УДК 004.056.55:004.42

І. О. Бондаренко
У. О. Макогін

ОГЛЯД СУЧАСНИХ МЕТОДІВ ГЕНЕРАЦІЇ КРИПТОГРАФІЧНО СТІЙКИХ ВИПАДКОВИХ ЧИСЕЛ

Вінницький національний технічний університет

Анотація

У роботі розглянуто сучасні методи генерації криптографічно стійких випадкових чисел, необхідних для забезпечення безпеки криптографічних систем. Проаналізовано два основні класи генераторів, їхні обмеження та гібридний підхід, що застосовується в реальних системах. Описано джерела ентропії, механізм reseed, реалізації в операційних системах Linux та Windows, а також наведено рекомендації щодо перевірки якості генераторів.

Ключові слова: генератор випадкових чисел, криптографічна стійкість, ентропія, DRBG, TRNG.

Abstract

This work examines modern methods of generating cryptographically secure random numbers, which are necessary for ensuring the security of cryptographic systems. It analyzes the two main classes of generators, their limitations, and the hybrid approach used in real-world systems. The paper describes sources of entropy, the reseed mechanism, implementations in Linux and Windows operating systems, and also provides recommendations for testing the quality of generators.

Keywords: random number generator, cryptographic security, entropy, DRBG, TRNG.

Вступ

Надійність систем захисту інформації значною мірою залежить від якості випадкових значень, що використовуються під час генерації ключів, векторів ініціалізації, nonce та інших службових параметрів. Якщо ці дані можна передбачити частково чи повністю, математична стійкість алгоритму вже не гарантує захисту. З цієї причини генератор випадкових чисел (RNG) розглядається як окремий критичний компонент системи захисту інформації. Його криптографічна стійкість передбачає три вимоги: непередбачуваність, тобто неможливість визначити наступні біти на основі попередніх навіть за умови знання алгоритму; стійкість до компрометації внутрішнього стану, яка виключає відновлення інших частин послідовності при частковому розкритті стану; та forward secrecy, за якої компрометація стану в момент часу t не дозволяє відновити значення, згенеровані до цього моменту [1].

Результати дослідження

Ключовим поняттям у теорії генераторів випадкових чисел є ентропія. У стандарті NIST SP 800-90B вона визначається як міра непередбачуваності джерела даних. Для оцінювання використовується min-entropy, яка характеризує найгірший випадок передбачуваності значень. Низький рівень цього показника означає, що навіть стійкий RNG не може забезпечити належну безпеку [2].

У криптографії виділяють два основні класи генераторів випадкових чисел: TRNG та DRBG. TRNG (rue Random Number Generator) є фізичним джерелом ентропії, яке ґрунтується на природних процесах, однак характеризується низькою швидкодією. DRBG (Deterministic Random Bit Generator) є детермінованим алгоритмом, що на основі початкового seed формує псевдовипадкову послідовність із високою швидкістю.

Оскільки жоден із цих підходів окремо не задовольняє усіх практичних вимог: TRNG надто повільний для великих обсягів даних, а DRBG потребує якісного початкового значення, у реальних системах використовується їхнє поєднання. Фізичні джерела ентропії забезпечують формування початкового стану, а детермінований RNG будує подальшу псевдовипадкову послідовність, періодично оновлюючи свій стан через механізм reseed. Через обмежену швидкодію та необхідність статистичного контролю для ініціалізації використовується лише невеликий обсяг випадкових даних, на основі якого генератор формує вихідний потік [3].

На практиці початкова ентропія для криптографічних систем формується з різних фізичних і системних процесів, що використовуються для її накопичення. До них належать тепловий шум електронних компонентів, джитер як випадкові відхилення тактових сигналів, часові затримки процесів та асинхронні події операційної системи [2]. Окремим джерелом ентропії є апаратні генератори випадкових чисел, такі як Intel RDRAND, який генерує значення безпосередньо на рівні процесора. У цьому випадку ентропія формується апаратним модулем CPU та може використовуватися як додаткова складова випадковості або як частина системного генератора. Отримані дані можуть також проходити криптографічну обробку для зменшення статистичних зміщень і підвищення рівномірності розподілу [4].

Проблема недостатньої ентропії найчастіше виникає на етапі запуску системи, коли обсяг випадкових подій ще недостатній. У серверних та вбудованих системах це вирішується шляхом фонової акумуляції даних та збереження стану генератора між перезавантаженнями [5].

Окрім криптографічних задач, в операційних системах ентропія використовується також для внутрішніх механізмів безпеки: рандомізації адресного простору (ASLR), формування ідентифікаторів процесів, часових міток та інших параметрів, що потребують непередбачуваності.

У Linux генерація випадкових чисел реалізується через механізми ядра, до яких належать /dev/random, /dev/urandom та системний виклик getrandom(). Рекомендованим є використання getrandom(), оскільки він безпосередньо працює з ядром і не використовує файлові інтерфейси. Історично /dev/random блокував виконання до накопичення ентропії, тоді як /dev/urandom продовжував роботу без очікування, використовуючи криптографічне розширення початкового значення. У сучасних версіях Linux ці відмінності значною мірою нівельовані завдяки уніфікованому системному генератору ядра [6].

