ADVANCES IN CYBER-PHYSICAL SYSTEMS

Vol. 7, Num. 2, 2022

TESTING OF THE RANDOM CODES GENERATOR OF EMBEDDED CRYPTO PROTECTION SYSTEM

Volodymyr Bilenko, Rahma Mohammed Kadhim Rahma, Valeriy Hlukhov

Lviv Polytechnic National University, 12, Bandera Str., Lviv, 79013, Ukraine.
The Iraqi Engineers' Union, P O Box 6204, Al-Mansour, Baghdad, Iraq
Authors' e-mail*: volodymyr.bilenko.mkiks.2021@lpnu.ua*

https://doi.org/10.23939/acps2022.__.___
Submitted on 22.09.2022

***Abstract*: The goal of the publication is to test the random codes generator of the built-in crypto-protection system**

Following the list of critical technologies in the production of weapons field and military equipment (following the Decree of the Cabinet of Ministers of Ukraine dated 30.08.2017 No. 600-r), an embedded microcontroller system for the protection of information for on-board equipment has been developed. A valuable stage of the system introduction is its research and testing. One of the stages of testing is the verification of the generator of random codes to use for the generation of encryption keys and digital signatures.

Based on the previous works and research of the Kalyna algorithm, methods and tools for creating a random code generator have been studied to use in the built-in cryptographic data protection system for data encryption/decryption and for working with a digital signature. Means of checking generated random codes and comparing them with existing counterparts have been developed.

The purpose of this article is to check the generator of random codes before using it in the built-in information protection system.

***Index Terms*: random codes generator, TRNG, PRNG, cryptography, microcontroller, FPGA**

I. INTRODUCTION

In the work, methods and tools for creating a random code generator [1] are investigated to use in the built-in cryptographic data protection system for data encryption/decryption and for working with a digital signature. In [2] and [3], based on the Kalyna algorithm, implementation methods using a microcontroller are described. The main purpose of [4] is to describe how to use big keys in crypto ciphering algorithms based using the Kalyna example.

Classification of generators:

Random code generators (TRNG);

Pseudo-random code generators (PRNG).

A pseudo-random number generator (PRNG) [5] refers to an algorithm that uses mathematical formulas to generate sequences of random numbers. PRNGs generate a sequence of numbers that approximates the properties of random numbers.

A PRNG starts from an arbitrary initial state using the initial state. Many numbers are generated in a short time and can be reproduced later if the starting point of the sequence is known. Therefore, the numbers are deterministic and efficient.

Characteristics of PRNG:

- Efficient. PRNG can generate plenty of numbers in a short time and is advantageous for applications that need a lot of numbers
- Deterministic. The given sequence of numbers can be reproduced later if the starting point in the sequence is known.
- Periodic. PRNGs are periodic, meaning that the sequence will repeat itself over time.

A true random number generator (TRNG) is described in [6]. It is a device that generates random numbers from a physical process rather than an algorithm. Such devices are often based on microscopic phenomena that generate low-level, statistically random "noise" signals, such as thermal noise, the photoelectric effect involving a beam splitter, and other quantum phenomena. This contrasts with the pseudorandom number generation paradigm commonly implemented in computer programs. Its principles are explained in [7].

A hardware random number generator typically consists of a converter to transfer some aspect of physical phenomena into an electrical signal, an amplifier, and other electronic circuitry to increase the amplitude of the random oscillation to a measurable level, and some type of analog-to-digital converter to transmit the output signal into a digital number, often the simple binary digit 0 or 1. By repeatedly sampling a randomly varying signal, a series of random numbers is produced.

