Вісник ЛДУБЖД
Bulletin of Lviv State University of Life Safety
https://journal.ldubgd.edu.ua/index.php/Visnuk
ISSN 2078-4643 (print), ISSN 2708-1389 (online)

UDC 004.056.2:004.421.5
DOI: 10.32447/20784643.30.2024.12

М .В. Кіх, О. А. Нємкова
Національний університет «Львівська Політехніка», м. Львів, Україна
ORCID: https://orcid.org/0009-0007-1847-1862 – М . В. Кіх
https://orcid.org/0000-0003-0690-2657 – О. А. Нємкова
mykhailo.v.kikh@lpnu.ua

МЕТОДОЛОГІЯ ПОКРАЩЕННЯ ПРОЦЕСУ ГЕНЕРУВАННЯ ПСЕВДОВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ

Постановка проблеми. У сфері кібербезпеки генерація псевдовипадкових послідовностей є ключовим механізмом, який забезпечує захист даних і цілісність криптографічних систем. Псевдовипадкові послідовності є важливою складовою багатьох комп'ютерних систем та програм, що використовують випадковість для різних цілей, від криптографії до симуляцій. Однак існують деякі проблеми та виклики, пов'язані з процесом генерації псевдовипадкових послідовностей, які потребують уваги та вирішення. Одна з ключових проблем – недостатня якість та безпека псевдовипадкових послідовностей, яка може призвести до вразливостей систем та недостатньої стійкості криптографічних протоколів. Інші проблеми включають недостатню швидкість генерації, обмеженість доступних ресурсів, а також недостатню випадковість у деяких сценаріях використання. Для вирішення цих проблем потрібно розглянути різні шляхи покращення процесу генерації псевдовипадкових послідовностей, включаючи оптимізацію швидкості та ефективності, а також збільшення ентропії та криптостійкості.

Мета. Дослідити та систематизувати основні шляхи покращення процесу генерування псевдовипадкових послідовностей для забезпечення підвищеної надійності та ефективності. У роботі ставиться завдання розглянути та проаналізувати методи, які включають впровадження більш надійних алгоритмів, вибір оптимальних початкових значень, перевірку на відповідність стандартам, моніторинг та аудит системи генерації, а також оптимізацію швидкості та продуктивності. Це дозволить створити методологію, застосування якої підвищить якість згенерованих псевдовипадкових послідовностей.

Результати. У ході дослідження була розроблена комплексна методологія покращення процесу генерації псевдовипадкових послідовностей. Ця методологія охоплює кілька ключових аспектів, включаючи вибір оптимальних алгоритмів генерації, налаштування початкових значень, перевірку на відповідність міжнародним стандартам, а також проведення моніторингу та аудиту системи генерації. Автори дослідили стратегії застосування цих методів, а також їх переваги для покращення процесу генерації псевдовипадкових чисел. У статті було ретельно проаналізовано характеристики криптографічно стійких алгоритмів, таких як Fortuna, Yarrow, Blum Blum Shub та ISAAC. Результати дослідження підкреслюють необхідність комплексного підходу до підвищення якості генерації псевдовипадкових послідовностей, що передбачає не лише збільшення криптографічної стійкості, але й забезпечення високої ефективності роботи генераторів, а також їх відповідності чинним стандартам. Запропоновані методи та підходи можуть суттєво полегшити роботу розробникам систем безпеки, дослідникам у галузі криптографії та фахівцям, які займаються генерацією псевдовипадкових послідовностей у своїй професійній діяльності.

Висновки. Було запропоновано методологію покращення процесу генерування псевдовипадкових послідовностей, яка включає в себе використання кращих алгоритмів, вибір оптимальних початкових значень, перевірку на відповідність стандартам, моніторинг та аудит системи генерації, оптимізацію швидкості та ефективності. Ці підходи дозволяють забезпечити високу якість та надійність псевдовипадкових послідовностей у різних сценаріях використання.

Ключові слова: кібербезпека, генерація псевдовипадкових послідовностей, криптографічна стійкість, стандарти, моніторинг та аудит, оптимізація.

M. V. Kikh, E. A. Nyemkova
Lviv Polytechnic National University, Lviv, Ukraine

METHODOLOGY FOR IMPROVING THE PROCESS OF PSEUDO-RANDOM SEQUENCES GENERATING

Introduction. The generation of pseudorandom sequences is a key mechanism that ensures data protection and the integrity of cryptographic systems in cybersecurity. Pseudorandom sequences are an essential component of many computer systems and programs that use randomness for a variety of purposes, from cryptography to simulations.

However, there are some problems and challenges associated with the process of generating pseudorandom sequences that need attention and solutions. One of the key problems is the insufficient quality and security of pseudorandom sequences, which can lead to system vulnerabilities and insufficient stability of cryptographic protocols. Other problems include insufficient generation speed, limited resources available, and insufficient randomness in some cases. To address these issues, various ways to improve the pseudorandom sequence generation process, including optimizing speed and efficiency as well as increasing entropy and cryptoresistance must be considered.

Purpose. The purpose of the research is to explore and systematize the main ways to improve the process of generating pseudorandom sequences to ensure increased reliability and efficiency. The paper aims to review and analyze methods that include the implementation of more reliable algorithms, selection of optimal initial values, verification of compliance with standards, monitoring, and auditing of the generation system, as well as optimization of speed and performance. This will enable the development of a methodology that will enhance the quality of generated pseudorandom sequences.

Results. A comprehensive methodology for improving the process of generating pseudorandom systems was developed during the research. This methodology covers several key aspects, including the generation of optimal algorithms, setting of initial values, sampling for compliance with international standards, and monitoring and auditing of system generation. The authors explored strategies for applying these methods, as well as their advantages for improving the pseudorandom number generation process. The characteristics of cryptographically robust algorithms such as Fortuna, Yarrow, Blum Blum Shub and ISAAC were analyzed. The results of the study emphasize the necessary comprehensive approach to improving the quality of generation of pseudorandom systems, which provides not only an increase in cryptoresistance but also ensures the high efficiency of the generators, as well as their compliance with current standards. The proposed methods and approaches can significantly facilitate the work of security system developers, cryptographic researchers and specialists dealing with the generation of pseudorandom sequences.

Conclusion. A methodology for improving the process of generating pseudorandom sequences was proposed, which includes the use of better algorithms, the selection of optimal initial values, verification of compliance with standards, monitoring and auditing of the generation system, and optimization of speed and efficiency. These approaches allow for the high quality and reliability of pseudorandom sequences in various usage scenarios.

