

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЮРІЯ ФЕДЬКОВИЧА

Кваліфікаційна наукова
праця на правах рукопису

ДЯЧУК РОСТИСЛАВ ЛЮБОМИРОВИЧ

УДК 004.421.5:004.056.55

ДИСЕРТАЦІЯ

**РОЗРОБКА ТА ДОСЛІДЖЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ
ГЕНЕРУВАННЯ ВИСОКОЕНТРОПІЙНОЇ ПОСЛІДОВНОСТІ
ВИПАДКОВИХ ЧИСЕЛ**

12 – Інформаційні технології

121 – Інженерія програмного забезпечення

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей,
результатів і текстів інших авторів мають посилання на відповідне джерело

 Р. Л. Дячук

**Науковий керівник Добровольський Юрій Георгійович, доктор
технічних наук, професор**

Чернівці – 2026

АНОТАЦІЯ

Дячук Р.Л. Розробка та дослідження інформаційної системи генерування високоентропійної послідовності випадкових чисел. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 121 – «Інженерія програмного забезпечення» – Чернівецький національний університет імені Юрія Федьковича, Чернівці, 2026.

Дисертаційна робота присвячена створенню та дослідженню інформаційної системи для генерування високоентропійної послідовності випадкових чисел (ПВЧ) на основі сучасних технологій та методів програмної інженерії, а також використання фотоелектричних явищ в якості джерела ентропії. Створена інформаційна система і її програмне забезпечення побудована на основі технології екстракції хаосу з стохастичних фізичних явищ, а саме – інтенсивності пікселів зображення з веб-камери, або цифрової відеокамери, фото- та темнового струму фотодіода з високою квантовою ефективністю. Отримані послідовності випадкових чисел також можуть оброблятися за допомогою клітинних автоматів, що суттєво збільшує рівень хаотичності випадкової послідовності. Створені генератори випадкових послідовностей, на відміну від існуючих, є простими, доступними, високопродуктивними, здатними до гнучкого налаштування. Інформаційна система загалом містить аналітичний інтелектуальний модуль спрощеної статистичної оцінки послідовності випадкових чисел, згенерованих вбудованими в неї генераторів, відрізняється високим рівнем криптостійкості, сприяє ефективному шифруванню та кодуванню для мінімізації кіберзагроз.

Проведено порівняльне дослідження статистичних і робочих характеристик аналогічних відомих генераторів ПВЧ, яке показало перевагу створеної інформаційної системи за рядом показників. А саме: за швидкістю генерації послідовності випадкових чисел, високим рівнем недетермінованої

ентропії, гнучким налаштуванням, відкритим дизайном та швидкою пост обробкою. Створена інформаційна система має базові внутрішні перевірки, здатна регулювати співвідношення якості та швидкості вихідного потоку та ефективну пост обробку за допомогою клітинних автоматів. Окрім того новостворена система має відкритість апаратури та програмного забезпечення Open-hardware та open-source (GPLv3) в тому числі для застосування у нейронних мережах.

Також, для роботи з крипто-примітивами на основі клітинних автоматів створено бібліотеку на мові програмування Java. Запропоновано і експериментально доведено високу якість, що генерується клітинними автоматами хаотичного типу (правила 30, 90, 105). Оптимізовано високу продуктивності і низьку ресурсоемності функціоналу на клітинних автоматах. Забезпечено неперервну генерацію послідовності випадкових чисел заданої продуктивності та якості.

Результати роботи є підґрунтям для подальших теоретичних і практичних наукових розробок із досліджень проблематики теорії генерації ПВЧ.

Дисертація складається зі вступу, чотирьох розділів, висновків, переліку використаних джерел та чотирьох додатків. У **вступі** обґрунтовано актуальність теми дослідження, сформульовано мету, завдання, предмет, об'єкт та методи дослідження, вказано наукову новизну, теоретичне та практичне значення отриманих результатів, подано та проаналізовано зв'язок роботи з науковими темами. Зазначено особистий внесок здобувача, а також наведено відомості про апробацію та публікації основних результатів дисертації. Описано структуру та обсяг дисертаційної роботи.

Перший розділ містить ключові відомості з теорії генерації високоентропійних ПВЧ, а також відомості про апаратні генератори ПВЧ в системах інформаційної безпеки, зокрема у криптобезпеці та крипто-кодуванні. Розглянуто різні підходи до джерела ентропії та програмно-

конструкторського оточення. Проаналізовано їх значення у системах на основі криптокодових та шифрувальних конструкцій та показані сильні сторони і вузькі місця з точки зору ціни, складності, надійності, довіри, продуктивності, керованості, моніторингу і ліцензування. Розглянуто теоретичні основи оцінювання випадковості та ентропії, а також проведено порівняльний аналіз сучасних найбільш відомих генераторів ПВЧ в системах криптобезпеки.

Обґрунтовано доцільність дослідження фото-електричних явищ (ФЕЯ) та квантових як джерела ентропії для генерації ПВЧ. Така конструкція дозволить частково зменшити недоліки, які притаманні системам, що вже існують.

На основі проведеного дослідження сформовані завдання дослідження.

У **другому розділі** дисертаційного дослідження описано дослідження інтенсивності пікселів зображення з веб-камери, або цифрової відеокамери, фото- та темного струму фотодіода з високою квантовою ефективністю як природних джерел для генерації послідовності випадкових чисел. Показано, що послідовність випадкових чисел, згенерована на основі інтенсивності пікселів зображення відеокамери має непередбачуваний характер і продуктивність від 85 Мбіт/сек до 288 Мбіт/сек, в залежності роздільної здатності відеокамери. Але, така послідовність випадкових чисел не відповідає вимогам стандартів NIST. Послідовність випадкових чисел, згенерована на основі фотоструму фотодіода з високою квантовою ефективністю є непередбачуваною, а її продуктивність досягає 2 Мбіт/сек. Такі послідовності випадкових чисел, частково відповідають тестам NIST. Послідовності випадкових чисел, створені за допомогою генератора на основі темного струму фотодіода, має абсолютно непередбачуваний характер і продуктивність до 980 Мбіт/сек, яка може бути збільшена при застосування високочастотної апаратної частини. Такий генератор може використовуватись для заправки (вектора ініціалізації) гібридних генераторів випадкових послідовностей.

Проведено дослідження гібридного генератора послідовностей випадкових чисел, який поєднує у собі генератор на основі відео, або веб-камери і клітинних автоматів. Таке поєднання вирішує питання по швидкодії, надійності, контрольованості, відповідності статистичним вимогам та вимогам до безпеки і стабільності пост-обробки. Також програмна частина гібридного генератора відкриває теоретично необмежені можливості по масштабованості та паралелізації процесу генерації, а також машинного навчання за допомогою нейронної мережи.

Третій розділ присвячено створенню інформаційної системи для генерації послідовності випадкових чисел на основі усіх трьох генераторів із обробкою отриманих послідовності випадкових чисел за допомогою клітинних автоматів. Таке об'єднання повністю вирішуватиме питання про непередбачуваність, хаотичність, криптостійкість та ізолюваність згенерованих послідовностей. Інтерфейс інформаційної системи дозволяє керувати параметрами генерації числових послідовностей, а саме: обирати джерело послідовностей випадкових чисел (база даних, файл, на основі відео-, або веб-камери, фото- та темного струму фотодіода з високою квантовою ефективністю, гібридного з обробкою числових послідовностей за допомогою клітинних автоматів); обирати між різними діапазонами значень (біти 0-1, стандартний діапазон 1-256, або користувацький діапазон); регулювати баланс між швидкістю генерації та рівнем безпеки, який забезпечує згенерована послідовність випадкових чисел, зберігати її та аналізувати.

Продуктивність інформаційної системи лежить у межах 0,980 - 288 Мбіт/с і може бути суттєво збільшена при використанні високочастотної апаратної частини. Також система передбачає можливість масштабувати всю програмну частину генератора, або ж розпаралелити процес обробки «сирої» числової послідовності всередині блоку програмної частини. Що, наприкладі послідовності випадкових чисел, згенерованої на основі відео-зображення, або веб-камери, дозволило

підвищити продуктивність у 4 рази.

У **четвертому розділі** висвітлено результати застосування розробленої інформаційної системи при формування вхідного вектора для хеш-функції, для стеганографічних систем із підвищеними вимогами до прихованості та стійкості, а також для формування криптографічних ключів із високим рівнем ентропії та криптостійкості. Результати роботи, а саме удосконалений алгоритм створення хеш-функції, стеганографічні алгоритми та алгоритми генерації криптографічних ключів за допомогою розробленої інформаційної системи генерації послідовностей випадкових чисел, впроваджені та використовується у ТДВ ЗАВОД «Кварц», Kaskad Developers Group та компанії «Datawiz» для підвищення криптостійкості програмного забезпечення.

У **висновках** підсумовано основні результати дисертаційного дослідження.

У **додатках** подано наукові публікації, в яких відображено основні наукові результати роботи, відомості про апробацію результатів дисертації, акти та довідки про впровадження результатів роботи, лістинг частини коду програми.

Теоретичне значення. Результати теоретичних досліджень, а саме методологію екстракції хаосу з таких стохастичних фізичних явищ як інтенсивність пікселів з зображення веб-камери, фото- та темновий струм фотодіода, а також методологія використання клітинних автоматів для забезпечення високої якості хаотичності та оптимізація високої продуктивності і низької ресурсоємності функціоналу можуть застосовуватись для досліджень у галузі програмної інженерії, а також у навчальних курсах кафедри програмного забезпечення комп'ютерних систем Чернівецького національного університету імені Юрія Федьковича (та інших ЗВО), пов'язаних з криптозахистом, методичних розробках, навчальних посібниках, які застосовуються в освітньому процесі і науково-дослідної роботи студентів та аспірантів.

Практичне значення. Розроблені у дисертаційній роботі інформаційна

система для генерації послідовностей випадкових чисел може в подальшому використовуватися для практичної реалізації. Запропоновані підходи до генерації вхідного вектора хеш-функцій із використанням клітинних автоматів та послідовностей випадкових чисел, стеганографічні алгоритми та алгоритми генерації криптографічних ключів, використовуються у компанії ТДВ ЗАВОД «Кварц», Kaskad Developers Group та компанії «Datawiz» – для розробки власного програмного забезпечення підвищеної криптостійкості.

Ключові слова: програмне забезпечення, випадкові числа, джерела ентропії, хаотична система, кібербезпека, крипто-кодування, інформаційна система, шифрування, генератор, відеокамера, зображення, фотодіод, квантова ефективність, хеш-функція, нейромережа.

ABSTRACT

Diachuk R. Information System for Generating a High-Entropy Random Number Sequence. – Qualification scientific work as a manuscript.

Thesis for the degree of Doctor of Philosophy in the Specialty 121 – “Software Engineering” – Yuriy Fedkovych Chernivtsi National University, Chernivtsi, 2026.

The proceeding is devoted to the development and investigation of an information system for generating high-entropy random number sequences (RNS) based on modern technologies and software engineering methods, as well as to the use of photoelectric phenomena as entropy sources. The developed information system and its software are based on the technology of chaos extraction from stochastic physical phenomena, namely the intensity of image pixels obtained from a webcam or digital video camera, as well as the photo-current and dark current of a photodiode with high quantum efficiency. The resulting random number sequences can additionally be processed using cellular automata, which significantly increases the level of chaoticity of the generated sequences. The developed random sequence

generators, in contrast to existing solutions, are characterized by simplicity, accessibility, high performance, and flexible configurability. The information system incorporates an analytical intelligent module for simplified statistical evaluation of random sequences generated by its built-in generators. It demonstrates a high level of cryptographic strength, facilitates effective encryption and encoding, and contributes to minimizing cybersecurity threats.

A comparative study of statistical and operational characteristics of known random number generators has been held, demonstrating the advantages of the proposed information system across several indicators. In particular, these include higher generation speed, a high level of non-deterministic entropy, flexible configuration, open design, and efficient post-processing. The developed system also includes basic internal validation mechanisms and provides the capability to regulate the balance between output quality and generation speed of the output stream and efficient post-processing using cellular automata. Furthermore, the newly developed system is characterized by open hardware and open-source software (GPLv3), including its applicability in neural network systems.

In addition, a Java-based library has been developed for working with cryptographic primitives based on cellular automata. A high quality of sequences generated by chaotic cellular automata (Rules 30, 90, and 105) has been proposed and experimentally validated. The functionality of cellular automata has been optimized to ensure high performance and low computational resource consumption. Continuous generation of random number sequences with specified performance and quality parameters has been achieved.

The results of this work provide a foundation for further theoretical and applied research in the field of random number generation theory.

The dissertation consists of an introduction, four chapters, conclusions, a list of references, and four appendices. The **introduction** substantiates the relevance of the research topic, formulates the aim, objectives, subject, object, and research methods, and highlights the scientific novelty as well as the theoretical and practical

significance of the obtained results. It also presents and analyzes the relationship of the work with ongoing research topics. The personal contribution of the author is specified, along with information on the approbation and publications of the main dissertation results. The structure and scope of the dissertation are provided.

The first section contains key information on the theory of high-entropy random number generation, as well as an overview of hardware random number generators in information security systems, particularly in cryptographic security and cryptographic encoding. Various approaches to entropy sources and software–hardware environments were investigated. Their role in systems based on cryptographic coding and encryption schemes is analyzed, highlighting strengths and limitations in terms of cost, complexity, reliability, trust, performance, controllability, monitoring, and licensing. The theoretical foundations of randomness and entropy evaluation were investigated, and a comparative analysis of the most well-known modern random number generators used in cryptographic security systems has been held.

The feasibility of investigating photoelectric phenomena (PhEPh), as well as quantum phenomena, as entropy sources for random number generation has been substantiated. Such an approach makes it possible to partially decrease the shortcomings inherent in existing systems. Based on the research carried out, the set of main tasks have been formulated.

The second chapter of the dissertation presents an experimental study of pixel intensity from webcam or digital video camera images, as well as the photo-current and dark current of a high-quantum-efficiency photodiode, as natural entropy sources for generating random number sequences. It is demonstrated that sequences generated from video camera pixel intensity exhibit an unpredictable nature and achieve throughput ranging from 85 Mbit/s to 288 Mbit/s, depending on camera resolution. However, such sequences do not fully comply with National Institute of Standards and Technology (NIST) standards. Sequences generated from the photo-current of a high-quantum-efficiency photodiode are also unpredictable, with performance

reaching up to 2 Mbit/s. These sequences partially satisfy NIST statistical tests. Random number sequences produced by a generator based on the photodiode dark current exhibit a completely unpredictable nature and achieve performance up to 980 Kbit/s, which can be further increased through the use of high-frequency hardware components. Such a generator can be effectively used for seeding (initialization vector generation) in hybrid random number generators. A hybrid random number generator combining a video/webcam-based generator with cellular automata has also been investigated. This combination addresses key challenges related to performance, reliability, controllability, compliance with statistical requirements, and security, as well as ensuring stable post-processing. Furthermore, the software component of the hybrid generator provides theoretically unlimited performance due to scalability and parallelization of the generation process, as well as the application of machine learning using neural networks.

The third chapter is devoted to the development of an information system for generating random number sequences based on all three generators, with post-processing of the obtained sequences using cellular automata. Such integration comprehensively addresses the issues of unpredictability, chaoticity, cryptographic strength, and isolation of the generated sequences. The interface of the information system enables control over generation parameters, including: selection of the random number source (database, file, video or webcam-based source, photodiode photo-current and dark current with high quantum efficiency, or a hybrid generator with cellular automata-based post-processing); selection of value ranges (binary range 0–1, standard range 1–256, or a user-defined range); adjustment of the balance between generation rate (performance) and security level of the produced sequence; as well as storage and analysis of the generated data. The performance of the information system ranges from 0.980 to 288 Mbit/s and can be significantly increased through the use of high-frequency hardware components. The system also supports scalability of the entire software component of the generator, as well as parallelization of the processing of “raw” numerical sequences within the software

module. For instance, in the case of sequences generated from video imagery or a webcam, this approach enabled a fourfold increase in performance.

The fourth chapter presents the results of applying the developed information system for forming input vectors for hash functions, for steganographic systems with enhanced requirements for concealment and robustness, and for generating cryptographic keys with a high level of entropy and cryptographic strength. The results of the study, including an improved hash function construction algorithm, steganographic algorithms, and cryptographic key generation algorithms based on the developed information system, demonstrate the generation of random number sequences have been implemented and are being used at a plant Kvarts, Kaskad Developers Group, and Datawiz to enhance the cryptographic strength of software systems.

The **Conclusions** summarize the main results of the dissertation research.

The **Appendices** include scientific publications reflecting the principal research findings, information on the approbation of the dissertation results, certificates and reports on the implementation of the research outcomes, and listings of parts of the program code.

Theoretical significance. The results of the theoretical research — namely, the methodology for extracting chaos from stochastic physical phenomena such as webcam image pixel intensity, photodiode photo-current and dark current, as well as the methodology for applying cellular automata to ensure a high level of chaoticity while optimizing performance and minimizing resource consumption—can be applied in further research within the field of software engineering. They may also be utilized in academic courses of the Department of Software Engineering at Yurii Fedkovych Chernivtsi National University (and other higher education institutions), particularly in areas related to cryptographic protection, as well as in methodological materials and teaching manuals used in the educational process and research activities of students and postgraduate researchers.

Practical significance. The information system for generating random number sequences developed in this dissertation can be further employed for practical implementation. The proposed approaches to generating input vectors for hash functions using cellular automata and random number sequences, as well as the developed steganographic algorithms and cryptographic key generation algorithms, are being utilized at ZAVOD Kvarts, Kaskad Developers Group, and Datawiz for the development of proprietary software systems with enhanced cryptographic strength.

Keywords: software, random numbers, entropy sources, chaotic system, cybersecurity, cryptographic encoding, information system, encryption, generator, video camera, image, photodiode, quantum efficiency, hash function, neural network.

СПИСОК ПУБЛІКАЦІЙ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці, в яких опубліковані основні наукові результати дисертації

Наукові праці у періодичних наукових виданнях, проіндексованих у наукометричних базах даних Scopus:

1. Diachuk R., Dmytrashchuk K., Mazurets A., Prokhorov H., Yanushevskyi S. Photosensitive matrix as a source of entropy. *Proceedings of SPIE*. 13813, Seventeenth International Conference on Correlation Optics. 2025. 138132C. ISSN:0277-786X (Scopus) DOI: 10.1117/12.3092587 (*Внесок авторів: Diachuk R. – програмна реалізація збору та обробки даних, експериментальні дослідження ентропійних характеристик, аналіз результатів, підготовка початкового варіанту статті; Dmytrashchuk K. – апаратна реалізація та налаштування фоточутливої матриці, збір експериментальних даних; Mazurets A. – аналіз літературних джерел, статистична обробка результатів; Prokhorov H. – методологія дослідження, формалізація моделі джерела ентропії; Yanushevskyi S. – концептуалізація, наукове керівництво, рецензування та редагування рукопису.*)

Наукові праці у виданнях, включених до переліку

наукових фахових видань України:

2. Diachuk R., Dobrovolsky Y., Hanzhelo D., Prokhorov H., Trembach D. Research the Level of Chaotic and Reliability in Webcam-generated Random Number Sequences, *Security of infocommunication systems and Internet of things*. 2024. Vol. 2, №. 1. P. 01004. DOI: 10.31861/sisiot2024.1.01004 (*Внесок авторів: Dobrovolsky Y. – концептуалізація; Prokhorov H. – методологія та дослідження, Diachuk R. – програмна розробка, та експерименти; Hanzhelo D. – статистичні дослідження; Trembach D. – концептуалізація та написання*).
3. Добровольський, Ю., Дячук, Р. Дослідження зворотного струму фотодіода для генерації надійної випадкової послідовності чисел. *Вісник Хмельницького національного університету. Серія: Технічні науки*. 2025. Том 357, № 5.1. С. 126-132. DOI: 10.31891/2307-5732-2025-357-15 (*Внесок авторів: Добровольський Ю. – концептуалізація, методологія, аналіз результатів, редагування; Дячук Р. – аналіз джерел, програмна розробка, дослідження результатів та написання*).
4. Prokhorov H., Hanzhelo M., Diachuk R., Yanushevskyi S. Investigation of statistical characteristics of random number sequences generated by a webcam using cellular automata functionality and NIST patterns. *Security of infocommunication systems and Internet of things*. 2025. Vol 3, № 1. P. 01007. DOI: 10.31861/sisiot2025.1.01007 (*Внесок авторів: Prokhorov H. – концептуалізація, дослідження, нагляд; Hanzhelo M., Diachuk R. – програмне забезпечення, ресурси, підготовка початкового варіанту рукопису, візуалізація, валідація; Yanushevskyi S. – методологія, концептуалізація, рецензування та редагування рукопису, нагляд, дослідження*).
5. Добровольський Ю., Дячук Р. Дослідження генерації випадкової послідовності чисел на основі темного струм фотодіода при різних температурах. *Вчені записки Таврійського національного університету імені*

В.І. Вернадського. Серія: Технічні науки. 2025. Том 36 (75), № 4. Частина 2. С. 119-126. DOI: 10.32782/2663-5941/2025.4.2/16 (Внесок авторів: Добровольський Ю. – концептуалізація, методологія, написання; Дячук Р. – аналіз джерел, програмна розробка, аналіз результатів та редагування).

Наукові праці, які засвідчують апробацію матеріалів дисертації:

6. Дячук Р.Л., Павлюченко О.С., Прохоров П.А., Прохоров Г.В., Добровольський Ю.Г. Клітинні автомати як генератори Хаосу в криптографічних алгоритмах. Проблеми інформатики та комп'ютерної техніки : матеріали XI міжнар. наук.-практ. конф., м. Чернівці, 10–13 лист. 2022 р. Черн. нац. ун-т ім. Ю Федьковича, Чернівці, 2022. – С. 44-46. *(Внесок авторів: Дячук Р.Л. – концептуалізація, програмна реалізація моделі, експериментальні дослідження, написання рукопису; Павлюченко О.С. – аналіз літератури, дослідження параметрів клітинних автоматів; Прохоров П.А., Прохоров Г.В. – методологія, формалізація моделі, валідація результатів; Добровольський Ю.Г. – наукове керівництво, узагальнення результатів, редагування).*
7. Prokhorov G., Dobrovolsky Y., Dyachuk R. Hash-Function Algorithms Balanced on Reliability and Data Processing Speed. IEEE 4th International Conference on Advanced Trends in Information Theory, 15-16 December, 2022, Kyiv, UKRAINE. P.111-114. (Scopus). DOI: 10.1109/ATIT58178.2022.10024181 *(Внесок авторів: Prokhorov G. – концептуалізація, методологія, постановка задачі; Dobrovolsky Y. – аналіз надійності, редагування, наукове керівництво; Dyachuk R. – програмна реалізація алгоритмів, експериментальні дослідження швидкодії, візуалізація результатів, підготовка початкового варіанту статті).*
8. Дячук Р.Л., Добровольський Ю.Г. Алгоритм хеш-функції з підвищеною криптостійкістю. Покращення лавинного ефекту. Інформаційні технології: наука, техніка, технологія, освіта, здоров'я: тези доповідей XXXI міжнар. наук.-практ. конф., MicroCAD-2023, 17-20 травня 2023. Харків: НТУ «ХПІ».

- С. 1113. (*Внесок авторів: Дячук Р.Л. – розробка алгоритму, програмна реалізація, експериментальна перевірка лавинного ефекту, написання рукопису; Добровольський Ю.Г. – концептуалізація, аналіз криптостійкості, редагування, наукове керівництво*).
9. Дячук Р.Л., Комісарчук В.В., Прохоров П.А., Прохоров Г.В., Добровольський Ю.Г. Вдосконалення функції стискання хеш-функції з застосуванням клітинних автоматів. Проблеми інформатики та комп'ютерної техніки: матеріали XII міжнар. наук.-практ. конф., м. Чернівці, 10–12 лист. 2023. Чернівці: Черн. нац. ун-т, 2023. - С. 82-83. (*Внесок авторів: Дячук Р.Л. – програмна модифікація функції стискання, експериментальні дослідження; Комісарчук В.В. – аналіз літератури, підготовка теоретичного обґрунтування; Прохоров П.А., Прохоров Г.В. – методологія, математичне моделювання, валідація; Добровольський Ю.Г. – концептуалізація, узагальнення результатів, редагування*).
10. Добровольський Ю.Г, Прохоров Г.В., Янушевський С.В., Прохоров П.А, Дячук Р.Л., Трембач Д. В. Оцінка надійності програмного забезпечення з точки зору його довговічності. Проблеми інформатики та комп'ютерної техніки: матеріали XII міжнар. наук.-практ. конф., м. Чернівці, 01–03 лист. 2024. Чернівці: Черн. нац. ун-т, С. 36-37. (*Внесок авторів: Добровольський Ю.Г. – концептуалізація, методологія оцінювання, редагування; Прохоров Г.В. – математична модель довговічності, аналіз результатів; Янушевський С.В. – статистична обробка даних, валідація; Прохоров П.А. – аналіз показників надійності; Дячук Р.Л. – програмна реалізація моделі, експериментальні дослідження, візуалізація; Трембач Д.В. – збір даних, підготовка матеріалів*).
11. Добровольський Ю.Г., Прохоров Г.В., Дячук Р.Л. Генерація послідовності випадкових чисел на основі темнового струму фотодіода. Current trends in scientific research development. Proceedings of the 13th International scientific and practical conference. BoScience Publisher. Boston, USA. 2025. P. 65-67.

(Внесок авторів: Добровольський Ю.Г. – концептуалізація, постановка задачі, редагування; Прохоров Г.В. – методологія експерименту, аналіз фізичних параметрів; Дячук Р.Л. – програмна обробка сигналу, реалізація генератора, статистичне тестування випадковості, написання рукопису).

12. Prokhorov G.V., Diachuk R.L., Hanzhelo M.G., Yanushevskiy S.V. New approaches to the design of a hybrid random sequence generator. Physical and technological problems of transmission, processing and storage of information in infocommunication systems: Proceedings of Xth International Scientific-Practical Conference. Chernivtsi : Yuriy Fedkovych Chernivtsi National University, 2025. P. 134-136. *(Внесок авторів: Prokhorov G.V. – концептуалізація, методологія гібридної моделі; Diachuk R.L. – програмна реалізація генератора, експериментальні дослідження, візуалізація результатів, підготовка початкового варіанту статті; Hanzhelo M.G. – аналіз існуючих підходів, валідація; Yanushevskiy S.V. – рецензування та редагування, наукове керівництво).*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	21
ВСТУП	23
РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ГЕНЕРАЦІЇ ВИСОКОЕНТРОПІЙНОЇ ПОСЛІДОВНОСТІ ВИПАДКОВИХ ЧИСЕЛ	28
1.1. Апаратні генератори випадкових чисел в системах інформаційної безпеки	29
1.1.1 Роль апаратних генераторів випадкових чисел у криптографічних системах.....	29
1.1.2. Теоретичні основи оцінювання випадковості та ентропії	30
1.1.3. Класифікація сучасних апаратних генераторів випадкових чисел	31
1.1.4. Технічний огляд і порівняльний аналіз сучасних апаратних генераторів	32
1.1.4.1. Квантові генератори ID Quantique (Quantis QRNG)	32
1.1.4.2. Високошвидкісні USB-TRNG: SwiftRNG Pro та TrueRNGpro	32
1.1.4.3. Open-hardware рішення: ChaosKey, OneRNG, NeuG	33
1.1.4.4. TRNG на основі дробового шуму: Comscire PQ4000KS	34
1.1.4.5. Вбудовані RNG у захищених елементах та HSM: ATECC608B, nShield / Luna	35
1.1.4.6. Узагальнення щодо характеристик найбільш відомих апаратних генераторів випадкових чисел в системах інформаційної безпеки.....	41
1.2. Генератори ПВЧ на основі фотоприймальної техніки	42
Висновки до розділу 1.....	44
РОЗДІЛ 2. ДОСЛІДЖЕННЯ ВЕБ-КАМЕРИ ФОТО- ТА ТЕМНОВОГО СТРУМІВ ФОТОДІОДА В ЯКОСТІ	

ДЖЕРЕЛА ДЛЯ ГЕНЕРАЦІЇ ПВЧ	46
2.1. Дослідження генерації ПВЧ за допомогою веб-камери	46
2.2. Фотострум фотодіода як джерело ПВЧ	53
2.3. Темновий струм фотодіода як джерело ПВЧ	59
2.4. Дослідження генерації ПВЧ на основі темного струму фотодіода при різних температурах	64
2.5. Гібридний генератор ПВЧ із застосуванням КА	73
2.6. Порівняльний аналіз розроблених генераторів ПВЧ	75
2.6.1. Веб-камера, як джерело ПВЧ.....	75
2.6.1.1. Недоліки та обмеження створення ПВЧ за допомогою веб-камери	76
2.6.1.2. Сфера застосування ПВЧ, отриманих за допомогою веб-камери	78
2.6.2. Фотострум фотодіода як джерело ентропії	78
2.6.2.1. Переваги генераторів ПВЧ на основі фотоструму фотодіодів	78
2.6.2.2. Недоліки та обмеження генераторів ПВЧ на основі фотоструму фотодіода	79
2.6.2.3. Сфера застосування генераторів ПВЧ на основі фотоструму фотодіода	80
2.6.3. Генератор ПВЧ на основі темного струму фотодіода	82
2.6.3.1 Переваги генератора ПВЧ на основі темного струму фотодіода	82
2.6.3.2. Недоліки генератора ПВЧ на основі темного струму фотодіода	83
2.6.3.3. Сфера застосування генератора ПВЧ на основі темного струму фотодіода	84
2.6.4. Висновки щодо порівняльного аналізу розроблених генераторів ПВЧ	85

Висновки до розділу 2	88
РОЗДІЛ 3. РОЗРОБКА ТА ДОСЛІДЖЕННЯ	
ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ГЕНЕРАЦІЇ	
ВИСОКОЕНТРОПІЙНОЇ ПВЧ НА ОСНОВІ ФЕЯ	89
3.1. Програмний комплекс для генерації та аналізу	
хаотичних числових послідовностей на основі ФЕЯ	89
3.2. Інформаційна система для генерації ПВЧ на основі ФЕЯ	95
3.2.1. Створення апаратної частини інформаційної системи	95
3.2.2. Створення ПВЧ	97
3.2.3. Забезпечення перевірки отриманих ПВЧ на	
відповідність критеріях хаотичності	98
3.2.4. Формування архітектури інформаційної системи	
для генерації ПВЧ на основі ФЕЯ	99
3.2.4.1. Реалізація принципів СІА	99
3.2.5. Інформаційна система для генерації ПВЧ на основі ФЕЯ	102
3.2.6. Діаграми класів інформаційної системи	105
3.2.7. Тестування одержаних ПВЧ	109
3.3. Порівняльні дослідження технічних характеристик створеної	
інформаційної системи на основі ФЕЯ та відомих аналогів	110
3.3.1. Узагальнення результатів порівняння технічних	
характеристик створеної інформаційної системи на основі	
ФЕЯ та відомих аналогів	116
Висновки до розділу 3	118
РОЗДІЛ 4. ЗАСТОСУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ	
ГЕНЕРАЦІЇ ВИСОКОЕНТРОПІЙНОЇ ПВЧ НА ОСНОВІ ФЕЯ	
4.1. Сфери застосування інформаційної системи для	
генерації високоентропійної ПВЧ на основі ФЕЯ	119
4.1.1 Створення вхідного вектора для хеш-функції	119
4.1.1.1 Алгоритм хеш-функції.....	119

4.1.1.2. Удосконалений алгоритм створення хеш-функції за допомогою розробленої інформаційної системи генерації ПВЧ на основі ФЕЯ	125
4.1.2. Застосування інформаційної системи на основі ФЕЯ для створення ПВЧ у стеганографії	126
4.1.3 Застосування інформаційної системи на основі ФЕЯ для створення ПВЧ при формуванні криптографічних ключів	130
Висновки до розділу 4	133
ВИСНОВКИ	135
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	138
ДОДАТКИ	153
Додаток А. Лістинг частини коду програми	153
Додаток Б. Акти впровадження результатів дисертаційної роботи.....	250
Додаток В. Список публікацій за темою дисертації	255
Додаток Г. Відомості про апробацію матеріалів дисертації.....	260

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

- ПВЧ – послідовність випадкових чисел;
- PRNG - програмні генератори псевдовипадкових чисел;
- TRNG - апаратні генератори випадкових чисел;
- NIST – система стандартів оцінювання та тестування на випадковість;
- X - дискретна випадкова величина;
- x_i - значення дискретної випадкової величини;
- $H(X)$ - ентропія Шеннона;
- хеш – функція, що перетворює вхідні дані будь-якого (як правило великого) розміру в дані фіксованого розміру;
- КА – клітинний автомат
- КА-30 – клітинний автомат за правилом 30;
- КА-90 – клітинний автомат за правилом 90;
- КА-105 – клітинний автомат за правилом 105;
- КА-150 – клітинний автомат за правилом 150;
- AES, ChaCha20 – симетричні криптографічні алгоритми;
- RSA, ECC – асиметричні криптографічні алгоритми;
- MD, SHA - стандартні алгоритми хешування;
- UAT - вхідний тест користувача;
- f - функція стиснення;
- $BIC(a_j, a_k)$ - коефіцієнт кореляції (Bayesian information criterion) між j -ю та k -ю компонентою вихідного вектора;
- M_0 – повідомлення довжиною n біт;
- C_i – стискання частини вхідного повідомлення M_i ;
- h_{i-1} – ХЕШ-образ попереднього блоку;
- S_k – циклічний побітовий зсув в залежності від k ;
- k – номер 128 бітного слова у 512-бітному блоці;
- XOR – логічна операція та бітовий оператор;

- F - функція хешування;
- Hamming (x, y) - відстань Хеммінга між двома n -бітними блоками x і y , тобто кількість збігів;
- C_i - функція стискування;
- ФЕЯ – фотоелектричні явища;
- ФЧЕ – фоточутливий елемент фотодіоду;
- ФД – фотодіод;
- I_{Ti} – темновий струм фотодіоду;
- U_{ni} – спад напруги на опорі навантаження R_{ni} i -того ФЧЕ;
- $Z_{\min}, Z_{\max}$ - мінімальне і максимальне значення співпадінь однакових чисел у ПВЧ;
- Z_{absent} - процент відсутніх елементів у ПВЧ;
- CIA - принципи конфіденційності, цілісності, доступності;
- N-tier - багаторівнева архітектура.

ВСТУП

Актуальність та обґрунтування теми дисертаційного дослідження.

Суттєвою ознакою сучасного стану суспільства є створення, обробка, накопичення, передача та отримання величезних інформаційних потоків, які повинні мати достатній рівень кібербезпеки, бути захищені для забезпечення як приватності даних, власниками яких є звичайні громадяни, бізнес об'єднання та держави. З точки зору захисту інтересів людини, держави і суспільства в цілому одним з напрямків розвитку програмної інженерії згідно указу президента України №37/2022 "Про рішення Ради національної безпеки і оборони України від 30 грудня 2021 року "Про План реалізації Стратегії кібербезпеки України", є створення алгоритмів та програмного забезпечення для кібербезпеки, яка, зокрема ґрунтується на генерації ПВЧ. Цей напрям програмної інженерії є одним із визначальних компонентів забезпечення кібербезпеки, а саме - забезпечення безпеки вхідних та вихідних потоків даних. Сучасні процесори не можуть самостійно генерувати ПВЧ достатнього рівня надійності, тому, для виконання такої задачі, їм потрібно відповідне програмне забезпечення. Результативні дослідження в цієї галузі сприяє кращому розумінню організації безпеки потоків даних шляхом шифрування та кодування. інтенсивності пікселів зображення з веб-камери, або цифрової відеокамери, фото- та темнового струму фотодіода з високою квантовою ефективністю. З огляду на існуючий математичний, алгоритмічний апарати та програмне забезпечення у галузі захисту інформації, залишається не до кінця зрозумілими цілий ряд питань, а саме: оптимальний вибір співвідношення між продуктивністю генерації ПВЧ і їх надійністю (стійкістю до зламу), ефективності застосування клітинних автоматів для підвищення рівня хаотичності ПВЧ, методики експрес діагностики рівня хаотичності згенерованих ПВЧ, пошук нових джерел ПВЧ, максимально наближених за своєю природою до дійсно хаотичної структури, совтрєння вхідного вектора хеш-функції, тощо. Зважаючи на вище наведене, розробка та дослідження

інформаційної системи генерування високоентропійної послідовності випадкових чисел є актуальною задачею.

Об'єктом дослідження є хаотичні системи, які забезпечують генерацію послідовності випадкових чисел.

Предметом дослідження є моделі та методи для генерації високоентропійної послідовності випадкових чисел.

Метою дисертаційного дослідження є розробка інформаційної системи для генерації високоентропійних послідовності випадкових чисел та оцінювання ступеня їх хаотичності на основі програмного забезпечення власної розробки та нових підходів для визначення балансу між продуктивністю на надійністю створених послідовності випадкових чисел.

Відповідно до поставленої мети в дисертаційній роботі розв'язуються **основні задачі**:

- провести аналітичний огляд існуючих програмних та апаратних засобів та методів генерації високоентропійних ПВЧ;
- розробити та дослідити технологію генерації ПВЧ з пікселів зображення фоточутливої матриці відео, або веб-камери;
- розробити та дослідити технологію генерації ПВЧ з фотоструму фотодіода;
- розробити та дослідити технологію генерації ПВЧ з темного струму фотодіода;
- розробити та дослідити технологію гібридної генерації ПВЧ із застосуванням клітинних автоматів;
- розробити інформаційну систему для генерації ПВЧ на основі створених генераторів, в основу якої має бути покладено можливість обрання балансу між швидкістю генерації ПВЧ та її надійністю, а також перевірку ПВЧ на відповідність критеріям хаотичності;
- провести порівняльні дослідження характеристик створеної інформаційної системи та відомих аналогів;

- запропонувати приклади застосування створеної інформаційної системи для синтезу криптографічних алгоритмів для хеш-функцій, стеганографії та криптографічних ключів на основі ПВЧ і КА;
- впровадити результати дисертаційної роботи на кафедрах та підприємствах.

Методи дослідження. В основу дисертаційної роботи покладені методи інтелектуального аналізу даних, а саме: отримання та аналіз ПВЧ за допомогою технологій екстракції даних, обробки даних за допомогою КА хаотичного типу, спрощені методи оцінки статистичних характеристик ПВЧ.

Наукова новизна. В дисертаційній роботі

1) вперше:

- Запропоновано метод генерації ПВЧ, особливістю якого є використання фотоелектричних явищ, що, на відміну від існуючих методів, забезпечує швидкість генерації ПВЧ від 0,288 до 1 Гбіт/сек при високому рівні випадковості, є доступними, високопродуктивними, здатними для гнучкого налаштування, що дозволяє підвищити ефективність генерації ПВЧ;
- розроблено інформаційну систему екстракції ПВЧ з стохастичних фізичних явищ, а саме, інтенсивності пікселів зображення веб-камери, фото- та темнового струму фотодіода, яка, на відміну від аналогів, містить модуль обробки отриманих ПВЧ за допомогою технології КА та аналітичний модуль інтелектуальної спрощеної статистичної оцінки згенерованих ПВЧ, що дозволяє пройти перевірку 12 тестів NIST з 15 можливих;
- запропоновано метод балансу між продуктивністю та якістю генерації ПВЧ, яка на відміну від аналогів, може працювати у режимі неперервного потоку, що дозволяє у процесі генерації ПВЧ віддавати

перевагу або швидкодії, або випадковості, що дозволяє керувати рівнем надійності згенерованої ПВЧ;

2) *набуло подальшого розвитку:*

- методика використання КА: створено бібліотеку на мові програмування Java по роботі з лінійними КА на основі примітивних побітових операцій низького рівня, запропоновано і експериментально доведено високу продуктивність, низьке ресурсоспоживання і високу якість генерації і обробки ПВЧ запропонованим функціоналом лінійних клітинних автоматів хаотичного типу (правила 30, 90, 105), що дозволило скоротити час обробки майже на порядок у порівнянні з парадигмою об'єктноорієнтованого програмування;
- оптимізація балансу між високою продуктивністю і низькою ресурсоемністю функціоналу на КА; технологія генерації ПВЧ, що переходить на якісний рівень – неперервна генерація ПВЧ заданої продуктивності та якості.

Зв'язок роботи з науковими програмами, планами, темами.

Дисертаційне дослідження виконано на кафедрі програмного забезпечення комп'ютерних систем Чернівецького національного університету імені Юрія Федьковича. Її зміст відповідає тематиці науково-дослідної роботи: «Дослідження, моделювання та розробка програмного забезпечення складних динамічних систем» (Державний реєстраційний номер 0121U109232).

Публікації. За темою дисертації опубліковано 11 робіт, в яких подано результати досліджень. З них 4 статті у рецензованих виданнях (1 з яких – в журналі, що індексується у наукометричній базі SCOPUS, 4 – в українських фахових виданнях), у збірниках матеріалів міжнародних та всеукраїнських наукових конференцій – 7 робіт.

Особистий внесок здобувача. Дисертант брав безпосередню участь у постановці задач дослідження, виборі методів їх розв'язання, аналізі та інтерпретації отриманих результатів, а також у підготовці матеріалів до

публікації в усіх працях [1–12]. Основний внесок здобувача полягає у розробці програмного забезпечення для генерації та дослідження випадкових послідовностей, проведенні експериментальних досліджень ентропійних джерел (фоточутливих матриць, темного струму фотодіодів, відеопотоку вебкамер), статистичному аналізі отриманих послідовностей, зокрема із застосуванням клітинних автоматів та тестів NIST, а також у розробці та дослідженні алгоритмів і моделей генераторів випадкових чисел. Результати досліджень [6–12] доповідались та обговорювались на міжнародних і всеукраїнських наукових та науково-практичних конференціях.

Апробація результатів. Основні результати роботи доповідались та обговорювались на наукових семінарах кафедри програмного забезпечення комп'ютерних систем, а також на Всеукраїнських та Міжнародних наукових і науково-практичних конференціях: «Проблеми інформатики та комп'ютерної техніки» (Чернівці, 2022, 2023, 2024), «IEEE 4th International Conference on Advanced Trends in Information Theory» (Київ, 2022), «Інформаційні технології: наука, техніка, технологія, освіта, здоров'я» (MicroCAD)» (Харків, 2023), «Current trends in scientific research development» (Boston, 2025), «Physical and technological problems of transmission, processing and storage of information in infocommunication systems» (Чернівці, 2025).

У травні 2025 року зроблено доповідь на «Physical and technological problems of transmission, processing and storage of information in infocommunication systems», де доповідались основні наукові результати, які увійшли у дисертаційне дослідження.

Структура та обсяг роботи. Дисертаційна робота є завершеним дисертаційним дослідженням, загальним обсягом 260 сторінок (з них: 137 сторінок основного тексту, 15 сторінок – література, 108 сторінок – додатки).

Дисертація складається зі вступу, чотирьох розділів з підрозділами, висновків, списку використаних джерел (114 позицій) та додатків.

РОЗДІЛ 1

АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ГЕНЕРАЦІЇ ВИСОКОЕНТРОПІЙНОЇ ПОСЛІДОВНОСТІ ВИПАДКОВИХ ЧИСЕЛ

Одним з напрямків розвитку програмної інженерії, з точки зору захисту інтересів людини, держави і суспільства в цілому є створення алгоритмів та програмного забезпечення для кібербезпеки [1]. Методи генерації дійсно непередбачуваної ПВЧ є, в свою чергу, є одним із визначальних компонентів забезпечення потреб кібербезпеки, а саме - забезпечення безпеки вхідних та вихідних потоків даних.

Відомі різні методи генерації ПВЧ. Зокрема, комп'ютер використовує обсяг стекової пам'яті, або використовує поточне значення часу, що вимірюється в наносекундах (час Unix) в якості випадкового значення, з якого формується відповідна послідовність. Можна застосовувати дані з зовнішніх пристроїв. Наприклад, миша, USB, клавіатура, інші джерела. Вони усі називаються джерелами ентропії (хаосу). Слід зауважити, що ці значення, самі по собі, не є випадковими повністю. Вони знаходяться у певних межах, або мають коливання, які можна передбачити. Для перетворення таких послідовностей цих чисел на дійсно випадкову у потрібному діапазоні можна застосувати криптографічні перетворення. Наприклад такі, що притаманні клітинним автоматами [2-5], для отримання рівномірно розподілених випадкових значень, які мають нерівномірно розподілені значення джерела ентропії. Такі значення називають псевдовипадковими. Тому, що вони генеруються детерміновано з ентропії, а не є насправді випадковими.

Зрозуміло, що джерело ентропії (хаосу) можна використати для генерації ПВЧ. Це забезпечує певну унікальність і, що головне, непередбачуваність значень. Такі псевдовипадкові числові послідовності застосовуються в різноманітних мережах для забезпечення ефективних та безпечних з'єднань,

генерації криптографічних ключів, моніторингу цілісності, а також багатьох інших напрямках [6].

Тому створення надійних, безпечних програмних систем, які базуються на генерації зовні непередбачувані ПВЧ є актуальним завданням сучасної програмної інженерії та кібербезпеки.

1.1. Апаратні генератори випадкових чисел в системах інформаційної безпеки

1.1.1. Роль апаратних генераторів випадкових чисел у криптографічних системах

Однією з ключових передумов криптографічної стійкості сучасних протоколів захисту інформації є наявність надійного джерела випадкових чисел. Більшість криптографічних алгоритмів – як симетричних (AES, ChaCha20 тощо), так і асиметричних (RSA, ECC, постквантові схеми) – вимагають генерації секретних ключів, ініціалізаційних векторів, одноразових чисел (nonce) та сеансових параметрів, які повинні бути непередбачуваними з точки зору потенційного зломисника. Використання програмних псевдовипадкових генераторів (PRNG), що базуються на детермінованих алгоритмах, неминуче призводить до залежності якості випадковості від якості початкової ентропії та стійкості внутрішнього стану.

У цьому контексті апаратні генератори випадкових чисел (Hardware Random Number Generators, HWRNG або TRNG – True Random Number Generators) є критично важливими компонентами, оскільки використовують фізичні (у тому числі квантові) шумові процеси як джерело ентропії. На відміну від PRNG, апаратні TRNG не є детермінованими алгоритмами в математичному сенсі, а їх вихідні послідовності відображають флуктуації, обумовлені теплофізичними, електронними, радіочастотними або квантово-механічними явищами [7–9].

Стандарти NIST наголошують на необхідності застосування апаратних джерел ентропії для ініціалізації детермінованих генераторів випадкових бітів (Deterministic Random Bit Generators, DRBG (криптографічно стійкий генератор), регламентуючи вимоги до їх оцінювання та тестування в документах NIST SP 800-90A/B/C [10, 11], а також до статистичних тестів випадковості в NIST SP 800-22 [12]. У сертифікованих криптомодулях (FIPS 140-2/140-3) апаратний RNG є обов'язковим елементом криптографічної підсистеми.

1.1.2. Теоретичні основи оцінювання випадковості та ентропії фівфів

Формальне оцінювання якості випадкових послідовностей ґрунтується на понятті інформаційної ентропії. Для дискретної випадкової величини X зі значеннями x_i та ймовірностями $p_i = Pr\{X = x_i\}$ ентропія Шеннона [13] визначається як:

$$H(X) = - \sum p_i \log_2 p_i. \quad (1.1)$$

Якщо розглядати бінарну випадкову величину (біт), то максимальне значення ентропії досягається при $p_0 = p_1 = 0,5$ і становить $H(X) = 1$ біт. На практиці, для оцінювання «найгіршого» випадку часто використовують міні-ентропію:

$$H_{min}(X) = - \log_2 (\max p_i). \quad (1.2)$$

У контексті апаратних генераторів випадкових чисел вихідний сирий шум (raw noise) часто має певні статистичні спотворення: зсув бітового балансу, кореляції між сусідніми бітами, залежність від температури або електроживлення. Тому на практиці застосовують етап постобробки (whitening), який зазвичай реалізується через криптографічні хеш-функції, блокові/потоківі шифри або DRBG, що перетворюють сирий потік у

послідовність з мінімальною кореляцією та рівномірним розподілом. Вимоги до такої постобробки та до процедур health-тестування джерел ентропії докладно сформульовані в NIST SP 800-90B [12].

1.1.3. Класифікація сучасних апаратних генераторів випадкових чисел

З огляду на природу фізичного джерела шуму, сучасні апаратні генератори випадкових чисел доцільно поділити на такі основні класи:

1. Квантові генератори випадкових чисел (QRNG) – використовують квантово-механічні процеси, наприклад розподіл окремих фотонів на напівпрозорому дзеркалі (beam splitter), квантові флуктуації фази та інші явища, де результат фундаментально непередбачуваний [7, 8].

2. Класичні шумові генератори – ґрунтуються на тепловому шумі (Johnson–Nyquist noise), дробовий шум (shot noise), фліккер-шумі та шумами лавинного пробоя у напівпровідникових діодах, які підсилюються, дискретизуються та проходять постобробку [9, 14–19].

3. Вбудовані TRNG у захищених елементах – інтегровані в мікросхеми secure element / TPM / HSM як внутрішні джерела ентропії для генерації ключів і параметрів протоколів (приклади – мікросхеми сімейства ATECC [9], HSM-модулі Thales nShield / Luna [20]).

4. Open-hardware та open-source рішення – пристрої, апаратна схема й ПЗ яких доступні для незалежного аудиту (OneRNG, ChaosKey, NeuG тощо [15–18]). Вони часто мають нижчу швидкість та відсутність формальної сертифікації, але є важливими з точки зору прозорості реалізації та дослідницьких застосувань.

У цьому розділі розглянуто 10 представників зазначених класів, які на сьогодні є репрезентативними для ринку апаратних RNG, та проведено їх порівняльну характеристику.

1.1.4. Технічний огляд і порівняльний аналіз сучасних апаратних генераторів

1.1.4.1. Квантові генератори ID Quantique (Quantis QRNG)

Швейцарська компанія **ID Quantique (IDQ)** є одним із світових лідерів у галузі квантових генераторів випадкових чисел. Пристрій **Quantis QRNG USB** реалізує генерацію бітів на основі випадкового вибору траєкторії поодиноких фотонів у напівпрозорому дзеркалі, де два взаємовиключні результати детектування відповідають логічним «0» та «1» [7]. Швидкість генерації становить до 4 Мбіт/с, передбачено вбудовані процедури health-тестування та постобробки; пристрій пройшов випробування метрологічними інституціями та статистичні тести NIST SP 800-22 [12].

PCIe-карти **Quantis QRNG PCIe** (моделі 4M, 16M, а також новіші високошвидкісні варіанти) забезпечують вищу пропускну здатність (до сотень Мбіт/с) та інтегрують DRBG, сумісні зі стандартом NIST SP 800-90A [10]. Вони орієнтовані на серверні платформи, HSM-модулі, системи керування ключами та телекомунікаційне обладнання.

Переваги продукту:

- фундаментальна непередбачуваність квантового джерела,
- наявність сертифікації,
- висока якість статистики,
- відпрацьована технологія.

Недоліки продукту:

- висока вартість,
- пропрієтарність апаратури та ПЗ,
- орієнтація на корпоративний сегмент;

1.1.4.2. Високошвидкісні USB-TRNG: SwiftRNG Pro та TrueRNGpro

SwiftRNG Pro (TectroLabs) є прикладом високопродуктивного USB-TRNG, який досягає швидкості до 210 Мбіт/с завдяки використанню двох

незалежних шумових джерел, апаратного змішування та криптографічної постобробки [9]. Пристрій супроводжується бібліотеками та прикладами інтеграції для основних ОС і призначений для серверних сценаріїв, де потрібне швидке наповнення пулу ентропії, генерація ключів у PKI-інфраструктурі, VPN-шлюзах, криптографічних сервісах.

TrueRNGpro (ubld.it) забезпечує швидкість понад 3,2 Мбіт/с, надаючи дані через інтерфейс USB CDC як потік байтів [14]. Конструктивно пристрій використовує шум напівпровідникових елементів, проходить елементарні внутрішні тести й орієнтований на застосування в робочих станціях, невеликих серверах та віртуалізованих середовищах.

Переваги продукту:

- висока пропускна здатність, простота інтеграції («plug-and-play» через USB),
- широкий спектр підтримуваних платформ.

Недоліки продукту:

- відсутність повністю відкритого дизайну та формальної високорівневої сертифікації (FIPS 140-2/3, CC),
- слабка довіра до виробника у питанні унікальності та стороннього доступу.

1.1.4.3. Open-hardware рішення: ChaosKey, OneRNG, NeuG

ChaosKey (Altus Metrum) – USB-TRNG, який використовує шум зворотно зміщеного p–n-переходу, інтегрується з ядром Linux через спеціальний драйвер chaoskey і може безпосередньо підживлювати пристрій /dev/random [15].

OneRNG – відкрите апаратне рішення, що поєднує шум лавинного діода та, за потреби, радіочастотний шум з антени як джерела ентропії; схема й прошивка доступні для аудиту [16].

NeuG TRNG – проект, орієнтований на екосистему вільного ПЗ GNU; він реалізує генерацію випадкових чисел із подальшою відкритою криптографічною постобробкою, забезпечуючи прозорість реалізації [18].

Переваги продукту:

- відкритість апаратної та програмної частини,
- можливість незалежного аудиту,
- низька вартість,
- гнучкість налаштування.

Недоліки продукту:

- відносно невисока швидкість генерації,
- відсутність формальної сертифікації за FIPS/CC,
- залежність якості від активності спільноти розробників.

1.1.4.4. TRNG на основі дробового шуму: Comscire PQ4000KS

Пристрій **Comscire PQ4000KS** використовує дробовий шум як первинне джерело ентропії, що виникає через дискретну природу носіїв заряду під час руху крізь потенційні бар'єри [17]. Вихідний аналоговий сигнал підсилюється, оцифровується й піддається криптографічній постобробці. У документації декларовано швидкість до 4 Мбіт/с, наявність безперервних health-тестів як для сирого, так і для обробленого потоку, а також блокування виходу в разі виявлення аномалій [17].

Переваги продукту:

- фізично обґрунтоване джерело ентропії,
- орієнтація на критичні сфери (гемблінг, безпека),
- підтримка процедур тестування відповідно до рекомендацій NIST SP 800-22/90B.

Недоліки продукту:

- висока вартість,

- закритість реалізації,
- відносно вузький ринок.

1.1.4.5. Вбудовані RNG у захищених елементах та HSM: ATECC608B, nShield / Luna

Мікросхема **Microchip ATECC608B** поєднує в собі захищене зберігання ключів, апаратну підтримку криптографічних операцій (ECDSA, AES тощо) та внутрішній генератор випадкових чисел [16]. TRNG використовується для генерації ключів, попси та інших випадкових параметрів, а його параметри заявлені як відповідні вимогам NIST SP 800-90A/B/C. Інтерфейс I²C та компактний корпус роблять пристрій придатним для вбудованих і IoT-застосувань.

Апаратні модулі безпеки **Thales nShield / Luna HSM** містять сертифіковані RNG, які є невід’ємною частиною криптомодуля, сертифікованого за FIPS 140-2 (рівні 2–3) та іншими стандартами [20]. RNG у таких пристроях використовуються для генерації ключів верхнього рівня, сертифікатів, токенів автентифікації, а також для підживлення внутрішніх DRBG відповідно до NIST SP 800-90A.

Переваги продукту:

- висока довіра,
- підтверджена сертифікацією;
- глибока інтеграція в архітектуру безпеки (secure element / HSM);
- наявність механізмів фізичного та логічного захисту.

Недоліки продукту:

- відсутність доступу до RNG як до «чистого» окремого модуля;
- висока вартість і складність інтеграції;
- необхідність роботи через специфічні API виробника.

Порівняльні характеристики розглянутих генераторів ChaosKey, OneRNG, NeuG та PQ4000KS наведено в таблиці 1.1. Характеристики узагальнено за відкритими технічними описами, публікаціями розробників та результатами незалежних досліджень.

Таблиця 1.1.

