# Literature Review: Randomness Test Suites for Encryption Algorithms

Luke Y. Tao

Massanutten Regional Governor’s School

Research

Ms. Klus

January 29, 2021

# Introduction

Cybersecurity is an important part of everyday life in the information age. The personal information and privacy of people must be protected as they browse, bank, and buy on the web. This is accomplished using encryption: a process in which the original information, known as plaintext, is converted into an unreadable form called ciphertext using a unique and random string called a key. One of the many properties that help determine the strength of an encryption algorithm is the randomness of the ciphertext. This property is called entropy and increases the strength of an encryption algorithm because it makes the relationship between the key and ciphertext more complex (Patil et al., 2016). Testing randomness may initially seem impossible. However, randomness, somewhat paradoxically, has certain properties and expected behavior (Gevorkyan et al., 2020). In order to test each of these properties, test suites are typically used. The randomness of a sequence or generator is determined by the number of tests that the sequence or generator passes (Gevorkyan et al., 2020). Each experimental result is compared to its theoretical counterpart to see how close it is to the expected result. There is little agreement among researchers when it comes to which randomness test suite to use when testing block ciphers. This study aims to compare the results between the more popular NIST test suite and CryptoStat, a new test suite designed specifically for block ciphers.

# NIST

From 1997 to 2000, the National Institute of Standards and Technology (NIST) held a competition to decide on a new Advanced Encryption Standard (AES) to replace the deprecated Data Encryption Standard (National Institute of Standards and Technology, 2016). One of the criteria used to evaluate the encryption algorithms was the randomness of the ciphertext they produced (Soto, 1999). To help evaluate the submitted encryption algorithms, NIST created the NIST Test Suite which has been updated twice, most recently in April of 2010 (Rukhin et al., 2010). This test suite was actually designed to test random and pseudorandom number generators (Rukhin et al., 2010). However, the test suite was still used. This is likely because NIST reasoned that how the input was generated should not matter, especially since their ideal ciphertext was indistinguishable from truly random text (Soto, 1999). This test suite is still rather popular: in 2014, Ahmad and Younis used the NIST test suite in their study on the viability of using non-cryptographic hashes as random number generators. Hashes are algorithms that take an input of any size and output a string of fixed size. Similar to encryption algorithms, the randomness of its output is an important property of a hash function (Ahmad & Younis, 2014). In their study, they combined the hashed output of 2,744,529 URLs in ASCII format. Although they did not specify how they were combined to create sufficient data, it is likely that they were simply concatenated without any special procedures. This is very similar to how block ciphers encrypt the plaintext in chunks before combining them in some way.

# Modes of Operation

AES, 3DES, and Blowfish, some of the most popular encryption algorithms in use today, are all block ciphers (Patil, et al., 2016). Unlike a stream cipher which encrypts continuously, a block cipher encrypts in chunks typically 128-bits in size. This causes a few issues for both the NIST test suite and DieHarder which both prefer arbitrarily long, continuous streams of 1s and 0s. To solve this issue, modes of operation are used to connect the different chunks.

In Electronic Codebook, the plaintext is split into chunks and then each chunk is encrypted before the ciphertext is reassembled without changing the order of the chunks (Dworkin, 2001). This mode of operation was used on eight out of nine datasets during the NIST competition (Soto, 1999). While this method is very fast, it raises quite a few issues when it comes to security and randomness testing. Since each chunk of the plaintext is encrypted independently, chunks that contain identical information will be encrypted to create identical ciphertext (Dworkin, 2001). A classic example of this is the ECB penguin, in which a simple image of a penguin is encrypted, but because many of the chunks contain identical colors, the encrypted image did not conceal enough information and still shows the pattern of a penguin.

Another common mode of operation is Cipher Block Chaining Mode (CBC) which also splits the plaintext into chunks before encrypting them, however the ciphertext for each chunk is used to perform an exclusive or (XOR) operation with the plaintext of the next chunk before it is encrypted (Dworkin, 2001). The first chunk is XORed with an initialization vector (IV). This makes encryption much slower as each chunk’s encryption is dependent on the one before it.

Output Feedback Mode (OFB) makes the block cipher into a stream cipher. The IV is encrypted using the cipher to create an output block. This output block is then XORed with the plaintext to create the ciphertext (Dworkin, 2001). The output block is also encrypted again to create a new output block which is then XORed with the next part of the plaintext (Dworkin, 2001). This process is repeated until the entire plaintext is encrypted.

Counter Mode (CTR) also makes a block cipher into a stream cipher. Instead of an IV it uses a counter. The counter is encrypted and then XORed with the plaintext to create the ciphertext (Dworkin, 2001). The counter is then incremented before being encrypted again and then XORed with the next portion of the plaintext (Dworkin, 2001). This process is repeated until the entire plaintext is encrypted. This mode allows for encryption or decryption in parallel because each part is independent of the others. This makes the speed basically on par with ECB which also allows for encryption or decryption in parallel, but is much less secure.

