ATTACK TO BLOCK CIPHERS BY CLASS ELIMINATION USING THE GENETIC ALGORITHM

Osmani Tito-Corrioso*, Mijail Borges-Quintana**†, Miguel A. Borges-Trenard***
*Universidad de Matanzas, Autopista a Varadero km 3.5, Matanzas 40100, Cuba.
**Universidad de Oriente, Av. Patricio Lumumba s/n, Santiago de Cuba 90500, Cuba.
***Universidad Antonio Nariño, Bogotá 111321, Colombia.

ABSTRACT
In recent years, the use of Genetic Algorithms in symmetric cryptography has increased, in particular in the cryptanalysis of block ciphers. However, it is still necessary to continue the study and search for better characteristics. In this sense, two methodologies of partition of the key space that connect the Genetic Algorithms with the attack to block ciphers have been proposed in previous work. In this paper, we propose a methodology for attacking, where it is possible to discard some classes, reducing the total number that would be necessary to go over. In connection with this approach, we obtain necessary conditions and formulas to estimate the parameters's value that determines a balance between the number of classes into which the key space is divided and the number of elements of each one of them, solving the problem of optimal selection of the parameters.

KEYWORDS: Key Space Partition, Quotient Group of Keys, Cryptanalysis, Reference Identity
MSC: 90C59, 94A60.

RESUMEN
En los últimos años, el empleo del Algoritmo Genético en la criptografía simétrica ha aumentado, en particular en el criptoanálisis a cifrados en bloque. Sin embargo, aun es necesario continuar el estudio y la búsqueda de mejores características. En este sentido, dos metodologías de partición del espacio de las claves han sido propuestas en trabajos previos las cuales conectan el Algoritmo Genético con el ataque a cifrados a bloque. En este artículo, se propone una metodología de ataque, donde es posible desechar algunas clases, reduciendo el número total que sería necesario recorrer. En conexión con esta propuesta, se obtienen condiciones necesarias y fórmulas para estimar los valores de los parámetros que determinan un balance entre el número de clases en que se divide el espacio de las claves y el número de elementos en cada una de ellas, resolviendo el problema de la elección óptima de los parámetros.

PALABRAS CLAVE: Partición del Espacio de las Claves, Grupo Cociente de las Claves, Criptoanálisis, Cifrado en Bloque, Identidad de Referencia

†Corresponding author email: mijail@uo.edu.cu

1. INTRODUCTION

There are several methods and tools that are used as optimization and prediction methods. Several heuristic algorithms have been used in the context of Cryptography; in [8], the ACO (*Ant Colony Optimization*) heuristic method is used, and a methodology was tested with the S-AES (*Simplified Advanced Encryption Standard*) block cipher, studying two pairs of plain and encrypted texts. In [2], a combination of Genetic Algorithm (GA) and ACO is used for cryptanalysis of stream ciphers. In [5, 14, 19], the combination and design possibilities of these algorithms were shown, analyzed using the machine learning and deep learning tools. In [13] the authors design and implement a symmetric encryption algorithm whose internal structure is based on the Taboo Search Algorithm.

The Genetic Algorithm is an optimization method used in recent years in Cryptography for various purposes, particularly in carrying out attacks and modeling various types of encryption. Some of the research conducted in this direction is mentioned below.

In [15] cryptanalysis using GA is applied to obtain letters from plaintexts using affine ciphers. In [12] the GA is used to attack encrypted images using chaotic maps (functions). These functions derive from Chaos Theory and are based on nonlinear dynamical systems. In the proposed method, the image encrypted with chaotic maps is used as a ciphertext, and the GA is used to decrypt the image. In [9] a hybrid tool is developed that creates ciphertexts from the combination of the GA and the Particle Swarm Algorithm. Shannon's Entropy method was used as a fitness function in both algorithms. The authors claim that the proposed application offers an alternative method of data encryption and decryption that can be used to transmit messages.

In [18] an information security technique based on Elliptic Curve Cryptography and GA to protect E-education system is proposed. In [1] a technique for the encryption of texts, based on the mutation and crossing operations of the GA, is presented. The proposed encryption technique divides the plaintext characters into parts and applies the crossover operation between them, followed by the mutation operation to obtain the ciphertext. In [4] the authors discuss the comparison of traditional cryptographic algorithms and GA-based cryptosystems. In [3] author discusses about comparison of traditional cryptographic algorithms and genetic algorithm based cryptography methods. More details on the use of GA in Cryptography can be seen, for example, in [22], [7], [10], [16] and [17].

