CAES Cryptosystem: Advanced Security Tests and Results

Said Bouchkaren, Saiida Lazaar
(Corresponding author: Said Bouchkaren)

Department of Mathematics and Computer Science, ENSA of TANGIER, AbdelMalek Essaadi University
P.O. Box 1818 Principal Tangier, Tangier, Morocco
(Email: saidbouchkaren1@hotmail.com)

(Received Sept. 21, 2016; revised and accepted Jan. 15 & Feb. 19, 2017)

Abstract

A robust and secure cryptosystem is an encrypting system that resists against all practical cryptanalysis methods such as statistical attacks, differential cryptanalysis and linear cryptanalysis. To prove the resistance against these attacks, the cryptosystem designer must carry out a list of robustness tests. Considering these constraints, we present in the current paper results of robustness and security tests conducted on the *CAES* (Cellular automata Encryption System) cryptosystem published in a previous article. The presented tests focus on randomness tests and on differential cryptanalysis. As results of these tests, we concluded that the cryptosystem *CAES* gives a pseudo-random output regardless the input. Also the differential attack needs huge number of chosen plaintexts which make it impractical.

Keywords: Block Cipher; Cellular Automata; Differential Cryptanalysis; Randomness Test

1 Introduction

In the new era, the use of networks to communicate becomes a necessity, which means that there is huge amounts of data transmitted between communicating entities. These data are classified as normal, secret or top secret. To transmit secret or top secret data, a secure and trusted communication channel must be created. This secret communication channel can be established using a robust and reliable cryptosystem [12].

A robust and reliable cryptosystem is an encryption algorithm that can be used to encrypt and decrypt data, if and only if the communicating entities have the encryption keys [6, 8]. In other words the cryptosystem must resists against all feasible cryptanalysis methods such as *statistical attacks*; which exploits the statistical properties of the input to guess the output; and the *differential attack*; which is a kind of statistical attack, but in lieu of exploiting statistical properties of the input, it exploits the statistical differences in inputs to guess the differences in outputs.

As method of validating the reliability of an encryption system, designers conduct series of theoretical and experimental tests.

In the current article, we present some advanced validation tests to prove the robustness of a previously published algorithm named *CAES* [3]; *CAES* is a symmetric encryption scheme based on cellular automata theories defined in [3]. It encrypts blocs of 256 bits using 256 bits keys; In this article *Randomness tests* and *differential cryptanalysis* are applied. The results obtained in this paper show that *CAES* generates pseudo-random output regardless the input which means it resists against statistical attacks and also we proved that the differential attack is practically impossible.

We remember that the previous paper [3] proved that *CAES* have a good confusion and diffusion properties and it has a hight performance rate. Also the brute force attack against *CAES* has no effects.

The rest of this article is structured as follow: the second section gives brief description of *CAES* cryptosystem, the third section describes the differential cryptanalysis, the fourth section gives an overview of statistical tests, the fifth section describes data generation for experimental tests, the sixth section gives the obtained results and discussion, the seventh section describes the results of the differential attack and the last section is a conclusion and perspectives of the work.

2 *CAES* Cryptosystem

CAES (*Cellular Automata Encryption System*) is a symmetric encryption scheme based on cellular automata defined and published previously in [3]. This algorithm uses cellular automata for encryption, decryption and sub keys generation process. As technical specification, *CAES* processes data in blocs of 256 bits and uses a key of 256 bits and the encryption or decryption is accomplished after 12 iterations. For each iteration, a sub key is generated from the encryption key using a reversible and irreversible cellular automata. The encryption and decryption processes are given respectively in Algorithm 1 and Algorithm 2.