У Windows для отримання випадкових значень використовується функція BCryptGenRandom(). Вона надає доступ до системного вбудованого генератора та виключає необхідність реалізації власного RNG [7].

Криптографічні бібліотеки реалізують подібний функціонал на рівні програмного середовища. Наприклад, в OpenSSL функція RAND_bytes() застосовується для створення ключів, IV, nonce та інших параметрів безпеки [8]. У PyCryptodome генерація значень також базується на ентропії операційної системи через системний генератор, тоді як стандартний модуль Python random не призначений для криптографічних задач [9].

Для перевірки якості RNG застосовуються статистичні тести випадковості. У стандарті NIST SP 800-22 визначено 15 методів аналізу бітових послідовностей. Вони дозволяють оцінити рівномірність розподілу та виявити структурні закономірності, однак не гарантують криптографічної стійкості. Такі тести використовуються для виявлення реалізаційних дефектів, включаючи зміщення бітів, періодичність та кореляції у даних [10].

Якість випадкових чисел безпосередньо впливає на безпеку захищених протоколів TLS, SSH, VPN та електронного цифрового підпису. Некоректна генерація таких параметрів може призвести до відновлення секретного ключа навіть за умови використання стійких алгоритмів [11].

Висновки

Рекомендації OWASP передбачають використання лише перевірених апаратних або системних генераторів випадкових чисел. Основними помилками є використання некриптографічних RNG, повторне використання nonce та ініціалізація пристрою передбачуваними значеннями, зокрема часовими мітками, що суттєво знижує ентропію вихідних даних [12].

Сучасна генерація криптографічно стійких випадкових чисел базується на поєднанні фізичних джерел ентропії та детермінованих криптографічних генераторів з обов'язковим механізмом reseed. Реалізації в операційних системах та криптографічних бібліотеках забезпечують необхідний рівень безпеки за умови правильного використання. Основними ризиками залишаються недостатня ентропія на старті системи та використання некриптографічних RNG.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Eastlake D., Schiller J., Crocker S. RFC 4086: Randomness Requirements for Security. *Internet Engineering Task Force (IETF)*. 2005. URL: https://www.rfc-editor.org/rfc/rfc4086.html (дата звернення: 21.04.2026).

2. NIST SP 800-90B. Recommendation for the Entropy Sources Used for Random Bit Generation. *National Institute of Standards and Technology*. 2018. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf (дата звернення: 21.04.2026).

3. Renesas Electronics. RA Family: Secure Random Generation with TRNG and DRBG. 2026. URL: https://en-support.renesas.com/knowledgeBase/22372154 (дата звернення: 21.04.2026).

4. Intel Corporation. Intel Digital Random Number Generator (DRNG) Software Implementation Guide. 2018. URL: https://www.intel.com/content/www/us/en/developer/articles/training/intel-digital-random-number-generator-drng-software-implementation-guide.html (дата звернення: 22.04.2026).

5. Lange D. Openssh taking minutes to become available, booting takes half an hour ... because your server waits for a few bytes of randomness. 2018. URL: https://daniel-lange.com/archives/152-hello-buster.html (дата звернення: 22.04.2026).

6. Linux Man Pages. random(4) – kernel random number generator devices. *Linux Kernel Architecture*. URL: https://man7.org/linux/man-pages/man2/getrandom.2.html (дата звернення: 22.04.2026).

7. Microsoft Learn. BCryptGenRandom function (bcrypt.h). *Microsoft Corporation*. 2024. URL: https://learn.microsoft.com/en-us/windows/win32/api/bcrypt/nf-bcrypt-bcryptgenrandom (дата звернення: 30.04.2026).

8. OpenSSL Project. RAND_bytes – generate random data. *OpenSSL Documentation*. URL: https://www.openssl.org/docs/manmaster/man3/RAND_bytes.html (дата звернення: 23.04.2026).

9. PyCryptodome. PyCryptodome Overview. *Anaconda.org*. 2025. URL: https://anaconda.org/channels/buildozer/packages/pycryptodome/overview (дата звернення: 23.04.2026).

10. NIST SP 800-22 Rev.1a. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. *National Institute of Standards and Technology*. 2010. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf (дата звернення: 25.04.2026).

11. RFC 6979. Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA). Internet Engineering Task Force (IETF). 2013. URL: https://www.rfc-editor.org/rfc/rfc6979.html (дата звернення: 30.04.2026).

12. OWASP Foundation. Cryptographic Storage Cheat Sheet. *OWASP Cheat Sheet Series*. URL: https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html (дата звернення: 25.04.2026).

Бондаренко Ірина Олексіївна **–** асистент кафедри менеджменту та безпеки інформаційних систем, Вінницький національний технічний університет, м. Вінниця, e-mail: bondarenko.i@vntu.edu.ua

Макогін Ульяна Олександрівна – студентка групи 2КІТС-23б, Факультет менеджменту та інформаційної безпеки, Вінницький національний технічний університет, м. Вінниця, e-mail: makog.uliana@gmail.com

Bondarenko Iryna O. **–** assistant of the Department of Management and Security of Information Systems Vinnytsia National Technical University, Vinnytsia, e-mail: bondarenko.i@vntu.edu.ua

Makohin Uliana O. – – student of group 2KITS-23b, Faculty of Management and Information Security, Vinnytsia National Technical University, Vinnytsia, e-mail: makog.uliana@gmail.com