Statistical Assessment of two Rekeying Mechanisms applied to the Generation of Random Numbers

Adrián Alfonso Peñate, Daymé Almeida Echevarria, Laura Castro Argudín

***Abstract*—The block ciphers modes of operation with internal rekeying mechanisms, used during the encryption of a message to increase their security, have been a subject of analysis in recent years. In this paper, we will analyze the randomness of the sequences generated by two of these modes of operation, which also will be used in the generation of pseudo-random numbers.**

***Tóm tắt*—Trong những năm qua, các chế độ mã hóa khối hoạt động với cơ chế tạo lại khóa bên trong, được sử dụng trong quá trình mã hóa tin nhắn đã trở thành đối tượng nghiên cứu trong những năm qua để tăng tính bảo mật. Trong bài báo này, chúng tôi sẽ phân tích tính ngẫu nhiên của chuỗi được tạo bởi hai trong số các chế độ này, mà cũng sẽ được sử dụng trong việc tạo ra các số giả ngẫu nhiên.**

Keywords—rekeying; randomness.

Từ khóa—tạo lại khóa; ngẫu nhiên.

I. Introduction

During the encryption process of a message with a mode of operation, the values of the pseudo-random permutation determined by the secret key are tours. Then, if the length of the message exceeds the lifetime of the key, the encryption process can be seriously compromised [1]. This determines for each mode of operation the maximum number of blocks that can be processed with the same key, and therefore, a message of length greater than this amount cannot be encrypted without changing the key first. Bounds for the life time of a key in some modes of operation can be seen in [2], [3].

Rekeying mechanisms are used to encrypt a large amount of data with a specific mode of operation, with the peculiarity that for every certain amount of processed blocks, a new key is generated and used to process the following blocks of the message used to process the following blocks of the message. This new approach, which is well-studied and recommended in the last few years [4]-[12], ensures that encryption remains secure even when the message is very large, by changing the key periodically. It can be applied on three levels: on the block cipher level (fresh rekeying), on the block cipher mode of operation level (internal rekeying), and on the message processing level (external rekeying).

In our work, we will analyze the randomness properties of the outputs produced by the pseudo-random number generator Fortuna [13], which uses the block cipher mode of operation CTR with an internal rekeying mechanism proposed by their own designers. Then we will perform a similar analysis replacing the original proposal for the recently standardized in the Russian Federation mode of operation CTR-ACPKM [see 14 for example]. For the analysis we intend to perform, we will use the NIST statistical test suite [15].

II. How Fortuna works

A. The original Fortuna

Generation of keys, initialization vectors, seeds, and other kinds of sequences with random behavior play a fundamental role in cryptography. For this reason, a pseudo-random number generator that can be used in real cryptographic applications has become an important tool. Fortuna [13] is that kind of tool due to the relevance of its design and to the strength of the underlying cryptographic primitives chosen by their designers [16]-[18].

This particular generator has three specific components: the accumulator (to extract and pool entropy from some external sources), the generator (to produce the requested sequences), and the seed file control (to produce random data

This manuscript is received on May 22, 2020. It is commented on May 28, 2020 and is accepted on July 14, 2020 by the first reviewer. It is commented on December 8, 2020 and is accepted on December 26, 2020 by the second reviewer.

at any time). In this paper, we only focus on the generator due to the character of our investigation.

The mechanism that produces the sequences of interest consist of the block cipher standard AES with 256 bits of key length working in the mode of operation CTR. The input is one random counter of 128 bits and one random key of 256 bits, with which we can only produce 1MB of pseudo-random data.

If the request of a user exceeds this amount, then the next two blocks are generated and used as a new key for the generation of the following pseudo-random data. This principle of Fortuna constitutes an internal rekeying mechanism, offering greater security than if we always use the mode CTR with the initial key [13].

More detailed, the generator of Fortuna only produces $N = 65536$ blocks (AES encryptions) with the same key K^i for the requested sequence

$$E_{K^i}(C_1), \cdots, E_{K^i}(C_N).$$

After that, two new blocks are produced to compute the new key for the next data

$$K^{i+1} = E_{K^i}(C_{N+1}) || E_{K^i}(C_{N+2})$$

and the actual counter is not restarted ($C_{N+3} = C_1$) to encrypt with the next key K^{i+1}.