Block cipher cryptanalysis is a complex problem with no certain route, no magic or precise recipes that can solve the problem. Block ciphers such as AES, currently used in standard security protocols, continue to guarantee their security due to their good properties. Hence, the different attempts to build attack methodologies constitute new knowledge and tools in order to continue developing this branch of Cryptography, which also contributes to evaluating the security of block ciphers and their possible vulnerabilities.

As in all evolutionary algorithms, it is always a difficulty in GA that as the number of individuals in the space of admissible solutions grows, in this case, the set of keys, it is necessary to carry out a greater number of generations to obtain best results. In this sense, in the key space partition methodologies proposed in [6] and [21], the problem of choosing the parameter that determines the number of classes to traverse and the number of elements within each one: the more classes, the fewer elements to traverse in each one; and fewer classes implies that each one will have more elements and therefore, the GA will take longer to go through it. It is clear that the greater the number of generations, the algorithm consumes more time, so it is interesting to be able to estimate the time that may be necessary to execute a certain desired number of generations.

In [6] and [21] authors analyze different parameters of the GA separately. In this sense, the contribution of the present paper is the generalization of these results, obtaining formulas that combine all the parameters at the same time. Additionally, we propose and describe an attack methodology that, mainly, allows us to reduce the number of classes where the key will be searched.

In the present work, an attack methodology is proposed where some classes are discarded, reducing the total number that would be necessary to go through: the Class Elimination Attack (CEA). Necessary conditions and formulas are obtained to calculate the parameter's value that determines the balance between the number of classes and the number of elements in each of them, solving the problem of its optimal choice. At the same time, these formulas can be used to estimate several parameters assuming other known ones.

This paper is structured as follows. In Section 2. the results on the Probabilistic Membership Problem are presented. Section 3. focuses on the fundamental contributions of the work: the Direct Attack (Section 3.1.), the Class Elimination Attack (Section 3.2.), and the results on the choice of various parameters involved in the GA and the key space partition methodologies (Section 3.3.).

2. PROBLEM OF MEMBERSHIP OF KEYS TO EQUIVALENCE CLASSES

We use two key space partition methodologies proposed, the BBM and the TBB ([6], [21]). This methodologies allow the GA to work on a certain subset of the set of admissible solutions as if it were the full set. The partition into equivalence classes allows it to use this algorithm in parallel, in several classes simultaneously and independently.

Let $\mathbb{F}_2^{k_1}$ be the key space of length $k_1 \in \mathbb{Z}_{>0}$, $k_2, k_d \in \mathbb{Z}_{>0}$, such that, $1 \leq k_2 < k_1$, $k_d = k_1 - k_2$, and $Q = \{0, 1, 2, \ldots, 2^{k_d} - 1\}$. Then, in both methodologies the formulas to represent the elements of $\mathbb{F}_2^{k_1}$ are identical:

$$q\, 2^{k_2} + r,\; q \in Q,\; r \in \mathbb{Z}_{>0}. \tag{2.1}$$

This equation can be used to summarize the differences between the methodologies. Both consist of keeping the GA running on a subset of the key space rather than the entire key space. In the case of BBM, the subset is associated with the class of the keys that correspond to the same quotient (q). The TBB methodology consists of working with the subset given by the keys with the same remainder (r), the elements of each class are scattered throughout the set of keys. The parameters q, r, k_2 and k_d have dual role in both methodologies. To avoid ambiguities in the notation, from now on, we will refer to the parameters of the BBM methodology as q, r, k_2 and k_d. While $\hat{q}$, $\hat{r}$, $\hat{k}_2$ and $\hat{k}_d$ will be to refer to the same parameters in the TBB methodology. For more details about this methodologies, see Section 2.2 of [21].

