аналізуючи дані, варто зазначити, що збільшення параметрів перетворення для генераторів, заснованих на задачі факторизації та на задачі дискретного логарифмування не призводить до суттєвого збільшення стійкості до квантового криптоаналізу, а також, порівняно із класичним криптоаналізом застосування квантових комп'ютерів значно зменшує їх стійкість. У той же час квантовий криптоаналіз синдромного декодування зменшує стійкість генератора відносно класичного криптоаналізу приблизно вдвічі (у логарифмічному масштабі за основою 2), що надає значну перевагу для постквантового періоду[16].

Дослідження довжини періоду послідовностей

Дослідження однієї з важливих характеристик генераторів псевдовипадкових чисел, які можуть застосуватися у криптографії, є довжина періоду послідовності, що гарантує відсутність зациклювання в межах задачі, що вирішується.

Довжина максимального періоду, що забезпечує використання генератора дорівнює $L = 2^m - 1$. Ці розрахунки випливають з того, що вектор ініціалізації, що подається на кожну ітерацію роботи генератора, $m = \left\lfloor \log_2 \binom{n}{t} \right\rfloor$, тому максимальна кількість різних ненульових векторів на кожній ітерації алгоритму генератора дорівнює $2^m - 1$ [17].

Розроблені програмні реалізації були протестовані з точки зору довжини періоду послідовності, що формується згідно з ними. На вхід кожного з генераторів було подано усі можливі конфігурації вектору ініціалізації, тобто було здійснено $2^m - 1$ формувань послідовностей. Параметри коду, для якого було здійснено тестування для кожного з генераторів, дорівнюють $(31,16,7)$. Таким чином, довжина вектору ініціалізації $m = 12$ біт, а довжина періоду відповідно до теоретичних розрахунків повинна бути $L = 2^{12} - 1 = 4095$. Другим випадком, що було розглянуто, є формування послідовності, з параметрами коду $(31,11,5)$. Довжина вектору ініціалізації при вказаних параметрах $m = 17$, а відповідна теоретична довжина періоду $L = 2^{17} - 1 = 131071$. Результати, отримані для генератора Фішера – Штерна, наведено на графіках (рис. 7, 8) [18]. На графіках продемонстровано відношення довжини періоду до кількості векторів ініціалізації, при яких вона отримана. З метою кращого візуального сприйняття графік побудовано у логарифмічному масштабі за основою 10 щодо осі ординат (формула $x = \log_{10} X$, де X є параметром, який слід перетворити; x представляє результат обчислення десяткового логарифма над масштабованим значенням).

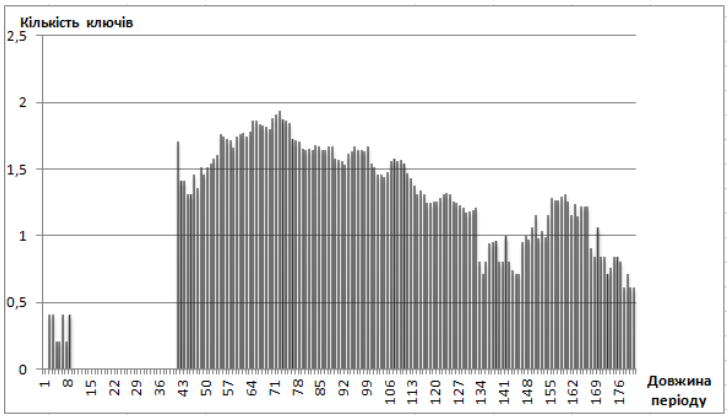

Рис. 7. Розподілення кількості ключів за довжинами періодів послідовності згідно з генератором Фішера – Штерна для коду $(31,16,7)$

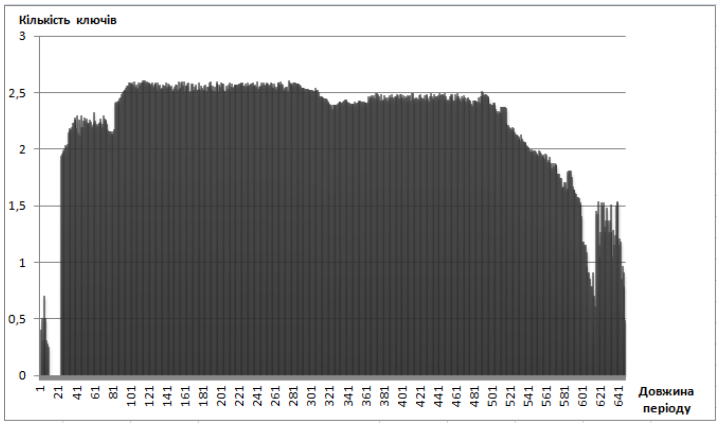

Рис. 8. Розподілення ключів за довжинами періодів послідовності згідно з генератором Фішера – Штерна для $(31,11,5)$ коду

