Journal of Applied Mathematics and Bioinformatics, Vol. x, No. xx, 2021, xxx-xxx
ISSN: 1792-6602(print), 1792- 6939(online)
Scientific Press International Limited

Rating the Security Strength of Cryptographic Algorithms

George Marinakis [1]

Abstract

In this study, we propose a method in order to estimate the strength of a cryptographic algorithm. The method combines the evaluation of the cryptographic key length and the evaluation of the success rate of the randomness tests in the algorithm output samples. In the first step the algorithm is classified into one of four general categories, according to its key size, taking into account the current computer power which a cryptanalyst can use for exhaustive key search. In the second step we examine the success rate of the tests on the output samples. For this, the maximum accepted number of the rejected samples is calculated, taking as parameters the total number of samples (which depends from the selected sampling error) and the desired significance level and confidence interval for the success rate of the tests. If the rejected samples do not exceed the maximum number, the algorithm is considered as "random" and it is rated in the initial strength category due to its key size. If the rejected samples exceed the maximum number, the algorithm is submitted to further tests under certain conditions.

Keywords: Cryptography, Data encryption, Communication security, Computer security, Data security, Information security.

1. Introduction

Due to the huge size of cryptographic keys (usually from 128 to 256 bits), it is practically impossible to test the algorithm for all key combinations (from 2^{128} to 2^{256} respectively). Therefore, the sampling method is used, in which from the total number of the N key combinations, a much smaller number of n keys is selected. Using a software simulation of the algorithm, for each of the n sampling keys, a sample output of the algorithm is generated and these n samples are submitted to statistical randomness tests. Then, the tests results are processed for the calculation of the total performance of the algorithm for all the keys, using a predetermined sampling error. The final decision of the algorithm cryptographic strength is made based on the overall success rate of the tests, which are extremely time consuming. Therefore, if we want a reliable sampling (small sampling error) but also a practically feasible time to perform the tests, the main problems are:

a. How many output samples should we check?
b. What should be the size of each sample?
c. How can we reduce the extremely long time required for the tests?
d. What criteria should we use in order to select the sampling keys?
e. How do we rate the strength of the algorithm based on the test results of its samples?

Solutions to the problems (a), (b), (c) were addressed in (Marinakis, May 2021) [1] and solutions to the problem (d) were addressed in (Marinakis, July 2021) [2]. The problem (e) which remains, will be addressed in the present study.

[1] Hellenic Army Academy - gmari@tee.gr

Article Info: *Received:* …. .., 2022. *Revised*: …. .., 2022.
Published online: …. .., 2022.

The methods which will be proposed are focused on symmetric cryptographic algorithms (block ciphers and stream ciphers), but similar methods can be applied for asymmetric cryptographic algorithms.

2. Length of the Key

The security of a cryptographic algorithm is based on its internal complexity and its key length. But when the algorithm does not have a known and exploitable defect in its internal structure, then the only cryptanalytic attack that can be applied to it is the method of the exhaustive search of its key (known as Exhaustive Key Search or Brute Force Attack). This attack process is extremely time consuming and if the length of the key is big enough, then the exhaustive search is practically impossible and therefore we can say that the algorithm is practically secure.

Table 1 gives the relative strengths of symmetric and asymmetric algorithms based on their key length and known cryptanalytic attacks, according to (NIST.SP.800-57 pt1r4, 2016) [3]. It is obvious that Table 1 is valid when the algorithm does not have a vulnerability (the exploitation of which can reduce the number of keys or totally bypass them), so the only possible attack to it is the exhaustive key search [1].

Table 1. Correspondence of symmetric and asymmetric algorithms security strength, based on their key length (from NIST SP800-57 pt1r4).