Algorithm 1 *CAES* Encryption algorithm

```
procedure ENCRYPT($M, Key$)    ▷ $M$ is the plaintext message block and $Key$ is the encryption key
    $SKeys[12] \leftarrow SubKeys(K)$;    ▷ Generating 12 sub keys
    for $i$ from 0 to 11 do
        $M = Shift(M)$
        $M = IMix(M)$
        $M = PMix(M)$
        $M = AddKey(M, SKeys[i])$
    end for
    return $M$    ▷ $M$ contains the encrypted message
end procedure
```

Algorithm 2 *CAES* Decryption algorithm

```
procedure DECRYPT($Mc, Key$)    ▷ Mc is the encrypted message block and Key is the encryption key
    $SKeys[12] \leftarrow SubKeys(K)$;    ▷ Generating 12 sub keys
    for $i$ from 11 downto 0 do
        $Mc = AddKey(Mc, SKeys[i])$
        $Mc = invPMix(Mc)$
        $Mc = invIMix(Mc)$
        $Mc = invShift(Mc)$
    end for
    return $Mc$    ▷ $Mc$ contains the plaintext
end procedure
```

We remember here that a detailed description of $IMix$, $PMix$, $Shift$ are given in [3].

3 Differential Cryptanalysis Overview

Differential cryptanalysis was not known publicly until the year 1990. The first published work was the cryptanalysis of the *FEAL* algorithm by Murphy [9]. Since this time, Biham and Shamir demonstrated the feasibility of this method against a variety of encryption and hashing algorithm [1].

Today differential cryptanalysis are widely used to break some encryption algorithms and hashing functions [2]. The idea of differential cryptanalysis is to track the behaviour of pairs of plaintext blocs evolving along each iteration of the encryption process, in lieu of tracking the evolution of single plaintext block. The differential cryptanalysis is an attack of *chosen plaintext attack* family. That means the enemy needs to have the ability to encipher plaintexts using the secret key which is unknown to him.

4 Statistical Tests Overview

Statistical tests are series of mathematical operations used to prove the randomness of data samples. To prove the robustness of an encryption or hashing algorithm, **NIST** (*National Institute of Standards and Technology*) proposes 16 main tests [10]. These tests can be decomposed to sub tests, in this case we can have 189 tests in total. Brief description of the main tests is presented in [10]:

- Monobit frequency test: The purpose of this test is to determine whether the number of '1' and '0' in a binary sequence are approximately the same as would be expected for a truly random sequence.

- Frequency Test within a Block: The purpose of this test is to determine whether the frequency of '1' in an M-bit block is approximately $\frac{M}{2}$.

- Runs Test: This test determines whether the oscillation between '0' and '1' is too fast or too slow.

- Test for the Longest Run of Ones in a Block: The purpose of this test is to determine whether the length of the longest run of '1' within the tested sequence is consistent with the length of the longest run of '1' that would be expected in a random sequence.

- Binary Matrix Rank Test: The purpose of this test is to check for linear dependence among fixed length substrings of the original sequence.

- Discrete Fourier Transform Test: The purpose of this test is to detect periodic features in a binary sequence.

- Non-overlapping Template Matching Test: The purpose of this test is to detect generators that produce too many occurrences of a given aperiodic pattern.

- Overlapping Template Matching Test: Both this test and the Non-overlapping Template Matching test use an m-bit window to search for a specific m-bit pattern. The difference between this test and the Non-overlapping Template Matching test is that when the pattern is found, the window slides only one bit before resuming the search.

- Maurer's Universal Statistical Test: The purpose of the test is to detect whether or not the sequence can be significantly compressed without loss of information.

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

Table 1: Statistical tests and sub tests

Test name	**Number of P-Value**	**Identifiers**
Monobit frequency test	1	0
Frequency Test within a Block	1	1
Runs Test	1	2
Test for the Longest Run of Ones in a Block	1	3
Binary Matrix Rank Test	1	4
Discrete Fourier Transform Test	1	5
Non-overlapping Template Matching Test	148	6-153
Overlapping Template Matching Test	1	154
Maurer's Universal Statistical Test	1	155
Linear Complexity Test	1	156
Serial Test	2	157-158
Approximate Entropy Test	1	159
Cumulative Sums Test	2	160-161
Random Excursions Test	8	162-169
Random Excursions Variant Test	18	170-187
Lempel-Ziv Compression	1	188

- Serial Test: The purpose of this test is to determine whether the number of occurrences of the 2^m m-bit overlapping patterns is approximately the same as would be expected for a random sequence.

- Approximate Entropy Test: The purpose of the test is to compare the frequency of overlapping blocks of two consecutive/adjacent lengths (m and $m+1$) against the expected result for a random sequence.

