Randomness Testing of the Advanced Encryption Standard Candidate Algorithms

Affiliation
Juan Soto, Jr.
National Institute of Standards and Technology
100 Bureau Drive, Stop 8930
Gaithersburg, MD 20899-8930
(301) 975-4641 (Voice)
(301) 948-1233 (Fax)
soto@nist.gov

Abstract
One of the criteria used to evaluate the Advanced Encryption Standard candidate algorithms was their demonstrated suitability as random number generators. That is, the evaluation of their output utilizing statistical tests should not provide any means by which to computationally distinguish them from a truly random source. This internal report lists several characteristics which an encryption algorithm exhibiting random behavior should possess, describes how the output for each candidate algorithm was evaluated for randomness, discusses what has been learned utilizing the NIST statistical tests, and finally provides an interpretation of the results.

1. Introduction

One of the criteria used to evaluate the Advanced Encryption Standard (AES) candidate algorithms was their demonstrated suitability as random number generators. That is, the evaluation of their outputs utilizing statistical tests should not provide any means by which to *computationally distinguish* them from truly random sources. The evaluation reported on here focused on 128-bit keys. Future work will address the 192-bit and 256-bit key sizes.

This paper *lists several characteristics that an encryption algorithm exhibiting random behavior should possess*, *describes how the output for each candidate algorithm was evaluated for randomness, discusses what has been learned utilizing the NIST statistical tests*, and *finally provides an interpretation of the results*.

At least 135 distinct data sets were obtained, analyzed and re-analyzed to ensure accuracy. In all, a total of **226,106,880** *128-bit* blocks (28,941,680,640 bits) were *stored*, *processed* and *analyzed*. File management and data management for this amount of data was always a major concern.

The NIST statistical tests were executed on several platforms in order to manage the volume of data. On many occasions, testing was discontinued due to seemingly non-random behavior. Such behavior could be attributed to any number of factors, which include *the data*, *the data extraction source codes*, *the cryptographic algorithm source codes*, and *the statistical tests*.

In Section 2.0, the categories of data are defined and described. In Section 3.0, the empirical results compiled to date are discussed. In Section 4.0, a summary of lessons learned and an interpretation of the test results is presented. Lastly, an appendix that briefly describes the individual statistical tests has been included.

2. AES Data

Fifteen encryption algorithms were selected as candidates for the AES. NIST harnessed and analyzed nine different sets of data for each of these algorithms (135 data sets in all). These data sets were selected because it was believed that they would be useful in evaluating the randomness of cryptographic algorithms. **Table 1** highlights these categories of data.

Table 1. Categories of Data

1. 128-Bit Key Avalanche
2. Plaintext Avalanche
3. Plaintext/Ciphertext Correlation
4. Cipher Block Chaining Mode
5. Random Plaintext/Random 128-Bit Keys
6. Low Density Plaintext
7. Low Density 128-Bit Keys
8. High Density Plaintext
9. High Density 128-Bit Keys

2.1 128-Bit Key Avalanche and Plaintext Avalanche

To examine the sensitivity of individual algorithms to changes in input parameters (i.e., the key or the plaintext), 384 binary sequences were analyzed in each case. In the **key avalanche** case, 384 sequences (1,048,576 bits per sequence) were parsed from the resulting string, constructed as follows: given 24,576 random 128-bit keys, and a plaintext of all zeroes, 3,145,728 *derived blocks* were concatenated. Each derived block is based on the XOR of the "ciphertext formed using the fixed plaintext and a random 128-bit key," and the "ciphertext formed using the fixed plaintext and the perturbed random 128-bit key with the i^{th} bit changed, for $1 \leq i \leq 128$." In the case of the **plaintext avalanche property**, substitute *"random plaintext(s)"* for *"random 128-bit key(s)"* in the above description, *"128-bit key of all-zeroes"* for *"plaintext of all zeroes"* and *"fixed 128-bit key"* for *"fixed plaintext."*

2.2 Plaintext/Ciphertext Correlation

In order to study the correlation of plaintext/ciphertext pairs, 128 sequences (1,048,576 bits per sequence) were examined for each algorithm. Given a random 128-bit key and 8,128 random plaintext blocks, a binary sequence was constructed concatenating 8,128 *derived blocks* (where a derived block is the result of applying the XOR operator on the plaintext block and its corresponding ciphertext block computed in the ECB mode).

Using the 8,128 (*previously selected)* plaintext blocks, the procedure was repeated 127 times, i.e., once for each (additional) random 128-bit key.