Security strength	**Symmetric key algorithms**	**Asymmetric key algorithms**		
		FFC (e.g., DSA, D-H)	**IFC (e.g., RSA)**	**ECC (e.g., ECDSA)**
$\leq$ 80	**2TDEA (2-DES)**	***L* = 1024** ***N* = 160**	**1024**	**160-223**
112	**3TDEA (3-DES)**	***L* = 2048** ***N* = 224**	**2048**	**224-255**
128	**AES-128**	***L* = 3072** ***N* = 256**	**3072**	**256-383**
192	**AES-192**	***L* = 7680** ***N* = 384**	**7680**	**384-511**
256	**AES-256**	***L* = 15360** ***N* = 512**	**15360**	**512+**

[1] The first column of Table 1 expresses the active (actual) length of the key, which may be shorter than the nominal. E.g. while 3DES has a theoretical key length of 3x56 = 168 bits, however there is a cryptanalytic attack which reduces its active length to 112 bits. Similarly, for 2DES it has been found that if the cryptanalyst has at its disposal about 2^40 pairs of plain / crypto texts, the active key is reduced from the nominal value of 112 bits to 80 bits, while if the cryptanalyst knows 2^56 pairs of plain / crypto texts, the active key is reduced to 56 bits. (In the third column, *L* is the public key and *N* is the private key).

As is it shown, Table 1 contains only specific key lengths, from cryptographic algorithms which are designed in the US (excluding AES) and approved by NIST. In order to classify the security strength of the algorithms including all possible intermediate lengths of the keys between 80 bits and 256 bits, we constructed Table 2, in which we classified the strength of the algorithms into four strength categories: Low, Medium, High and Very High, based on the value range to which their key length falls.

Table 2. Comparative strength of symmetric cryptographic algorithms based on the length of their key, according to current cryptanalytic and technological data (year 2022).

KEY LENGTH (K)	**$80 \leq K \leq 112$**	**$112 < K < 128$**	**$128 \leq K \leq 192$**	**$192 < K \leq 256$**
ALGORITHM STRENGTH	**LOW**	**MEDIUM**	**HIGH**	**VERY HIGH**

We note that the above strength classification of cryptographic algorithms mainly shows the comparison between them, i.e. it is relative and not absolute. For example, for current computer technology, the 128-bit key length it is considered to give a high strength against the Exhaustive Key Search. However, the length of 128 bits compared to 256 bits should be considered at least one degree lower. In this study, we consider the key length of 256 bits as the upper limit for today symmetric ciphers, but it is obvious that an algorithm with a longer key length, from a cryptanalytic point of view, can "withstand" the Exhaustive Key Search attack over a longer period of time.

Table 2 shows the strength of cryptographic algorithms, based on current cryptanalytic and technological data (year 2022). But as it is mentioned in (Marinakis, 2013) [4], in order to compensate for the constant evolution of integrated circuits (due to the Moore's law) and the relative increase of computer power, the key must increase by one bit each year. In this way the cryptographic algorithm will be safe from the evolution of the exhaustive key search.

As a comparative example, we designed Table 3, which shows the key lengths that must be applied after 20 years, in order the cryptographic algorithms be secure against the exhaustive key search, according to the expected technological development (year 2042). We see that compared to Table 2, the values of the keys have increased by 20 (one bit increment for each year).

Table 3. Comparative strength of symmetric cryptographic algorithms based on the length of their key, after 20 years, due to the expected technological evolution (year 2042).

KEY LENGTH (K)	**$100 \leq K \leq 132$**	**$132 < K < 148$**	**$148 \leq K \leq 212$**	**$212 < K \leq 276$**
ALGORITHM STRENGTH	**LOW**	**MEDIUM**	**HIGH**	**VERY HIGH**

In any case, the final choice of the appropriate cryptographic algorithm (or the cryptographic system in general), must be made based on the desired duration of the protection of the encrypted information, combined with the analysis of the risks that the cryptographic system faces from potential threats which will take advantage of its weaknesses. These issues will be considered in a future study.

3. Calculation of the Success Rate

As mentioned in paragraph 1, in order to investigate the required randomness, independence and unpredictability of the generated digital sequences of an algorithm, special statistical and cryptanalytic tests are performed on its output bits. Three suites of these statistical tests are shown in Table 4, as they are referred to (Marinakis, 2015) [5]. These tests, essentially examine the randomness in the output bitstreams of Random Number Generators (RNG), Pseudo Random Number Generators (PRNG) and symmetric cryptographic algorithms (Stream and Block Ciphers).