Modern applications of random number generators (RNG) (explanation of random number generator used in the modern application is provided in [8]) extend to the field of information technology from the point of view of:

applications in the field of cryptography

— for individual user programs;

— for keys generation [9];

— to generate random initialization sequences (so-called initial) for encryption, authentication, or digital signature algorithms [10];

for asymmetric and symmetric cryptography, one-time/initial vectors, for authentication, selection of exponents in the Diffie-Hellman protocol;

other IT applications: for example, tags/tokens for communication protocols, indexing in databases, etc.;

statistical applications (for example, choosing a representative sample for statistical analysis);

Monte Carlo numerical simulation;

non-deterministic behavior of artificial intelligence (AI) — AI in computer games, self-driving devices (e.g., drones), etc.;

AI algorithms: neural networks (for example, random weighting of networks) and genetic algorithms (for example, the random introduction of mutations, random mixing of representatives);

structures and support services of currently popular cryptocurrencies (for example, bitcoin wallets, bitcoin exchanges, etc.);

gambling (for example, online casinos, also for cryptocurrency);

randomness in control processes (problem of sample selection for the quality of control processes);

randomness in administration (for example, establishing order in electoral lists);

probabilistic calculations, generated by quantum computing algorithms.

The above list briefly shows the extent of the range of applications of randomness and random number generators. In this context, the quality of randomness and its veracity becomes a fundamental problem.

Self-testing of quantum random number generators — the generated sequence is checked for randomness due to the limited confidence in the implementation of the physical process. This process is considered in [11]. Testing can be based on classical tests, as well as, for example, on testing the existence of quantum entanglement by testing Bell's inequalities 24. Due to the complexity of the testing process, such generators are usually slow or require additional sophisticated testing devices.

Federal Information Processing Standards (FIPS) 140-2 publication (for details see [12]) for cryptographic modules specifies four statistical tests for randomness. Instead of making the user select appropriate significance levels for these tests, explicit bounds are provided, which the computed value of a statistic must satisfy. A single bit stream of 20000 bits, output from a generator, is subjected to each of the four tests [13]: monobit test, block test, series test, and series length test. If any of the tests fail, then the generator fails the FIPS 104-2 statistical test for randomness. The distribution functions are derived and significance levels are calculated for each of the four statistical tests which are used for the verification of random number generators based on the DSTU 4145-2002 standard [13].

II. PROBLEM STATEMENT

The implications of the predictability of generated classical pseudorandom sequences are obvious—the absence of true randomness in any of the previously mentioned applications is an obstacle to predictable functioning. In the case of cryptographic applications, the consequences can be particularly severe. The problem of classic random number generators [6], i.e., generators of pseudo-random numbers, is the ability to know the deterministic process of generating pseudo-random numbers by unwanted persons. In the case of cryptography, this can compromise security confidence. Another problem can be improper handling of the generated sequence—mostly for cryptographic purposes, the generated random sequence is used only once. Its repeated use can lead to a security breach (for example, in the case of an OTP cipher, a long enough key must be truly random and used only once in this protocol, otherwise the code can be broken).

For a long time, cryptography meant only encryption — the process of transforming ordinary information (plain text) into incomprehensible "garbage."

Decryption is the reverse process of reproducing information from the ciphertext. A cipher is a pair of encryption/decryption algorithms. The operation of the cipher is controlled by both algorithms and the key. A key is a secret parameter (ideally known only to two parties) for a particular context during message transmission. Keys are very important because, without variable keys, encryption algorithms are easily broken and unusable in most cases. Historically, ciphers are often used for encryption and decryption, without performing additional procedures such as authentication or integrity checks.

There are many types of random number generators [6]. They can be divided, for example, by the type of generation process — software random number generators (software RNG — based on deterministic software) or hardware RNG (based on a physical phenomenon — classical or quantum). The different division is based on the physical nature of the generation process — classical RNG and quantum RNG. The perspectives of these two main subdivisions partially coincide — software RNGs are purely classical, while hardware RNGs are divided into classical, quantum, and generator, in which the nature of the physical process cannot be distinguished.

Classical hardware random number generators do not require initial entropy — in this case, the source of entropy is a classical physical process. If the available entropy is consumed, such a generator must wait until the generation process provides a sufficient fraction of new entropy. Generators of this class are also pseudo-random generators due to the determinism of classical physics, and therefore can be a potential target of attack.

III. PURPOSE OF THE WORK

The purpose of this work is the verification of a random code generator as a part of an embedded cryptography system. The system shall be based on the Ukrainian National Standard of Cryptosecurity DSTU4145-2002 [13] and MCU STM32G071. Using specialized test firmware perform the following checks: monobit test, blocks (poker) test, series test, and logs the series test.

