Balancing security and efficiency in deterministic random bit generators for post-quantum cryptography

Maksim Iavich[1,†], Sergiy Gnatyuk[2,3,*,†], Tamari Kuchukhidze[1,†] and Giorgi Iashvili[1,†]

[1] *Caucasus University, Paata Saakadze Str., 1, Tbilisi, 0102, Georgia*

[2] *National Aviation University, Liubomyra Huzara Ave., 1, Kyiv, 03058, Ukraine*

[3] *State Scientific and Research Institute of Cybersecurity Technologies and Information Protection, Maksym Zalizniak Str., 3/6, Kyiv, 03142, Ukraine*

Abstract

The creation of secure random numbers is essential to cryptography since it guarantees key management, encryption, and authentication. The pseudo randomness is provided by deterministic random bit generators (DRBGs), among which Hash-DRBG, HMAC-DRBG, KHF-DRBG, AES-CTR DRBG, and TDEA-CTR DRBG are important techniques specified in NIST standards. Their applicability for post quantum cryptography (PQC), security characteristics, and architecture are all examined in this article. While Hash-DRBG and KHF-DRBG provide a balance between efficiency and security, HMAC-DRBG and AES-CTR DRBG exhibit robust resistance against state compromise. System needs determine whether deterministic random bit generators (DRBG) is best, with computing cost, entropy management, and resistance to future cryptographic attacks all being important considerations. Choosing the most secure and flexible deterministic random bit generator will be crucial as cryptographic systems develop in order to maintain long-term security in both conventional and post-quantum settings.

Keywords

quantum cryptography, post-quantum cryptography, random number, pseudo random number generator, DRBG

1. Introduction

With the arrival of quantum encryption and post-quantum cryptography as defenses against the exponential speed advantage of quantum computers, computing capabilities are being altered. The difference between the speed at which quantum computing can tackle complicated issues and the lengthy execution times of regular computers makes this shift critical. But as quantum computing advances, questions are raised regarding the practicality of existing cryptographic techniques, especially those that depend on RSA, which uses mathematical issues like integer factorization. Due to their ability to solve complex mathematical problems quickly, large-scale quantum computers outfitted with Shor's algorithm represent a serious threat to current public key cryptography methods [1, 2].

Post-quantum cryptosystems are being developed to resist and defeat quantum assaults in response to this looming issue. Since traditional asymmetric techniques like RSA may not be sufficient to protect private information, the development of quantum technology demands the ongoing search for robust post-quantum systems [3].

In order to foresee how quantum computing will affect cryptographic security, the National Institute of Standards and Technology (NIST) is working to develop strong cryptographic algorithm

CH&CMiGIN'24: Third International Conference on Cyber Hygiene & Conflict Management in Global Information Networks, January 24–27, 2024, Kyiv, Ukraine

* Corresponding author.

† These authors contributed equally.

miavich@cu.edu.ge (M. Iavich); serhii.hnatiuk@npp.nau.edu.ua (S. Gnatyuk); tkuchukhidze@cu.edu.ge (T. Kuchukhidze); giiashvili@cu.edu.ge (G. Iashvili)

0000-0002-3109-7971 (M. Iavich); 0000-0003-4992-0564 (S. Gnatyuk); 0000-0003-1997-465X (T. Kuchukhidze); 0000-0002-1855-2669 (G. Iashvili)

standards that can withstand quantum computer attacks and safeguard private information in the post-quantum computing era [4].

When it comes to encryption and security protocols, random bits are essential components of safe cryptographic systems. In order to produce uniform random bits with complete entropy that are independently dispersed and identically distributed, they should be made to be similar to an ideal randomness source. It is difficult to accomplish this security objective, though, particularly in post-quantum cryptography, which calls for a higher-quality and more accurate randomness source [5, 6].

Cryptographic systems are changing quickly due to quantum computing, especially those that use traditional techniques that are susceptible to quantum assaults. As cryptography shifts to a post-quantum paradigm, deterministic random bit generators (DRBGs) [7], which are crucial for safe key generation, nonce creation, and digital signatures, require a reevaluation.