2.3 Cipher Block Chaining Mode

Given a random 128-bit key, a 128-bit initialization vector (IV) of all zeroes, and a 128-bit plaintext block (PT) of all zeroes, a binary sequence of 1,048,576 bits was constructed using ciphertext computed in the CBC mode. That is, a binary sequence consisted of 8,192 concatenated 128-bit ciphertext blocks. The first ciphertext block (CT_1) is defined by $CT_1 = E_k(IV \oplus PT)$. Subsequent ciphertext blocks were defined by $CT_{i+1} = E_k(CT_i \oplus PT)$ for $1 \leq i \leq 8{,}191$. In all, 300 binary sequences were constructed, each with a different random 128-bit key.

2.4 Random Plaintext/Random 128-Bit Keys

In order to examine the randomness of ciphertext (based on random plaintext and random 128-bit keys), 128 sequences were constructed. Each sequence was a result of the concatenation of 8,128 *ciphertext blocks* using 8,128 random plaintexts and a random 128-bit key in the ECB mode.

2.5 Low Density Plaintext and Low Density 128-Bit Keys

Two data sets were created based on low-density blocks used as either plaintext or 128-bit keys. Each data set consisted of 128 sequences. Each sequence consisted of 8,257 ciphertext blocks computed in the ECB mode. These ciphertext blocks were formed from one all zero plaintext block (128-bit key), 128 plaintext blocks (128-bit keys) of a single one and 127 zeroes (the one appearing in each of the possible 128 bit positions), and 8,128 plaintext blocks (128-bit keys) of two ones and 126 zeroes (the two ones appearing in each combination of two bit positions within the 128-bit positions).

2.6 High Density Plaintext and High Density 128-Bit Keys

Two data sets were created based on high-density blocks used as either plaintext or 128-bit keys. Each data set consisted of 128 sequences. Each sequence consisted of 8,257 ciphertext blocks computed in the ECB mode. These ciphertext blocks were formed from one all ones plaintext block (128-bit key), 128 plaintext blocks (128-bit keys) of a single zero and 127 ones (the zero appearing in each of the possible 128 bit positions), and 8,128 plaintext blocks (128-bit keys) of two zeroes and 126 ones (the two zeroes appearing in each combination of two bit positions within the 128-bit positions).

3. Empirical Analysis

Table 2 describes the breakdown of the simulation-input parameters per data category. The number of sequences utilized in the samples varied from 128 to 384. Based on earlier experiments conducted on pseudo-random number generators with sequences consisting of 1,000,000 bits, sequences consisting of 2^{20} (i.e., 1,048,576) bits were

chosen. In four instances, sequences consisting of 1,056,896 bits were chosen, which corresponded to the number of bits in 8,257 ciphertexts. The number of statistical tests applied varied from 59 to 187. The difference was due to the specification of alternative input parameters such as the number of templates to be used.

For each experiment, the *significance level* was fixed at 0.01, which implies that, ideally, no more than one binary sequence should be rejected for each sample of 100 binary sequences evaluated by a statistical test. However, in all likelihood, any given data set will deviate from this ideal case. A more realistic interpretation is to use a *confidence interval* (CI) for the "*proportion of binary sequences*" that should pass at the 0.01 level. Thus, a data set that fails more than 5% of the sequences is flagged as suspect.

Table 2. Breakdown of the Numerical Simulation Input Parameters

Data Category	Sample Size/ Sequence Length	Number of Tests Applied
128-Bit Key Avalanche	384/1,048,576	187
Plaintext Avalanche	384/1,048,576	187
Plaintext/Ciphertext Correlation	128/1,048,576	59
Cipher Block Chaining Mode	300/1,048,576	59
Random Plaintext/Random 128-Bit Key	128/1,048,576	59
Low Density Plaintext Input	128/1,056,896	59
Low Density 128-Bit Key Input	128/1,056,896	59
High Density Plaintext Input	128/1,056,896	59
High Density 128-Bit Key Input	128/1,056,896	59

Table 3 highlights the maximum number of binary sequences that are expected to be rejected at the chosen significance level. For example, if the sample consists of 128 sequences, the rejection rate should not exceed 4.657, or simply expressed, 4 sequences. The maximum number of rejections was computed using the formula:

$$s\left(a + 3\sqrt{\frac{a(1-a)}{s}}\right),$$

where s is the *sample size* and α is the *significance level*.

Table 3. Decision Rules for the Determination of Non-Randomness

Number of Sequences	Significance Level $\alpha = 0.01$	Significance Level $\alpha = 0.01$ with CI
128	1.28	4.657
300	3.00	8.170
384	3.84	9.689

Based on the current version of the NIST statistical tests, each of the algorithms was evaluated. Those algorithms that did not demonstrate deviation from randomness include **CAST-256**, **DFC**, **E2**, **LOKI-97**, **MAGENTA**, **MARS**, **RIJNDAEL**, **SAFER+**, and

SERPENT. The remaining algorithms *appeared to have displayed*[1] deviation from randomness. A case by case description follows.