The result of the research shall be used for debugging information security systems. All tests must be executed in a time interval of at least 20 seconds to be synchronized with other system components.

IV. STRUCTURAL SCHEME OF CRYPTOGRAPHY DATA PROTECTION SYSTEM

The structural scheme of the cryptography protection system is described in Fig. 1.

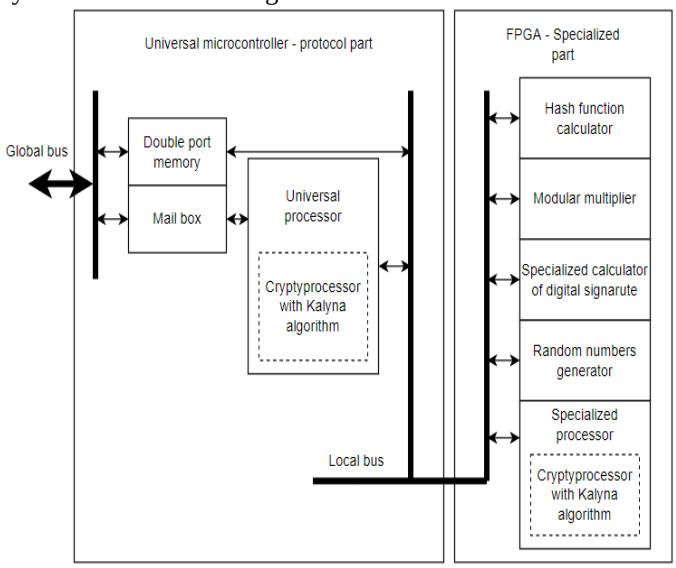

Fig. 1. Structural scheme of cryptography data protection system

It consists of two main parts:

FPGA – Specialized part (special processors);

A universal microcontroller is responsible for the protocol part of the operation.

Communication between these parts is conducted through the local bus of the 32-bit cipher processor.

The microcontroller includes the following nodes:

the core of the universal processor, where the data encryption algorithm works - Viburnum in modes (128, 128) and (128, 256);

mailbox, which implements a means of synchronization (semaphore) between the central processor and a specialized part;

two-port memory that provides direct communication of the specialized part directly with the global bus of the cipher processor, bypassing the core of the universal processor.

The FPGA implements the following blocks:

a hash function calculator that converts a given array of data of any length into a resulting string of predetermined length. These transformations have another, shorter name - hashing or convolution functions, and the result of their work is called a hash;

a modular multiplier, which performs addition with the help of one-digit adders, which are capable of accumulation and can form and accumulate the sign of the transition of the maximum permissible value at the same time. If such excess was detected, the next summation of the result by the specified modulus from the intermediate result takes place, which at the end leads to the addition of the supplementary code of the number with the previous value;

a specialized digital signature calculator based on elliptic curves over finite fields and rules for applying the mechanism to data sent over communication channels. The crypto resistance of the algorithm according to which it works on a straight line depends on the number of points falling into the elliptic curve group on the Galois field. It contains an algorithm for checking the digital signature for correctness and implementation of the signature;

random number generator controller to continuously provide non-repeating sets of numbers that could be used for safe and secure encryption.

V. USAGE OF KEYS IN CRYPTOGRAPHY DATA PROTECTION SYSTEM

Key is just a very large number. Keys are a fundamental component of cryptography, used in cryptographic operations such as encryption, hashing, and signing to provide desired properties such as confidentiality, integrity, or authenticity [9].

The length of a key is measured by the number of binary digits needed to represent its value. Keys are typically hundreds or, in some cases, several thousand bits long. This strikes a balance between security and computational efficiency - too short and the key is insecure, too long and it makes cryptography practically slow.

There are two main types of keys: symmetric and asymmetric.

Symmetric keys are usually used to encrypt and decrypt information. They are analogous to physical keys in that they are used to securely "lock" information so that it can only be "unlocked" by someone with the same key.