The DRBG uses a seed, which is a hidden beginning value, to generate a series of bits. Another characteristic of a cryptographic DRBG is that, since the seed is unknown, the result is unexpected. Other names for a DRBG include deterministic random number generator and pseudo-random number generator (PRNG) [8]. With repeatability, dependability, and robust cryptographic security, DRBGs provide an effective way to generate high-quality pseudorandom bits. Because of their deterministic nature, they are also ideal for contexts with limited resources and embedded devices, where true random number generators (TRNGs) would not be feasible.

Since quantum-safe alternatives are replacing conventional primitives like RSA or ECC in post-quantum cryptography, DRBGs must adapt to produce secure randomness. Because of this shift, DRBGs must be able to withstand quantum assaults on the algorithms they use in order to remain dependable in a cryptography environment that is evolving quickly [9, 10].

The importance of DRBGs in cryptographic systems and how they have evolved to meet post-quantum security standards are examined in this article. It demonstrates the integration of DRBGs into post-quantum cryptography protocols and assesses the appropriateness of several DRBG types, including hash-based and block cipher-based DRBGs, for quantum-resistant cryptography. Our goal is to aid in the creation of safe cryptography systems for the post-quantum era by determining the most reliable and workable methods.

2. Deterministic random bit generator (DRBG)

Generating safe and unexpected random numbers is crucial to current cryptography in order to safeguard private data and maintain system integrity. The two basic approaches to designing random bit generators are as follows: the first is non-deterministic, in which each bit of output is based on an unpredictable physical process; the second computes bits deterministically by means of an algorithm seeded with an initial value that provides enough entropy to ensure randomness. The latter kind is known as deterministic random bit generators (DRBGs) or pseudo-random bit generators [11]. PRBGs produce pseudo-random (as opposed to really random) bits because of their determinism.

The method of creating random bits in DRBGs is split into two steps: a cryptographic algorithm creates the output bits after an entropy source supplies an unexpected input string as a seed. Cryptographic primitives such as stream ciphers, block ciphers, hash functions, and elliptic curves are typically used as fundamental building blocks in real-world PRBGs. For example, Hash-DRBG, HMAC-DRBG and other DRBGs are suggested by the updated NIST SP 800-90A standard [12] and are based on authorized hash functions and block ciphers [13].

An input string with a guaranteed minimum entropy is used to seed a DRBG. It will produce output bits that are computationally identical to ideal random bits if it is seeded correctly. Moreover, output bits produced before to a compromise must be identical to ideal random bits ("backtracking resistance" [14]) due to the possibility of a DRBG's internal state being compromised. Once enough fresh entropy is supplied following a compromise, the DRBG needs to recover ("prediction resistance") [15]. There has been criticism of the NIST SP 800-90A standard, particularly in relation to the Dual_EC_DRBG algorithm's inclusion, which was shown to contain a backdoor [16].

Additionally, the suggested DRBGs do not fit well into the typical security models of PRBGs since they offer a wide range of alternate inputs and settings. Concerns have also been raised about the inadequate formal study of these DRBGs and the absence of formal competition during the standardization process.

DRBGs are typically built using other cryptographic primitives like stream ciphers, block ciphers, or hash functions. For example, the bases for DRBGs are frequently AES, SHA-2, and SHA-3. The ChaCha20 cipher is utilized for DRBG in Linux. The security, effectiveness, and appropriateness of these DRBGs for post-quantum cryptography (PQC) are assessed. DRBGs may be roughly divided into two types: block cipher-based DRBGs, which employ symmetric block ciphers like AES, and hash-based DRBGs, which rely on cryptographic hash functions [17].

Secure procedures for deterministic random bit generation are defined by the NIST SP 800-90A standard, which was initially published in June 2006 and updated in 2015 [18]. Significant changes have been made to it, such as the elimination of the Dual_EC_DRBG algorithm because of security issues. Due to possible backdoors that were purportedly created by the NSA and might enable attackers to predict DRBG results, Dual_EC_DRBG had drawn criticism. The significance of public review and transparent cryptography design was highlighted by that conflict.

