CA-PCS: A Cellular Automata based Partition Ciprohing System

Fatima Ezzahra Ziani, Anas Sadak, Charifa Hanin, Bouchra Echandouri, Fouzia Omary
Faculty of Sciences, University Mohammed V
Computer Science Department
Rabat, Morocco

Abstract—In this paper, the authors present a modified version of the Partition Ciphering System (PCS) encryption system previously proposed. The previously developed encryption system PCS uses the partition problem to encrypt a message. The goals of newly developed system are avoiding statistical and frequency attacks, by providing a balance between 0s and 1s, ensuring a good level of entropy and achieving confidentiality through encryption. One of the novelties of the new design compared to its predecessor is the use of cellular automata (CAs) during the encryption. The use of CAs is justified by their good cryptographic properties that provide a level of security against attacks, and better confusion and diffusion properties. The new design is first presented with details of the encryption and decryption mechanisms. Then, the results of the DIEHARDER battery of tests, the results of the avalanche test, a security analysis and the performance of the system are outlined. Finally, a comparison between CA-PCS and PCS as well as the AES encryption system is provided. The paper shows that the modified version of PCS displays a better performance as well as a good level of security against attacks.

Keywords—Partition ciphering system; partition problem; frequency analysis; cellular automata; avalanche effect; confusion; diffusion; statistical properties; cryptographic properties

I. INTRODUCTION

One of the five pillars of cryptography is achieving confidentiality. This latter comprises two principles: data confidentiality and privacy. Data confidentiality ensures that no data is accessed or revealed to unauthorized parties. Privacy controls the access to data and storage of data by concerned parties [1]. This paper presents a modified version of the Partition Ciphering System (PCS), which was previously developed by the authors [2]. It is a symmetrical encryption system based on the partition problem, more precisely the Card-Partition version. The use of the partition problem in PCS was motivated by the fact that it changes the frequency of the appearance of characters between the plaintext and the ciphertext. Consequently, PCS is robust against frequency cryptanalysis: an adversary cannot learn any information about the plaintext from the ciphertext. However, PCS has some limitations to check the diffusion property and resistance to some attacks like linear and differential attacks. A cellular automaton (CA) is a suitable candidate to provide better confusion and diffusion. Also, the CA cryptographic properties could be studied to verify the security level. These later are nonlinearity, algebraic degree, balancedness, resiliency, and correlation immunity. A CA is a dynamic system involving a network of cells. CAs are widely used in cryptography and other fields to benefit from their simplicity, parallelism, and unpredictability. Besides, CAs make the hardware and software implementations easier [3]. In this paper, a new design called CA-PCS (Cellular Automata based Partition Ciphering System) is proposed. It consists of a hybrid CA, with satisfying cryptographic properties, that evolves multiple iterations to increase resistance to linear and differential attacks, followed by the insertions of necessary blocks so that the frequency of all the blocks is the same. In addition to a random permutation is applied to the results of the second step. Each layer produces better confusion and diffusion, and consequently, better resistance to linear and differential cryptanalysis. Also, the cryptographic properties of the CA ruleset are studied and display good results. A high nonlinearity, high algebraic degree, and balancedness are satisfied. CA-PCS was compared to AES and PCS in terms of randomness, security, and performance. Thus, the CA-PCS results are satisfying.

The rest of this article is organized as follows: In Section 2, a brief background on cellular automata is presented. Next, in Section 3, the related works are included. Then, CA-PCS is detailed in Section 4. Section 5 provides a brief description of the PCS and AES encryption systems. Finally, Section 6 presents results and security analysis.