Let M be a plaintext, K a key, and C be the ciphertext of M with K (regardless of the encryption used). Let J be the collection of the q intervals into which $[0, 2^{k_1} - 1]$ is divided by the BBM methodology, taking the intervals as equivalence classes. In the classes of J, two elements are related if they have the same quotient when divided by 2^{k_2}. Given $A, B \in [0, 2^{k_1} - 1]$, the fact that A and B

leave the same quotient is equivalent to,

$$\frac{A - A\,(\text{mod } 2^{k_2})}{2^{k_2}} = \frac{B - B\,(\text{mod } 2^{k_2})}{2^{k_2}}. \tag{2.2}$$

Let G_K be the quotient group of keys in TBB methodology. Let $\zeta(n) \in G_K \vee \zeta(n) \in J$ be the equivalence class of $n \in \mathbb{F}_2^{k_1}$ in G_K or J. Let $P_{\zeta(n)}(m)$ be the probability that $m \in \mathbb{F}_2^{k_1}$ belongs to $\zeta(n)$ (note that $m \in \zeta(n) \Leftrightarrow \zeta(m) = \zeta(n)$). Then, the Probabilistic Membership Problem (PMP) is: Given M and C (one or several pairs), with $C \in \mathbb{F}_2^h$, $h \in \mathbb{Z}_{>0}$, such that, $|\mathbb{F}_2^h| \leq |\mathbb{F}_2^{k_1}|$. To calculate $P_{\zeta(C)}(K)$.
Regarding PMP, the following results are obtained in both key space partition methodologies (see [20]).
Note that the fact that C and K belong to the same class is equivalent to their leaving the same remainder r modulo $2^{\hat{k}_2}$, by the definition of G_K. The parameter $\hat{r}$ has block length $\hat{k}_2$; and $\hat{q}$ has length $\hat{k}_d$, which are equivalent to the remaining components to reach the total k_1. By (2.1), the binary components of $\hat{r}$ correspond to the last $\hat{k}_2$ of the complete binary block of K or C. Therefore, taking into account the binary representation of C and K, $\zeta(C) = \zeta(K)$ is equivalent to the last $\hat{k}_2$ components of C and K are the same. This proves the following theorem,

Theorem 2..1 (Equivalence of classes in TBB) *Given $k_1, \hat{k}_2, h \in \mathbb{Z}_{>0}$, $C \in \mathbb{F}_2^h$ and $K \in \mathbb{F}_2^{k_1}$, such that, $|\mathbb{F}_2^h| \leq |\mathbb{F}_2^{k_1}|$. The following three statements are equivalent in G_K:*
a) $\zeta(C) = \zeta(K)$.
b) $C \equiv K \equiv r\,(mod\ 2^{\hat{k}_2})$.
c) The last $\hat{k}_2$ components of C and K are the same.

From Theorem 2..1 we have the following corollary,

Corollary 2..1 (Membership probability in TBB) *Given C and $\hat{k}_2$, the probability that K belongs to the same class of C is equal to $\frac{1}{2^{\hat{k}_2}}$. That is,*

$$P_{\zeta(C)}(K) = \frac{1}{2^{\hat{k}_2}}. \tag{2.3}$$

Proof. Since $K \in \zeta(C) \Leftrightarrow \zeta(C) = \zeta(K)$ in G_K, then,

$$P_{\zeta(C)}(K) \Leftrightarrow P(\zeta(C) = \zeta(K)). \tag{2.4}$$

Applying the 2..1 Theorem and the implication a) $\Rightarrow$ c) on the right side of (2.4), we have that $P(\zeta(C) = \zeta(K))$ is equal to the probability that the last $\hat{k}_2$ components of C and K are equal. Therefore, to calculate $P_{\zeta(C)}(K)$ it is necessary and sufficient to calculate the probability that the last $\hat{k}_2$ components of K are equal to those of C. Since they are $\hat{k}_2$ components and each one has two possibilities, 0 or 1, then there are $2^{\hat{k}_2}$ possible combinations for K, but only one of them matches C, so applying the classical definition of probability we arrive at (2.3). ■
Similar results are obtained in the BBM methodology. Note that the fact that C and K belong to the same class is equivalent to leaving the same quotient q when dividing by 2^{k_2}. The parameter r

has a block length of k_2 and q has a length of k_d, which is equivalent to the remaining components to arrive at the total k_1. The binary components of r correspond to the last k_2 of the complete binary block of K or C, which implies that the binary components of q correspond to the remaining first k_d. Therefore, $\zeta(C) = \zeta(K)$ is equivalent to the first k_2 components of C and K being equal. This proves the following theorem,

Theorem 2..2 (Equivalence of classes in BBM) *Given $k_1, k_2, h \in \mathbb{Z}_{>0}$, $C \in [0, 2^h - 1]$ and $K \in [0, 2^{k_1} - 1]$ such that, $h \leq k_1$. The following three statements are equivalent in J:*
a) $\zeta(C) = \zeta(K)$.
b) $\frac{C - C\,(mod\ 2^{k_2})}{2^{k_2}} = \frac{K - K\,(mod\ 2^{k_2})}{2^{k_2}}$.
c) The first k_d components of C and K are the same.