Although there are a number of other DRBGs, not all of them are appropriate for PQC. Based on modular arithmetic and quadratic residues, the Blum-Blum-Shub (BBS) generator squares an integer modulo a product of two big primes to produce pseudorandom integers. Because it relies on integer factorization, which is liable to quantum attacks, it is extremely slow and inappropriate for PQC even if it is secure under conventional assumptions. Similar to this, the Dual_EC_DRBG generated pseudorandom numbers using elliptic curve cryptography (ECC), but it was eliminated from NIST standards because of flaws and inefficiency. ECC-based systems are not appropriate for PQC as, like BBS, they are susceptible to Shor's algorithm.

Fortuna PRNG is a versatile pseudorandom number generator that employs the AES block cipher for randomness and other entropy sources. Although it is not officially standardized, it is very adaptable and impervious to governmental compromise. Depending on the block cipher being used, its PQC applicability varies. When combined with AES-256, it can offer quantum resistance, although it does not have explicit post-quantum security assurance [19].

ChaCha20-based DRBGs create pseudorandom numbers using the ChaCha20 stream cipher. They are commonly used in contemporary cryptography frameworks and are renowned for their excellent performance in software implementations. ChaCha20 was not created especially for quantum resistance, and NIST has not standardized it as a DRBG. Although it is resistant to classical assaults, the absence of formal analysis in the quantum setting results in a moderate PQC applicability [20, 21].

We need to integrate quantum resistant DRBGs into post-quantum cryptography protocols and assesses the appropriateness of several DRBG types, including hash-based and block cipher-based DRBGs. Our goal is creating cryptography systems for the post-quantum cryptography by determining the most reliable and workable methods.

3. Hash-based deterministic random bit generators

Since the production of keys is one of the most crucial aspects of defending a cryptographic system, a random number generator is one of the most vital components. A pseudo random number generator with characteristics that make it appropriate for use in cryptography systems for the creation of keys is known as a cryptographically secure pseudorandom number generator (CSPRNG) [22].

NIST SP 800-90A Rev.1 has a handful of these standards, including Hash_DRBG, HMAC_DRBG, and CTR_DRBG. HMAC_DRBG and Hash_DRBG are both hash-based DRBGs. Every internal Hash_DRBG process, including the instantiation, reseeding, and pseudorandom number generation processes, uses a hash function. As a result, choosing the right hash function for the Hash_DRBG is crucial. Only a few SHA families, including SHA1, SHA2 and SHA3 are available. According to earlier

research, general assaults such brute force attacks, domain extender attacks, poisoned block attaches, etc., can break SHA1 and earlier versions. So, it is very important to use quantum resistant hash functions for our post quantum cryptography systems.

Hash-DRBG uses cryptographic hash methods like SHA-256 or SHA-3 to produce pseudo random numbers. The approach is straightforward and ensures unpredictability by updating the internal state after each output. Hash-DRBG produces pseudo random outputs by using cryptographic hash methods like SHA-256 or SHA-3. It is suitable with lightweight cryptography systems because to its efficiency and simplicity. It does not, however, have a keyed mechanism, making the internal state susceptible to compromise. Although SHA-256 and SHA-3 are thought to be quantum-resistant, their resilience is limited when compared to keyed DRBGs due to their dependence on hash function strength alone. Furthermore, the main prerequisite for Hash-DRBG's security is pseudo randomness. The pseudo random output is not directly weakened by collision resistance or preimage resistance, despite the fact that these are desired characteristics.

The Hash-DRBG generation procedure consists of the following steps:

1. Revise the internal state:
$$V = V + 1. \tag{1}$$
2. Calculate the output:
$$\text{Returned\_Bits} = H(V) \| H(V+1) \| \ldots \tag{2}$$
3. For the following iteration, update the internal state:
$$V = V + 1. \tag{3}$$
where: V: Internal state variable updated after each generation, H: The hash function used (e.g., SHA-256 or SHA-3).