CRYPTON

(a) **128-Bit Key Avalanche**: Preliminary analysis suggests too many rejections of the 001001111 non-periodic template. A total of **11** rejections was reported when the expected number should be no more than **9**.

(b) **Random Plaintext/Random 128-Bit Keys**: Preliminary analysis suggests a problem given an excessive number of visits to the states x = -3 and x = -4 in the random excursion test. In both instances, there were **4** rejections out of **76** sequences processed when the expected number should be no more than **3**.

DEAL

(a) **Random Plaintext/Random 128-Bit Keys**: Preliminary analysis suggests a problem given an excessive number of visits to the state x = +2 in the random excursions test. In this instance, there were **4** rejections out of the **76** sequences processed when the expected number should be no more than **3**.

FROG

(a) **Low Density 128-Bit Keys**: Preliminary analysis suggests a problem given an excessive number of visits to the state x = +3 in the random excursions test. In this instance, there were **4** rejections out of the **71** sequences processed when the expected number should be no more than **3**.

HPC

(a) **128-Bit Key Avalanche**: Preliminary analysis suggests that this algorithm dramatically fails many statistical tests. The specific tests that rejected the sample of 384 sequences follows, with the number of sequences rejected specified in parenthesis (in each case, the expected number of rejections should not exceed **9**): frequency (**75**), block frequency (**91**), cumulative sum forward (**71**), cumulative sum reverse (**75**), runs (**77**), rank (**29**), universal statistical (**92**), approximate entropy (**92**), and Lempel-Ziv complexity (**91**). It has been determined that 1 in 256 (128-bit) keys is an equivalent key[2]. NIST's experiments support this theoretical finding. Clearly,

[1] For example, the *random excursion* test and the *random excursion variant* test *only apply* whenever the number of cycles exceeds 500. If a sample has sequences with cycles fewer than 500, then they will not be evaluated by the random excursion tests, and thus the proportion of applicable sequences will be reduced (by as much as 42%). In this event, a small sample size may *incorrectly* suggest deviation from randomness. Whenever applicable, the actual sample size processed (for these two tests) is indicated in parentheses in the next section. It is important to keep in mind that only one sample was constructed for each algorithm and data category. In the unlikely event that a poor sample is observed, additional experimentation is warranted.

a rejection of 23.96% in one test (the largest rejection by any statistical test) is far too large. Hence, this algorithm failed, given the statistically significant results at the 0.01 level.

RC6

(a) **128-Bit Key Avalanche**: Preliminary analysis suggests that there were too many rejections for the non-periodic templates test. There were three templates (01011011, 010111011, 110001010) which rejected **10**, **10**, and **11** binary sequences respectively. The expected number of rejections should not exceed **9**.

(b) **High Density Plaintext**: Preliminary analysis suggests a problem given **6** rejections for both the frequency test, and the cumulative sum forward test. The expected number of rejections should not exceed **4**.

TWOFISH

(a) **Random Plaintext/Random 128-Bit Keys**: Preliminary analysis suggests a problem given an excessive number of visits to the state x = +9 in the random excursions variant test. In this instance there were **4** rejections out of the **83** sequences processed when the expected number should not exceed **3**.

(b) **Low Density Plaintext**: Preliminary analysis seems to indicate sensitivity to the cumulative sums forward test. The total number of rejections was **7,** which represents over 5% of the sample. The expected number of rejections should not exceed **4**.

4. Conclusion

During round one testing, numerous statistical tests were applied on 135 sets of data, which collectively span many well-known properties that any good cryptographic algorithm should satisfy. These properties include *any detectable correlation between plaintext/ciphertext pairs, any detectable bias due to single bit changes to either a plaintext or a 128-bit key*, in addition to many others.

It appears that the **HPC** algorithm is non-random given the statistically significant results obtained at the 0.01 level for the *128-bit key avalanche* data set. Unfortunately, the results for five algorithms, **Crypton**[3], **DEAL**[4], **Frog**[5], **RC6**[6] and **Twofish**[7] are

[2] Carl D'Halluin, Gert Bijnens, Bart Preneel, Vincent Rijmen, *"Equivalent Keys of HPC,"* Katholieke Universiteit Leuven, ESAT-COSIC, 4/8/99.

[3] The 128-bit key avalanche and random plaintext/random 128-bit key data sets.
[4] The random plaintext/random 128-bit keys data set.
[5] The low density 128-bit keys data set.
[6] The 128-bit key avalanche and high-density plaintext data set.
[7] The random plaintext/random 128-bit keys and low-density plaintext data set.

inconclusive. Although preliminary testing suggests minor deviations from randomness[8] for these five algorithms, one sample alone cannot provide sufficient evidence. In each case more samples should be evaluated to determine whether statistically significant results are consistently present.