Asymmetric keys always come in pairs, each pair containing a "private key" and a "public key" that are mathematically related to each other. The private key must be kept secret, but the public key, as its name suggests, can be shared with anyone. Such keys make "public key cryptography" possible.

Keys are typically generated by computers in software using random number generator functions built into operating systems and programming language libraries. This is sufficient for most everyday purposes but can lead to weak keys and offers little protection against determined attackers.

Keys can be operated manually, but this is time-consuming, error-prone, and dangerous. To avoid this, there is an electronic key management system that allows to manage keys efficiently and securely throughout their life cycle, automatically keeping a permanent electronic record of everything that happens to each key in the form of a security audit.

In addition, the key management system can apply best-in-class security policies to ensure secure key management and protection against internal and external threats.

Such a system is an important tool for maintaining and demonstrating compliance with many different standards (such as PCI-DSS), especially in highly regulated sectors such as finance, healthcare, and government.

Finally, key management systems provide the ability to centralize and automate key management, resulting in a much more efficient way to manage the ever-growing number of keys that often protect many of a company's most valuable assets.

The further goal after the generation of keys is their use as part of information protection algorithms. One of these is

DSTU 7624:2014 [1] with the code name Kalyna - it is a symmetric block cipher for protecting information.

The Kalyna algorithm provides:

extremely high resistance;

high speed of implementations, both on lost and hardware platforms;

comparable, and in some cases, higher efficiency compared to the best global solutions;

availability of various modes of operation necessary for modern crypto protection (see Table 1).

Table 1.

Cipher states

Word length, bits	Block size, bits	Key size, bits	Identifier	Rounds
64	128	1 x 128 = 128	Kalyna 128/128	10
		2 x 128 = 256	Kalyna 128/256	14
	256	1 x 256 = 256	Kalyna 256/256	
		2 x 256 = 512	Kalyna 256/512	18
	512	1 x 512 = 512	Kalyna 512/512	

VI. USAGE OF ELECTRONIC DIGITAL SIGNATURE IN CRYPTOGRAPHY DATA PROTECTION SYSTEM

An electronic digital signature [10] is a type of electronic signature obtained because of the cryptographic transformation of a set of electronic data, which is added to this set or logically combined with it and allows it to confirm its integrity and identify the signer. An electronic digital signature is imposed using a private key and verified using a public key.

A reliable means of an electronic digital signature is a means of an electronic digital signature that has a certificate of conformity, or a positive expert opinion based on the results of a state examination in the field of cryptographic information protection.

One of the mandatory requisites is an electronic signature, which is used to authenticate the author and/or signer of an electronic document by other subjects of electronic document circulation.

An electronic copy with an electronic digital signature of the author is considered the origin of the electronic document.

An electronic digital signature is an integral part of the public key infrastructure. It is imposed using a private key and verified using a public key. In terms of legal status, it is equivalent to a handwritten signature. An electronic signature cannot be invalidated simply because it is in electronic form or is not based on a strengthened key certificate.

The EDS personal key is based on completely random numbers generated by the random number generator, and the public key is calculated from the EDS personal key in such a way that it was impossible to obtain the second one from the first. It is a unique sequence of characters with a length of 264 bits. The private key works only in a pair with a public key.

The public key is used to check the EDS of received documents (files). The public key works only in pair with the private key. The public key is contained in the Public Key Certificate and confirms the ownership of the EDS public key to a specific person.

Typically, three algorithms are used for the digital signature process:

Key Generation – This algorithm provides a private key along with the corresponding public key.

Signature – this algorithm creates a signature after receiving the private key and the message to be signed.

Verification – This algorithm verifies the authenticity of a message by verifying it against the signature and public key.

The digital signature process requires that the signature generated by both the fixed message and the private key can then be authenticated by the accompanying public key. Using these cryptographic algorithms, a user's signature cannot be reproduced without access to their private key.