B. Fortuna working with CTR-ACPKM

Recently, the mode of operation with internal rekeying mechanism CTR-ACPKM was adopted in the Russian Standardization System. The main characteristic of this mode is that, for every certain amount of blocks processed during the encryption of a same message, another master key is derived to the previous key.

That message is split into sections, each one of them composed of the same amount of blocks (except for the last section), and with different keys for the encryption process. The parameter N, the section length, is established independently taking into account the specific protocol where it will be used, depending on the capacity of the system and the lifetime of the keys.

Given P one m blocks message of n bits each, it splits into $l = \lceil m/N \rceil$ sections denoted as P^1, $\cdots$, P^l, then we have $P^i \in \{0,1\}^{nN}$ for $i \in \{1, \cdots, l-1\}$ and $P^l \in \{0,1\}^r, r \leq nN$.

The blocks of the first section are encrypted using the mode of operation CTR with the section key $K^1 = K$. And consequently, the blocks of the i^{th} section are encrypted using the same mode of operation with the section key

$$K^{i+1} = E_{K^i}(D_1) \| \cdots \| E_{K^i}(D_N).$$

Here $D_1, D_2, \cdots, D_N \in \{0,1\}^n$ are different constants established in the standard, aimed to prevent collisions between the cycles of the permutations determined by the section keys, and the counter block is never changed between sections. A better description of the mode of operation with internal rekeying mechanism CTR-ACPKM can be seen in [10], [11].

The only difference of the internal rekeying in CTR-ACPKM with respect to the internal rekeying in Fortuna lies in the fact that, in the first case constants are always encrypted to produce the next section key, while in the second case the counter block is encrypted to produce the next section key, then the mode CTR-ACPKM may replace the mode CTR used in Fortuna. We implemented Fortuna using CTR-ACPKM such that the section keys are replaced once there are 1MB of pseudo-random data generated.

III. About the NIST test suite

Statistical assessments have been performed for several cryptographic primitives, including random and pseudo-random number generators [20]-[25].

In [16], a statistical evaluation of random data generated by Fortuna was already performed using the Diehard test suite [19]. In this case, fifty files of 12MB each were generated and analyzed in both software and hardware implementations, showing a highly random behavior.

In this paper, we will analyze the statistical behavior of random data generated by Fortuna, but using the NIST statistical test suite [15]. It contains 15 tests for randomness, each of which establishes a comparison between the pseudo-random sequences generated and a "truly" random sequence.

Since the first publication of the NIST statistical test suite in 2000, it has been well studied through the years [26]-[32], and several corrections have been made. For the evaluation of the sequences generated in this paper, we will use the latest version of this set of tests [15], proposed

in 2010, which continues to be a powerful tool for randomization assessments.

Since 2010 until now, there have been a number of publications [35]-[39] showing inaccuracies in the NIST SP 800-22, but NIST has not shown that they will update this test suite. Furthermore, there have been many works [40]-[44] showing the correlations of several tests in this test suite. Therefore, the selection of the most efficient tests is a research direction in the future. We do not consider this issue in the scope of this article.

We don't describe the internal tests or the complete suite at all. A wise description of the test, the suite, or even the interpretation of the results can appear in any one of the papers cited before, including [15].

To emit a reasonable criteria about the samples, each one must contain a sufficient amount of data. For every sample, we are generating a file with 1000 sequences of 1000000 bits each (≈120 MB), and something less than in [16], we are analyzing twenty files for every one of the rekeying mechanisms of interest. For every file, we check the proportion and the uniformity of the p-values, the two approaches recommended in [15]; and also we use a useful approach utilized in [16] to establish a comparison, where the range of possible proportions in this case is scored.

A. Practical results

We implement Fortuna using the language C# in Visual Studio 2017, and run for random data in a PC Intel(R) Core (TM) i3-4130 CPU (3.40GHz, 4GB-RAM and OS-Windows 7x64). A pseudo-code of the generator of Fortuna is showing in Appendix A.

With respect to NIST statistical test suite, we excluded in the analysis: Random Excursions Test and Random Excursions Variant Test, this way we get 162 testing for every file distributed in the following tests:

Statistical test	Number of testing
frequency	1
block-frequency	1
cumulative-sums	2
runs	1
longest-run	1
rank	1
fft	1
nonperiodic-templates	148
overlapping-templates	1
universal	1
apen	1
serial	2
linear-complexity	1

To make the statistical assessments, we took into account:

[**Aspect 1.**] As indicated in [15], for a sample of size 1000 and the significance level $\alpha = 0.01$, the range for acceptable proportions (number of p-values greater or equal than 0.01 divided by 1000) is (0.9805608 ; 0.9994392).