- Cumulative Sums Test: The purpose of the test is to determine whether the cumulative sum of the partial sequences occurring in the tested sequence is too large or too small relative to the expected behaviour of that cumulative sum for random sequences.

- Random Excursions Test: The purpose of this test is to determine if the number of visits to a particular state within a cycle deviates from what one would expect for a random sequence.

- Random Excursions Variant Test: The purpose of this test is to detect deviations from the expected number of visits to various states in the random walk.

- Lempel-Ziv Compression: The purpose of this test is determine if the compression of a random binary sequence always give random sequence..

Table 1 gives a summary of statistical tests with the expected P-Value for each test.

5 Experimental Data Generation

To carry out statistical tests, 6 data sets are generated according to **NIST** recommendations. These data sets are generated as described in the following sub section.

5.1 Plaintext and Key Avalanche

To examine the sensibility of $CAES$ algorithm to input parameters changes (key or plaintext), 768 binary sequences of size 1048576 bits are tested. In case of key avalanche these sequences are generated as follow: Let $K_0, K_1 \ldots, K_{12287}$ be 12288 random encryption keys of 256 and a plaintext M with all bits equal to '0'. We have exactly 3145728 blocs of 256 as output of $CAES$. Each bloc is $B_i = E(M, K_i) \oplus E(M, K_i^j)$, where E is the encryption function, K_i is the i^{th} encryption key and K_i^j is the i^{th} key with the j bit is flipped for $0 \leqslant j \leqslant 255$. In case of plaintext avalanche, data are generated in the same fashion except the word 'key' is substituted with the word 'plaintext'.

5.2 CBC Encryption Mode

In category, binary sequences of 2097152 bits are generated using the CBC encryption mode. In total, we generate 200 sequences. Each sequence is created using a random key, an initialization vector (IV) with all bits equals to '0' and plaintext message with all bits equal to '0'.

5.3 Random Plaintext/Key

In this data set, we analyse 256 binary sequences. Each sequence is a concatenation of 4096 ciphertexts. These ciphertexts are generated using 4096 random plaintexts (respectively 4096 random keys) and a random key (respectively random plaintext) using CBC mode.

5.4 Plaintext/Ciphertext Correlation

To study the correlation between plaintexts and ciphertexts, 128 binary sequences of 1048576 bits are examined. Given a random key and 4096 random plaintexts, a binary

Table 2: Summary of binary sequences and size of each one

Data set	Number of sequences	Size of sequence (bits)
Plaintext avalanche	768	1048576
Key Avalanche	768	1048576
CBC encryption mode	20	2097152
Random plaintext	256	1040384
Random key	256	1040384
Plaintext/Ciphertext correlation	128	1048576
Low density plaintext	256	8421632
Low density key	256	8421632
High density plaintext	256	8421632
High density key	256	8421632

Table 3: Maximal acceptable number of sequences that maybe rejected by a test

Data set	Number of tests	Maximal (expected) number of rejected sequences
Plaintext avalanche	25	40
Key Avalanche	25	400
CBC encryption mode	25	150
Random plaintext	25	175
Random key	25	175
Plaintext/Ciphertext correlation	25	125
Low density plaintext	25	175
Low density key	25	175
High density plaintext	25	175
High density key	25	175

sequence is formed by concatenating the sum of plaintexts blocs and the corresponding ciphertexts blocs using XOR operator. The ciphertexts are calculated using ECB mode. By keeping plaintexts unchanged and changing the random key, we obtain the rest of the data sets.

5.5 Low Density Key/Plaintext

In this category, two data sets are created which can be used either as plaintexts or as keys. Each set is formed of 256 sequences. Each sequence consists of 32897 ciphertexts blocs calculated using ECB mode. Ciphertexts are formed by a plaintext (or key) of 256 bits with all bits are '0', 256 plaintexts (or keys) one bit equal to '1' and other bits equal '0' (each plaintexts corresponds to a given position of bit '1'), and 32640 plaintexts with two bits equal to '1' and other bits equal to '0' (all possible combination).

5.6 High Density Key/Plaintext

Data of this category are generated in the same manner as of the previous category, except that data of this category is the binary negation of data of the previous category.