By applying asymmetric cryptography techniques, the digital signature process works to prevent several common attacks where an attacker tries to gain access using the following attack methods:

Key only – the attacker has access to the public key

Known message – the attacker has access to valid signatures for known messages, but not the ones he selected

Adaptive Selected Message – An attacker has access to signatures on different messages they have selected

VII. ALGORITHM OF RANDOM CODES GENERATOR

The algorithm for the random sequence generator is based on the DSTU4145-2002 standard [13]. This generator will be used to generate random integers, random elements of the basic field, and random points of elliptic curves. The generator produces a random string of length t=1 for one call to it. As a cryptographic transformation in the generator, the cryptographic transformation algorithm according to GOST 28147 is used in the simple replacement mode [11]. The substitution table and the private key of this conversion must comply with GOST 28147-89. The conditions of obtaining and using the personal key must make it impossible to access it or its part, modify, replace, or destroy it. The private key of the cryptographic transformation according to GOST 28147, which is used in the generator of random sequences, cannot be used for any other purpose.

We denote by E_k the encryption of a binary string of length 64 by the GOST 28147 algorithm in the mode of simple substitution with a key k of length 256 binary digits. Let s, D, I, x – binary strings with a length of 64 binary digits. Before use, the initial state of the random sequence generator is set.

Setting the initial state of the random sequence generator:

Set the initial values of the generator. For this, a physical source of randomness is used. As such a source, you can use, for example, quantum effects in semiconductors (noise diodes, etc.), a signal from a microphone input with the microphone turned off, time intervals between keyboard key presses, and time intervals between mouse key presses. The initial state of the generator is secret. The conditions for obtaining the initial state of the generator should make it impossible to access it or its parts, modify, replace, or destroy it.

• Set the value of the binary string D. For this, use the current value of the date and time with an accuracy of 64 binary digits.

• Calculate the binary string $I = E_k(D)$.

Using a random sequence generator at each call to the random sequence generator, the following calculations are performed

- $x = E_k(I \oplus s)$;
- $s = E_k(x \oplus I)$.

A random binary string is a binary string of length 1, which consists of the rightmost digit x0 of the binary string $x = (x_{63}, x_{62}, ...x_0)$.

VIII. VERIFICATION

To check random sequences for correctnes, it is necessary to have a verification program and the testing plan. The following steps will be included for the verification of random code generator:

- Compilation of the program code into an executable file in the KEIL uVision environment;
- Downloading the executable .axf file to the microcontroller with a memory integrity check;
- Checking the UART communication between the PC and the microcontroller;
- Request to transfer a random sequence to the PC screen via UART;
- Making a second request to verify that the sequence is changing;
- Sending a <start> message to start the randomness tests;
- Make sure that all tests have been completed successfully.

Random sequences generator has been tested using specific test suite [13] which includes the following tests:

1) Monobit. The purpose of the test is to check whether the number of 1s and 0s in a random sequence is the same as would be expected from a truly random sequence. Let n_1 and n_2 denote the number of zeros and ones in the sequence x, respectively. If the sequence is random, then the values of n_1 and n_2 must satisfy the condition $9654 < n_1 \bigcap n_2 < 10346$.

2) Blocks test. Let m be a positive integer such that $\frac{n}{m} \geq 5(2^m)$ and let $k = \frac{n}{m}$. The sequence is successively divided into disjoint subsequences of length $m, (m = 2, 3, ...)$. Let n_i be the number of occurrences of the ith type of sequence of length m. The block test determines whether sequences of length m appear approximately as many times in the sequence x as it can be expected for a random sequence. The parameter calculation is used to apply the criterion $X_3 = \frac{2^m}{k}(\sum_{i=1}^{2^m} n^2{}_i) - k$ which has a distribution close to the x^2 distribution with 2^{m-1} degrees of freedom. The statistical parameter given by the equation is calculated for m = 4. The statistic must satisfy the condition $1.03 < X_3 < 57.4$.

3) Series test. A series is understood as a sequence of identical symbols, i.e. ones or zeros in a row. The essence of the test is that series of 1, 2, 3, 4, 5, and 6 elements are counted on the given length of the sequence being tested (series with a length of more than 6 elements are considered as series of length 6). If the sequence is random, then the number of series of each length must be in intervals (see Table 2);

Table 2.