We carefully observe each of the 162 proportion values obtained in the 20 files generated with every rekeying mechanism, and conclude that only a few of those values are out of range. More specifically, only 14 proportions are out of range in case of Fortuna for an average of 0.7 faults per file, and only 16 proportions are out of range in case of Fortuna with CTR-ACPKM for an average of 0.8 faults per file, which we interpret as a well random behavior in both cases.

Two graphs for every variant of Fortuna, original and with CTR-ACPKM, are shown in the Appendix B and C.

[**Aspect 2.**] The idea behind the analysis of [16] is split the range of possible p-values resulting from the Diehard statistical test suite, and score each of the sub-range considering a meaning of goodness. A similar assignment is done in our case but scoring the possible range of proportions.

The next table shows how we did this.

Proportion range	Meaning of goodness	Assigned score
[0.985 ; 0.995]	Desirably good	0
[0.980561; 0.985) (0.995 ; 0.999439]	Almost not good but still good	2
[0 ; 0.980561) (0.999439 ; 1]	Not good	4

Only for uniformity at the check of that criteria, we excluded of the NIST suite: Random Excursions Test and Random Excursions Variant Test, since both analyze only an arbitrary amount of sequences which satisfy some conditions. A similar approach is adopted in [33], [34].

This way, with the remaining tests, we evaluate 162 proportion values per file, and the accumulated score gives us another idea of goodness. The possible best score is 0 and the possible worst score is 648 for every file.

A comparative analysis between the twenty files generated with Fortuna and the twenty files generated with the proposed variant is shown below. All results have been tabulated and are shown in the Appendix D, E. These results do not throw significant differences between both variants, and we appreciate a random behavior since the final score of every file is very low.

Generator	Score
Fortuna (original)	12 – best score 38 – worst score
Fortuna-CTR-ACPKM	8 – best score 40 – worst score

[**Aspect 3.**] The other approach according to [15] is to examine the distribution of the p-values in the range [0 ; 1], so this interval is divided into ten equal sub-intervals and then the number of p-values that lie within each sub-interval are counted. The optimal case for randomness is that 100 p-values fall in each sub-interval.

The NIST test suite returns a p-value of p-values in every testing, corresponding to the chi square test for the same ones. If that p-value is greater than 0.0001, then the sequence can be considered to be uniformly distributed, in all the checks carried out this was satisfactorily fulfilled.

B. Final considerations

Both the original Fortuna and Fortuna with the internal rekeying mechanism CTR-ACPKM, have shown that the generated sequences have a random behavior. The performing tests do not offer sufficient reasons to establish a significant comparison between both types of generated sequence actually, using the criteria established in [16] the sequences generated by Fortuna have similar randomness than the other kind of generated sequences.

We recommend the use of Fortuna, and so of Fortuna with CTR-ACPKM, in real and practical applications. Nevertheless, it should be taken into account that each generated sequence must be carefully analyzed for randomness before being used although similar results are expected.

IV. CONCLUSION

Generation of keys, initialization vectors, seeds, and other kinds of sequences with random behavior play a fundamental role in cryptography. Thus, a pseudo-random number generator cryptographically strong with proven pseudo-random behavior is a powerful tool in this area.

The used simple technique is the implementation of one block cipher algorithm using one mode of operation. This way, good random behavior is achieved if the underlying encryption method has good diffusion properties.

The pseudo-random number generator Fortuna is based on this principle, using the standard AES and the mode of operation CTR as underlying primitives, and one internal rekeying mechanism proposed by their own designers.

In this paper, we have analyzed the random behavior of the sequences generated by Fortuna as well as a variant that uses another internal rekeying mechanism that included in the Russian Standardization System and named CTR-ACPKM. In both cases, twenty files of 120 MB were generated, all of them show a good random behavior.

As reference we used the NIST test suite, although we have removed from it two tests and have included an extra analysis technique performed previously, but using the Diehard statistical test suite. In terms of this approach, the sequences generated by Fortuna seem as random as the sequences generated by the variant of Fortuna, although the difference observed is really small to make a comparison between the two kinds of sequences.