Порівняльні характеристики генераторів ChaosKey, OneRNG, NeuG,
PQ4000KS

Характеристика	ChaosKey	OneRNG	NeuG	PQ4000KS
Тип генератора	Апаратний TRNG на основі електронного шуму [15, 21-28]	Апаратний TRNG на основі лавинного шуму [29-33]	Апаратний TRNG на основі шуму АЦП [33-35]	Високоточний квантовий апаратний TRNG [17, 36-39, 40]
Фізичне джерело ентропії	Зворотно зміщені біполярні транзистори (reverse-biased p–n junction) [15, 21]	Лавинний діод; радіочастотний шум з антени [29-33]	Шум процесу дискретизації (ADC noise) у мікроконтролері STM32F103 [33, 34]	Квантовий/дробового шуму у спеціалізованому кванто-вому генераторі [17, 36-39, 40]

Продовження таблиці 1.1.

Характеристика	ChaosKey	OneRNG	NeuG	PQ4000KS
Форм-фактор	Зовнішній USB-модуль у компакт-ному корпусі [15, 21-28]	Зовнішній USB-модуль (стикова плата з екрануванням) [29–31]	USB-модуль, інтегрований у невеликий корпус [33–35]	Зовнішній USB-пристрій настільного/рекового класу [17,36–38]
Інтерфейс підключення	USB 2.0 Full Speed; інтеграція через /dev/hwrng, /dev/chaoskeyX [15, 21,26–30]	USB/Serial (CDC); доступ як послідовний порт [29–33]	USB CDC; пристрій /dev/ttyACM0 у GNU/Linux [33–35]	USB; власні драйвери та ПЗ для основних ОС [17, 36–38]
Орієнтовна швидкість генерації	≈ 10 Мбіт/с [28]	≈ 0,35 Мбіт/с [29, 33]	≈ 0,64 Мбіт/с (>80 кБ/с) [35]	≈ 4 Мбіт/с [17, 36, 37]
Підтримувані платформи	Переважно GNU/Linux; використання драйвера ядра [27]	GNU/Linux, *BSD та інші ОС із підтримкою послідовного USB-інтерфейсу [29–33]	GNU/Linux та інші POSIX-системи з підтримкою USB-CDC [33–35]	Windows, GNU/Linux та інші ОС, що підтримуються виробником [17, 36–38]

Продовження таблиці 1.1.

Характеристика	ChaosKey	OneRNG	NeuG	PQ4000KS
Відкритість апаратури та ПЗ	Відкрита апаратна схема (TAPR OHL) та прошивка (GPLv2) [15, 21]	Open-hardware та open-source ПЗ (GPLv3) [29–31]	Вільне ПЗ (GPLv3); відкритий вихідний код та документація [33–35]	Промислове пропрієтарне рішення; закрита апаратура та ПЗ [17, 36–38]
Відкритість апаратури та ПЗ	Відкрита апаратна схема (TAPR OHL) та прошивка (GPLv2) [15, 21]	Open- hardware та open-source ПЗ (GPLv3) [29–31]	Вільне ПЗ (GPLv3); відкритий вихідний код та документація [33–35]	Промислове пропрієтарне рішення; закрита апаратура та ПЗ [1, 36–38]
Вбудовані тести та постобробка	Базові внутрішні перевірки; подальший аналіз покладається на ОС [15, 21–28]	Примітивне відбілювання та моніторинг стану шумового джерела [29–32]	Криптографічне кондиціонування на основі SHA-256; використання статистичних пакетів [33–35, 39]	Безперервне апаратне тестування сирого та остаточного потоку; блокування виходу при зниженні ентропії [17, 36, 37, 28]

Продовження таблиці 1.1.

Характеристика	ChaosKey	OneRNG	NeuG	PQ4000KS
Відкритість апаратури та ПЗ	Відкрита апаратна схема (TAPR OHL) та прошивка (GPLv2) [15, 21]	Open- hardware та open-source ПЗ (GPLv3) [29–31]	Вільне ПЗ (GPLv3); відкритий вихідний код та документація [33–35]	Промислове пропрієтарне рішення; закрита апаратура та ПЗ [1, 36–38]
Вбудовані тести та постобробка	Базові внутрішні перевірки; подальший аналіз покладається на ОС [15, 21–28]	Примітивне відбілювання та моніторинг стану шумового джерела [29–32]	Криптографічне кондиціонування на основі SHA-256; використання статистичних пакетів [33–35, 39]	Безперервне апаратне тестування сирого та остаточного потоку; блокування виходу при зниженні ентропії [17, 36, 37, 28]
Сертифікація та стандарти	Формальної сертифікації FIPS/CC немає; орієнтація на FLOSS [15, 21–28]	Відсутність FIPS/CC; наголос на відкритості та верифікованості [29–32]	Відсутність формальної сертифікації; орієнтація на GNU/FLOSS-спільноту [33–35]	Орієнтація на відповідність NIST SP 800-90B/C, застосування у FIPS-сумісних системах [17, 36–38, 28, 41]

Продовження таблиці 1.1.

Характеристика	ChaosKey	OneRNG	NeuG	PQ4000KS
Типові сценарії використання	Підживлення пулу ентропії серверів і робочих станцій, дослідження, віртуалізовані середовища [15, 21–28, 42, 43]	Робочі станції, невеликі сервери, маршрутизатори, open source інфраструктура [29–32, 42, 43]	Інфраструктура вільного ПЗ, лабораторні дослідження, навчальні стенди [33–35]	Гемблінг, фінансові системи, критичні інформаційні системи [17, 36–39; 40]
Основні переваги	Повністю відкритий дизайн; підтримка ядром Linux; висока швидкодія; компактність [15, 21–28]	Відкрите апаратне забезпечення; можливість перевірки та перепрошивки; використання двох джерел шуму [29–32]	Повністю вільне ПЗ; прозора реалізація; використання криптографічного кондиціонування [33–35]	Висока якість випадковості; безперервні тести; орієнтація на суворі галузеві вимоги [1, 36–39; 40]
Основні недоліки	Відсутність сертифікації; зняття з виробництва; необхідність USB- порту [15, 21–28]	Низька швидкодія; відсутність сертифікації; залежність від активності спільноти [29–32]	Мала пропускна здатність; орієнтація переважно на GNU/Linux-середовище [33–35]	Висока вартість; закритість внутрішньої реалізації; нішевий характер продукту [17, 36–39; 40]

* Примітка: наведені швидкості є орієнтовними та ґрунтуються на відкритих технічних специфікаціях і узагальнених результатах вимірювань [18, 19, 22, 25, 7, 26, 30].

1.1.4.6. Узагальнення щодо характеристик найбільш відомих апаратних генераторів випадкових чисел в системах інформаційної безпеки

Проведений аналіз 10 сучасних апаратних генераторів випадкових чисел показує, що на ринку співіснують рішення з принципово різною фізичною природою джерела ентропії та орієнтацією на різні сегменти застосувань – від високошвидкісних квантових карт PCIe та HSM до відкритих USB-пристроїв для ентузіастів і компактних TRNG, інтегрованих у захищені елементи [21].

Квантові генератори (IDQ Quantis) забезпечують максимальний рівень теоретичної непередбачуваності та мають потужну сертифікаційну підтримку, але характеризуються високою вартістю та закритістю. Комерційні USB-TRNG (SwiftRNG Pro, TrueRNGpro, Comscire PQ4000KS) демонструють високі швидкості та орієнтовані на серверні та спеціалізовані задачі, проте також залишаються пропрієтарними. Відкриті рішення (ChaosKey, OneRNG, NeuG) є важливими з точки зору аудиту та наукових досліджень, хоча поступаються за пропускну здатністю та формальною відповідністю регуляторним вимогам. Вбудовані RNG у secure-елементах (ATECC608B) та HSM-модулях (nShield / Luna) слід розглядати як складову комплексної архітектури безпеки, а не як ізольоване джерело випадковості.

З урахуванням вимог стандартів NIST (SP 800-22, SP 800-90A/B/C) і FIPS 140-2/3 [12-14] вибір конкретного генератора повинен ґрунтуватися на поєднанні характеристик ентропії, пропускну здатності, форм-фактора, рівня сертифікації та моделі загроз. При цьому рекомендується використовувати апаратні TRNG як джерело початкової ентропії для криптографічних DRBG із подальшим регулярним health-тестуванням, що відповідає сучасним криптографічним практикам і рекомендаціям міжнародних стандартів.

1.2. Генератори ПВЧ на основі фотоприймальної техніки

Існуючі програмні генератори ПВЧ (ПГПВЧ) на основі відомих математичних алгоритмів є досить передбачуваними [47]. Друга вада існуючих методів генерації псевдовипадкових послідовностей, полягає у їх загальнодоступності. Наприклад, генерації, створені мовою Java [48, 49], теоретично, можуть бути умовно доступні для успішної атаки на алгоритм шифрування. Розвиток обчислювальних потужностей, зокрема квантові обчислення [50, 51], обумовлює успішність атаки майже до практичного рівня.

Загалом, для генерації випадкових чисел використовуються два основних методи. Один базується на розробці і використанні спеціалізованих пристроїв, які використовують певні фізичні джерела шуму [52, 53]. Такі пристрої вимагають додаткових адаптерів для використання з звичайними комп'ютерами.

Другий методичний підхід передбачає застосування також фізичних подій, але таких, що відбуваються у комп'ютерних пристроях. Він, очевидно, є більш доцільним.

Наочний приклад цього методу генерації випадкових чисел є застосування лічильника тактової частоти процесора. Але, цей метод чутливий до зовнішніх чинників, і, відповідно, до зовнішнього впливу на процес генерації випадкових чисел [54, 55]. У [56] пропонується генерація випадкової послідовності на основі оптичного маніпулятора миші. Такий метод надає можливість генерувати випадкові числа, які мають неоднорідний розподіл. Але, такий метод забезпечує швидкість генерації випадкової послідовності чисел лише 1 кбіт/с. Це обмежує його застосування у високошвидкісних системах шифрування.

Принципово новий підхід до генерації випадкової числової послідовності запропоновано у [57, 58]. Проте останні публікації були майже десять років назад, і стан обчислювальної техніки не дозволяв генерувати ПВЧ з великою продуктивністю. А сучасний рівень криптографічних вимог до продуктивності

генерації — не менше 100 Мбіт/сек. Тож наші дослідження були присвячені веб-камері, як потенційному надійному високопродуктивному джерелу ентропії.

Виявлено, що значення інтенсивності пікселів веб-камери мають непередбачуваний характер, їх не видно неозброєним оком, але вони чітко фіксуються апаратно-програмними засобами. Оскільки веб-камери зазвичай побудовані на основі ПЗС матриць, створених на основі кремнію, нам здалося доцільним розглянути наступне питання. Чи можливе застосування в якості джерела ПВЧ не зображення, яке створює веб-камера, а, власне значення темного струму, наприклад, затемненого фотодіоду, створеного на основі кремнію. Подібні роботи проводилися раніше [59, 60], але в них, в якості джерела випадкової послідовності, розглядалися інші напівпровідникові вироби.

Цікавість до цього питання зумовлена тією обставиною, що зворотний струм фотодіоду, який працює у фотодіодному (прямому) режимі, при електричному зміщенні на р-п переході, є дійсно не цілком передбачувана величина [61]. Зазвичай, при створенні фотодіодів, або інших фоточутливих напівпровідникових приладів, нормується максимальна межа темного струму при певному зміщенні на р-п переході [62]. А точне значення постійно змінюється в певних межах. Це зумовлено і великою кількістю чинників, що впливають на генерацію темного струму фотодіода, в першу чергу — величиною електричного зміщення на р-п переході.

Попри безліч методів генерації ПВЧ, як було показано у попередньому розділі, серед відомих методів переважають системи, орієнтовані насамперед на спільноту відкритого програмного забезпечення, де ключовими є прозорість реалізації, можливість незалежного аудиту та низький поріг входження за вартістю. Вони забезпечують достатню для більшості серверних і робочих сценаріїв пропускну здатність, але не мають формальної сертифікації за FIPS

або Common Criteria, що може обмежувати їх застосування в строго регульованих галузях.

З другого боку є продукти високої ціни, орієнтовані на сегмент високої довіри, де домінують вимоги до відповідності рекомендаціям NIST і галузевим стандартам, а також необхідність безперервного моніторингу якості випадковості. Закритість реалізації частково компенсується наявністю детальної документації щодо процедур тестування. Відомі системи генерації ПВЧ мають досить низьку продуктивність (швидкодію) і не забезпечують баланс між продуктивністю та надійністю згенерованої ПВЧ. А саме – між швидкістю генерації та ступенем відповідності вимогам, що висуваються до генерації.

На противагу відомим система генерації джерело ентропії, що базується на ФЕЯ, зокрема фотодіод, має одночасно і низьку ціну і стохастичну природу і високу продуктивність і прозору реалізацію.

Тому, постало питання щодо розробки та дослідження технологій генерації ПВЧ, джерелом яких є фотоелектричні явища у фотоприймачах, а саме – інтенсивність пікселів веб-камери у затемненому стані, фото- та темновий струм фотодіода, і, як результат, створення інформаційної системи для генерації ПВЧ на основі проведених розробок та досліджень.

Висновки до розділу 1

У цьому розділі проведено аналітичний огляд методів генерації високоентропійної ПВЧ, зокрема розглянута саме апаратна генерація випадкових чисел в системах інформаційної безпеки, де джерелом ентропії виступає непередбачуване стохастичне фізичне явище.

Розглянуто різні підходи до джерела ентропії та програмно-конструкторського оточення. Проаналізовано їх значення у криптографічних системах та показані сильні сторони і вузькі місця з точки зору ціни, складності, надійності, довіри, продуктивності, керованості, моніторингу і

ліцензуванню.

Обґрунтовано доцільність дослідження ФЕЯ як джерела ентропії для генерації ПВЧ. Така конструкція дозволить частково покрити недоліки, що притаманні існуючим системам.

На основі проведеного дослідження сформовані завдання дослідження.

Основні наукові результати розділу опубліковані в працях [45, 57].

РОЗДІЛ 2

ДОСЛІДЖЕННЯ ВЛАСТИВОСТЕЙ ВЕБ-КАМЕРИ ФОТО- ТА ТЕМНОВОГО СТРУМІВ ФОТОДІОДА В ЯКОСТІ ДЖЕРЕЛА ДЛЯ ГЕНЕРАЦІЇ ПВЧ

2.1. Дослідження генерації ПВЧ за допомогою веб-камери

Відповідно до мети та завдань дисертаційного дослідження було проведено вивчення можливості генерації ПВЧ на основі вилучення значень інтенсивності пікселів з матриці цифрової веб-камери (відеокамери) [57], яка створює відео зображення і може здійснювати його сегментацію. Для цього було розроблено метод вилучення значень інтенсивності пікселів зображення з кадру веб-камери та перетворення їх у ПВЧ, а також досліджено отримані ПВЧ на випадковість значень та ефект лавини.

У дослідженні було застосовано наступне обладнання.

Настільний комп'ютер ASUS Z97K: процесор Intel® Core™ i3-4170 CPU @ 3.70 GHz × 4,32 Gb RAM, SSD Kingston 240 Gb.

Цифрова веб-камера: FULL HD 1080P, TrueColor, QQVGA (176×144), QVGA (320 × 240), VGA (640 × 480), SVGA (800 × 600).

У вибраній веб-камері верхня межа роздільної здатності, згідно зі специфікацією, становить 800×600 пікселів (VGA), а режим за замовчуванням – 176×144 (QQVGA), роздільна здатність чверть-QVGA. За бажанням, цей розмір можна розширити до роздільної здатності HXGA (4096×3072) – це залежить від роздільної здатності вибраної камери [63].

Програмне забезпечення: ОС Ubuntu 22 LTS 64 bit, Java Amazon Corretto 17.0.5, IntelliJ IDEA 2023.3.4 (Ultimate Edition), пакет com.github.sarxos.webcam версії 0.3.12 для захоплення кадрів, пакет javax.imageio для обробки відеозображень, пакет java.security.SecureRandom – програмне забезпечення для

генерації ПВЧ.

Для спрощення процесу дослідження кадрів, їх захоплення виконувалося в режимі мінімальної роздільної здатності QQVGA (176×144). Технологія захоплення кадрів та екстракція значень пікселів у ПВЧ базується на можливостях пакету бібліотеки `com.github.sarxos.webcam` версії 0.3.12 Java 17, наведена на рис. 2.1.

Спочатку роблять послідовно роблять два знімки у форматі BMP або TIFF (1). Розмір за замовчуванням кожного з цих знімків становить 176×144 пікселів, що загалом становить 25344 пікселів. Цей розмір є розміром за замовчуванням для класу `Webcam` пакета захоплення веб-камери мови програмування Java версії 17. Очевидно, що чим коротший часовий інтервал між кадрами, тим менша ймовірність того, що щось може змінитися на них, і тим краще для експерименту. Мінімальний часовий інтервал залежить від обчислювальної потужності обладнання та становив 100 мілісекунд. Потім отримані фотографії надсилаються на сервер (2).

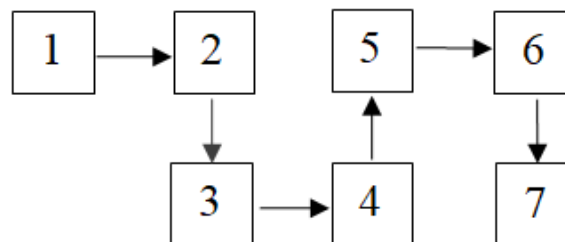


Рисунок 2.1 – Технологія генерації ПВЧ з пікселів зображення, сформованого веб-камерою.

Далі необхідно отримати з кожного фото так зване растрове зображення – тривимірна матриця (3), в якій зберігаються значення елементів зображення (пікселів).

Отримана матриця розміщується у пам'яті комп'ютера у вигляді об'єкта Java-класа `BufferedImage` (4). З цього об'єкту екстрагуються значення

відповідних пікселів веб-камери як одномірний масив (5). Масив обробляється згідно вимог замовника (6). Це може бути, наприклад, як скорочення масиву до потрібної довжини, так і об'єднання з іншим масивом, якщо споживач вимагає довгу послідовність або неперервний потік. Після обробки і перевірки послідовність передається споживачу (7).

Для 24-бітного растрового зображення кожен піксель містить значення трьох кольорів (кольоровий режим RGB) – червоного, зеленого, синього відповідно. Кожному кольору відводиться 1 байт = 8 бітів, тобто максимально можливе число в десятковому форматі дорівнює 255, а мінімальне – 0. Число 255 відповідає максимальній інтенсивності кольору, а число 0 – мінімальній інтенсивності.

З точки зору мови Java, діапазон даних байтового типу є числовим інтервалом цілих чисел $[-128 \dots +127]$, але це долається арифметичним перетворенням.

Таким чином, отримується матриця (тривимірний масив) випадкових чисел у діапазоні $[0 \dots 255]$:

$$A = (a_{i,j,k})^{m,n,p}, m=176, n=144, p=3, \quad (2.1)$$

На наступному кроці (4), після інтервалу 100 мс, береться ще один кадр того ж зображення та формується інша матриця:

$$B = (b_{i,j,k})^{m,n,p}, m = 176, n = 144, p = 3, \quad (2.2)$$

На наступному кроці (5) виконується елементарна операція віднімання матриці:

$$C = B - A = (b_{i,j,k} - a_{i,j,k})^{m,n,p}, \quad (2.3)$$

в яких записуються відповідні значення різниць у кольірних компонентах двох зображень кожного з пікселів.

Далі матриця сплющується в одновимірний вектор:

$$E = (e_i)^t, \quad (2.4)$$

де $t = m \times n \times 1 = 176 \times 144 \times 3 = 76032$.

Різниці отриманих значень можуть змінюватися в діапазоні від -255 до $+255$. Наступним кроком проводиться групування та сортування, і отриманий вектор випадкових значень додатково досліджується на відповідність вимогам швидкості, криптостійкості, ефекту лавини тощо (7).

Щоб уникнути стороннього впливу, веб-камера, підключена до комп'ютера, розташовувалася в затемненій коробці. Рівень освітленості становив 10^{-4} лк, що вважається повною темрявою. Температура в кімнаті становить 20°C . Для контрасту дослідження, як протилежний бік граничних випробувань, було зроблено зображення повністю білої однорідної стіни за рівномірного денного освітлення у хмарний день рівно опівдні (рис. 2.2), а також його профіль (яскравості рядка пікселів з номером 100).

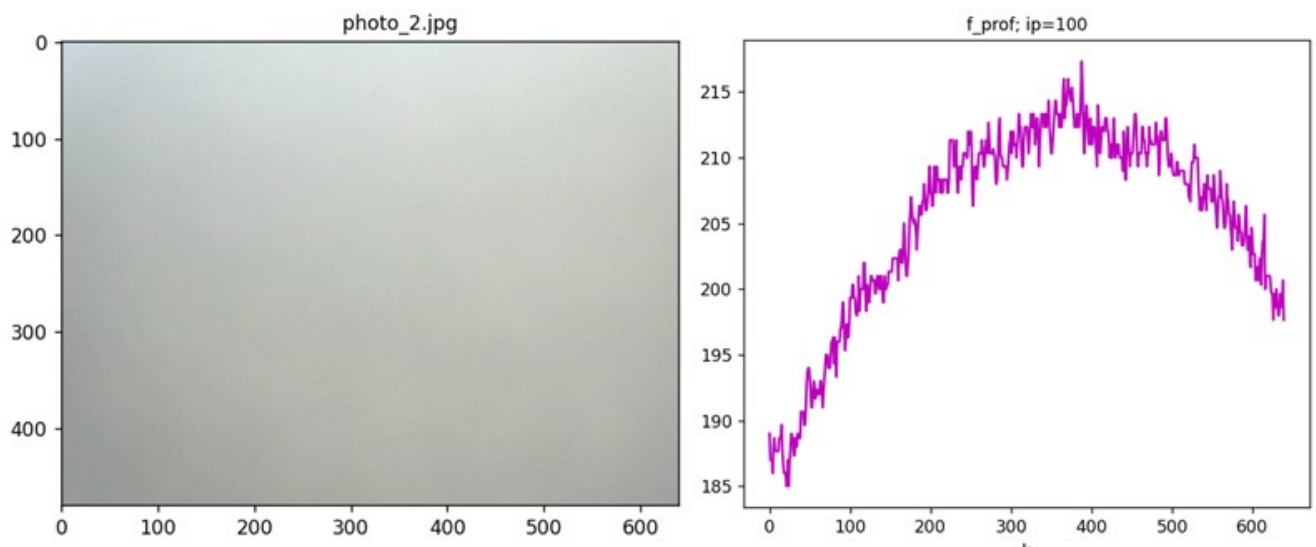


Рисунок 2.2 – Зображення білої стіни з веб-камери та його профіль.

На кадрі видно, що, незважаючи на однорідність та рівномірне освітлення зображення, картинка не була однорідною. У центрі вона яскравіша, ніж на периферії, чітко видно вкраплення інших кольорів, зокрема червоного та зеленого. Це дозволяє припустити, що в цьому байтовому масиві є елементи випадковості.

Дослідження згенерованих ПВЧ вимогам криптостійкості здійснювалося у відповідності до [64].

Гнучкі методи класу Webcam та BufferedImage мови програмування Java дозволили швидко та оптимально витягувати числову послідовність з екземпляра кадру негайно без складних матричних перетворень. Сама послідовність була поміщена в просту структуру даних Array та готова до подальшого дослідження.

Базовий програмний код програми:

```
Webcam webcam = Webcam.getWebcams().get(0);  
Dimension dimension = WebcamResolution.VGA.getSize();  
webcam.setViewSize(dimension);  
webcam.open()
```

Спочатку ініціалізується системна веб-камера. За замовчуванням режим роздільної здатності за замовчуванням - QQVGA. Камера відкривається для роботи, а доступ до неї іншим програмам блокується. За допомогою цього налаштування можна вибрати саму камеру (якщо їх кілька) та режим захоплення кадру (роздільну здатність кадру). Наведений нижче код отримує об'єкт знімка з камери, перетворює зображення в потік байтів TIFF та зберігає його в одновимірному масиві байтів.

```
BufferedImage image = webcam.getImage();  
ByteArrayOutputStream stream=new ByteArrayOutputStream()  
ImageIO.write(image, "tiff", stream);  
byte[] bytes = stream.toByteArray()
```

Створена таким чином технологія вилучення значень пікселів з кадру за допомогою процесу перетворення тривимірного масиву в одновимірний значно спрощується. Коли два послідовні кадри з веб-камери були отримані із затримкою 100 мілісекунд, їх додатково досліджували на наявність

розбіжностей у значеннях відповідних пікселів у колірній гамі TrueColor. Перераховані значення різниці освітленості для всіх точок двох кадрів.

```
list.add(bytes1[i] – bytes2[i])
```

```
list.add(bytes1[i] – bytes2[i])
```

Таким чином, для кожного пікселя в матриці зображення першого кадру до списку заноситься його різниця з відповідним пікселем другого зображення.

Щоб оцінити відсоток хаосу між двома кадрами, потрібно обчислити кількість усіх пікселів, які змінили свої значення, і поділити на загальну кількість. З двох послідовних кадрів веб-камери в режимі QQVGA було згенеровано 76032 числа між -255 та +255 як різницю відповідних значень пікселів. Відсоток пікселів, які змінили своє значення, становить 63% для повної темряви, для білої освітленої поверхні (200 люкс) – 82%, для звичайного офісного приміщення – 96%. Для повноти експерименту додатково було обчислено спектр розбіжностей, тобто кількість точок, що відповідають кожній розбіжності. Це стало можливим завдяки методам агрегації мови Java. Гістограма розподілу розбіжностей (Δx) за кількістю для двох кадрів представлена на рис. 2.3.

Таким чином, було перевірено випадковість отриманих ПВЧ, згенерованих веб-камерою, а саме - для кожного пікселя була обчислена різниця значень пікселів між першим і другим зображенням – відстань Хемінга.

Як можна бачити з гістограми, різниця переважно була в межах ± 10 . Приблизно 28000 пікселів (або 37%) мають нульове відхилення. А 9,5 тис. пікселів мають відхилення ± 1 тощо. Максимальне відхилення -50 має один пункт із 75 тис. Запропонований метод демонструє низьку міжкадрову кореляцію, що становить щонайменше 37% для повної темряви.

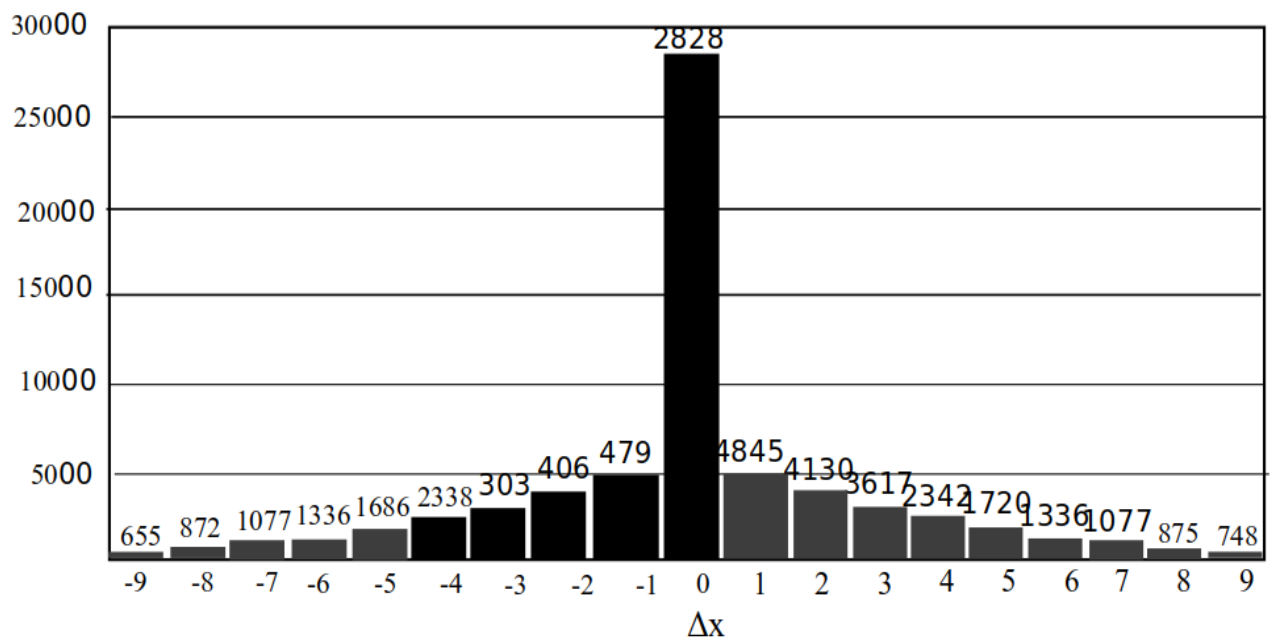


Рисунок 2.3 – Гістограма розподілу різниць Δx двох кадрів камери.

В деякому наближенні цей феномен можна розглядати як ефект лавин, хоча цей термін використовується у криптографії і означає, що зміна значення малої кількості бітів у вхідному тексті чи ключі веде до «лавинної» зміни значень вихідних бітів шифротекста. У нашому випадку: незначні зміни вхідного кадру (або два однакових кадри з різницею в 0.025 сек) приводять до лавинної зміни згенерованих з них ПВЧ. Якщо прийняти це допущення, то отримані результати (значення хаосу 63% за найнесприятливіших умов – повної темряви) свідчать про високий рівень ефекту лавин – згідно з вимогами критерію, цього достатньо для забезпечення мінімального рівня 50%. Це пояснюється підтвердженням гіпотези про хаотичну природу та непередбачуваність цифрового шуму в матриці камери через шум, що має походження від стохастичного характеру взаємодії фотонів з атомами фотодіодного матеріалу сенсора. Отримані результати надаються завдяки розвитку методу, описаного в [65], а саме генерації пікселів у темряві при освітленості близько 10^{-4} люкс.

На практиці така низька міжкадрова кореляція дає змогу говорити про неперервну генерацію ПВЧ, якщо у вихідний потік вбудовувати ПВЧ, що генеруються з послідовних кадрів камери. Ці ПВЧ між собою будуть слабо корелювати, що дозволяє говорити про довгий період зациклювання, на практиці - нескінченний. А сам по собі період зациклювання — одна з криптографічних вимог до генерації ПВЧ.

Розвиток дослідження має перспективу, а саме, коли будуть зняті обмеження, наприклад, не повна темрява, а яскравий барвистий фон (200 люкс), рівень хаосу буде ще вищим (до 100%). Теоретично можливо об'єднати систему зберігання кадрів (СЗК) в та отримати генерацію з високою продуктивністю до 1,25 Гбіт/с.

2.2. Фотострум фотодіода як джерело ПВЧ

Відповідно до мети та завдань дослідження було проведено вивчення доцільності застосування в якості джерела ПВЧ веб-камери. У попередніх дослідженнях веб-камера розглядалася як джерело ентропії [57]. Фотодіод є основою матриці веб-камери, тому його поведінка та характеристики, зокрема фотострум[66] потребували більш ретельних досліджень.

Теоретичний підхід до розв'язання завдання полягає у наступному.

Через стохастичну природу, фотодіод генерує значення фотоструму в певному діапазоні. Щільність ймовірності цих значень відповідає нормальному (гауссівському) розподілу.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, \quad (2.5)$$

де $f(x)$ – функція щільності ймовірності,

σ – стандартне відхилення,

μ – середнє значення.

Іншими словами, всі згенеровані значення повинні лежати в діапазоні

$[I_{min}, I_{max}]$, а гістограма їх розподілу подібна до показаної на рис. 2.4.

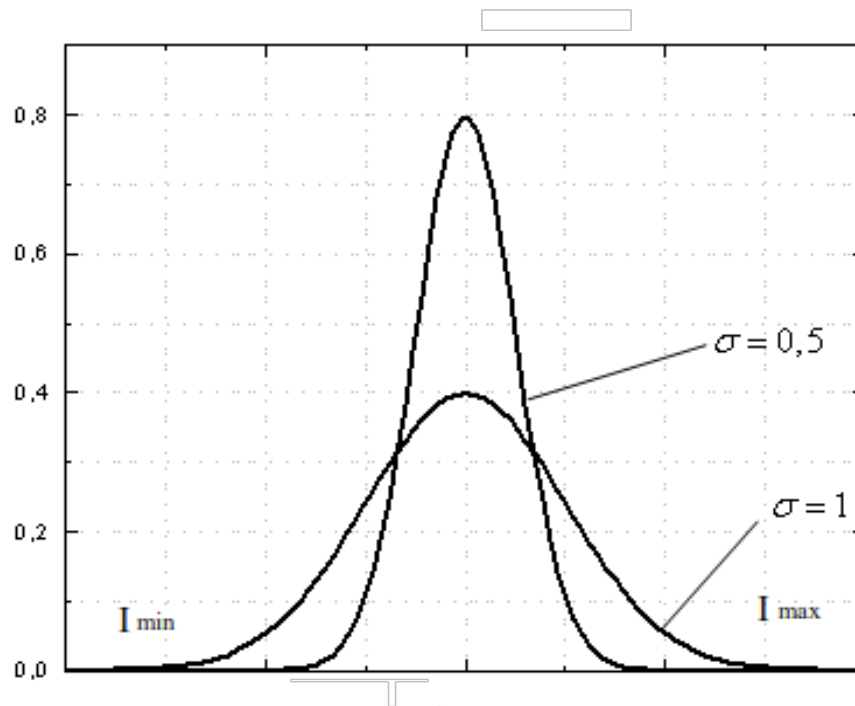


Рисунок 2.4 – Гістограма розподілу значень фотоструму.

Для отримання випадкових чисел з цього процесу генерації шкалу фотоструму необхідно розділити на набір безперервних інтервалів, кожен з яких відповідає певному цілочисельному значенню. Таким чином, кожне виміряне значення фотоструму зіставляється з певним числом. Таким чином, безперервне вимірювання струму створює послідовність випадкових цілих чисел.

Практична реалізація цієї ідеї полягає в наступному. Була розроблена експериментальна установка для генерації високочастотних імпульсів на основі виміряних значень фотоструму кремнієвого фотодіода ФД-288.

Фотодіод ФД-288 покритий світлорозсіювальним екраном та розміщений всередині ізольованої камери разом зі випромінювачем, який забезпечує рівень освітленості до 200 люкс та живиться від зовнішнього джерела. Фотодіод піддається впливу світла різної інтенсивності за допомогою спеціального блоку живлення. Виміряний фотострум фотодіода перетворюється на напругу

прецизійним перетворювачем струму в напругу, потім посилюється підсилювачем для подальшої обробки аналого-цифровим перетворювачем. Це створює цифровий сигнал, еквівалентний фотоструму, який обробляється процесором Arduino (112593) для адаптації цифрового сигналу та передачі його на USB-адаптер. Потім сигнал подається на USB-порт комп'ютера і, згодом, на процесор комп'ютера. Керування напругою фотодіода, інтенсивністю освітлення, тривалістю вибірки струму та кількістю вимірювань за одиницю часу здійснюється через інтерфейс інформаційної системи. На рис. 2.5 наведено фото експериментальної установки.

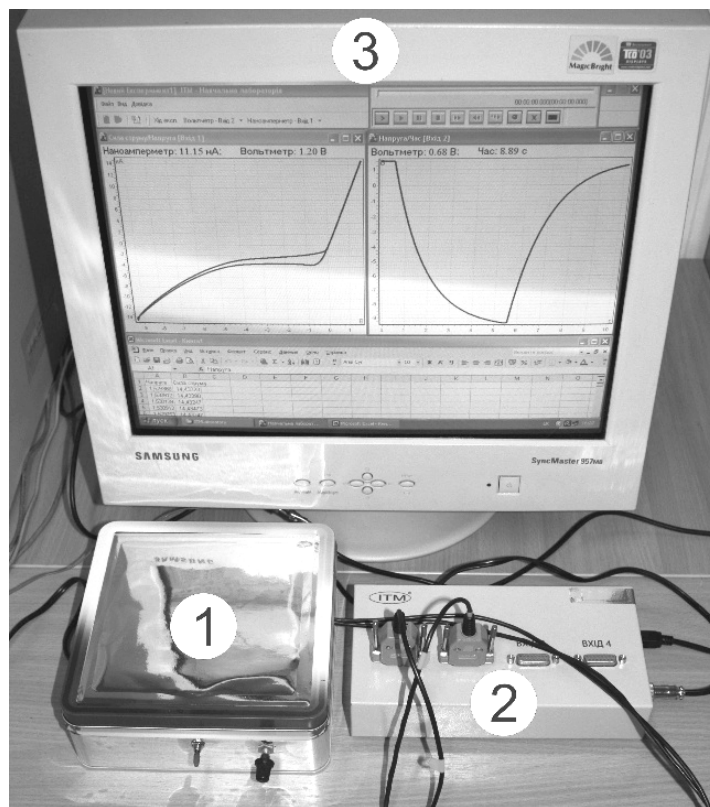


Рисунок 2.5 – Фото експериментальної установки

- 1 – вимірювальна приставка в заземленому металевому корпусі;
2 – електронний блок ITM; 3 – монітор комп'ютера.

Збільшення інтенсивності променя освітлення зміщує розподіл псевдовипадкових чисел у бік більш чіткого гауссівського профілю. Виміряні

значення напруги, еквівалентні фотоструму фотодіода, обробляються як потік вхідних даних комп'ютером за допомогою мови програмування Java (версія 17). Бібліотека `javaх-usb` використовується для завантаження вхідних даних у структурований список, де значення обробляються як випадкові числа в діапазоні від 0 до 255. Отриманий список чисел потім аналізується на відповідність статистичним вимогам за допомогою бібліотеки `DescriptiveStatistics`.

Для візуалізації статистичних характеристик ПВЧ застосовуються бібліотеки `jfree.chart` та `jfree.data`.

Гістограма виміряного розподілу фотоструму при рівні освітленості 1,0 люкс (майже повна темрява) показана на рис. 2.6.

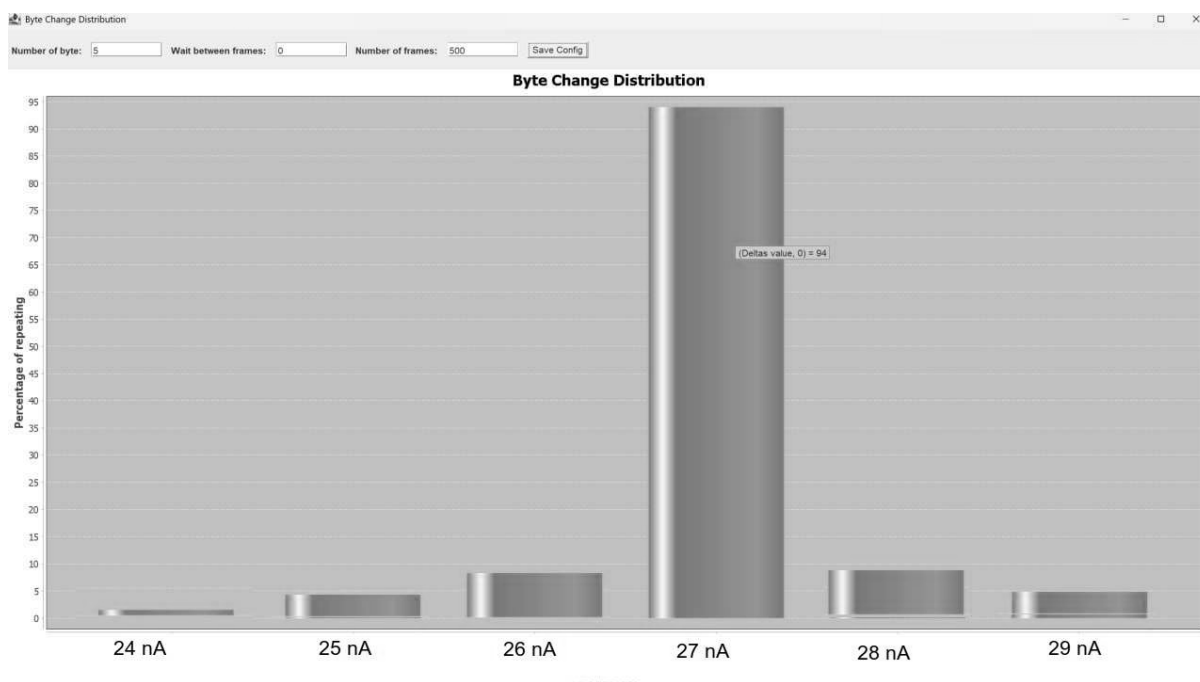


Рисунок 2.6 – Гістограма розподілу значень (спектр) фотоструму фотодіода ФД-288 (освітленість 1,0 лк)

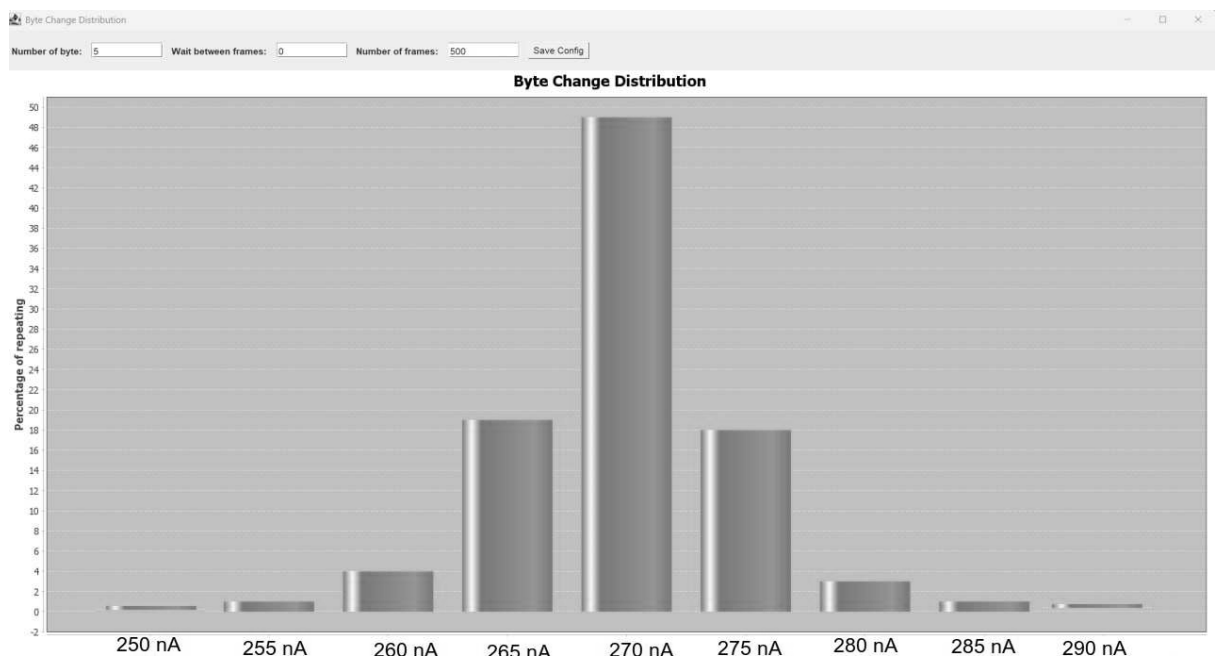
На рис. 2.6 показано, що після 500 вимірювань у майже повній темряві фотострум становив приблизно 27 нА з дуже малим відхиленням (~5%).

Практично це відповідає дельта-функції, хоча формально це можна описати як гауссівський розподіл, як і стверджувалось вище, див. формулу (2.5)

Така сама картина спостерігається і при освітленні 10 люкс – рис. 2.7, при цьому фотострум становив приблизно 270 нА з більш вираженим відхиленням (~50%). Це як формально, так і практично відповідає класичному гауссівському розподілу.

Зі статистичної точки зору розподіл таких цілочисельних значень описується пуассонівським розподілом. Ключова відмінність полягає в тому, що гаусовий розподіл є неперервним, оскільки змінна n може приймати будь-яке дійсне значення, тоді як у пуассонівському розподілі $n = 0, 1, 2, 3, \dots$ є дискретною.

Припустимо, що у нас є завдання згенерувати послідовність випадкових чисел у діапазоні 0 ..9. Тоді ми можемо формалізувати завдання, визначивши, наприклад, що 250 нА відповідає числу 0, ..., а 290 нА відповідає числу 9.



исунок 2.7 – Гістограма розподілу значень фотоструму фотодіода ФД-288 при освітленні 10 люкс.

Таким чином, виміряні неперервні значення перетворюються на дискретні цілочисельні випадкові числа, які відповідають пуассонівському розподілу.

Розподіл Пуассона – розподіл випадкової величини, що приймає невід'ємні цілі значення:

$$P(n) = \frac{(\bar{n})^n}{n!} e^{-\bar{n}}, \quad (2.6)$$

Він визначає ймовірність спостереження значення n у заданому вимірюванні, за умови, що події відбуваються незалежно та характеризуються середньою швидкістю (середнім значенням):

$$\bar{n} = \frac{1}{N} \sum_{i=1}^N n_i, \quad (2.7)$$

Матриця фотодіодів є основою сучасної веб-камери. У режимі SVGA матриця складається з 800x600 пікселів, де кожен піксель є триплетом RGB (червоний, зелений, синій) фотодіодів. Кожен фотодіод у матриці дещо відрізняється чутливістю від інших і виявляє різні світлові промені залежно від кадру веб-камери.

З цих причин, матрицю фотодіода можна розглядати як матрицю випадкових чисел.

$$A_{mn} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & \dots & \dots & a_{mn} \end{pmatrix}, \quad (2.8)$$

де $m = 800$,

$n = 600$,

a_{mn} – випадкове значення у власному діапазоні, що відповідає розподілу Пуассона.

Вирівнювання цієї матриці в одновимірний масив створює послідовність із 144 000 випадкових незалежних чисел. Таким чином, наша експериментальна установка функціонує як справжній генератор ПВЧ. Звичайно, набір вимог, що накладаються на такий ПВЧ, необхідно перевірити та дослідити далі.

Таким чином показано, що фотострум фотодіода за постійного освітлення має стохастичний характер і відповідає гауссівському розподілу. Зі збільшенням освітленості дисперсія значень зростає, але гауссівському характер розподілу залишається незмінним.

Кожен елемент матриці фотодіода генерує випадкові числа з нормальним (гауссовим) розподілом, що характеризується власним середнім значенням та стандартним відхиленням.

Матриця фотодіода здатна генерувати набір випадкових чисел, що відповідають значенням кожного елемента (фотодіода). Фотодіодна матриця є основою звичайної веб-камери та здатна генерувати послідовність приблизно 144 000 випадкових чисел у режимі SVGA. В нашому випадку було досягнуто значення продуктивності до 288 Мбіт/сек.

Недоліком технології генерації ПВЧ за допомогою фотоструму фотодіода є необхідність застосування в апаратній частині джерела світла, що певним чином ускладнює її конструкцію. Тому було проведено дослідження другої складової струму фотодіоду, так званого темнового струму. Для його генерації світло є зайвим фактором. Тому, згідно завданням дослідження, у подальшому проводився аналіз темнового струму фотодіода як джерела ПВЧ.

2.3. Темновий струм фотодіода як джерело ПВЧ

Темновий струм фотодіоду визначають при вимірювання вольт-амперної характеристики фотодіоду. Величину темнового струму ФЧЕ фотодіоду I_{Ti} розраховують за формулою

$$I_{Ti} = \frac{U_{Hi}}{R_{Hi}}, \quad (2.9)$$

де U_{Hi} – спад напруги на опорі навантаження;

R_{Hi} – опір навантаження на ФЧЕ.

В загальному випадку темновий струм, що протікає через р-п перехід (I_t), визначається сумою дифузійного струму в нейтральній області (I_D) та генераційного в збідненій області (I_G) [67]:

$$I_t = I_D + I_G. \quad (2.10)$$

Але, крім згаданих, загально відомих чинників що впливають на величину темнового струму, існує ще декілька, пов'язаних з особливістю технологічних процесів, за допомогою яких виготовляються сучасні кристали р-п кремнієвих фотодіодів, до цього можна віднести генерацію носіїв струму на поверхні розподілу кремній – окисел кремнію, генерацію носіїв струму торцевою поверхнею кристалу, генерацію носіїв струму на зворотному боці кристалу та ін., що вносить ще більший хаос у непередбачуваність та випадковість генерації.

Іншими словами, вся зовнішня поверхня кристалу р-п фотодіоду може бути джерелом носіїв зворотного (темнового) струму. Всі ці згадані чинники генерують струм за межами розповсюдження збідненої області, але здатні досягати її за рахунок дрейфу.

Джерелом носіїв струму, які здатні збільшувати темновий струм, може бути також і торцева поверхня кристалу фотодіода, яка утворюється під час вирізання топології фотодіоду з кремнієвої пластини [68].

Таким чином, з огляду на фізичні процеси, які зумовлюють величину темнового струму фотодіоду, він, темновий струм, може бути розглянутий як джерело генерації випадкової послідовності для створення відповідного генератора методами мікроелектроніки та програмної інженерії [69].

Блок-схема вимірювання темнового струму фотодіода наведена на рис. 2.8.

Як видно з рисунку 2.8, блок-схема для вимірювання темнового струму досить проста. Основна вимога до неї - забезпечення відсутності будь-якого

освітлення фотодіоду.

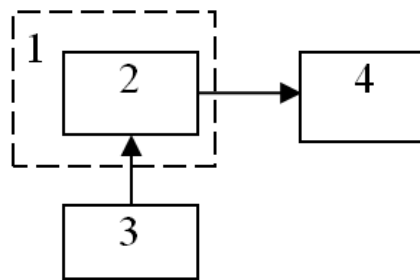


Рисунок 2.8 – Блок-схема установки для вимірювання темного струму фотодіода.

1 – світлонепроникний екран; 2 – фотодіод; 3 – блок живлення;
4 – вольтметр постійного струму.

Створення ПВЧ з значень темного струму полягає у наступному.

Задача одержання цифрового сигналу у комп'ютер з аналогового джерела, яким є звичайний фотодіод, вирішена давно. Це може бути USB-адптер на базі Arduino (112593), або навіть лінійний аудіо-вхід мікрофона [70]. В нашому випадку використовувався кремнієвий фотодіод ФД-288.

Темновий струм генерується у деякому діапазоні 0 .. 10, мА, що відповідає напрузі 0..100 мВольт. Роздільна здатність Arduino складає 5 mV. Таким чином весь діапазон 0 .. 100 мВ можна розділити на 10 піддіапазонів, і кожен з них буде відповідати числу в діапазоні 0 ... 9.

Для програмної розробки використовувалась мова програмування Java версії 17 [71], зокрема бібліотека `javaх-usb`. Величини виміряної напруги вводяться в комп'ютер через USB-порт і складаються у список у вигляді випадкових чисел [0..9]. У подальшому цей список випадкових чисел досліджується на відповідність статистичним характеристикам за допомогою бібліотеки `DescriptiveStatistics` [72].

Для графічного відображення статистичних характеристик використовувалась бібліотека `jfree.chart` [73] та `jfree.data` [74].

На рис. 2.9 приведена гістограма розподілення випадкових чисел по значенню (спектр).

На рис. 2.9 видно, що 50 % всіх чисел це п'ятірки, четвірки і шестірки взяли по 20 %, трійки - 4 % і т.д. З гістограми видно, що розподіл носить нормальний (гаусівський) характер.

Таким чином одержаний ряд випадкових чисел у діапазоні [0 .. 9]. Швидкість генерації, або продуктивність (P) у такому випадку становитиме для Arduino: $P = 980$ байт/сек.

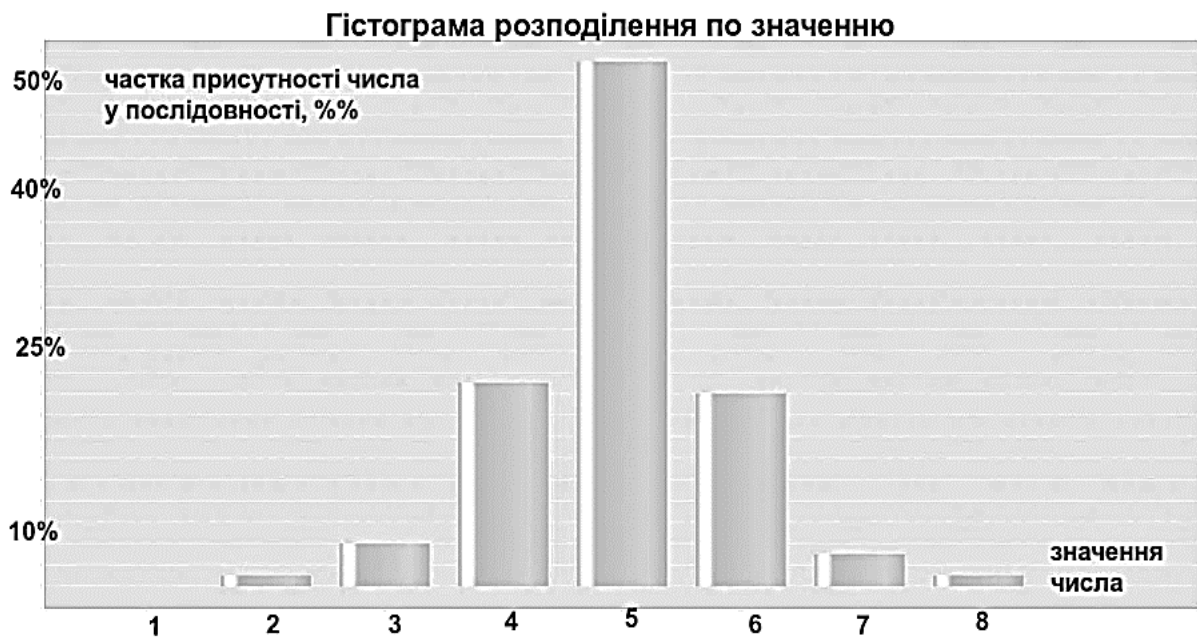


Рисунок 2.9 – Гістограма розподілення по значенню випадкових чисел, що одержані з фотодіоду.

Для зручності аналізу випадкові числа у подальшому були згенеровані у діапазоні [0 .. 255], що відповідає типу даних byte мови програмування Java.

Для забезпечення такого діапазону варто підсилити сигнал темнового струму до 5 Вольт (верхня межа Arduino).

У цьому випадку роздільна здатність отриманого сигналу буде складати: $5.0 \text{ Вольт} / 1024 = 5 \text{ мілівольт}$.

Таким чином вимір у діапазоні [0..5] мілівольт буде відповідати нульовому байту, а вимір у діапазоні [4.995..5.0] Вольт — 255-ому байту.

У нашому експерименті протягом 100 сек було згенеровано ПВЧ розміром у 98 кілобайт. На рис. 2.10 наведено гістограму розподілу по значенню випадкових чисел, що згенеровані за допомогою генерації темного струму фотодіода.

На гістограмі (рис. 2.10) видно, що підсилювач сигналу вніс свій додатковий хаос у процес генерації випадкових чисел. Тепер отриманий характер розподілу є «рівномірно-хаотичний». Рівномірність - кожне число представлено приблизно рівномірно у діапазоні 0.3 — 0.7 %. Хаотичність полягає у тому, що нема чітко виражених мінімумів чи максимумів, як у випадку з гаусовою кривою. Нагадаємо, що ідеальна частка для кожного числа $1/256 = 0.004$, або 0.4%.