II. BACKGROUND ON CELLULAR AUTOMATA

The history of cellular automata goes back to the 1940s when Stanislaw Ulam [4] initiated their study by taking interest in self-replicating automata. Then in the 1960s, John von Neumann used them in Biology for modeling self-reproduction [5]. They were later on popularized by John Conway’s game of life in the 1970s [6]. They were first use in cryptography by Stephen Wolfram in the 1980s [7]. Simply put, a cellular automaton is a network of cells, each of which has a state that changes from a time step $t$ to a time step $t+1$ according to a defined local rule and depending on its neighbors. The interest of the scientific community in cellular automata stems from the fact that simple local calculations at the cells scale produce a complex behavior at the automaton scale. Another interesting aspect of using cellular automata is that both uniformity and non-uniformity can be modeled. A cellular automaton is defined as $[3] (d, L, S, N, f)$, where $d$ represents the cellular space dimension, $L$ represents the cellular space, $S$ is the finite set of states, $N$ is the neighborhood vector and $f$ or $(f_1, f_2, ...)$ is the local rule or ruleset respectively. The global rule of the cellular automata is designated by $\Phi$.

By modifying the tuple $(d, L, S, N, f)$, different kinds of cellular automata can be obtained. One interesting type of cellular automata was introduced by Wolfram in [8]. This kind
CAs were also used for image encryption by Li in [22], who studied by Mehta [18] and Bouchkaren [19]. Two dimensional
[16] and Roy [17]. Non uniform one-dimensional CAs were
block cipher using one dimensional programmable CAs. Other
algorithm is the design of Das et al. [15] who proposed a
A more recent example of the use of CAs in an encryption
back to Wolfram in [7]. He applied rule 30 to design a
produces better results compared to the previous algorithms.

<table>
<thead>
<tr>
<th>Rule</th>
<th>105</th>
<th>135</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear?</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Algebraic Normal Form</td>
<td>$1 \oplus x_{i-1} \oplus x_i \oplus x_{i+1}$</td>
<td>$1 \oplus x_{i-1} \oplus x_i \oplus x_{i+1}$</td>
</tr>
<tr>
<td>Truth table</td>
<td>111</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>110</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>101</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>011</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>010</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>001</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>000</td>
<td>1</td>
</tr>
</tbody>
</table>

III. RELATED WORK

The partition problem or Equal Piles Problem, which is the
source of inspiration for this work, was first studied by
Jones and Beltramo in [9], where they defined a challenging
instance. They tried nine standard genetic algorithms, but without
finding an optimal solution. To solve this instance of the problem,
Falkenauer [10] and William [11] proposed particular types of genetic algorithms. Concretely, Falkenauer [10] tried to adjust the grouping genetic algorithm that he designed previously using specific crossover and mutation operators suited for similar problems. William [11] used a particular approach in the design of the Eager Breeder genetic algorithm, which makes the manipulation of genetic materials easier and produces better results compared to the previous algorithms. However, their results are not that good for this article’s proposed design. More recently, evolutionist algorithms were also used to come up with a solution to the partition problem as in the works of Trichni [12], Bourgrine [13] and Kaddouri [14].

The first use of cellular automaton in cryptography goes back to Wolfram in [7]. He applied rule 30 to design a pseudorandom number generator (PRNG) and a stream cipher. A more recent example of the use of CAs in an encryption algorithm is the design of Das et al. [15] who proposed a block cipher using one dimensional programmable CAs. Other works using one dimensional uniform CAs include Bhaumik [16] and Roy [17]. Non uniform one-dimensional CAs were studied by Mehta [18] and Bouchkaren [19]. Two dimensional uniform CAs were used by Bouchkaren [20] and Farazou [21]. CAs were also used for image encryption by Li in [22], who made use of two dimensional non-uniform CAs. Other image encryption schemes can be found in [23] and [24].

IV. CA-PCS DESIGN

A. CA-PCS Encryption Algorithm

The CA-PCS encryption scheme goes through three steps:

1) CA Evolution: The first step includes the hybrid CA evolving of the binary message using the rules [90, 150, 30, 180, 45, 90, 150, 30]. Linear rules 90 and 150 provide better diffusion property and high cycle length [25]. While nonlinear rules 30, 45, and 180 provide better confusion property [26]. Moreover, these rules provide resistance to linear attacks and differential attacks. Because of the high nonlinearity met after a few iterations and the significant algebraic degree.