Keywords: cybersecurity, generation of pseudorandom sequences, cryptoresistance, standards, monitoring and auditing, optimization.

Вступ. Покращення процесу генерування псевдовипадкових послідовностей має велике значення в різних сферах та застосуваннях. Ось декілька ключових причин, чому варто розглядати покращення цього процесу [1-12]:

1. Кібербезпека. Псевдовипадкові послідовності широко використовуються в криптографічних протоколах, таких як шифрування даних [1, 2], створення цифрових підписів [2, 3] та автентифікація [2, 4]. Якщо процес генерації псевдовипадкових чисел не є надійним, це може призвести до порушень безпеки, таких як підбір паролів або атаки на шифри.

2. Криптовалютні технології. Криптовалютні системи, такі як Bitcoin та Ethereum, використовують псевдовипадкові послідовності для генерації ключів шифрування та підписів транзакцій [5, 6]. Ненадійність генерації псевдовипадкових чисел може призвести до вразливостей та втрати коштів.

3. Тестування та валідація. У тестуванні програмного забезпечення [7], особливо у сценаріях тестування на велику кількість випадкових входів [8, 9], важливо мати доступ до високоякісних псевдовипадкових чисел. Покращений процес генерації допомагає забезпечити надійне тестування та валідацію програм.

4. Моделювання та симуляція. У науці [3], інженерії [7] та інших галузях [7, 10] моделювання та симуляції використовують псевдовипадкові числа для створення реалістичних та непередбачуваних сценаріїв. Покращений процес генерації допомагає забезпечити точність та достовірність таких моделей.

5. Ігрова та розважальна індустрія. У відеоіграх та інтерактивних додатках псевдовипадкові послідовності використовуються для генерації випадкових подій, рівнів, предметів [7, 11, 12] тощо, що забезпечує різноманіття та цікавість гри.

Отже, покращення процесу генерації псевдовипадкових послідовностей є критично важливим для кібербезпеки, ефективності та надійності різних систем та застосувань у сучасному інформаційному світі.

Методи дослідження. Застосовано такі методи досліджень: структурно-системний метод, метод порівняльного аналізу, формально-логічний метод та метод експертного оцінювання. Структурно-системний метод дав змогу проаналізувати процес генерації псевдовипадкових послідовностей як цілісну систему, визначити її основні елементи та взаємозв'язки між ними. Застосування цього методу допомогло виявити критичні аспекти, що впливають на надійність і безпеку генерації. Метод порівняльного аналізу було використано для оцінки різних алгоритмів генерації псевдовипадкових послідовностей (Fortuna, Yarrow, Blum Blum Shub та ISAAC), зокрема їхньої продуктивності, криптографічної стійкості та відповідності стандартам безпеки. Це дало змогу обґрунтовано вибрати кращі алгоритми для конкретних застосувань. Формально-логічний

метод забезпечив логічне обґрунтування застосованих методів покращення, таких як вибір оптимальних початкових значень, моніторинг і аудит. Метод використано для структурування та впорядкування рекомендацій, що підвищують ефективність генерації псевдовипадкових чисел. Метод експертного оцінювання використовувався для збору думок фахівців у галузі криптографії та інформаційної безпеки, що дало змогу отримати цінні рекомендації щодо покращення процесу генерації псевдовипадкових чисел.

Використання цих методів забезпечило комплексний підхід до аналізу та оптимізації процесу генерації псевдовипадкових послідовностей, дозволяючи зробити висновки щодо поліпшення їхньої ефективності, безпеки та стабільності.

Огляд попередніх досліджень.

Результати досліджень. Генератори псевдовипадкових послідовностей використовують детерміновані алгоритми для створення випадкових послідовностей. Однак ці генератори не є справді випадковими і можуть бути передбачуваними, якщо зловмиснику відомі алгоритм і початкове значення.

Покращення процесу генерування псевдовипадкових послідовностей можна розглядати з різних точок зору, включаючи якість випадковості, швидкість генерації та використання ресурсів. Було досліджено такі шляхи покращення до процесу генерування псевдовипадкових послідовностей.

Використання кращих алгоритмів. Використання кращих алгоритмів генерації псевдовипадкових послідовностей може покращити якість випадковості послідовностей.

Cryptographically Secure Pseudo-Random Number Generators (CSPRNGs) – криптографічно стійкі генератори псевдовипадкових чисел (КСГПВЧ) – це алгоритми генерації псевдовипадкових послідовностей, які мають властивості криптографічної стійкості [3]. Вони використовуються для створення випадкових послідовностей, які важко передбачити або відновити, що робить їх ідеальними для застосувань, де важлива безпека, такі як криптографія, генерація ключів, автентифікація та інші.

CSPRNGs гарантують, що послідовності, які вони генерують, мають властивості криптографічної стійкості – їх важко передбачити або відновити без знання початкових умов. КСГПВЧ повинні надавати випадковість, тобто кожне число в послідовності має з'являтися незалежно та рівномірно, не маючи жодних кореляцій з іншими числами. CSPRNGs використовуються в криптографії для створення ключів шифрування та цифрового підпису, у процесах автентифікації, при генерації випадкових чисел для захисту конфіденційної інформації, та в інших застосуваннях, де важлива безпека генерації випадкових чисел. КСГПВЧ проходять різноманітні тестування на криптографічну стійкість, щоб забезпечити відповідність міжнародним стандартам безпеки та надійності [13]. Використання CSPRNGs дає змогу забезпечити високий рівень безпеки та надійності в проектах, де потрібна генерація псевдовипадкових чисел.

Існує багато криптографічно стійких алгоритмів, які можуть бути використані для генерації псевдовипадкових послідовностей. Було досліджено – Fortuna, Yarrow, Blum Blum Shub (BBS) та ISAAC.

1. Fortuna – це криптографічно стійкий алгоритм генерації псевдовипадкових послідовностей, який був розроблений Брюсом Шнайером та Нілом Фергюсоном. Його основна ідея полягає в тому, щоб комбінувати надійність різних джерел випадковості для створення псевдовипадкових послідовностей [2].

Основні характеристики алгоритму Fortuna:

- Fortuna має багаторівневу архітектуру, що дозволяє комбінувати випадковість з різних джерел. Це включає в себе фонове джерело, яке постійно генерує випадкові дані, та генератори, які запускаються за необхідності.

- Fortuna використовує криптографічні хеш-функції для створення псевдовипадкових чисел з високою випадковістю. Кожен генератор підвищення рівня може бути запущений після певної кількості вхідних байтів або після певного часу.