To conclude this paper, we propose the application of the two analyzed pseudo-random number generators, Fortuna and Fortuna with internally rekeying mechanism CTR-ACPKM, in real and practical applications, as long as the generated sequences in both cases are previously analyzed and show a random behavior similar to the examples shown here.

V. APPENDIX

A. Pseudo-code of Fortuna with both, original and new rekeying mechanisms

Function: Pseudo-random generator Fortuna

Input: Counter C with 128 bits (random)

Key K with 256 bits (random)

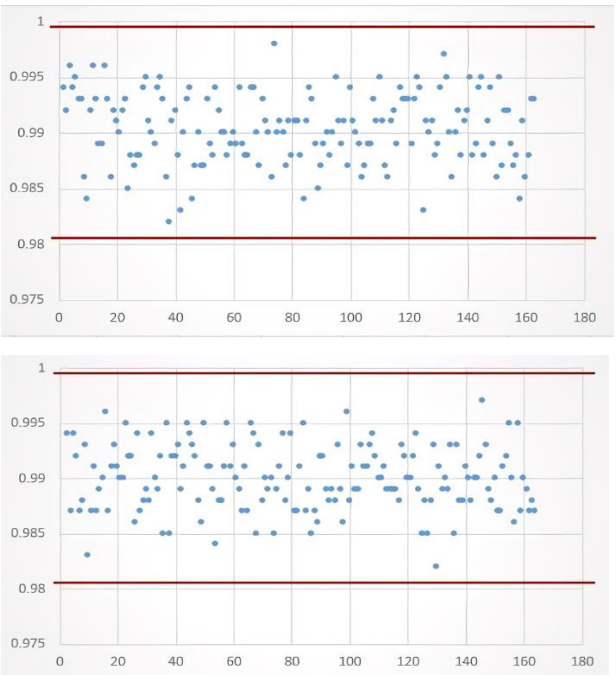
Number of bytes n to generate

Procedure:

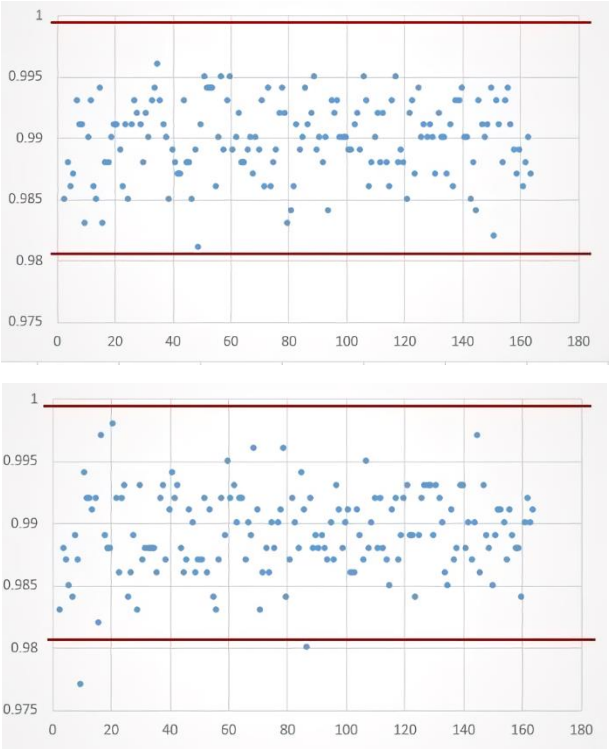
1. Define an empty string r
2. Define $k = \lceil n/16 \rceil$
3. For i from 1 to $k - 1$ do
4. $r = r||E_K(C)$
5. $C = C + 1$
6. $n = n - 16$
7. If $i = 2^{16}$ then
8. Change the key K using the internal rekeying method of the original Fortuna or the ACPKM method
9. $C = C + 1$
10. End If
11. $i = i + 1$
12. End For
13. $r = r||$first-n-bytes-$(E_K(C))$

Return: Pseudo-random string r of n bytes

B. Proportion analysis of two samples generated by the original Fortuna



C. Proportion analysis of two samples generated by Fortuna with CTR-ACPKM



D. Score analysis of the 20 samples generated by Fortuna

24	14	32	12	26	26	38	30	22	20
22	32	34	26	24	30	24	32	26	22

E. Score analysis of the 20 samples generated by Fortuna with CTR-ACPKM

18	40	34	18	36	28	34	12	34	30
26	24	24	32	30	8	18	28	8	32

REFERENCES

[1] Abdalla, M. and Bellare, M. "Increasing the lifetime of a key: a comparative analysis of the security of re-keying techniques." International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, 2000.