2) Blocks Insertion: The second step consists of representing the first step’s result as a partition and add some blocks at random positions to get the same appearance frequency for all blocks. At first, the CA output is split into blocks of a randomly chosen size $2 \leq k \leq 16$. Then the ideal cardinality IC is computed $IC = max\{\text{Card}(L_1), \text{Card}(L_2), ..., \text{Card}(L_m)\}$. Next, for each block $B_i$, the cardinality of the corresponding $L_i$, representing the positions of $B_i$ in the CA output, is compared to the IC. Accordingly, if $\text{Card}(L_i) < IC$, then $B_i$ is inserted in a random position $1 < p_{ij} < \text{size(CAoutput)}$ where $0 < j < IC - \text{Card}(L_i)$. Next, the $p_{ij}$ is inserted in the ListOfInsertedBlocksPositions.

3) Permutation: Finally, a random permutation is applied to the set $\{L_1, L_2, ..., L_m\}$. This permutation is useful to change the blocks’ occurrence lists $L_i$. It is denoted formally by $\pi: S \rightarrow S$ where $S$ is a set of $m$ elements. $m!$ permutation of $\{L_1, L_2, ..., L_m\}$ are possible. A possible example of a random permutation for $m=10$, $\pi: \{L_1, L_2, L_3, L_4, L_5, L_6, L_7, L_8, L_9, L_{10}\} \rightarrow \{L_2, L_4, L_1, L_6, L_3, L_9, L_7, L_{10}, L_5, L_8\}$. Following this example, $L_1 \rightarrow L_2, L_2 \rightarrow L_4, L_3 \rightarrow L_1, L_4 \rightarrow L_6, L_5 \rightarrow L_3, L_6 \rightarrow L_9, L_7 \rightarrow L_7, L_8 \rightarrow L_{10}, L_9 \rightarrow L_8, L_{10} \rightarrow L_5$. Accordingly, $B_1$ will appear in the positions of $B_2, B_2$ will appear in those of $B_3$, and so on.

4) Key generation: The secret key comprises four elements:

$SK = \{k, CASeq, ListOfInsertedBlocksPositions, PSeq\}$

The random integer $k$ is the blocks size. The CASeq binary sequence where $CASeq = M \oplus M'$ where $M$ is the plaintext, and $M'$ is the output of the CA evolution step. The ListOfInsertedBlocksPositions which comprises the positions where blocks are inserted. The PSeq binary sequence $PSeq = M'' \oplus C$ where $M''$ is the output of the blocks insertion step and $C$ is the ciphertext. Fig. 1 summarizes the encryption process of CA-PCS.

B. CA-PCS Decryption Algorithm

The CA-PCS decryption process, as Fig. 2 displays, is as follows, given the ciphertext $C$ and the secret key $SK = \{k, CASeq, ListOfInsertedBlocksPositions, PSeq\}$:

At first, the PSeq sequence is XORed with the ciphertext to get $M''$. Then, $M''$ is split into blocks of size $k$. Next, inserted blocks are removed from $M''$ using the ListOfInsertedBlocksPositions to get $M'$. Then $M'$ is XORed with the CASeq to get the plaintext.