- Fortuna забезпечує високий рівень надійності та безпеки. Він відповідає вимогам для криптографічно стійкого генератора псевдовипадкових чисел.

- Якщо один з генераторів Fortuna стає недоступним або виявляється компрометованим, інші генератори можуть бути використані для продовження генерації псевдовипадкових чисел.

Fortuna є популярним алгоритмом, оскільки він забезпечує високий рівень безпеки та надійності, що робить його підходящим для різних криптографічних застосувань та систем, де важлива генерація випадкових послідовностей.

2. Yarrow – це криптографічно стійкий алгоритм генерації псевдовипадкових послідовностей, розроблений Брюсом Шнайером та Нілом Фергюсоном. Цей алгоритм спеціально призначений для створення високоякісних випадкових послідовностей, що використовуються в криптографічних додатках [14].

Основні характеристики алгоритму Yarrow:

- Yarrow базується на зборі ентропії з різних джерел, таких як часові затримки, мережевий

трафік, дії користувача тощо. Ці джерела надають випадковість, яка використовується для генерації псевдовипадкових послідовностей.

- Yarrow може адаптуватися до змін у середовищі, включаючи зміни в якості джерел ентропії або кількості доступної ентропії. Це дозволяє алгоритму підтримувати високу якість випадкових послідовностей в різних умовах.

- У разі, якщо рівень ентропії стає надто низьким, Yarrow може перейти у реактивний режим, де він продовжує генерацію псевдовипадкових послідовностей, використовуючи алгоритми, що базуються на блочних шифрах.

- Yarrow відомий своєю високою стійкістю до криптоаналізу, що робить його придатним для використання в криптографічних застосунках, де потрібна надійна генерація випадкових послідовностей.

- Yarrow став основою для стандарту NIST SP 800-90A, який визначає вимоги до генераторів псевдовипадкових послідовностей в криптографічних застосунках [14].

Yarrow є важливим інструментом для забезпечення безпеки в криптографічних системах, де потрібна надійна генерація.

3. Blum Blum Shub (BBS) – це криптографічно стійкий алгоритм, який був розроблений Леонардом Адлеманом, Мануєлем Блюмом та Майклом Шубом. Він базується на складності факторизації великих цілих чисел і відомий своєю високою безпекою та простотою реалізації [1].

Основні характеристики алгоритму Blum Blum Shub (BBS):

- BBS відомий своєю високою криптографічною стійкістю. Ця стійкість полягає в тому, що обчислення наступного числа в послідовності вимагає знання простих множників великого числа N, що є важко в обчислювальному сенсі [1].

- Алгоритм BBS має просту структуру та легко реалізується на різних платформах. Він складається з кількох базових операцій з великими цілими числами, таких як модульне піднесення до степеня та взяття залишку.

- BBS потребує вибору двох великих простих чисел p та q, які близькі за розміром. Ці параметри можуть бути вибрані таким чином, щоб забезпечити високу криптографічну стійкість.

- Однак, через велику складність операцій з великими цілими числами, BBS може бути повільним у порівнянні з іншими генераторами псевдовипадкових послідовностей, особливо при генерації великих обсягів випадкових даних.

- Криптографічна стійкість BBS ґрунтується на складності задачі факторизації великих цілих чисел. Знайдення простих множників великого числа N є важкою обчислювальною задачею, що робить алгоритм відносно стійким до атак.

- При виборі відповідних параметрів, BBS може мати дуже велику періодичність, що дає змогу генерувати довгі послідовності псевдовипадкових чисел без повторень.

Загалом, Blum Blum Shub є потужним криптографічно стійким генератором псевдовипадкових послідовностей, який забезпечує високий рівень безпеки у відношенні до криптографічних атак. Однак він може бути менш ефективним у порівнянні з іншими генераторами через свою низьку швидкість.

4. ISAAC (indirection, shift, accumulate, add, and count) – це криптографічно стійкий генератор псевдовипадкових послідовностей, розроблений Бобом Дженкінсоном. ISAAC відомий своєю високою якістю випадковості та швидкістю генерації, а також простотою реалізації [15].

Основні характеристики алгоритму ISAAC:

- ISAAC – це криптографічно стійкий генератор псевдовипадкових чисел (CSPRNG), який забезпечує високий рівень випадковості та стійкості до криптографічних атак. Випадкові послідовності, згенеровані ISAAC, відповідають високим стандартам безпеки.

- ISAAC генерує послідовності псевдовипадкових чисел високої якості, що важливо для безпечного застосування в криптографічних протоколах, шифруванні, генерації ключів та інших застосуваннях.

- ISAAC відомий своєю високою швидкістю генерації випадкових чисел. Це дозволяє ефективно використовувати його в різних програмних та апаратних застосуваннях, де важлива продуктивність та швидкодія [15].

- Алгоритм ISAAC має просту структуру та легко реалізовується на різних платформах. Його реалізація може бути виконана як на програмному, так і на апаратному рівнях, що робить його дуже доступним для різноманітних застосувань.

- ISAAC має дуже велику періодичність, що дозволяє генерувати довгі послідовності випадкових чисел без повторень. Це важливо для багатьох застосувань, де потрібно забезпечити велику кількість унікальних випадкових значень.

Загалом, ISAAC є потужним і надійним криптографічно стійким генератором псевдовипадкових чисел, який добре підходить для багатьох криптографічних застосувань та систем, де важлива якість та безпека випадкових даних.

Вибір оптимальних початкових значень. Вибір оптимальних початкових значень (seed) є критично важливим для забезпечення якості та безпеки генерації псевдовипадкових послідовностей. Правильний вибір початкових значень може значно покращити випадковість,

захистити від атак та забезпечити стабільну роботу системи. Відзначимо декілька ключових аспектів та методів вибору оптимальних початкових значень. Початкові значення повинні мати високу ентропію, тобто бути максимально випадковими. Це унеможливлює передбачення згенерованих послідовностей. Потрібно використовувати надійні джерела ентропії, такі як системні події, час виконання програм тощо [10]. Варто поєднувати ентропію з різних незалежних джерел для формування початкового значення [10]. Це може включати час запуску системи, активність користувача, мережевий трафік тощо. Важливо уникати ситуацій, коли зловмисник може передбачити або повторити початкові значення. Потрібно використовувати процедури, які забезпечують, що кожне початкове значення є унікальним і непередбачуваним [11]. Початкові значення мають оновлюватися через певні інтервали часу або після визначеної кількості використань для забезпечення безпеки і непередбачуваності вихідних послідовностей [16]. Розмір початкового значення повинен відповідати вимогам конкретного генератора. Наприклад, для деяких алгоритмів необхідно мати початкові значення певної довжини (наприклад, 256 бітів).