[2] Lavrikov, I. and Shishkin, V. "Within a Friend Zone: How Far Can We Proceed with Data Encryption not Getting Out." 7th Workshop on Current Trends in Cryptology. Suzdal, Russia, 2018.

[3] Lavrikov, I. and Shishkin, V. "How much data may be safely processed on one key in different modes?" Mathematical Aspects of Cryptography. Vol. 10(2) 2019.

[4] Medwed, M., Standaert, F., Großschädl, J. and Regazzoni, F. "Fresh re-keying: Security against side-channel and fault attacks for low-cost devices." International Conference on Cryptology in Africa. Springer, Berlin, Heidelberg, 2010.

[5] Abdalla, M., Belaïd, S. and Fouque, P. "Leakage-resilient symmetric encryption via re-keying." International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, 2013.

[6] Dobraunig, C., et al. "Towards fresh and hybrid re-keying schemes with beyond birthday security." International Conference on Smart Card Research and Advanced Applications. Springer, 2015.

[7] Gueron, S. and Yehuda L. "Better bounds for block cipher modes of operation via nonce-based key derivation." Proceedings of the 2017 Conference on Computer and Communications Security. ACM, 2017.

[8] Ahmetzyanova, R., et al. "Increasing the Lifetime of Symmetric Keys for the GCM Mode by Internal Re-keying." IACR Cryptology ePrint Archive, 697, 2017.

[9] Goncharenko, K., Alekseev, E. and Marshalko, G. "Provably secure counter mode with related key-based internal rekeying." 7th Workshop on Current Trends in Cryptology. Suzdal, Russia, 2018.

[10] Akhmetzyanova, L., Alekseev, K. and Smyshlyaev, V. "Security bound for CTR-ACPKM internally re-keyed encryption mode." 2018.

[11] Akhmetzyanova, L., et al. "Security bounds for standardized internally re-keyed block cipher modes and their practical significance." 7th Workshop on Current Trends in Cryptology. Suzdal, Russia, 2018.

[12] Akhmetzyanova, L., et al. "Practical significance of security bounds for standardized internally re-keyed block cipher modes" Mathematical Aspects of Cryptography. Vol. 10(2) 2019.

[13] Ferguson, N., Schneier, B. and Kohno, T. "Cryptography Engineering. Design, Principles and Practical Applications." Wiley Publishing Inc., 2010. Chapter 9. (Second version of "Practical Cryptography." Wiley Publishing Inc., 2003.)

[14] CryptoPro. "Re-keying Mechanisms for Symmetric Keys draft-irtf-cfrg-re-keying-00." Internet-Draft, 2017.

[15] National Institute of Standards and Technology. "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications." NIST Special Publication 800-22, 2010.

[16] McEvoy, R., Curran, J., Cotter, P. and Murphy, C. "Fortuna: cryptographically secure pseudo-random number generation in software and hardware." 2006.

[17] Akbar, M. and Zulkifl, M. "Fuzzy-Fortuna: A fuzzified approach to generation of cryptographically secure pseudo-random numbers." IEEE International Multitopic Conference. IEEE, 2008.

[18] Yevgeniy, D., Shamir, A., Stephens-Davidowitz, N. and Wichs, D. "How to eat your entropy and have it too: Optimal recovery strategies for compromised RNGs." Algorithmica, 79 (4), 2017.

[19] Marsaglia, G. "Diehard Battery of Tests of Randomness." 1985.

[20] Soto, J. "Randomness testing of the advanced encryption standard candidate algorithms." National Institute of Standards and Technology, 1999.

[21] El-Fotouh, M. and Diepold, K. "Statistical Testing for Disk Encryption Modes of Operations." IACR Cryptology ePrint Archive, 362, 2007.

[22] Santoro, R., Sentieys, O. and Roy, S. "On-the-fly evaluation of FPGA-based true random number generator." IEEE, 2009.

[23] Doganaksoy, A. et al. "Cryptographic Randomness Testing of Block Ciphers and Hash Functions." IACR Cryptology ePrint Archive, 564, 2010.

[24] Chen, X., et al. "Evaluation of ECG random number generator for wireless body sensor networks security." 5th International Conference on BioMedical Engineering and Informatics. IEEE, 2012.