Series test requirements

The length of the series	Required interval
1	2267 - 2733
2	1079 -1421
3	502 - 748
4	223 - 402
5	90 - 223
6	90 - 223

4) The length of the series. The essence of the test is to check the maximum length of a series of identical elements. If the sequence is random, then the maximum length of the sequence should not exceed 34.

The technological computer screen with the results of checking the random code generator is shown in Fig. 2. It takes 20 seconds on average to perform all checks as it is required by the purpose of the work.

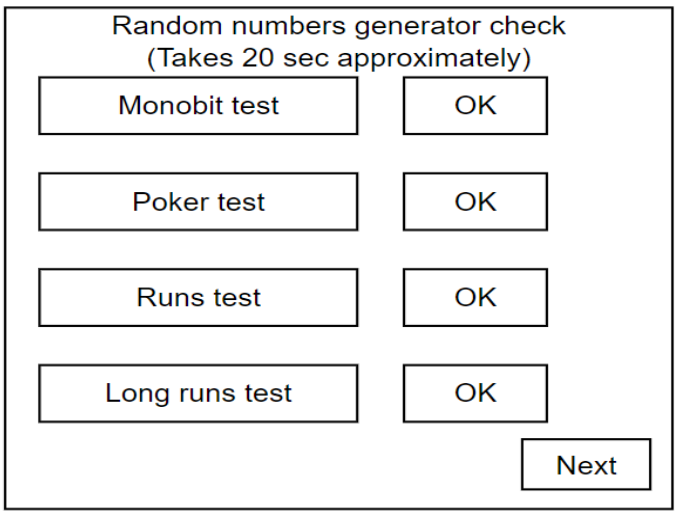

Fig. 2. Random codes generator checks

IX. CONCLUSION

The random code generator of the embedded crypto protection system was tested according to the DSTU 4145-2002 standard. STM32G071 microcontroller program was created for generator testing using monobit test, block test, series test, and series length test. The results of the research were used to adjust information protection devices. High speed of random number generator checks was achieved (all checks are performed within 20 seconds).

References

[1] DSTU 7624: 2014 (2015) Information Technology. Cryptographic information protection. Symmetric block transformation algorithm. Kyiv, Ukraine: Ministry of Economic Development of Ukraine, 228 p. http://online.budstandart.com/ua/catalog/doc-page?id_doc=65314 (Accessed: 1 October 2022)

[2] Bilenko V.M. Implementation of Kalyna algorithm at microcontroller (2021). Student technical-scientific conference of IKTA. Lviv, Ukraine. pp. 167–168. https://science.lpnu.ua/sites/default/files/attachments/2021/nov/25578/maket2021n.pdf (Accessed: 1 October 2022)

[3] Bilenko V.M., Hlukhov V.S. Implementation Kalyna algorithm in microcontroller “Advances in cyber-physical system” – (2021). vol.6, num. 1. pp. 8–13. DOI: https://doi.org/10.23939/acps2021.01.008 (Accessed: 1 October 2022)

[4] Zayats T. Bilenko V.M., Hlukhov V. Features of using large keys in “Kalyna” algorithm. “Advances in cyber-physical systems” (2022). vol.7 num. 1. pp. 55–62. DOI: https://doi.org/10.23939/acps2022.01.055

[5] Fabio Acerdi, Nicola Massari, Leonardo Gasparini. Structures and Methods for Fully Integrated Quantum Random Number Generators. Journal of Selected Topics in Quantum Electronics (2020). vol.36 num. 3. pp. 36–51. DOI: https://doi.org/10.1109/JSTQE.2020.2990216

[6] Yuan Zhang, Xiaofeng Shi. The reach of pseudo-random signals generator based on FPGA (2018). IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). Chongqing, China. pp. 5–14. DOI: https://doi.org/10.1109/IAEAC.2018.8577873

[7] Yuanhao Li, Hong Wang, Zhi Ma. Quantum random number generator using cloud superconducting (2021). Scientific reports. Hong Kong, China. pp. 21–32. DOI: https://doi.org/10.1038%2Fs41598-021-03286-9