www.ijacsa.thesai.org 604 | Page
Algorithm 1 CA-PCS Encryption Algorithm

Input: The message M
Output: The ciphertext C and the secret key K
Begin
\[ M \leftarrow M \oplus \text{secret K} \]
\[ \text{size} \leftarrow \text{sizeOf}(M) \]
\[ k \leftarrow \text{random}(2, 16) \quad \text{random integer} 2 < k < 16 \]
\[ \text{ruleSet} \leftarrow \{30, 90, 150, 30, 180, 45, 90, 150\} \]
for \( i < j \) do
  \[ M''[i] \leftarrow M''[i] \oplus M[i] \oplus M[i + 1] \]
end for
\[ \text{CASeq} \leftarrow \text{CASeq} \oplus \text{Partition} \]
\[ \text{ListOfInsertedBlocksPositions} \leftarrow \{B_1, \ldots, B_m\} \]
\[ \text{IC} \leftarrow \text{Earl} \leftarrow \text{IdealCardinality} \]
for \( i < m \) do
  \[ \text{Card}(L_i) < \text{IC} \]
  \[ M'' \leftarrow \text{insert}(B_i, M'', \text{randomPosition}) \]
  \[ \text{InsertListOfInsertedBlocksPositions}, \text{randomPosition} \]
  \[ \text{InsertListCASeq, randomPosition} \]
end while
\[ \text{CASeq} \leftarrow \text{CASeq} \oplus M'' \]
\[ \text{secret K} \leftarrow \{k, \text{CASeq}, \text{ListOfInsertedBlocksPositions}, \text{P Sec}\} \]
End

Algorithm 2 Decryption Algorithm

Input: The secret key SK and the ciphertext C
Output: The message M
Begin
\[ M'' \leftarrow C \oplus \text{P Sec} \]
\[ M'' \leftarrow \text{DivideIntoBlocks}(M'', k) \]
for \( i \) from \( \text{sizeOf}(\text{ListOfInsertedBlocksPositions}) \) to 1 do
  \[ M' \leftarrow \text{Remove}(M''[i], \text{ListOfInsertedBlocksPositions}[i]) \]
end for
\[ M \leftarrow M' \oplus \text{CASeq} \]
End

V. THE PCS AND AES DESCRIPTION

This section presents a brief description of a previously developed scheme: Partition Ciphering System (PCS) and the Advanced Encryption Standard (AES).

A. Partition Ciphering System (PCS)

The Partition Ciphering System PCS [2] is a symmetric encryption scheme that encrypts a plaintext in three steps. The first step consists of the construction of a partition from the plaintext, which is initially split into blocks of size \( k \geq 2 \). Each block is associated with a list of occurrences. This partition undergoes some transformations in a way to make the ciphertext resistant to frequency cryptanalysis. Next, the ideal cardinality IC is computed: let \( c = \frac{n}{m} \). The secret key is computed: the secret key SK is generated as a random integer between 1 and 16. The ideal cardinality IC is computed: let \( c = \frac{n}{m} \), where \( n \) is the number of blocks in the plaintext, and \( m \) is the number of different blocks in the plaintext.
the plaintext. If $c \in \mathbb{N}$ then $IC = c$ else $IC = [c]$. This cardinality defines the number of occurrences of each block in the ciphertext. In the last step, the blocks $B_i$s are inserted or deleted according to the cardinal of the corresponding list of appearances $L_i$. When the $Card(L_i) < IC$, then the block $B_i$ is appended to the message. When the $Card(L_i) > IC$, then the block $B_i$ is deleted from a random position.

B. Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) [27] is a symmetric cipher that encrypts 128-bit blocks using keys of size 128 bit, 192 bit, or 256 bit. It comprises N rounds, where N changes according to the length of the key: for a 128-bit key, 12 for a 256-bit key, and 14 for a 192-bit key. In the first step, the plaintext is XORed by the first 128 bit of the key. Next, for N-1 iteration, four operations are performed: SubBytes, ShiftRows, MixColumns, and AddRoundKey. [27] provides a detailed description of these operations. Finally, the last round consists of only SubBytes, ShiftRows, and AddRoundKey operations.

VI. RESULTS AND SECURITY ANALYSIS

This section displays the statistical tests and the confusion and diffusion properties of CA-PCS compared to the AES.