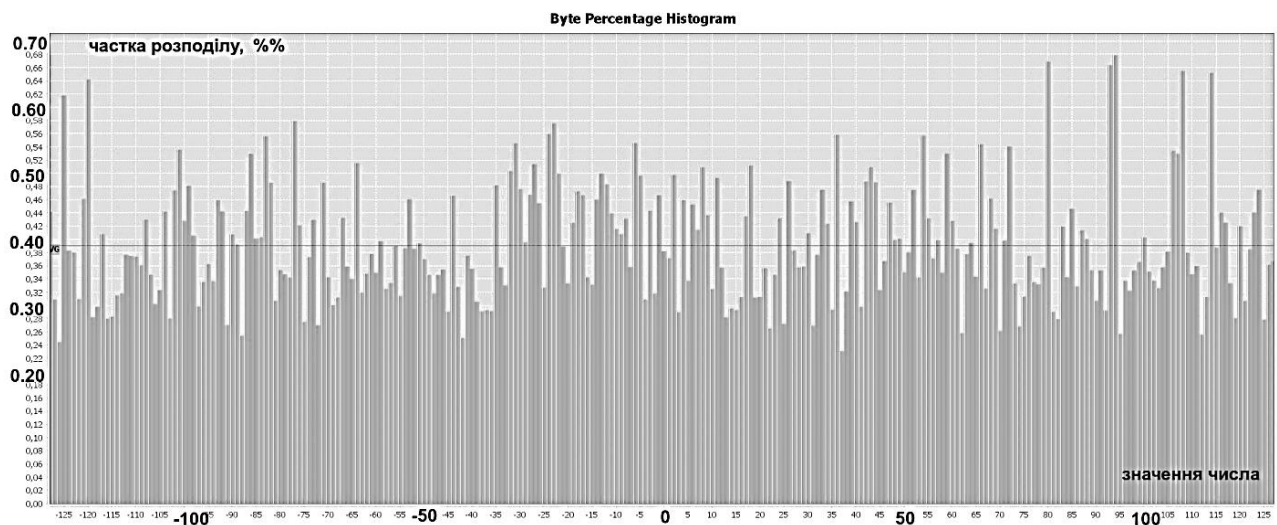


Рисунок 2.10 – Гістограма розподілу по значенню випадкових чисел , що згенеровані ФД.

В результаті наведеного дослідження вивчена доцільність застосування в якості джерела ПВЧ темновий струм фотодіода на основі кремнію, який працює у фотодіодному режимі. Показано, що фотодіод може бути джерелом ентропії

для генерації ПВЧ. Такий розподіл є рівномірно-хаотичним і кожне число представлено приблизно рівномірно у діапазоні 30 – 70 %.

2.4. Дослідження генерації ПВЧ на основі темного струму фотодіода при різних температурах

Раніше, у попередньому параграфі, було показано результати дослідження можливості та ефективності генерації ПВЧ, заснованої на фіксації темного струму кремнієвого фотодіода ФД-288 [69]. Ці дослідження показали, що ряд випадкових чисел у діапазоні $[0 \dots 9]$, які відповідають значенням темного струму фотодіода, і цифровані за допомогою USB-адптера на базі Arduino (112593), мають швидкість генерації, інакше - продуктивність $P = 980$ байт/сек. Що є дуже мало. Основи інформаційних технологій вимагають, щоби випадкові числа хаотичної послідовності були у діапазоні $[0 \dots 255]$. Це відповідає типу даних `byte`, які притаманні мови програмування Java. Підсилення сигналу темного струму на 5 Вольт, переводить характеристику розподілу по значенню псевдовипадкових чисел у бік «рівномірно-хаотичної».

З урахуванням вище наведеного, було запропоновано підвищити температуру фотодіода для згладжування розкиду значень темного струму [75]. Було проведено вимірювання темного струму фотодіода ФД-288 при температурі 10, 30 та 50 °C, що відповідає умовам його експлуатації за допомогою установки, блок-схема якої наведена на рисунку 2.11.

Температуру на фотодіоді змінювали у діапазоні 10 - 50 °C за допомогою термоелектричного охолоджувача.

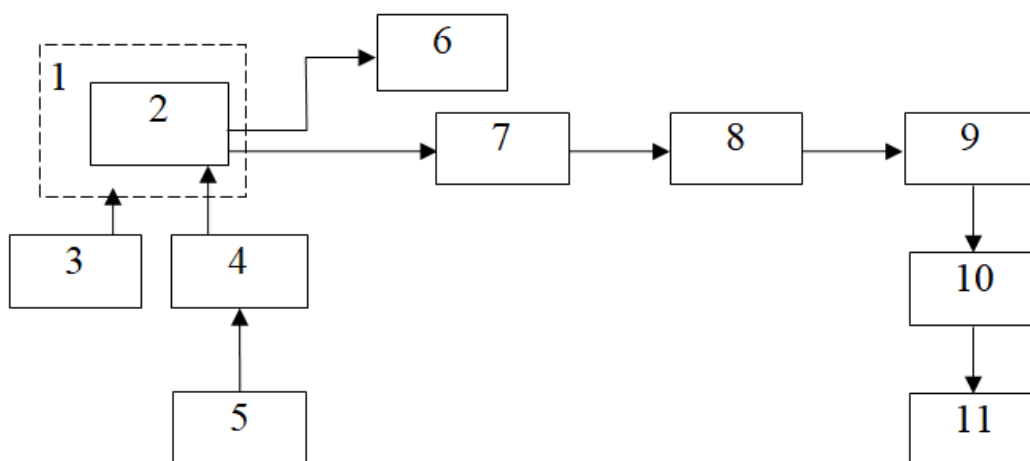


Рисунок 2.11 – Блок-схема установки для вимірювання темного струму фотодіода ФД-288 при температурі 10, 30 та 50 °С.

1 - світлонепроникний екран, 2 - фотодіод, 3 - блок живлення фотодіода, 4 - термоелектричний охолоджувач, 5 - блок живлення термоелектричного охолоджувача, 6 - термометр цифровий Тензор-42, 7 - ППЧН, 8 - контролер Arduino (112593), 9 - USB-адаптер, 10 - USB порт комп'ютера, 11 - процесор комп'ютера.

Зменшення чи збільшення температури на фотодіоді здійснювали змінюючи полярність живлення на охолоджувачі. Температура фотодіода контролювалася в цокольній області його корпусу, за допомогою термометра цифрового Тензор-42, який забезпечує вимірювання температури у діапазоні 0 – 200 °С із дискретністю показів 0,01 °С із межею основної відносної похибки $\pm 0,05$ °С. Темновий струм вимірювався при зміщенні на р-п переході фотодіоду 1 В.

Далі, виміняні значення темного струму у режимі on-line за допомогою прецизійного перетворювача струм-напруга (ППЧН) перетворюються у напругу (U_T) і, використовуючи USB-адаптер на основі Arduino (112593), передавалися до комп'ютера, як це показано на рис. 2.11. В комп'ютері значення U_T відображалися у вигляді гістограм.

Далі, за допомогою мови програмування Java (версія 17) [71], із використанням бібліотеки `javax-usb` [76], розроблений алгоритм генерації ПВЧ, основних на значеннях темного струму фотодіода, блок-схема якого показана на рис. 2.12.

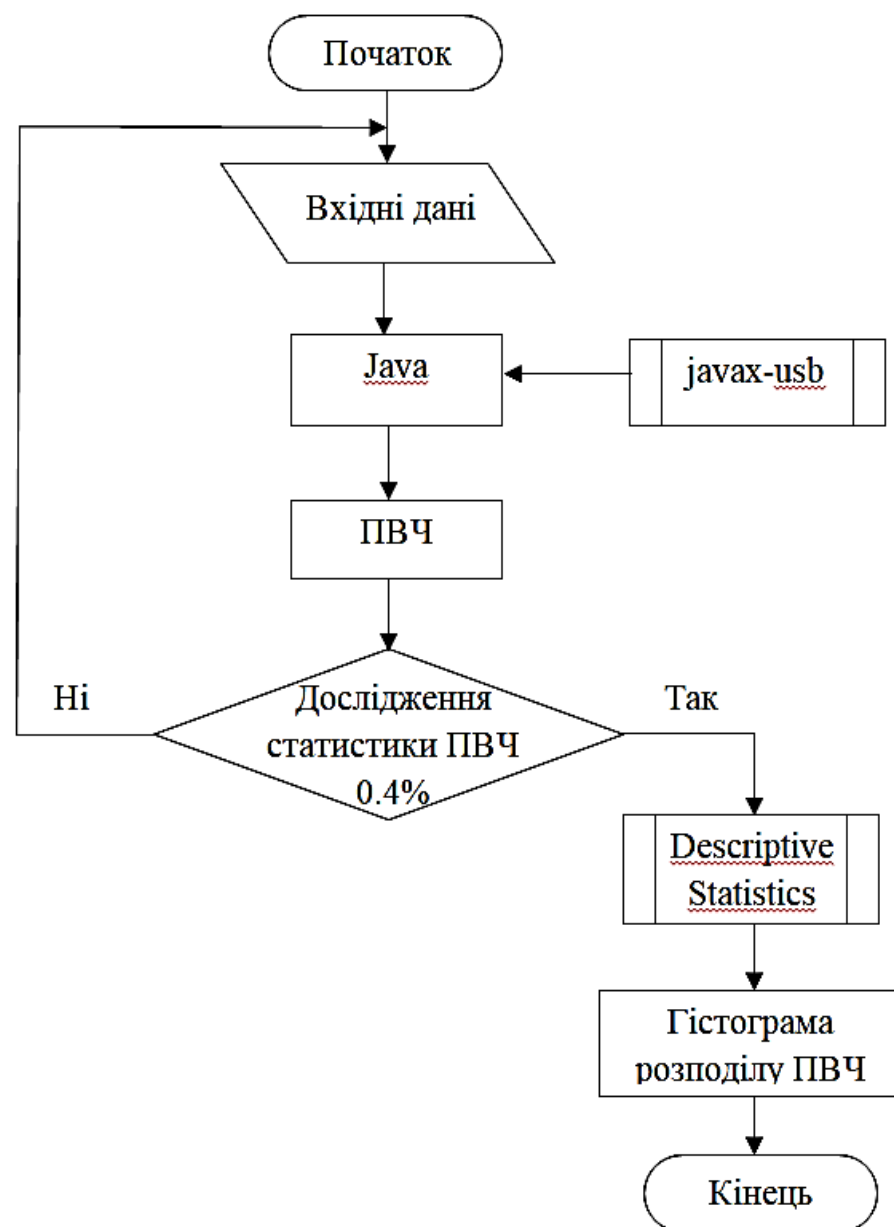


Рисунок 2.12 – Блок-схема алгоритму генерації псевдовипадкової послідовності чисел.

У подальшому цей список випадкових чисел досліджується на відповідність статистичним характеристикам за допомогою бібліотеки DescriptiveStatistics [72]. Для графічного відображення статистичних характеристик використовувалась бібліотека jfree.chart [73] та jfree.data [74]. При кожній температурі було виміряне по 20000 значень темного струму в секунду.

Динаміка темного струму при трьох температурах – 10, 30 та 50 °C наведена на рисунках. 2.13, 2.14, 2.15.

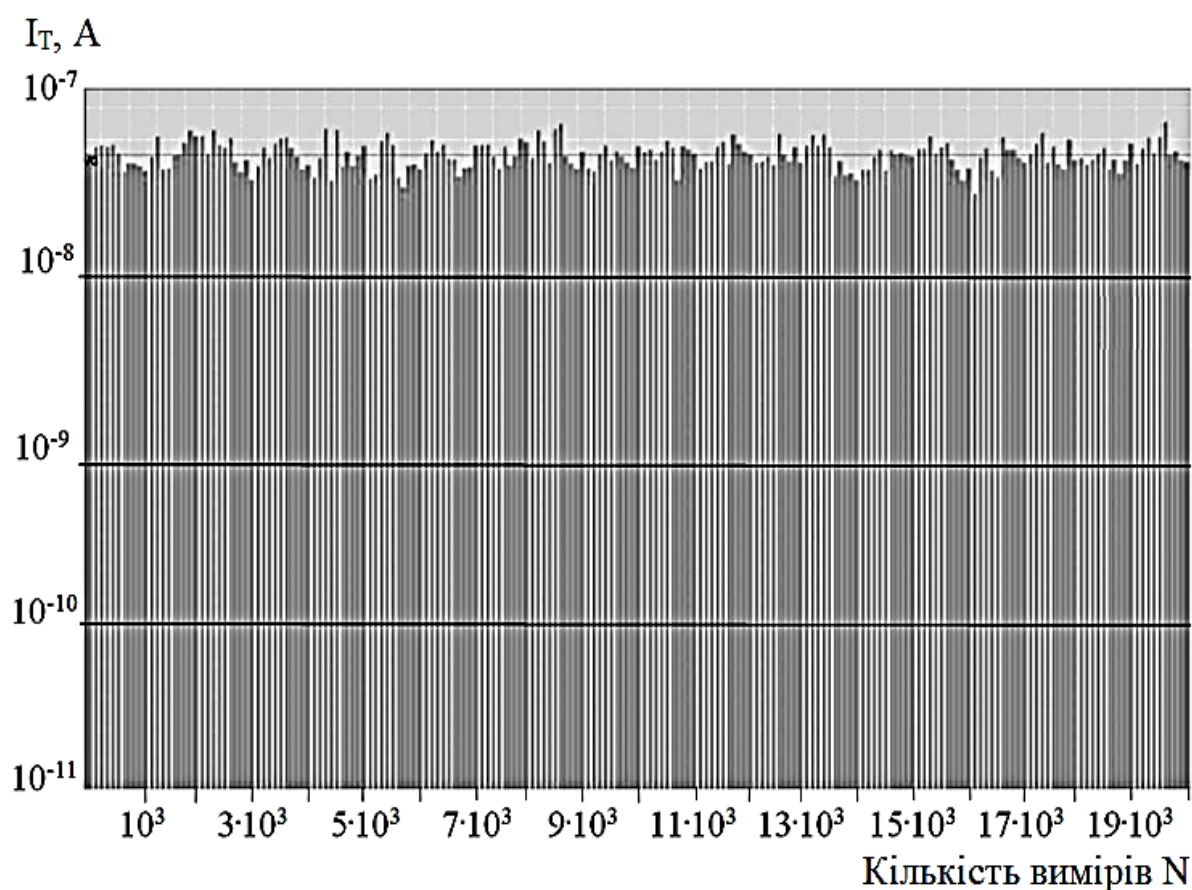


Рисунок 2.13 – Динаміка темного струму при температурі 50 °C.

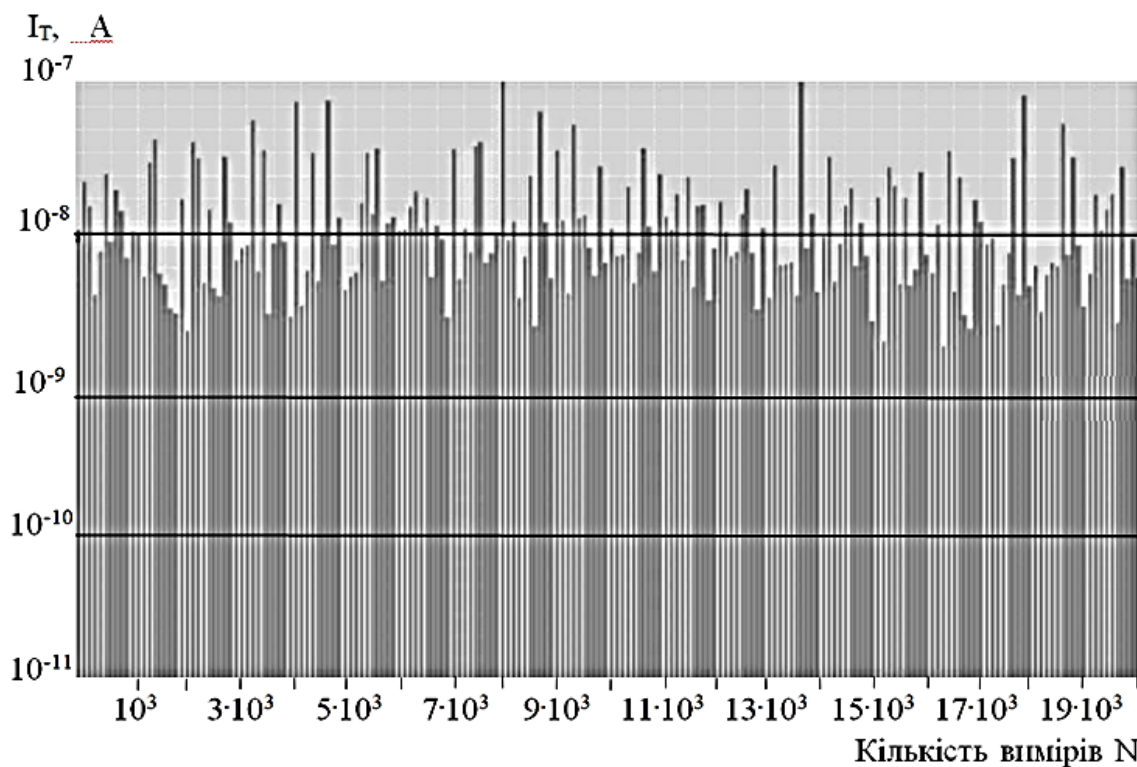


Рисунок 2.14 – Динаміка темного струму при температурі 30 °С.

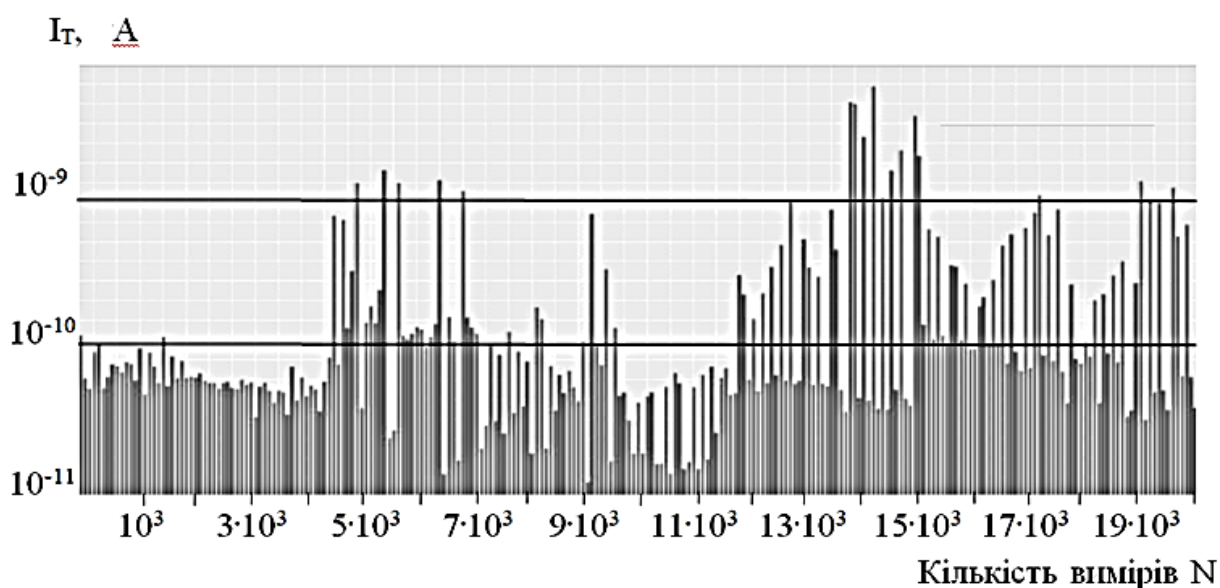


Рисунок 2.15 – Динаміка темного струму при температурі 10 °С.

З рисунків 2.13 – 2.15 видно, що при температурі 10 °С діапазон вимірних значень темного струму складає від 10^{-11} до $8 \cdot 10^{-9}$ А, при температурі 30 °С діапазон вимірних значень темного струму складає від

10^{-11} до $8 \cdot 10^{-7}$ А, а при температурі 50 °С діапазон вимірюваних значень темнового струму складає від 10^{-11} до $7 \cdot 10^{-7}$ А. При цьому візуально помітно, що найбільший розкид значень спостерігається при 10 °С, а найменший при 50 °С.

З отриманого набору даних були згенеровані ПВЧ згідно алгоритму, наведеному на рис. 2.12 Для отримання випадкових чисел з цього процесу генерації шкалу фотоструму необхідно розділити на набір послідовних інтервалів, кожен з яких відповідає певному цілочисельному значенню. Таким чином, кожне вимірне значення фотоструму зіставляється з певним числом, періодичне вимірювання струму створює послідовність випадкових цілих чисел..

На рис. 2.16. наведена гістограма розподілення випадкових чисел по значенню для температури 10 °С.

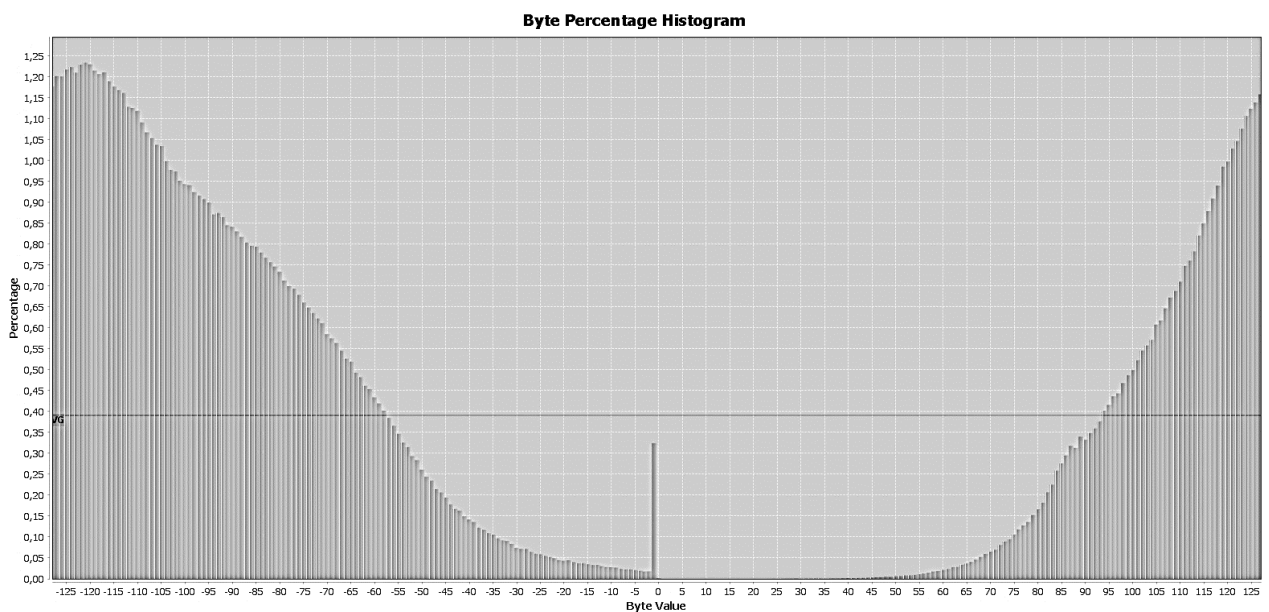


Рисунок 2.16 – Гістограма розподілення випадкових чисел, що були згенеровані, по значенню величини для температури 10 °С.

З рисунку видно, що не всі елементи випадкової послідовності (байти) представлені з однаковою частотою. Байти у діапазоні [5 ... 25] взагалі відсутні у послідовності, а байти у діапазоні [120 ... 125] займають приблизно 1,2 %

частки від усіх елементів послідовності. Хоча всі числа в силу стохастичності процесу носять абсолютно випадковий характер, такий розподіл не є задовільним з точки зору вимог до статистичних характеристик ПВЧ. На рис. 2.17 наведена гістограма розподілу випадкових чисел по значенню для температури 30 °С.

Для температури 30 °С видно з гістограми, що розподілення дещо вирівнялась – немає відсутніх елементів. Проте середнє квадратичне відхилення у майже 50%, як буде показано нижче, не дозволяє нам віднести тип розподілу до рівномірного.

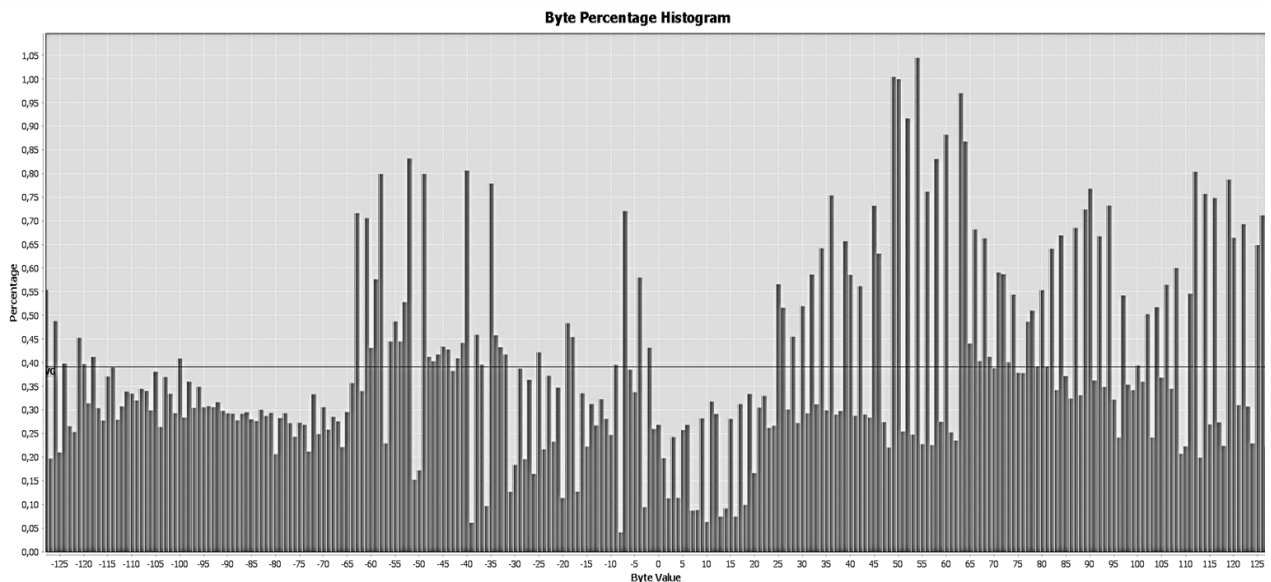


Рисунок 2.17 – Гістограма розподілення випадкових чисел, що були згенеровані, по значенню величини для температури 30 °С.

На рис. 2.18 наведена гістограма розподілення випадкових чисел по значенню для температури 50 °С.

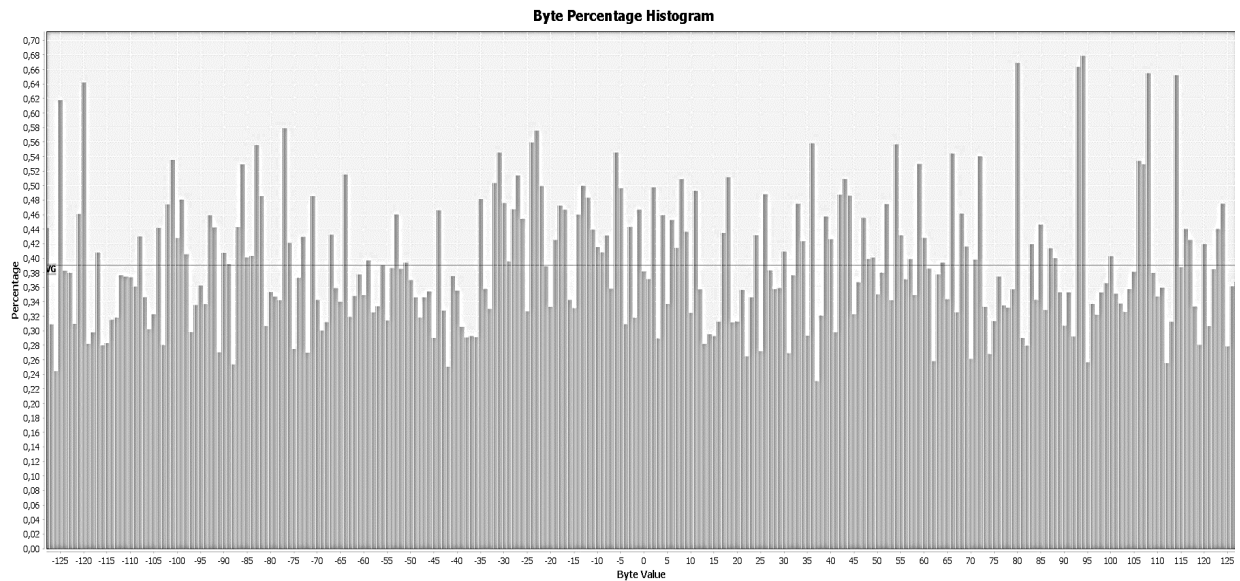


Рисунок 2.18 – Гістограма розподілення випадкових чисел, що були згенеровані, по значенню величини для температури 50 °C.

Для температури 50 °C видно з гістограми, що розподілення дещо стала плавною. Статистичні характеристики, які будуть викладені нижче, дають нам змогу характеризувати цей розподіл, як рівномірно-хаотичний. І для деяких випадків (особливо у стеганографії) такий розподіл є цілком задовільний.

Проте візуальний огляд гістограми не дає нам кількісних показників якості згенерованих ПВЧ. За допомогою мови Java, зокрема статистичної бібліотеки `DescriptiveStatistics`, вдалось обчислити параметри, що характеризують якість згенерованої ПВЧ для кожної температури.

Порівняльні статистичні характеристики згенерованих послідовностей, отриманих з значень темного струму прирізних температурах, наведені у таблиці 2.1.

Нагадаємо, що для ідеального рівномірного розподілу [77]:

- мінімальне значення $\min = 0.39$;
- максимальне значення $\max = 0.39$;
- середнє значення $\text{mean} = 0.39$;
- середньоквадратичне відхилення $\text{st dev} = 0.04$;

- процент відсутніх елементів absent = 0%.

Таблиця 2.1.

Статистичні характеристики згенерованих послідовностей.

Температура	10°C	30°C	50°C	Вимоги ідеального розподілу ПВП
Статистичні параметри розподілу, %	min: 0.0 max: 1.27 mean: 0.39 std dev: 0.85 absent: 8%	min: 0.10 max: 1.05 mean: 0.39 std dev: 0.33 absent: 0	min: 0.24 max: 0.68 mean: 0.39 std dev: 0.18 absent: 0	min: 0.39 max: 0.39 mean: 0.39 st dev: 0.04 absent: 0

Самою показовою характеристикою є середнє квадратичне відхилення. Чим менше відхилення, тим менше розкид значень, тим більш рівномірним є розподіл.

Як видно з таблиці 2.1, середнє квадратичне відхилення зменшується із зростанням температури. Зокрема встановлено, що найкращі результати досягнути при генерації випадкової послідовності чисел на основі темного струму фотодіода, отриманої при 50 °C. А саме - мінімальне значення складає 0,0 при вимозі 0.39; максимальне значення складає 0.68 при вимозі 0.39; середнє значення 0.39 при вимозі 0.39; середньоквадратичне відхилення 0.18 при вимозі 0.04; процент відсутніх елементів 0 при вимозі 0.

З цього видно, що наближче відповідає практичним вимогам застосування фотодіод при температурі 50 °C як джерело ПВЧ. При цьому розподіл чисел є рівномірно-хаотичним, а кожне число рівномірно представлене не частіше 0.18 , що менше за ідеальні умови, але достатньо для практичного застосування, наприклад у стеганографії.

Встановлено, що найкращі результати досягнуто при генерації випадкової послідовності чисел на основі темного струму фотодіода,

отриманої при 50 °С, а саме: мінімальне значення складає 0,24 при вимозі 0.39; максимальне значення складає 0.68 при вимозі 0.39; середнє значення 0.39 при вимозі 0.39; середньоквадратичне відхилення 0.18 при вимозі 0.04; процент відсутніх елементів 0 при вимозі 0.

Показано, що фотодіод, при температурі 50 °С, може бути джерелом для генерації послідовності псевдовипадкових чисел. При цьому розподіл чисел є рівномірно-хаотичним, а кожне число рівномірно представлене не частіше 0.18 %.

Продуктивність створеного генератора є невисокою – до 98 Кбіт/сек і залежить від можливостей апаратної частини, а саме від її швидкодії. Із застосування підсилювачів та контролерів, призначених для забезпечення передачі відео-зображень на частоті наприклад 12 МГц, продуктивність генератора може бути збільшена відповідно до кількох десятків Мбіт/сек.

2.5. Гібридний генератор ПВЧ із застосуванням КА

Оскільки параметри фотодіода не дають змоги побудувати безпосередньо на ньому без допоміжних обв'язок високопродуктивний генератор ПВЧ, то ми далі будемо вдосконалювати статистику, обробляючи вхідну послідовність програмними методами, наприклад функціоналом клітинних автоматів або функцією хешування.

Для цього розглянуто питання застосування клітинних автоматів хаотичного типу для покращення статистичних характеристик створюваних ПВЧ [78]. Така модернізація підпадає під визначення гібридних генераторів ПВЧ. Це питання було проаналізовано на прикладі генератора ПВЧ на основі веб-камери.

Для згладжування недоліків «сирої» генерації веб-камери (незадовільні статистичні характеристики, а саме, не рівномірний розподіл елементів по значенню), було запропоновано використати функціональність клітинних автоматів (КА).

Одномірні клітинні автомати хаотичних правил (30, 90, 105, 150) здатні обробляти упорядковану послідовність бітів в хаотично-керовану послідовність без змоги «відкрити назад» (roll-back). Така трансформація поліпшує статистичні характеристики ПВЧ.

З [56] відомо, що найефективнішим хаотичним правилом взаємодії одновимірних КА є правило 30. На рис. 2.19 подано серветку Серпінського для правила 30, яка демонструє перетворення за цим правилом [56].

Однак цей процес вимагає значного часу. Вже через це після 10 ітерацій швидкість генерації зменшилася в 10 разів. У нашому експериментальному випадку (режим SVGA, 25 кадрів/с) швидкість генерації знизилася з 288 Мбіт/с до 20 Мбіт/с.

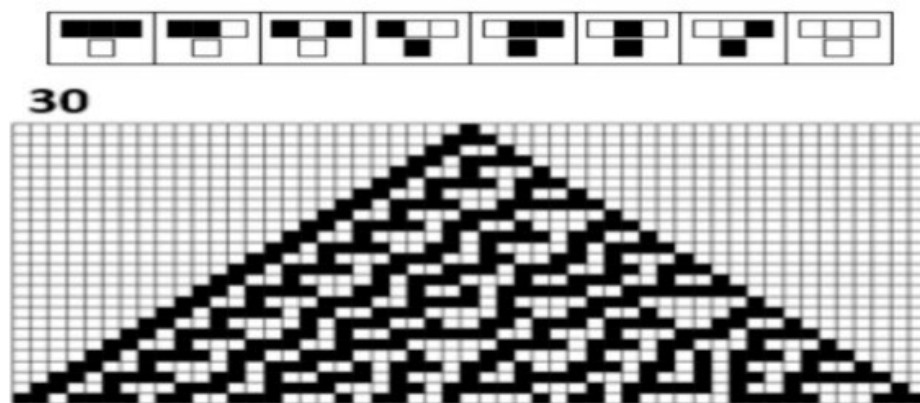


Рисунок 2.19 – Серветка Серпінського.

Пояснення алгоритму правила 30 КА.

Ефективність використання КА у гібридному генераторі ПВЧ базується насамперед на тій властивості, що КА у нашому випадку це примітивна дискретна самоорганізуюча модель, і як наслідок це найшвидші операції, що споживають мінімум ресурсів для бітових перетворень. Тому їх можна застосовувати навіть на самих мінімальних обчислювальних ресурсах включаючи смартфони і мікроконтролери. Вони не вимагають великих і

потужних бібліотек, а їх комбінації здатні генерувати хаос високого рівня з мінімального початкового значення.

На сучасному етапі розвитку програмної інженерії програмні генератори ПВЧ для генерації послідовності певної довжини вимагають певну затравку (seed). Чим більший розмір затравки, тим довше період зациклення ПВЧ. Після виснаження затравки рекомендується замінити її на іншу для уникнення повтору циклу.

У випадку функціоналу КА можна говорити, що в цьому разі генерація ПВЧ зовсім не потребує вхідного значення – період зациклення нескінченний.

Гібридний генератор на основі фізичного джерела ентропії і функціоналу КА для пост-обробки вирішує питання по швидкодії, надійності, контрольованості, відповідності статистичним вимогам та вимогам до безпеки стабільності пост-обробки.

Програмна частина гібридного генератора відкриває можливості по масштабуванню та паралелізації процесів обробки. Таким чином швидкість генерації можливо збільшити в кілька разів по числу відкритих потоків, наприклад у мові програмування Java, де кількість потоків визначається налаштуванням JVM і може підвищити продуктивність генератора на порядок.

Згенеровані послідовності можуть бути відсортовані згідно їх статистичних характеристик (тести NIST) та збережені у базі даних, що дає змогу споживачу вибрати ПВЧ заданої якості і кількості.

2.6. Порівняльний аналіз розроблених генераторів ПВЧ

2.6.1. Веб-камера, як джерело ПВЧ

Однією з ключових переваг даного підходу до створення ПВЧ за допомогою відео, або веб-камери є істинна фізична випадковість, обумовлена природою шумових процесів у фоточутливих сенсорах. Це дозволяє

отримувати непередбачувані послідовності, які принципово неможливо відтворити за допомогою суто математичних або алгоритмічних методів.

Важливою практичною перевагою є доступність та низька вартість реалізації. Веб-камери є масовими пристроями, інтегрованими у більшість сучасних ноутбуків, мобільних телефонів і персональних комп'ютерів, що усуває необхідність використання спеціалізованих апаратних модулів.

Крім того, такі генератори характеризуються гнучкістю програмної реалізації. Алгоритми обробки відеопотоку можуть бути адаптовані під конкретні вимоги до швидкодії, обсягу ентропії та рівня криптографічної стійкості. Це робить веб-камерні генератори ПВЧ привабливими подальшого дослідження і застосування.

2.6.1.1. Недоліки та обмеження створення ПВЧ за допомогою веб-камери

Незважаючи на концептуальну привабливість використання веб-камер як джерела фізичної ентропії, такі генератори випадкових чисел характеризуються низкою суттєвих обмежень, які обмежують сферу їх практичного застосування, особливо в задачах криптографічної безпеки.

Передусім, якість ентропії істотно залежить від умов зовнішнього середовища. За стабільного освітлення, відсутності руху в кадрі та незмінної сцени рівень шумової складової різко зменшується, що призводить до зростання кореляції між послідовними кадрами.

У таких умовах веб-камера фактично перестає бути надійним джерелом випадковості, оскільки значна частина сигналу набуває детермінованого характеру.

Обмежена пропускна здатність та низька швидкість генерації. Кількість доступної ентропії напряму залежить від частоти кадрів, роздільної здатності сенсора та продуктивності обчислювальної підсистеми. На відміну від спеціалізованих апаратних TRNG, які здатні генерувати випадкові біти зі

швидкістю в десятки або сотні мегабіт за секунду, «сирі» (необроблені) рішення на основі веб-камери зазвичай забезпечують значно нижчі показники, що унеможлиблює їх використання у високонавантажених системах. Зазвичай вузьким місцем являється пропускна здатність USB-порта.

Окремої уваги заслуговує складність коректного відокремлення шуму від корисного сигналу. Наявність шуму в зображенні сама по собі не гарантує високої ентропії. Без належних алгоритмів фільтрації, усунення зміщення (debiasing) та криптографічної пост-обробки отримані дані можуть демонструвати статистичні відхилення від рівномірного розподілу, що виявляється під час застосування стандартних тестів випадковості.

Крім того, генератори на основі веб-камер погано піддаються формальній валідації та сертифікації. Висока залежність від апаратних характеристик конкретної камери, драйверів, налаштувань експозиції та програмного середовища ускладнює побудову відтворюваної моделі загроз і унеможлиблює гарантування стабільних параметрів ентропії у різних інсталяціях.

Не менш важливим недоліком є енергетична та обчислювальна неефективність. Безперервне захоплення та обробка відеопотоку потребують значних ресурсів центрального процесора або графічного прискорювача, що робить такі генератори малопридатними для вбудованих систем та пристроїв з обмеженим енергоспоживанням.

Таким чином, генератори ПВЧ на основі веб-камер, попри здатність забезпечувати фізично зумовлену випадковість, характеризуються комплексом технічних, експлуатаційних та безпекових недоліків. Проте вказані недоліки не вважаються фатальними, а скоріше як «вузькі місця», які можуть бути технічно подолані згідно поставленої задачі. Одним зі способів виділення випадкової складової з яскравості зображення є високочастотна фільтрація Лапласа, яка дає змогу відкинути детерміновану складову яскравості. Застосування такої фільтрації та КА потенційно дозволяє отримати ще більш високоентропійну послідовність випадкових чисел, ніж застосування тільки КА. Проте, для

багатьох прикладних завдань застосування тільки КА забезпечує достатній рівень ентропії.

2.6.1.2. Сфера застосування ПВЧ, отриманих за допомогою веб-камери

Генератори ПВЧ, отримані на основі веб-камер доцільно використовувати у навчальних та демонстраційних системах, для експериментальних досліджень ентропії фізичних процесів, як допоміжне джерело ентропії для ініціалізації криптографічних алгоритмів, у системах з низьким або середнім рівнем вимог до безпеки. Водночас їх не рекомендується застосовувати як єдине джерело випадковості в критично важливих криптографічних системах без додаткових механізмів контролю та комбінування з іншими джерелами ентропії.

Отже, використання веб-камери як джерела ентропії для генерації ПВЧ є технічно простим і економічно доцільним підходом, який забезпечує фізично обґрунтовану випадковість. Проте нестабільність умов зйомки та обмеження швидкодії зумовлюють необхідність ретельної постобробки та комбінування з іншими методами. У науково-дослідному та освітньому контексті даний підхід є перспективним і наочно демонструє принципи побудови апаратних генераторів випадкових чисел.

2.6.2. Фотострум фотодіода як джерело ентропії

2.6.2.1. Переваги генераторів ПВЧ на основі фотоструму фотодіодів

1. Однією з ключових переваг генераторів випадкових чисел, побудованих на основі фотодіодів, є фундаментальна фізична природа джерела ентропії. Випадковість формується за рахунок квантових флуктуацій фотонного потоку та дискретності процесів переносу заряду в напівпровіднику, що створює фотострум. Це робить такий підхід принципово недетермінованим і несприйнятливим до алгоритмічного передбачення.

2. Важливою характеристикою є висока якість статистичних властивостей шуму, зокрема дробового шуму, який за певних умов наближається до ідеального білого шуму. Це дозволяє отримувати бітові послідовності з високою ентропією на біт після відносно простої пост-обробки та забезпечує успішне проходження стандартних тестів випадковості.

3. На відміну від візуальних або середовище-залежних джерел ентропії, фотодіодні генератори характеризуються стабільністю параметрів у контрольованих умовах експлуатації. Використання фіксованого джерела світла або роботи фотодіода в режимі темного струму дозволяє мінімізувати вплив зовнішніх факторів і досягти відтворюваних характеристик ентропії.

4. Ще однією суттєвою перевагою є порівняно висока швидкодія. Частотні характеристики фотодіодів і сучасних підсилювальних каскадів дають змогу формувати випадкові послідовності зі швидкістю від сотень кілобіт до десятків мегабіт за секунду, що істотно перевищує можливості генераторів, заснованих на обробці зображень або повільних фізичних процесах. В нашому випадку було досягнуто значення до 2 Мбіт/сек, що не є граничним значенням, і може бути нарощено, але в подальшому ПВЧ з фізичного джерела планується використовувати як затравку для гібридного генератора, де швидкість генерації затравки не грає великої ролі. З огляду на зазначене, генератори ПВЧ на основі фотодіодів вважаються придатними для криптографічних застосувань, зокрема як первинні джерела ентропії для ініціалізації ключів, протоколів автентифікації та апаратних модулів безпеки.

2.6.2.2. Недоліки та обмеження генераторів ПВЧ на основі фотоструму фотодіода

Попри значні переваги, генератори на основі фотоструму фотодіода мають низку істотних недоліків, які ускладнюють їх практичну реалізацію.

1. Передусім це висока складність апаратної частини. Для виділення корисної шумової складової необхідні малошумні підсилювачі, стабільні

джерела живлення та високоточні аналого-цифрові перетворювачі, що підвищує вимоги до схемотехнічного проектування.

2. Суттєвим обмеженням є чутливість до електромагнітних та температурних впливів. Зміни температури, паразитні наводки та флуктуації напруги живлення можуть істотно змінювати спектральні характеристики шуму, що знижує ефективну ентропію та потребує додаткових механізмів компенсації.

3. Крім того, фотодіодні генератори потребують регулярного калібрування та моніторингу параметрів. Інтенсивність світла, напруга зсуву фотодіода та робоча температура безпосередньо впливають на співвідношення корисного шуму та детермінованих компонентів сигналу. Без належного контролю ці фактори можуть призвести до деградації якості випадковості.

4. Окремої уваги потребує ризик формування псевдоентропії внаслідок некоректної реалізації. Якщо детерміновані складові фотоструму домінують над шумовими, або якщо пост-обробка реалізована неналежним чином, вихідна послідовність може демонструвати приховані кореляції, що створює хибне відчуття криптографічної стійкості.

Таким чином, генератори ПВЧ на основі фотоструму фотодіода поєднують високу якість фізично зумовленої ентропії, стабільність та достатню швидкодію, що робить їх перспективними для використання в криптографічно чутливих системах. Водночас складність апаратної реалізації, необхідність калібрування та підвищені вимоги до захисту від завад обмежують їх застосування випадками, де якість випадковості є критично важливою та виправдовує додаткові витрати.

2.6.2.3. Сфера застосування генераторів ПВЧ на основі фотоструму фотодіода

Проведений аналіз показує, що генератори ПВЧ на основі фотоструму фотодіода, займають особливе місце серед апаратних джерел ентропії,

поєднуючи фундаментальні фізичні принципи з практичною придатністю до використання в сучасних інформаційних системах. Квантово зумовлений характер дробового шуму та супутніх теплових флуктуацій забезпечує істинну, фізично обґрунтовану випадковість, яка принципово відрізняється від алгоритмічної псевдовипадковості та не піддається детермінованому відтворенню.

На відміну від середовище-залежних джерел ентропії, зокрема візуальних генераторів на основі веб-камер, генератори ПВЧ на основі фотоструму фотодіода, характеризуються вищим рівнем стабільності та керованості параметрів. Можливість роботи в контрольованих умовах із фіксованим джерелом світла або в режимі темного струму дозволяє досягати відтворюваних статистичних характеристик і прогнозованого рівня ентропії, що є критично важливим для криптографічних та апаратно-захищених систем.

Також, високий рівень якості випадковості досягається складністю апаратної реалізації, точного схемотехнічного проектування, регулярного контролю параметрів. Чутливість до температурних та електромагнітних впливів, а також ризик деградації ентропії у разі некоректної пост-обробки зумовлюють потребу в комплексному підході до проектування, який поєднує апаратні засоби з криптографічно стійкими методами аналізу та «очищення» шуму.

У контексті практичного застосування генератори ПВЧ на основі фотоструму фотодіода доцільно розглядати як ключовий компонент гібридних систем генерації ентропії. Поєднання фотодіодного TRNG з іншими фізичними та програмними джерелами дозволяє підвищити надійність, стійкість до атак та відповідність вимогам міжнародних стандартів у сфері інформаційної безпеки.

Таким чином, генератори ПВЧ на основі фотоструму фотодіода є технологічно зрілим і перспективним рішенням для систем, у яких вимоги до якості та фізичної обґрунтованості випадковості переважають над простотою реалізації та вартістю. Їх застосування є найбільш виправданим у криптографії,

апаратних модулях безпеки, квантових технологіях і критично важливих інформаційних системах, де компрометація джерела випадковості може мати системні наслідки. Водночас подальший розвиток методів калібрування, моніторингу ентропії та інтеграції з програмними механізмами безпеки визначає перспективні напрями вдосконалення даного класу генераторів у майбутніх дослідженнях.

2.6.3. Генератор ПВЧ на основі темнового струму фотодіода

2.6.3.1 Переваги генератора ПВЧ на основі темнового струму фотодіода

Основною перевагою генераторів, що використовують темновий струм фотодіода для генерації ПВЧ, є фундаментальна фізична природа джерела ентропії. Темновий струм формується внаслідок теплової генерації носіїв заряду, тунельних ефектів та флуктуацій енергетичних рівнів у напівпровіднику. Ці процеси мають стохастичний характер і не піддаються алгоритмічному прогнозуванню, що забезпечує істинну, а не псевдовипадкову природу генерованих чисел.

На відміну від генераторів, заснованих на фотострумі або візуальних даних, темновий струм формується за повної відсутності освітлення, що практично усуває можливість зовнішнього оптичного впливу. Це значно зменшує площину атаки та робить такі генератори привабливими для використання в системах з підвищеними вимогами до безпеки.

Ще одна з переваг є висока керованість умов експлуатації. Темновий струм визначається внутрішніми параметрами фотодіода, напругою зсуву та температурою. За умови стабілізації цих параметрів можливо досягти відтворюваних характеристик шуму, що є важливою перевагою для сертифікації та довготривалої експлуатації апаратних TRNG.

Фотодіоди легко інтегруються в інтегральні схеми та мікроелектронні модулі. Це дозволяє створювати компактні апаратні генератори ентропії,

придатні для вбудованих систем, смарт-карт, криптографічних токенів і пристроїв Інтернету речей.

Генератор на основі темного струму фотодіода має підвищену стійкість до активних атак. Відсутність зовнішнього входу для фізичного сигналу знижує ймовірність маніпуляцій з боку злоумисника. У порівнянні з іншими фізичними TRNG, генератори на основі темного струму менш уразливі до інжекції керованих сигналів або нав'язування детермінованих шаблонів.

2.6.3.2. Недоліки генератора ПВЧ на основі темного струму фотодіода

1. Надзвичайно малий рівень корисного сигналу. Темновий струм фотодіода має дуже малу амплітуду, що ускладнює його вимірювання. Це вимагає застосування високочутливих малошумних підсилювачів, що суттєво ускладнює схемо-технічну реалізацію та підвищує вимоги до якості компонентів.

2. Сильна температурна залежність. Одним із найсерйозніших недоліків є експоненційна залежність темного струму від температури. Навіть незначні температурні коливання можуть суттєво змінювати інтенсивність шуму, що призводить до нестабільності рівня ентропії та потребує термокомпенсації або активної температурної стабілізації.

3. Обмежена швидкодія. Через малу потужність шумової складової швидкість генерації випадкових бітів зазвичай є нижчою, ніж у генераторів, заснованих на дробовому шумі фотоструму або інших квантових процесах. Це обмежує використання таких TRNG у високопродуктивних або високонавантажених системах.

4. Високі вимоги до пост-обробки. Сигнал темного струму часто містить низькочастотні компоненти, зокрема $1/f$ -шум, які можуть мати корельований характер. Для забезпечення прийнятної якості випадковості необхідні складні процедури фільтрації, усунення зміщення та

криптографічного хешування, що збільшує обчислювальні витрати та складність реалізації.

5. Ризик псевдоентропії при некоректному проектуванні. За недостатнього екранування, неякісного живлення або помилкового вибору робочого режиму фотодіода детерміновані шумові компоненти можуть домінувати над випадковими. У таких випадках вихідна послідовність може формально виглядати випадковою, але мати знижену ефективну ентропію, що є критичним для криптографічних застосувань.

2.6.3.3. Сфера застосування генератора ПВЧ на основі темнового струму фотодіода

Апаратні модулі безпеки та довірені платформи. Однією з основних сфер

застосування генераторів на основі темнового струму є апаратні модулі безпеки, зокрема HSM, TPM, криптографічні токени та смарт-карти. У таких системах першочергове значення має не швидкість генерації, а гарантована фізична випадковість і стійкість до активного зовнішнього впливу. Темновий струм, як ізольоване джерело ентропії, добре відповідає цим вимогам.

Вбудовані та ресурсно обмежені системи. У вбудованих системах і пристроях Інтернету речей, де програмні джерела ентропії часто є недостатніми або нестабільними, генератори на основі темнового струму можуть використовуватися для ініціалізації криптографічних ключів; генерації унікальних ідентифікаторів пристроїв; підтримки механізмів безпечного завантаження та автентифікації. У таких застосуваннях невисока швидкодія компенсується низькими вимогами до пропускної здатності та високою надійністю джерела випадковості.

Криптографічні системи з підвищеними вимогами до захисту. У криптографічних системах, орієнтованих на довготривалу безпеку, темновий струм фотодіода може використовуватися як первинне або додаткове джерело

ентропії для генерації ключового матеріалу. Особливо це актуально у сценаріях, де існує загроза фізичного доступу до пристрою або можливість навмисного впливу на середовище його роботи.

Наукові та експериментальні дослідження. Темновий струм фотодіода широко використовується в наукових дослідженнях для вивчення шумових процесів у напівпровідниках, а також для експериментальної перевірки методів оцінювання ентропії. У цьому контексті такі генератори слугують інструментом для дослідження меж апаратної випадковості та порівняння різних фізичних джерел ентропії.

Системи з високими вимогами до ізоляції. У спеціалізованих системах, де необхідно мінімізувати будь-які зовнішні канали впливу на апаратну частину, генератори на основі темнового струму є доцільним вибором. Відсутність світлового входу та мінімальна взаємодія з навколишнім середовищем роблять їх придатними для використання в захищених або закритих апаратних середовищах.

2.6.4. Висновки щодо порівняльного аналізу розроблених генераторів ПВЧ

Система генерації ПВЧ є критичним компонентом багатьох інформаційних, криптографічних і керуючих систем. Її якість безпосередньо впливає на рівень безпеки, достовірність моделювання та надійність функціонування прикладних алгоритмів. У зв'язку з цим до таких систем висувається комплекс взаємопов'язаних вимог, які охоплюють фізичні, статистичні, криптографічні та експлуатаційні аспекти. Розглянемо ці вимоги більш детально для оцінки ступені відповідності створених генераторів ПВЧ цим вимогам [11, 87-89].

1. Вимоги до джерела ентропії:

Основною вимогою є наявність надійного джерела ентропії, яке має фізичну або статистично обґрунтовану природу. Джерело повинно:

- забезпечувати істинну або ефективну випадковість;
- бути мінімально залежним від детермінованих факторів;
- демонструвати стійкі стохастичні властивості в часі.
- для апаратних генераторів додатково важливо, щоб джерело ентропії було ізольоване від неконтрольованого зовнішнього впливу або мало механізми виявлення такого впливу.

2. Статистичні вимоги. Згенеровані послідовності повинні відповідати критеріям статистичної випадковості, а саме:

- рівномірному розподілу бітів або чисел;
- відсутності коротко- та довгострокових кореляцій;
- відсутності періодичності або структурних шаблонів.

Система має забезпечувати можливість перевірки якості вихідних даних за допомогою стандартних статистичних тестів випадковості та контролювати деградацію характеристик у процесі експлуатації.

3. Криптографічні вимоги:

- непередбачуваність наступних значень навіть за часткового знання попередніх;
- стійкість до відновлення внутрішнього стану генератора;
- неможливість відтворення послідовності без доступу до джерела ентропії або секретних параметрів.
- у криптографічних системах генератор випадкових чисел не повинен бути єдиною точкою компрометації всієї системи безпеки.

4. Вимоги до пост-обробки та усунення зміщення:

- усунення статистичних зміщень;
- зменшення кореляції між елементами послідовності;
- підвищення ефективної ентропії на біт.
- методи постобробки мають бути формально обґрунтованими та не знижувати криптографічну стійкість системи.

5. Вимоги до продуктивності:

- швидкість генерації згідно сучасних вимог не менше 100 Мбіт/сек;
- стабільність продуктивності за тривалої роботи.
- для високонавантажених систем важливою є масштабованість і можливість паралельної генерації.

6. Надійність і стабільність роботи:

- стійкість до температурних і електромагнітних впливів;
- відсутність деградації ентропії з часом;
- коректна робота за граничних режимів;
- система має підтримувати механізми самодіагностики та виявлення аномалій.

7. Вимоги безпеки та захисту від атак:

- спроб прогнозування або нав'язування вихідної послідовності;
- фізичного впливу на джерело ентропії;
- побічних каналів (електромагнітних, теплових, часових).
- за необхідності система повинна мати механізми реагування, зокрема припинення генерації або сигналізацію про порушення.