6 Statistical Tests: Results and Discussion

Statistical tests are the most advanced tests that must be achieved to prove the robustness of a given cryptosystem. these tests are also used to test the reliability of encryption algorithms such as AES [11], hashing functions such as SHA-3 [5] and pseudo random number generator such as Blum-Blum-Shub [7] and the algorithm described in [4]. In this section we present the results of these tests when applied to $CAES$ algorithm.

6.1 Empirical Analysis

In our experimental analysis, the significance level is fixed at $\alpha = 0.01$, that is, to say a test is successful if the rate of rejected sequences is less or equal to 1%, which is the ideal case. In practice, the interval of confidence is used. In this case, the maximal number of rejected sequence is $n(\alpha + 3\sqrt{\frac{\alpha(1-\alpha)}{n}})$ where n is the number of binary sequences and α is the significance level. Table 2 gives a summary of data sets sizes and Table 3 gives the number of carried out tests and maximal number of rejected sequences.

6.2 Results and Discussion

After running various statistical tests using data categories and sequences defined previously we got the results shown in Figures 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10.

It is observed, in these figures, that the number of rejected sequences is less than the maximal (expected) number of rejected sequences, which means that the test was successful.

According to these results, it is clear that the $CAES$ algorithm generates pseudo-random outputs regardless the inputs. This result demonstrates a highly sought after property in robust cryptosystems to resists against cryptanalytic attacks. As consequence, the $CAES$ resists perfectly against statistical attacks and can be used to send safely secret data over a public network.