Table 4. Three suites of statistical tests for randomness (available in software)

NIST SP 800-22 (for RNG and PRNG)	**DIEHARD (Marsaglia)** (for RNG)	**CRYPT-X** (Stream/Block Ciphers)
1) Frequency 2) Cumulative Sum 3) Runs 4) Rank 5) Spectral 6) Templates Matching 7) Universal Statistical 8) Approximate Entropy 9) Random Excursions 10) Moving Averages 11) Lempel-Ziv Compression 12) Linear Complexity 13) Bayes	1) Birthday Spacings 2) Overlapping 5-permutation 3) Binary Rank (6x8 Matrices) 4) Binary Rank (31x31 & 32x32 Matrices) 5) Monkey tests (20-bit words) 6) Monkey tests (OPSO, OQSO,DNA) 7) Number of 1's in stream of bytes 8) Number of 1's in specific bytes 9) Parking Lot 10) Overlapping Sums 11) Squeeze 12) Minimum Distance 13) Random Sphere's 14) Runs 15) Craps	STREAM CIPHERS 1) Frequency 2) Binary derivatives 3) Change points 4) Runs 5) Sequence complexity 6) Linear complexity BLOCK CIPHERS (1) Frequency (2) Binary Derivative (3) Linear (4) Affine (5) Avalanche (Plaintext) (6) Complementation

For each statistical test suite from those that will be selected from Table 4, we must examine as many output samples of the algorithm as possible and then calculate the overall success rate, i.e. how many of the samples successfully passed each individual statistical test of the suite.

As it is mentioned in (NIST.SP.800-22, 2010) [6] , for each statistical test we must define a success criterion, which is called *significance level* and is denoted by α. The α expresses the probability that a Type 1 error occurs, i.e. the test shows that the sequence is not random, when in fact it is random. For example, if we set $\alpha = 0.03$, it means that an algorithm successfully passes the test, if out of the 100 output samples that we have examined, at most three are not random.

The standard values of the significance level α for cryptography are around 0.01, which means that at most one in 100 algorithm output sequences we accept that it is not random. However, as it is mentioned in (Soto, 1999) [7], in practice any set of digital sequences of an algorithm that we will choose, it will most likely deviate from this ideal case.

A more realistic approach is to use a *confidence interval* (CI) for the percentage of the sequences that are expected to pass the desired α = 0.01. In this case the most appropriate confidence interval (CI) is 95%. This means that if more than 5% of the samples fail a test, then the algorithm is considered "suspicious" for generating non-random outputs. The maximum number of the rejected samples m that can be accepted in each test with a 95% confidence interval is given in (Soto, 1999) [7] by the formula (1):

$$m = n\left(\alpha + 3\sqrt{\frac{\alpha(1-\alpha)}{n}}\right) \quad (1)$$

where n is the total number of samples tested and α is the significance level (the formula is derived from the normal distribution curve, which approximates the binomial distribution curve for large n).

Table 5 shows the maximum accepted number of the rejected samples, based on the desired sampling error e (column 1) and the corresponding number of the samples n that must be tested (column 2). The first and second column of Table 5 were taken from (Marinakis, 2021) [1]. The maximum accepted number of the rejected samples, are calculated first taking into account only the significance level (column 3) and then taking into account both the significance level and the confidence interval CI (column 4) based on formula (1).

Table 5. Maximum accepted number of samples to be rejected, based on the desired sampling error e and the corresponding number of samples n that must be tested.

SAMPLING ERROR (e)	**NUMBER OF SAMPLES** (n)	**MAXIMUM ACCEPTED NUMBER OF REJECTED SAMPLES** (m) (the integer part of the number is taken into account)	
		For significance level α = 0.01	For significance level α = 0.01 and CI = 95%
5 %	384	3,84	9,689
4 %	600	6,00	13,311
3 %	1067	10,67	20,420
2 %	2401	24,01	38,636
1 %	9604	96,04	125,292

Example: Suppose that during a randomness test we want to have a very small sampling error of 1%. Therefore, from the second column of Table 5 we see that we will need to examine 9604 samples. If we want the test to be strict, we must choose a success rate of 99%. This will give a failure rate of 1% (significance level $\alpha = 0.01$), which means that up to 96 samples can be accepted as rejected (third column of Table 5). But since, as mentioned, an absolute success rate of 99% of the samples is almost impossible in practice, a more realistic approach is to use a confidence interval in which the desired success rate will most likely be found. So, if we choose a 95% confidence interval, from the fourth column of Table 5 we find that in this test we can accept up to 125 rejected samples.