8. Експлуатаційні та інтеграційні вимоги:

- легко інтегруватися в цільову апаратну або програмну платформу;
- мати стандартизований інтерфейс доступу;
- відповідати вимогам енергоспоживання та ресурсних обмежень;
- простота налаштування та обслуговування;

Таким чином, система генерації ПВЧ повинна використовувати генератори фотодіодного, вебкамерного та темнового струму типу плюс до того мати створений на основі цих генераторів гібридний генератор, який здійснює обробку отриманих ПВЧ. Три фізичні генератори повністю вирішують питання про непередбачуваність, хаотичність, криптостійкість та ізольованість. Загалом такий набір генераторів ПВЧ можна назвати генераторами на основі ФЕЯ,

оскільки в його основі лежать такі фотоелектричні явища як генерація інтенсивності пікселів веб-камери, фото- та темнового струму фотодіода.

Гібридний генератор на такій основі вирішує питання по швидкодії, надійності, контрольованості, відповідності статистичним вимогам та вимогам до безпеки, надійності і стабільності пост-обробки. Важливим додатком є те, що програмна частина генератора відкриває теоретично необмежені можливості по масштабованості та паралелізації процесу генерації.

Висновки до розділу 2

Проведено дослідження веб-камери фото- та темнового струмів фотодіода як джерел для генерації ПВЧ. Показано, що генератор ПВЧ на основі звичайної веб-камери має непередбачуваний характер ПВЧ і продуктивність до 288 Мбіт/сек, в залежності роздільної здатності веб-камери (SVGA).

Генератор на основі фотоструму фотодіода здатен генерувати непередбачувану ПВЧ, його продуктивність складає до 2 Мбіт/сек і ПВЧ, створени ним, частково відповідають тестам NIST. ПВЧ, створена за допомогою генератора на основі темнового струму фотодіода, має абсолютно непередбачуваний характер і продуктивність до 980 Кбіт/сек, яка може бути збільшена при застосування височастотної апаратної частини. Такий генератор може використовуватись для затравки гібридних генераторів ПВЧ.

Запропоновано ідею гібридного генератора, який може поєднувати у собі генератор на основі веб-камери і КА, який зможе вирішувати питання по швидкодії, надійності, контрольованості, відповідності статистичним вимогам та вимогам до безпеки і стабільності пост-обробки.

Таким чином показано, що можливо створення інформаційної системи для генерації ПВЧ на основі усіх трьох генераторів із обробкою отриманих ПВЧ за допомогою КА. Таке об'єднання повністю вирішуватиме питання про непередбачуваність, хаотичність, криптостійкість та ізолюваність.

Основні наукові результати розділу опубліковані в працях [66, 69, 75, 89].

РОЗДІЛ 3

РОЗРОБКА ТА ДОСЛІДЖЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ГЕНЕРАЦІЇ ВИСОКОЕНТРОПІЙНОЇ ПВЧ НА ОСНОВІ ФЕЯ

3.1. Програмний комплекс для генерації та аналізу хаотичних числових послідовностей на основі ФЕЯ

Вище отримані результати щодо дослідження технологій генерації ПВЧ з інтенсивності пікселів фоточутливої матриці веб-камери (або відеокамери), з фото- та темнового струму фотодіода дозволили запропонувати новий тип генераторів ПВЧ, заснований на фотоелектричних явищах (ФЕЯ) із застосуванням КА для пост-обробки. Спираючись на отримані результати було створено програмний комплекс для керування процесом генерації та аналізу ПВЧ.

Розроблений програмний комплекс представляє собою спеціалізований інструмент для генерації та аналізу ПВЧ на основі вище перелічених явищ та технологій, розроблений з використанням сучасних технологій програмування. Проект реалізовано на мові Java [71] з використанням фреймворку JavaFX [86] для створення інтерактивного графічного інтерфейсу користувача.

Концептуальна основа проекту полягає у створенні гнучкого інструменту, який дозволяє генерувати ПВЧ різними методами в різних умовах та проводити детальний аналіз згенерованих ПВЧ на відповідність критеріям випадковості.

При розробці архітектури програмного комплексу було прийнято рішення використовувати патерн Model-View-Controller (MVC), що забезпечує чітке розділення логіки програми від її представлення. Це дозволило створити модульну структуру, яка легко розширюється та підтримується. Вибір JavaFX як технології для розробки інтерфейсу користувача обумовлений його кросплатформенністю, багатим набором компонентів та можливістю створення сучасного, естетично привабливого інтерфейсу.

Програмний комплекс надає користувачам наступні можливості:

- налаштування параметрів генерації: користувач може обирати між різними діапазонами значень (біти 0-1, стандартний діапазон 1-256, або користувацький діапазон), а також встановлювати параметри, для регулювання балансу між швидкістю генерації та рівнем безпеки.

- вибір джерела даних: підтримується як генерація на основі файлів з попередніми вимірюваннями, так і отримання даних у режимі реального часу з мікроконтролерів (Arduino, ESP32, Raspberry Pi Pico тощо).

- генерація послідовностей: користувач може генерувати послідовності заданої довжини з заданими параметрами, зберігати їх та аналізувати.

- детальний аналіз: програма проводить комплексний аналіз згенерованих послідовностей, включаючи обчислення ентропії, аналіз розподілу та виконання стандартних тестів на випадковість (частотний тест, тест на серії (series test), тест на приблизну ентропію, тест на найдовшу серію одиниць, тест рангу бінарної матриці, тест дискретного перетворення Фур'є, тест кумулятивних сум).

- візуалізація результатів: результати аналізу представляються у вигляді гістограм та інших графічних елементів для полегшення інтерпретації.

- управління історією: програма зберігає історію згенерованих послідовностей та їх налаштувань, дозволяючи порівнювати різні спроби та аналізувати їх відмінності.

Вибір методів аналізу випадковості базується на загальноприйнятих наукових підходах, зокрема на рекомендаціях NIST (Національного інституту стандартів і технологій США) щодо тестування генераторів випадкових чисел. Реалізовані тести дозволяють всебічно оцінити якість згенерованих послідовностей та їх відповідність критеріям випадковості.

Особливу увагу приділено обчисленню ентропії як ключової характеристики

випадковості послідовності. Високе значення ентропії свідчить про більшу непередбачуваність та, відповідно, вищу якість генератора випадкових чисел.

Зовнішній вигляд інтерфейсу програмного комплексу для генерації ПВЧ наведено на рис. 3.1.

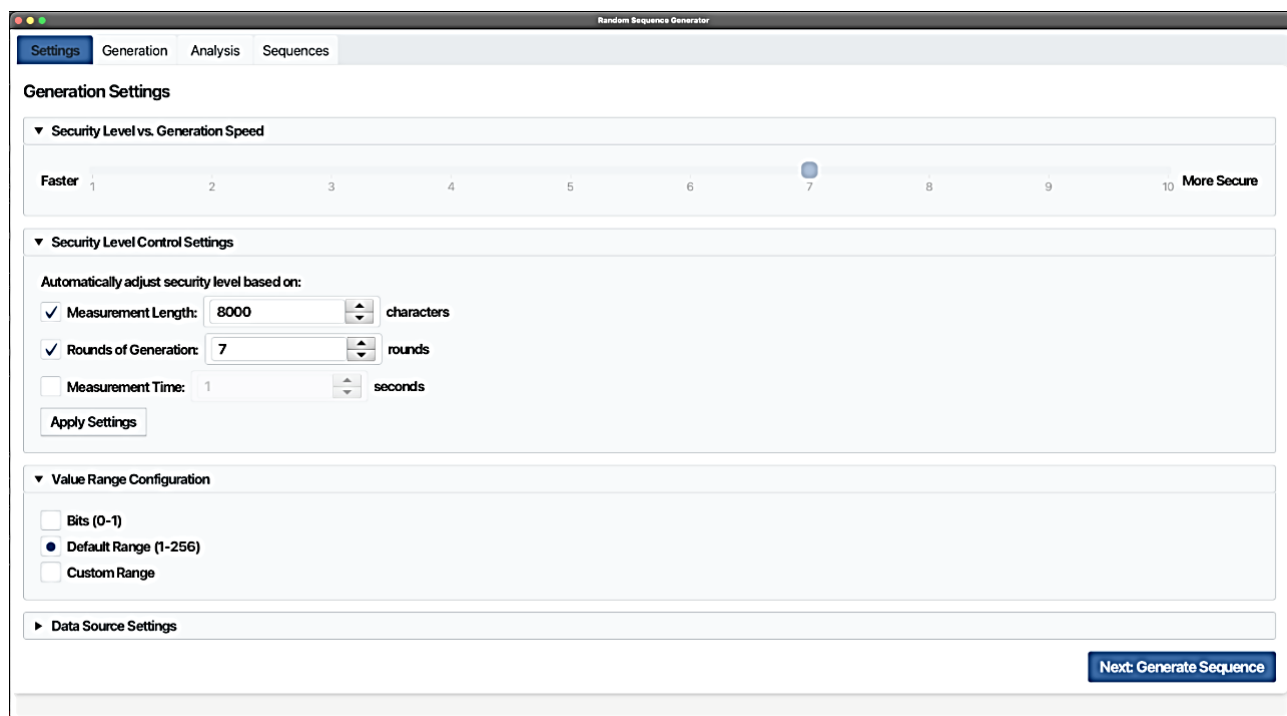


Рисунок 3.1 – Зовнішній вигляд інтерфейсу програмного комплексу.

Інтерфейс користувача розроблено з урахуванням сучасних принципів UX/UI дизайну. Програма має табовий інтерфейс з чотирма основними розділами.

Вкладка "Settings" (Налаштування): містить елементи керування для налаштування параметрів генерації, вибору джерела даних та встановлення рівня безпеки. Інтерфейс організовано у вигляді логічних груп елементів керування, розміщених у панелях з заголовками для полегшення навігації (рис. 3.1).

Вкладка "Generation" (Генерація) (рис. 3.2): надає інструменти для генерації послідовностей, відображення згенерованих даних та проведення базового аналізу. Містить текстове поле для відображення згенерованої послідовності та елементи керування для її збереження та копіювання.

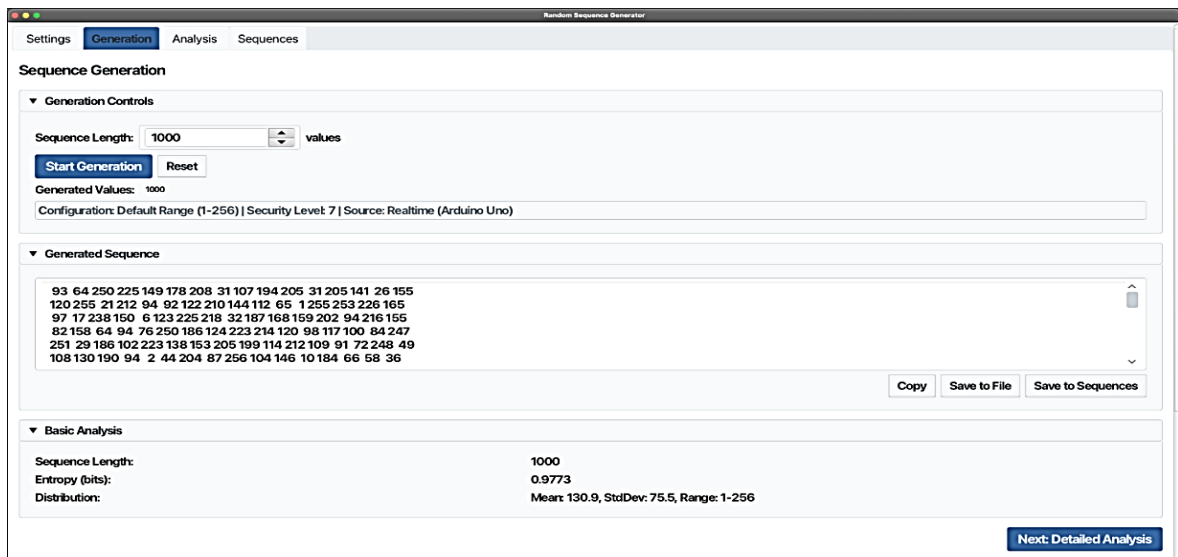


Рисунок 3.2 – Вигляд вкладки "Generation" для генерації послідовностей.

Вкладка "Analysis" (Аналіз) (рис. 3.3) представляє результати детального аналізу згенерованих послідовностей, включаючи різноманітні тести на випадковість та візуалізацію розподілу даних у вигляді гістограм.

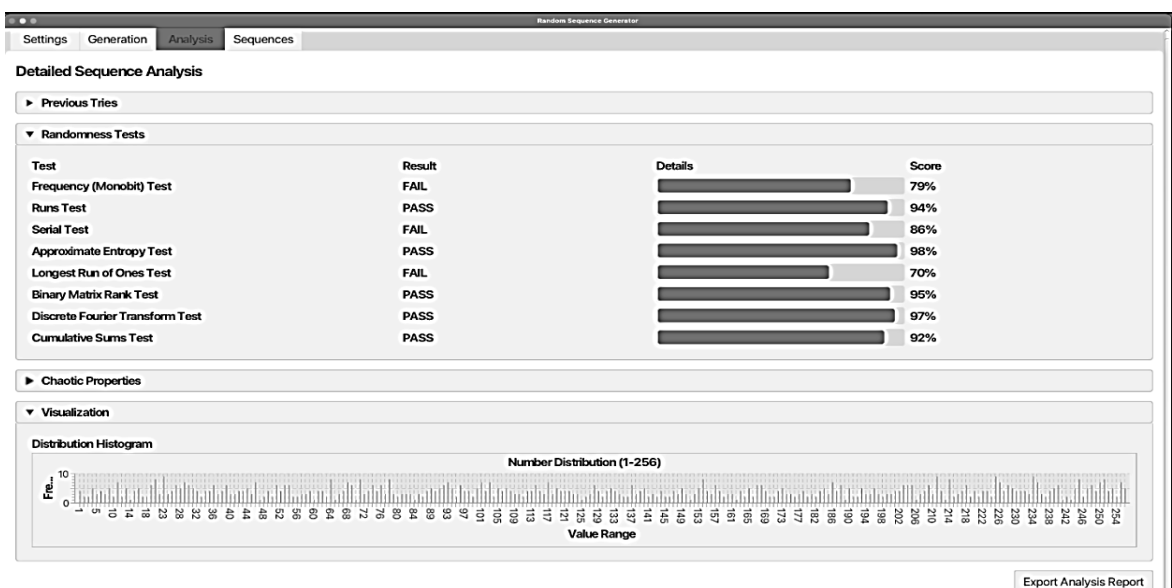


Рисунок 3.3 – Вигляд вкладки "Analysis", яка представляє результати детального аналізу згенерованих послідовностей.

Вкладка "Sequences" (Послідовності): дозволяє переглядати та керувати раніше згенерованими послідовностями, завантажувати їх параметри для подальшого аналізу або порівняння (рис. 3.4).

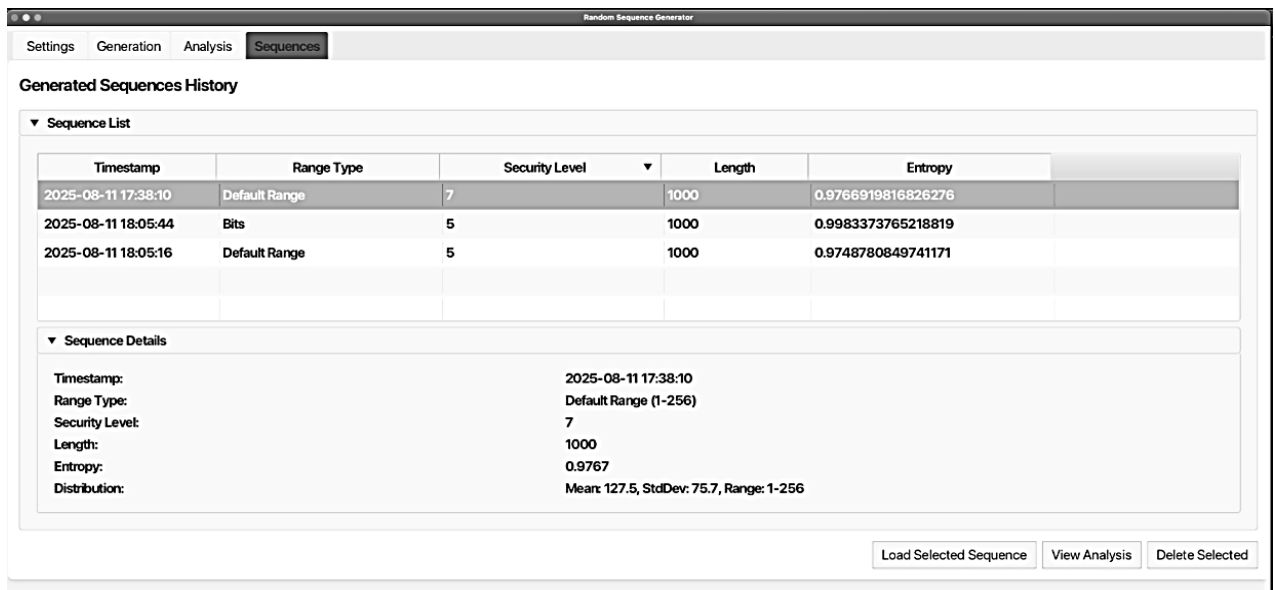


Рисунок 3.4 – Вигляд вкладки "Sequences", яка дозволяє переглядати та керувати раніше згенерованими послідовностями.

Сучасність та актуальність Java.

Постійний розвиток мови: Java регулярно оновлюється (поточна версія 21) [137], впроваджуючи сучасні функції програмування, такі як лямбда-вирази, модульність, текстові блоки та записи (records), що спрощують розробку та підтримку коду.

Віртуальна машина Java (JVM): Сучасна JVM забезпечує високу продуктивність завдяки оптимізаціям, таким як JIT-компіляція та адаптивна оптимізація, що особливо важливо для обчислювально-інтенсивних операцій генерації та аналізу випадкових послідовностей.

Project Loom: Нова функція Java, що впроваджує віртуальні потоки (virtual threads), дозволяє ефективно обробляти тисячі одночасних операцій, що ідеально підходить для паралельної генерації та аналізу послідовностей.

Переваги Java для генератора випадкових послідовностей.

JavaFX для сучасного інтерфейсу: JavaFX пропонує багатий набір компонентів для створення інтерактивного графічного інтерфейсу з

підтримкою CSS, анімацій та ефектів, що дозволяє створити професійний та зручний інтерфейс для користувачів.

Кросплатформенність: Java-додаток працюватиме однаково на Windows, macOS та Linux без необхідності перекомпіляції, що важливо для широкого розповсюдження інструменту.

Потужні можливості для аналізу даних: Java має бібліотеки для статистичного аналізу (Apache Commons Math), обробки великих масивів даних та візуалізації результатів, що необхідно для аналізу якості згенерованих послідовностей.

Інтеграція з апаратними засобами: Java підтримує роботу з послідовними портами через бібліотеки як RXTX або jSerialComm, що дозволяє легко інтегрувати мікроконтролери (Arduino, ESP32) для отримання даних у реальному часі.

В результаті проведеного дослідження створено програмний комплекс для керування процесом генерації та аналізу ПВЧ на основі ФЕЯ фотодіода.

Інтерфейс комплексу дозволяє керувати параметрами генерації ПВЧ, а саме - обирати між різними діапазонами значень (біти 0-1, стандартний діапазон 1-256, або користувацький діапазон) і регулювати баланс між швидкістю генерації та рівнем безпеки, який забезпечує згенерована ПВЧ, зберігати її та аналізувати.

Комплекс підтримує генерацію ПВЧ на основі файлів з попередніми вимірюваннями і дозволяє отримувати дані у режимі реального часу з мікроконтролерів типу Arduino, ESP32, Raspberry Pi Pico, або інших, що передають дані до комп'ютера у форматі txt, проводить комплексний аналіз згенерованих ПВЧ, включаючи обчислення ентропії, аналіз розподілу та виконання стандартних тестів на випадковість. Результати аналізу представляються у вигляді гістограм та інших графічних елементів для полегшення інтерпретації.

3.2. Інформаційна система для генерації ПВЧ на основі ФЕЯ

Вище описані результати було покладено в основу інформаційної системи для генерації ПВЧ на основі ФЕЯ [87, 88, 89]. Для досягнення мети роботи потрібно було розв'язати наступні завдання.

1. Створення апаратної частини інформаційної системи. А саме.
 - 1.1. Забезпечення екстракції інтенсивності пікселів веб-камери із забезпеченням певної кількості вимірювань в одиницю часу.
 - 1.2. Забезпечення певної кількості вимірювань в одиницю часу темного та фотоструму фотодіоду.
 - 1.3. Забезпечення роботи гібридного генератора, який обробляє отримані значення фізичних величин за допомогою КА.
 - 1.4. Забезпечення передачі виміряних значень фізичних величин до комп'ютера.
2. Створення ПВЧ.
3. Забезпечення перевірки ПВЧ на відповідність критеріям хаотичності.
4. Розробка інформаційної системи для генерації ПВЧ на основі ФЕЯ.

3.2.1. Створення апаратної частини інформаційної системи

Для вирішення поставлених завдань було розроблено установку для генерації ПВЧ на основі виміряних значень темного та фотоструму кремнієвого фотодіода ФД-288 [90], а також веб-камери. Блок-схема установки наведена на рис. 3.5.

Установка працює наступним чином. Фотодіод ФД-288 (3) закритий світлонепроникним екраном (2) і разом з ним розміщений у термостаті типу НВ120-S [91] (1), який має діапазон підтримування температур від 5 до 120 °С з точністю 0,5 °С і живиться як від мережі, так і від блоку живлення (5). На термостаті встановлюється температура 50 оС. За допомогою блоку живлення фотодіоду (4) на ньому встановлюється зміщення 10 мВ. Виміряне значення темного струму фотодіоду перетворюється у напругу за допомогою

прецизійного перетворювача струм-напруга (ППСН) (7), підсилюється підсилювачем (8) для подальшої обробки аналого-цифровим перетворювачем (9) з метою отримання цифрового сигналу, еквівалентного темновому струму, який оброблюється процесором Arduino (112593) (10) для адаптування цифрового сигналу і передачі його до USB-адаптера (11), за допомогою якого цифровий сигнал передається до USB порту комп'ютера (12), з якого потрапляє до процесора комп'ютера (13). Керування встановленням напруги на фотодіоді, температури фотодіода, тривалістю відбору значень темнових струмів, кількості вимірів в одиницю часу, здійснюється за допомогою інтерфейсу інформаційної системи.

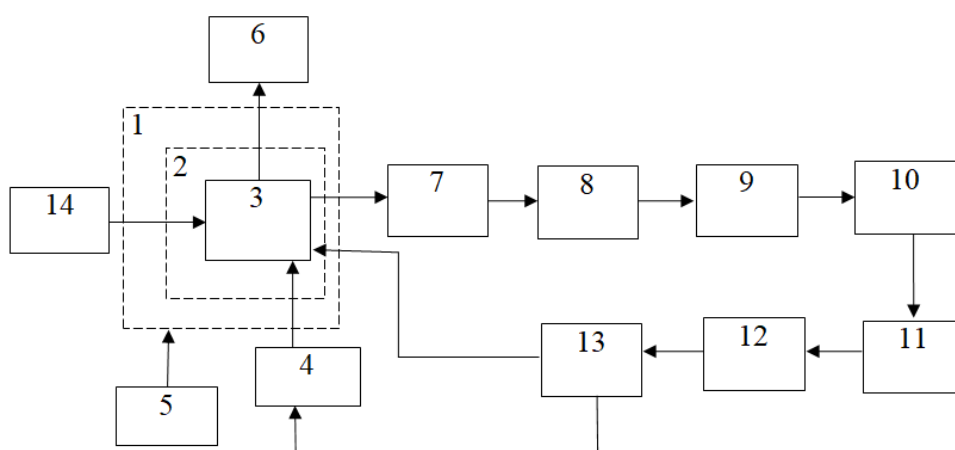


Рисунок 3.5 – Блок-схема установки для генерації ПВЧ на основі вимірювань значень фотоелектричних параметрів фотодіода: 1 – термостат; 2 – світлонепроникний екран, 3 – фотодіод ФД-288, 4 - блок живлення фотодіода, 5 - блок живлення термостата, 6 - термометр цифровий Тензор-42, 7 - ППСН, 8 - підсилювач постійної напруги; 9 - аналого-цифровий перетворювач; 10 - центральний процесор Arduino (112593), 11 - USB-адаптер, 12 - USB порт комп'ютера, 13 - процесор комп'ютера, 14 – освітлювач.

Підсилення сигналу темного струму на 5 Вольт, переводить характеристику розподілу по значенню псевдовипадкових чисел у бік «рівномірно-хаотичної». Для вимірювання фотоструму фотодіоду до блок-

схеми додається освітлювач (14). Для екстракції інтенсивності пікселів з веб-камери до блок-схеми на рис. 3.5. додається блок, наведений на рис. 2.1. Опис його роботи також наведено у розділі два.

3.2.2. Створення ПВЧ

Формування ПВЧ здійснюється згідно алгоритму, наведеному на рис. 3.6.

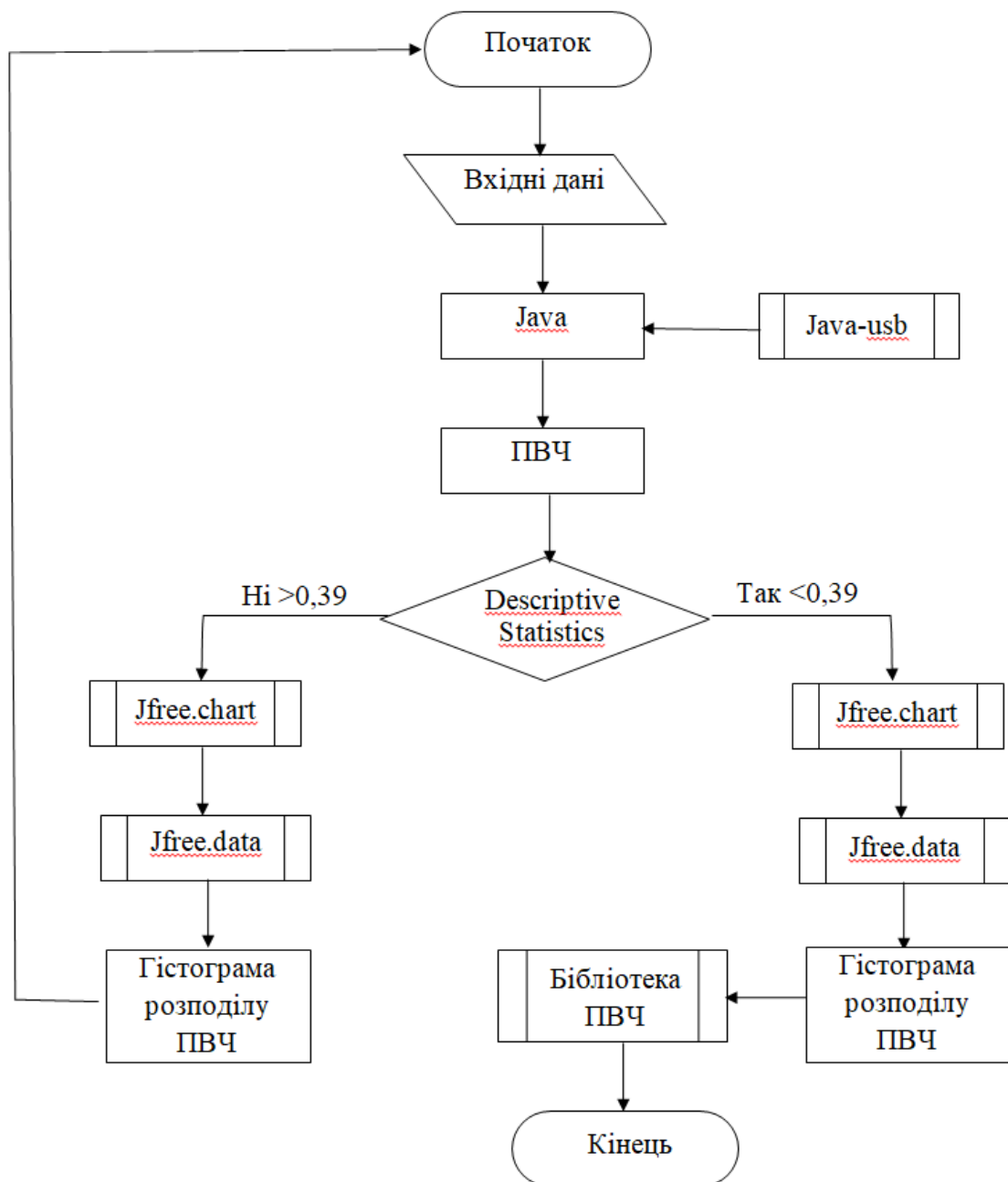


Рисунок 3.6 – Алгоритм формування ПВЧ

Виміряні значення напруги, яка еквівалентна струму (темновому або фото-) фотодіода, у вигляді вхідного потоку даних опрацьовуються комп'ютером із використанням мови програмування Java (17 версія). При цьому застосовується бібліотека `java-usb`, за допомогою якої вхідні дані завантажуються у певний список. При цьому вони вважаються випадковими числами від 0 до 10. Далі отриманий список чисел вивчається на відповідність статистичним вимогам із використанням бібліотеки `Descriptive Statistics`. Бібліотеки `jfree.chart` та `jfree.data` застосовуються для графічного представлення статистичних характеристик ПВЧ.

3.2.3. Забезпечення перевірки отриманих ПВЧ на відповідність критеріям хаотичності

Як відомо [77] ідеальний рівномірний розподіл повинен мати середнє значення (Z_{mean}) на рівні 0,39 і відхилення від середнього - 0. Основним критерієм ідеально рівномірного розподілу в нашому випадку обрано саме цю величину – середнє квадратичне відхилення. Програмно згенерована послідовність має відхилення від середнього не більше 0,002 [92]. Як що отримана ПВЧ відповідає цьому критерію, вона направляється до бібліотеки ПВЧ. Як що не відповідає – відкидається, або формується повторно.

Окрім того, має відбуватися перевірка на відповідність іншим критеріям, таким як мінімальне (Z_{min}) і максимальне значення (Z_{max}) співпадінь однакових чисел у послідовності. Ці значення мають бути також в околі 0,39. Також варто враховувати процент відсутніх елементів (Z_{absent}), який має складати 0 %.

Зрозуміло, що оскільки вимоги до Z_{mean} , Z_{min} , та Z_{max} є однакові – не повинні перевищувати 0,39, то можна обмежитись вимірюванням значення Z_{mean} .

Але, для забезпечення точності вимірів варто ці показники контролювати. Тому у рамках інформаційної системи, коли на формування ПВЧ не впливає час, ці параметри мають перевірятись.

3.2.4. Формування архітектури інформаційної системи для генерації ПВЧ на основі ФЕЯ

Основне завдання створюваної інформаційної системи полягає у генерації ПВЧ з одного боку, захищеності системи від стороннього втручання з другого та зручність користування користувачем з третього.

Сформуємо основні вимоги до такої інформаційної системи.

1. Має бути реалізований принципи CIA (confidentiality, integrity, availability, або конфіденційність, цілісність, доступність) [93, 94].

2. Має бути створена ієрархія інформаційної безпеки, яка має містити управління згенерованими ПВЧ, інформаційний та мережевий менеджмент, системний, сервісний і бізнес-рівень, а також постійний моніторинг вимог процесу. Також інформаційна система має відслідковувати такі чинники як політичні та законодавчі зміни. Ці фактори повинні мати безперервний аудит.

3. Має бути зручний інтерфейс генерації, дослідження та використання ПВЧ.

3.2.4.1. Реалізація принципів CIA

Кожне з перерахованих вище завдань реалізується у вигляді мікросервіса в загальному контексті архітектури "N-tier", тобто багаторівневої архітектури.

В основі роботи розробленої інформаційної системи лежить принцип ієрархічності допусків власника, менеджерів та користувачів системи, як це показано на рис. 3.7.

CIA повинна мати документальну базу, а також затверджену регуляторну політику. В них документально мають бути визначені вимоги до параметрів і показників, за якими потрібно безперервно стежити (монітори). Це дозволить своєчасно реагувати системі на загрози. Регуляторна політика має формуватися на основі принципу Gap-Analysis. А саме - оцінці зазорів (gap) між ймовірним зломом і арсеналом протидії зламу.

Основні вимоги гарантії безпеки CIA.

C - confidentiality – забезпечення недоступності до закритої (конфіден-

ційної) інформації для не уповноваженої особи або процесу (бота, сканування, тощо).

I - integrity - властивість підтримки правильності і повноти активів. Якщо з'являється нова загроза (ризик), то наш арсенал повинен мати можливість, бути доповненим. Оцінка ризику також завдання не просте.

A - availability - властивість бути доступним і готовим до використання за запитом уповноваженої особи.

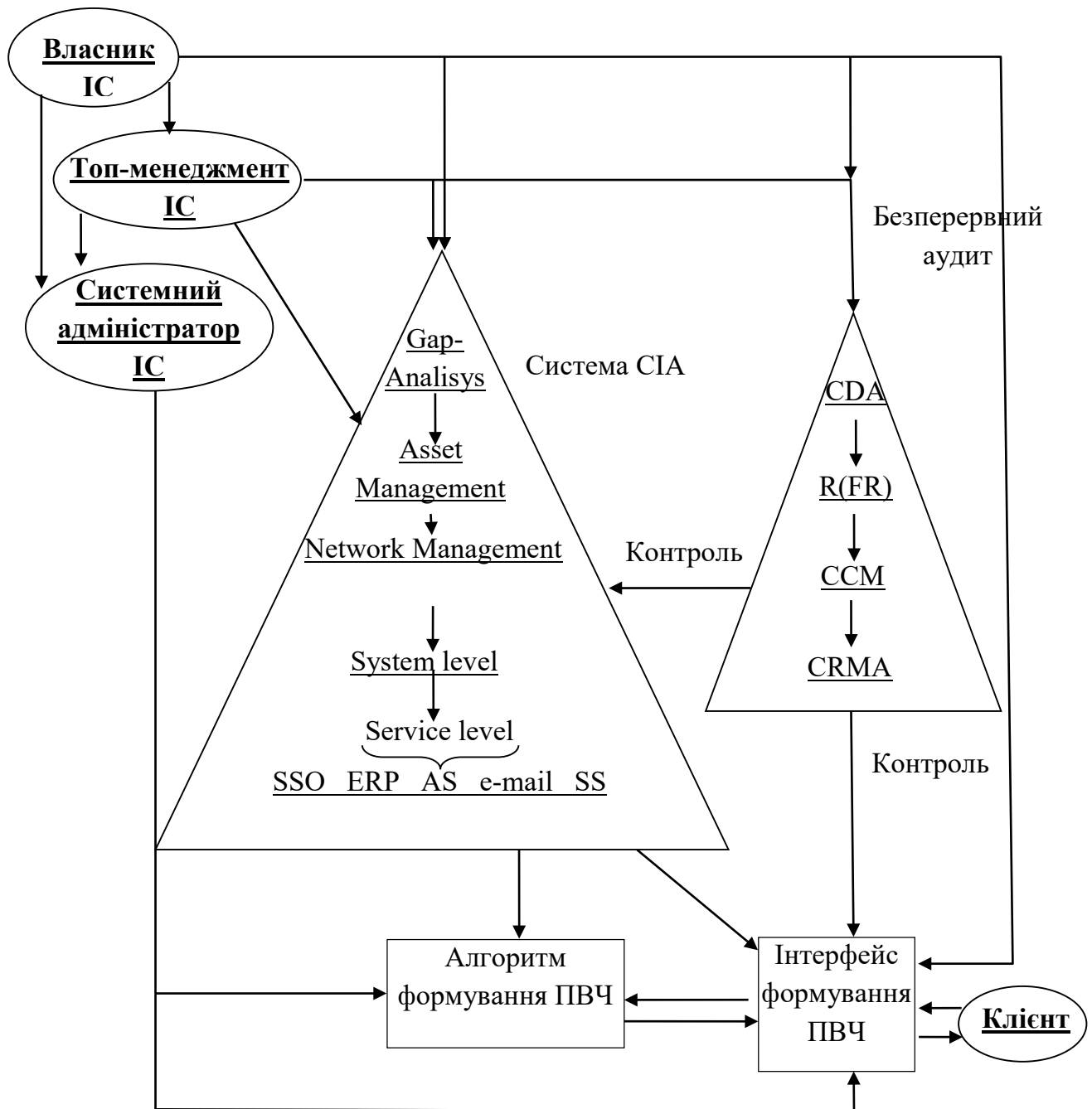


Рисунок 3.7 – Архітектура інформаційної системи

Ієрархія інформаційної безпеки передбачає управління активами (Asset management). Будь-який матеріальний носій або передавач інформації є Asset. Жорсткі, диски, роутери, комп'ютери, мобільні телефони, флешпам'яті, CD-ROM. Всі ці Asset management можуть бути використані для несанкціонованого доступу або витоку інформації.

Asset - набір всіх даних, правил і процедур, які в сукупності представляють собою концепцію, значиму для бізнесу. Будь-який збій в цьому алгоритмі повинен знайти відгук в системі безпеки.

Network Management (Мережевий менеджмент) передбачає моніторинг ризиків пов'язаних з несанкціонованого доступу до інформації при передачі через мережі.

System level (Системний рівень) передбачає моніторинг системного рівня і відстеження подій/змін в роботі: domain контролерів; бази даних; сховища даних; серверних додатків; систем віртуалізації; операційних систем; хмарних кластерів.

Service level (Сервісний рівень) розділяється на SSO - Single Sign-On (Система єдиного входу), ERP (enterprise resource planning - планування ресурсів підприємства), AS (authorization service - сервіс авторизації), e-mail сервіс, SS (supporting services - допоміжні сервіси).

Складовою частиною безперервного аудиту (моніторингу) є:

CDA - безперервне забезпечення цілісності даних (Continuous data assurance). CDA перевіряє цілісність даних, що проходять через інформаційні системи.

Основними методами забезпечення цілісності інформації (даних) при зберіганні в автоматизованих системах є:

- R(FR) (Reliability (failure resistance) - надійність (відмово стійкість) - надмірність, дублювання, віддзеркалення обладнання та даних, наприклад, за рахунок використання RAID-масивів;

- безпечне відновлення (резервне копіювання і електронне архівування інформації).

CCM (Continuous control monitoring) - безперервний моніторинг управління ресурсами. CCM забезпечує постійний моніторинг використання ресурсів, порівнюючи їх із заздалегідь визначеними ключовими показниками для виявлення нештатних ситуацій. Найбільш поширеним рішенням CCM є створення управління Центру безпекою (SOC). CCM також включає регулярне сканування компонентів ІТ-інфраструктури на наявність вразливостей. Часто ці обов'язки покладаються на системного адміністратора і технічної підтримки.

CRMA (Continuous risk monitoring and assessment) - постійний ризик-моніторинг. В рамках CRMA аналізуються результати CCM і CDA, а також аналізуються і прогнозуються ризики і можливі втрати. В рамках CRMA відбувається управління ризиками - процес прийняття і реалізації управлінських рішень, спрямованих на зниження ймовірності інцидентів і мінімізацію втрат. Найбільш часто використовуваним інструментом управління ризиками є страхування.

3.2.5. Інформаційна система для генерації ПВЧ на основі ФЕЯ

Розробка алгоритму, програмного комплексу та архітектури генерації ПВЧ дозволило створити інформаційну систему, яка дозволяє генерувати ПВЧ різних видів, довжин і рівнів якості. Структурна схема інформаційної системи наведена на рис. 3.8.

Оператор спочатку за допомогою первинного інтерфейсу <<SETTINGS>> (рис. 3.1) налаштовує параметри генерації.

Перш за все виставляється рівень балансу між якістю і швидкістю генерації – поле «Security Level vs. Generation Speed». Рівень якості визначається on-line, і при зниженні якості автоматично знижується швидкість генерації за рахунок того, що неякісні послідовності обробляються додатковими ітераціями клітинних автоматів.

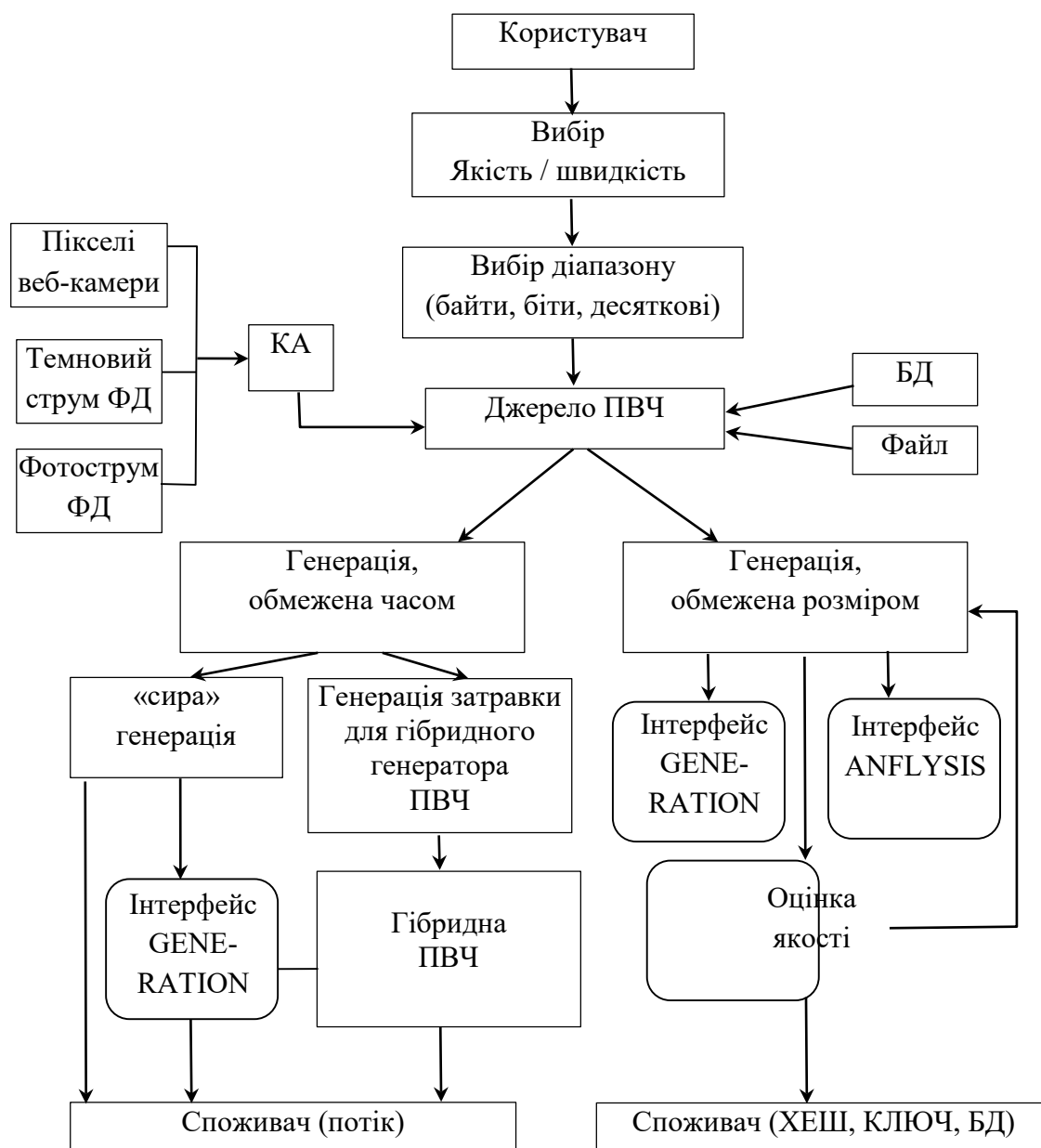


Рисунок 3.8 – Структурна схема інформаційної системи для генерації ПВЧ на основі ФЕЯ.

Далі вибирається вид згенерованих послідовностей - «Security Level Control Settings». Вибирається довжина випадкових послідовностей, кількість послідовностей або послідовність, що згенерована за виставлений час.

Після цього оператор вибирає тип чисел, що будуть згенеровані - біти (0,1), десяткові (0 ... 9), шістнадцяткові (0 .. F), байти (0 .. 255).

Останнє налаштування – це вибір джерела. Із доступних джерел це:

1. Фізичне джерело на основі ентропії темного струму фотодіода.
2. Файл з завчасно накопиченими випадковими послідовностями.
3. База даних з завчасно накопиченими послідовностями.
4. Гібридний генератор ПВЧ, який для затравки використовує накопичені випадкові послідовності.
5. Програмний генератор ПВЧ на основі Java (наприклад, SecureRandom).

Вибір джерела визначається поставленою задачею. Так, наприклад, джерело ентропії на основі фотодіоду здатне генерувати потік випадкових послідовностей практично без обмежень по часу. Але без програмної обробки числа хоча і будуть абсолютно непередбачувані, проте якість саме випадкової послідовності буде невисока. Файл з завчасно накопиченими даними здатен генерувати послідовності високої якості, проте обмежений інтервал часу.

Після всіх налаштувань оператор переходить на інтерфейс <<GENERATION>> (рис.3.2) і запускає процес генерації ПВЧ. В процесі генерації оператор може бачити візуально саму згенеровану послідовність, а також оперативний аналіз якості. Зокрема рівень ентропії та статистичні дані (середнє значення, середнє квадратичне відхилення, діапазон та інші). При цьому є можливість відправляти споживачу як «сиру» необроблену послідовність, так і високоякісну, але програмно оброблену, де дані з фотодіоду виступають як затравка для гібридного генератора.

У випадку генерації затравки для програмної частини засобами програмної інженерії, можливо масштабувати всю програмну частину генератора або ж розпаралелити процес обробки «сирої» ПВЧ всередині блоку програмної частини.

Така потреба в основному виникає у випадку, коли споживач обмежений в часі, і йому важливіша швидкість і тривалість генерації більше ніж якість ПВЧ, наприклад для захисту каналу передачі даних (стеганографія).

У випадку, коли споживачу важливіша якість ПВЧ, наприклад, проходження тестів NIST, то у такому випадку джерелом може виступити файл або база даних. Таке джерело було заповнено завчасно, сирі дані були одержані методом, що описаний вище, але у подальшому були оброблені клітинними автоматами до певної відповідності тестам NIST. Слід зауважити, що 100% відповідності по кожному тесту досягти не вдається, проте кожна величина тесту лежить в допустимому діапазоні. У цьому випадку оператор і споживач за допомогою інтерфейсу «ANALYSIS» (рис 3.3) можуть вже детально оцінити якість згенерованих ПВЧ, а саме кількісні показники усіх тестів NIST, і відібрати задовільні для споживача. Перевага бази даних над файлом у тому, що можна досить швидко знайти послідовності певної якості (наприклад, Monobit Test > 75%), проте робить систему більш громіздкою.

Паралелізація процесу фільтрації значно прискорить результат по числу запущених паралельних потоків (кількості ядер процесора). Проте, зазвичай якщо якість ПВЧ стоїть на першому місці, швидкість не має великого значення.

При тестуванні системи, в якості експерименту, було впроваджено масштабування апаратної частини гібридного генератора засобами технологій Kubernetes [95] та Docker [96]. Вдалося розгорнути 4 клони апаратної частини, а затравку для них забезпечувала веб-камера. Кожен клон запускався у окремому Docker-контейнері і оркеструвався за допомогою технології k8s. При цьому продуктивність дійсно зросла кратно кількості клонів. У нашому випадку при чотирьох розгорнутих клонах з 288 Мбіт/сек до 1.1 Гбіт/сек, але таке масштабування вимагає додаткового інтерфейсу оркестрації, що ускладнює систему.

3.2.6. Діаграми класів інформаційної системи

Система була побудована у дослідницьких цілях по патерну MVC (Model-View-Controller) на базі монолітної архітектури. На рис.3.9. показана діаграма класів рівня «Controller».

технології JavaFx взаємодіє з методами контролерів згідно принципів патерна проектування «Фасад».

На рис. 3.10 зображено діаграма класів рівня «Model», який являється базовою основою всієї системи.

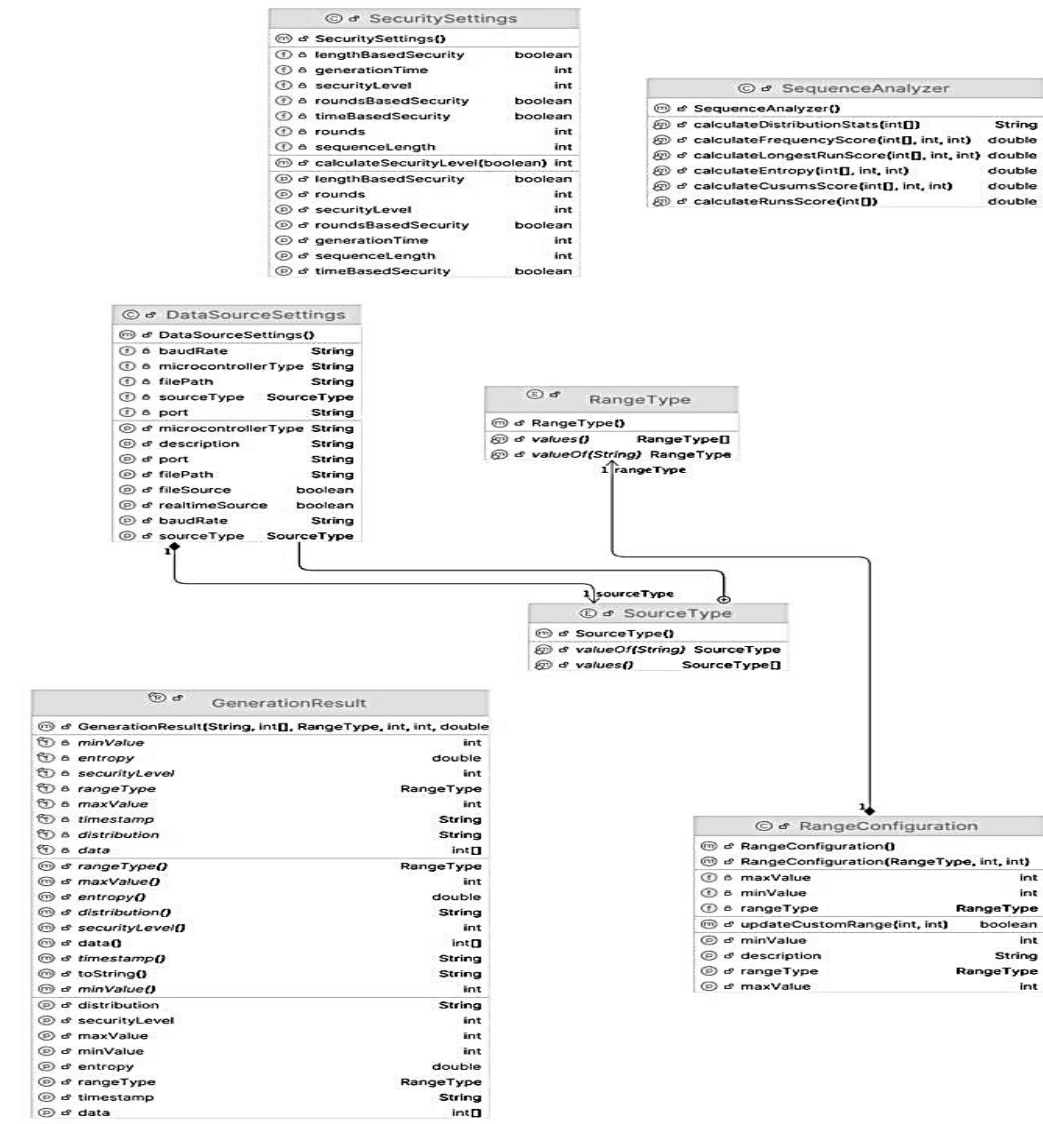


Рисунок 3.10 – Діаграма класів рівня «Model»

На рис. 3.10 діаграми класів рівня «Model» показано, що рівень характеризує РОЮ-класи, об'єкти яких постійно обробляються в пам'яті системи. Туди відносяться об'єкти налаштування, наприклад DataSourceSettings; об'єкти згенерованих послідовностей – GenerationResult: і

об'єкти, що характеризують статистичні характеристики згенерованих послідовностей — SequenceAnalyzer.

На рис.3.11 зображено діаграма класів рівня «Service», який являється основним функціональним рівнем інформаційної системи.

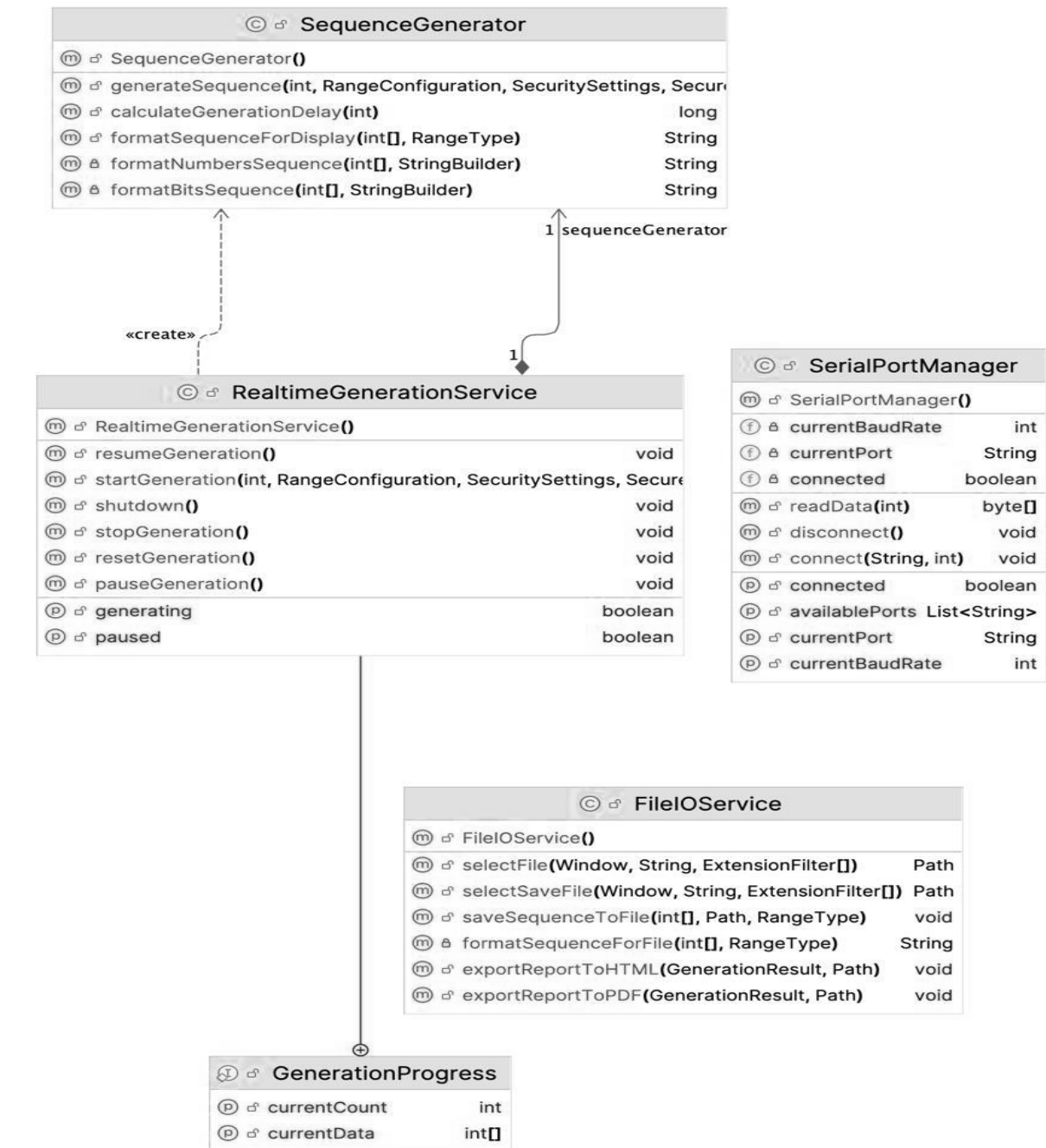


Рисунок 3.11 – Діаграма класів рівня «Service»

На рис. 3.11 зображена діаграма класів рівня «Service». Саме на цьому

рівні відбуваються основні внутрішні операції інформаційної системи: операції вводу даних через USB, сегрегація одержаних величин, екстракція даних з кадру веб-камери, аналіз одержаних послідовностей та ін.