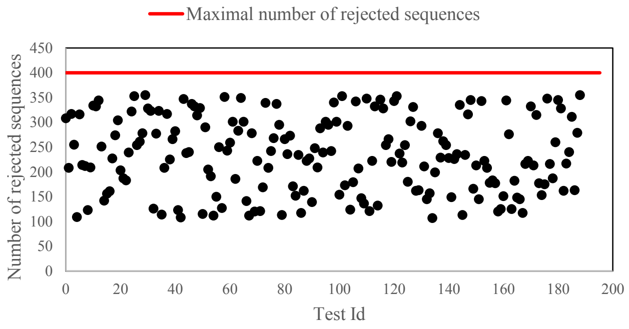

Figure 1: Statistics results using "Plaintext avalanche" data set

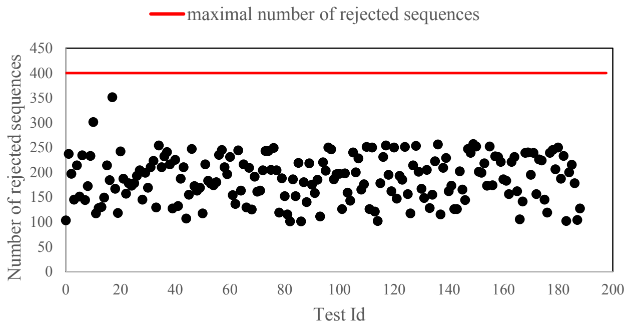

Figure 2: Statistics results using "Key avalanche" data set

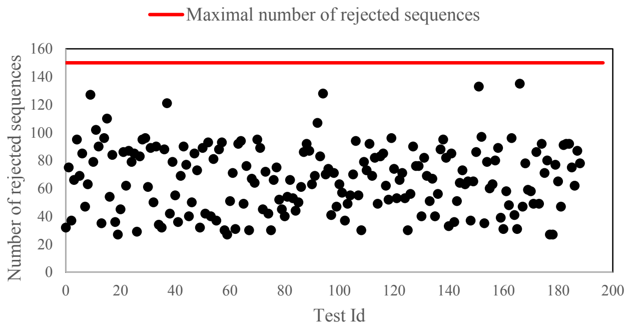

Figure 3: Statistics results using "CBC encryption mode" data set

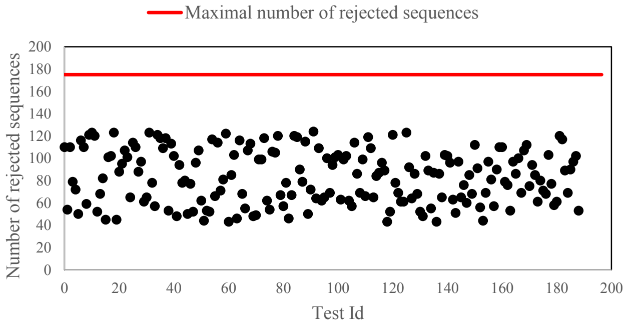

Figure 4: Statistics results using "Random plaintext" data set

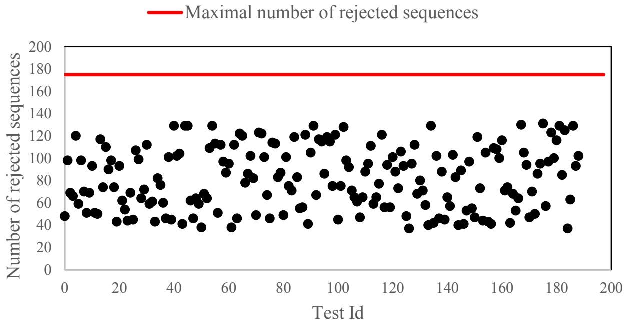

Figure 5: Statistics results using "Random key" data set

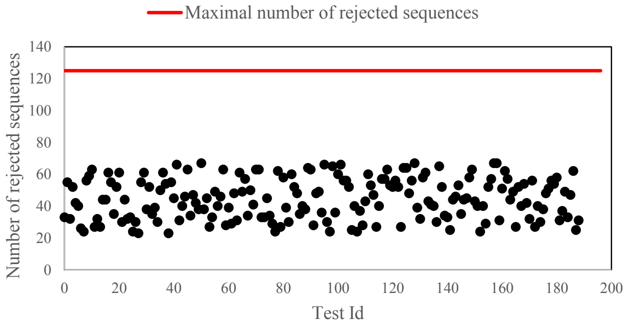

Figure 6: Statistics results using "Plaintext/Ciphertext correlation" data set

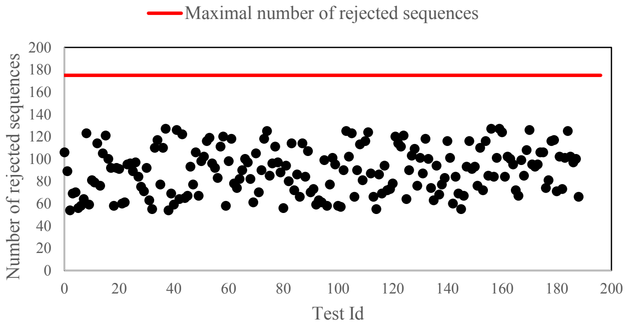

Figure 7: Statistics results using "Low density plaintext" data set

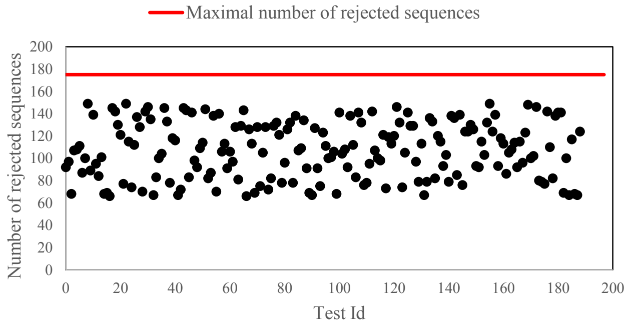

Figure 8: Statistics results using "Low density key" data set

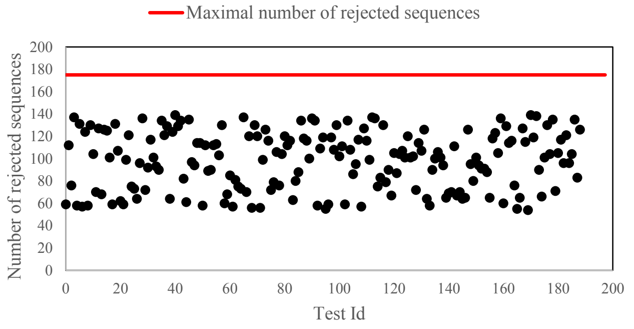

Figure 9: Statistics results using "High density plaintext" data set

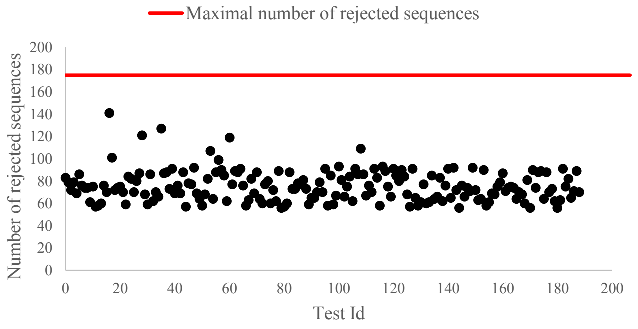

Figure 10: Statistics results using "High density key" data set

7 Differential Cryptanalysis Results

To prove the resistance of *CAES* against differential cryptanalysis, several tests and calculations are carried out. These tests and calculations focused on non linear transformations, i.e *IMix* and *PMix*. Our goals are to find plaintext messages m_i with a difference $X_i = m_i \oplus m_{i+1}$ producing ciphertext messages c_i with a difference $Y_i = c_i \oplus c_{i+1}$ with high probability. Table 4 gives the probability distribution of all possible differences X_i and the corresponding differences Y_i of *IMix* and *PMix* (they have the same distribution difference table).

Table 4: Differences distribution of *IMix* and *PMix* transformations

X_i \ Y_i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	2	1	0	1	0	0	1	1	1	0
2	0	0	0	0	1	0	0	1	0	1	1	0	2	0	0	2
3	0	1	1	1	0	0	1	0	1	0	2	0	0	0	1	0
4	0	2	0	0	0	0	0	0	2	1	0	1	1	0	1	0
5	0	0	0	0	1	1	1	1	1	0	1	0	0	1	0	1
6	0	0	1	0	2	1	1	1	1	0	0	0	0	0	1	0
7	0	0	2	1	0	0	1	0	1	1	0	1	0	0	0	1
8	0	0	1	0	1	0	0	0	0	4	1	0	1	0	0	0