Аналізуючи дані, представлені на графіках, можна зробити висновок, що, незважаючи на те, який вектор ініціалізації буде подано на вхід генератора, досягти максимальної довжина періоду не є можливим на практиці. Більш того, практично отримувані довжини набагато менші з теоретично очікуваною. У кращому випадку при застосуванні коду $(31,16,7)$ практична довжина періоду приблизно в 22 рази менша, ніж теоретично заявлена. У свою чергу кращий випадок для коду $(31,11,5)$ демонструє різницю між довжинами приблизно у 200 разів. Таким чином, зі зростанням довжини вектору ініціалізації, що подається на вхід, а відповідно і вектору кожної ітерації, різниця між теоретичним та практичним результатами буде збільшуватися.

Слід зазначити, що аналогічне тестування запропонованого генератора підтверджує формування ним послідовностей максимального періоду. Дійсно, для усіх можливих значень векторів ініціалізації було сформовано псевдовипадкову послідовність з періодами $L = 2^{12} - 1 = 4095$ (для випадку коду $(31,16,7)$) і $L = 2^{17} - 1 = 131071$ (при застосуванні коду $(31,11,5)$).

Дослідження швидкодії представлених генераторів

Швидкодія кодових генераторів перевірена за допомогою їх програмних реалізацій на мові програмування Java. Реалізаціє не є еталонною, але дозволяє перевірити відносну швидкість двох генераторів. Встановлена довжина формуємої послідовності дорівнює 10^6 біт. Результати швидкодії для кожного генератору отримані для різних комбінацій параметрів коду, що забезпечують різний рівень стійкості, для 10 випадків та освітлені усереднені показники.

Таблиця 1

Показники швидкодії представлених генераторів

Генератор Фішера – Штерна			
Параметри (n,k,t)	Стійкість квантова, біт	Швидкість формування ключових даних, мс	Швидкість генерації послідовності, мс
(127,85,6)	5	4	1,352
(255,177,11)	8	9	6, 317
(511,340,20)	15	23	8,903
(1023,678,37)	27	45	19,111
(2047,1328,60)	50	117	36,261
(4095,3376,67)	91	180	82,663
(8192,6892,100)	167	555	218,238
Запропонований генератор			
Параметри (n,k,t)	Стійкість квантова, біт	Швидкість формування ключових даних, мс	Швидкість генерації послідовності, мс
(127,85,6)	5	4	1,364
(255,177,11)	8	9	6,448
(511,340,20)	15	23	9,213
(1023,678,37)	27	45	19,632
(2047,1328,60)	50	117	38,569
(4095,3376,67)	91	180	86,557
(8192,6892,100)	167	555	228,119

Представимо наведені дані у вигляді залежності часу формування послідовності від параметра *n* використовуваного коду.

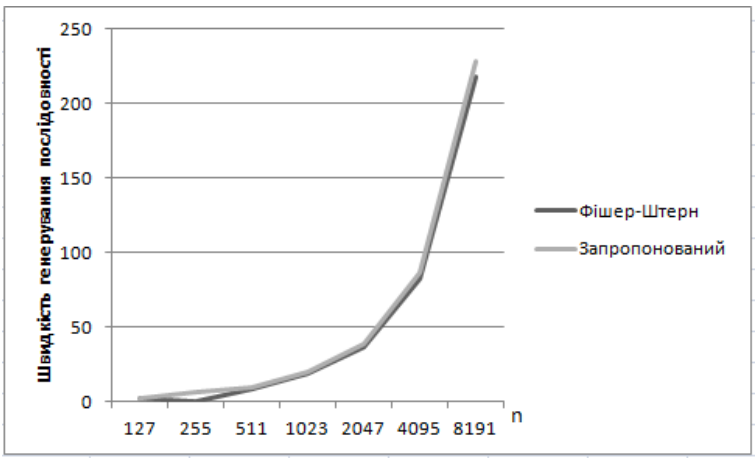

Рис. 9. Залежність швидкості формування послідовності від параметрів коду

Таким чином, можна зробити висновок, що збільшення часу формування псевдовипадкової послідовності згідно з запропонованим генератором за рахунок додання до схеми регістру зсуву з лінійним зворотнім зв'язком є незначним, при цьому він дозволяє зберігати стійкість і забезпечує максимальний період послідовності, на відміну від свого попередника.

Висновки

Перспективним напрямком у сфері постквантової криптографії вважається криптографія, що базується на кодах. Відповідно до статистики конкурсу NIST вона посідає друге місце не тільки у кількості поданих на розгляд проектів, але і у кількості схем, що пройшли до другого туру шляхом досліджень криптографів з усього світу. При цьому застосування принципів побудови кодових криптосистем до реалізації генераторів є можливим варіантом вирішення задачі реалізації постквантового генератора псевдовипадкових послідовностей. Використання даного підходу дозволяє забезпечити ряд переваг: стійкість до класичного та квантового крипто аналізу, високу швидкість криптоперетворень, а також доказову безпеку.