3.2.7. Тестування одержаних ПВЧ

Для дослідження було згенеровано 10 ПВЧ відповідно до функціоналу гібридного алгоритму: веб-камера – початкове значення (seed) – гібридний генерація на основі КА-30. Розмір кожного дорівнює 100 Мбіт. Отримані результати NIST-тестування (NIST TESTING RESULTS) представлені в таблиці 3.1.

Таблиця 3.1.

Результати тестування NIST

№	Test name	True Percentage
1	Frequency (Monobit)	100
2	Block Frequency	100
3	The Runs Test	100
4	Longest Run of Ones	100
5	Binary Matrix Rank	100
6	Discrete Fourier	100
7	Non-overlapping Template Matching	25
8	Overlapping Template Matching	75
9	Universal Statistical	100
10	Linear Complexity	100
11	Serial	100
12	Approximate Entropy	100
13	Cumulative Sums	100
14	Random Excursions	55
15	Random Excursions Variant	60

Згідно з результатами, система генерує криптографічно захищені випадкові послідовності. 9 з 15 тестів показують 100% сумісність.

Тести «Random Excursions» та «Random Excursions Variant» відповідають за величину кумулятивної суми випадкових блукань. Вони знаходять послідовності фіксованої довжини і перевіряють повтор кумулятивні суми.

Система була протестована шляхом масштабування апаратної частини гібридного генератора за допомогою двох технологій Kubernetes [95] та Docker [96]. Це дозволило розгорнути 4 клони апаратної частини, а початкове значення (seed) для них створювала веб-камера. Кожен окремий клон завантажувався у окремому Docker-контейнері, а оркестрування здійснювалося за допомогою технології k8s. За результатами тестування виявилось, що продуктивність зростає кратно кількості клонів. У нашому випадку при чотирьох розгорнутих клонах продуктивність збільшилася майже у чотири рази - від 288 Мбіт/сек на одному клоні до 1.1 Гбіт/сек на чотирьох. Недоліком масштабування є додатковий інтерфейс оркестрації, а це ускладнює систему.

3.3. Порівняльні дослідження технічних характеристик створеної інформаційної системи на основі ФЕЯ та відомих аналогів

За результатами дослідження новоствореної інформаційної системи для генерації ПВЧ на основі ФЕЯ та КА, яка отримала назву PhPhRNG, було проведено порівняльне дослідження її характеристик та характеристик аналогів, описаних у першій главі.

Порівняльні характеристики генераторів ChaosKey, OneRNG, NeuG та PQ4000KS та створеної інформаційної системи PhPhRNG, наведено в таблиці 3.1. Характеристики узагальнено за відкритими технічними описами, публікаціями розробників та результатами незалежних досліджень [1–43].

Таблиця 3.1

Порівняльні характеристики генераторів ChaosKey, OneRNG, NeuG,
PQ4000KS та PhPhRNG

Характеристика	ChaosKey	OneRNG	NeuG	PQ4000KS	PhPhRNG
Тип генератора	Апаратний TRNG на основі електронного шуму [15, 21-28]	Апаратний TRNG на основі лавинного шуму [29-33]	Апаратний TRNG на основі шуму АЦП [33-35]	Високоточний квантовий апаратний TRNG [17, 36-39, 40]	Апаратний на основі ФЕЯ [123,133,138]
Фізичне джерело ентропії	Зворотно зміщені біполярні транзистори (reverse-biased p–n junction) [15, 21]	Лавинний діод; радіочастотний шум з антени [29-33]	Шум процесу дискретизації (ADC noise) у мікроконтролері STM32F103 [33, 34]	Квантовий/дробового-шум у спеціалізованому квантовому генераторі [17, 36-39, 40]	ФЕЯ, шум процесу дискретизації [123, 133,138]
Форм-фактор	Зовнішній USB-модуль у компактному корпусі [15, 21-28]	Зовнішній USB-модуль (стикова плата з екрануванням) [29–31]	USB-модуль, інтегрований у невеликий корпус [33–35]	Зовнішній USB-пристрій настільного/рекового класу [17, 36–38]	Зовнішній USB-модуль (стендова плата з термостатом) [123, 133,138]

Продовження таблиці 3.1

Характеристика	ChaosKey	OneRNG	NeuG	PQ4000KS	PhPhRNG
Інтерфейс підключення	USB 2.0 Full Speed; інтеграція через <code>/dev/hwrng</code> , <code>/dev/chaoskeyX</code> [15, 21, 26-30]	USB/Serial (CDC); доступ як послідовний порт [29–33]	USB CDC; пристрій / <code>dev/ttyACM0</code> у GNU/Linux [33–35]	USB; власні драйвери та ПЗ для основних ОС [17, 36–38]	USB 2.0 Full Speed; інтеграція через Java <code>usbx</code> -драйвер [123, 133,138]
Орієнтовна швидкість генерації	≈ 10 Мбіт/с [28]	$\approx 0,35$ Мбіт/с [29, 33]	$\approx 0,64$ Мбіт/с (понад 80 кБ/с) [35]	≈ 4 Мбіт/с [17, 36, 37]	0,980 - 288 Мбіт/с [123, 133,138]
Підтримувані платформи	Переважно GNU/Linux; використання драйвера ядра [27]	GNU/Linux, *BSD та інші ОС із підтримкою послідовного USB- інтерф. [29–33]	GNU/Linux та інші POSIX-системи з підтримкою USB-CDC [33–35]	Windows, GNU/Linux та інші ОС, що підтримуються виробником [17, 36–38]	Windows, GNU/Linux, MacOS [123, 133,138]

Продовження таблиці 3.1

Характеристика	ChaosKey	OneRNG	NeuG	PQ4000KS	PhPhRNG
Підтримувані платформи	Переважно GNU/Linux; використання драйвера ядра [27]	GNU/Linux, *BSD та інші ОС із підтримкою послідовного USB-інтерфейс [29–33]	GNU/Linux та інші POSIX-системи з підтримкою USB-CDC [33–35]	Windows, GNU/Linux та інші ОС, що підтримуються виробником [17, 36–38]	Windows, GNU/Linux, MacOS [123, 133, 138]
Відкритість апаратури та ПЗ	Відкрита апаратна схема (TAPR OHL) та прошивка (GPLv2) [15, 21]	Open-hardware та open-source ПЗ (GPLv3) [29–31]	Вільне ПЗ (GPLv3); відкритий вихідний код та документація [33–35]	Промислове пропріетарне рішення; закрита апаратура та ПЗ [17, 36–38]	Open-hardware та open-source ПЗ (GPLv3) [144]
Вбудовані тести та пост-обробка	Базові внутрішні перевірки; подальший аналіз покладється на ОС [15, 21]	Примітивне відбілювання та моніторинг стану шумового джерела [29–31]	Криптографічне кондиціонування на основі SHA-256; Використання статистич-	Безперервне апаратне тестування сирого та остаточного потоку; блокування виходу при зниженні	Базові внутрішні перевірки; регулювання якості і швидкості вихідного потоку, регулювання

Продовження таблиці 3.1

Характеристика	ChaosKey	OneRNG	NeuG	PQ4000KS	PhPhRNG
			них пакетів [33–35, 39]	ентропії [17, 36, 37, 28]	балансу, ефективна постобробка КА [123, 133, 138]
Сертифікація та стандарти	Формальної сертифікації FIPS/CC немає; орієнтація на FLOSS [15, 21–28]	Відсутність FIPS/CC; наголос на відкритості та верифікованості [29–32]	Відсутність формальної сертифікації; орієнтація на GNU/FLOSS-спільноту [33–35]	Орієнтація на відповідність NIST SP 800-90B/C, застосування у FIPS-сумісних системах [17, 36–38, 28, 41]	Відсутність формальної сертифікації
Типові сценарії використання	Підживлення пулу ентропії серверів і робочих станцій, дослідження, віртуалізо-	Робочі станції, невеликі сервери, маршрутизатори, open-source інфраструктура [29–32,	Інфраструктура вільного ПЗ, лабораторні дослідження, навчальні стенди [33–35]	Гемблінг, фінансові системи, критичні інформаційні системи [17, 36–39; 40]	Стеганографія, хеш-функції, Електронні ключі

Продовження таблиці 3.1

Характеристика	ChaosKey	OneRNG	NeuG	PQ4000KS	PhPhRNG
	вані середовища [15, 21–28, 42, 43]	42, 43]			
Основні переваги	Повністю відкритий дизайн; підтримка ядром Linux; висока швидкодія; компактність [15, 21–28]	Відкрите апаратне забезпечення; можливість перевірки та перепрошивки; використання двох джерел шуму [29–32]	Повністю вільне ПЗ; прозора реалізація; використання криптографічного кондиціонування [33–35]	Висока якість випадковості; безперервні тести; орієнтація на суворі галузеві вимоги [1, 36–39; 40]	Висока недетермінована ентропія, гнучке налаштування, відкритий дизайн, швидка постобробка [123, 133, 138]
Основні недоліки	Відсутність формальної сертифікації; зняття з виробництва; необхідність USB-порту	Порівняно низька швидкодія; відсутність офіційної сертифікації; залежність від активності	Невисока пропускну здатність; орієнтація переважно на GNU/Linux-середовище [33–35]	Висока вартість; закритість внутрішньої реалізації; нішевий характер продукту [17, 36–39;	Потреба у мініатюризації, підвищення продуктивності

Продовження таблиці 3.1

Характеристика	ChaosKey	OneRNG	NeuG	PQ4000KS	PhPhRNG
	[15, 21–28]	спільноти [29–32]		40]	

* Примітка: наведені швидкості є орієнтовними та ґрунтуються на відкритих технічних специфікаціях і узагальнених результатах вимірювань [17, 28, 29, 33, 35, 36, 37].

3.3.1. Узагальнення результатів порівняння технічних характеристик створеної інформаційної системи на основі ФЕЯ та відомих аналогів

Проведений порівняльний аналіз показує, що ChaosKey, OneRNG та NeuG орієнтовані насамперед на спільноту відкритого програмного забезпечення, де ключовими є прозорість реалізації, можливість незалежного аудиту та низький поріг входження за вартістю.

Вони забезпечують достатню для більшості серверних і робочих сценаріїв пропускну здатність, але не мають формальної сертифікації за FIPS або Common Criteria, що може обмежувати їх застосування в строго регульованих галузях [11–21, 26, 27].

Об'єднуючи позитивні моменти вищезгаданих рішень PhPhRNG має на додаток гнучку систему налаштувань для балансу оптимальних значень швидкості\якість.

На противагу цьому, Comscire PQ4000KS є продуктом високої ціни, орієнтованим на сегмент високої довіри, де домінують вимоги до відповідності рекомендаціям NIST і галузевим стандартам, а також необхідність безперервного моніторингу якості статистичних характеристик згенерованих ПВЧ.

Закритість реалізації частково компенсується наявністю детальної документації щодо процедур тестування та позиціонуванням як квантового генератора з внутрішніми механізмами контролю.

У порівнянні з сучасними квантовими генераторами на прикладі RQ4000KS, можна сказати що PhPhRNG наближається до квантових генераторів по продуктивності при низькій ціні (майже на 2 порядки – 100 разів).

У порівнянні з генераторами на основі шуму PhPhRNG відрізняється в позитивну сторону своєю простотою, надійністю та гнучкістю настройки, при цьому випереджаючи відомі аналоги по продуктивності майже на порядок.

Варто зазначити, що створена інформаційна система передбачає у своєму розвитку машинне навчання на основі нейронних мереж та хмарних технологій.

Висновки до розділу 3

Розроблено та досліджено інформаційну систему для генерації високоентропійної ПВЧ на основі ФЕЯ. Інтерфейс системи дозволяє керувати параметрами генерації ПВЧ, а саме:

- обирати джерело ПВЧ (база даних, файл, на основі інтенсивності пікселів веб-камери, фото- та темного струму фотодіода, гібридного з обробкою ПВЧ за допомогою КА);
- обирати між різними діапазонами значень (біти 0-1, стандартний діапазон 1-256, або користувацький діапазон);
- регулювати баланс між швидкістю генерації та рівнем безпеки, який забезпечує згенерована ПВЧ, зберігати її та аналізувати.

Продуктивність інформаційної системи лежить у межах 0,980 – 288Мбіт/с і може бути суттєво збільшена при використанні високочастотної апаратної частини. Також система передбачає можливість масштабувати всю програмну частину генератора або ж розпаралелити процес обробки «сирої» ПВЧ

всередині блоку програмної частини. Що, на прикладі ПВЧ, згенерованої на основі веб-камери, дозволило підвищити продуктивність у 4 рази і досягти рівня продуктивності у 1 Гбіт/сек.

Основні наукові результати розділу опубліковані в працях [92].

РОЗДІЛ 4

ЗАСТОСУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ГЕНЕРАЦІЇ ВИСОКОЕНТРОПІЙНОЇ ПВЧ НА ОСНОВІ ФЕЯ

4.1. Сфери застосування інформаційної системи для генерації високоентропійної ПВЧ на основі ФЕЯ

Створена інформаційна система для генерації високоентропійної ПВЧ на основі ФЕЯ може бути застосована у низці напрямків інформаційних технологій загалом і інженерії програмного забезпечення зокрема. Частина інформації щодо застосування згенерованих ПВЧ, подана у третьому розділі (рис. 3.8, таблиця 3.1). Тут, ми більш детально висвітлимо результати, отримані у дослідженні.

4.1.1 Створення вхідного вектора для хеш-функції

4.1.1.1 Алгоритм хеш-функції

Однією з основних задач будь-якої криптографічної функції є генерування випадкових результатів. Будь-яка хеш-функція не є генератором випадкових значень сама по собі, оскільки вона не продукує бінарні послідовності довільної довжини [97 - 99].

Математична модель, що описує рівень чутливості до лавинного ефекту (лавинний критерій) заснована на критерії незалежності бітів, згідно з яким при зміні одного вхідного біта будь-які два вихідні біти змінюються незалежно один від одного [100].

Функція $f: \{0,1\}^n$ задовольняє критерію незалежності бітів, якщо будь-яких $i, j, k \in \{1, 2, \dots, n\}$, де $j \neq k$, інвертування біта на вході викликає зміни бітів j і k на виході причому ці зміни незалежні до певного ступеня.

Для вимірювання ступеня незалежності двох вихідних бітів вводять коефіцієнт кореляції $BIC(a_j, a_k)$ – (Bayesian information criterion [101]) між j -ю та

k -ю компонентою вихідного вектора для зміненого i -ого компонента вхідного вектора.

$$BIC = \max_{1 \leq j, k \leq n, j \neq k} BIC(a_j, a_k) \quad (4.1.)$$

Цей параметр показує, наскільки функція f відповідає критерію незалежності бітів.

Він приймає значення в проміжку $[0,1]$, і в кращому випадку дорівнює 0, тоді можна говорити про повну незалежність, у найгіршому 1, коли є повна залежність.

У запропонованому алгоритмі [102] залежність вихідного біта від вхідного має псевдо-хаотичний характер згідно з визначенням хаотичної групи клітинних автоматів.

Таким чином, для однієї ітерації коефіцієнт кореляції $BIC = 0,5$. Для 128 ітерацій цей коефіцієнт дорівнюватиме

$$BIC(a_i, a_j) = (0.5)^{128} \rightarrow 0. \quad (4.2)$$

Сучасні алгоритми хешування забезпечують коефіцієнт кореляції 0,5. Тобто при хешуванні двох однакових рядків, що відрізняються на 1 біт, приблизно половина (0,5) байт збігатиметься.

У разі запропонованого алгоритму, збіги мають практично незалежний характер. Це означає, що, керуючи кількістю ітерацій, можна звести чутливість алгоритму до лавинного ефекту практично до максимальної. При цьому значно зростають ресурсні витрати.

З іншого боку, зменшуючи кількість ітерацій можна досягти прийняттого рівня кореляції між швидкодією та чутливістю алгоритму до лавинного ефекту.

Оскільки для функції стиснення на вхід потрібно 2 масиви, а на першому етапі ще немає хеш-масиву з попереднього кроку, то замість нього використовується вхідний вектор.

Блок-схему формування вхідного вектора наведено на рис. 4.1.

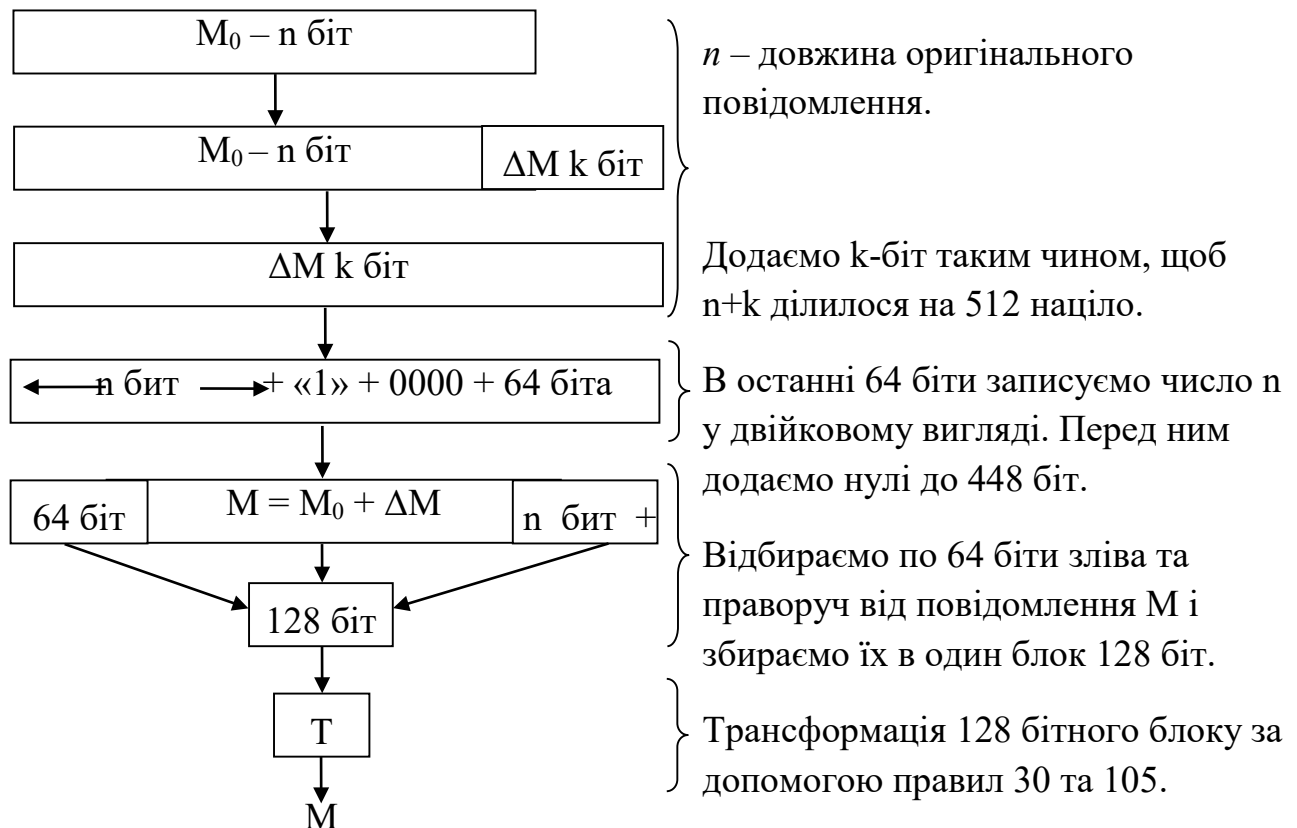


Рисунок 4.1 – Блок-схема формування вхідного вектора

M_0 – повідомлення довжиною n біт.

Вхідне повідомлення доповнюється нульовими бітами так, щоб його довжина була кратна 512 і потім використовується як джерело даних для вектора. З доповненого вхідного повідомлення вибираються перші 64 і останні 64 біти, і поєднуються в 128-бітний масив. Використовуючи отриманий ключ як вихідний стан клітинного автомата, до нього застосовують змішані правила 30 і 105, таким чином, що до всіх непарних осередків застосовується 30 правило, а до парних - 105. В результаті 128 ітерацій виходить 128-бітний блок, який буде використаний разом з першим блоком тексту.

Безумовно, 128 ітерацій під час створення вхідного блоку уповільнює роботу алгоритму, але збільшується чутливість до лавинного ефекту. Але це,

своєю чергою, ускладнює «оборотність» процесу, тобто отримання прообразу з хеш-образу [103].

Як зазначалося вище, порожні місця вхідного вектора, заповнюються нулями до його потрібного розміру, у байтах. І ці нулі при обробці клітинними автоматами, не змінюються. Тому, рівень хаотичності створеної конструкції, знижується пропорційно кількості цих нулів, що впливає на його криптостійкість.

Для усунення цієї проблеми, а саме підвищення ступеня хаотичності та продуктивності генерації вхідного вектора хеш-функції, запропоновано ввести у систему хешування ПВЧ, згенеровану за допомогою нашої інформаційної системи з використанням гібридного генератора на основі функціоналу КА та темного струму фотодіода (рис. 3.8). Згенеровані генератором ПВЧ числа можуть заповнювати вільний простір вхідного вектора.

А саме – замість нулів, мають стояти числа із згенерованої ПВЧ [102]. Якщо останній блок і без того 512, як варіант пропонується згенерувати ПВЧ рівно у 512 біт і через функціонал XOR розчинити останній блок у згенерованій послідовності.

Блок схема формування вхідного вектора при використанні ПВЧ, згенерованої за допомогою створеної інформаційної системи, наведена на рис. 4.2.

На відміну від алгоритму зображеному на рис. 4.1 даний алгоритм оминає досить затратний функціонал формування вхідного вектора за допомогою обробки вхідного повідомлення на основі 128 ітерацій клітинних автоматів. Хоча сам функціонал клітинних автоматів базується на бітових операціях мови програмування Java, які являються самими примітивними і в результаті швидкісними, все ж таки 128 ітерацій займуть деякий час.

У нашому випадку розглядається обчислення 128-бітного хешу, а у випадку 512-бітного хешу та 1024-бітного кількість ітерацій зростає відповідно до 512 та 1024, і цей час зростає на порядок, блок схема такого випадку

аналогічна схемі на рис.4.2. за винятком того, що додавати додаткові біти треба так, щоб загальна довжина ділилась на 1024.

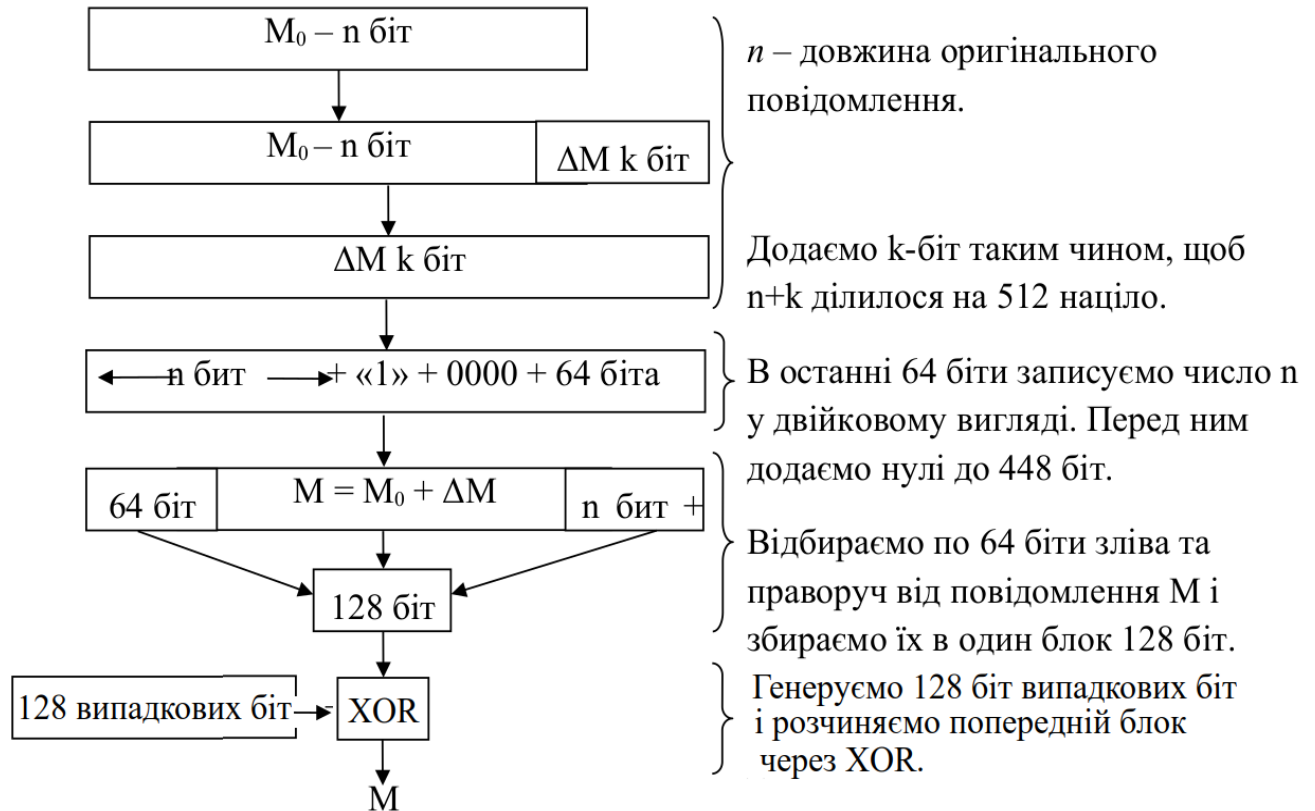


Рисунок 4.2 – Блок схема формування вхідного вектора хеш-функції при використанні випадкової послідовності бітів.

Такий підхід має сенс при порівняно невеликих інформативних повідомлення – до кілобайта. У випадку великих об'єктів, що потребують хешування, наприклад, фото та відео об'єкти, які можуть займати пам'ять до 1 гігабайту, формування вхідного вектора займе малий час у порівнянні з обробкою основного об'єму інформації. Проте при пересиланні невеликих повідомлень (наприклад, банківські транзакції та перетворення блокчейну) швидкість генерації хешу може зіграти далеко не другорядну роль.

При цьому лавинний ефект генерації хешу досягає самого високого рівня, оскільки залежить виключно від рівня хаосу і ентропії, що забезпечує генератор ПВЧ.

Для візуального експерименту ефективності функціоналу КА було взято вже знайому з попередніх експериментів фразу «the quick brown fox jumps over the lazy dog». На відміну від попередніх експериментів, де фраза мінялась на цілий байт (8 біт), у цьому експерименті фраза мінялась на біт. Результати експерименту наведено у таблиці 4.1.

Візуально видно помітно високий рівень лавинного ефекту на прикладі цих речень. Проте для чисельної оцінки рівня лавинного ефекту були проведені додаткові експерименти.

Таблиця 4.1.

Набір рядків та хеш-зображень, які були оцінені під час експерименту з дослідженням лавинного ефекту, змінюючи речення «the quick brown fox jumps over the lazy dog» на 1 біт

Рядок	Хеш-зображення
the quick brown fox jumps over the lazy dog	b8f1116a1814feadd6f1d45333298041
the quick brown fox jumps over the lazy dof	ae66ade8191152faed78ced52ff67ea87
the quick brown foS jumps over the lazy doi	6cdeacf56fe3906a1eccd7731bcc7e759

В процесі експерименту виявилось, що спрощення генерації вхідного вектора скорочує повний час генерації хешу приблизно на 30%. А рівень лавинного ефекту прямо пропорційно залежить від випадковості і непередбачуваності ПВЧ, що згенеровані генератором.

4.1.1.2. Удосконалений алгоритм створення хеш-функції за допомогою розробленої інформаційної системи генерації ПВЧ на основі ФЕЯ

З урахуванням вище отриманих результатів удосконалено алгоритм хешування, наведений у попередньому параграфі. В удосконаленому алгоритмі вхідний вектор створюється за допомогою розробленого гібридного генератора ПВЧ на основі темного струму фотодіода. Послідовність створення хеш-функції наведена на рис. 4.3.

Вхідний текст (об'єкт) M , що потребує хешування, в самому кінці повідомлення доповнюється нулями ΔM так, щоб сумарна кількість бітів ділилась на 512 націло. Потім перший блок розбивається на 4 підблоки по 128 біт. Для кожного з цих 4-х блоків застосовуємо циклічний зсув вліво. Паралельно з цим готується вхідний вектор 128 біт. Генерується він за допомогою генератора ПВЧ на основі темного струму фотодіода.

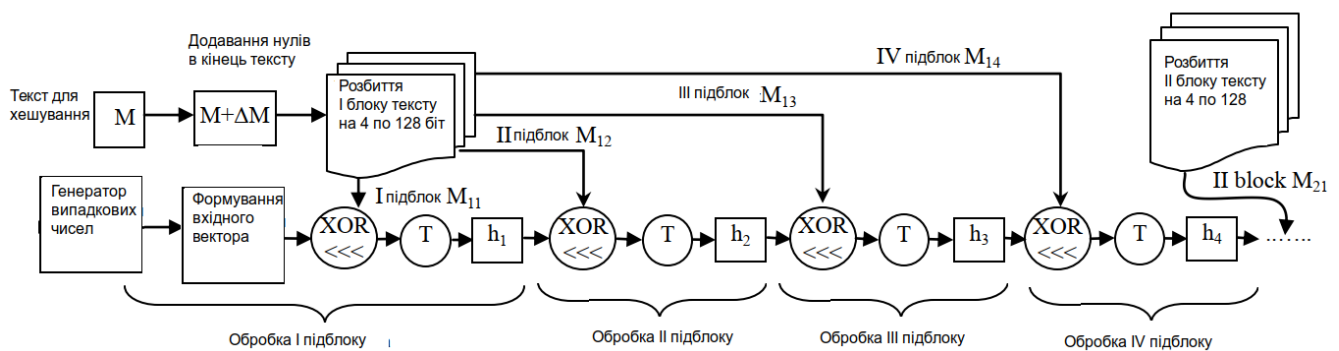


Рисунок 4.3 – Послідовність створення хеш-функції $M1$ (першого 512 біт) блоку тексту. T – трансформація за допомогою клітинних автоматів, XOR <<< – стиск зі зсувом.

Далі перший підблок стискається ($XOR<<<$) з вхідним вектором, що згенеровано за допомогою генератора ПВЧ. Оброблений перший підблок (I) трансформується (T) за допомогою клітинних автоматів. На цьому етапі

можливе налаштування алгоритму для досягнення необхідного рівня лавинного ефекту. А саме:

1. Мінімальні трансформації – це один цикл обробки клітинними автоматами.

2. Максимальні трансформації – це 128 циклів обробки клітинними автоматами – правила 90 та 150. І, як результат, взяти діагоналі побудованої матриці 128×128 , де по вертикалі представлені результати трансформації для кожного з 128 кроків. Парні рядки – це результат роботи 90-го правила, непарні – 150-го правила. Взяті діагоналі обробляються тим самим XORом.

3. Проміжне значення n -ої ітерації ($n < 128$).

Після виконання цієї функції виходить 128-бітний хеш-образ першого підблоку першого блоку вхідного тексту. Який надалі буде вхідним вектором обробки наступного підблоку.

Переваги даного підходу від аналогів [104 - 110] полягають у тому, що заміна програмної генерації вхідного вектора на генератор ПВЧ і гнучке налаштування функції трансформації дає змогу зменшити час генерації у деяких випадках майже на порядок.

Так, наприклад, при генерації 1024-бітного хеша відомої фрази «Hello, world!» час генерації зменшився з 35 мкс до 7 мкс.

Такий експеримент не дає повної картини всіх переваг, бо значною мірою час генерації залежить від обчислювальних потужностей і довжини тексту. Проте для відносного порівняння можливостей такий експеримент може мати сенс.

4.1.2. Застосування інформаційної системи на основі ФЕЯ для створення ПВЧ у стеганографії

Стеганографія є важливим напрямом інформаційної безпеки, спрямованим на приховування самого факту передавання повідомлення шляхом його вбудовування в цифрові контейнери різної природи. На відміну

від криптографічних методів, які забезпечують конфіденційність змісту повідомлення, стеганографічні системи орієнтовані на досягнення статистичної непомітності прихованого каналу зв'язку [111]. Саме випадковість розподілу змін у контейнері дозволяє істотно знизити ефективність класичних методів стеганоаналізу.

Актуальність застосування високоякісних ПВЧ зростає у зв'язку з розвитком методів машинного навчання та статистичного аналізу, які здатні виявляти навіть незначні закономірності у структурі контейнера [112, 113], який застосовується для збереження корисної інформації, розчиненої у хаосі ПВЧ, наприклад біти інформативного повідомлення інжектуються у ПВЧ по ключу.

Для використання у стеганографічних системах генератори ПВЧ повинні відповідати криптографічним і статистичним вимогам. Згідно з рекомендаціями [22], такі генератори мають відповідати низці вимог:

- високий рівень ентропії;
- відсутність коротких періодів повторення;
- криптографічна стійкість;
- відсутність корельованих підпоследовностей;
- стабільність характеристик у часі.

У контексті підвищених вимог до непередбачуваності та ентропії доцільним є застосування апаратних або гібридних генераторів випадкових чисел. Саме таким вимогам відповідає створений генератор ПВЧ на основі ФЕЯ, який входить до складу відповідної інформаційної системи.

Фізичні джерела шуму, які в ньому застосовані, забезпечують істинну випадковість, тоді як програмна обробка дозволяє стабілізувати та контролювати статистичні характеристики вихідної послідовності [112].

Для перевірки ефективності застосування ПВЧ, створених нашою інформаційною системою, у складі стеганографічних алгоритмів було проведено відповідне дослідження. Було реалізовано декілька алгоритмів.

1. Формальна модель гібридного генератора ПВЧ.

Нехай:

- $H = \{h_1, h_2, \dots, h_n\}$ – бітова послідовність, отримана з апаратного джерела ентропії;

- K – секретний стеганографічний ключ;

- $G(\cdot)$ – криптографічно стійкий ПВГ.

Формування початкового стану ПВГ здійснюється за схемою:

$$S_0 = H(K \parallel H),$$

де $H(\cdot)$ – криптографічна хеш-функція,

$\parallel$ - операція конкатенації.

Отриманий стан S_0 використовується для ініціалізації ПВЧ, який генерує випадкову послідовність:

$$P = \{p_1, p_2, \dots, p_L\}.$$

Таким чином, кожна реалізація послідовності P є:

- відтворюваною для легітимного користувача (за наявності ключа K);
- криптографічно непередбачуваною для зломисника через вплив фізичної ентропії.

2. Алгоритм вбудування (гібридний варіант).

Нехай:

$C = \{c_1, c_2, \dots, c_N\}$ – контейнер;

$M = \{m_1, m_2, \dots, m_L\}$ – бітова послідовність повідомлення.

Тоді алгоритм вбудування має вигляд:

1. Зчитуємо апаратну ентропію H (згенеровану ПВЧ).

2. Обчислюємо початковий стан S_0 .

3. Ініціалізуємо ПВЧ станом S_0 .

4. Генеруємо ПВЧ позиції P .

5. Для кожного m_i виконати: $c'_{p_i} = (c_{p_i \& \sim 1}) \mid m_i$

У результаті, таким чином формується стеганоконтейнер C' .

3. Алгоритм вилучення.

Процедура вилучення є симетричною, за винятком джерела ентропії:

- Використовуємо збережений або відновлюваний фрагмент H (або його хеш/seed).

- Повторно обчислюємо $S_0 = H(K \parallel H)$.

- Генеруємо ідентичну послідовність P .

- Витягаємо біти: $m_i = c_{p_i} \& 1$.

У практичних реалізаціях допускається збереження H у зашифрованому вигляді, або накопичення ентропії лише на етапі ініціалізації сеансу.

У таблиці 4.2 наведено порівняння вищеописаних алгоритмічних підходів, які базуються на звичайній (програмній) ПВЧ, апаратній ПВЧ та ПВЧ, отриманій гібридним методом, запропонованим нами, а саме – генерація ПВЧ з фотоелектричних явищ із подальшою обробкою цих ПВЧ за допомогою КА.

Таблиці 4.2.

Порівняння алгоритмічних підходів стеганографії із застосування ПВЧ, створених у різний спосіб.

Критерій	Програмна ПВЧ	Апаратний генератор ПВЧ	Гібридний генератор ПВЧ
Відтворюваність	Так	Ні	Так
Істинна ентропія	Ні	Так	Так
Продуктивність	Висока	Низька–середня	Висока
Стійкість до прогнозування	Середня	Висока	Дуже висока
Придатність для стеганографії	Обмежена	Обмежена	Висока

Як видно з таблиці 4.2 гібридний генератор ПВЧ, який входить до складу створеної інформаційної системи, відповідає усім вимогам, що висуваються до стеганографічних систем у порівнянні з програмними та апаратними генераторами ПВЧ. Аналіз наукових джерел свідчить, що ефективність стеганографічних алгоритмів значною мірою визначається якістю використаних ПВЧ та відповідністю генераторів сучасним криптографічним вимогам.

Гібридний генератор ПВЧ, створений у процесі виконання дисертаційного дослідження, поєднує переваги псевдовипадкових і апаратних підходів, що робить його особливо придатним для стеганографічних систем із підвищеними вимогами до прихованості та стійкості.

Використання апаратної ентропії на етапі ініціалізації дозволяє усунути детермінованість, тоді як програмна генерація забезпечує ефективність, синхронізацію та масштабованість алгоритму.

4.1.3 Застосування інформаційної системи на основі ФЕЯ для створення ПВЧ при формуванні криптографічних ключів

Криптографічні ключі є базовим елементом безпеки симетричних і асиметричних криптосистем. Їх криптостійкість безпосередньо визначається рівнем ентропії та непередбачуваності процесу генерації. У разі недостатньої випадковості навіть формально коректні алгоритми шифрування можуть бути скомпрометовані шляхом перебору або статистичного аналізу ключового простору [114].

Згідно з рекомендаціями NIST, генерація криптографічних ключів повинна ґрунтуватися на джерелах ентропії з доведеними статистичними властивостями, а також на механізмах, стійких до прогнозування та відновлення внутрішнього стану генератора [22, 24].

Псевдовипадкові генератори, реалізовані виключно програмними засобами, характеризуються детермінованістю: за відомого початкового стану

(seed) їхній вихід є повністю відтворюваним. У контексті криптографії це означає, що:

- компрометація seed призводить до повної компрометації ключів;
- можливі атаки відновлення внутрішнього стану;
- існує ризик повторного використання або кореляції ключів у різних сеансах.

Саме ці обмеження зумовлюють необхідність застосування гібридного підходу, який поєднує програмну генерацію з фізичними джерелами ентропії.

Виходячи з наведеного вище, з урахуванням архітектури генератора для утворення ключів, розроблена інформаційна система була застосована для розв'язання цієї задачі.

Гібридний генератор псевдовипадкових чисел для криптографічних застосувань складається з таких компонентів:

1. Апаратне джерело ентропії (фотодіод);
2. Механізм накопичення (БД на рис 3.8);
3. Функціонал вирівнювання ентропії - (КА у нашому випадку);
4. Криптографічна функція змішування;
5. Криптографічно стійкий генератор ПВЧ (як приклад, функціонал `java.util.SplittableRandom`).

Нехай:

- $H = \{h_1, h_2, \dots, h_n\}$ – бітова послідовність апаратної ентропії (наприклад фотодіод);
- K_u – користувачський або системний секрет;
- $\text{ХЕШ}(\cdot)$ – криптографічна хеш-функція.

Початковий стан генератора формується як:

$$S_0 = \text{ХЕШ}(H \parallel K_u)$$

Такий підхід гарантує, що навіть у разі часткової компрометації одного з компонентів (наприклад, (K_u)) повна відтворюваність стану генератора залишається неможливою без знання апаратної ентропії.

Після ініціалізації гібридного генератора формується послідовність псевдовипадкових бітів:

$$R = \{r_1, r_2, \dots, r_m\},$$

яка використовується для побудови криптографічного ключа заданої довжини (l):

$$H(K_{\text{crypto}}) = \text{Trunc}_l(R).$$

Для симетричних алгоритмів (AES, ChaCha20) типові значення

$l \in \{128, 192, 256\}$, тоді як для асиметричних криптосистем (RSA, ECC) вихідна послідовність використовується для генерації параметрів ключової пари.

Сумарна ентропія ключа, згенерованого гібридним методом, оцінюється як:

$$H(K_{\text{crypto}}) > \min (H(H) + H(K_u), l).$$

На відміну від чисто програмного підходу, де:

$$H(K_{\text{PRNG}}) = H(K_u),$$

гібридний генератор забезпечує додаткову фізичну невизначеність, що унеможливорює ефективні атаки прогнозування або відновлення ключів за спостережуваними вихідними даними.

Переваги гібридного підходу з погляду безпеки.

Використання гібридного генератора ПВЧ при створенні криптографічних ключів забезпечує:

- стійкість до атак із відомою або частково відомою затравкою;
- захист від повторного використання ключів;
- зменшення ризику системних дефектів реалізації;
- відповідність сучасним стандартам криптографічної безпеки.

Особливо важливою є властивість некорельованості ключів, згенерованих у різних сеансах, що має критичне значення для систем масового шифрування, РКІ та захищених стеганографічних каналів.

Висновки до розділу 4

Створена інформаційна система для генерації високоентропійної ПВЧ знайшла застосування при формуванні вхідного вектора для хеш-функції, що привело до спрощення генерації самого вхідного вектора та скорочення повного часу генерації хешу приблизно на 30 %.

Гнучке налаштування функції трансформації за допомогою створеної інформаційної системи, дає змогу зменшити час генерації у деяких випадках майже на порядок.

Гібридний генератор ПВЧ, який розроблено у складі створеної інформаційної системи, поєднує переваги псевдовипадкових і апаратних підходів, що робить його особливо придатним для стеганографічних систем із підвищеними вимогами до прихованості та стійкості. Використання апаратної ентропії на етапі ініціалізації дозволяє усунути детермінованість, тоді як програмна генерація забезпечує ефективність, синхронізацію та масштабованість алгоритму.

Також створений гібридний генератор ПВЧ є ефективним та обґрунтованим інструментом для формування криптографічних ключів із високим рівнем ентропії та криптостійкості.

Поєднання апаратної ентропії з програмними криптографічно стійкими генераторами дозволяє усунути детермінованість класичних генераторів ПВЧ та забезпечити відповідність сучасним вимогам стандартів інформаційної безпеки. Зазначений підхід є доцільним як для класичних криптосистем, так і для комплексних рішень, що поєднують криптографію та стеганографію.

Результати роботи, а саме удосконалений алгоритм створення хеш-функції, стеганографічні алгоритми та алгоритми генерації криптографічних ключів за допомогою розробленої інформаційної системи генерації ПВЧ,

впроваджені та використовується у ТДВ ЗАВОД «Кварц», системою автоматизації управління підприємством АІС «Kaskad» та компанії «Datawiz» для підвищення криптостійкості обміну даними.

Основні наукові результати розділу опубліковані в працях [102, 103].

ВИСНОВКИ

У дисертаційній роботі поставлено і розв'язано актуальну науково-прикладну задачу розробки інформаційної системи для генерації високоентропійних послідовностей випадкових чисел та оцінювання ступеня їх хаотичності на основі програмного забезпечення власної розробки та нових підходів для визначення балансу між продуктивністю та надійністю створених послідовностей випадкових чисел.

На основі аналізу результатів, отриманих в ході дослідження, створена відповідна інформаційна система генерації ПВЧ, яка застосовується до задач швидкого формування вхідного вектора для хеш-функції, для стеганографічних систем із підвищеними вимогами до прихованості та стійкості, а також для формування криптографічних ключів із високим рівнем ентропії та криптостійкості.

Отримані результати впроваджено у освітній процес вищого навчального закладу та роботу підприємств.

Основні результати дисертаційної роботи:

1. Проведено аналітичний огляд існуючих програмних та апаратних засобів для генерації високоентропійної ПВЧ в системах інформаційної безпеки. Розглянуто їх значення у криптографічних системах та показані теоретичні основи оцінювання випадковості та ентропії. Показано, що відомі системи генерації ПВЧ мають досить низьку продуктивність (швидкодію) і не забезпечують баланс між продуктивністю та якістю згенерованої ПВЧ.

2. Вперше створено низку генераторів ПВЧ на основі фотоелектричних явищ для екстракції ПВЧ з низки стохастичних фізичних явищ – інтенсивності пікселів веб-камери, фото- та темнового струму фотодіода, а також обробки отриманих ПВЧ за допомогою технології КА.

3. Вперше, на основі створених генераторів ПВЧ розроблено інформаційну систему для генерації високоентропійної ПВЧ, яка містить

аналітичний модуль інтелектуальної спрощеної статистичної оцінки згенерованих ПВЧ, технологію утримання балансу між продуктивністю та якістю генерації ПВЧ та фреймворк на мові програмування Java по роботі з крипто-примітивами на основі КА.

4. Завдяки проведеному дослідженню набуло подальшого розвитку технологія використання КА. А саме - запропоновано і експериментально доведено високу якість, що генерується КА хаотичного типу (правила 30, 90, 105) про створенні ПВЧ, здійснена оптимізація високої продуктивності і низької ресурсоемності функціоналу на КА.

5. Створена технологія генерації ПВЧ, що переходить на якісний рівень – неперервна генерація ПВЧ заданої продуктивності та якості на простому доступному обладнанні.

6. Проведено порівняльні дослідження характеристик створеної інформаційної системи та відомих аналогів, які показали, що її перевагами є висока недетермінована ентропія, гнучке налаштування, відкритий дизайн, швидка постобробка. Продуктивність інформаційної системи лежить у межах 0,980 – 288 Мбіт/с і може бути суттєво збільшена при використанні високочастотної апаратної частини.

Також система передбачає можливість масштабувати всю програмну частину генератора або ж розпаралелити процес обробки «сирої» ПВЧ всередині блоку програмної частини, що, на прикладі ПВЧ, згенерованої на основі веб-камери, дозволило підвищити продуктивність у 4 рази.

7. Розроблено та запропоновано застосування створеної інформаційної системи для синтезу криптографічних алгоритмів. А саме для генерації хеш-функцій, стеганографії та криптографічних ключів

8. Результати дисертаційної роботи впроваджено у компаніях ТДВ ЗАВОД «Кварц», системою автоматизації управління підприємством АІС «Kaskad» та компанії «Datawiz» для підвищення криптостійкості програмного забезпечення і використовуються в освітньому процесі відділу комп'ютерних

технологій Навчально-наукового інституту фізико-технічних та комп'ютерних наук, Чернівецького національного університету імені Юрія Федьковича.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Указ президента України №37/2022 “Про рішення Ради національної безпеки і оборони України від 30 грудня 2021 року "Про План реалізації Стратегії кібербезпеки України". URL: <https://www.president.gov.ua/documents/372022-41289> (Дата звернення 21.02.2026).
2. H. Fukś, “Four state deterministic cellular automaton rule emulating random diffusion,” In: Chopard, B., Bandini, S., Dennunzio, A., Arabi Haddad, M. (eds) Cellular Automata. ACRI 2022. Lecture Notes in Computer Science, vol. 13402, 2022. Springer, Cham. https://doi.org/10.1007/978-3-031-14926-9_13.
3. Dobrovolsky Y., Hanzhelo D., Hanzhelo M., Trembach D., Prokhorov G. Development of a hash algorithm based on cellular automata and chaos theory. *Eastern-European Journal of Enterprise Technologies*. 2021. № 5/9. V. 113. P. 48-55. DOI: 10.15587/1729-4061.2021.242849.
4. Cicuttin A., De Micco L., Crespo M. Looking for suitable rules for true random number generation with asynchronous cellular automata. *Nonlinear Dynamics*. 2022. V. 111, P. 2711-2722, 2022. <https://doi.org/10.1007/s11071-022-07957-8>.
5. Li L., Luo Y., Qiu S. Image encryption using chaotic map and cellular automata. *Multimed Tools Appl.* V. 81. P. 40755–40773. <https://doi.org/10.1007/s11042-022-12621-9>.
6. Aljahdal A. Random Number Generators Survey. *International Journal of Computer Science and Information Security*. 2020. V. 18. No. 10. P. 57-68. <https://doi.org/10.5281/zenodo.4249406>
7. ID Quantique. Quantis QRNG USB – Quantum Random Number Generator : URL: <https://www.idquantique.com/random-number-generation/products/quantis-random-number-generator/> (дата звернення: 21.02.2026).

8. ID Quantique. Quantis QRNG PCIe – High performance PCIe Quantum Random Number Generator. URL: <https://www.idquantique.com/random-number-generation/products/quantis-qrng-pcie/> (дата звернення: 21.02.2026).

9. TectroLabs. SwiftRNG Pro – Hardware Random Number Generator, V2.2 – March 2024, Rev. 2.6 URL: <https://tectrolabs.com/swiftrng-pro/> (дата звернення: 21.02.2026).

10. NIST. Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised) : NIST Special Publication 800-90A. – Gaithersburg, MD : National Institute of Standards and Technology. 2015. URL: <https://csrc.nist.gov/pubs/sp/800/90/a/final> (дата звернення: 21.02.2026).

11. NIST. Recommendation for the Entropy Sources Used for Random Bit Generation : NIST Special Publication 800-90B. – Gaithersburg, MD : National Institute of Standards and Technology. 2018. URL: <https://csrc.nist.gov/pubs/sp/800/90/b/final> (дата звернення: 21.02.2026).

12. NIST. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications : NIST Special Publication 800-22 (rev. 1a). – Gaithersburg, MD : National Institute of Standards and Technology. 2010. URL: <https://dl.acm.org/doi/10.5555/2206233> (дата звернення: 21.02.2026).

13. Прокопишин І. А., Рикалюк Р. Є., Чекурін В. Ф., Червінк К. А. *Основи теорії інформації та кодування : навч. посібник*. Львів : ЛНУ ім. Івана Франка, 2023. 156 с.

14. Ubld.it. TrueRNGpro – USB Hardware Random Number Generator. URL: <https://ubld.it/products/truerngpro> (дата звернення: 21.02.2026).

15. Altus Metrum. ChaosKey – Hardware True Random Number Generator. URL: <https://altusmetrum.org/ChaosKey/> (дата звернення: 21.02.2026).

16. Moonbase Otago. OneRNG – Open Hardware Random Number Generator : URL: <https://onerng.info/> (дата звернення: 21.02.2026).

17. Comscire. QNG Model PQ4000KS – Data Sheet. URL: https://comscire.com/files/datasheet/PQ4000KS_datasheet.pdf (дата звернення: 21.02.2026).

18. Free Software Foundation. Now available from GNU Press, the NeuG True Random Number Generator. URL: <https://www.fsf.org/blogs/gnu-press/neug-trng> (дата звернення: 21.02.2026).

19. Microchip Technology. ATECC608B CryptoAuthentication Device – Summary Datasheet, DS40002239. URL: <https://ww1.microchip.com/downloads/aemDocuments/documents/SCBU/ProductDocuments/DataSheets/ATECC608B-CryptoAuthentication-Device-Summary-Data-Sheet-DS40002239B.pdf> (дата звернення: 21.02.2026).

20. Thales / Entrust. nShield HSM Family – General Purpose Hardware Security Modules. URL: https://dymarjaya.co.id/wp-content/uploads/2018/02/ThalesEsecurity_nShield_HSM_Family_br-ISO27001.pdf (дата звернення: 21.02.2026).

21. Atoponce A. Awesome Hardware Random Number Generators (curated table). URL: <https://github.com/atoponce/awesome-hwrng> (дата звернення: 21.02.2026).

22. NIST. Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised) : NIST Special Publication 800-90A. – Gaithersburg, MD : National Institute of Standards and Technology, 2015. URL: <https://csrc.nist.gov/pubs/sp/800/90/b/final> (дата звернення: 21.02.2026).

23. NIST. Recommendation for the Entropy Sources Used for Random Bit Generation : NIST Special Publication 800-90B. – Gaithersburg, MD : National Institute of Standards and Technology, 2018. URL: <https://csrc.nist.gov/pubs/sp/800/90/b/final> (дата звернення: 21.02.2026).

24. NIST. A Statistical Test Suite for Random and Pseudorandom Number

Generators for Cryptographic Applications : NIST Special Publication 800-22 (rev. 1a). – Gaithersburg, MD : National Institute of Standards and Technology, 2010. URL: <https://dl.acm.org/doi/10.5555/2206233> (дата звернення: 21.02.2026).

25. Packard K. ChaosKey v1.0 Released — USB Attached True Random Number Generator. URL: <https://keithp.com/blogs/chaoskey/> (дата звернення: 21.02.2026).

26. Linux Kernel Documentation. USB ChaosKey random number generator driver support. URL: https://cateee.net/lkddb/web-lkddb/USB_CHAOSKEY.html. (дата звернення: 21.02.2026).

27. ChaosKey: This is a hardware True Random Number Generator that attaches via USB. Information Security Newspaper. 2016. URL: <https://www.securitynewspaper.com/2016/04/26/chaoskey-hardware-true-random-number-generator-attaches-via-usb/> (дата звернення: 21.02.2026).

28. OneRNG — Hardware Random Number Generator. URL: <https://onerng.info/> (дата звернення: 21.02.2026).

29. OneRNG — What is OneRNG? URL: <https://onerng.info/whatisonerng.html>. (дата звернення: 21.02.2026).

30. Moonbase Otago. OneRNG Project Documentation. URL: <https://moonbaseotago.com/onerng/> (дата звернення: 21.02.2026).

31. OneRNG Hardware Random Number Generator : USB Devices Wiki. 2020. URL: <https://wiki.chucknemeth.com/usb-devices/onerng> (дата звернення: 21.02.2026).

32. Free Software Foundation. Now available from GNU Press, the NeuG True Random Number Generator. 2015. URL: <https://www.fsf.org/blogs/gnu-press/neug-trng/> (дата звернення: 21.02.2026).

33. Niibe Y. NeuG, a True Random Number Generator Implementation. 2015. URL: <https://www.gniibe.org/memo/development/gnuk/rng/neug.html> (дата звернення: 21.02.2026).

34. Exploring true random number generators built on embedded devices : MSc Thesis / Malmö University. 2023. URL: <https://mau.diva-portal.org/smash/get/diva2:1803408/FULLTEXT02.pdf> (дата звернення: 21.02.2026).
35. Comscire. Random Number Generator Selection Guide. URL: <https://comscire.com/random-number-generator-selection-guide/> (дата звернення: 21.02.2026).
36. Comscire. QNG Model PQ4000KS – Validation Tests of Randomness. URL: <https://comscire.com/files/cert/comscire-pq4000ks-qngmeter-validation-tests.pdf> (дата звернення: 21.02.2026).
37. Comscire. QNG Model PQ4000KS – Certification Letter. 2017. URL: <https://comscire.com/files/cert/pq4000ks-certification-letter.pdf> (дата звернення: 21.02.2026).
38. Quantum generators of random numbers / ResearchGate preprint. URL: https://www.researchgate.net/publication/353775354_Quantum_generators_of_random_numbers (дата звернення: 21.02.2026).
39. Atoponce A. Awesome Hardware Random Number Generators. URL: <https://github.com/atoponce/awesome-hwrng> (дата звернення: 21.02.2026).
40. A Comprehensive Review of Quantum Random Number Generators. arXiv, 2022. URL: <https://arxiv.org/html/2203.00261v3> (дата звернення: 21.02.2026).
41. ISO/IEC 18031:2025. Information technology – Security techniques – Random bit generation. Geneva : ISO/IEC. 2025. URL: <https://www.iso.org/standard/81645.html> (дата звернення: 21.02.2026).
42. Random number generation : ArchWiki. 2025. URL: https://wiki.archlinux.org/title/Random_number_generation (дата звернення: 21.02.2026).