[8] Randy Kuang, Dafu Lou, Alez He, Chris McKenzie, Michael Redding. Pseudo Quantum Random Number Generator with Quantum Permutation Pad (2021). IEEE International Conference on Quantum Computing and Engineering, Broomfield, CO, USA. pp. 6–11. DOI: https://doi.org/10.1109/QCE52317.2021.00053

[9] Mohamed Ali Kandi, Djamel Eddie Kouicem, Hicham Lakhlef, Abdelmadjid Bouabdallah, Yacine Challal (2020). A blockchain-Base Key Management Protocol for Secure Device-to-Device Communication in the Internet of things. 19th International Conference on Trust, Security, and Privacy in Computing and Communications (TrustCom). Guangzhou, China. pp. 17–21. DOI: https://doi.org/10.1109/TrustCom50675.2020.00255

[10] Surendra Singh Chauhan, Nitin Jain, Satish Chandra Pandey. Digital Signature with Message Security Process (2022). 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE). Greater Noida, India. pp. 36–48. DOI: https://doi.org/10.1109/ICACITE53722.2022.9823539

[11] Vynokurov A. Ciphering algorithm GOST 28147-89, its usage and implementation for computer platform Intel x86 (2001). Moscow. https://tzi.com.ua/downloads/28147-89.pdf (Accessed: 1 October 2022)

[12] FIPS PUB 140-2 (2007). Security Requirements for Cryptographic Modules. National Institute of Standards and Technology. Heytersberg, USA, p. 140. DOI: https://doi.org/10.6028/NIST.FIPS.140-2 (Accessed: 1 October 2022)

[13] Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographical Applications (2010). NIST Spec. Pub. 800-22, rev. 1a. 131 p. https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762 (Accessed: 1 October 2022)

[14] DSTU 4145-2002 (2003). Cryptography data protection. Digital signature based on elliptical curves. Kyiv, Ukraine: Small Enterprise “Dyna” 39 p. https://itender-online.ru/wp-content/uploads/2017/09/dstu-4145-2002-1.pdf (Accessed: 1 October 2022)

Volodymyr Bilenko was born in 2000 in Talne, Cherkasy region, Ukraine. In 2021 he graduated with a Bachelor’s degree of Computer Engineering at Lviv Polytechnic National University. After that, he enrolled for a Master’s degree studies and now he is a six-year student in Computer Systems and Networks at Lviv Polytechnic National University. He was involved in the development of systems for military purposes with encryption topics and medical devices. He is interested in topics related to embedded system engineering such as the Internet of Things (IoT), robotics, and applied automation systems.

Rahma Mohammed Kadhim Rahma was born in 1979 in Al-Qadisiyyah, Iraq. He received the B.S. degree in "Computer Engineering" at College of Engineering, Al-Mustansiriya University in (2004-Baghdad, Iraq) and M.S degree in "Computer Systems and Networks" at Lviv Polytechnic National University in 2015, Lviv. In 2020 he successfully defended his dissertation on "Models and methods for constructing operating units for Galois fields used in cryptographic data protection based on elliptic curves" at Lviv Polytechnic National University. Now he is a member of the Iraqi Engineers’ Union. His research interests include algorithms of hardware data protection in cryptography.

E-mail: muhamed_kadhem@yahoo.com

Valerii Hlukhov is a professor of the Department of Computer Engineering Department at Lviv Polytechnic National University, Ukraine. He graduated from Lviv Polytechnic Institute with the engineer degree in Computer Engineering in 1977. In 1991 he obtained his Ph.D. at the Institute of Modeling Problems in Power Engineering of the National Academy of Science of Ukraine. He was recognized for his outstanding contributions into special-purpose computer systems design as a Senior Scientific Researcher in 1995. He was awarded the academic degrees of Doctor of Technical Sciences in 2013 at Lviv Polytechnic National University. He became a Professor of Computer Engineering in 2014. He has scientific, academic, and hands-on experience in the field of computer systems research and design proven contribution into IP Cores design methodology and high-performance reconfigurable computer systems design methodology. He is an experienced researcher in computer data protection, including cryptographic algorithms, cryptographic processors design and implementation. Prof. Hlukhov is an author of more than 100 scientific papers, patents, and monographs.