[25] Zubkov, A. and Serov, A. "Testing the NIST Statistical Test Suite on artificial pseudorandom sequences." Mathematical Aspect of Cryptography, 10(2), 2019.

[26] Kim, S., Ken U. and Hasegawa, A. "Corrections of the NIST statistical test suite for randomness." 2004.

[27] Suciu, A., et al. "Parallel implementation of the NIST statistical test suite." Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing. IEEE, 2010.

[28] Zhu, S., et al. "More powerful and reliable second-level statistical randomness tests for NIST SP 800-22." International Conference on the Theory and

Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, 2016.

[29] Chugunkov, I., Prokofiev, A. and Strelchenko, P. "The optimization of statistical tests for pseudorandom number generators." IEEE, 2016.

[30] Simion, Emil, and Paul Burciu. "A Note On the Correlations Between NIST Cryptographic Statistical Tests Suite." 2019.

[31] Burciu, P. and Simion, E. "A Systematic Approach of NIST Statistical Tests Dependencies." Journal of Electrical Engineering, Electronics, Control and Computer Science. 5(1), 2019.

[32] Mishra, P., Nandan, B. and Gaba, N. "An Efficient and Compact Reformulation of NIST Collision Estimate Test." IACR Cryptology ePrint Archive, 481, 2019.

[33] Okutomi, H., Nakamura, K., and Aihara, K. "A study on rational judgment method of randomness property using NIST randomness test (NIST SP. 800-22)." IEICE Trans. A, 93 (1), 2010, pp. 11-22.

[34] Iwasaki, A. "Analysis of NIST SP800-22 focusing on randomness of each sequence." JSIAM Letters, Vol. 10, pp. 1-4, 2018.

[35] T. Yuichi, H.M., K. Toshinari, W. Norio, S. Takakazu, The Suggestion of Corrected Non-overlapping Template Matching Test [in Japanese]. Technical report of IEICE., 2010.

[36] Pareschi, F., R. Rovatti, and G. Setti, On statistical tests for randomness included in the NIST SP800-22 test suite and based on the binomial distribution. IEEE Transactions on Information Forensics and Security, 2012. 7(2): pp. 491-505.

[37] Takeda, Y., et al., Modified Non-overlapping template matching test and proposal on setting template. 2014. 27(1): pp. 49-60.

[38] Okada, H. and K. Umeno, Randomness evaluation with the discrete Fourier transform test based on exact analysis of the reference distribution. IEEE Transactions on Information Forensics Security, 2017. 12(5): pp. 1218-1226.

[39] Iwasaki, A. and K. Umeno, A new randomness test solving problems of Discrete Fourier Transform Test. arXiv preprint arXiv:.08218, 2017.

[40] DOĞANAKSOY, ALİ, et al. "Mutual correlation of NIST statistical randomness tests and comparison of their sensitivities on transformed sequences." Turkish Journal of Electrical Engineering & Computer Sciences 25.2 (2017): 655-665.

[41] Doğnaksoy, A., Barış Ege, and Köksal Muş. "Extended results for independence and sensitivity of NIST randomness tests." Information Security and Cryptography Conference, ISC Turkey. 2008.

[42] Jorge Augusto Karell-Albo, Carlos Miguel Legón-Pérez, Evaristo José Madarro-Capó, Omar Rojas, and Guillermo Sosa-Gómez. Measuring independence between statistical randomness tests by mutual information. Entropy, 22(7):741, 2020.

[43] Koçak, Onur. "A unified evaluation of statistical randomness tests and experimental analysis of their relations." 2016.

[44] Sulak, Fatih, et al. "On the independence of statistical randomness tests included in the NIST test suite." Turkish Journal of Electrical Engineering & Computer Sciences 25.5. 2017. pp. 3673-3683.

ABOUT THE AUTHOR

Adrián Alfonso Peñate

Workplace: Institute of Cryptography. University of Havana.

Education: Graduated of Mathematics in 2014; received his Master's degree in 2018.

Current research direction: design and analysis of block ciphers.

Daymé Almeida Echevarria

Workplace: Institute of Cryptography. University of Havana.

Education: Graduated of Mathematics in 2013; currently aspires to pursue her Master's degree.

Current research direction: design and analysis of rekeying mechanisms for block ciphers modes of operation.

Laura Castro Argudín

Workplace: Institute of Cryptography. University of Havana.

The education process: Graduated of Cryptology in 2019.

Current research direction: design and analysis of pseudo-random number generators.