Приклади методів вибору початкових значень [17]:

- /dev/random та /dev/urandom: у Unix-подібних системах ці джерела можуть бути використані для отримання високоякісних випадкових значень з системної ентропії.

- Hardware Random Number Generators (HRNG): метод забезпечує створення випадкових значень на основі фізичних процесів з використанням апаратних генераторів.

- Cryptographic functions: метод використовує криптографічні хеш-функції, такі як SHA-256, для обробки зібраної ентропії та отримання високоякісних початкових значень.

Вибір оптимальних початкових значень для генераторів псевдовипадкових послідовностей має кілька ключових переваг, що безпосередньо впливають на якість, безпеку та ефективність генерованих послідовностей. Оптимальні початкові значення, що мають високу ентропію, унеможливлюють передбачення вихідних даних. Це захищає систему від атак, таких як атаки з відновлення початкових значень або передбачення майбутніх значень псевдовипадкової послідовності. Використання оптимальних початкових значень покращує статистичні властивості генерованих послідовностей. Це забезпечує рівномірний розподіл і відсутність помітних шаблонів, що є важливим для багатьох застосувань, включаючи моделювання, криптографію та ігрові алгоритми [18]. Оптимальні початкові значення забезпечують унікальність згенерованих послідовностей, що зменшує ймовірність повторення тих самих послідовностей при кожному запуску генератора. Це особливо важливо для безпечної передачі даних і автентифікації [10]. Висока ентропія початкових значень робить вихідні послідовності стійкими до криптоаналізу, знижуючи можливість зловмисникам знайти вразливості в алгоритмі генерації випадкових чисел. Оптимальні початкові значення зменшують можливість прогнозування вихідних даних на основі попередніх значень, що підвищує загальну стабільність і надійність роботи системи.

Хоча вибір оптимальних початкових значень може вимагати певних витрат ресурсів на генерацію і обробку ентропії, це забезпечує ефективне і безпечне функціонування системи в довгостроковій перспективі. Вибір різних оптимальних початкових значень для декількох генераторів в одній системі забезпечує асинхронність генерованих послідовностей, що запобігає їх синхронізації і повторенню в багатокористувацьких або розподілених середовищах. Проте, збільшення довжини seed може призвести до збільшення обчислювальних витрат на генерацію псевдовипадкових чисел, особливо в разі великої кількості генерованих чисел або при обмежених ресурсах. Тому важливо підтримувати баланс між безпекою, ефективністю та потребами конкретного застосування.

Перевірка на відповідність стандартам. Перевірка на відповідність стандартам є важливим етапом для покращення процесу генерації псевдовипадкових послідовностей. На відповідність стандартам можна перевірити як сам процес генерації, так і самі алгоритми, що використовуються [19, 20]. Потрібно почати з вибору стандартів, які відповідають вибраним вимогам щодо кібербезпеки та надійності. Для криптографічних застосувань, наприклад, можуть бути використані стандарти NIST SP 800-90A/B/C, FIPS 140-2/140-3, ISO/IEC 18031 і ANSI X9.82, які визначають вимоги до криптографічних алгоритмів та методів [19]. Потрібно виконати тестування генератора псевдовипадкових послідовностей на відповідність встановленим стандартам та вимогам. Це може включати виконання криптографічних тестів на стійкість, тестування на випадковість та інші тести, які перевіряють якість та безпеку генерованих чисел. Стандартні набори тестів для перевірки якості випадкових чисел: NIST Statistical Test Suite [8], Diehard tests [22] або TestU01 [19]. Варто використовувати

криптографічні бібліотеки та інструменти, які мають відповідні сертифікати відповідності стандартам кібербезпеки [21]. Наприклад, можна використовувати бібліотеки, які підтримують FIPS 140-2 (Federal Information Processing Standard) для відповідності вимогам до кібербезпеки.

Потрібно періодично переглядати та оновлювати процедури та алгоритми генерації псевдовипадкових чисел для відповідності останнім стандартам та рекомендаціям з кібербезпеки. Технології та вимоги до кібербезпеки постійно розвиваються, тому важливо залишатися в курсі останніх змін. Перевірка на відповідність стандартам дає змогу переконатися, що розроблений генератор відповідає встановленим вимогам та стандартам якості та безпеки. Переваги такої перевірки очевидні. Стандарти встановлюють найважливіші вимоги до криптостійкості псевдовипадкових послідовностей, зокрема стійкість до криптографічних атак та відповідність до криптографічних стандартів [20, 21]. Стандарти часто встановлюють вимоги до випадковості псевдовипадкових чисел, такі як рівномірність розподілу, відсутність кореляцій, та інші. Перевірка дозволяє підтвердити, що генератор відповідає цим вимогам.

Використання генератора, що відповідає стандартам, забезпечує сумісність з іншими системами та програмним забезпеченням, які використовують псевдовипадкові послідовності. Це важливо для інтеграції та обміну даними між різними компонентами і збільшує довіру користувачів та клієнтів до продукту чи послуги. Це особливо важливо в критичних галузях, таких як фінансові та медичні системи. Перевірка на відповідність стандартам допомагає забезпечити якість генератора псевдовипадкових чисел шляхом встановлення чітких вимог до його реалізації та функціонування. Багато країн і галузей вимагають використання сертифікованих криптографічних методів. Відповідність стандартам допомагає уникнути юридичних та регуляторних проблем, що можуть виникнути через використання небезпечних алгоритмів.

Отже, перевірка на відповідність стандартам допомагає забезпечити безпеку, випадковість та якість генерації псевдовипадкових послідовностей, що робить її важливим етапом у процесі розробки та використання таких систем.

Моніторинг та аудит системи генерації. Постійний моніторинг та аудит процесу генерації псевдовипадкових послідовностей допоможуть виявити можливі проблеми та вразливості, що можуть знизити якість випадковості. Ці процедури є важливими етапами для забезпечення кібербезпеки та надійності процесу генерації псевдовипадкових послідовностей [5]. Для ефективного проведення моніторингу та аудиту потрібно визначити ключові метрики безпеки, які будуть використовуватися для оцінки ефективності системи генерації псевдовипадкових послідовностей [5, 6]. Це може включати кількість згенерованих чисел, рівень ентропії, результати криптографічних тестів на стійкість, а також виявлення можливих аномалій або вразливостей. Варто налаштувати систему моніторингу для відстеження важливих параметрів та подій, пов'язаних з генерацією псевдовипадкових послідовностей. Це може включати відстеження діяльності генератора чисел, збереження журналів подій, а також виявлення непередбачених змін чи аномалій.