Corollary 2..2 (Membership probability in BBM) *Given C and k_d, the probability that K belongs to the same class of C is equal to $\frac{1}{2^{k_d}}$. That is,*

$$P_{\zeta(C)}(K) = \frac{1}{2^{k_d}}. \tag{2.5}$$

The proof of Corollary 2..2 is similar to what was done in Corollary 2..1. Note that at point *c)*, of Theorems 2..1 and 2..2, C and K are referred to as binary blocks with the most significant bit on the left. The application of Corollaries 2..1 and 2..2 is useful when in practice you have more than one ciphertext. In this case, the most interesting result, as a consequence of the above, is the following. Suppose we have w ciphertexts, then,

$$n_w = wP_{\zeta(C)}(K) = \frac{w}{2^{\hat{k}_2}}, \tag{2.6}$$

and,

$$n_w = wP_{\zeta(C)}(K) = \frac{w}{2^{k_d}}, \tag{2.7}$$

where n_w is the number of ciphertexts of the w initials, according to the probability of membership in Corollaries 2..1 and 2..2, to whose class K belongs for a previously fixed value $\hat{k}_2$. This implies, in particular, that for an attack, it is not necessary to search in all the classes of G_K (or intervals of J) for each one of the ciphertexts, but that choosing a good value for $\hat{k}_2$, just look at the same class as the ciphertext. Since of the w, according to (2.6), it is probable that in at least n_w texts, the key is found in said n_w corresponding classes (one class for each ciphertext that complies with (2.6), in case the keys are different; it would be the same class for the n_w ciphertexts in case it is the same key for all). For this it is necessary that at least $n_w \geq 1$, which implies that $w \geq 2^{\hat{k}_2}$.
The results on the PMP give a partial solution to the problem of choosing a value for k_2. Limiting it to the possibilities of the values that n_w can take and to the knowledge of the last or first bits of the key.

3. CLASS ELIMINATION ATTACK

In this section, we will work on the experiments with the TBB methodology and the HTC (*Heys Toy Cipher*) cipher. The HTC is a Permutation Substitution Network type block cipher (see [11]).

It works with 16-bit blocks for the plaintexts, ciphertexts and the key (the same in each round). It consists of 4 rounds. Each round consists of an XOR addition with the key, a substitution, and a permutation of the bit positions. On the other hand, see [21] for more details about the values of the parameters and operators of the Genetic Algorithm that we use for the experiments.

We focus on a way to choose $\hat{k}_2$ using the results of the PMP, the Direct Attack, and we end with a proposal of a Class Elimination Attack methodology that allows us to restrict, with a certain probability, the number of classes to go through and to improve the Direct Attack. Remember that an attack model is being applied through the GA to known plaintext.

3.1. DIRECT ATTACK

Suppose we have w pairs of plaintexts and their corresponding ciphertexts, and it is the only information we have to search for the key, of which everything is unknown. The first thing is to choose $\hat{k}_2$, which since the attack is in G_K, then represents the number of classes in which the quotient group will be divided. We know, according to the PMP, that there is a range to select $\hat{k}_2$, fulfilling that $n_w > 1$, according to (2.6) and depending on the computation capacity and time consumed. After choosing $\hat{k}_2$ and calculating G_K, a class must be chosen to search for the key, an issue resolved according to PMP, with which the key will be searched for in the same class as which the corresponding ciphertext belongs. Then, it would only be necessary to go through each plaintext and ciphertext pair and look for the key in the same class of the ciphertext.