This framework lowers the chance of compromise by updating the state with each output, ensuring forward security. The stability of Hash-DRBG under appropriate initialization and reseeding procedures is confirmed by security proofs, such as those presented by Woodage and Shumow in 2019. Although Hash-DRBG is effective, it does not have a keyed mechanism, therefore the strength of the hash function is the only factor affecting its security. When employing a quantum resistant function such as SHA-3, it offers good applicability for PQC.

HMAC-DRBG uses a keyed hash function (HMAC) to improve the security of the DRBG process, making it more resilient to backtracking and state compromise attacks. Hash-based Message Authentication Codes (HMAC) are used by HMAC-DRBG to generate pseudo-random integers. For increased security, it incorporates a keyed mechanism that is resistant to state compromise and backtracking. The introduction of a second secret (key) in the HMAC procedure, which improves security, is a significant distinction from Hash-DRBG. However, since recovering the key undermines all outputs, resistance to key recovery assaults is essential. HMAC-DRBG is thought to be particularly effective for PQC, while being computationally more costly.

The following is the HMAC-DRBG generation process:

1. Use HMAC to update the internal value:
$$V = \text{HMAC}(K, V). \tag{4}$$
2. Determine the output by computing "Returned_Bits":
$$\text{Returned\_Bits} - V \| \text{HMAC}(K, V+1) \| \ldots \tag{5}$$
3. Update the internal state:
$$V = \text{HMAC}(K, V). \tag{6}$$

While V is the value, K is the internal key, and extra entropy is provided via optional input. This guarantees resistance to internal state tampering and a strong pseudorandom output.

Because of its robust defense against key recovery attacks, HMAC-DRBG is regarded as extremely safe and a great choice for post-quantum cryptography.

The Keyed Hash Function DRBG (KHF-DRBG) is comparable to HMAC-DRBG, but it applies the key in a different way throughout the hashing process. KHF-DRBG offers a balance between security

and efficiency by using a keyed hash function. When using more recent hash algorithms like SHA-3, it maintains significant quantum resistance even though it lacks the advanced keyed mechanisms of HMAC-DRBG. For systems that need to generate randomness in an efficient and safe manner, this makes it a suitable option [23].

The structure of the generation process is as follows:

1. Utilizing the keyed hash function, update the internal state:
$$V = H(K \| V \| \text{ input }). \tag{7}$$
2. Create the output:
$$\text{Returned\_Bits } - H(K \| V) \| H(K \| V + 1) \| \ldots \tag{8}$$

Although KHF-DRBG is effective and offers balanced security, its resilience to state compromise is not as strong as that of HMAC-DRBG.

4. Block-based deterministic random bit generators

AES-CTR DRBG creates pseudo random numbers by using the AES block cipher in counter mode (CTR). Grover's technique reduces the effective security of AES-256 to 128 bits while maintaining good quantum resistance. In systems with hardware acceleration, it is quite effective, but for best results, careful key management is needed. AES-CTR DRBG is a great option for post quantum cryptography as it is widely trusted and extremely resilient to quantum attacks [24, 25].

The AES-CTR DRBG generation procedure is as follows:

1. The counter value should be encrypted:
$$\text{Output }_i = \text{AES}(K, \text{ Counter }_i). \tag{9}$$
2. Increment the counter:
$$\text{Counter } = \text{ Counter } + 1. \tag{10}$$
3. To get the pseudo random bits, concatenate the outputs:
$$\text{Returned\_Bits } = \text{ Output }_1 \| \text{ Output }_2 \| \ldots \tag{11}$$

AES-CTR DRBG is a great option for post quantum cryptography with robust resilience to quantum assaults since it leverages the well-established security of AES-256.