Потрібно реалізувати процес аудиту кібербезпеки для періодичної перевірки та оцінки безпеки системи генерації псевдовипадкових послідовностей. Це може включати регулярне проведення аудиту коду, аналіз результатів криптографічних тестів, а також перевірку відповідності встановленим стандартам та рекомендаціям безпеки. Необхідно розробити механізми виявлення та реагування на потенційні аномалії чи вразливості у системі генерації псевдовипадкових послідовностей. Це може включати автоматизовані системи виявлення вторгнень, моніторинг системи на виявлення несподіваних змін, а також розробку процедур реагування на інциденти безпеки.

Важливим моментом є проведення навчання для персоналу, який відповідає за моніторинг та аудит системи генерації псевдовипадкових послідовностей, з використанням інструментів моніторингу та аудиту, а також процедур кібербезпеки та виявлення вразливостей. Варто постійно вдосконалювати процеси моніторингу та аудиту на основі отриманих даних та результатів [4, 24]. Потрібно використовувати зворотний зв'язок для удосконалення системи генерації псевдовипадкових чисел та підвищення рівня безпеки.

Моніторинг та аудит системи генерації псевдовипадкових чисел є важливим процесом для забезпечення безпеки, якості та надійності цих послідовностей. Перевагами цього процесу є можливість виявляти незвичайні та небажані патерни у генерації псевдовипадкових послідовностей, що можуть свідчити про проблеми в алгоритмах чи вихідних джерелах випадковості. Аудит системи генерації дає змогу виявити можливі вразливості, які можуть бути використані зловмисниками для атак на систему [9]. Це допомагає у запобіганні можливим порушенням безпеки.

Моніторинг та аудит дає змогу переконатися, що система генерації псевдовипадкових чисел відповідає вимогам стандартів безпеки та якості, таким як FIPS, NIST SP 800-90B/C, ISO/IEC 27001 тощо [9]. Шляхом аналізу результатів моніторингу та аудиту можна виявити можливі проблеми з якістю генерації псевдовипадкових чисел та вжити заходів для їх виправлення. Регулярний моніторинг та аудит демонструють зобов'язання компанії чи організації до забезпечення безпеки та якості генерації псевдовипадкових послідовностей, що підвищує довіру клієнтів, користувачів та регуляторів.

Моніторинг дає змогу вчасно виявляти можливі інциденти або порушення безпеки у генерації псевдовипадкових послідовностей та приймати відповідні заходи для їх усунення та запобігання можливим інцидентам. Отже, моніторинг та аудит системи генерації псевдовипадкових послідовностей є важливими інструментами для забезпечення безпеки, якості та надійності цих послідовностей у різних застосунках.

Оптимізація швидкості та ефективності. Використання оптимізованих алгоритмів та програмного забезпечення може покращити швидкість генерації псевдовипадкових послідовностей та ефективність використання ресурсів. Оптимізація є важливим завданням, особливо у тих застосунках, де велика кількість випадкових чисел генерується швидко та ефективно. Стратегія для покращення цього процесу потребує використання швидких та ефективних алгоритмів генерації псевдовипадкових послідовностей, які мають низьку обчислювальну складність та високу швидкість [7]. Варто використовувати можливості апаратного прискорення, якщо вони доступні, для генерації псевдовипадкових послідовностей. Багато сучасних процесорів з підтримкою SIMD (Single Instruction, Multiple Data) та мікроконтролерів мають вбудовані інструкції або модулі для випадкового генерування. Потрібно використовувати багатопотоковість та паралелізацію (GPU для паралельної обробки) для розпаралелювання обчислень та прискорення генерації псевдовипадкових чисел [12]. Це може бути особливо корисним на багатоядерних або розподілених системах.

Для зменшення використання ресурсів при генерації псевдовипадкових послідовностей необхідно реалізувати зменшення кількості звернень до пам'яті або оптимізації алгоритмів генерації, а також використовувати кешування та попереднє обчислення, де це можливо, для збереження результатів попередніх генерацій та уникнення повторного обчислення [25]. Варто максимально оптимізувати код генератора псевдовипадкових послідовностей для підвищення швидкості та ефективності. Це може включати використання вбудованих функцій та оптимізацію циклів та інших конструкцій коду.

Потрібно періодично тестувати швидкість генератора та виконувати оптимізації для підвищення швидкості та ефективності [26]. Збільшення швидкості генерації псевдовипадкових послідовностей дає можливість системі обробляти більше запитів чи операцій за один часовий період [25, 26]. Це особливо важливо для високонавантажених систем, де швидкість має велике значення. Швидший процес генерації псевдовипадкових чисел дає змогу зменшити затримки та очікування в системі. Це може покращити відгук системи та загальне користувацьке враження. Збільшення ефективності генерації псевдовипадкових чисел дозволяє системі легше масштабуватися з ростом навантаження [12]. Це робить систему більш гнучкою та готовою до зростання обсягів даних та користувацьких запитів.

Оптимізація швидкості та ефективності дає змогу економити ресурси, такі як процесорний час, енергія та обсяг пам'яті. Це особливо важливо для систем з обмеженими ресурсами, такими як мобільні пристрої або вбудовані системи. Таким чином, оптимізація швидкості та ефективності генерації псевдовипадкових послідовностей є важливим кроком для покращення продуктивності, ефективності та користувацького досвіду системи.

Обговорення результатів досліджень. Результати дослідження покращення процесу генерування псевдовипадкових послідовностей за допомогою використання кращих алгоритмів відображені у таблиці 1. Було проаналізовано основні характеристики чотирьох відомих алгоритмів, які використовуються при розробці генераторів.

Таблиця 1

Основні характеристики алгоритмів

Алгоритм	Основні характеристики
Fortuna	1. Криптостійкий. 2. Багаторівнева архітектура. 3. Використовуються генератори підвищення рівня випадковості. 4. Надійний та безпечний. 5. Застосовується механізм переривчастого використання генераторів.