It is worth noting that the specific statistical tests that predominantly flagged several algorithms as suspect were the *cumulative sums* (cusum) *test* and the *random excursion test*. To further evaluate the *cusum test*, extensive simulations were conducted on the **SHA-1** generator and the **Twofish** algorithm. The results suggest that data flagged as non-random for the **Twofish** and **RC6** algorithms should be treated as statistical anomalies. Similarly, due to the natural filtering process for the random excursion test, small sample sizes may *incorrectly* lead one to commit a type I error, i.e., labeling data as nonrandom when it truly appears to be random.

For all of the remaining algorithms: **CAST-256**, **DFC**, **E2**, **LOKI97**, **MAGENTA**, **MARS**, **RIJNDAEL**, **SAFER+**, and **SERPENT**, the results to date suggest that there is no deviation from randomness.

Acknowledgments

In particular, I would like to acknowledge the efforts of the *RNG Technical Working Group*, especially *Dr. Andrew Rukhin* who derived the statistical tests and *Dr. Jim Nechvatal* who proposed the categories of data to be analyzed. Additionally, I'd like to express my gratitude to *Larry Bassham, Sharon Keller, Neil Molino* and *Murugiah Souppaya* who have always provided their technical assistance.

References

[1] William Caelli, et. al., *Crypt X Package Documentation*, Information Security Research Centre and School of Mathematics, Queensland University of Technology, 1992.

[2] Carl D'Halluin, et. al., *"Equivalent Keys of HPC,"* Katholieke Universiteit Leuven, ESAT-COSIC, April 1999.

[3] Helen Gustafson, et. al., *"A computer package for measuring the strength of encryption algorithms,"* Computers & Security, Volume 13, pp. 687-697, 1994.

[4] Alfred Menezes, et. al., Handbook of Applied Cryptography, CRC Press, 1997.

[5] Andrew Rukhin, et. al., *"A Statistical Test Suite for the Validation of Cryptographic Random Number Generators,"* NIST Computer Security Division/Statistical Engineering Division Internal Document, September 1999.

[8] In the interest of foregoing analyzing hundreds or thousands of samples (for a particular algorithm and specific data type), the empirical rule, described in section 3.0, was introduced.

Appendix: Description of the Statistical Tests

Frequency Test: The purpose of this test is to determine whether the number of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence.

Block Frequency Test: The purpose of this test is to determine whether the frequency of *m*-bit blocks in a sequence appears as often as would be expected for a truly random sequence.

Cumulative Sums Forward (Reverse) Test: The purpose of this test is to determine whether the maximum of the cumulative sums in a sequence is too large or too small; indicative of too many ones or zeroes in the early (late) stages.

Runs Test: The purpose of this test is to determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence. In particular, this test determines whether the oscillation between such substrings is too fast or too slow.

Long Runs of Ones Test: The purpose of this test is to determine whether the distribution of long runs of ones agrees with the theoretical probabilities.

Rank Test: The purpose of this test is to determine whether the distribution of the rank of 32x32 bit matrices agrees with the theoretical probabilities.

Spectral (Discrete Fourier Transform) Test: The purpose of this test is to determine whether the spectral frequency of the binary sequence agrees with what would be expected for a truly random sequence.

Non-periodic Templates Test: The purpose of this test is to determine whether the number of occurrences for a specified nonperiodic template agrees with the number expected for a truly random sequence.

Overlapping Template Test: The purpose of this test is to determine whether the number of occurrences for a template of all ones agrees with what is expected for a truly random sequence.

Universal Statistical Test: The purpose of this test is to determine whether a binary sequence does not compress beyond what is expected of a truly random sequence.

Approximate Entropy Test: The purpose of this test is to compare the frequency of overlapping blocks of two consecutive/adjacent lengths (*m* and *m+1*) against the expected result for a normally distributed sequence. In short, it determines whether a sequence appears more regular than is expected from a truly random sequence.

Random Excursion Test: The purpose of this test is to examine the number of cycles within a sequence and determine whether the number of visits to a given state, [-4, -1] and [1, 4], exceeds the expected for a truly random sequence.

Random Excursion Variant Test: The purpose of this test is to determine if the total number of visits to states, between [-9, -1] and [1, 9] exceeds the expected for a truly random sequence.

Lempel-Ziv Complexity Test: The purpose of this test is to determine whether or not the sequence compresses no more than a truly random sequence.

Linear Complexity Test: The purpose of this test is to determine whether or not the sequence is complex enough to be considered truly random.