43. FIPS 140-2. Security Requirements for Cryptographic Modules. NIST. 2001. URL: <https://csrc.nist.gov/pubs/fips/140-2/upd2/final> (дата звернення: 21.02.2026).

44. Mendel F.: "Analysis of Cryptographic Hash Functions". URL: <https://diglib.tugraz.at/download.php?id=576a7a85b2dc5&location=browse> (дата звернення: 21.02.2026).

45. Дячук Р.Л., Павлюченко О.С., Прохоров П.А., Прохоров Г.В., Добровольський Ю.Г. Огляд статистичних методів оцінки якості криптографічних хеш-функцій. Проблеми інформатики та комп'ютерної техніки : праці XI міжнар. наук.-практ. конф. (ПІКТ – 2022), м. Чернівці, 10–13 лист. 2022. Черн. нац. ун-т ім. Ю Федьковича, Чернівці, 2022. – С. 51-53. <https://drive.google.com/drive/folders/1wE3cqbrt1ZluNBCO5jvT---aaNvyJsmX>].

46. Stevens M. Attacks on Hash Functions and Applications. URL: <https://www.cwi.nl/system/files/PhD-Thesis-Marc-Stevens-Attacks-on-Hash-Functions-and-Applications.pdf> (дата звернення: 21.02.2026).

47. Martinez F. Attacks on Pseudo Random Number Generators Hiding a Linear Structure. Cryptographers' Track. RSA Conference : Virtual Event, United States. 2022, Mar 2022. P.145-168 URL: <https://link.springer.com/book/10.1007/978-3-030-95312-6> (дата звернення: 21.02.2026).

48. Class SecureRandom. All Implemented Interfaces. URL: <https://docs.oracle.com/javase/8/docs/api/java/security/SecureRandom.html> (дата звернення: 21.02.2026).

49. Cornejo M., Ruhault S., (In) Security of Java SecureRandom Implementations. *Journées Codage et Cryptographie*. 2014. URL: <https://github.com/openjdk/jdk/blob/master/src/java.base/share/classes/java/security/SecureRandom.java> (дата звернення: 21.02.2026).

50. Остапов С.Е., Добровольський Ю.Г. Квантова інформатика та квантові обчислення. -Чернівці: ЧНУ, 2021. - 99 с.

51. Jacak J.E., Jacak W.A., Donderowicz W.A. Quantum random number generators with entanglement for public randomness testing. *Sci Rep.* 2020. № 10. P. 164. URL: <https://doi.org/10.1038/s41598-019-56706-2> (дата звернення: 21.02.2026).

52. Park S., Choi D., Kang T., Park K., Kwon Y., Kim J., Efficient hardware implementation and analysis of true random-number generator based on beta source. *ETRI.* 2020. V. 42. № 4. P. 518-526. URL: <https://doi.org/10.4218/etrij.2020-0083>. (дата звернення: 21.02.2026).

53. Barannik V., Sidchenko S., Barannik N., Khimenko A. The method of masking overhead compaction in video compression systems. *Radioelectronic and Computer Systems.* 2021. № 2. P. 51-63. URL: [doi:https://doi.org/10.32620/reks.2021.2.05](https://doi.org/10.32620/reks.2021.2.05) (дата звернення: 21.02.2026).

54. Kaźmierczyk A., Chojnacki A., Banasik K. Pseudorandom number generators as applied in reliability analysis. 2022. Kielce University of Technology, Faculty of Electrical Engineering, Automatic Control and Computer Science, Department of Power Engineering, Power Electronics and Electrical Machines. URL: [doi:10.15199/48.2022.12.44](https://doi.org/10.15199/48.2022.12.44). (дата звернення: 21.02.2026).

55. Barannik V., Barannik N., Ignatiev O., Khimenko V. Method of indirect hiding of information in the process of compressing video images. *Radioelectronic and Computer Systems.* 2021. № 4. P. 119-131. [doi:https://doi.org/10.32620/reks.2021.4.10](https://doi.org/10.32620/reks.2021.4.10).

56. Ostapov S., Diakonenko B., Fylypiuk M., Hazdiuk K., Shumylyak L., Tarnovetska O. Symmetrical Cryptosystems based on Cellular Automata. *International Journal of Computing.* 2023. № 2(1). P. 15-20. <https://doi.org/10.47839/ijc.22.1.2874>.

57. Diachuk R., Dobrovolsky Y., Hanzhelo D., Prokhorov H., Trembach D. Research the Level of Chaotic and Reliability in Webcam-generated Random Number Sequences. *Security of infocommunication systems and Internet of things*. 2024. V. 2. № 1. P. 01004. doi: 10.31861/sisiot2024.1.01004.

58. Dobrovolsky Y. Prokhorov G. Primary processing of an optical image on autonomous mobile optical systems using cellular automata. Proceedings of SPIE. 12938, Sixteenth International Conference on Correlation Optics, 129380K (5 January 2024); <https://doi.org/10.1117/12.3009624>.

59. Soorat R., Ashok M. Vudayagiri Hardware Random number Generator for cryptography. *arXiv*. 2015. <https://doi.org/10.48550/arXiv.1510.01234>.

60. Soorat R. Kandukuri A. Vudayagiri Hardware Random number Generator for cryptography. *Nanosystems Physics Chemistry Mathematics*. 2015. № 3. P. 121-127. DOI: 10.17586/2220-8054-2017-8-5-600-605.

61. Rogalski A. Infrared and Terahertz Detectors, Third Edition Published June 13, 2022 by CRC Press, 1066 Pages. URL: <https://www.routledge.com/Infrared-and-Terahertz-Detectors-Third-Edition/Rogalski/p/book/9781032338668#top> (дата звернення: 21.02.2026).

62. Dobrovolsky Yu., Lipka V., Strebezhev V., Sorokatyi Yu., Sorokatyi M., Andreeva O. Photodiode on the basis of epitaxial phosphate gallium with increased sensitivity at a wavelength of 254 nm. *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska*. 2020. № 1. P.36-39. URL: <https://e-iapgos.pl/resources/html/articlesList?issueId=12149> (дата звернення: 21.02.2026).

63. Webcam-capture Resolution. GitHub. URL: <https://github.com/sarxos/webcam-capture/blob/master/webcam-capture/src/main/java/com/github/sarxos/webcam/WebcamResolution.java> (дата звернення: 21.02.2026).

64. Afflerbach L., Criteria for the assessment of random number generators. *Journal of Computational and Applied Mathematics*. 1990. V. 31. №. 1. P. 3–10. doi: 10.1016/0377-0427(90)90330-3.
65. Li R., A true random number generator algorithm from digital camera image noise for varying lighting conditions. Southeast Con. 2015. Fort Lauderdale, FL, USA, 2015, P. 1–8. URL: <https://ieeexplore.ieee.org/document/7132901> (дата звернення: 21.02.2026).
66. Diachuk R., Dmytrashchuk K., Mazurets A., Prokhorov H., Yanushevskiy S. Photosensitive matrix as a source of entropy. *Proceedings of SPIE*. 2025. 13813, Seventeenth International Conference on Correlation Optics. 2025. 138132C. URL: <https://doi.org/10.1117/12.3092587> (дата звернення: 21.02.2026).
67. Sze S., Ngk K. Physics of Semiconductor Devices. 3rd Edition. John Wiley & Sons Inc., New Jersey 2006. 315p.
68. Dobrovolsky Yu., Andreeva O., Gavriyak M., Pidkamin L., Prokhorov G. p-i-n Photodiode Based on Silicon with Short Rise Time. *Journal of nano- and electronic physics*. 2018. V. 10. No 4. P. 04019-04024. [http://dx.doi.org/10.21272/jnep.10\(4\).04019](http://dx.doi.org/10.21272/jnep.10(4).04019)
69. Добровольський Ю., Дячук Р. Дослідження зворотного струму фотодіода для генерації надійної випадкової послідовності чисел. *Вісник Хмельницького національного університету. Серія: Технічні науки*. 2025. Т. 357. № 5.1. С. 126-132. <https://doi.org/10.31891/2307-5732-2025-357-15>.
70. Прохоров Г.В., Прохоров В.Г., Добровольський Ю.Г. Аналогово-частотний перетворювач для введення даних в комп'ютер. *Науковий вісник Чернівецького університету. Комп'ютерні системи та компоненти*. 2011. Т. 2. Вип. 1. С.101-105.
71. JDK 17 Documentation. Java Components page URL: <https://docs.oracle.com/en/java/javase/17/> (дата звернення: 23.02.2026).

72. Class DescriptiveStatistics. URL: <https://commons.apache.org/proper/commons-math/javadocs/api-3.6.1/org/apache/commons/math3/stat/descriptive/DescriptiveStatistics.html> (дата звернення: 23.02.2026).

73. JFreeChart 1.5.0 API Documentation. URL: <https://www.jfree.org/jfreechart/api.html> (дата звернення: 23.02.2026).

74. Package org.jfree.data.time. URL: <https://www.jfree.org/jfreechart/api/javadoc/org/jfree/data/time/package-summary.html> (дата звернення: 23.02.2026).

75. Добровольській Ю., Дячук Р. Дослідження генерації випадкової послідовності чисел на основі темнового струм фотодіода при різних температурах. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки*. 2025. Том 36 (75). № 4. Частина 2. С. 119-126. <https://doi.org/10.32782/2663-5941/2025.4.2/16>.

76. javax.usb API. URL: https://javax-usb.sourceforge.net/jdoc/?utm_source=chatgpt.com (дата звернення: 23.02.2026).

77. Веригіна І. В., Островська О. В. Теорія ймовірностей та математична статистика: Частина 2. Випадкові величини: Лекції і практикум. Київ, КПІ ім. Ігоря Сікорського, 2021. 77 с. URL: <https://ela.kpi.ua/server/api/core/bitstreams/4c25bed3-0ae6-4677-b2d2-aa3d047c2910/content> (дата звернення: 23.02.2026).

78. Prokhorov G.V., Diachuk R.L., Hanzhelo M.G, Yanushevskyi S.V. New approaches to the design of a hybrid random sequence generator. Physical and technological problems of transmission, processing and storage of information in infocommunication systems: Proceedings of Xth International Scientific-Practical Conference. Chernivtsi : Yuriy Fedkovych Chernivtsi National University, 2025. 202 p. P. 134-135.

79. Технічні вимоги до генерації випадкових значень для криптографії та правила застосування. URL: <https://datatracker.ietf.org/doc/html/rfc4086> (дата звернення: 23.02.2026).

80. Технічні вимоги для QRNG, включно з forward/backward secrecy. URL: https://www.tec.gov.in/pdf/GRs/Standard_GR_QRNG_TEC_91020_2024_Final%20%281%29.pdf (дата звернення: 23.02.2026).

81. Рекомендації для DRBG (крипто-стійких генераторів) з вимогами до безпеки та архітектури URL: https://ua.wikipedia.org/wiki/NIST_SP_800-90A (дата звернення: 23.02.2026).

82. Post-Processing Functions for a Biased Physical RNG URL: https://www.researchgate.net/publication/220942600_Post-Processing_Functions_for_a_Biased_Physical_Random_Number_Generator (дата звернення: 23.02.2026).

83. High throughput true random number generator based on dynamically superimposed hybrid entropy sources. URL: <https://www.sciencedirect.com/science/article/abs/pii/S0167926025000379>. (дата звернення: 23.02.2026).

84. A Proposal for Functionality Classes for Random Number Generators. URL: https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Certification/Interpretations/AIS_31_Functionality_classes_for_random_number_generators_e.pdf (дата звернення: 23.02.2026).

85. Технічні вимоги до генерації випадкових значень для криптографії та правила застосування. URL: <https://datatracker.ietf.org/doc/html/rfc4086> (дата звернення: 23.02.2026).

86. JavaFX. Documentation. URL: <https://openjfx.io/> (дата звернення: 23.02.2026).

87. Prokhorov H., Hanzhelo M., Diachuk R., Yanushevskiy S. Investigation of statistical characteristics of random number sequences generated by a webcam using cellular automata functionality and NIST patterns. *Security of infocommunication*

systems and Internet of things. 2025. Vol 3, № 1. P. 01007. DOI: 10.31861/sisiot2025.1.01007

88. Добровольський Ю., Дячук Р. Дослідження генерації випадкової послідовності чисел на основі темнового струм фотодіода при різних температурах. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки*. 2025. Том 36 (75), № 4. Частина 2. С. 119-126. DOI: 10.32782/2663-5941/2025.4.2/16

89. Добровольський Ю.Г., Прохоров Г.В., Дячук Р.Л. Генерація послідовності випадкових чисел на основі темнового струму фотодіода. Current trends in scientific research development. Proceedings of the 13th International scientific and practical conference. BoScience Publisher. Boston, USA. 2025. P. 65 URL:<https://sci-conf.com.ua/xiii-mizhnarodna-naukovo-praktichna-konferentsiya-current-trends-in-scientific-research-development-1-3-08-2025-boston-ssha-arhiv/> (дата звернення: 23.02.2026).

90. Фотодіод ФД-288. Технічні характеристики. URL:<https://zapadpribor.com/fd-288/> (дата звернення: 23.02.2026).

91. Термостаті типу HB120-S. Технічні характеристики. URL:<https://spectrolab.com.ua/ua/p664984190-termostat-dlya-probirok.html> (дата звернення: 23.02.2026).

92. Черненко Р. «Генерація псевдовипадкових послідовностей на мікроконтролерах з обмеженими обчислювальними ресурсами, джерела ентропії та тестування статистичних властивостей». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»*. 2023. Том. 2 Вип. 22. с. 191-203 doi:10.28925/2663-4023.2023.22.191203.

93. Prokhorov H., Trembach D. Research Into the Efficiency of Processing a Numerical Random Sequence by Chaotic-type Cellular Automata. *Security of infocommunication systems and Internet of things*. 2024. V. 2. № 2. P. 02010. doi: 10.31861/sisiot2024.2.02010.

94. Gallou E., Fouseki K. Applying social impact assessment (SIA) principles in assessing contribution of cultural heritage to social sustainability in rural landscapes. *Journal of Cultural Heritage Management and Sustainable Development*. 2019. V.9. № 3. P. 352-375. URL: <https://www.sciencedirect.com/org/science/article/abs/pii/S2044126619000051> (дата звернення: 23.02.2026).

95. Коваленко В. В., Букасов М. М. Методи та моделі складання розкладів для оркестратора kubernetes. *Вісник Вінницького політехнічного інституту*. 2024. №4. С. 86–94. <https://doi.org/10.31649/1997-9266-2024-175-4-86-94>.

96. Косенко М. А. Використання Docker для спрощення розробки та деплою застосунків. Радіoeлектроніка та молодь у ХХІ столітті : матеріали 29-го Міжнар. молодіж. форуму, 16–19 квітня 2025 р. – Харків : ХНУРЕ, 2025. – Т. 3. – С. 515–516. URI: <https://openarchive.nure.ua/handle/document/30678> (дата звернення: 23.02.2026).

97. Гресь О., Галюк С., Круліковський О., Заяц Р., Ластівка Г., Шпатар П. Розроблення генераторів псевдовипадкових послідовностей на основі хеш-функцій з використанням багатовимірних хаотичних систем. *Measuring and computing devices in technological processes* 2024. № 4. С. 362–371.

98. Кіх М., Нємкова Е. Методологія покращення процесу генерування псевдовипадкових послідовностей. *Вісник Львівського державного університету безпеки життєдіяльності*. 2024. № 30. С. 123-133.

99. Muhammed A. J., Woldiegiworgies T. A., Tsegaye G. G. Security Enhancement of Playfair Cipher Using Modified Blum Blum Shub Algorithm and Keystream Values. URL:<https://doi.org/10.21203/rs.3.rs-3847682/v1> (дата звернення: 23.02.2026).

100. Dobrovolsky Y., Prokhorov G., Hanzhelo M., Hanzhelo D., Trembach D. Development of a hash algorithm based on cellular automata and chaos theory. *Eastern-European Journal of Enterprise Technologies*. 2021. V. 5(9). № 113. P. 48–55. doi:10.15587/1729-4061.2021.242849.

101. Vergili I., Yucel M.D. Avalanche and Bit Independence Properties for the Ensembles of Randomly Chosen $n \times n$ S-Boxes. *Turk J Elec Engin.* 2001. V.9. P. 137-145. URL: <https://journals.tubitak.gov.tr/elektrik/issues/elk-01-9-2/elk-9-2-3-0008-1.pdf>. (дата звернення: 23.02.2026).

102. Добровольський Ю.Г., Прохоров Г.В., Дячук Р.Л. Алгоритм ХЕШ-функції на основі клітинних автоматів та теорії хаосу. Innovation and development in world science. Proceedings of the 3rd International scientific and practical conference. MDPC Publishing. Zurich, Switzerland. 2025. Pp. 153-157. URL: <https://sci-conf.com.ua/iii-mizhnarodna-naukovo-praktichna-konferentsiya-innovation-and-development-in-world-science-29-31-12-2025-tsyurih-shvejtsariya-arhiv/>.

103. Prokhorov G., Dobrovolsky Y., Dyachuk R. Hash-Function Algorithms Balanced on Reliability and Data Processing Speed. 2022 IEEE 4th International Conference on Advanced Trends in Information Theory, ATIT 2022, 15-16 December, 2022, Kyiv, UKRAINE. pp.111-114. DOI: 10.1109/ATIT58178.2022.10024181

104. Akbar A., Zulkifl Khalid M., Fuzzy-Fortuna M. A fuzzified approach to generation of cryptographically secure pseudo-random numbers. 2008 IEEE International Multitopic Conference, P. 213–217. URL: doi.org/10.1109/INMIC.2008.4777738 (дата звернення: 23.02.2026).

105. Crocetti L., Nannipieri P., Di Matteo S., Fanucci L., Saponara S. Review of Methodologies and Metrics for Assessing the Quality of Random Number Generators. *Electronics*. 2023. № 12. V. 3. P. 723. URL: <https://doi.org/10.3390/electronics12030723> (дата звернення: 23.02.2026).

106. Bansal A., Kandikuppa A., Hasan M., Chen C.-Y., Bates A., Mohan S. System Auditing for Real-Time Systems. *ACM Transactions on Privacy and Security*. 2023. № 26. V 4. P. 1–37. URL: <https://doi.org/10.1145/3625229> (дата звернення: 23.02.2026).

107. Xiang W., Zhao J., Huang H., Zhang, X., Jiang Z. L., He D. Blockchain-Assisted Privacy-Preserving Public Auditing Scheme for Cloud Storage Systems. *Algorithms and Architectures for Parallel Processing*. 2024. V. 14488. P. 292–310.
108. Di P., Pareschi R., Salzano F., Zappone F. Listening to what the system tells us: Innovative auditing for distributed systems. *Frontiers in Computer Science*. 2023. № 4. P. 1020946.
109. Pushkar O., Hrabovskyi Y., Gordyeyev A. Development of a method for optimizing the site loading speed. *Eastern-European Journal of Enterprise Technologies*. 2020. № 6 (2). V. 108. P. 21–29.
110. Pikuza M. O., Mikhnevich S. Yu. Testing a hardware random number generator using NIST statistical test suite. *Doklady BGUIR*. 2021. № 19(4). P. 37–42.
111. Fridrich J. *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge : Cambridge University Press, 2009. 432 p.
112. Johnson N. F., Duric Z., Jajodia S. *Information Hiding: Steganography and Watermarking – Attacks and Countermeasures*. Boston : Kluwer Academic Publishers, 2001. – 545 p.
113. Eastlake D., Schiller J., Crocker S. Randomness Requirements for Security. RFC 4086. : IETF, 2005. 16 p.
114. Cover T. M., Thomas J. A. *Elements of Information Theory*. 2nd ed. New York : Wiley-Interscience, 2006. 748 p.

ДОДАТКИ

Додаток А.

Лістинг частини коду програми

```
=====
=====
FILE: ./pom.xml
=====
=====

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>ua.edu.chnu</groupId>
  <artifactId>impr-rand-gen</artifactId>
  <version>1.0-SNAPSHOT</version>
  <name>impr-rand-gen</name>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <junit.version>5.10.2</junit.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-controls</artifactId>
      <version>21</version>
    </dependency>
    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-fxml</artifactId>
      <version>21</version>
    </dependency>
    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-web</artifactId>
      <version>21</version>
    </dependency>
  </dependencies>
</project>
```

```

</dependency>
<dependency>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-swing</artifactId>
  <version>21</version>
</dependency>
<dependency>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-media</artifactId>
  <version>21</version>
</dependency>
<dependency>
  <groupId>org.controlsfx</groupId>
  <artifactId>controlsfx</artifactId>
  <version>11.2.1</version>
</dependency>
<dependency>
  <groupId>com.dlsc.formsfx</groupId>
  <artifactId>formsfx-core</artifactId>
  <version>11.6.0</version>
  <exclusions>
    <exclusion>
      <groupId>org.openjfx</groupId>
      <artifactId>*</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<dependency>
  <groupId>net.synedra</groupId>
  <artifactId>validatorfx</artifactId>
  <version>0.5.0</version>
  <exclusions>
    <exclusion>
      <groupId>org.openjfx</groupId>
      <artifactId>*</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<dependency>
  <groupId>org.kordamp.ikonli</groupId>
  <artifactId>ikonli-javafx</artifactId>
  <version>12.3.1</version>
</dependency>
<dependency>
  <groupId>org.kordamp.bootstrapfx</groupId>
  <artifactId>bootstrapfx-core</artifactId>
  <version>0.4.0</version>
</dependency>

```

```

<dependency>
  <groupId>eu.hansolo</groupId>
  <artifactId>tilesfx</artifactId>
  <version>21.0.3</version>
  <exclusions>
    <exclusion>
      <groupId>org.openjfx</groupId>
      <artifactId>*</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<dependency>
  <groupId>com.github.almasb</groupId>
  <artifactId>fxgl</artifactId>
  <version>17.3</version>
  <exclusions>
    <exclusion>
      <groupId>org.openjfx</groupId>
      <artifactId>*</artifactId>
    </exclusion>
  </exclusions>
</dependency>
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter-api</artifactId>
  <version>${junit.version}</version>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.junit.jupiter</groupId>
  <artifactId>junit-jupiter-engine</artifactId>
  <version>${junit.version}</version>
  <scope>test</scope>
</dependency>
</dependencies>

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.13.0</version>
      <configuration>
        <source>17</source>
        <target>17</target>
      </configuration>
    </plugin>
  </plugins>

```

```

    <groupId>org.openjfx</groupId>
    <artifactId>javafx-maven-plugin</artifactId>
    <version>0.0.8</version>
    <executions>
      <execution>
        <!-- Default configuration for running with: mvn clean javafx:run -->
        <id>default-cli</id>
        <configuration>

<mainClass>ua.edu.chnu.imprrandgen/ua.edu.chnu.imprrandgen.RandomGenApplication</mainClass>
        <launcher>app</launcher>
        <jlinkZipName>app</jlinkZipName>
        <jlinkImageName>app</jlinkImageName>
        <noManPages>true</noManPages>
        <stripDebug>true</stripDebug>
        <noHeaderFiles>true</noHeaderFiles>
      </configuration>
    </execution>
  </executions>
</plugin>
</plugins>
</build>
</project>

```

```

=====
=====
FILE: ./src/main/java/module-info.java
=====
=====

```

```

module ua.edu.chnu.imprrandgen {
  requires javafx.controls;
  requires javafx.fxml;
  requires javafx.web;

  requires org.controlsfx.controls;
  requires com.dlsc.formsfx;
  requires net.synedra.validatorfx;
  requires org.kordamp.ikonli.javafx;
  requires org.kordamp.bootstrapfx.core;
  requires eu.hansolo.tilesfx;
  requires com.almasb.fxgl.all;

  opens ua.edu.chnu.imprrandgen to javafx.fxml;
  opens ua.edu.chnu.imprrandgen.controller to javafx.fxml;
  exports ua.edu.chnu.imprrandgen;
  exports ua.edu.chnu.imprrandgen.controller;

```

```

    exports ua.edu.chnu.imprrandgen.model;
    exports ua.edu.chnu.imprrandgen.service;
}

```

```

=====
=====
FILE: ./src/main/java/ua.edu.chnu.imprrandgen/RandomGenApplication.java
=====
=====

```

```

package ua.edu.chnu.imprrandgen;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.TabPane;
import javafx.stage.Stage;
import ua.edu.chnu.imprrandgen.controller.*;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class RandomGenApplication extends Application {

    private static final Map<String, Object> controllerMap = new HashMap<>();

    @Override
    public void start(Stage stage) throws IOException {
        // Create a custom controller factory that stores controllers
        FXMLLoader.setDefaultClassLoader(getClass().getClassLoader());
        var mainLoader = new FXMLLoader(RandomGenApplication.class.getResource("main-view.fxml"));
        mainLoader.setControllerFactory(clazz -> {
            try {
                var controller = clazz.getDeclaredConstructor().newInstance();
                controllerMap.put(clazz.getSimpleName(), controller);
                return controller;
            } catch (Exception e) {
                throw new RuntimeException("Failed to create controller: " + clazz.getName(), e);
            }
        });

        var root = (javafx.scene.Parent) mainLoader.load();

        // Get the main controller
        var mainController = (ua.edu.chnu.imprrandgen.controller.MainController)
            controllerMap.get("MainController");
    }
}

```

```

if (mainController == null) {
    mainController = mainLoader.getController();
}

// Get child controllers from the map
var settingsController = (SettingsController) controllerMap.get("SettingsController");
var generationController = (GenerationController) controllerMap.get("GenerationController");
var analysisController = (AnalysisController) controllerMap.get("AnalysisController");
var sequencesController = (SequencesController) controllerMap.get("SequencesController");

// Wire controllers to main controller if they exist
if (mainController != null) {
    if (settingsController != null) {
        mainController.setSettingsController(settingsController);
        settingsController.initializeWithModels();
    }

    if (generationController != null) {
        mainController.setGenerationController(generationController);
        generationController.initializeWithModels();
    }

    if (analysisController != null) {
        mainController.setAnalysisController(analysisController);
        analysisController.initializeWithModels();
    }

    if (sequencesController != null) {
        mainController.setSequencesController(sequencesController);
        sequencesController.initializeWithModels();
    }

    // Initialize the main controller, which will initialize all sub-controllers
    mainController.refreshAllViews();
}

// Wrap the root in a ScrollPane to enable scrolling
var scrollPane = new ScrollPane();
scrollPane.setContent(root);
scrollPane.setFitToWidth(true);
scrollPane.setFitToHeight(false); // Allow vertical scrolling
scrollPane.setPannable(true);
scrollPane.setVbarPolicy(ScrollPane.ScrollBarPolicy.AS_NEEDED); // Show vertical scrollbar when needed
scrollPane.setHbarPolicy(ScrollPane.ScrollBarPolicy.AS_NEEDED); // Show horizontal scrollbar when needed

var scene = new Scene(scrollPane, 1000, 800);
stage.setTitle("Random Sequence Generator");

```

```

        stage.setScene(scene);

        // Set minimum size to ensure all UI elements are visible
        stage.setMinWidth(900);
        stage.setMinHeight(700);

        // Make the window resizable
        stage.setResizable(true);

        stage.show();
    }

    public static void main(String[] args) {
        launch();
    }
}

```

```

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/controller/AnalysisController.java
=====
=====

```

```

package ua.edu.chnu.imprrandgen.controller;

import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.fxml.FXML;
import javafx.scene.Group;
import javafx.scene.chart.BarChart;
import javafx.scene.chart.CategoryAxis;
import javafx.scene.chart.NumberAxis;
import javafx.scene.chart.XYChart;
import javafx.scene.control.*;
import javafx.scene.layout.Pane;
import javafx.stage.FileChooser;
import ua.edu.chnu.imprrandgen.model.GenerationResult;
import ua.edu.chnu.imprrandgen.model.RangeConfiguration;
import ua.edu.chnu.imprrandgen.model.RangeType;
import ua.edu.chnu.imprrandgen.model.SequenceAnalyzer;
import ua.edu.chnu.imprrandgen.service.FileIOService;

import java.io.File;
import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

```

```

/**
 * Controller for the analysis tab.
 */
public class AnalysisController extends BaseController implements AnalysisControllerInterface {

    @FXML private ComboBox<String> previousTriesCombo;

    // Test result labels and progress bars
    @FXML private Label freqTestResult;
    @FXML private ProgressBar freqTestBar;
    @FXML private Label freqTestPercentage;

    @FXML private Label runsTestResult;
    @FXML private ProgressBar runsTestBar;
    @FXML private Label runsTestPercentage;

    @FXML private Label serialTestResult;
    @FXML private ProgressBar serialTestBar;
    @FXML private Label serialTestPercentage;

    @FXML private Label entropyTestResult;
    @FXML private ProgressBar entropyTestBar;
    @FXML private Label entropyTestPercentage;

    @FXML private Label longestRunTestResult;
    @FXML private ProgressBar longestRunTestBar;
    @FXML private Label longestRunTestPercentage;

    @FXML private Label matrixRankTestResult;
    @FXML private ProgressBar matrixRankTestBar;
    @FXML private Label matrixRankTestPercentage;

    @FXML private Label dftTestResult;
    @FXML private ProgressBar dftTestBar;
    @FXML private Label dftTestPercentage;

    @FXML private Label cusumsTestResult;
    @FXML private ProgressBar cusumsTestBar;
    @FXML private Label cusumsTestPercentage;

    // Chaotic properties
    @FXML private Label lyapunovLabel;
    @FXML private Label correlationLabel;
    @FXML private Label recurrenceLabel;

    // Histogram
    @FXML private Pane histogramPane;

```

```

private BarChart<String, Number> histogramChart;

// Current data
private int[] sequenceData;
private RangeConfiguration rangeConfiguration;

// Services
private final FileIOService fileIOService = new FileIOService();

@Override
public void initialize(URL url, ResourceBundle resourceBundle) {
    // Initialize histogram chart
    initializeHistogramChart();
}

/**
 * Initializes the controller with the models from the main controller.
 */
public void initializeWithModels() {
    MainController mainController = getParentController(MainController.class);
    if (mainController != null) {
        this.rangeConfiguration = mainController.getRangeConfiguration();
        this.sequenceData = mainController.getSequenceData();

        // Initialize previous tries combo box
        updatePreviousTriesCombo();
    }
}

private void updatePreviousTriesCombo() {
    MainController mainController = getParentController(MainController.class);
    if (mainController != null) {
        previousTriesCombo.setItems(FXCollections.observableArrayList(
            mainController.getPreviousResults().stream().map(GenerationResult::toString).toList()
        ));
        if (!mainController.getPreviousResults().isEmpty()) {
            previousTriesCombo.getSelectionModel().selectFirst();
        }
    }
}

private void initializeHistogramChart() {
    // Create axis
    CategoryAxis xAxis = new CategoryAxis();
    NumberAxis yAxis = new NumberAxis();
    xAxis.setLabel("Value Range");
    yAxis.setLabel("Frequency");
}

```

```

// Create chart
histogramChart = new BarChart<>(xAxis, yAxis);
histogramChart.setTitle("Distribution Histogram");
histogramChart.setLegendVisible(false);
histogramChart.setAnimated(false);

// Add chart to pane
histogramPane.getChildren().add(histogramChart);
histogramChart.prefWidthProperty().bind(histogramPane.widthProperty());
histogramChart.prefHeightProperty().bind(histogramPane.heightProperty());
}

@Override
public void refreshView() {
    updatePreviousTriesCombo();
    updateHistogramChart();
}

@Override
public void runAnalysis() {
    // Get the latest sequence data from MainController
    MainController mainController = getParentController(MainController.class);
    if (mainController != null) {
        this.sequenceData = mainController.getSequenceData();
        this.rangeConfiguration = mainController.getRangeConfiguration();
    }

    // Check if we have sequence data
    if (sequenceData == null || sequenceData.length == 0) {
        return;
    }

    // Simulate running analysis tests
    new Thread() -> {
        try {
            // Simulate test execution with progress updates
            updateTestProgress(freqTestBar, freqTestResult, 0.2);
            Thread.sleep(200);

            updateTestProgress(runsTestBar, runsTestResult, 0.4);
            Thread.sleep(200);

            updateTestProgress(serialTestBar, serialTestResult, 0.6);
            Thread.sleep(200);

            updateTestProgress(entropyTestBar, entropyTestResult, 0.8);
            Thread.sleep(200);
        }
    };
}

```

```

// Calculate actual test scores
int minVal = rangeConfiguration.getMinValue();
int maxVal = rangeConfiguration.getMaxValue();

double freqScore = SequenceAnalyzer.calculateFrequencyScore(sequenceData, minVal, maxVal);
double runsScore = SequenceAnalyzer.calculateRunsScore(sequenceData);
double entropyScore = SequenceAnalyzer.calculateEntropy(sequenceData, minVal, maxVal);
double longestRunScore = SequenceAnalyzer.calculateLongestRunScore(sequenceData, minVal,
maxVal);
double cusumsScore = SequenceAnalyzer.calculateCusumsScore(sequenceData, minVal, maxVal);

// Calculate simplified scores for other tests
double serialScore = (freqScore + runsScore) / 2;

// Calculate chaotic properties
double lyapunovExponent = 0.6 + (entropyScore * 0.3);
double correlationDim = 1.8 + (entropyScore * 0.2);

// Update UI on JavaFX thread
Platform.runLater(() -> {
    // Update test results
    updateTestResult(freqTestBar, freqTestResult, freqTestPercentage, freqScore);
    updateTestResult(runsTestBar, runsTestResult, runsTestPercentage, runsScore);
    updateTestResult(serialTestBar, serialTestResult, serialTestPercentage, serialScore);
    updateTestResult(entropyTestBar, entropyTestResult, entropyTestPercentage, entropyScore);
    updateTestResult(longestRunTestBar, longestRunTestResult, longestRunTestPercentage,
longestRunScore);
    updateTestResult(matrixRankTestBar, matrixRankTestResult, matrixRankTestPercentage,
matrixRankScore);
    updateTestResult(dftTestBar, dftTestResult, dftTestPercentage, dftScore);
    updateTestResult(cusumsTestBar, cusumsTestResult, cusumsTestPercentage, cusumsScore);

    // Update chaotic properties
    lyapunovLabel.setText(String.format("%.4f", lyapunovExponent));
    correlationLabel.setText(String.format("%.4f", correlationDim));
    recurrenceLabel.setText("Deterministic (RQA: " + String.format("%.2f", 0.7 + entropyScore * 0.1) + ")");

    // Update histogram
    updateHistogramChart();
});

} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
}
}).start();
}

private void updateTestProgress(ProgressBar bar, Label result, double progress) {

```

```

Platform.runLater() -> {
    bar.setProgress(progress);
    result.setText("Running...");

    // Update percentage label
    if (bar == freqTestBar) {
        freqTestPercentage.setText(String.format("%.0f%%", progress * 100));
    } else if (bar == runsTestBar) {
        runsTestPercentage.setText(String.format("%.0f%%", progress * 100));
    } else if (bar == serialTestBar) {
        serialTestPercentage.setText(String.format("%.0f%%", progress * 100));
    } else if (bar == entropyTestBar) {
        entropyTestPercentage.setText(String.format("%.0f%%", progress * 100));
    } else if (bar == longestRunTestBar) {
        longestRunTestPercentage.setText(String.format("%.0f%%", progress * 100));
    } else if (bar == matrixRankTestBar) {
        matrixRankTestPercentage.setText(String.format("%.0f%%", progress * 100));
    } else if (bar == dfiTestBar) {
        dfiTestPercentage.setText(String.format("%.0f%%", progress * 100));
    } else if (bar == cusumsTestBar) {
        cusumsTestPercentage.setText(String.format("%.0f%%", progress * 100));
    }
});
}

private void updateTestResult(ProgressBar bar, Label result, Label percentage, double score) {
    bar.setProgress(score);
    result.setText(score > 0.9 ? "PASS" : "FAIL");
    percentage.setText(String.format("%.0f%%", score * 100));
}

@Override
public void loadGenerationResult(GenerationResult result) {
    if (result != null) {
        // Update the current data
        sequenceData = result.getData();

        // Update the main controller with the loaded data
        MainController mainController = getParentController(MainController.class);
        if (mainController != null) {
            mainController.setSequenceData(sequenceData);
        }

        // Run analysis on the loaded data
        runAnalysis();
    }
}
}

```

```

@Override
public void loadGenerationResultByIndex(int index) {
    MainController mainController = getParentController(MainController.class);
    if (mainController != null && index >= 0 && index < mainController.getPreviousResults().size()) {
        loadGenerationResult(mainController.getPreviousResults().get(index));
    }
}

@Override
public void compareWithGenerationResult(GenerationResult result) {
    if (result != null && sequenceData != null && sequenceData.length > 0) {
        // Create a comparison dialog
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Comparison Results");
        alert.setHeaderText("Comparing current sequence with " + result.getTimestamp());

        // Build comparison text
        StringBuilder comparison = new StringBuilder();
        comparison.append("Current Sequence:\n");
        comparison.append("- Length: ").append(sequenceData.length).append("\n");

        double currentEntropy = SequenceAnalyzer.calculateEntropy(
            sequenceData,
            rangeConfiguration.getMinValue(),
            rangeConfiguration.getMaxValue()
        );

        comparison.append("- Entropy: ").append(String.format("%.4f", currentEntropy)).append("\n");
        comparison.append("- Range Type: ").append(rangeConfiguration.getRangeType()).append("\n");

        MainController mainController = getParentController(MainController.class);
        if (mainController != null) {
            comparison.append("- Security Level: ");
            comparison.append(mainController.getSecuritySettings().getSecurityLevel()).append("\n\n");
        } else {
            comparison.append("- Security Level: Unknown\n\n");
        }

        comparison.append("Selected Previous Sequence:\n");
        comparison.append("- Length: ").append(result.getData().length).append("\n");
        comparison.append("- Entropy: ").append(String.format("%.4f", result.getEntropy())).append("\n");
        comparison.append("- Range Type: ").append(result.getRangeType()).append("\n");
        comparison.append("- Security Level: ").append(result.getSecurityLevel()).append("\n");

        alert.setContentText(comparison.toString());
        alert.showAndWait();
    }
}

```

```

@Override
public void exportReport() {
    if (sequenceData == null || sequenceData.length == 0) {
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setTitle("No Data");
        alert.setHeaderText(null);
        alert.setContentText("No sequence data available to export");
        alert.showAndWait();
        return;
    }

    var filePath = fileIOService.selectSaveFile(
        histogramPane.getScene().getWindow(),
        "Export Analysis Report",
        new FileChooser.ExtensionFilter("HTML Files", "*.html"),
        new FileChooser.ExtensionFilter("PDF Files", "*.pdf")
    );

    if (filePath != null) {
        try {
            // Get the current generation result or create one
            MainController mainController = getParentController(MainController.class);
            GenerationResult result = null;

            if (mainController != null && !mainController.getPreviousResults().isEmpty()) {
                result = mainController.getPreviousResults().get(0);
            }

            // If no result exists, create a temporary one
            if (result == null) {
                double entropy = SequenceAnalyzer.calculateEntropy(
                    sequenceData,
                    rangeConfiguration.getMinValue(),
                    rangeConfiguration.getMaxValue()
                );

                String distribution = SequenceAnalyzer.calculateDistributionStats(sequenceData);
                result = new GenerationResult(
                    java.time.LocalDateTime.now().format(
                        java.time.format.DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")
                    ),
                    sequenceData,
                    rangeConfiguration.getRangeType(),
                    rangeConfiguration.getMinValue(),
                    rangeConfiguration.getMaxValue(),
                    entropy,
                    distribution,
                    mainController != null ? mainController.getSecuritySettings().getSecurityLevel() : 5
                );
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

    );
}

// Export based on file extension
var fileName = filePath.getFileName().toString().toLowerCase();
if (fileName.endsWith(".html")) {
    fileIOService.exportReportToHTML(result, filePath);
} else if (fileName.endsWith(".pdf")) {
    fileIOService.exportReportToPDF(result, filePath);
}

Alert alert = new Alert(Alert.AlertType.INFORMATION);
alert.setTitle("Export Successful");
alert.setHeaderText(null);
alert.setContentText("Analysis report exported to " + filePath.getFileName());
alert.showAndWait();
} catch (IOException e) {
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setTitle("Export Error");
    alert.setHeaderText(null);
    alert.setContentText("Error exporting report: " + e.getMessage());
    alert.showAndWait();
}
}
}
}

```

```

@Override
public void updateHistogramChart() {
    if (sequenceData == null || sequenceData.length == 0 || histogramChart == null) {
        return;
    }
}

```

```

XYChart.Series<String, Number> series = new XYChart.Series<>();

```

```

if (rangeConfiguration.getRangeType() == RangeType.BITS) {
    // For bits (0-1), just count occurrences of each value
    int zeros = 0;
    int ones = 0;
    for (int value : sequenceData) {
        if (value == 0) zeros++;
        else if (value == 1) ones++;
    }
}

```

```

series.getData().add(new XYChart.Data<>("0", zeros));
series.getData().add(new XYChart.Data<>("1", ones));

```

```

    histogramChart.setTitle("Bit Distribution (0-1)");
} else {

```

```

// Count occurrences of each value
int minValue = rangeConfiguration.getMinValue();
int maxValue = rangeConfiguration.getMaxValue();
int[] valueCounts = new int[maxValue - minValue + 1];

for (int value : sequenceData) {
    if (value >= minValue && value <= maxValue) {
        valueCounts[value - minValue]++;
    }
}

// Create data series for each value that appears in the data
for (int i = 0; i < valueCounts.length; i++) {
    if (valueCounts[i] > 0) { // Only add values that actually appear
        int actualValue = minValue + i;
        series.getData().add(new XYChart.Data<>(String.valueOf(actualValue), valueCounts[i]));
    }
}

histogramChart.setTitle("Number Distribution (" + minValue + "-" + maxValue + ")");
}

// Update chart
histogramChart.getData().clear();
histogramChart.getData().add(series);

// Add value labels to the bars
for (XYChart.Data<String, Number> item : series.getData()) {
    Label valueLabel = new Label(item.getYValue().toString());
    valueLabel.setStyle("-fx-font-size: 10px; -fx-text-fill: black;");

    item.getNodeProperty().addListener((ov, oldNode, newNode) -> {
        if (newNode != null) {
            // Position the label above the bar
            valueLabel.setTranslateY(-valueLabel.getHeight() - 5);
            // Center the label on the bar
            valueLabel.setTranslateX(item.getNode().getBoundsInLocal().getWidth() / 2 - valueLabel.getWidth() / 2);

            // Add the label to the chart
            ((Group) item.getNode().getParent()).getChildren().add(valueLabel);
        }
    });
}

}

@Override
public void updateTestResults() {
    runAnalysis();
}

```

```

    }

    /**
     * Event handler for load selected button.
     */
    @FXML
    protected void onLoadPreviousTry() {
        int selectedIndex = previousTriesCombo.getSelectionModel().getSelectedIndex();
        loadGenerationResultByIndex(selectedIndex);
    }

    /**
     * Event handler for compare with current button.
     */
    @FXML
    protected void onComparePreviousTry() {
        MainController mainController = getParentController(MainController.class);
        if (mainController != null) {
            int selectedIndex = previousTriesCombo.getSelectionModel().getSelectedIndex();
            if (selectedIndex >= 0 && selectedIndex < mainController.getPreviousResults().size()) {
                compareWithGenerationResult(mainController.getPreviousResults().get(selectedIndex));
            }
        }
    }

    /**
     * Event handler for export report button.
     */
    @FXML
    protected void onExportReport() {
        exportReport();
    }
}

```

```

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/controller/AnalysisControllerInterface.java
=====
=====

```

```

package ua.edu.chnu.imprrandgen.controller;

import ua.edu.chnu.imprrandgen.model.GenerationResult;

/**
 * Interface for the analysis tab controller.
 */
public interface AnalysisControllerInterface extends Controller {

```

```

/**
 * Runs analysis on the current sequence data.
 */
void runAnalysis();

/**
 * Loads a previous generation result for analysis.
 *
 * @param result The generation result to load
 */
void loadGenerationResult(GenerationResult result);

/**
 * Loads a previous generation result by index.
 *
 * @param index The index of the result in the previous results list
 */
void loadGenerationResultByIndex(int index);

/**
 * Compares the current sequence with a previous generation result.
 *
 * @param result The generation result to compare with
 */
void compareWithGenerationResult(GenerationResult result);

/**
 * Exports the analysis report to a file.
 */
void exportReport();

/**
 * Updates the histogram chart with the current sequence data.
 */
void updateHistogramChart();

/**
 * Updates the test results display.
 */
void updateTestResults();
}

```

```

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/controller/BaseController.java
=====
=====

```

```

package ua.edu.chnu.imprrandgen.controller;

import java.net.URL;
import java.util.ResourceBundle;

/**
 * Abstract base controller class that implements the Controller interface.
 * Provides common functionality for all controllers.
 */
public abstract class BaseController implements Controller {

    /**
     * The parent controller that coordinates between all controllers.
     */
    protected Object parentController;

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        // Default implementation does nothing
    }

    @Override
    public void setParentController(Object parentController) {
        this.parentController = parentController;
    }

    /**
     * Gets the parent controller cast to the specified type.
     *
     * @param <T> The type to cast the parent controller to
     * @param type The class object of the type
     * @return The parent controller cast to the specified type, or null if the cast fails
     */
    @SuppressWarnings("unchecked")
    protected <T> T getParentController(Class<T> type) {
        if (parentController != null && type.isInstance(parentController)) {
            return (T) parentController;
        }
        return null;
    }
}

```

```

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/controller/Controller.java
=====
=====

```

```

package ua.edu.chnu.imprrandgen.controller;

import javafx.fxml.Initializable;

/**
 * Base interface for all controllers in the application.
 * Extends JavaFX's Initializable interface to ensure all controllers have an initialize method.
 */
public interface Controller extends Initializable {

    /**
     * Refreshes the controller's view with the latest data.
     * This method should be called when data that affects the view has changed.
     */
    void refreshView();

    /**
     * Sets the parent controller that coordinates between all tab controllers.
     * This allows for communication between controllers.
     *
     * @param parentController The parent controller instance
     */
    void setParentController(Object parentController);
}

```

```

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/controller/GenerationController.java
=====
=====

```

```

package ua.edu.chnu.imprrandgen.controller;

import javafx.application.Platform;
import javafx.fxml.FXML;
import javafx.scene.control.*;
import javafx.scene.input.Clipboard;
import javafx.scene.input.ClipboardContent;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.FileChooser;
import ua.edu.chnu.imprrandgen.model.*;
import ua.edu.chnu.imprrandgen.service.FileIOService;
import ua.edu.chnu.imprrandgen.service.SequenceGenerator;

import java.io.File;

```

```

import java.io.IOException;
import java.net.URL;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ResourceBundle;

/**
 * Controller for the generation tab.
 */
public class GenerationController extends BaseController implements GenerationControllerInterface {

    @FXML private Spinner<Integer> generatedSequenceLengthSpinner;
    @FXML private HBox standardGenerationControls;
    @FXML private VBox realtimeGenerationControls;
    @FXML private Button generateButton;
    @FXML private Button startButton;
    @FXML private Button resetButton;
    @FXML private Label valueCounterLabel;
    @FXML private ProgressIndicator generationProgress;
    @FXML private Label statusLabel;
    @FXML private Label configInfoLabel;
    @FXML private TextArea sequenceDisplay;
    @FXML private Label lengthLabel;
    @FXML private Label entropyLabel;
    @FXML private Label distributionLabel;
    @FXML private Button analysisButton;

    // Models
    private RangeConfiguration rangeConfiguration;
    private SecuritySettings securitySettings;
    private DataSourceSettings dataSourceSettings;

    // Generation state
    private volatile boolean isGenerating = false;
    private volatile boolean isPaused = false;
    private Thread generationThread;
    private int[] sequenceData;

    // Services
    private final FileIOService fileIOService = new FileIOService();
    private final SequenceGenerator sequenceGenerator = new SequenceGenerator();

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        // This will be called after all @FXML fields are injected
    }

}

```

```

* Initializes the controller with the models from the main controller.
*/
public void initializeWithModels() {
    MainController mainController = getParentController(MainController.class);
    if (mainController != null) {
        this.rangeConfiguration = mainController.getRangeConfiguration();
        this.securitySettings = mainController.getSecuritySettings();
        this.dataSourceSettings = mainController.getDataSourceSettings();

        // Initialize UI with model values
        updateUIFromModels();
    }
}

/**
* Updates the UI components with values from the models.
*/
private void updateUIFromModels() {
    // Update generation controls visibility based on selected source
    boolean isFileSource = dataSourceSettings.isFileSource();

    standardGenerationControls.setVisible(isFileSource);
    standardGenerationControls.setManaged(isFileSource);
    realtimeGenerationControls.setVisible(!isFileSource);
    realtimeGenerationControls.setManaged(!isFileSource);

    // Update configuration info
    updateConfigInfo();
}

@Override
public void refreshView() {
    updateUIFromModels();
}

@Override
public void generateSequence() {
    // Disable generate button during generation
    generateButton.setDisable(true);
    generationProgress.setVisible(true);
    statusLabel.setText("Generating sequence...");

    // Get security level
    int securityLevel = securitySettings.getSecurityLevel();

    // Run generation in background thread
    generationThread = new Thread() -> {
        try {

```

```

MainController mainController = getParentController(MainController.class);
if (mainController == null) return;

// Calculate and apply generation delay based on security level
long delay = sequenceGenerator.calculateGenerationDelay(securityLevel);
Thread.sleep(delay);

// Use the user-specified sequence length
int length = generatedSequenceLengthSpinner.getValue();

// Generate sequence using SequenceGenerator service
sequenceData = sequenceGenerator.generateSequence(
    length,
    rangeConfiguration,
    securitySettings
);

// Format sequence for display using SequenceGenerator service
String formattedSequence = sequenceGenerator.formatSequenceForDisplay(
    sequenceData,
    rangeConfiguration.getRangeType()
);

// Calculate entropy
double entropy = SequenceAnalyzer.calculateEntropy(
    sequenceData,
    rangeConfiguration.getMinValue(),
    rangeConfiguration.getMaxValue()
);

// Calculate distribution statistics
String distribution = SequenceAnalyzer.calculateDistributionStats(sequenceData);

// Update UI on JavaFX thread
final double finalEntropy = entropy;
final String finalDistribution = distribution;

Platform.runLater(() -> {
    // Update the main controller with the generated data
    MainController mainController2 = getParentController(MainController.class);
    if (mainController2 != null) {
        mainController2.setSequenceData(sequenceData);
    }

    sequenceDisplay.setText(formattedSequence);
    lengthLabel.setText(String.valueOf(length));
    entropyLabel.setText(String.format("%.4f", finalEntropy));
    distributionLabel.setText(finalDistribution);

```

```

        generateButton.setDisable(false);
        generationProgress.setVisible(false);
        statusLabel.setText("Sequence generated successfully");
        analysisButton.setDisable(false);
    });
} catch (InterruptedException e) {
    Platform.runLater() -> {
        statusLabel.setText("Generation interrupted");
        generateButton.setDisable(false);
        generationProgress.setVisible(false);
    });
}
});

generationThread.setDaemon(true);
generationThread.start();
}

@Override
public void startRealtimeGeneration() {
    // Update UI
    startButton.setDisable(true);
    resetButton.setDisable(false);
    statusLabel.setText("Generating sequence...");

    // Set flags
    isGenerating = true;
    isPaused = false;

    // Start generation in background thread
    if (generationThread == null || !generationThread.isAlive()) {
        generationThread = new Thread(this::runRealtimeGeneration);
        generationThread.setDaemon(true);
        generationThread.start();
    }
}

@Override
public void pauseRealtimeGeneration() {
    isPaused = true;
    startButton.setDisable(false);
    statusLabel.setText("Real-time generation paused");
}

@Override
public void resetGeneration() {
    // Stop generation if it's running

```

```

stopRealtimeGeneration();

// Clear data
sequenceData = new int[0];
sequenceDisplay.clear();
valueCounterLabel.setText("0");

// Reset UI
startButton.setDisable(false);
resetButton.setDisable(true);
statusLabel.setText("Ready to generate");

// Clear analysis data
lengthLabel.setText("0");
entropyLabel.setText("0.0");
distributionLabel.setText("N/A");

// Update the main controller with the cleared data
MainController mainController = getParentController(MainController.class);
if (mainController != null) {
    mainController.setSequenceData(new int[0]);
}
}

private void stopRealtimeGeneration() {
    isGenerating = false;
    isPaused = false;

    // Wait for thread to finish if it's running
    if (generationThread != null && generationThread.isAlive()) {
        try {
            generationThread.join(1000); // Wait up to 1 second
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}

private void runRealtimeGeneration() {
    // Get the target sequence length
    int targetLength = generatedSequenceLengthSpinner.getValue();

    MainController mainController = getParentController(MainController.class);
    if (mainController == null) return;

    // Initialize sequence data array
    sequenceData = new int[targetLength];

```

```

// Calculate delay per value based on security level
long delayPerValue = sequenceGenerator.calculateGenerationDelay(
    securitySettings.getSecurityLevel()
) / Math.max(1, targetLength / 10); // Divide total delay across values

// Generate values one by one with delays
for (int i = 0; i < targetLength && isGenerating; i++) {
    // Check if paused
    while (isPaused && isGenerating) {
        try {
            Thread.sleep(100);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
            return;
        }
    }

    // Stop if no longer generating
    if (!isGenerating) break;

    sequenceData[i] = value;

    // Create partial array for display
    final int[] partialSequence = new int[i + 1];
    System.arraycopy(sequenceData, 0, partialSequence, 0, i + 1);

    // Format partial sequence for display
    final String formattedSequence = sequenceGenerator.formatSequenceForDisplay(
        partialSequence,
        rangeConfiguration.getRangeType()
    );

    final int currentCount = i + 1;

    Platform.runLater(() -> {
        // Update sequence display
        sequenceDisplay.setText(formattedSequence);

        // Update counter
        valueCounterLabel.setText(String.valueOf(currentCount));

        // Update length label
        lengthLabel.setText(String.valueOf(currentCount));

        // Calculate and update entropy
        double entropy = SequenceAnalyzer.calculateEntropy(
            partialSequence,
            rangeConfiguration.getMinValue(),

```

```

        rangeConfiguration.getMaxValue()
    );
    entropyLabel.setText(String.format("%.4f", entropy));

    // Calculate and update distribution
    String distribution = SequenceAnalyzer.calculateDistributionStats(partialSequence);
    distributionLabel.setText(distribution);

    // Update the main controller with the generated data
    MainController mainController2 = getParentController(MainController.class);
    if (mainController2 != null) {
        mainController2.setSequenceData(partialSequence);
    }

    // Update status
    statusLabel.setText("Generating: " + currentCount + " / " + targetLength + " values");
});

// Apply delay between values
try {
    Thread.sleep(delayPerValue);
} catch (InterruptedException e) {
    Thread.currentThread().interrupt();
    return;
}
}

// Final update when generation is complete
if (isGenerating) {
    Platform.runLater(() -> {
        // Update the main controller with the final data
        MainController mainController2 = getParentController(MainController.class);
        if (mainController2 != null) {
            mainController2.setSequenceData(sequenceData);
        }

        // Update status and buttons
        statusLabel.setText("Generation complete: " + targetLength + " values generated");
        startButton.setDisable(false);
        analysisButton.setDisable(false);
    });

    // Set flag to indicate generation is complete
    isGenerating = false;
}
}

```

@Override

```

public void saveSequenceToFile() {
    if (sequenceData == null || sequenceData.length == 0) {
        statusLabel.setText("No sequence data to save");
        return;
    }

    var filePath = fileIOService.selectSaveFile(
        sequenceDisplay.getScene().getWindow(),
        "Save Sequence",
        new FileChooser.ExtensionFilter("Text Files", "*.txt")
    );

    if (filePath != null) {
        try {
            fileIOService.saveSequenceToFile(sequenceData, filePath, rangeConfiguration.getRangeType());
            statusLabel.setText("Sequence saved to " + filePath.getFileName());
        } catch (IOException e) {
            statusLabel.setText("Error saving file: " + e.getMessage());
        }
    }
}

@Override
public void copySequenceToClipboard() {
    if (sequenceData == null || sequenceData.length == 0) {
        statusLabel.setText("No sequence data to copy");
        return;
    }

    String sequenceText = sequenceDisplay.getText();
    if (sequenceText != null && !sequenceText.isEmpty()) {
        ClipboardContent content = new ClipboardContent();
        content.putString(sequenceText);
        Clipboard.getSystemClipboard().setContent(content);
        statusLabel.setText("Sequence copied to clipboard");
    }
}

@Override
public GenerationResult saveSequenceToHistory() {
    // Check if there's valid sequence data
    if (sequenceData == null || sequenceData.length == 0) {
        return null;
    }

    // Create a timestamp for this save
    String timestamp = LocalDateTime.now().format(
        DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")
    );
}

```

```

);

// Calculate entropy
double entropy = SequenceAnalyzer.calculateEntropy(
    sequenceData,
    rangeConfiguration.getMinValue(),
    rangeConfiguration.getMaxValue()
);

// Calculate distribution statistics
String distribution = SequenceAnalyzer.calculateDistributionStats(sequenceData);

// Create a new GenerationResult
GenerationResult result = new GenerationResult(
    timestamp, sequenceData, rangeConfiguration.getRangeType(),
    rangeConfiguration.getMinValue(), rangeConfiguration.getMaxValue(),
    entropy, distribution, securitySettings.getSecurityLevel()
);

// Add to previous results in the main controller
MainController mainController = getParentController(MainController.class);
if (mainController != null) {
    mainController.addGenerationResult(result);
}

// Show confirmation
statusLabel.setText("Sequence saved to history");

return result;
}

@Override
public int[] getSequenceData() {
    return sequenceData != null ? sequenceData.clone() : new int[0];
}

@Override
public boolean isGenerating() {
    return isGenerating;
}

@Override
public boolean isPaused() {
    return isPaused;
}

@Override
public void updateConfigInfo() {

```

```

    StringBuilder configText = new StringBuilder("Configuration: ");

    // Add range configuration
    configText.append(rangeConfiguration.getDescription());

    // Add security level
    configText.append(" | Security Level: ").append(securitySettings.getSecurityLevel());

    // Add data source
    configText.append(" | Source: ").append(dataSourceSettings.getDescription());

    configInfoLabel.setText(configText.toString());
}

/**
 * Event handler for generate button.
 */
@FXML
protected void onGenerateSequence() {
    generateSequence();
}

/**
 * Event handler for start button.
 */
@FXML
protected void onStartRealtimeGeneration() {
    startRealtimeGeneration();
}

/**
 * Event handler for pause button.
 */
@FXML
protected void onPauseRealtimeGeneration() {
    pauseRealtimeGeneration();
}

/**
 * Event handler for reset button.
 */
@FXML
protected void onResetRealtimeGeneration() {
    resetGeneration();
}

/**
 * Event handler for copy button.