4. Algorithm Rating Process

The security strength of a cryptographic algorithm (hereafter will be referred as cryptographic strength), concerns the randomness, independence and unpredictability of its output bitstreams and is essentially the measure of the difficulty that an cryptanalyst pays to break it. The process of strength rating of a cryptographic algorithm that we propose in the present study is as follows:

In the first step, the algorithm is classified into one of four general cryptographic strength categories, based on its key size (Low, Medium, High, Very High). In the second step, the final degree of cryptographic strength is determined gradually, based on the results of the statistical randomness tests on its output bitstream samples, using one of the test suites from these that are shown in Table 4. It is at the discretion of the evaluator to use more than one test suite, but this will take much more time.

We must emphasize that the final strength degree cannot be larger than the initial strength degree of the cryptographic algorithm due to the size of its key. On the contrary, it is very likely that the strength degree will be lower than the initial degree, due to significant weaknesses that the algorithm may present either in its internal structure (e.g. known cryptanalytic attacks against the whole structure or against a reduced structure of the algorithm), or in the possible non randomness of its digital output sequences.

The rating process of the algorithm strength is summarized in Figure 1: Initially, the cryptographic algorithm is classified into one of four general strength categories, based on its key size K, as they were presented in paragraph 2 (Table 2):

1st category :	$80 \le K \le 112$	3rd category :	$128 \le K \le 192$
2nd category :	$112 < K < 128$	4th category :	$192 < K \le 256$

After classifying the tested algorithm in one of the above categories, then we examine the success rate of the randomness tests on its n output samples. If the randomness test is positive, i.e. if the rejected samples do not exceed the maximum accepted number m which is calculated from formula (1) of the previous paragraph, then we classify the algorithm in the strength category resulting from its key size (Low, Medium, High, Very High), as shown in Figure 1. If the randomness test of the algorithm is negative, i.e. if the rejected samples exceed the maximum accepted number m, then we generate a new set of n output samples and perform for a second time the statistical randomness tests, with the procedures which are described in the next paragraph 4.1.