When a block cipher is converted into a pseudorandom number generator using these methods, it is unclear whether the randomness of the block cipher is being tested or the randomness of the mode of operation (Kaminsky, 2019). To solve this problem, Kaminsky, a professor at the Rochester Institute of Technology, created CryptoStat, which instead of converting block ciphers into pseudorandom number generators, directly tests the mapping of the block cipher. A map in the context of computer science can be thought of using IDs. Each ID corresponds to a certain person. In this case, the ID is the plaintext and the person is the ciphertext. By directly testing the block cipher mapping, Kaminsky aims to circumvent the issue that modes of operation, especially ECB, bring. While Kaminsky’s logic appears to be rather sound, it is important to note that his article is only published in the Cryptology ePrint Archive which is meant for recent research on cryptology, but only has a brief review process and is therefore somewhat unreliable.

# Multiple Scores

For each test in the test suite, NIST gives a pass/fail decision based on a P-value and a critical value that was set at 0.01 (Rukhin, 2011). The P-value is the probability that the observed value would occur by chance even if the null-hypothesis, that the sequence is random, is true. Given the large number of tests included in the NIST test suite, test sequences will likely fail some of them. However, this does not mean that the sequence is non-random. To truly test this, another statistical test is needed. This time, the statistical test is applied to the p-values themselves to see if they are within expected ranges for a truly random sequence (Kaminsky, 2019). This process was repeated for 300 different ciphertexts created by the encryption algorithm (Rukhin, 2011). This means that the test suite outputs many statistics based on short subsequences instead of a single summary statistic from one long sequence. This makes it difficult to accurately compare test results across multiple encryption algorithms because different tests may have different levels of importance or accuracy.

# Conclusion

Although NIST, DieHarder and other randomness tests are frequently used to test the randomness of block ciphers, they are actually rather unsuited due to being originally designed for the purpose of testing pseudorandom number generators that could generate arbitrarily long sequences. This is because it is unclear whether their results indicate the randomness of the block cipher itself or the mode of operation that used to combine the chunks. It is unclear why NIST and other tests designed for PRNGs continue to be used. It may be because many people are simply unaware of the issues that modes of operation bring or that test suites designed specifically for block ciphers are relatively new and unknown.

**References**

Ahmad, T., & Younis, U. (2014). Randomness testing of non-cryptographic hash functions for real-time hash table based storage and look-up of URLs. *Journal of Network and Computer Applications*. 41, 197-205. [doi.org/10.1016/j.jnca.2013.11.007](https://doi.org/10.1016/j.jnca.2013.11.007)

Dworkin, M. (2001). *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*. United States Department of Commerce, National Institute of Standards and Technology. <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf>

Gevorkyan, M.N., Demidova, A.V., Korol’kova, A.V. & Kulyabov, D.V. (2020). A Practical Approach to Testing Random Number Generators in Computer Algebra Systems. *Comput. Math. and Math. Phys*. 60, 65–73. [doi.org/10.1134/S096554252001008X](https://doi.org/10.1134/S096554252001008X)

Kaminsky, A. (2019). Testing the randomness of cryptographic function mappings. *Cryptology ePrint Archive*. [eprint.iacr.org/2019/078.pdf](https://eprint.iacr.org/2019/078.pdf)

National Institute of Standards and Technology. (2016). *AES Development - Cryptographic Standards and Guidelines*. United States Department of Commerce. [csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-crypto-projects/aes-development](https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-crypto-projects/aes-development)

Patil, P., Narayankar, P., Narayan, D.G., & Meena S.M. (2016). A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish. *Procedia Computer Science*, 78, 617-624. [doi.org/10.1016/j.procs.2016.02.108](https://doi.org/10.1016/j.procs.2016.02.108)

Rukhin, A. (2011). Statistical Testing of Randomness: New and Old Procedures appeared as Chapter 3 in *Randomness through Computation,* H. Zenil ed. World Scientific, 2011, 33-51. [doi.org/10.1142/9789814327756\_0003](https://doi.org/10.1142/9789814327756_0003)

Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., & Vo, S. (2010). *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. United States Department of Commerce, National Institute of Standards and Technology. [tsapps.nist.gov/publication/get\_pdf.cfm?pub\_id=906762](https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762)

Soto, J. (1999). *Randomness Testing of the Advanced Encryption Standard Candidate Algorithms* (NISTIR 6390). United States Department of Commerce, National Institute of Standards and Technology. [tsapps.nist.gov/publication/get\_pdf.cfm?pub\_id=151193](https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=151193)

Sulak, F., Doğanaksoy, A., Ege, B., & Koçak, O. (2010). Evaluation of Randomness Test Results for Short Sequences. *Lecture Notes in Computer Science*, 309–319. [doi.org/10.1007/978-3-642-15874-2\_27](http://www.doi.org/10.1007/978-3-642-15874-2_27)