Дослідження кодових генераторів виявили, що класичним у цьому контексті є генератор, розроблений Фішером та Штерном, на його основі базуються подальші варіації кодових генераторів. Однак, алгоритм Фішера – Штерна має вагомий недолік: теоретично обчислювана та реальна довжина періоду послідовності, яка є важливою з криптографічної точки зору, суттєво відрізняються. Для його подолання було запропоновано альтернативний варіант генератору, особливістю реалізації якого є додаткове використання суматору та регістру зсуву з лінійним зворотнім зв'язком. Тестування підтвердило, що подібна побудова генератору, зберігає стійкість до класичного та квантового криптоаналізу, при цьому збільшення часу, що необхідний для формування послідовностей, є незначний у порівнянні з попередником. Також швидкодія може бути збільшена за рахунок застосування швидкого перетворення послідовності у послідовність з постійною вагою, що є актуальною темою для подальших досліджень.

Список літератури:

1. Johnston D. Random Number Generators-Principles and Practices // Sep. 2018. doi:10.1515/9781501506062.
2. Menezes A., Oorschot P. van, Vanstone S. Handbook of Applied Cryptography. CRC-Press, 1996. 816 p.
3. Perlner R. A., Cooper D.A. Quantum Resistant Public Key Cryptography: A Survey IDtrust '09, April 14- 16, 2009, Gaithersburg, MD, pp. 85-93. URL: https://ws680.nist.gov/publication/get_pdf.cfm?pub_id= 901595
4. Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner and Daniel Smith-Tone. NISTIR 8105. Report on Post-Quantum Cryptography. National Institute of Standards and Technology, Internal Report 8105, April 2016. 10 p.
5. Elaine Barker and John Kelsey. Recommendation for random number generation using deterministic random bit generators. National Institute of Standards and Technology, January 2012. 124 p. URL: https://doi.org/10.6028/NIST.SP.800–90A
6. Jean-Dernard Fisher, Jacques Stern. An efficient PseudoRandom Generator Provably as Secure as Syndrome Decodingr // EUROCRYPT'96 Proceeding, LNCS 1070. P. 245–255.

7. Andrea Rock. Pseudorandom Number Generators for Cryptographic Applications // Diplomarbeit zur Erlangung des Magistergrades an der Naturwissenschaftlichen Fakultat der Paris-Lodron-Universitat Salzburg. Salzburg. 2005.

8. Hastad J. Pseudorandom number generators from any one-way function / J. Hastad, R. Impagliazzo, L.A. Levin, M. Luby // SIAM Journal on Computing. 1999.Vol. 28. P.1364-1396.

9. Kuznetsov A., Kavun S., Panchenko V., Prokopovych-Tkachenko D., Kurinniy F. and Shoiko V. Periodic Properties of Cryptographically Strong Pseudorandom Sequences // 2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T), Kharkiv, Ukraine, 2018. P. 129-134. doi: 10.1109/INFOCOMMST.2018.8632021

10. Dubrova E. (2009). How to speed-up your NLFSR-based stream cipher. 878 – 881. 10.1109/DATE.2009.5090786.

11. Kuznetsov A.A., Kiian A.S., Prokopovich-Tkachenko D.I., Zverev V.P., Kotuh E.V., Kuznetsova T.Y. Periodical properties of cryptographic resistance pseudorandom sequences // Applied redioelectronics: sci.-tech. magazine. 2018. Vol.17, №. 3, 4. P. 96–103.

12. Grover L. A fast quantum mechanical algorithm for database search // Proceedings of the 28th annual ACM symposium on the theory of computing (STOC, 96). ACM Press, New York. 1996. P. 212–219.

13. Stipcevic M., Koc C.K. True random number generators // Open Problems in Mathematics and Computational Science. Springer, 2014. P. 275–315.

14. Grover L. A fast quantum mechanics algorithm for database search», Proceeding of the 28th ACM Symposium on Theory of Computation. New York: ACM Press, 1996. P. 212-219.

15. S. P. W: Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer // S IAM J. Comput, 1997.

16. Andrew Rukhin. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf

17. Vlad Gheorghiu, Michele Mosca. Benchmarking the quantum cryptanalysis of symmetric, public-key and hash-based cryptographic schemes,URL: https://arxiv.org/pdf/1902.02332.pdf

18. Blum M.. How to generate cryptographycally strong sequences of pseudorandom bits / M. Blum, S. Micali // SIAM Journal on Computing. 1986. Vol. 1. P.850-864.

Харківський національний
університет імені В. Н. Каразіна;
АТ «Інститут інформаційних технологій»

Надійшла до редколегії 14.01.2020