A. Dieharder Test

The battery of tests Dieharder was designed by Robert G. Brown to check out the behavior of PRNGs and cryptographic primitives like encryption systems, hash functions, and MACs. It involves tests from diehard, some NIST tests, and other tests developed by Brown and Bauer [28]. The authors generated three files of 10 Mb using PCS, CA-PCS, and AES ciphers. Then, they run the battery overs these files. Table II displays the results. The P-values are the probability that the generated sequences are random. If $0.005 < P-value < 0.995$, then the systems pass the test. Since $0.10 < P-values(PCS) < 0.91$, $0.2 < P-values(CA-PCS) < 0.92$, and $0.05 < P-values(AES) < 0.95$, then all the systems pass all the tests. Also, the P-values of the ciphers are uniformly distributed in the range [0, 1], to conclude, CA-PCS displays good results regarding the statistical tests compared to PCS and AES.

B. Confusion and Diffusion Tests

This section presents the confusion and diffusion properties of the CA-PCS system in comparison with AES. A secure encryption system from statistical analysis, as stated by Shannon [29], has good confusion and diffusion properties (e.g., AES is a secure system). If the relation between the ciphertext and the secret key is hidden, then the confusion property is verified. In other terms, replacing one bit in the secret key has an impact on most of the bits in the ciphertext. If the relation between the plaintext and the ciphertext is masked, then the diffusion property is checked. In other words, changing one bit in the plaintext affect almost all the bits of the ciphertext. Fig. 3 shows the confusion property for CA-PCS compared to the AES. According to Fig. 3, the percentage of the changed bits in the ciphertext is approximately 50% for CA-PCS and AES. Concretely, the values for CA-PCS are between 0.40% and 0.61%, while the values for AES are between 0.36% and 0.61%. These values confirm that CA-PCS has better diffusion property. Fig. 4 illustrates the diffusion property of CA-PCS and AES. The mean value of the percentages of changed bits in the ciphertext is nearly 50%. The values for CA-PCS are