```

```

    */
    @FXML
    protected void onCopySequence() {
        copySequenceToClipboard();
    }

    /**
     * Event handler for save to file button.
     */
    @FXML
    protected void onSaveSequence() {
        saveSequenceToFile();
    }

    /**
     * Event handler for save to sequences button.
     */
    @FXML
    protected void onSaveToSequences() {
        saveSequenceToHistory();
    }

    /**
     * Event handler for next to analysis button.
     */
    @FXML
    protected void onNextToAnalysis() {
        // First, save the sequence to history
        saveSequenceToHistory();

        // Switch to the analysis tab
        MainController mainController = getParentController(MainController.class);
        if (mainController != null) {
            mainController.switchToTab(2); // Analysis tab index

            // Get the analysis controller and run analysis
            AnalysisControllerInterface analysisController = mainController.getAnalysisController();
            if (analysisController != null) {
                analysisController.runAnalysis();
            }
        }
    }
}

```

```

=====
=====

```

FILE: ./src/main/java/ua/edu/chnu/imprrandgen/controller/GenerationControllerInterface.java

```

=====

package ua.edu.chnu.imprrandgen.controller;

import ua.edu.chnu.imprrandgen.model.GenerationResult;

/**
 * Interface for the generation tab controller.
 */
public interface GenerationControllerInterface extends Controller {

    /**
     * Generates a new random sequence based on the current settings.
     */
    void generateSequence();

    /**
     * Starts real-time generation of random numbers.
     */
    void startRealtimeGeneration();

    /**
     * Pauses real-time generation of random numbers.
     */
    void pauseRealtimeGeneration();

    /**
     * Resets the generation state.
     */
    void resetGeneration();

    /**
     * Saves the current sequence to a file.
     */
    void saveSequenceToFile();

    /**
     * Copies the current sequence to the clipboard.
     */
    void copySequenceToClipboard();

    /**
     * Saves the current sequence to the history.
     *
     * @return The generated result object
     */
    GenerationResult saveSequenceToHistory();
}

```

```

/**
 * @return The current sequence data
 */
int[] getSequenceData();

/**
 * @return Whether generation is currently in progress
 */
boolean isGenerating();

/**
 * @return Whether generation is currently paused
 */
boolean isPaused();

/**
 * Updates the configuration info display.
 */
void updateConfigInfo();
}

```

```

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/controller/MainController.java
=====
=====

```

```

package ua.edu.chnu.imprrandgen.controller;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.TabPane;
import ua.edu.chnu.imprrandgen.model.*;

import java.net.URL;
import java.util.ResourceBundle;

/**
 * Main controller that coordinates between all tab controllers and manages shared data.
 */
public class MainController implements Initializable {

    @FXML private TabPane mainTabPane;

    // Tab controllers

```

```

private SettingsControllerInterface settingsController;
private GenerationControllerInterface generationController;
private AnalysisControllerInterface analysisController;
private SequencesControllerInterface sequencesController;

// Shared models
private final RangeConfiguration rangeConfiguration;
private final SecuritySettings securitySettings;
private final DataSourceSettings dataSourceSettings;
private final ObservableList<GenerationResult> previousResults;
private final PhPhRandom phphRandom;

// Current sequence data
private int[] sequenceData;

/**
 * Creates a new MainController with default model values.
 */
public MainController() {
    this.rangeConfiguration = new RangeConfiguration();
    this.securitySettings = new SecuritySettings();
    this.dataSourceSettings = new DataSourceSettings();
    this.previousResults = FXCollections.observableArrayList();
    this.phphRandom = new PhPhRandom();
    this.sequenceData = new int[0];
}

@Override
public void initialize(URL url, ResourceBundle resourceBundle) {
    // This method will be called after all @FXML fields are injected
}

/**
 * Sets the settings controller.
 *
 * @param settingsController The settings controller
 */
public void setSettingsController(SettingsControllerInterface settingsController) {
    this.settingsController = settingsController;
    settingsController.setParentController(this);
}

/**
 * Sets the generation controller.
 *
 * @param generationController The generation controller
 */
public void setGenerationController(GenerationControllerInterface generationController) {

```

```

        this.generationController = generationController;
        generationController.setParentController(this);
    }

    /**
     * Sets the analysis controller.
     *
     * @param analysisController The analysis controller
     */
    public void setAnalysisController(AnalysisControllerInterface analysisController) {
        this.analysisController = analysisController;
        analysisController.setParentController(this);
    }

    /**
     * Sets the sequences controller.
     *
     * @param sequencesController The sequences controller
     */
    public void setSequencesController(SequencesControllerInterface sequencesController) {
        this.sequencesController = sequencesController;
        sequencesController.setParentController(this);
    }

    /**
     * Switches to the specified tab.
     *
     * @param tabIndex The index of the tab to switch to
     */
    public void switchToTab(int tabIndex) {
        if (mainTabPane != null) {
            mainTabPane.getSelectionModel().select(tabIndex);
        }
    }

    /**
     * @return The range configuration model
     */
    public RangeConfiguration getRangeConfiguration() {
        return rangeConfiguration;
    }

    /**
     * @return The security settings model
     */
    public SecuritySettings getSecuritySettings() {
        return securitySettings;
    }

```

```

/**
 * @return The data source settings model
 */
public DataSourceSettings getDataSourceSettings() {
    return dataSourceSettings;
}

/**
 * @return The list of previous generation results
 */
public ObservableList<GenerationResult> getPreviousResults() {
    return previousResults;
}

/**
 * @return The secure random generator
 */
public PhPhRandom getPhPhRandom() {
    return phphRandom;
}

/**
 * @return The settings controller
 */
public SettingsControllerInterface getSettingsController() {
    return settingsController;
}

/**
 * @return The generation controller
 */
public GenerationControllerInterface getGenerationController() {
    return generationController;
}

/**
 * @return The analysis controller
 */
public AnalysisControllerInterface getAnalysisController() {
    return analysisController;
}

/**
 * @return The sequences controller
 */
public SequencesControllerInterface getSequencesController() {
    return sequencesController;
}

```

```

}

/**
 * @return The current sequence data
 */
public int[] getSequenceData() {
    return sequenceData != null ? sequenceData.clone() : new int[0];
}

/**
 * Sets the current sequence data.
 *
 * @param sequenceData The new sequence data
 */
public void setSequenceData(int[] sequenceData) {
    this.sequenceData = sequenceData != null ? sequenceData.clone() : new int[0];
}

/**
 * Adds a new generation result to the list of previous results.
 *
 * @param result The generation result to add
 */
public void addGenerationResult(GenerationResult result) {
    previousResults.add(0, result); // Add at the beginning (most recent first)

    // Refresh views that display the results
    if (analysisController != null) {
        analysisController.refreshView();
    }
    if (sequencesController != null) {
        sequencesController.refreshView();
    }
}

/**
 * Removes a generation result from the list of previous results.
 *
 * @param result The generation result to remove
 */
public void removeGenerationResult(GenerationResult result) {
    previousResults.remove(result);

    // Refresh views that display the results
    if (analysisController != null) {
        analysisController.refreshView();
    }
    if (sequencesController != null) {

```

```

        sequencesController.refreshView();
    }
}

/**
 * Updates all views with the latest data.
 */
public void refreshAllViews() {
    if (settingsController != null) {
        settingsController.refreshView();
    }
    if (generationController != null) {
        generationController.refreshView();
    }
    if (analysisController != null) {
        analysisController.refreshView();
    }
    if (sequencesController != null) {
        sequencesController.refreshView();
    }
}
}

```

```

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/controller/SequencesController.java
=====
=====

```

```

package ua.edu.chnu.imprrandgen.controller;

import javafx.fxml.FXML;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import ua.edu.chnu.imprrandgen.model.GenerationResult;

import java.net.URL;
import java.util.ResourceBundle;

/**
 * Controller for the sequences tab.
 */
public class SequencesController extends BaseController implements SequencesControllerInterface {

    @FXML private TableView<GenerationResult> sequencesTableView;
    @FXML private TableColumn<GenerationResult, String> timestampColumn;

```

```

@FXML private TableColumn<GenerationResult, String> rangeTypeColumn;
@FXML private TableColumn<GenerationResult, Integer> securityLevelColumn;
@FXML private TableColumn<GenerationResult, Integer> lengthColumn;
@FXML private TableColumn<GenerationResult, Double> entropyColumn;

@FXML private Label detailsTimestampLabel;
@FXML private Label detailsRangeTypeLabel;
@FXML private Label detailsSecurityLevelLabel;
@FXML private Label detailsLengthLabel;
@FXML private Label detailsEntropyLabel;
@FXML private Label detailsDistributionLabel;

@Override
public void initialize(URL url, ResourceBundle resourceBundle) {
    // Initialize table view
    initializeSequencesTableView();
}

/**
 * Initializes the controller with the models from the main controller.
 */
public void initializeWithModels() {
    MainController mainController = getParentController(MainController.class);
    if (mainController != null) {
        // Set the items in the table
        sequencesTableView.setItems(mainController.getPreviousResults());
    }
}

private void initializeSequencesTableView() {
    // Set up cell value factories for the table columns
    timestampColumn.setCellValueFactory(cellData -> new
javafx.beans.property.SimpleStringProperty(cellData.getValue().getTimestamp()));
    rangeTypeColumn.setCellValueFactory(cellData -> {
        String rangeTypeStr = switch (cellData.getValue().getRangeType()) {
            case BITS -> "Bits";
            case DEFAULT -> "Default Range";
            case CUSTOM -> "Custom Range";
        };
        return new javafx.beans.property.SimpleStringProperty(rangeTypeStr);
    });
    securityLevelColumn.setCellValueFactory(cellData -> new
javafx.beans.property.SimpleIntegerProperty(cellData.getValue().getSecurityLevel()).asObject());
    lengthColumn.setCellValueFactory(cellData -> new
javafx.beans.property.SimpleIntegerProperty(cellData.getValue().getData().length).asObject());
    entropyColumn.setCellValueFactory(cellData -> new
javafx.beans.property.SimpleDoubleProperty(cellData.getValue().getEntropy()).asObject());
}

```

```

// Add listener for selection changes to update the details view
sequencesTableView.getSelectionModel().selectedItemProperty().addListener((obs, oldSelection, newSelection) ->
{
    if (newSelection != null) {
        updateSequenceDetails(newSelection);
    } else {
        clearSequenceDetails();
    }
});
}

```

@Override

```

public void refreshView() {
    // Refresh the table view
    sequencesTableView.refresh();
}

```

// Update details if a row is selected

```

GenerationResult selectedResult = sequencesTableView.getSelectionModel().getSelectedItem();
if (selectedResult != null) {
    updateSequenceDetails(selectedResult);
} else {
    clearSequenceDetails();
}
}

```

@Override

```

public void loadSelectedSequence(GenerationResult result) {
    if (result != null) {
        MainController mainController = getParentController(MainController.class);
        if (mainController != null) {
            // Update the main controller with the selected data
            mainController.setSequenceData(result.getData());

            // Switch to the generation tab
            mainController.switchToTab(1); // Generation tab index

            // Get the generation controller and update the display
            GenerationControllerInterface generationController = mainController.getGenerationController();
            if (generationController != null) {
                generationController.refreshView();
            }
        }
    }
}
}

```

@Override

```

public void viewSequenceAnalysis(GenerationResult result) {
    if (result != null) {

```

```

MainController mainController = getParentController(MainController.class);
if (mainController != null) {
    // Update the main controller with the selected data
    mainController.setSequenceData(result.getData());

    // Switch to the analysis tab
    mainController.switchToTab(2); // Analysis tab index

    // Get the analysis controller and run analysis
    AnalysisControllerInterface analysisController = mainController.getAnalysisController();
    if (analysisController != null) {
        analysisController.runAnalysis();
    }
}
}

@Override
public void deleteSelectedSequence(GenerationResult result) {
    if (result != null) {
        MainController mainController = getParentController(MainController.class);
        if (mainController != null) {
            // Remove the result from the main controller's list
            mainController.removeGenerationResult(result);

            // Clear the details view
            clearSequenceDetails();
        }
    }
}

@Override
public void updateSequenceDetails(GenerationResult result) {
    detailsTimestampLabel.setText(result.getTimestamp());

    String rangeTypeStr = switch (result.getRangeType()) {
        case BITS -> "Bits (0-1)";
        case DEFAULT -> "Default Range (1-256)";
        case CUSTOM -> "Custom Range (" + result.getMinValue() + "-" + result.getMaxValue() + ")";
    };

    detailsRangeTypeLabel.setText(rangeTypeStr);
    detailsSecurityLevelLabel.setText(String.valueOf(result.getSecurityLevel()));
    detailsLengthLabel.setText(String.valueOf(result.getData().length));
    detailsEntropyLabel.setText(String.format("%.4f", result.getEntropy()));
    detailsDistributionLabel.setText(result.getDistribution());
}

```

```

@Override
public void clearSequenceDetails() {
    detailsTimestampLabel.setText("");
    detailsRangeTypeLabel.setText("");
    detailsSecurityLevelLabel.setText("");
    detailsLengthLabel.setText("");
    detailsEntropyLabel.setText("");
    detailsDistributionLabel.setText("");
}

@Override
public GenerationResult getSelectedSequence() {
    return sequencesTableView.getSelectionModel().getSelectedItem();
}

/**
 * Event handler for load selected sequence button.
 */
@FXML
protected void onLoadSelectedSequence() {
    GenerationResult selectedResult = sequencesTableView.getSelectionModel().getSelectedItem();
    loadSelectedSequence(selectedResult);
}

/**
 * Event handler for view analysis button.
 */
@FXML
protected void onViewSequenceAnalysis() {
    GenerationResult selectedResult = sequencesTableView.getSelectionModel().getSelectedItem();
    viewSequenceAnalysis(selectedResult);
}

/**
 * Event handler for delete selected button.
 */
@FXML
protected void onDeleteSelectedSequence() {
    GenerationResult selectedResult = sequencesTableView.getSelectionModel().getSelectedItem();
    deleteSelectedSequence(selectedResult);
}
}

```

```

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/controller/SequencesControllerInterface.java
=====
=====

```

```

package ua.edu.chnu.imprrandgen.controller;

import ua.edu.chnu.imprrandgen.model.GenerationResult;

/**
 * Interface for the sequences tab controller.
 */
public interface SequencesControllerInterface extends Controller {

    /**
     * Loads a selected sequence for viewing or editing.
     *
     * @param result The generation result to load
     */
    void loadSelectedSequence(GenerationResult result);

    /**
     * Views the analysis of a selected sequence.
     *
     * @param result The generation result to analyze
     */
    void viewSequenceAnalysis(GenerationResult result);

    /**
     * Deletes a selected sequence from the history.
     *
     * @param result The generation result to delete
     */
    void deleteSelectedSequence(GenerationResult result);

    /**
     * Updates the sequence details display.
     *
     * @param result The generation result to display details for
     */
    void updateSequenceDetails(GenerationResult result);

    /**
     * Clears the sequence details display.
     */
    void clearSequenceDetails();

    /**
     * @return The currently selected generation result, or null if none is selected
     */
    GenerationResult getSelectedSequence();
}

```

```
=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/controller/SettingsController.java
=====
=====
```

```
package ua.edu.chnu.imprrandgen.controller;

import javafx.fxml.FXML;
import javafx.scene.control.*;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.FileChooser;
import ua.edu.chnu.imprrandgen.model.DataSourceSettings;
import ua.edu.chnu.imprrandgen.model.RangeConfiguration;
import ua.edu.chnu.imprrandgen.model.RangeType;
import ua.edu.chnu.imprrandgen.model.SecuritySettings;
import ua.edu.chnu.imprrandgen.service.FileIOService;
import ua.edu.chnu.imprrandgen.service.SerialPortManager;

import java.io.File;
import java.net.URL;
import java.util.ResourceBundle;

/**
 * Controller for the settings tab.
 */
public class SettingsController extends BaseController implements SettingsControllerInterface {

    // Range Configuration
    @FXML private RadioButton bitsRangeRadio;
    @FXML private RadioButton defaultRangeRadio;
    @FXML private RadioButton customRangeRadio;
    @FXML private HBox customRangeOptions;
    @FXML private TextField minValueField;
    @FXML private TextField maxValueField;

    // Security Settings
    @FXML private Slider securitySlider;
    @FXML private CheckBox lengthBasedSecurityCheck;
    @FXML private Spinner<Integer> sequenceLengthSpinner;
    @FXML private CheckBox roundsBasedSecurityCheck;
    @FXML private Spinner<Integer> roundsSpinner;
    @FXML private CheckBox timeBasedSecurityCheck;
    @FXML private Spinner<Integer> generationTimeSpinner;

    // Data Source
```

```

@FXML private RadioButton fileSourceRadio;
@FXML private RadioButton realtimeSourceRadio;
@FXML private VBox fileSourceOptions;
@FXML private VBox realtimeSourceOptions;
@FXML private TextField filePathField;
@FXML private ComboBox<String> microcontrollerCombo;
@FXML private ComboBox<String> portCombo;
@FXML private ComboBox<String> baudRateCombo;

// Models
private RangeConfiguration rangeConfiguration;
private SecuritySettings securitySettings;
private DataSourceSettings dataSourceSettings;

// Services
private final FileIOService fileIOService = new FileIOService();
private final SerialPortManager serialPortManager = new SerialPortManager();

@Override
public void initialize(URL url, ResourceBundle resourceBundle) {
    // This will be called after all @FXML fields are injected
}

/**
 * Initializes the controller with the models from the main controller.
 */
public void initializeWithModels() {
    MainController mainController = getParentController(MainController.class);
    if (mainController != null) {
        this.rangeConfiguration = mainController.getRangeConfiguration();
        this.securitySettings = mainController.getSecuritySettings();
        this.dataSourceSettings = mainController.getDataSourceSettings();

        // Initialize UI with model values
        updateUIFromModels();
    }
}

/**
 * Updates the UI components with values from the models.
 */
private void updateUIFromModels() {
    // Range Configuration
    switch (rangeConfiguration.getRangeType()) {
        case BITS:
            bitsRangeRadio.setSelected(true);
            customRangeOptions.setVisible(false);
            customRangeOptions.setManaged(false);

```

```

        break;
    case DEFAULT:
        defaultRangeRadio.setSelected(true);
        customRangeOptions.setVisible(false);
        customRangeOptions.setManaged(false);
        break;
    case CUSTOM:
        customRangeRadio.setSelected(true);
        customRangeOptions.setVisible(true);
        customRangeOptions.setManaged(true);
        minValueField.setText(String.valueOf(rangeConfiguration.getMinValue()));
        maxValueField.setText(String.valueOf(rangeConfiguration.getMaxValue()));
        break;
}

// Security Settings
securitySlider.setValue(securitySettings.getSecurityLevel());
lengthBasedSecurityCheck.setSelected(securitySettings.isLengthBasedSecurity());
sequenceLengthSpinner.getValueFactory().setValue(securitySettings.getSequenceLength());
sequenceLengthSpinner.setDisable(!securitySettings.isLengthBasedSecurity());

roundsBasedSecurityCheck.setSelected(securitySettings.isRoundsBasedSecurity());
roundsSpinner.getValueFactory().setValue(securitySettings.getRounds());
roundsSpinner.setDisable(!securitySettings.isRoundsBasedSecurity());

timeBasedSecurityCheck.setSelected(securitySettings.isTimeBasedSecurity());
generationTimeSpinner.getValueFactory().setValue(securitySettings.getGenerationTime());
generationTimeSpinner.setDisable(!securitySettings.isTimeBasedSecurity());

// Data Source Settings
if (dataSourceSettings.isFileSource()) {
    fileSourceRadio.setSelected(true);
    fileSourceOptions.setVisible(true);
    fileSourceOptions.setManaged(true);
    realtimeSourceOptions.setVisible(false);
    realtimeSourceOptions.setManaged(false);
    filePathField.setText(dataSourceSettings.getFilePath());
} else {
    realtimeSourceRadio.setSelected(true);
    fileSourceOptions.setVisible(false);
    fileSourceOptions.setManaged(false);
    realtimeSourceOptions.setVisible(true);
    realtimeSourceOptions.setManaged(true);
    microcontrollerCombo.getSelectionModel().select(dataSourceSettings.getMicrocontrollerType());
    portCombo.getSelectionModel().select(dataSourceSettings.getPort());
    baudRateCombo.getSelectionModel().select(dataSourceSettings.getBaudRate());
}
}

```

```

@Override
public void refreshView() {
    updateUIFromModels();
}

@Override
public void updateRangeConfiguration() {
    if (bitsRangeRadio.isSelected()) {
        rangeConfiguration.setRangeType(RangeType.BITS);
    } else if (defaultRangeRadio.isSelected()) {
        rangeConfiguration.setRangeType(RangeType.DEFAULT);
    } else if (customRangeRadio.isSelected()) {
        rangeConfiguration.setRangeType(RangeType.CUSTOM);
        try {
            int min = Integer.parseInt(minValueField.getText());
            int max = Integer.parseInt(maxValueField.getText());

            if (min < max) {
                rangeConfiguration.updateCustomRange(min, max);
            } else {
                // Reset to default if invalid
                minValueField.setText("1");
                maxValueField.setText("100");
                rangeConfiguration.updateCustomRange(1, 100);
            }
        } catch (NumberFormatException e) {
            // Reset to default if invalid
            minValueField.setText("1");
            maxValueField.setText("100");
            rangeConfiguration.updateCustomRange(1, 100);
        }
    }
}

@Override
public void updateSecuritySettings() {
    securitySettings.setLengthBasedSecurity(lengthBasedSecurityCheck.isSelected());
    securitySettings.setRoundsBasedSecurity(roundsBasedSecurityCheck.isSelected());
    securitySettings.setTimeBasedSecurity(timeBasedSecurityCheck.isSelected());

    securitySettings.setSequenceLength(sequenceLengthSpinner.getValue());
    securitySettings.setRounds(roundsSpinner.getValue());
    securitySettings.setGenerationTime(generationTimeSpinner.getValue());

    // Calculate and update security level
    int securityLevel = securitySettings.calculateSecurityLevel(dataSourceSettings.isRealtimeSource());
    securitySlider.setValue(securityLevel);
}

```

```

}

@Override
public void updateDataSourceSettings() {
    if (fileSourceRadio.isSelected()) {
        dataSourceSettings.setSourceType(DataSourceSettings.SourceType.FILE);
    } else {
        dataSourceSettings.setSourceType(DataSourceSettings.SourceType.REALTIME);
    }

    dataSourceSettings.setFilePath(filePathField.getText());

    if (microcontrollerCombo.getSelectionModel().getSelectedItem() != null) {
        dataSourceSettings.setMicrocontrollerType(microcontrollerCombo.getSelectionModel().getSelectedItem());
    }

    if (portCombo.getSelectionModel().getSelectedItem() != null) {
        dataSourceSettings.setPort(portCombo.getSelectionModel().getSelectedItem());
    }

    if (baudRateCombo.getSelectionModel().getSelectedItem() != null) {
        dataSourceSettings.setBaudRate(baudRateCombo.getSelectionModel().getSelectedItem());
    }
}

@Override
public RangeConfiguration getRangeConfiguration() {
    return rangeConfiguration;
}

@Override
public SecuritySettings getSecuritySettings() {
    return securitySettings;
}

@Override
public DataSourceSettings getDataSourceSettings() {
    return dataSourceSettings;
}

/**
 * Event handler for range type radio buttons.
 */
@FXML
protected void onRangeTypeChanged() {
    updateRangeConfiguration();
    refreshView();
}

```

```

/**
 * Event handler for security control checkboxes.
 */
@FXML
protected void onSecurityControlChanged() {
    sequenceLengthSpinner.setDisable(!lengthBasedSecurityCheck.isSelected());
    roundsSpinner.setDisable(!roundsBasedSecurityCheck.isSelected());
    generationTimeSpinner.setDisable(!timeBasedSecurityCheck.isSelected());
}

/**
 * Event handler for applying security settings.
 */
@FXML
protected void onApplySecuritySettings() {
    updateSecuritySettings();
}

/**
 * Event handler for source type radio buttons.
 */
@FXML
protected void onSourceChanged() {
    boolean isFileSource = fileSourceRadio.isSelected();

    fileSourceOptions.setVisible(isFileSource);
    fileSourceOptions.setManaged(isFileSource);
    realtimeSourceOptions.setVisible(!isFileSource);
    realtimeSourceOptions.setManaged(!isFileSource);

    // Enable/disable measurement length option based on source
    if (isFileSource) {
        lengthBasedSecurityCheck.setSelected(false);
        lengthBasedSecurityCheck.setDisable(true);
        sequenceLengthSpinner.setDisable(true);
    } else {
        lengthBasedSecurityCheck.setDisable(false);
        // The spinner will be enabled/disabled based on checkbox
        sequenceLengthSpinner.setDisable(!lengthBasedSecurityCheck.isSelected());
    }

    updateDataSourceSettings();
}

/**
 * Event handler for browse file button.
 */

```

```

@FXML
protected void onBrowseFile() {
    var selectedPath = fileIOService.selectFile(
        filePathField.getScene().getWindow(),
        "Select Measurement File",
        new FileChooser.ExtensionFilter("Text Files", "*.txt")
    );

    if (selectedPath != null) {
        filePathField.setText(selectedPath.toAbsolutePath().toString());
        updateDataSourceSettings();
    }
}

/**
 * Event handler for refresh ports button.
 */
@FXML
protected void onRefreshPorts() {
    portCombo.getItems().clear();
    portCombo.getItems().addAll(serialPortManager.getAvailablePorts());

    if (!portCombo.getItems().isEmpty()) {
        portCombo.getSelectionModel().selectFirst();
    }

    updateDataSourceSettings();
}

/**
 * Event handler for next button.
 */
@FXML
protected void onNextToGenerate() {
    // Update all models before switching tabs
    updateRangeConfiguration();
    updateSecuritySettings();
    updateDataSourceSettings();

    // Switch to the generation tab
    MainController mainController = getParentController(MainController.class);
    if (mainController != null) {
        mainController.switchToTab(1); // Generation tab index

        // Refresh the generation controller to pick up the new settings
        GenerationControllerInterface generationController = mainController.getGenerationController();
        if (generationController != null) {
            generationController.refreshView();
        }
    }
}

```

```

    }
  }
}

```

```

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/controller/SettingsControllerInterface.java
=====
=====

```

```

package ua.edu.chnu.imprrandgen.controller;

import ua.edu.chnu.imprrandgen.model.DataSourceSettings;
import ua.edu.chnu.imprrandgen.model.RangeConfiguration;
import ua.edu.chnu.imprrandgen.model.SecuritySettings;

/**
 * Interface for the settings tab controller.
 */
public interface SettingsControllerInterface extends Controller {

    /**
     * Updates the range configuration based on user input.
     */
    void updateRangeConfiguration();

    /**
     * Updates the security settings based on user input.
     */
    void updateSecuritySettings();

    /**
     * Updates the data source settings based on user input.
     */
    void updateDataSourceSettings();

    /**
     * @return The current range configuration
     */
    RangeConfiguration getRangeConfiguration();

    /**
     * @return The current security settings
     */
    SecuritySettings getSecuritySettings();

    /**

```

```

    * @return The current data source settings
    */
    DataSourceSettings getDataSourceSettings();
}

```

```

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/model/DataSourceSettings.java
=====
=====

```

```

package ua.edu.chnu.imprrandgen.model;

/**
 * Model class for managing data source settings for random number generation.
 */
public class DataSourceSettings {
    /**
     * Enum representing the different types of data sources.
     */
    public enum SourceType {
        /**
         * Data from a file (past measurements)
         */
        FILE,

        /**
         * Real-time data from a device
         */
        REALTIME
    }

    private SourceType sourceType;
    private String filePath;
    private String microcontrollerType;
    private String port;
    private String baudRate;

    /**
     * Creates a new data source settings object with default values.
     */
    public DataSourceSettings() {
        this.sourceType = SourceType.FILE;
        this.filePath = "";
        this.microcontrollerType = "Arduino Uno";
        this.port = "COM1";
        this.baudRate = "115200";
    }
}

```

```

/**
 * @return The current source type
 */
public SourceType getSourceType() {
    return sourceType;
}

/**
 * Sets the source type.
 *
 * @param sourceType The new source type
 */
public void setSourceType(SourceType sourceType) {
    this.sourceType = sourceType;
}

/**
 * @return Whether the source type is FILE
 */
public boolean isFileSource() {
    return sourceType == SourceType.FILE;
}

/**
 * @return Whether the source type is REALTIME
 */
public boolean isRealtimeSource() {
    return sourceType == SourceType.REALTIME;
}

/**
 * @return The current file path
 */
public String getFilePath() {
    return filePath;
}

/**
 * Sets the file path.
 *
 * @param filePath The new file path
 */
public void setFilePath(String filePath) {
    this.filePath = filePath;
}

/**

```

```

    * @return The current microcontroller type
    */
    public String getMicrocontrollerType() {
        return microcontrollerType;
    }

    /**
     * Sets the microcontroller type.
     *
     * * @param microcontrollerType The new microcontroller type
     * */
    public void setMicrocontrollerType(String microcontrollerType) {
        this.microcontrollerType = microcontrollerType;
    }

    /**
     * * @return The current port
     * */
    public String getPort() {
        return port;
    }

    /**
     * Sets the port.
     *
     * * @param port The new port
     * */
    public void setPort(String port) {
        this.port = port;
    }

    /**
     * * @return The current baud rate
     * */
    public String getBaudRate() {
        return baudRate;
    }

    /**
     * Sets the baud rate.
     *
     * * @param baudRate The new baud rate
     * */
    public void setBaudRate(String baudRate) {
        this.baudRate = baudRate;
    }

    /**

```

```

    * @return A descriptive string of the current data source configuration
    */
    public String getDescription() {
        return switch (sourceType) {
            case FILE -> {
                if (filePath != null && !filePath.isEmpty()) {
                    String fileName = new java.io.File(filePath).getName();
                    String truncatedName = fileName.length() > 20
                        ? fileName.substring(0, 17) + "..."
                        : fileName;
                    yield "File (%s)".formatted(truncatedName);
                } else {
                    yield "File (No file selected)";
                }
            }
            case REALTIME -> "Realtime (%s)".formatted(microcontrollerType);
        };
    }
}

```

```

=====
=====

```

FILE: ./src/main/java/ua/edu/chnu/imprrandgen/model/GenerationResult.java

```

=====
=====

```

```

package ua.edu.chnu.imprrandgen.model;

/**
 * Represents the result of a random sequence generation.
 * Stores all relevant data about the generation including timestamp, data, range settings,
 * entropy, distribution statistics, and security level.
 *
 * @param timestamp The timestamp when the sequence was generated
 * @param data The generated sequence data (defensive copy made)
 * @param rangeType The type of range used for generation
 * @param minValu The minimum value in the range
 * @param maxValu The maximum value in the range
 * @param entropy The calculated entropy of the sequence
 * @param distribution The distribution statistics as a formatted string
 * @param securityLevel The security level used for generation
 */
public record GenerationResult(
    String timestamp,
    int[] data,
    RangeType rangeType,
    int minValu,
    int maxValu,

```

```

    double entropy,
    String distribution,
    int securityLevel
) {
    /**
     * Compact constructor that creates defensive copies of mutable parameters.
     */
    public GenerationResult {
        data = data.clone();
    }

    /**
     * Returns a defensive copy of the sequence data.
     *
     * @return A copy of the generated sequence data
     */
    @Override
    public int[] data() {
        return data.clone();
    }

    /**
     * Legacy getter for backward compatibility.
     *
     * @return The timestamp when the sequence was generated
     */
    public String getTimestamp() {
        return timestamp;
    }

    /**
     * Legacy getter for backward compatibility.
     *
     * @return A copy of the generated sequence data
     */
    public int[] getData() {
        return data();
    }

    /**
     * Legacy getter for backward compatibility.
     *
     * @return The type of range used for generation
     */
    public RangeType getRangeType() {
        return rangeType;
    }
}

```

```

/**
 * Legacy getter for backward compatibility.
 *
 * @return The minimum value in the range
 */
public int getMinValue() {
    return minValue;
}

/**
 * Legacy getter for backward compatibility.
 *
 * @return The maximum value in the range
 */
public int getMaxValue() {
    return maxValue;
}

/**
 * Legacy getter for backward compatibility.
 *
 * @return The calculated entropy of the sequence
 */
public double getEntropy() {
    return entropy;
}

/**
 * Legacy getter for backward compatibility.
 *
 * @return The distribution statistics as a formatted string
 */
public String getDistribution() {
    return distribution;
}

/**
 * Legacy getter for backward compatibility.
 *
 * @return The security level used for generation
 */
public int getSecurityLevel() {
    return securityLevel;
}

/**
 * Returns a string representation of the generation result.
 *

```

```

    * @return A formatted string describing the generation result
    */
    @Override
    public String toString() {
        String rangeDescription = switch (rangeType) {
            case BITS -> "Bits";
            case DEFAULT -> "Default Range";
            case CUSTOM -> "Custom Range (%d-%d)".formatted(minValue, maxValue);
        };

        return "%s - %s - Security: %d".formatted(timestamp, rangeDescription, securityLevel);
    }
}

```

```

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprandgen/model/RangeConfiguration.java
=====
=====

```

```

package ua.edu.chnu.imprandgen.model;

/**
 * Model class for managing range configuration settings for random number generation.
 */
public class RangeConfiguration {
    private RangeType rangeType;
    private int minValue;
    private int maxValue;

    /**
     * Creates a new range configuration with default values (BITS range).
     */
    public RangeConfiguration() {
        this.rangeType = RangeType.BITS;
        this.minValue = 0;
        this.maxValue = 1;
    }

    /**
     * Creates a new range configuration with specified values.
     *
     * @param rangeType The type of range
     * @param minValue The minimum value in the range
     * @param maxValue The maximum value in the range
     */
    public RangeConfiguration(RangeType rangeType, int minValue, int maxValue) {

```

```

        this.rangeType = rangeType;
        this.minValue = minValue;
        this.maxValue = maxValue;
    }

    /**
     * @return The current range type
     */
    public RangeType getRangeType() {
        return rangeType;
    }

    /**
     * Sets the range type and updates min/max values accordingly.
     *
     * @param rangeType The new range type
     */
    public void setRangeType(RangeType rangeType) {
        this.rangeType = rangeType;

        switch (rangeType) {
            case BITS -> {
                this.minValue = 0;
                this.maxValue = 1;
            }
            case DEFAULT -> {
                this.minValue = 1;
                this.maxValue = 256;
            }
            case CUSTOM -> {
                // Keep existing values
            }
        }
    }

    /**
     * @return The minimum value in the range
     */
    public int getMinValue() {
        return minValue;
    }

    /**
     * Sets the minimum value for the range.
     * Only effective when range type is CUSTOM.
     *
     * @param minValue The new minimum value
     */

```

```

public void setMinValue(int minValue) {
    if (rangeType == RangeType.CUSTOM) {
        this.minValue = minValue;
    }
}

/**
 * @return The maximum value in the range
 */
public int getMaxValue() {
    return max_value;
}

/**
 * Sets the maximum value for the range.
 * Only effective when range type is CUSTOM.
 *
 * @param max_value The new maximum value
 */
public void setMaxValue(int max_value) {
    if (rangeType == RangeType.CUSTOM) {
        this.max_value = max_value;
    }
}

/**
 * Updates the custom range with new min and max values.
 * Only effective when range type is CUSTOM.
 *
 * @param minValue The new minimum value
 * @param max_value The new maximum value
 * @return true if the update was successful, false otherwise
 */
public boolean updateCustomRange(int minValue, int max_value) {
    if (rangeType != RangeType.CUSTOM || minValue >= max_value) {
        return false;
    }

    this.minValue = minValue;
    this.max_value = max_value;
    return true;
}

/**
 * @return A descriptive string of the current range configuration
 */
public String getDescription() {
    return switch (rangeType) {

```

```

        case BITS -> "Bits (0-1)";
        case DEFAULT -> "Default Range (1-256)";
        case CUSTOM -> "Custom Range (%d-%d)".formatted(minValue, maxValue);
    };
}
}

```

```

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/model/RangeType.java
=====
=====

```

```

package ua.edu.chnu.imprrandgen.model;

/**
 * Enum representing the different types of ranges that can be used for random number generation.
 */
public enum RangeType {
    /**
     * Binary range (0-1)
     */
    BITS,

    /**
     * Default range (1-256)
     */
    DEFAULT,

    /**
     * Custom range with user-defined min and max values
     */
    CUSTOM
}

```

```

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/model/SecuritySettings.java
=====
=====

```

```

package ua.edu.chnu.imprrandgen.model;

/**
 * Model class for managing security settings for random number generation.
 */
public class SecuritySettings {

```

```

private int securityLevel;
private boolean lengthBasedSecurity;
private boolean roundsBasedSecurity;
private boolean timeBasedSecurity;
private int sequenceLength;
private int rounds;
private int generationTime;

/**
 * Creates a new security settings object with default values.
 */
public SecuritySettings() {
    this.securityLevel = 5; // Default middle value
    this.lengthBasedSecurity = false;
    this.roundsBasedSecurity = false;
    this.timeBasedSecurity = false;
    this.sequenceLength = 2000;
    this.rounds = 5;
    this.generationTime = 5;
}

/**
 * @return The current security level (1-10)
 */
public int getSecurityLevel() {
    return securityLevel;
}

/**
 * Sets the security level.
 *
 * @param securityLevel The new security level (1-10)
 */
public void setSecurityLevel(int securityLevel) {
    this.securityLevel = Math.max(1, Math.min(10, securityLevel));
}

/**
 * @return Whether length-based security is enabled
 */
public boolean isLengthBasedSecurity() {
    return lengthBasedSecurity;
}

/**
 * Enables or disables length-based security.
 *
 * @param lengthBasedSecurity The new state

```

```

*/
public void setLengthBasedSecurity(boolean lengthBasedSecurity) {
    this.lengthBasedSecurity = lengthBasedSecurity;
}

/**
 * @return Whether rounds-based security is enabled
 */
public boolean isRoundsBasedSecurity() {
    return roundsBasedSecurity;
}

/**
 * Enables or disables rounds-based security.
 *
 * @param roundsBasedSecurity The new state
 */
public void setRoundsBasedSecurity(boolean roundsBasedSecurity) {
    this.roundsBasedSecurity = roundsBasedSecurity;
}

/**
 * @return Whether time-based security is enabled
 */
public boolean isTimeBasedSecurity() {
    return timeBasedSecurity;
}

/**
 * Enables or disables time-based security.
 *
 * @param timeBasedSecurity The new state
 */
public void setTimeBasedSecurity(boolean timeBasedSecurity) {
    this.timeBasedSecurity = timeBasedSecurity;
}

/**
 * @return The current sequence length
 */
public int getSequenceLength() {
    return sequenceLength;
}

/**
 * Sets the sequence length.
 *
 * @param sequenceLength The new sequence length

```

```

*/
public void setSequenceLength(int sequenceLength) {
    this.sequenceLength = sequenceLength;
}

/**
 * @return The current number of rounds
 */
public int getRounds() {
    return rounds;
}

/**
 * Sets the number of rounds.
 *
 * @param rounds The new number of rounds
 */
public void setRounds(int rounds) {
    this.rounds = rounds;
}

/**
 * @return The current generation time in seconds
 */
public int getGenerationTime() {
    return generationTime;
}

/**
 * Sets the generation time.
 *
 * @param generationTime The new generation time in seconds
 */
public void setGenerationTime(int generationTime) {
    this.generationTime = generationTime;
}

/**
 * Calculates the security level based on the current settings.
 * Updates the securityLevel field and returns the calculated value.
 *
 * @param isRealtimeSource Whether the data source is real-time
 * @return The calculated security level
 */
public int calculateSecurityLevel(boolean isRealtimeSource) {
    int calculatedLevel = 5; // Default middle value

    if (lengthBasedSecurity) {

```

```

    if (isRealtimeSource) {
        // Map length from 1000-10000 to security level 1-10
        int lengthBasedLevel = Math.max(1, Math.min(10, (sequenceLength - 1000) / 1000 + 1));
        calculatedLevel = Math.max(calculatedLevel, lengthBasedLevel);
    } else {
        // For file-based measurements
        int mockFileLength = 2000;
        // Ensure measurement length is less than file length
        int length = Math.min(sequenceLength, mockFileLength - 1);
        // Map length from 1000-2000 to security level 1-10
        int lengthBasedLevel = Math.max(1, Math.min(10, (length - 1000) / 100 + 1));
        calculatedLevel = Math.max(calculatedLevel, lengthBasedLevel);
    }
}

if (roundsBasedSecurity) {
    // Map rounds from 1-20 to security level 1-10
    int roundsBasedLevel = Math.max(1, Math.min(10, rounds / 2));
    calculatedLevel = Math.max(calculatedLevel, roundsBasedLevel);
}

if (timeBasedSecurity) {
    // Map time from 1-30 to security level 1-10
    int timeBasedLevel = Math.max(1, Math.min(10, generationTime / 3));
    calculatedLevel = Math.max(calculatedLevel, timeBasedLevel);
}

this.securityLevel = calculatedLevel;
return calculatedLevel;
}
}

```

```

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/model/SequenceAnalyzer.java
=====
=====

```

```

package ua.edu.chnu.imprrandgen.model;

/**
 * Model class for analyzing random sequences and calculating various statistical properties.
 */
public class SequenceAnalyzer {

    /**
     * Calculates Shannon entropy for a sequence of numbers in the given range.
     */
}

```

```

* @param data    The sequence data to analyze
* @param minValue The minimum value in the range
* @param maxValue The maximum value in the range
* @return        The normalized entropy value (0-1)
*/
public static double calculateEntropy(int[] data, int minValue, int maxValue) {
    int range = maxValue - minValue + 1;

    // Count occurrences of each value in the range
    int[] counts = new int[range];
    for (int value : data) {
        if (value >= minValue && value <= maxValue) {
            counts[value - minValue]++;
        }
    }

    // Calculate entropy (Shannon formula)
    double entropy = 0.0;
    for (int count : counts) {
        if (count > 0) {
            double probability = (double) count / data.length;
            entropy -= probability * (Math.log(probability) / Math.log(2));
        }
    }

    // Normalize entropy to 0-1 range (max entropy depends on range size)
    double maxEntropy = Math.log(range) / Math.log(2);
    return maxEntropy > 0 ? entropy / maxEntropy : 0;
}

/**
 * Calculates distribution statistics for a sequence of numbers.
 *
 * @param data The sequence data to analyze
 * @return A formatted string with distribution statistics
 */
public static String calculateDistributionStats(int[] data) {
    if (data.length == 0) return "N/A";

    // Calculate distribution statistics
    double sum = 0;
    int min = Integer.MAX_VALUE;
    int max = Integer.MIN_VALUE;
    for (int value : data) {
        sum += value;
        if (value < min) min = value;
        if (value > max) max = value;
    }
}

```

```

    double mean = sum / data.length;

    // Calculate standard deviation
    double variance = 0;
    for (int value : data) {
        variance += Math.pow(value - mean, 2);
    }
    variance /= data.length;
    double stdDev = Math.sqrt(variance);

    return String.format("Mean: %.1f, StdDev: %.1f, Range: %d-%d",
        mean, stdDev, min, max);
}

/**
 * Calculates the frequency test score (uniformity of distribution).
 *
 * @param data    The sequence data to analyze
 * @param minValue The minimum value in the range
 * @param maxValue The maximum value in the range
 * @return        The frequency test score (0-1)
 */
public static double calculateFrequencyScore(int[] data, int minValue, int maxValue) {
    // Count occurrences of each value in the range
    int range = maxValue - minValue + 1;
    int[] counts = new int[range];
    for (int value : data) {
        if (value >= minValue && value <= maxValue) {
            counts[value - minValue]++;
        }
    }

    // Calculate chi-square statistic for uniform distribution
    double expectedCount = (double) data.length / range;
    double chiSquare = 0;
    for (int count : counts) {
        chiSquare += Math.pow(count - expectedCount, 2) / expectedCount;
    }

    // Convert chi-square to a score between 0 and 1
    // Lower chi-square means more uniform distribution
    double freqScore = Math.exp(-chiSquare / 1000); // Exponential decay
    return Math.max(0, Math.min(1, freqScore)); // Clamp to 0-1
}

/**
 * Calculates the runs test score (sequential correlation).
 *

```

```

* @param data The sequence data to analyze
* @return The runs test score (0-1)
*/
public static double calculateRunsScore(int[] data) {
    if (data.length < 2) return 0;

    int runs = 0;
    double sumDiff = 0;
    for (int i = 1; i < data.length; i++) {
        // Count direction changes (up/down)
        if ((data[i] > data[i-1] && i > 1 && data[i-1] <= data[i-2]) ||
            (data[i] < data[i-1] && i > 1 && data[i-1] >= data[i-2])) {
            runs++;
        }
        // Sum of absolute differences
        sumDiff += Math.abs(data[i] - data[i-1]);
    }

    // Calculate average difference
    double avgDiff = sumDiff / (data.length - 1);
    // Expected average difference for uniform random numbers in range 1-256 is about 85.33
    double expectedAvgDiff = 256.0 / 3.0;

    // Score based on how close the average difference is to expected
    double runsScore = 1.0 - Math.abs(avgDiff - expectedAvgDiff) / expectedAvgDiff;
    return Math.max(0, Math.min(1, runsScore)); // Clamp to 0-1
}

/**
 * Calculates the longest run test score.
 */
* @param data The sequence data to analyze
* @param minValue The minimum value in the range
* @param maxValue The maximum value in the range
* @return The longest run test score (0-1)
*/
public static double calculateLongestRunScore(int[] data, int minValue, int maxValue) {
    // This test focuses on the longest run of consecutive ones within the sequence
    int longestRun = 0;
    int currentRun = 0;
    int midpoint = (minValue + maxValue) / 2;

    for (int value : data) {
        if (value > midpoint) {
            currentRun++;
            if (currentRun > longestRun) {
                longestRun = currentRun;
            }
        }
    }
}

```

```

    } else {
        currentRun = 0;
    }
}

// Calculate expected longest run based on sequence length
double expectedLongestRun = Math.log(data.length) / Math.log(2);
double longestRunScore = 1.0 - Math.abs(longestRun - expectedLongestRun) / expectedLongestRun;
return Math.max(0, Math.min(1, longestRunScore)); // Clamp to 0-1
}

/**
 * Calculates the cumulative sums test score.
 *
 * @param data    The sequence data to analyze
 * @param minValue The minimum value in the range
 * @param maxValue The maximum value in the range
 * @return        The cumulative sums test score (0-1)
 */
public static double calculateCusumsScore(int[] data, int minValue, int maxValue) {
    // This test looks at the maximal excursion (from zero) of random walks
    int maxExcursion = 0;
    int currentExcursion = 0;
    int midpoint = (minValue + maxValue) / 2;

    for (int value : data) {
        // Add 1 if above midpoint, subtract 1 if below
        currentExcursion += (value > midpoint) ? 1 : -1;
        maxExcursion = Math.max(maxExcursion, Math.abs(currentExcursion));
    }

    // Calculate expected max excursion based on sequence length
    double expectedMaxExcursion = Math.sqrt(data.length);
    double cusumsScore = 1.0 - Math.abs(maxExcursion - expectedMaxExcursion) / expectedMaxExcursion;
    return Math.max(0, Math.min(1, cusumsScore)); // Clamp to 0-1
}
}

```

```

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/service/FileIOService.java
=====
=====

```

```

package ua.edu.chnu.imprrandgen.service;

import javafx.stage.FileChooser;
import javafx.stage.Window;

```

```

import ua.edu.chnu.imprrandgen.model.GenerationResult;
import ua.edu.chnu.imprrandgen.model.RangeType;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;

/**
 * Service for handling file input/output operations.
 */
public class FileIOService {

    /**
     * Shows a file selection dialog.
     *
     * @param owner The owner window for the dialog
     * @param title The title of the dialog
     * @param filters Extension filters for file selection
     * @return The selected file, or null if cancelled
     */
    public Path selectFile(Window owner, String title, FileChooser.ExtensionFilter... filters) {
        var fileChooser = new FileChooser();
        fileChooser.setTitle(title);
        fileChooser.getExtensionFilters().addAll(filters);
        var file = fileChooser.showOpenDialog(owner);
        return file != null ? file.toPath() : null;
    }

    /**
     * Shows a file save dialog.
     *
     * @param owner The owner window for the dialog
     * @param title The title of the dialog
     * @param filters Extension filters for file selection
     * @return The selected file, or null if cancelled
     */
    public Path selectSaveFile(Window owner, String title, FileChooser.ExtensionFilter... filters) {
        var fileChooser = new FileChooser();
        fileChooser.setTitle(title);
        fileChooser.getExtensionFilters().addAll(filters);
        var file = fileChooser.showSaveDialog(owner);
        return file != null ? file.toPath() : null;
    }

    /**
     * Saves a sequence to a file.
     *
     * @param data The sequence data to save

```

```

    * @param file    The file to save to
    * @param rangeType The type of range used for the sequence
    * @throws IOException If an I/O error occurs
    */
    public void saveSequenceToFile(int[] data, Path file, RangeType rangeType) throws IOException {
        var content = formatSequenceForFile(data, rangeType);
        Files.writeString(file, content);
    }

    /**
     * Formats a sequence for file output.
     *
     * @param data    The sequence data
     * @param rangeType The range type
     * @return The formatted sequence string
     */
    private String formatSequenceForFile(int[] data, RangeType rangeType) {
        var builder = new StringBuilder();

        for (int i = 0; i < data.length; i++) {
            if (rangeType == RangeType.BITS) {
                builder.append(data[i]);
                if ((i + 1) % 80 == 0) {
                    builder.append('\n');
                }
            } else {
                builder.append("%3d ".formatted(data[i]));
                if ((i + 1) % 16 == 0) {
                    builder.append('\n');
                }
            }
        }

        return builder.toString();
    }

    /**
     * Exports an analysis report to PDF format.
     *
     * @param result The generation result to export
     * @param file    The file to save to
     * @throws IOException If an I/O error occurs
     */
    public void exportReportToPDF(GenerationResult result, Path file) throws IOException {
        throw new UnsupportedOperationException("PDF export not yet implemented");
    }

    /**

```

```

* Exports an analysis report to HTML format.
*
* @param result The generation result to export
* @param file The file to save to
* @throws IOException If an I/O error occurs
*/
public void exportReportToHTML(GenerationResult result, Path file) throws IOException {
    var html = """
        <!DOCTYPE html>
        <html>
        <head>
            <title>Random Sequence Analysis Report</title>
            <style>
                body {
                    font-family: Arial, sans-serif;
                    margin: 20px;
                    background-color: #f5f5f5;
                }
                .container {
                    background-color: white;
                    padding: 20px;
                    border-radius: 8px;
                    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
                }
                h1 {
                    color: #333;
                    border-bottom: 2px solid #007bff;
                    padding-bottom: 10px;
                }
                .info-row {
                    margin: 10px 0;
                    padding: 8px;
                    background-color: #f8f9fa;
                    border-left: 3px solid #007bff;
                }
                .label {
                    font-weight: bold;
                    color: #555;
                }
                .value {
                    color: #333;
                }
            </style>
        </head>
        <body>
            <div class="container">
                <h1>Random Sequence Analysis Report</h1>
                <div class="info-row">

```

```

        <span class="label">Timestamp:</span>
        <span class="value">%s</span>
    </div>
    <div class="info-row">
        <span class="label">Range Type:</span>
        <span class="value">%s</span>
    </div>
    <div class="info-row">
        <span class="label">Sequence Length:</span>
        <span class="value">%d</span>
    </div>
    <div class="info-row">
        <span class="label">Security Level:</span>
        <span class="value">%d</span>
    </div>
    <div class="info-row">
        <span class="label">Entropy:</span>
        <span class="value">%.4f</span>
    </div>
    <div class="info-row">
        <span class="label">Distribution:</span>
        <span class="value">%s</span>
    </div>
</div>
</body>
</html>
""" .formatted(
    result.timestamp(),
    result.rangeType(),
    result.data().length,
    result.securityLevel(),
    result.entropy(),
    result.distribution()
);

Files.writeString(file, html);
}
}

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/service/RealtimeGenerationService.java
=====
=====

package ua.edu.chnu.imprrandgen.service;

```

```

import ua.edu.chnu.imprrandgen.model.RangeConfiguration;
import ua.edu.chnu.imprrandgen.model.SecuritySettings;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;
import java.util.concurrent.atomic.AtomicBoolean;
import java.util.function.Consumer;

/**
 * Service for managing real-time sequence generation.
 */
public class RealtimeGenerationService {

    private final ExecutorService executorService;
    private final SequenceGenerator sequenceGenerator;
    private final AtomicBoolean isGenerating;
    private final AtomicBoolean isPaused;
    private Future<?> currentTask;

    /**
     * Progress callback interface for real-time updates.
     */
    public interface GenerationProgress {
        int getCurrentCount();
        int[] getCurrentData();
    }

    public RealtimeGenerationService() {
        this.executorService = Executors.newSingleThreadExecutor(r -> {
            Thread thread = new Thread(r);
            thread.setDaemon(true);
            thread.setName("RealtimeGenerationThread");
            return thread;
        });
        this.sequenceGenerator = new SequenceGenerator();
        this.isGenerating = new AtomicBoolean(false);
        this.isPaused = new AtomicBoolean(false);
    }

    /**
     * Starts real-time generation.
     *
     * @param targetLength The target sequence length
     * @param rangeConfig The range configuration
     * @param securitySettings The security settings
     * @param random The PhPhRandom instance to use
     * @param progressCallback Callback for progress updates
     */