9	0	0	0	0	1	2	0	1	0	0	0	0	0	1	1	2
10	0	0	0	1	0	2	1	0	0	0	0	1	0	2	1	0
11	0	0	0	2	1	0	0	1	1	0	2	1	0	0	0	0
12	0	2	1	1	0	1	0	1	1	0	0	1	0	0	0	0
13	0	1	0	1	1	0	1	0	0	0	0	0	1	1	1	1
14	0	0	1	0	0	0	0	1	0	0	0	1	2	2	1	0
15	0	1	1	1	0	1	0	0	0	0	1	2	0	0	0	1

According to Table 4, the output difference $Y_i = 0$ is caused by the input difference $X_i = 0$ with probability $\frac{8}{16} = \frac{1}{2}$. If $X_i = m_i \oplus m_{i+1} = 0$ then $m_i = m_{i+1}$ therefore $c_i = c_{i+1}$ and as consequence no useful information about the key can be extracted using this highest value (8).

The highest exploitable value on the distribution table is 4, so an output difference Y_i of *PMix* and *IMix* is likely caused by an input difference X_i with probability $\frac{4}{16} \times \frac{2}{16} = \frac{1}{32}$.

To prove the robustness of *CAES* against differential attack, we have chosen plaintexts messages m_i of difference X_i producing ciphertexts messages c_i of difference Y_i using the highest probability according to table 4 ($\frac{1}{32}$). The plaintexts messages are generated using these steps:

1) Choose the difference X_i from 4 which have the highest probability.

2) Generate a random message.

3) XOR the data from Step (1) and Step (2).

Table 5 gives an example of plaintext messages generation process.

Table 5: Example of plaintext message generated from a given difference

Difference	00800000000000000000000000000000 00000000000000000000000000000000
Random message	F622919DE18B1FDAB0CA9902B9729D49 2C807EC599D5E980B2EAC9CC53BF67D6
Resulting message	F6A2919DE18B1FDAB0CA9902B9729D49 2C807EC599D5E980B2EAC9CC53BF67D6

Suppose that the probability to have an output difference Y_i caused by an input difference X_i is exactly the probability given by the distribution table 4. In this ideal case, we need to generate and encrypt at least 2^{71} plaintext messages or $6.9 \times 10^{10} TB$ (Tera Byte) of data, which is higher than all stocked data on the internet. Therefore, we can assume that differential attack against complete version of *CAES* cryptosystem is very difficult if not impossible.