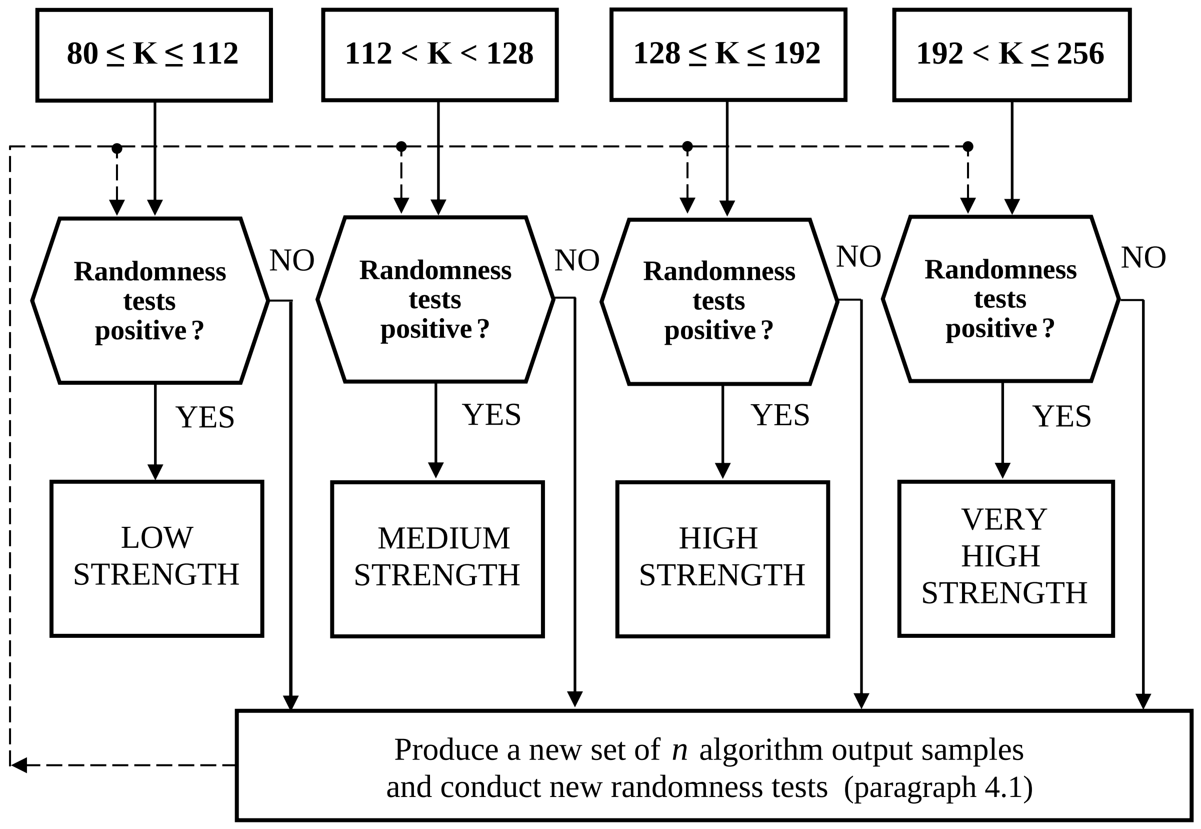
80 ≤ K ≤ 112
112 < K < 128
128 ≤ K ≤ 192
192 < K ≤ 256
Randomness tests positive ?
NO
YES
LOW STRENGTH
MEDIUM STRENGTH
HIGH STRENGTH
VERY HIGH STRENGTH
Produce a new set of n algorithm output samples and conduct new randomness tests (paragraph 4.1)

Figure 1. Process for the grading of cryptographic algorithm strength

4.1. Testing of new samples

If during the procedure shown in Figure 1, the total result of the algorithm randomness tests is negative, this does not necessarily mean that the algorithm has low cryptographic quality. According to (NIST.SP.800-22, 2010) [6] , the failure of a test may not be due to the low quality of the algorithm, but due to another cause, which may be one of the following:

a. Incorrect implementation of the tested cryptographic algorithm or its Random Number Generator (in hardware or software).

b. Incorrect software implementation of a statistical test or incorrect selection of its input parameters.

c. Inadequately designed statistical test (e.g. insufficient analysis and implementation based on probability theory or complexity theory).

d. Incorrect software for processing the input data of the statistical test.

e. Inaccurate mathematical calculation of constants (a, p), mainly in terms of the most perfect numerical approximation of their values.

f. Wrong selection in the characteristics of the samples (e.g. inappropriate number or size of the samples, inappropriate size of the blocks and patterns of the tests, etc.).

Because of the above, if the randomness tests of the first group of *n* algorithm samples are negative, it makes sense to give the algorithm a “second chance” in order to rule out the possibility that the failure is not due to a defective design of the algorithm but due to one of the above causes. To address this problem, we propose to generate a new set of *n* algorithm output samples and to perform again a randomness test under certain conditions. Thus, in the case of the “re-testing” we will have the following two possibilities:

a. The re-testing is negative: The algorithm is considered as “non-random” (since it failed twice in a row and for total *2n* samples).

b. The re-testing is positive: In this case, if m_1 and m_2 are the numbers of the rejected samples of the first and second test respectively, then their sum must not exceed the total *m* which is calculated from formula (1) of paragraph 3, where in the place of *n* we have to put *2n* (because we performed two tests with *n* samples each). Therefore, if the sum of the rejected samples is less than *m*, the algorithm is considered as “random” and it is rated in the strength category resulting from its key size (Low, Medium, High, Very High). If the sum of the rejected samples is greater than *m*, the algorithm is considered as “non-random” and it is not rated to a specific strength category.