```

```

*/
public void startGeneration(int targetLength, RangeConfiguration rangeConfig,
    SecuritySettings securitySettings, PhphRandom random,
    Consumer<GenerationProgress> progressCallback) {
    if (isGenerating.get()) {
        return;
    }

    isGenerating.set(true);
    isPaused.set(false);

    currentTask = executorService.submit() -> {
        try {
            int[] sequence = sequenceGenerator.generateSequence(
                targetLength, rangeConfig, securitySettings, random
            );

            if (!Thread.currentThread().isInterrupted() && !isPaused.get()) {
                progressCallback.accept(new GenerationProgress() {
                    @Override
                    public int getCurrentCount() {
                        return sequence.length;
                    }

                    @Override
                    public int[] getCurrentData() {
                        return sequence;
                    }
                });
            }
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            isGenerating.set(false);
        }
    });
}

/**
 * Pauses the current generation.
 */
public void pauseGeneration() {
    isPaused.set(true);
}

/**
 * Resumes a paused generation.
 */

```

```

public void resumeGeneration() {
    isPaused.set(false);
}

/**
 * Stops the current generation.
 */
public void stopGeneration() {
    isGenerating.set(false);
    isPaused.set(false);
    if (currentTask != null && !currentTask.isDone()) {
        currentTask.cancel(true);
    }
}

/**
 * Resets the generation state.
 */
public void resetGeneration() {
    stopGeneration();
}

/**
 * Checks if generation is currently in progress.
 *
 * @return true if generating, false otherwise
 */
public boolean isGenerating() {
    return isGenerating.get();
}

/**
 * Checks if generation is currently paused.
 *
 * @return true if paused, false otherwise
 */
public boolean isPaused() {
    return isPaused.get();
}

/**
 * Shuts down the executor service.
 */
public void shutdown() {
    stopGeneration();
    executorService.shutdown();
}
}

```

```
=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/service/SequenceGenerator.java
=====
=====
```

```
package ua.edu.chnu.imprrandgen.service;

import ua.edu.chnu.imprrandgen.model.RangeConfiguration;
import ua.edu.chnu.imprrandgen.model.RangeType;
import ua.edu.chnu.imprrandgen.model.SecuritySettings;

import java.util.stream.IntStream;

/**
 * Service for generating random sequences.
 */
public class SequenceGenerator {

    private static final int DEFAULT_MIN_VALUE = 1;
    private static final int DEFAULT_MAX_VALUE = 256;
    private static final int DEFAULT_RANGE = DEFAULT_MAX_VALUE - DEFAULT_MIN_VALUE + 1;

    /**
     * Generates a random sequence based on the given configuration.
     *
     * @param length      The length of the sequence to generate
     * @param rangeConfig The range configuration
     * @param securitySettings The security settings
     * @param random      The PhphRandom instance to use
     * @return The generated sequence
     */
    public int[] generateSequence(int length, RangeConfiguration rangeConfig,
                                  SecuritySettings securitySettings, PhphRandom random) {
        var minValue = rangeConfig.getMinValue();
        var maxValue = rangeConfig.getMaxValue();
        var range = maxValue - minValue + 1;

        // Validate range and use defaults if invalid
        if (range <= 0) {
            minValue = DEFAULT_MIN_VALUE;
            maxValue = DEFAULT_MAX_VALUE;
            range = DEFAULT_RANGE;
        }

        final int finalMinValue = minValue;
```

```

    final int finalRange = range;

    return IntStream.range(0, length)
        .map(i -> random.nextInt(finalRange) + finalMinValue)
        .toArray();
}

/**
 * Formats a sequence for display.
 *
 * @param data    The sequence data
 * @param rangeType The type of range used
 * @return The formatted sequence as a string
 */
public String formatSequenceForDisplay(int[] data, RangeType rangeType) {
    var builder = new StringBuilder(data.length * 4); // Pre-size for efficiency

    return switch (rangeType) {
        case BITS -> formatBitsSequence(data, builder);
        default -> formatNumbersSequence(data, builder);
    };
}

/**
 * Formats a sequence of bits (0s and 1s).
 *
 * @param data    The sequence data
 * @param builder The StringBuilder to use
 * @return The formatted string
 */
private String formatBitsSequence(int[] data, StringBuilder builder) {
    for (int i = 0; i < data.length; i++) {
        builder.append(data[i]);
        if ((i + 1) % 80 == 0) {
            builder.append('\n');
        }
    }
    return builder.toString();
}

/**
 * Formats a sequence of numbers.
 *
 * @param data    The sequence data
 * @param builder The StringBuilder to use
 * @return The formatted string
 */
private String formatNumbersSequence(int[] data, StringBuilder builder) {

```

```

        for (int i = 0; i < data.length; i++) {
            builder.append("%3d ".formatted(data[i]));
            if ((i + 1) % 16 == 0) {
                builder.append("\n");
            }
        }
        return builder.toString();
    }

    /**
     * Calculates the generation delay based on security level.
     *
     * @param securityLevel The security level (1-10)
     * @return The delay in milliseconds
     */
    public long calculateGenerationDelay(int securityLevel) {
        return Math.clamp(securityLevel, 1, 10) * 500L;
    }
}

```

```

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/service/SerialPortManager.java
=====
=====

```

```

package ua.edu.chnu.imprrandgen.service;

import java.util.Arrays;
import java.util.List;

/**
 * Service for managing serial port connections to microcontrollers.
 * This is currently a mock implementation.
 */
public class SerialPortManager {

    private boolean connected = false;
    private String currentPort;
    private int currentBaudRate;

    /**
     * Gets a list of available serial ports.
     *
     * @return List of port names
     */
    public List<String> getAvailablePorts() {

```

```

    return Arrays.asList("COM1", "COM2", "COM3", "/dev/ttyUSB0", "/dev/ttyACM0");
}

/**
 * Connects to a serial port.
 *
 * @param port The port to connect to
 * @param baudRate The baud rate to use
 * @throws IllegalStateException if already connected
 */
public void connect(String port, int baudRate) {
    if (connected) {
        throw new IllegalStateException("Already connected to " + currentPort);
    }

    this.currentPort = port;
    this.currentBaudRate = baudRate;
    this.connected = true;
}

/**
 * Disconnects from the current serial port.
 */
public void disconnect() {
    this.connected = false;
    this.currentPort = null;
    this.currentBaudRate = 0;
}

/**
 * Reads data from the serial port.
 *
 * @param length The number of bytes to read
 * @return The data read from the port
 * @throws IllegalStateException if not connected
 */
public byte[] readData(int length) {
    if (!connected) {
        throw new IllegalStateException("Not connected to any port");
    }

    byte[] data = new byte[length];
    for (int i = 0; i < length; i++) {
        data[i] = (byte) (PhPhRandom.next());
    }
    return data;
}

```

```

/**
 * Checks if currently connected to a serial port.
 *
 * @return true if connected, false otherwise
 */
public boolean isConnected() {
    return connected;
}

/**
 * Gets the current port name.
 *
 * @return The port name, or null if not connected
 */
public String getCurrentPort() {
    return currentPort;
}

/**
 * Gets the current baud rate.
 *
 * @return The baud rate, or 0 if not connected
 */
public int getCurrentBaudRate() {
    return currentBaudRate;
}
}

```

```

=====
=====
FILE: ./src/main/java/ua/edu/chnu/imprrandgen/service/package-info.java
=====
=====

```

```

/**
 * Service layer for the random generator application.
 * Contains business logic and utility services that are used by controllers.
 */
package ua.edu.chnu.imprrandgen.service;

```

```

=====
=====
FILE: ./src/main/resources/ua/edu/chnu/imprrandgen/analysis-view.fxml
=====
=====

```

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<VBox spacing="20" style="-fx-padding: 20;" styleClass="card" xmlns:fx="http://javafx.com/fxml"
    fx:controller="ua.edu.chnu.imprrandgen.controller.AnalysisController">
    <Label text="Detailed Sequence Analysis" styleClass="section-header"/>

    <VBox spacing="20" VBox.vgrow="ALWAYS">
        <!-- Previous Tries -->
        <TitledPane text="Previous Tries" expanded="true">
            <VBox spacing="10">
                <HBox spacing="10" alignment="CENTER_LEFT">
                    <Label text="Select Previous Generation:"/>
                    <ComboBox fx:id="previousTriesCombo" prefWidth="300" HBox.hgrow="ALWAYS"/>
                </HBox>
                <HBox spacing="10" alignment="CENTER_RIGHT">
                    <Button text="Load Selected" onAction="#onLoadPreviousTry" styleClass="secondary"/>
                    <Button text="Compare with Current" onAction="#onComparePreviousTry" styleClass="secondary"/>
                </HBox>
            </VBox>
        </TitledPane>

        <!-- Randomness Tests -->
        <TitledPane text="Randomness Tests" expanded="true">
            <VBox spacing="10">
                <GridPane hgap="10" vgap="10">
                    <columnConstraints>
                        <ColumnConstraints hgrow="SOMETIMES" minWidth="200"/>
                        <ColumnConstraints hgrow="SOMETIMES" minWidth="100"/>
                        <ColumnConstraints hgrow="SOMETIMES"/>
                        <ColumnConstraints hgrow="SOMETIMES" minWidth="60"/>
                    </columnConstraints>

                    <Label text="Test" style="-fx-font-weight: bold;"
                        GridPane.rowIndex="0" GridPane.columnIndex="0"/>
                    <Label text="Result" style="-fx-font-weight: bold;"
                        GridPane.rowIndex="0" GridPane.columnIndex="1"/>
                    <Label text="Details" style="-fx-font-weight: bold;"
                        GridPane.rowIndex="0" GridPane.columnIndex="2"/>
                    <Label text="Score" style="-fx-font-weight: bold;"
                        GridPane.rowIndex="0" GridPane.columnIndex="3"/>

                    <Label text="Frequency (Monobit) Test"
                        GridPane.rowIndex="1" GridPane.columnIndex="0"/>
                    <Label fx:id="freqTestResult" text="N/A"
                        GridPane.rowIndex="1" GridPane.columnIndex="1"/>
                </GridPane>
            </VBox>
        </TitledPane>
    </VBox>
</VBox>

```

```

<ProgressBar fx:id="freqTestBar" progress="0" maxWidth="Infinity"
    GridPane.rowIndex="1" GridPane.columnIndex="2"/>
<Label fx:id="freqTestPercentage" text="0%" style="-fx-font-weight: bold;"
    GridPane.rowIndex="1" GridPane.columnIndex="3"/>

<Label text="Runs Test"
    GridPane.rowIndex="2" GridPane.columnIndex="0"/>
<Label fx:id="runsTestResult" text="N/A"
    GridPane.rowIndex="2" GridPane.columnIndex="1"/>
<ProgressBar fx:id="runsTestBar" progress="0" maxWidth="Infinity"
    GridPane.rowIndex="2" GridPane.columnIndex="2"/>
<Label fx:id="runsTestPercentage" text="0%" style="-fx-font-weight: bold;"
    GridPane.rowIndex="2" GridPane.columnIndex="3"/>

<Label text="Serial Test"
    GridPane.rowIndex="3" GridPane.columnIndex="0"/>
<Label fx:id="serialTestResult" text="N/A"
    GridPane.rowIndex="3" GridPane.columnIndex="1"/>
<ProgressBar fx:id="serialTestBar" progress="0" maxWidth="Infinity"
    GridPane.rowIndex="3" GridPane.columnIndex="2"/>
<Label fx:id="serialTestPercentage" text="0%" style="-fx-font-weight: bold;"
    GridPane.rowIndex="3" GridPane.columnIndex="3"/>

<Label text="Approximate Entropy Test"
    GridPane.rowIndex="4" GridPane.columnIndex="0"/>
<Label fx:id="entropyTestResult" text="N/A"
    GridPane.rowIndex="4" GridPane.columnIndex="1"/>
<ProgressBar fx:id="entropyTestBar" progress="0" maxWidth="Infinity"
    GridPane.rowIndex="4" GridPane.columnIndex="2"/>
<Label fx:id="entropyTestPercentage" text="0%" style="-fx-font-weight: bold;"
    GridPane.rowIndex="4" GridPane.columnIndex="3"/>

<Label text="Longest Run of Ones Test"
    GridPane.rowIndex="5" GridPane.columnIndex="0"/>
<Label fx:id="longestRunTestResult" text="N/A"
    GridPane.rowIndex="5" GridPane.columnIndex="1"/>
<ProgressBar fx:id="longestRunTestBar" progress="0" maxWidth="Infinity"
    GridPane.rowIndex="5" GridPane.columnIndex="2"/>
<Label fx:id="longestRunTestPercentage" text="0%" style="-fx-font-weight: bold;"
    GridPane.rowIndex="5" GridPane.columnIndex="3"/>

<Label text="Binary Matrix Rank Test"
    GridPane.rowIndex="6" GridPane.columnIndex="0"/>
<Label fx:id="matrixRankTestResult" text="N/A"
    GridPane.rowIndex="6" GridPane.columnIndex="1"/>
<ProgressBar fx:id="matrixRankTestBar" progress="0" maxWidth="Infinity"
    GridPane.rowIndex="6" GridPane.columnIndex="2"/>
<Label fx:id="matrixRankTestPercentage" text="0%" style="-fx-font-weight: bold;"

```

```

        GridPane.rowIndex="6" GridPane.columnIndex="3"/>

<Label text="Discrete Fourier Transform Test"
    GridPane.rowIndex="7" GridPane.columnIndex="0"/>
<Label fx:id="dftTestResult" text="N/A"
    GridPane.rowIndex="7" GridPane.columnIndex="1"/>
<ProgressBar fx:id="dftTestBar" progress="0" maxWidth="Infinity"
    GridPane.rowIndex="7" GridPane.columnIndex="2"/>
<Label fx:id="dftTestPercentage" text="0%" style="-fx-font-weight: bold;"
    GridPane.rowIndex="7" GridPane.columnIndex="3"/>

<Label text="Cumulative Sums Test"
    GridPane.rowIndex="8" GridPane.columnIndex="0"/>
<Label fx:id="cusumsTestResult" text="N/A"
    GridPane.rowIndex="8" GridPane.columnIndex="1"/>
<ProgressBar fx:id="cusumsTestBar" progress="0" maxWidth="Infinity"
    GridPane.rowIndex="8" GridPane.columnIndex="2"/>
<Label fx:id="cusumsTestPercentage" text="0%" style="-fx-font-weight: bold;"
    GridPane.rowIndex="8" GridPane.columnIndex="3"/>
</GridPane>
</VBox>
</TitledPane>

<!-- Chaotic Properties -->
<TitledPane text="Chaotic Properties" expanded="true">
    <VBox spacing="10">
        <GridPane hgap="10" vgap="10">
            <columnConstraints>
                <ColumnConstraints hgrow="SOMETIMES" minWidth="200"/>
                <ColumnConstraints hgrow="SOMETIMES"/>
            </columnConstraints>

            <Label text="Property" style="-fx-font-weight: bold;"
                GridPane.rowIndex="0" GridPane.columnIndex="0"/>
            <Label text="Value" style="-fx-font-weight: bold;"
                GridPane.rowIndex="0" GridPane.columnIndex="1"/>

            <Label text="Lyapunov Exponent"
                GridPane.rowIndex="1" GridPane.columnIndex="0"/>
            <Label fx:id="lyapunovLabel" text="N/A"
                GridPane.rowIndex="1" GridPane.columnIndex="1"/>

            <Label text="Correlation Dimension"
                GridPane.rowIndex="2" GridPane.columnIndex="0"/>
            <Label fx:id="correlationLabel" text="N/A"
                GridPane.rowIndex="2" GridPane.columnIndex="1"/>

            <Label text="Recurrence Plot Analysis"

```

```

        GridPane.rowIndex="3" GridPane.columnIndex="0"/>
        <Label fx:id="recurrenceLabel" text="N/A"
        GridPane.rowIndex="3" GridPane.columnIndex="1"/>
    </GridPane>
</VBox>
</TitledPane>

<!-- Visualization -->
<TitledPane text="Visualization" expanded="true">
    <VBox spacing="5">
        <Label text="Distribution Histogram"/>
        <Pane fx:id="histogramPane" prefHeight="200" style="-fx-border-color: lightgray;"/>
    </VBox>
</TitledPane>
<!-- Export Button -->
<HBox alignment="CENTER_RIGHT">
    <Button text="Export Analysis Report" onAction="#onExportReport" styleClass="secondary"/>
</HBox>
</VBox>
</VBox>

```

```

=====
=====
FILE: ./src/main/resources/ua/edu/chnu/imprrandgen/generation-view.fxml
=====
=====

```

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<VBox spacing="20" style="-fx-padding: 20;" styleClass="card" xmlns:fx="http://javafx.com/fxml"
    fx:controller="ua.edu.chnu.imprrandgen.controller.GenerationController">
    <Label text="Sequence Generation" styleClass="section-header"/>

    <VBox spacing="20" VBox.vgrow="ALWAYS">
        <!-- Generation Controls -->
        <TitledPane text="Generation Controls" expanded="true">
            <VBox spacing="10">
                <!-- Sequence Length Selection -->
                <HBox spacing="10" alignment="CENTER_LEFT">
                    <Label text="Sequence Length:"/>
                    <Spinner fx:id="generatedSequenceLengthSpinner" prefWidth="300"
                        editable="true">
                        <valueFactory>
                            <SpinnerValueFactory.IntegerSpinnerValueFactory
                                min="100" max="10000" initialValue="1000" amountToStepBy="100"/>

```

```

        </valueFactory>
    </Spinner>
    <Label text="values"/>
</HBox>

<!-- Standard Generation Controls -->
<HBox spacing="10" alignment="CENTER_LEFT" fx:id="standardGenerationControls">
    <Button fx:id="generateButton" text="Generate Sequence" onAction="#onGenerateSequence"
styleClass="primary-action"/>
    <ProgressIndicator fx:id="generationProgress" visible="false"/>
    <Label fx:id="statusLabel" text="Ready to generate" styleClass="status-indicator, status-success"/>
</HBox>

<!-- Real-time Generation Controls -->
<VBox spacing="10" fx:id="realtimeGenerationControls" visible="false" managed="false">
    <HBox spacing="10" alignment="CENTER_LEFT">
        <Button fx:id="startButton" text="Start Generation" onAction="#onStartRealtimeGeneration"
styleClass="primary-action"/>
        <Button fx:id="resetButton" text="Reset" onAction="#onResetRealtimeGeneration" disable="true"
styleClass="secondary"/>
    </HBox>
    <HBox spacing="10" alignment="CENTER_LEFT">
        <Label text="Generated Values:"/>
        <Label fx:id="valueCounterLabel" text="0" styleClass="status-indicator"/>
    </HBox>
</VBox>

<Label fx:id="configInfoLabel" text="Configuration: Bits (0-1)"
styleClass="info-label"
wrapText="true" maxWidth="Infinity" VBox.vgrow="NEVER"/>
</VBox>
</TitledPane>

<!-- Generated Sequence Display -->
<TitledPane text="Generated Sequence" expanded="true">
    <VBox spacing="10">
        <TextArea fx:id="sequenceDisplay" prefHeight="200" editable="false"
wrapText="true"/>
        <HBox spacing="10" alignment="CENTER_RIGHT">
            <Button text="Copy" onAction="#onCopySequence" styleClass="secondary"/>
            <Button text="Save to File" onAction="#onSaveSequence" styleClass="secondary"/>
            <Button text="Save to Sequences" onAction="#onSaveToSequences" styleClass="secondary"/>
        </HBox>
    </VBox>
</TitledPane>

<!-- Basic Analysis -->
<TitledPane text="Basic Analysis" expanded="true">

```

```

<GridPane hgap="10" vgap="10">
  <columnConstraints>
    <ColumnConstraints hgrow="SOMETIMES" minWidth="150"/>
    <ColumnConstraints hgrow="SOMETIMES"/>
  </columnConstraints>

  <Label text="Sequence Length:" GridPane.rowIndex="0" GridPane.columnIndex="0"/>
  <Label fx:id="lengthLabel" text="0" GridPane.rowIndex="0" GridPane.columnIndex="1"/>

  <Label text="Entropy (bits):" GridPane.rowIndex="1" GridPane.columnIndex="0"/>
  <Label fx:id="entropyLabel" text="0.0" GridPane.rowIndex="1" GridPane.columnIndex="1"/>

  <Label text="Distribution:" GridPane.rowIndex="2" GridPane.columnIndex="0"/>
  <Label fx:id="distributionLabel" text="N/A" GridPane.rowIndex="2" GridPane.columnIndex="1"/>
</GridPane>
</TitledPane>
<!-- Navigation Button -->
<HBox alignment="CENTER_RIGHT">
  <Button text="Next: Detailed Analysis" onAction="#onNextToAnalysis"
    disable="true" fx:id="analysisButton" styleClass="primary-action"/>
</HBox>
</VBox>
</VBox>

```

```

=====
=====
FILE: ./src/main/resources/ua/edu/chnu/imprrandgen/main-view.fxml
=====
=====

```

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<TabPane fx:id="mainTabPane" xmlns:fx="http://javafx.com/fxml"
  fx:controller="ua.edu.chnu.imprrandgen.controller.MainController"
  tabClosingPolicy="UNAVAILABLE"
  stylesheets="@modern-styles.css">

  <!-- Settings Tab -->
  <Tab text="Settings">
    <fx:include source="settings-view.fxml"/>
  </Tab>

  <!-- Generation Tab -->
  <Tab text="Generation" fx:id="generationTab">
    <fx:include source="generation-view.fxml"/>
  </Tab>

```

```

</Tab>

<!-- Analysis Tab -->
<Tab text="Analysis" fx:id="analysisTab">
    <fx:include source="analysis-view.fxml"/>
</Tab>

<!-- Sequences Tab -->
<Tab text="Sequences" fx:id="sequencesTab">
    <fx:include source="sequences-view.fxml"/>
</Tab>
</TabPane>

=====
=====
FILE: ./src/main/resources/ua/edu/chnu/imprrandgen/modern-styles.css
=====
=====

/* Modern UI Styles for Random Generator Application */

/* Root styles and variables */
.root {
    -fx-font-family: 'Segoe UI', Arial, sans-serif;
    -fx-font-size: 24px;
    -fx-background-color: #f5f5f7;

    /* Color palette */
    -fx-primary: #4a6da7;
    -fx-primary-light: #6b8ec9;
    -fx-primary-dark: #2d4b7a;
    -fx-accent: #5c946e;
    -fx-secondary: #f8f9fa;
    -fx-light-gray: #e9ecef;
    -fx-mid-gray: #ced4da;
    -fx-dark-gray: #6c757d;
    -fx-text-dark: #343a40;
    -fx-text-light: #f8f9fa;
    -fx-error: #dc3545;
    -fx-warning: #ffc107;
    -fx-success: #28a745;
}

/* TabPane styling */
.tab-pane {
    -fx-tab-min-width: 100px;
    -fx-tab-max-width: 200px;

```

```

    -fx-tab-min-height: 40px;
    -fx-font-size: 24px;
}

.tab {
    -fx-background-color: #f8f9fa;
    -fx-background-radius: 5 5 0 0;
    -fx-padding: 5 15;
    -fx-font-size: 24px;
}

.tab:selected {
    -fx-background-color: #4a6da7;
}

.tab:selected .tab-label {
    -fx-text-fill: #f8f9fa;
    -fx-font-weight: bold;
    -fx-font-size: 24px;
}

.tab-pane > .tab-header-area > .headers-region > .tab > .tab-container > .tab-label {
    -fx-alignment: CENTER;
    -fx-text-fill: #343a40;
    -fx-font-size: 24px;
}

.tab-pane > .tab-header-area > .tab-header-background {
    -fx-background-color: #e9ecef;
}

/* TitledPane styling */
.titled-pane {
    -fx-text-fill: #343a40;
    -fx-font-weight: bold;
    -fx-font-size: 24px;
}

.titled-pane > .title {
    -fx-background-color: #f8f9fa;
    -fx-border-color: #ced4da;
    -fx-border-radius: 5;
    -fx-background-radius: 5;
    -fx-font-size: 24px;
}

.titled-pane > .title:hover {
    -fx-background-color: derive(#f8f9fa, -5%);
}

```

```

}

.titled-pane > .content {
  -fx-background-color: #f8f9fa;
  -fx-border-color: #ced4da;
  -fx-border-radius: 0 0 5 5;
  -fx-background-radius: 0 0 5 5;
  -fx-padding: 10;
  -fx-font-size: 24px;
}

/* Button styling */
.button {
  -fx-background-color: #4a6da7;
  -fx-text-fill: #f8f9fa;
  -fx-background-radius: 5;
  -fx-padding: 8 15;
  -fx-cursor: hand;
  -fx-effect: dropshadow(gaussian, rgba(0, 0, 0, 0.2), 2, 0, 0, 1);
}

/* Primary action button */
.button.primary-action {
  -fx-font-weight: bold;
  -fx-padding: 10 20;
  -fx-font-size: 26px;
  -fx-effect: dropshadow(gaussian, rgba(0, 0, 0, 0.3), 3, 0, 0, 1);
}

.button:hover {
  -fx-background-color: #6b8ec9;
}

.button:pressed {
  -fx-background-color: #2d4b7a;
  -fx-effect: dropshadow(gaussian, rgba(0, 0, 0, 0.3), 1, 0, 0, 1);
}

/* Secondary button style */
.button.secondary {
  -fx-background-color: #f8f9fa;
  -fx-text-fill: #343a40;
  -fx-border-color: #ced4da;
}

.button.secondary:hover {
  -fx-background-color: derive(#f8f9fa, -5%);
}

```

```

/* Text input styling */
.text-field, .text-area, .spinner, .combo-box {
    -fx-background-color: white;
    -fx-background-radius: 5;
    -fx-border-color: #ced4da;
    -fx-border-radius: 5;
    -fx-padding: 5 10;
    -fx-font-size: 24px;
}

.text-field:focus, .text-area:focus, .spinner:focus, .combo-box:focus {
    -fx-border-color: #4a6da7;
    -fx-effect: dropshadow(gaussian, rgba(74, 109, 167, 0.3), 3, 0, 0, 0);
}

/* Label styling */
.label {
    -fx-font-size: 24px;
}

/* ComboBox items */
.combo-box-popup .list-view .list-cell {
    -fx-font-size: 24px;
    -fx-padding: 5 10;
}

/* Slider styling */
.slider {
    -fx-font-size: 24px;
}

.slider .track {
    -fx-background-color: #e9ecef;
    -fx-background-radius: 5;
}

.slider .thumb {
    -fx-background-color: #4a6da7;
    -fx-background-radius: 10;
    -fx-effect: dropshadow(gaussian, rgba(0, 0, 0, 0.2), 2, 0, 0, 1);
}

.slider .thumb:hover {
    -fx-background-color: #6b8ec9;
}

/* Radio button and checkbox styling */

```

```
.radio-button, .check-box {
    -fx-font-size: 24px;
}
```

```
.radio-button .radio, .check-box .box {
    -fx-background-color: white;
    -fx-border-color: #ced4da;
```

```
=====
FILE: ./src/main/resources/ua/edu/chnu/imprrandgen/sequences-view.fxml
=====
```

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<?import javafx.scene.control.*?>
```

```
<?import javafx.scene.layout.*?>
```

```
<ScrollPane fitToWidth="true" vbarPolicy="AS_NEEDED" hbarPolicy="NEVER" xmlns:fx="http://javafx.com/fxml"
    fx:controller="ua.edu.chnu.imprrandgen.controller.SequencesController">
```

```
<VBox spacing="20" style="-fx-padding: 20;" styleClass="card">
```

```
<Label text="Generated Sequences History" styleClass="section-header"/>
```

```
<VBox spacing="20" VBox.vgrow="ALWAYS">
```

```
<!-- Sequences List -->
```

```
<TitledPane text="Sequence List" expanded="true">
```

```
<VBox spacing="10">
```

```
<TableView fx:id="sequencesTableView" VBox.vgrow="ALWAYS" prefHeight="300">
```

```
<columns>
```

```
<TableColumn fx:id="timestampColumn" text="Timestamp" prefWidth="150"/>
```

```
<TableColumn fx:id="rangeTypeColumn" text="Range Type" prefWidth="120"/>
```

```
<TableColumn fx:id="securityLevelColumn" text="Security Level" prefWidth="120"/>
```

```
<TableColumn fx:id="lengthColumn" text="Length" prefWidth="80"/>
```

```
<TableColumn fx:id="entropyColumn" text="Entropy" prefWidth="80"/>
```

```
</columns>
```

```
</TableView>
```

```
<!-- Sequence Details -->
```

```
<TitledPane text="Sequence Details" expanded="true">
```

```
<GridPane hgap="10" vgap="10">
```

```
<columnConstraints>
```

```
<ColumnConstraints hgrow="SOMETIMES" minWidth="150"/>
```

```
<ColumnConstraints hgrow="SOMETIMES"/>
```

```
</columnConstraints>
```

```
<Label text="Timestamp:" GridPane.rowIndex="0" GridPane.columnIndex="0"/>
```

```
<Label fx:id="detailsTimestampLabel" text="" GridPane.rowIndex="0"
```

```
GridPane.columnIndex="1"/>
```

```

        <Label text="Range Type:" GridPane.rowIndex="1" GridPane.columnIndex="0"/>
        <Label fx:id="detailsRangeTypeLabel" text="" GridPane.rowIndex="1"
GridPane.columnIndex="1"/>

        <Label text="Security Level:" GridPane.rowIndex="2" GridPane.columnIndex="0"/>
        <Label fx:id="detailsSecurityLevelLabel" text="" GridPane.rowIndex="2"
GridPane.columnIndex="1"/>

        <Label text="Length:" GridPane.rowIndex="3" GridPane.columnIndex="0"/>
        <Label fx:id="detailsLengthLabel" text="" GridPane.rowIndex="3" GridPane.columnIndex="1"/>

        <Label text="Entropy:" GridPane.rowIndex="4" GridPane.columnIndex="0"/>
        <Label fx:id="detailsEntropyLabel" text="" GridPane.rowIndex="4" GridPane.columnIndex="1"/>

        <Label text="Distribution:" GridPane.rowIndex="5" GridPane.columnIndex="0"/>
        <Label fx:id="detailsDistributionLabel" text="" GridPane.rowIndex="5"
GridPane.columnIndex="1"/>
    </GridPane>
</TitledPane>
</VBox>
</TitledPane>
</VBox>

<!-- Action Buttons -->
<HBox spacing="10" alignment="CENTER_RIGHT">
    <Button text="Load Selected Sequence" onAction="#onLoadSelectedSequence" styleClass="secondary"/>
    <Button text="View Analysis" onAction="#onViewSequenceAnalysis" styleClass="secondary"/>
    <Button text="Delete Selected" onAction="#onDeleteSelectedSequence" styleClass="secondary"/>
</HBox>
</VBox>
</ScrollPane>

=====
=====
FILE: ./src/main/resources/ua/edu/chnu/imprrandgen/settings-view.fxml
=====
=====

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.collections.*?>
<?import java.lang.String?>

<ScrollPane fitToWidth="true" vbarPolicy="AS_NEEDED" hbarPolicy="NEVER" xmlns:fx="http://javafx.com/fxml"
fx:controller="ua.edu.chnu.imprrandgen.controller.SettingsController">

```

```

<VBox spacing="20" style="-fx-padding: 20;" styleClass="card">
  <Label text="Generation Settings" styleClass="section-header"/>

  <VBox spacing="20" VBox.vgrow="ALWAYS">
    <!-- Security vs Speed Slider -->
    <TitledPane text="Security Level vs. Generation Speed" expanded="true">
      <VBox spacing="10">
        <HBox spacing="10" alignment="CENTER_LEFT">
          <Label text="Faster"/>
          <Slider fx:id="securitySlider" min="1" max="10" value="5"
            showTickMarks="true" showTickLabels="true"
            majorTickUnit="1" minorTickCount="0"
            snapToTicks="true" HBox.hgrow="ALWAYS"/>
          <Label text="More Secure"/>
        </HBox>
      </VBox>
    </TitledPane>

    <!-- Security Level Control Settings -->
    <TitledPane text="Security Level Control Settings" expanded="false">
      <VBox spacing="10">
        <Label text="Automatically adjust security level based on:"/>

        <!-- Measurement Length Setting -->
        <HBox spacing="10" alignment="CENTER_LEFT">
          <CheckBox fx:id="lengthBasedSecurityCheck" text="Measurement Length:"
            onAction="#onSecurityControlChanged"/>
          <Spinner fx:id="sequenceLengthSpinner" prefWidth="300"
            editable="true" disable="true">
            <valueFactory>
              <SpinnerValueFactory.IntegerSpinnerValueFactory
                min="1000" max="10000" initialValue="2000" amountToStepBy="500"/>
            </valueFactory>
          </Spinner>
          <Label text="characters"/>
        </HBox>

        <!-- Rounds of Generation Setting -->
        <HBox spacing="10" alignment="CENTER_LEFT">
          <CheckBox fx:id="roundsBasedSecurityCheck" text="Rounds of Generation:"
            onAction="#onSecurityControlChanged"/>
          <Spinner fx:id="roundsSpinner" prefWidth="300"
            editable="true" disable="true">
            <valueFactory>
              <SpinnerValueFactory.IntegerSpinnerValueFactory
                min="1" max="20" initialValue="5" amountToStepBy="1"/>
            </valueFactory>
          </Spinner>
        </HBox>
      </VBox>
    </TitledPane>
  </VBox>
</VBox>

```

```

        <Label text="rounds"/>
    </HBox>

    <!-- Generation Time Setting -->
    <HBox spacing="10" alignment="CENTER_LEFT">
        <CheckBox fx:id="timeBasedSecurityCheck" text="Measurement Time:"
            onAction="#onSecurityControlChanged"/>
        <Spinner fx:id="generationTimeSpinner" prefWidth="300"
            editable="true" disable="true">
            <valueFactory>
                <SpinnerValueFactory.IntegerSpinnerValueFactory
                    min="1" max="30" initialValue="5" amountToStepBy="1"/>
            </valueFactory>
        </Spinner>
        <Label text="seconds"/>
    </HBox>

    <Button text="Apply Settings" onAction="#onApplySecuritySettings" styleClass="secondary"/>
</VBox>
</TitledPane>

<!-- Value Range Configuration -->
<TitledPane text="Value Range Configuration" expanded="true">
    <VBox spacing="10">
        <fx:define>
            <ToggleGroup fx:id="rangeGroup"/>
        </fx:define>

        <RadioButton fx:id="bitsRangeRadio" text="Bits (0-1)"
            toggleGroup="$rangeGroup" selected="true"
            onAction="#onRangeTypeChanged"/>
        <RadioButton fx:id="defaultRangeRadio" text="Default Range (1-256)"
            toggleGroup="$rangeGroup"
            onAction="#onRangeTypeChanged"/>
        <RadioButton fx:id="customRangeRadio" text="Custom Range"
            toggleGroup="$rangeGroup"
            onAction="#onRangeTypeChanged"/>

        <!-- Custom Range Options -->
        <HBox fx:id="customRangeOptions" spacing="10" alignment="CENTER_LEFT"
            visible="false" managed="false">
            <Label text="Min:"/>
            <TextField fx:id="minValueField" prefWidth="80" text="1"/>
            <Label text="Max:"/>
            <TextField fx:id="maxValueField" prefWidth="80" text="100"/>
        </HBox>
    </VBox>
</TitledPane>

```

```

</TitledPane>

<!-- Data Source Settings -->
<TitledPane text="Data Source Settings" expanded="true">
  <VBox spacing="10">
    <!-- Source Selection -->
    <VBox spacing="10">
      <fx:define>
        <ToggleGroup fx:id="sourceGroup"/>
      </fx:define>

      <Label text="Data Source"/>
      <RadioButton fx:id="fileSourceRadio" text="From File (Past Measurements)"
        toggleGroup="$sourceGroup" selected="true"
        onAction="#onSourceChanged"/>
      <RadioButton fx:id="realtimeSourceRadio" text="Realtime Measurements"
        toggleGroup="$sourceGroup"
        onAction="#onSourceChanged"/>
    </VBox>

    <!-- File Source Options -->
    <VBox fx:id="fileSourceOptions" spacing="10">
      <Label text="Measurement File"/>
      <HBox spacing="10">
        <TextField fx:id="filePathField" promptText="Select measurement file"
          HBox.hgrow="ALWAYS"/>
        <Button text="Browse..." onAction="#onBrowseFile" styleClass="secondary"/>
      </HBox>
    </VBox>

    <!-- Realtime Source Options -->
    <VBox fx:id="realtimeSourceOptions" spacing="10" visible="false" managed="false">
      <Label text="Microcontroller Settings"/>
      <HBox spacing="10" alignment="CENTER_LEFT">
        <Label text="Microcontroller Type:"/>
        <ComboBox fx:id="microcontrollerCombo" prefWidth="200">
          <items>
            <FXCollections fx:factory="observableArrayList">
              <String fx:value="Arduino Uno"/>
              <String fx:value="Arduino Mega"/>
              <String fx:value="ESP32"/>
              <String fx:value="ESP8266"/>
              <String fx:value="Raspberry Pi Pico"/>
              <String fx:value="STM32"/>
            </FXCollections>
          </items>
        </ComboBox>
      </HBox>
    </VBox>
  </VBox>
</TitledPane>

```

```

<HBox spacing="10" alignment="CENTER_LEFT">
  <Label text="Communication Port:"/>
  <ComboBox fx:id="portCombo" prefWidth="200"/>
  <Button text="Refresh" onAction="#onRefreshPorts" styleClass="secondary"/>
</HBox>
<HBox spacing="10" alignment="CENTER_LEFT">
  <Label text="Baud Rate:"/>
  <ComboBox fx:id="baudRateCombo" prefWidth="200">
    <items>
      <FXCollections fx:factory="observableArrayList">
        <String fx:value="9600"/>
        <String fx:value="19200"/>
        <String fx:value="38400"/>
        <String fx:value="57600"/>
        <String fx:value="115200"/>
      </FXCollections>
    </items>
  </ComboBox>
</HBox>
</VBox>
</VBox>
</TitledPane>
</VBox>

<!-- Navigation Button -->
<HBox alignment="CENTER_RIGHT">
  <Button text="Next: Generate Sequence" onAction="#onNextToGenerate" styleClass="primary-action"/>
</HBox>
</VBox>
</ScrollPane>

```

Акти впровадження результатів дисертаційної роботи



ТОВАРИСТВО З ДОДАТКОВОЮ ВІДПОВІДАЛЬНІСТЮ

"ЗАВОД "КВАРЦ"

58032 Україна, м. Чернівці, вул.Головна, 246 ☎/факс (0372) 586100, 586114

E-mail: kvd@quartz.cv.ua

Код ЄДРПОУ 1431467

№ 42/205

Від 26 лютого 2026 р.

Акт

впровадження результатів дисертаційної роботи
Дячука Ростислава Любомировича
на тему «Розробка та дослідження інформаційної системи генерування
високоентропійної послідовності випадкових чисел»,
представленої до захисту на здобуття наукового ступеня
доктора філософії з галузі знань 12 – «Інформаційні технології»
за спеціальністю 121 – «Інженерія програмного забезпечення»

Цей акт складено на підтвердження того, що ТДВ «ЗАВАД «КВАРЦ» використовує у своїй діяльності результати, які отримано в дисертаційному дослідженні здобувача наукового ступеня доктора філософії Чернівецького національного університету імені Юрія Федьковича Дячука Ростислава Любомировича, а саме:

- інформаційну систему екстракції послідовності випадкових чисел з низки стохастичних фізичних явищ – інтенсивності пікселів веб-камери, фото- та темнового струму фотодіода, а також обробки отриманих послідовностей за допомогою технології клітинних автоматів для генерації вхідного вектора хеш-функцій, стеганографічних алгоритмів та алгоритмів генерації криптографічних ключів.

Використання вище зазначеної інформаційної системи дозволило покращити криптозахист програмного забезпечення, що використовується на нашому підприємстві.

Заступник генерального директора
Директор ЕМК
ТДВ «ЗАВОД «КВАРЦ»



Ігор РОМАНЮК

МПП ВКФ «КАСКАД ДЕВЕЛОПЕРС ГРОУП»

58000, Україна, м.Чернівці, вул.Університетська 14/7, тел.380505194858, факс 380372573975
ЄДРПОУ 35096461, ІПН №350964624126, № Свідотства ПДВ 100062138
IBAN:UA163808050000000026002640574 «Райффайзен банк Аваль» м. Київ E-mail:info@kdg.com.ua

«05» березня 2026 р.
вих. № 05/02-01

Акт

про впровадження результатів дисертаційної роботи
Дячука Ростислава Любомировича

РОЗРОБКА ТА ДОСЛІДЖЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ГЕНЕРУВАННЯ ВИСОКОЕНТРОПІЙНОЇ ПОСЛІДОВНОСТІ ВИПАДКОВИХ ЧИСЕЛ

представленої до захисту на здобуття наукового ступеня
доктора філософії з галузі знань 12 – «Інформаційні технології»
за спеціальністю 121 – «Інженерія програмного забезпечення»

Акт складено у зв'язку з висвітленням результатів впровадження у МПП ВКФ «КАСКАД ДЕВЕЛОПЕРС ГРОУП» результатів, які отримано в дисертаційному дослідженні здобувача наукового ступеня доктора філософії Чернівецького національного університету імені Юрія Федьковича Дячука Ростислава Любомировича, а саме технологію створення неперервного потоку випадкових чисел з інтенсивності пікселів веб-камери мобільного телефона.

Зазначена технологія наразі проходить випробовування у системах захисту каналів зв'язку, що розробляються нашим підприємством.

Директор

МПП ВКФ «Каскад Девелоперс
Гроуп»



Чалапчий Ю. Ю.

Товариство з обмеженою відповідальністю
«ДАТАВІЗ УКРАЇНА»
ЄДРПОУ: 40425206
58000, Чернівецька обл., м. Чернівці, вул. Рівненська, буд. 10, кв. 40
IBAN UA113510050000026007879162922 в АТ "УКРСИББАНК",
МФО: 351005

м. Чернівці

16 лютого 2026 року

Акт

про впровадження результатів дисертаційної роботи
Дячука Ростислава Любомировича
«Розробка та дослідження інформаційної системи генерування
високоентропійної послідовності випадкових чисел»
представленої до захисту на здобуття наукового ступеня
доктора філософії з галузі знань 12 – «Інформаційні технології»
за спеціальністю 121 – «Інженерія програмного забезпечення»

Акт складено у зв'язку з висвітленням результатів впровадження у ТОВ «ДАТАВІЗ Україна» результатів, які отримано в дисертаційному дослідженні здобувача наукового ступеня доктора філософії Чернівецького національного університету імені Юрія Федьковича Дячука Ростислава Любомировича, а саме технологію екстракції послідовності випадкових чисел з інтенсивності пікселів веб-камери, фото- та темного струму фотодіода, оброблених за технологією клітинних автоматів.

Зазначена технологія наданий час випробовується для подальшого застосування у програмних продуктах, що розробляються нашим підприємством.

Директор ТОВ «ДАТАВІЗ Україна»



Філіпська А.М.

«ЗАТВЕРДЖУЮ»

Директор Навчально-наукового
Інституту фізико-технічних та
комп'ютерних наук Чернівецького
національного університету
імені Юрія Федьковича



П.М. Шпатар

«24» Відомого 2026 р.

Акт

впровадження результатів дисертаційної роботи
Дячука Ростислава Любомировича
«Розробка та дослідження інформаційної системи генерування
високоентропійної послідовності випадкових чисел»
представленої до захисту на здобуття наукового ступеня
доктора філософії з галузі знань 12 – «Інформаційні технології»
за спеціальністю 121 – «Інженерія програмного забезпечення»
в навчальному процесі

Комісія Чернівецького національного університету імені Юрія Федьковича в складі:

заступник директора ННІФТКН з наукової роботи, д.ф-м.н., професор
Дуболазов О.В.;

завідувач кафедри радіотехніки та інформаційної безпеки, д.т.н., професор
Саміла А.П.;

завідувач кафедри програмного забезпечення комп'ютерних систем, Ph.D, доцент,
Газдюк К.П..

склали цей акт про те, що результати дисертаційної роботи здобувача наукового ступеня доктора філософії за спеціальністю «Інженерія програмного забезпечення» Дячука Р.Л. впроваджені у навчальний процес кафедр радіотехніки та інформаційної безпеки та програмного забезпечення комп'ютерних систем, зокрема в дисциплінах «Криптографічний захист інформації», «Безпека в цифровому просторі», «Безпека програм та даних».

При викладанні зазначених дисциплін використовуються наступні матеріали досліджень, які отримані у дисертаційній роботі:

- особливості існуючих програмних та апаратних засобів та методів генерації високоентропійних послідовностей випадкових чисел;

- розробка технології генерації послідовностей випадкових чисел з пікселів фоточутливої матриці веб-камери, з фотоструму фотодіода, з темного струму фотодіода;

- застосування технології гібридної генерації послідовностей випадкових чисел із застосуванням клітинних автоматів;

- використання розробленої інформаційної системи для генерації послідовностей випадкових чисел на основі створених генераторів;
- вивчення технології обрання балансу між швидкістю генерації послідовностей випадкових чисел та її надійністю;
- тестування послідовностей випадкових чисел на відповідність критеріям хаотичності;
- проведення порівняльного аналізу характеристик генераторів послідовностей випадкових чисел.

Використання зазначених результатів дозволило підвищити якість навчального процесу із згаданих дисциплін.

Заступник директора ННІФТКН з
наукової роботи, д.ф-м.н., професор

Завідувач кафедри радіотехніки
та інформаційної безпеки д.т.н., професор

Завідувач кафедри програмного
забезпечення комп'ютерних систем
Ph.D, доцент

О.В. Дуболазов

А.П. Саміла

К.П. Газдюк

Список публікацій за темою дисертації
Наукові праці, в яких опубліковані основні наукові результати
дисертації

*Наукові праці у періодичних наукових виданнях,
проіндексованих у наукометричних базах даних Scopus:*

1. Diachuk R., Dmytrashchuk K., Mazurets A., Prokhorov H., Yanushevskyi S. Photosensitive matrix as a source of entropy. *Proceedings of SPIE*. 13813, Seventeenth International Conference on Correlation Optics. 2025. 138132C. ISSN:0277-786X (Scopus) DOI: 10.1117/12.3092587 (Внесок авторів: *Diachuk R.* – програмна реалізація збору та обробки даних, експериментальні дослідження ентропійних характеристик, аналіз результатів, підготовка початкового варіанту статті; *Dmytrashchuk K.* – апаратна реалізація та налаштування фоточутливої матриці, збір експериментальних даних; *Mazurets A.* – аналіз літературних джерел, статистична обробка результатів; *Prokhorov H.* – методологія дослідження, формалізація моделі джерела ентропії; *Yanushevskyi S.* – концептуалізація, наукове керівництво, рецензування та редагування рукопису.)

*Наукові праці у виданнях, включених до переліку
наукових фахових видань України:*

2. Diachuk R., Dobrovolsky Y., Hanzhelo D., Prokhorov H., Trembach D. Research the Level of Chaotic and Reliability in Webcam-generated Random Number Sequences, *Security of infocommunication systems and Internet of things*. 2024. Vol. 2, №. 1. P. 01004. DOI: sisiot2024.1.01004 (Внесок авторів: *Dobrovolsky*

- Y. – концептуалізація; Prokhorov H. – методологія та дослідження, Diachuk R. – програмна розробка, та експерименти; Hanzhelo D. – статистичні дослідження; Trembach D. – концептуалізація та написання).*
3. Добровольський, Ю., Дячук, Р. Дослідження зворотного струму фотодіода для генерації надійної випадкової послідовності чисел. *Вісник Хмельницького національного університету. Серія: Технічні науки..* 2025. Том 357, № 5.1. С. 126-132. DOI: 10.31891/2307-5732-2025-357-15 (Внесок авторів: Добровольський Ю. – концептуалізація, методологія, аналіз результатів, редагування; Дячук Р. – аналіз джерел, програмна розробка, дослідження результатів та написання).
 4. Prokhorov H., Hanzhelo M., Diachuk R., Yanushevskiy S. Investigation of statistical characteristics of random number sequences generated by a webcam using cellular automata functionality and NIST patterns. *Security of infocommunication systems and Internet of things.* 2025. Vol 3, № 1. P. 01007. DOI: 10.31861/sisiot2025.1.01007 (Внесок авторів: Prokhorov H. – концептуалізація, дослідження, нагляд; Hanzhelo M., Diachuk R. – програмне забезпечення, ресурси, підготовка початкового варіанту рукопису, візуалізація, валідація; Yanushevskiy S.– методологія, концептуалізація, рецензування та редагування рукопису, нагляд, дослідження).
 5. Добровольський Ю., Дячук Р. Дослідження генерації випадкової послідовності чисел на основі темнового струм фотодіода при різних температурах. *Вчені записки Таврійського національного університету імені В.І. Вернадського. Серія: Технічні науки.* 2025. Том 36 (75), № 4. Частина 2. С. 119-126. DOI: 10.32782/2663-5941/2025.4.2/16 (Внесок авторів: Добровольський Ю. – концептуалізація, методологія, написання; Дячук Р. – аналіз джерел, програмна розробка, аналіз результатів та редагування).

Наукові праці, які засвідчують апробацію матеріалів дисертації:

6. Дячук Р.Л., Павлюченко О.С., Прохоров П.А., Прохоров Г.В., Добровольський Ю.Г. Клітинні автомати як генератори Хаосу в криптографічних алгоритмах. Проблеми інформатики та комп'ютерної техніки : матеріали XI міжнар. наук.-практ. конф., м. Чернівці, 10–13 лист. 2022 р. Черн. нац. ун-т ім. Ю Федьковича, Чернівці, 2022. – С. 44-46. *(Внесок авторів: Дячук Р.Л. – концептуалізація, програмна реалізація моделі, експериментальні дослідження, написання рукопису; Павлюченко О.С. – аналіз літератури, дослідження параметрів клітинних автоматів; Прохоров П.А., Прохоров Г.В. – методологія, формалізація моделі, валідація результатів; Добровольський Ю.Г. – наукове керівництво, узагальнення результатів, редагування).*
7. Prokhorov G., Dobrovolsky Y., Dyachuk R. Hash-Function Algorithms Balanced on Reliability and Data Processing Speed. IEEE 4th International Conference on Advanced Trends in Information Theory, 15-16 December, 2022, Kyiv, UKRAINE. P.111-114. (Scopus). DOI: 10.1109/ATIT58178.2022.10024181 *(Внесок авторів: Prokhorov G. – концептуалізація, методологія, постановка задачі; Dobrovolsky Y. – аналіз надійності, редагування, наукове керівництво; Dyachuk R. – програмна реалізація алгоритмів, експериментальні дослідження швидкодії, візуалізація результатів, підготовка початкового варіанту статті).*
8. Дячук Р.Л., Добровольський Ю.Г. Алгоритм хеш-функції з підвищеною криптостійкістю. Покращення лавинного ефекту. Інформаційні технології: наука, техніка, технологія, освіта, здоров'я: тези доповідей XXXI міжнар. наук.-практ. конф., MicroCAD-2023, 17-20 травня 2023. Харків: НТУ «ХПІ». С. 1113. *(Внесок авторів: Дячук Р.Л. – розробка алгоритму, програмна реалізація, експериментальна перевірка лавинного ефекту, написання*

- рукопису; Добровольський Ю.Г. – концептуалізація, аналіз криптостійкості, редагування, наукове керівництво).*
9. Дячук Р.Л., Комісарчук В.В., Прохоров П.А., Прохоров Г.В., Добровольський Ю.Г. Вдосконалення функції стискання хеш-функції з застосуванням клітинних автоматів. Проблеми інформатики та комп'ютерної техніки: матеріали XII міжнар. наук.-практ. конф., м. Чернівці, 10–12 лист. 2023. Чернівці: Черн. нац. ун-т, 2023. - С. 82-83. *(Внесок авторів: Дячук Р.Л. – програмна модифікація функції стискання, експериментальні дослідження; Комісарчук В.В. – аналіз літератури, підготовка теоретичного обґрунтування; Прохоров П.А., Прохоров Г.В. – методологія, математичне моделювання, валідація; Добровольський Ю.Г. – концептуалізація, узагальнення результатів, редагування).*
10. Добровольський Ю.Г., Прохоров Г.В., Янушевський С.В., Прохоров П.А., Дячук Р.Л., Трембач Д. В. Оцінка надійності програмного забезпечення з точки зору його довговічності. Проблеми інформатики та комп'ютерної техніки: матеріали XII міжнар. наук.-практ. конф., м. Чернівці, 01–03 лист. 2024. Чернівці: Черн. нац. ун-т, С. 36-37. *(Внесок авторів: Добровольський Ю.Г. – концептуалізація, методологія оцінювання, редагування; Прохоров Г.В. – математична модель довговічності, аналіз результатів; Янушевський С.В. – статистична обробка даних, валідація; Прохоров П.А. – аналіз показників надійності; Дячук Р.Л. – програмна реалізація моделі, експериментальні дослідження, візуалізація; Трембач Д.В. – збір даних, підготовка матеріалів).*
11. Добровольський Ю.Г., Прохоров Г.В., Дячук Р.Л. Генерація послідовності випадкових чисел на основі темнового струму фотодіода. Current trends in scientific research development. Proceedings of the 13th International scientific and practical conference. BoScience Publisher. Boston, USA. 2025. P. 65-67. *(Внесок авторів: Добровольський Ю.Г. – концептуалізація, постановка задачі, редагування; Прохоров Г.В. – методологія експерименту, аналіз*

фізичних параметрів; Дячук Р.Л. – програмна обробка сигналу, реалізація генератора, статистичне тестування випадковості, написання рукопису).

12. Prokhorov G.V., Diachuk R.L., Hanzhelo M.G., Yanushevskyi S.V. New approaches to the design of a hybrid random sequence generator. Physical and technological problems of transmission, processing and storage of information in infocommunication systems: Proceedings of Xth International Scientific-Practical Conference. Chernivtsi : Yuriy Fedkovych Chernivtsi National University, 2025. P. 134-136. (*Внесок авторів: Prokhorov G.V. – концептуалізація, методологія гібридної моделі; Diachuk R.L. – програмна реалізація генератора, експериментальні дослідження, візуалізація результатів, підготовка початкового варіанту статті; Hanzhelo M.G. – аналіз існуючих підходів, валідація; Yanushevskyi S.V. – рецензування та редагування, наукове керівництво).*

Відомості про апробацію матеріалів дисертації

1. XI Міжнародна науково-практична конференція «Проблеми інформатики та комп'ютерної техніки», 10–13 листопада 2022 р, Чернівці, Україна (очна з публікацією тез)
2. IEEE 4th International Conference on Advanced Trends in Information Theory, 15-16 грудня, 2022, Київ, Україна (дистанційна з публікацією тез)
3. XXXI Міжнародна науково-практична конференція «MicroCAD-2023», 17–20 травня 2023 р., Харків, Україна (заочна з публікацією тез).
4. XII Міжнародна науково-практична конференція «Проблеми інформатики та комп'ютерної техніки», 10–12 листопада. 2023 р, Чернівці, Україна (очна з публікацією тез)
5. XIII Міжнародна науково-практична конференція «Проблеми інформатики та комп'ютерної техніки», 01–03 листопада. 2024 р, Чернівці, Україна (очна з публікацією тез)
6. XIII Міжнародна науково-практична конференція «Current Trends in Scientific Research Development», 1–3 серпня 2025, Boston, USA (заочна з публікацією тез)
7. X Міжнародна науково-практична конференція «Physical and Technological Problems of Transmission, Processing and Storage of Information in Infocommunication Systems», 15-17 травня 2025, Чернівці, Україна, 2025(очна з публікацією тез)