In practice, we have been able to cryptanalyze the reduced version of CAES (one iteration version and without $Shift$ transformation) using 67 chosen plaintext messages. For a version of $CAES$ with higher number of iteration (> 2) tracking the encryption evolution at each iteration of the algorithm become very difficult. Indeed, we found that the probability to have the expected value of the output at the second iteration is $\frac{1}{128}$. As conclusion, differential attack against the full version of $CAES$ is impossible at this time.

8 Conclusion and Perspective

In the current paper we presented several tests to prove the robustness of $CAES$ encryption algorithm. The obtained results prove that the output of $CAES$ is random regardless the input, which prove that the algorithm hide all useful information about the original data. And also, we presented results of differential attack against $CAES$, the results proved that this attack have no effects against this algorithm. As perspectives, other tests and attacks; such as linear attacks and timing attacks; will be carried out in the near future.

References

[1] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, London, UK, Springer-Verlag, 1993.

[2] C. Blondeau, G. Leander, and K. Nyberg, "Differential-linear cryptanalysis revisited," in *International Workshop on Fast Software Encryption*, pp. 411–430, 2014.

[3] S. Bouchkaren and S. Lazaar, "A new iterative secret key cryptosystem based on reversible and irreversible cellular automata," *International Journal of Network Security*, vol. 18, no. 2, pp. 345–353, 2016.

[4] K. Charif, A. Drissi, and Z. Guennoun, "A pseudo random number generator based on chaotic billiards," *International Journal of Network Security*, vol. 19, no. 3, pp. 479–486, 2017.

[5] A. Doganaksoy, B. Ege, O. Koçak, and F. Sulak, "Statistical analysis of reduced round compression functions of sha-3 second round candidates.," *IACR Cryptology ePrint Archive*, vol. 2010, p. 611, 2010.

[6] T. Gulom, "The encryption algorithm GOST28147-89-PES16-2 and GOST28147-89-RFWKPES16-2," *International Journal of Electronics and Information Engineering*, vol. 6, no. 1, pp. 1–11, 2017.

[7] P. Junod, *Cryptographic Secure Pseudo-Random Bits Generation: The Blum-Blum-Shub Generator*, 1999. (`http://crypto.junod.info/bbs.pdf`)

[8] A. Mersaid, T. Gulom, "The encryption algorithm AES-RFWKIDEA32-1 based on network RFWKIDEA32-1," *International Journal of Electronics and Information Engineering*, vol. 4, no. 1, pp. 1–11, 2016.

[9] S. Murphy, "The cryptanalysis of FEAL-4 with 20 chosen plaintexts," *Journal of Cryptology*, vol. 2, no. 3, pp. 145–154, 1990.

[10] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, *A Statistical Test Suite for Random and Pseudorandom Number Generators For Cryptographic Applications*, Technical Report SP800-22, National Institute of Standards & Technology, 2001.

[11] J. Soto and L. Bassham, *Randomness Testing of the Advanced Encryption Standard Finalist Candidates*, Technical Report NIST IR 6483, National Institute of Standards and Technology, 2000.

[12] C. N. Zhang, Q. Yu, and X. W. Liu, "A hybrid fault tolerant approach for AES", *International Journal of Network Security*, vol. 15, no. 4, pp. 291–297, 2013.

Biography

Said Bouchkaren received his state engineer degree in software engineering and PhD degree in information security and cryptography from AbdelMalek Essaadi University, Morocco, in 2010 and 2016 respectively. In 2011, He joined the department of Computer sciences and mathematics as a professor. His research focuses on cryptography and information security.

Saiida Lazaar started her scientific career with a research contract funded by CNRS in France where she prepared her Ph.D. in applied mathematics. She has held positions as a researcher with IFP in France and with ONDRAF in Belgium. Currently, she is a full Professor at the University of AbdelMalek Essaadi in Morocco and Head of Master Cybersecurity and cybercriminality. She published various works, special issues in international journals, and a book on Security of networks and Cryptography