Продовження таблиці 1

Алгоритм	Основні характеристики
Yarrow	1. Криптостійкий. 2. Використовується ентропія з різних джерел. 3. Адаптивний. 4. Застосовується механізм переходу у реактивний режим. 5. Надійний та безпечний. 6. Стандартизований.
Blum Blum Shub (BBS)	1. Криптостійкий. 2. Простий у реалізації. 3. Параметризований. 4. Низька швидкість генерації. 5. Безпека ґрунтується на основі задачі факторизації. 6. Може мати дуже велику періодичність.
ISAAC (Innovative Synchronous-algorithm and Asynchronous-algorithm based on Cryptography)	1. Криптостійкий. 2. Забезпечує високу якість випадковості. 3. Відомий своєю високою швидкістю. 4. Простий у реалізації. 5. Може мати дуже велику періодичність.

Результати дослідження дали змогу сформулювати методологію покращення процесу генерування псевдовипадкових послідовностей як послідовність таких методів: вибір оптимальних початкових значень, перевірка на відповідність стандартам, моніторинг та аудит системи генерації, оптимізація швидкості та ефективності. У таблиці 2 наведено стратегії застосування і переваги кожного методу для покращення процесу генерування.

Таблиця 2

Методологія покращення: стратегії і переваги

Метод покращення	Стратегія	Переваги
Вибір оптимальних початкових значень	1. Забезпечення високої ентропії. 2. Збір ентропії з різних джерел. 3. Захист від прогнозування початкових значень. 4. Регулярне оновлення початкових значень. 5. Вибір відповідного розміру початкових значень.	1. Захист від атак. 2. Якість випадкових послідовностей. 3. Унікальність послідовностей (уникнення повторів). 4. Стійкість до криптоаналізу. 5. Запобігання прогнозуванню. 6. Ефективне використання ресурсів. 7. Асинхронність в системах з багатьма генераторами.
Перевірка на відповідність стандартам	1. Вибір відповідних стандартів. 2. Використання криптостійких алгоритмів. 3. Тестування на відповідність стандартам. 4. Використання сертифікованих бібліотек та інструментів. 5. Оновлення та перевірка стандартів.	1. Забезпечення кібербезпеки. 2. Гарантія випадковості. 3. Забезпечення сумісності. 4. Довіра користувачів. 5. Забезпечення якості. 6. Захист від юридичних та регуляторних ризиків.
Моніторинг та аудит системи генерації	1. Визначення метрик безпеки. 2. Налаштування системи моніторингу. 3. Реалізація аудиту безпеки. 4. Виявлення та реагування на аномалії. 5. Навчання персоналу. 6. Постійне вдосконалення.	1. Виявлення аномалій. 2. Попередження про вразливості. 3. Забезпечення відповідності. 4. Покращення якості генерації. 5. Підвищення довіри. 6. Реагування на інциденти.
Оптимізація швидкості та ефективності	1. Використання оптимізованих алгоритмів. 2. Використання апаратного прискорення. 3. Багатопотоковість та паралелізація. 4. Зменшення накладних витрат. 5. Кешування та попереднє обчислення. 6. Оптимізація коду. 7. Тестування та оптимізація швидкості.	1. Підвищення продуктивності. 2. Зменшення затримок. 3. Підвищення масштабованості. 4. Економія ресурсів.

Важливо також враховувати пріоритети різних методів (таблиця 3) для їх комбінації та послідовності застосування.

Таблиця 3

Пріоритетність методів

№ з/п	Метод покращення	Пріоритет
1	Виконання моніторингу та аудиту.	Високий
2	Налаштування початкових значень.	Високий
3	Перевірка системи на відповідність стандартам.	Середній
4	Виконання моніторингу та аудиту.	Середній
5	На завершальному етапі – оптимізація швидкості та ефективності.	Низький

Висновки. Генерація псевдовипадкових послідовностей є важливою складовою кібербезпеки, а також багатьох комп'ютерних систем та програм, що використовують випадковість для різних цілей, від криптографії до симуляцій. Одна з ключових проблем – недостатня якість та безпека псевдовипадкових послідовностей, яка може призвести до вразливості систем та недостатньої стійкості криптографічних протоколів. Інші проблеми включають недостатню швидкість генерації, обмеженість доступних ресурсів, а також недостатню випадковість у деяких сценаріях використання. Було запропоновано методологію покращення процесу генерування псевдовипадкових послідовностей, яка включає в себе використання кращих алгоритмів, вибір оптимальних початкових значень, перевірку на відповідність стандартам, моніторинг та аудит системи генерації, оптимізацію швидкості та ефективності. Ці підходи дозволяють забезпечити високу якість та надійність псевдовипадкових послідовностей у різних сценаріях використання.

Загальний висновок полягає в тому, що комбінація різних методів може бути найкращим підходом для розробки системи. Важливо також враховувати пріоритети запропонованих методів для їх комбінації та послідовності застосування. Це забезпечить надійний процес генерування псевдовипадкових послідовностей.

Список літератури:

1. Muhammed, A. J., Woldiegiworgies, T. A., Tsegaye, G. G. (2024). Security Enhancement of Playfair Cipher Using Modified Blum Blum Shub Algorithm and Keystream Values. https://doi.org/10.21203/rs.3.rs-3847682/v1

2. Ali Akbar, M., Zulkifl Khalid, M. (2008). Fuzzy-Fortuna: A fuzzified approach to generation of cryptographically secure pseudo-random numbers. 2008 IEEE International Multitopic Conference, 213–217. doi.org/10.1109/INMIC.2008.4777738

3. Crocetti, L., Nannipieri, P., Di Matteo, S., Fanucci, L., Saponara, S. (2023). Review of Methodologies and Metrics for Assessing the Quality of Random Number Generators. Electronics, 12(3), 723. https://doi.org/10.3390/electronics12030723

4. Bansal, A., Kandikuppa, A., Hasan, M., Chen, C.-Y., Bates, A., Mohan, S. (2023). System Auditing for Real-Time Systems. ACM Transactions on Privacy and Security, 26(4), 1–37. https://doi.org/10.1145/3625229

5. Xiang, W., Zhao, J., Huang, H., Zhang, X., Jiang, Z. L., He, D. (2024). Blockchain-Assisted Privacy-Preserving Public Auditing Scheme for Cloud Storage Systems. In Z. Tari, K. Li, H. Wu (Eds.), Algorithms and Architectures for Parallel Processing (Vol. 14488, pp. 292–310). Springer Nature Singapore. https://doi.org/10.1007/978-981-97-0801-7_17