<table>
<thead>
<tr>
<th>Tests names</th>
<th>P-values CA-PCS</th>
<th>P-values AES</th>
<th>P-values PCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diehard birthdays</td>
<td>0.5357</td>
<td>0.0836</td>
<td>0.8625</td>
</tr>
<tr>
<td>Diehard operm5</td>
<td>0.4946</td>
<td>0.0967</td>
<td>0.8971</td>
</tr>
<tr>
<td>Diehard rand 32x32</td>
<td>0.5887</td>
<td>0.7711</td>
<td>0.1402</td>
</tr>
<tr>
<td>Diehard rand 6x8</td>
<td>0.7192</td>
<td>0.6936</td>
<td>0.3240</td>
</tr>
<tr>
<td>Diehard bitstream</td>
<td>0.4615</td>
<td>0.6593</td>
<td>0.4530</td>
</tr>
<tr>
<td>Diehard opso</td>
<td>0.5559</td>
<td>0.7204</td>
<td>0.3559</td>
</tr>
<tr>
<td>Diehard oqso</td>
<td>0.5092</td>
<td>0.6363</td>
<td>0.1898</td>
</tr>
<tr>
<td>Diehard dna</td>
<td>0.5686</td>
<td>0.3142</td>
<td>0.2811</td>
</tr>
<tr>
<td>Diehard count 1s str</td>
<td>0.4114</td>
<td>0.8797</td>
<td>0.6988</td>
</tr>
<tr>
<td>Diehard count 1s byt</td>
<td>0.6995</td>
<td>0.8451</td>
<td>0.7611</td>
</tr>
<tr>
<td>Diehard parking lot</td>
<td>0.2622</td>
<td>0.8514</td>
<td>0.7033</td>
</tr>
<tr>
<td>Diehard 2dsphere</td>
<td>0.4555</td>
<td>0.5370</td>
<td>0.7910</td>
</tr>
<tr>
<td>Diehard 3dsphere</td>
<td>0.6735</td>
<td>0.3863</td>
<td>0.2487</td>
</tr>
<tr>
<td>Diehard squeeze</td>
<td>0.6888</td>
<td>0.8732</td>
<td>0.7991</td>
</tr>
<tr>
<td>Diehard sums</td>
<td>0.9130</td>
<td>0.0058</td>
<td>0.1779</td>
</tr>
<tr>
<td>Diehard runs</td>
<td>0.2342</td>
<td>0.3810</td>
<td>0.7702</td>
</tr>
<tr>
<td>Diehard craps</td>
<td>0.7063</td>
<td>0.8630</td>
<td>0.9093</td>
</tr>
<tr>
<td>Marsaglia tsang gcd</td>
<td>0.6682</td>
<td>0.7107</td>
<td>0.4046</td>
</tr>
<tr>
<td>Sts monobit</td>
<td>0.5815</td>
<td>0.6915</td>
<td>0.54319</td>
</tr>
<tr>
<td>Sts runs</td>
<td>0.4394</td>
<td>0.4656</td>
<td>0.1070</td>
</tr>
<tr>
<td>Sts serial</td>
<td>0.6616</td>
<td>0.5643</td>
<td>0.6388</td>
</tr>
<tr>
<td>Rgb bitdist</td>
<td>0.6689</td>
<td>0.5724</td>
<td>0.4844</td>
</tr>
<tr>
<td>Rgb minimum distance</td>
<td>0.5515</td>
<td>0.3475</td>
<td>0.4441</td>
</tr>
<tr>
<td>Rgb permutations</td>
<td>0.6639</td>
<td>0.6588</td>
<td>0.4145</td>
</tr>
<tr>
<td>Rgb lagged sum</td>
<td>0.5074</td>
<td>0.5363</td>
<td>0.6067</td>
</tr>
<tr>
<td>Rgb kstest test</td>
<td>0.2840</td>
<td>0.4934</td>
<td>0.1025</td>
</tr>
<tr>
<td>dab bytedistrib</td>
<td>0.5920</td>
<td>0.4758</td>
<td>0.2636</td>
</tr>
<tr>
<td>dab def</td>
<td>0.8842</td>
<td>0.9448</td>
<td>0.8735</td>
</tr>
<tr>
<td>dab filltree2</td>
<td>0.4757</td>
<td>0.4721</td>
<td>0.5212</td>
</tr>
<tr>
<td>dab filltree</td>
<td>0.8987</td>
<td>0.7090</td>
<td>0.3727</td>
</tr>
<tr>
<td>dab monobit2</td>
<td>0.8994</td>
<td>0.0507</td>
<td>0.6055</td>
</tr>
</tbody>
</table>
between 41% and 61%, and the values for AES are between 37% and 67%. Consequently, CA-PCS has better diffusion.

C. Encryption and Decryption Time of CA-PCS, AES and PCS

This part (Fig. 5) compares the encryption and decryption time of CA-PCS with the previously developed scheme PCS and AES. Fig. 5 shows that CA-PCS requires less time in the encryption process compared to PCS and AES. While the PCS and AES take the same time to encrypt. The time of decryption is approximately the same for CA-PCS, PCS, and AES. To conclude CA-PCS displays good results.

D. Frequency Analysis

This part presents the frequency analysis of the outputs of CA-PCS and PCS. As mentioned in [2], the purpose was to have a ciphertext with blocks appearing with the same frequency, so that frequency analysis does not reveal any information about the plaintext. As CA-PCS is an improved version of PCS, the same objective persists. CA-PCS is different from PCS in all steps. The CA evolution is the first step of CA-PCS. Next, the ideal cardinality computation. Later, the insertion of blocks follows. The resulting intermediate output undergoes a permutation. While in PCS, the ideal cardinality is computed in a way to have blocks to add or remove. The objective of CA-PCS design is to provide better confusion and diffusion, in addition to resistance to some attacks like linear and differential attacks. Fig. 6 represents the frequency analysis performed on the outputs of CA-PCS and PCS for the same plaintext. Fig. 6 shows that frequency analysis will never divulge any information. As a result, frequency cryptanalysis is impossible.