Example: Suppose during the initial test of an algorithm we examine 600 samples and from them 15 are rejected. According to Table 5 (column 4) the maximum accepted number of rejected samples is 13, so the first test is negative. So, we perform a second test, during which, out of the new 600 samples, 6 are rejected (i.e. the second test is positive). Thus, out of a total of 1200 tested samples, a total of 21 samples were rejected. Applying formula (1) for $n = 1200$, we find that the maximum number of the rejected samples is $m = 22$. Therefore, the algorithm passes the tests successfully, since a total of 21 samples were rejected (less than 22 which is the limit).

Alternatively, if an algorithm failed twice at the tests, instead of considered as “non-random” and not rated, can be rated to a lower strength category than this which is based on its key length. This decision is up to the evaluator, who may take into account some additional parameters (such as details which concern the design and implementation of the algorithm etc.).

In order to save time, the re-testing can be performed only for the specific statistical test (from the suites of Table 4) in which the samples failed. Also, we must note that a second, maybe a third re-testing could be carried out according to the above procedure. This, of course, is at the discretion of the evaluator. For example, an evaluator may want to repeat the tests in order to see if and when the results of the re-tests will "correct" the results of the first test. However, the number of the re-tests for an algorithm cannot be excessive, for practical and ethical reasons. The practical reasons concern the extremely time-consuming process for the production of the samples and the execution of the tests. The ethical reasons concern the avoidance of a more “favorable” treatment of the tested algorithm, compared to some other algorithms which may have succeeded with the first testing. For the above reasons, during a comparative evaluation between different algorithms, we propose that an optimal rule is that the re-tests should not exceed the number of two.

5. Conclusion

In this study we proposed a method in order to rate the strength of a cryptographic algorithm. The first step is to classify the algorithm into one of four general strength categories, based on its key size (Low, Medium, High, Very High). The second step is to examine the success rate of the randomness tests on its output samples, calculating the maximum accepted number of the rejected samples, based on the number of samples (according to the selected sampling error) and based on the desired significance level and confidence interval for the success rate of the tests. If the randomness test is positive (the rejected samples do not exceed the maximum number), the algorithm is considered as “random” and we rate it in the strength category resulting from its key size. If the randomness test is negative (the rejected samples exceed the maximum number), we generate a new set of *n* output samples and perform the tests for a second time. With this "re-testing", we rule out the possibility that the failure is not due to the algorithm but due to other causes. If the second set of *n* output samples fails again (the rejected samples exceed again the maximum number), then the algorithm is considered as "non- random" and it is not rated or it can be rated to a lower strength category than this which is based on its key length. If the second set of *n* output samples passes the tests, we examine if the sum of the rejected samples of the two tests exceeds the maximum number (which is calculated for *2n*). If the sum exceeds the maximum number, the algorithm is considered as "non- random" and it is not rated or rated to a lower strength category. If the sum does not exceed the maximum number, the algorithm is considered as "random", therefore it is rated based on its key length.

When an algorithm fails at the first re-test, it is at the discretion of the evaluator to perform a second re-test. However, the number of the re-tests cannot be excessive, firstly because the tests procedures are extremely time-consuming and secondly in order to avoid a "favorable" treatment of the tested algorithm, compared to other algorithms which may have succeeded with the first test. Therefore, during a comparative evaluation between different algorithms, an optimal rule is to perform not more than two re-tests.

References

[1]. George Marinakis, "Sampling methods for cryptographic tests", May 2021. https://www.scienpress.com/journal_focus.asp?main_id=57&Sub_id=IV&Issue=2143151

[2]. George Marinakis, "Selection of sampling keys for cryptographic tests", July 2021. https://www.scienpress.com/journal_focus.asp?main_id=57&Sub_id=IV&Issue=2202191

[3]. NIST.SP.800-57pt1r4 "Recommendation for Key Management-1", 2016. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

[4]. George Marinakis "Minimum key length for cryptographic security", March 2013. http://www.scienpress.com/journal_focus.asp?main_id=57&Sub_id=IV&Issue=597

[5]. George Marinakis, "Design and evaluation of random number generators", September 2015. http://www.scienpress.com/journal_focus.asp?main_id=57&Sub_id=IV&Issue=1608

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

[7]. Juan Soto, Jr.," Randomness Testing of the Advanced Encryption Standard Candidate Algorithms", NIST, IR 6390, September 1999.