6. Di Pilla, P., Pareschi, R., Salzano, F., Zappone, F. (2023). Listening to what the system tells us: Innovative auditing for distributed systems. Frontiers in Computer Science, 4, 1020946. https://doi.org/10.3389/fcomp.2022.1020946

7. Pushkar, O., Hrabovskyi, Y., Gordyeyev, A. (2020). Development of a method for optimizing the site loading speed. Eastern-European Journal of Enterprise Technologies, 6(2 (108)), 21–29. https://doi.org/10.15587/1729-4061.2020.216993

8. Pikuza, M. O., Mikhnevich, S. Yu. (2021). Testing a hardware random number generator using NIST statistical test suite. Doklady BGUIR, 19(4), 37–42. doi.org/10.35596/1729-7648-2021-19-4-37-42

9. Acar, D., Gal, G., Öztürk, M. S., Usul, H. (2021). A Case Study in the Implementation of a Continuous Monitoring System. Journal of Emerging Technologies in Accounting, 18(1), 17–25. https://doi.org/10.2308/JETA-17-04-29-9

10. Щербина, Ю., Казакова, Н., Фразе-Фразенко, О., Лаптєв, О., Собчук, А. (2023). Вибір джерела випадковості для комп’ютерного моделювання. Science-Based Technologies, 59(3), 233–238. doi.org/10.18372/2310-5461.59.17944

11. Artuğer, F., Özkaynak, F. (2024). A new chaotic system and its practical applications in substitution box and random number generator. Multimedia Tools and Applications. https://doi.org/10.1007/s11042-024-19053-7

12. Долгий, А. (2024, March 1). Оптимізація Випадкової Генерації Roguelike Рівнів. Débats Scientifiques Et Orientations Prospectives Du Développement Scientifique. Débats Scientifiques Et Orientations Prospectives Du Développement Scientifique. doi.org/10.36074/logos-01.03.2024.050

13. Crocetti, L., Di Matteo, S., Nannipieri, P., Fanucci, L., Saponara, S. (2022). Design and Test of an Integrated Random Number Generator with All-Digital Entropy Source. Entropy, 24(2), 139. https://doi.org/10.3390/e24020139

14. Kelsey, J., Schneier, B., Ferguson, N. (2000). Yarrow-160: Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator. In H. Heys C. Adams (Eds.), Selected Areas in Cryptography (Vol. 1758, pp. 13–33). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-46513-8_2

15. R.J. Jenkins, "ISAAC", Fast Software Encryption – Cambridge 1996, vol. 1039,D. Gollmann ed., Springer-Verlag.

16. Osara, J. A., Bryant, M. D. (2024). Methods to Calculate Entropy Generation. Entropy, 26(3), 237. https://doi.org/10.3390/e26030237

17. Долотій, М. Г., Мерзликін, П. В. (2017). Генератор випадкових чисел з апаратним джерелом ентропії. New Computer Technology, 15, 85–88. https://doi.org/10.55056/nocote.v15i0.635

18. Корчинський, В., Рябуха, О., Халед, А.-Ф., Гавель, С., Міненко, В., Криштафор, З. (2023). Дослідження варіаційних можливостей генераторів хаосу по формуванню псевдовипадкових послідовностей. Measuring and computing devices in technological processes, 1, 180–186. doi.org/10.31891/2219-9365-2023-73-1-24

19. Hurley-Smith, D., Patsakis, C., Hernandez-Castro, J. (2022). On the Unbearable Lightness of FIPS 140–2 Randomness Tests. IEEE Transactions on Information Forensics and Security, 17, 3946–3958. https://doi.org/10.1109/TIFS.2020.2988505

20. Kim, Y., Yeom, Y. (2021). Accelerated implementation for testing IID assumption of NIST SP 800-90B using GPU. PeerJ Computer Science, 7, e404. https://doi.org/10.7717/peerj-cs.404

21. Хомік, М., Гарасимчук, О. (2023). Застосування генераторів псевдовипадкових чисел та послідовностей в кібербезпеці, методи їх побудови та оцінки якості. Ukrainian Information Security Research Journal, 25(3), 147–159. https://doi.org/10.18372/2410-7840.25.17940

22. Sriram, V., Srikamakshi, M., Jegadish Kumar, K. J., Nagarajan, K. K. (2020). Randomness Analysis of YUGAM-128 Using Diehard Test Suite. In J. S. Raj, A. Bashar, S. R. J. Ramson (Eds.), Innovative Data Communication Technologies and Application (Vol. 46, pp. 600–607). Springer International Publishing. https://doi.org/10.1007/978-3-030-38040-3_68

23. Hurley-Smith, D., Hernandez-Castro, J. (2020). Quantum Leap and Crash: Searching and Finding Bias in Quantum Random Number Generators. ACM Transactions on Privacy and Security, 23(3), 1–25. doi.org/10.1145/3398726

24. Hong, S., Seo, H., Yoon, M. (2023). Data Auditing for Intelligent Network Security Monitoring. IEEE Communications Magazine, 61(3), 74–79. https://doi.org/10.1109/MCOM.003.2200046

25. Putra, E. (2023). Optimizing Management Information Systems Based on Efficiency and Quality Management. Nidhomul Haq : Jurnal Manajemen Pendidikan Islam, 8(3), 401–411. https://doi.org/10.31538/ndh.v8i3.3938

26. Wang, X., Zheng, T., Jia, Y., Huang, J., Zhu, X., Shi, Y., Wang, N., Lu, Z., Zou, J., Li, Y. (2024). Compact Quantum Random Number Generator Based on a Laser Diode and a Hybrid Chip with Integrated Silicon Photonics. Photonics, 11(5), 468. https://doi.org/10.3390/photonics11050468

References:

1. Muhammed, A. J., Woldiegiworgies, T. A., Tsegaye, G. G. (2024). Security Enhancement of Playfair Cipher Using Modified Blum Blum Shub Algorithm and Keystream Values. https://doi.org/10.21203/rs.3.rs-3847682/v1

2. Ali Akbar, M., Zulkifl Khalid, M. (2008). Fuzzy-Fortuna: A fuzzified approach to generation of cryptographically secure pseudo-random numbers. 2008 IEEE International Multitopic Conference, 213–217. doi.org/10.1109/INMIC.2008.4777738

3. Crocetti, L., Nannipieri, P., Di Matteo, S., Fanucci, L., Saponara, S. (2023). Review of Methodologies and Metrics for Assessing the Quality of Random Number Generators. Electronics, 12(3), 723. https://doi.org/10.3390/electronics12030723