The TDEA-CTR DRBG employs the same counter-mode technique but substitutes Triple DES (TDEA) for AES. Although it is compatible with older systems, it is susceptible to both quantum and classical assaults because to its small 64-bit block size. Additionally, it performs worse than AES. For post-quantum applications, TDEA-CTR DRBG is thus not advised.

The create procedure is as follows:

1. To encrypt the counter USEe TDEA:
$$\text{Output }_i = \text{TDEA}(K, \text{ Counter }_i). \tag{12}$$
2. Raise the counter:
$$\text{Counter } = \text{ Counter } + 1. \tag{13}$$
3. Concatenate outputs:
$$\text{Returned\_Bits } = \text{ Output }_1 \| \text{ Output }_2 \| \ldots \tag{14}$$

Although TDEA-CTR DRBG ensures compatibility with older systems, its small 64-bit block size makes it vulnerable to modern cryptographic techniques. It is not recommended for post-quantum cryptography. There are inherent vulnerabilities in block cipher DRBGs due to their reliance on pseudo random permutations. The unpredictability of outputs may be jeopardized since pseudo random permutations, in contrast to random oracles, do not meet independence. Furthermore, security is impacted by the output's length in relation to the block size, which typically makes block cipher DRBGs less appropriate for cryptographic applications. Although resistance to key recovery

attacks is a must, the basic drawbacks of pseudo random permutations still exist, even with secure block ciphers like AES [26].

5. Comparison of DRBGs in post quantum cryptography

NIST SP 800-90A defines DRBG techniques that are claimed to be backtracking and prediction resistant. Backtracking resistance makes ensuring that earlier outputs are safe even in the event that the internal state is corrupted. Even if the internal state was known beforehand, prediction resistance makes sure that the generator's outputs cannot be anticipated after reseeding with enough entropy. These characteristics are essential to the security assurances that the standard offers [27].

The advantages, disadvantages, and Post Quantum Cryptography appropriateness of these five DRBGs are compiled in Table 1.

Table 1
Comparison of DRBGs

DRBGs Type	Strengths	Weaknesses	Post Quantum Cryptography Suitability
HMAC-DRBG	Robust security, quantum-resistant	Performance overhead	Excellent (Keyed structure ensures state security)
Hash-DRBG	High efficiency, simple implementation	Security assumptions, no keyed protection	Moderate to Good (Strong hash function needed)
KHF-DRBG	Balanced efficiency, DRBG	Lacks HMAC-level state protection	Good (Efficient with modern hash functions)
AES-CTR	Strong security, performance efficiency	Implementation complexity	Excellent (AES-256 is quantum-resistant)
TDEA-CTR	Legacy system compatibility	Limited security, performance constraints	Poor (Not suitable for Post Quantum Cryptography)

The design of DRBGs must rely on primitives that are safe even in the presence of quantum adversaries in order to protect them against quantum threats. Selecting the best approach can be aided by assessing the various DRBG types and their applicability for post-quantum cryptography.

These DRBGs use hash methods like SHA-256 or SHA-3 to produce random bits. They depend on the underlying hashing function's cryptographic security. Newer hash functions like SHA-3 (based on the Keccak algorithm) or quantum-resistant hash schemes can be utilized to offer higher security, even if classical hash functions like SHA-256 may be susceptible to quantum assaults. Hash-based DRBGs have several benefits, such as excellent performance, ease of use, and wide standardization. Because hash functions can be readily improved or altered with little overhead, they are especially well-suited for applications that need a balance between security and performance.

These DRBGs (AES-CTR, TDEA-CTR) generate random bits by using feedback modes of symmetric block ciphers, including AES. DRBGs based on AES are specified by NIST SP 800-90A as one of its standards. Grover's technique can decrease the effective security of AES by half the key size, despite the fact that it is somewhat resistant to quantum attacks. AES-256 can offer a security level equal to 128 bits in a quantum setting. Although block cipher-based DRBGs are more extensively used and may be accelerated by hardware, their effective security against quantum assaults is lower than that of other DRBG kinds. Systems that need legacy compatibility or have hardware-optimized implementations are best suited for these DRBGs.