E. Cryptographic Properties of the Ruleset Used in the CA Evolution

This section presents the cryptographic properties, namely, nonlinearity, algebraic degree, correlation immunity, resiliency, and balancedness, of the CA ruleset \{30, 90, 150, 30, 180, 45, 90, 150\}. It is applied alternately on the CA cells in the evolution step. In this section, to study the ruleset, an example of 8 cells is considered. Tables III to VII shows the variation of the cryptographic properties with iterations.

Nonlinearity and algebraic degree increase significantly within iterations. Also, balancedness persists. The resiliency and the correlation immunity decrease with iterations because high nonlinearity affects the level of resiliency and correlation immunity. Most of the cryptographic systems require high

---

Fig. 3. Confusion Test of CA-PCS and AES

Fig. 4. Diffusion Test of CA-PCS and AES

Fig. 5. Encryption and decryption time of CA-PCS, PCS and AES

Fig. 6. Frequency of blocks before and after encryption for CA-PCS and PCS
Long term security. Since the AES has three versions, AES-128, AES-192, and AES-256, both security levels can be satisfied. PCS, from [2], has a secret key of size greater than 256 bit. Also, CA-PCS has a secret key of at least 256 bit. Unless an attacker has a quantum computer, he cannot get the secret key to decrypt to an intelligible plaintext. Table VIII summarises the security level of AES, PCS, and CA-PCS.

### G. Linear and Differential Attacks

Linear attack analyzes the linear approximations of the plaintext, the ciphertext, and the secret key [30]. It is a known-plaintext attack, while differential attack studies the differences between plaintexts and ciphertexts [31]. It is a chosen-plaintext attack. A cipher should be robust against the linear and differential attacks. The confusion property, which is satisfied using the nonlinear parts of the system, is necessary to resist these types of attacks. In general, S-Boxes are responsible for this purpose. But, other primitives, like nonlinear cellular automata, can lead to the same results. In CA-PCS, the ruleset used to evolve the CA has high nonlinearity, and maintain the balancedness. These features make these attacks difficult for a cryptanalyst.

### VII. Conclusion

In this article, an enhanced version of PCS, a previously developed encryption scheme, is proposed. The proposed system, called CA-PCS, makes use of cellular automata to increase the security level of the design. Precisely, the ruleset used provides satisfying results in terms of cryptographic properties, randomness tests, confusion, and diffusion properties. Linear and differential attacks are difficult to achieve because of the high non-linearity and the high algebraic degree provided by the ruleset. Also, the balancedness and the randomness produce resistance to statistical cryptanalysis. Moreover, CA-PCS is robust against brute force attacks. Besides, the performance of CA-PCS is better than PCS and AES. In future work, the authors will extend the proposed scheme to ensure authentication.

### References


---

**TABLE VIII. B RUTE F ORCE A TTA CK OF AES, PCS AND CA-PCS**

<table>
<thead>
<tr>
<th>Encryption schemes</th>
<th>AES-128</th>
<th>AES-192</th>
<th>AES-256</th>
<th>PCS</th>
<th>CA-PCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key length</td>
<td>128 bit</td>
<td>192 bit</td>
<td>256 bit</td>
<td>≥ 256 bit</td>
<td>≥ 256 bit</td>
</tr>
<tr>
<td># possible keys</td>
<td>$2^{128}$</td>
<td>$2^{192}$</td>
<td>$2^{256}$</td>
<td>$&gt; 2^{256}$</td>
<td>$&gt; 2^{256}$</td>
</tr>
<tr>
<td>Security level</td>
<td>near term</td>
<td>near term</td>
<td>long term</td>
<td>long term</td>
<td>long term</td>
</tr>
</tbody>
</table>