4. Bansal, A., Kandikuppa, A., Hasan, M., Chen, C.-Y., Bates, A., Mohan, S. (2023). System Auditing for Real-Time Systems. ACM Transactions on Privacy and Security, 26(4), 1–37. https://doi.org/10.1145/3625229

5. Xiang, W., Zhao, J., Huang, H., Zhang, X., Jiang, Z. L., He, D. (2024). Blockchain-Assisted Privacy-Preserving Public Auditing Scheme for Cloud Storage Systems. In Z. Tari, K. Li, & H. Wu (Eds.), Algorithms and Architectures for Parallel Processing (Vol. 14488, pp. 292–310). Springer Nature Singapore. https://doi.org/10.1007/978-981-97-0801-7_17

6. Di Pilla, P., Pareschi, R., Salzano, F., Zappone, F. (2023). Listening to what the system tells us: Innovative auditing for distributed systems. Frontiers in Computer Science, 4, 1020946. https://doi.org/10.3389/fcomp.2022.1020946

7. Pushkar, O., Hrabovskyi, Y., Gordyeyev, A. (2020). Development of a method for optimizing the site loading speed. Eastern-European Journal of Enterprise Technologies, 6(2 (108)), 21–29. https://doi.org/10.15587/1729-4061.2020.216993

8. Pikuza, M. O., Mikhnevich, S. Yu. (2021). Testing a hardware random number generator using NIST statistical test suite. Doklady BGUIR, 19(4), 37–42. doi.org/10.35596/1729-7648-2021-19-4-37-42

9. Acar, D., Gal, G., Öztürk, M. S., Usul, H. (2021). A Case Study in the Implementation of a Continuous Monitoring System. Journal of Emerging Technologies in Accounting, 18(1), 17–25. https://doi.org/10.2308/JETA-17-04-29-9

10. Shcherbina, Yu., Kazakova, N., Fraze-Frazenko, O., Laptev, O., Sobchuk, A. (2023). Selection Of Randomness Source For Computer Simulation. Science-Based Technologies, 59(3), 233–238. doi.org/10.18372/2310-5461.59.17944 [in Ukrainian].

11. Artuğer, F., Özkaynak, F. (2024). A new chaotic system and its practical applications in substitution box and random number generator. Multimedia Tools and Applications. https://doi.org/10.1007/s11042-024-19053-7

12. Dolgyi, A. (2024, March 1). Optimization Of Roguelike Random Generation Of Levels. Débats Scientifiques Et Orientations Prospectives Du Développement Scientifique. Débats Scientifiques Et Orientations Prospectives Du Développement Scientifique. https://doi.org/10.36074/logos-01.03.2024.050 [in Ukrainian].

13. Crocetti, L., Di Matteo, S., Nannipieri, P., Fanucci, L., Saponara, S. (2022). Design and Test of an Integrated Random Number Generator with All-Digital Entropy Source. Entropy, 24(2), 139. https://doi.org/10.3390/e24020139

14. Kelsey, J., Schneier, B., Ferguson, N. (2000). Yarrow-160: Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator. In H. Heys & C. Adams (Eds.), Selected Areas in Cryptography (Vol. 1758, pp. 13–33). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-46513-8_2

15. R.J. Jenkins, "ISAAC", Fast Software Encryption – Cambridge 1996, vol. 1039, D. Gollmann ed., Springer-Verlag.

16. Osara, J.A., Bryant, M.D. (2024). Methods to Calculate Entropy Generation. Entropy, 26(3), 237. https://doi.org/10.3390/e26030237

17. Dolotii, M. G., Merzlykin, P. V. (2017). Random number generator with a hardware source of entropy. New Computer Technology, 15, 85–88. doi.org/10.55056/nocote.v15i0.635 [in Ukrainian].

18. Korchynskyi, V., Ryabukha, O., Khaled, A.-F., Havel, S., Minenko, V., Kryshtafor, Z. (2023). Study Of Variation Possibilities Of Chaos Generators For The Formation Of Pseudo-Random Sequences. Measuring And Computing Devices In Technological Processes, 1, 180–186. doi.org/10.31891/2219-9365-2023-73-1-24 [in Ukrainian].

19. Hurley-Smith, D., Patsakis, C., Hernandez-Castro, J. (2022). On the Unbearable Lightness of FIPS 140–2 Randomness Tests. IEEE Transactions on Information Forensics and Security, 17, 3946–3958. https://doi.org/10.1109/TIFS.2020.2988505

20. Kim, Y., Yeom, Y. (2021). Accelerated implementation for testing IID assumption of NIST SP 800-90B using GPU. PeerJ Computer Science, 7, e404. https://doi.org/10.7717/peerj-cs.404

21. Khomik, M., Garasymchuk, O. (2023). 26. Ukrainian Information Security Research Journal, 25(3), 147–159. https://doi.org/10.18372/2410-7840.25.17940 [in Ukrainian].

22. Sriram, V., Srikamakshi, M., Jegadish Kumar, K. J., Nagarajan, K. K. (2020). Randomness Analysis of YUGAM-128 Using Diehard Test Suite. In J. S. Raj, A. Bashar, & S. R. J. Ramson (Eds.), Innovative Data Communication Technologies and Applications (Vol. 46, pp. 600–607). Springer International Publishing. https://doi.org/10.1007/978-3-030-38040-3_68

23. Hurley-Smith, D., Hernandez-Castro, J. (2020). Quantum Leap and Crash: Searching and Finding Bias in Quantum Random Number Generators. ACM Transactions on Privacy and Security, 23(3), 1–25. https://doi.org/10.1145/3398726

24. Hong, S., Seo, H., Yoon, M. (2023). Data Auditing for Intelligent Network Security Monitoring. IEEE Communications Magazine, 61(3), 74–79. https://doi.org/10.1109/MCOM.003.2200046

25. Putra, E. (2023). Optimizing Management Information Systems Based on Efficiency and Quality Management. Nidhomul Haq: Jurnal Manajemen Pendidikan Islam, 8(3), 401–411. https://doi.org/10.31538/ndh.v8i3.3938

26. Wang, X., Zheng, T., Jia, Y., Huang, J., Zhu, X., Shi, Y., Wang, N., Lu, Z., Zou, J., Li, Y. (2024). Compact Quantum Random Number Generator Based on a Laser Diode and a Hybrid Chip with Integrated Silicon Photonics. Photonics, 11(5), 468. https://doi.org/10.3390/photonics11050468

Науково-методична стаття.
Надійшла до редакції 04.11.2024.
Прийнято до публікації 18.12.2024.