The cryptographic protocol, performance needs, and available resources are some of the variables that influence the choice of DRBG. In light of the assessment:

- Hash-Based DRBGs provide a useful and adaptable solution, especially when SHA-3 or other quantum-resistant hash functions are employed. For general-purpose applications, they are perfect.

- DRBGs based on block ciphers are appropriate for systems that require conformance to current standards. AES-256 offers a respectable degree of quantum resistance, but it might not be the most resilient choice in the long run.

6. Conclusions and future plans

Although they may appear to be a minor component of cryptography, random number generators are among its most crucial elements. Without safe randomization, even the most robust cryptographic systems might become susceptible due to predictable encryption keys. Because they guarantee that cryptographic keys, signatures, and secure communications stay unexpected and safe from assaults, deterministic random bit generators, or DRBGs, are crucial.

This article examined the use of five DRBG techniques for post-quantum cryptography: Hash-DRBG, HMAC-DRBG, KHF-DRBG, AES-CTR DRBG, and TDEA-CTR DRBG. Although each has advantages, HMAC-DRBG and AES-CTR DRBG are the most suitable options for this idea. Because it employs a keyed technique to prevent attackers attempting to reconstruct its internal state, HMAC-DRBG is very robust. Another great choice is AES-CTR DRBG, particularly when used AES-256, because to its proven security and resilience to quantum assaults.

For systems that value efficiency, Hash-DRBG and KHF-DRBG are suitable substitutes; nevertheless, they fall short of HMAC-DRBG in terms of security. However, TDEA-CTR DRBG is no longer in use since it is more susceptible to contemporary assaults due to its lower 64-bit block size.

Threats are evolving along with technology. Many of the encryption techniques used today might eventually be broken by quantum computers, so it's critical to plan ahead. A robust, secure DRBG is an essential component of the post-quantum cryptography systems that cryptographers are now developing. Selecting HMAC-DRBG or AES-CTR DRBG is the ideal strategy for enterprises and developers creating safe systems to keep ahead of emerging security threats.

Acknowledgement

This work was supported by the Shota Rustaveli National Foundation of Georgia (YS-24-3272).

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] D. R. L. Brown, Breaking RSA may be as difficult as factoring, Journal of Cryptology 29.1 (2016) 220–241.

[2] M. Sharma, et al., Leveraging the power of quantum computing for breaking RSA encryption, Cyber-Physical Systems 7.2 (2021) 73–92.

[3] V. Kharchenko, I. Chyrka, Detection of airplanes on the ground using YOLO neural network, in: Proceedings of 17th International Conference on Mathematical Methods in Electromagnetic Theory (MMET), IEEE, Kyiv, Ukraine, 2018, pp. 294–297.

[4] G. Alagic, et al., Status report on the third round of the NIST post-quantum cryptography standardization process, NIST Interagency/Internal Report, National Institute of Standards and Technology, 2022. URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=934458.

[5] M. Iavich, T. Kuchukhidze, R. Bocu, A Post-Quantum Digital Signature Using Verkle Trees and Lattices. Symmetry 15(12) (2023) 2165.

[6] M. Iavich, T. Kuchukhidze, Digital Signature Design Using Verkle Tree, IVUS (2023) 83-91.

[7] W. Kan, Analysis of underlying assumptions in NIST DRBGs, Cryptology ePrint Archive (2007).

[8] J.S. Al-Azzeh, M. Al Hadidi, R.S. Odarchenko, S. Gnatyuk, Z. Shevchuk, Z. Hu, Analysis of self-similar traffic models in computer networks, International Review on Modelling and Simulations 10(5) (2017) 328–336. doi: 10.15866/iremos.v10i5.12009.

[9] L. Crocetti, S. Di Matteo, P. Nannipieri, L. Fanucci, S. Saponara, Design and test of an integrated random number generator with all-digital entropy source, Entropy 24(2) (2022) 139.
[10] L. Chen, et al., Report on post-quantum cryptography. Vol. 12. Gaithersburg, MD, USA: US Department of Commerce, National Institute of Standards and Technology, 2016.
[11] H. Davis, M. D. Green, N. Heninger, K. Ryan, A. Suhl, On the possibility of a backdoor in the Micali-Schnorr generator, in: Proceedings of IACR International Conference on Public-Key Cryptography, Springer Nature, Cham, Switzerland, 2024, pp. 352–386.
[12] E. Barker, J. Kelsey, Nist special publication 800-90a: Recommendation for random number generation using deterministic random bit generators (2012).
[13] National Institute of Standards and Technology. Recommendation for random number generation using deterministic random bit generators (NIST SP 800-90A Rev. 1). U.S. Department of Commerce, 2015. URL: https://doi.org/10.6028/NIST.SP.800-90Ar1.
[14] E. Barker, J. Kelsey, K. McKay, A. Roginsky, M. Sönmez Turan, Recommendation for random bit generator (rbg) constructions (3rd draft) (No. NIST Special Publication (SP) 800-90C (Draft)). National Institute of Standards and Technology (2022).
[15] M. J. Fischer, M. Paterson, E. Syta, On backtracking resistance in pseudorandom bit generation. Technical Report TR-1466, 2012. URL: http://cs.yale.edu/publications/techreports/tr1466.pdf.
[16] V. Tkachuk, Y. Yechkalo, S. Semerikov, M. Kislova, Y. Hladyr, Using mobile ICT for online learning during COVID-19 lockdown, Communications in Computer and Information Science, 1308 (2021) 46–67. doi: 10.1007/978-3-030-77592-6_3.
[17] Y. Dodis, et al., A formal treatment of backdoored pseudorandom generators, in: Proceedings of Advances in Cryptology--EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, 2015, Proceedings, Part I 34, Springer, Berlin, 2015, pp. 101–126.
[18] J. Woodage, D. Shumow, An analysis of NIST SP 800-90A, in: Proceedings of Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Part II, Springer, Berlin, 2019, pp. 151–180.
[19] National Institute of Standards and Technology. Recommendation for random number generation using deterministic random bit generators (NIST SP 800-90A Rev. 1). U.S. Department of Commerce, 2015. URL: https://doi.org/10.6028/NIST.SP.800-90Ar1.
[20] N. Ferguson, B. Schneier, T. Kohno, Cryptography engineering: design principles and practical applications, John Wiley & Sons, NY, 2011.
[21] Y. Nir, A. Langley, RFC 8439: ChaCha20 and Poly1305 for IETF Protocols, 2018.
[22] P. Kietzmann, T. C. Schmidt, M. Wählisch, A guideline on pseudorandom number generation (PRNG) in the IoT, ACM Computing Surveys 54(6) (2021) 1–38.
[23] J. Kelsey, Five drbg algorithms based on hash functions and block ciphers, in: Presentation at NIST Random Number Generation Workshop (2004).
[24] V. T. Hoang, Y. Shen, Security Analysis of NIST CTR-DRBG, in: Proceedings of Annual International Cryptology Conference, Springer, Cham, 2020, pp. 218–247.
[25] S. Gnatyuk, T. Zhmurko, P. Falat, Efficiency Increasing Method for Quantum Secure Direct Communication Protocols, in: Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2015), IEEE, Warsaw, Poland, 2015, Vol. 1, pp. 468–472.
[26] M. Iavich, S. Gnatyuk, E. Jintcharadze, Yu. Polishchuk, R. Odarchenko, Hybrid Encryption Model of AES and ElGamal Cryptosystems for Flight Control Systems, in: Proceedings of the 2018 IEEE 5th International Conference on Methods and Systems of Navigation and Motion Control, 2018, IEEE, Kyiv, Ukraine, pp. 229–233.
[27] M. Iavich, T. Kuchukhidze, S. Gnatyuk, A. Fesenko, Novel certification method for quantum random number generators, International Journal of Computer Network and Information Security 13(3) (2021) 28–38.