For the experiments, a PC with an Inter(R) Core(TM) i5-3340 CPU @3.10GHz (4 CPUs) and 4GB of RAM was used. We start from $w = 100$ pairs of texts, therefore, a value of $\hat{k}_2$ between 1 and 6 (of the 16 possible) can be chosen, since if $\hat{k}_2 = 7$, then, $n_w = 100/2^7 = 100/128 < 1$, with which it would be unlikely to find a ciphertext whose class contains the key. But of the values from 1 to 6, the latter is in a similar situation, since $n_w = 100/2^6 = 100/64 \approx 1.6$. Instead, there seem to be better results for values from 1 to 5. This was tested for values of $\hat{k}_2$ from 2 to 10, performing the attack searching through each of the w pairs until finding the key, what we have called Direct Attack. At each value of $\hat{k}_2$ were done ten tests, generating the 100 pairs each time. The results can be seen in Table 1.

$\hat{k}_2$	No. of classes	Time	Generations	Faults (%)
2	6.3	304.975	68.1	0
3	11.2	257.25	45.5	0
4	19	130.158	9.1	0
5	62.8	217.516	12.7	30
6	67.5	111.559	8.3	60
7	56.9	44.559	3.1	30
8	86.9	28.586	2.2	60
9	86.7	17.253	1.7	70
10	101	10,116	2	100

Table 1: HTC Direct Attack with $w = 100$

The second column is the number of classes necessary to go through on average to find the key in each

test. The third and fourth columns show the time in seconds and the number of generations that the GA took on average to find the key in each class. And the fifth column is the percent of the number of trials where the key was not found. In table, one can see how the time decreases as $\hat{k}_2$ increases, which is clear because, with the increase of $\hat{k}_2$, the number of classes also increases. Therefore, it decreases the number of elements inside them, making the GA finish faster. It is interesting to note that failures start to appear from $\hat{k}_2 = 5$ and gradually increase as $\hat{k}_2$ increases. In other words, although there is a range to choose from 1 to 16, the best values are up to 4 and a maximum of 5 concerning the 100 ciphertexts that are being used. This point corroborates the idea of a solution to the problem of choosing $\hat{k}_2$.
One drawback of doing the attack this way is that more classes are typically traversed than the partition has. For example, despite we could choose one of the best choice $\hat{k}_2 = 4$, it is expected on average that it traverses 19 classes (pairs of plain and encrypted texts) until it finds the key, according to Table 1. But G_K would only have $2^{\hat{k}_2} = 2^4 = 16$ classes, and more are being traversed. What happens is, on the one hand, that there are several ciphertexts that belong to the same class where the key is not so that the GA will go through that class several times; and on the other hand, that the GA, with the aptitude function that is being used many times, is not able to find the key by looking in the correct class, so it keeps looking for the next ciphertext until it finds it, and some classes are repeated. Another point of view is to check if the way of traversing the ciphertexts can be improved. The latter will be seen in the next section.

3.2. CLASS ELIMINATION ATTACK

Suppose given w ciphertexts and $\hat{k}_2$, such that $n_w \geq 1$, that is, $w \geq 2^{\hat{k}_2}$. There are $2^{\hat{k}_2}$ equivalence classes, and the w texts can be grouped into those classes. To know the probability for w such that, w_1 belong to the class X_1, w_2 to the X_2, $\cdots$, $w_{2^{\hat{k}_2}}$ to the $X_{2^{\hat{k}_2}}$, the Multinomial Distribution can be used,

$$P(X_1 = w_1, \cdots, X_{\hat{k}_2} = w_{\hat{k}_2}) = \frac{w!}{w_1! w_2! \cdots w_{2^{\hat{k}_2}}!} p_1^{w_1} \cdots p_{\hat{k}_2}^{w_{\hat{k}_2}}, \tag{3.1}$$

where $\sum_{i=1}^{2^{\hat{k}_2}} w_i = w$.
All classes have equal probability $p = \frac{1}{2^{\hat{k}_2}}$ of being chosen, therefore, (3.1) reduces to,

$$P(X_1 = w_1, \cdots, X_{\hat{k}_2} = w_{\hat{k}_2}) = \frac{w!}{w_1! w_2! \cdots w_{2^{\hat{k}_2}}!} p^w. \tag{3.2}$$

The expected value of X is,

$$\mathrm{E}(X) = wp = w\frac{1}{2^{\hat{k}_2}}, \tag{3.3}$$

which is equivalent to n_w. This implies that most likely, by organizing the w ciphertexts into the $2^{\hat{k}_2}$ classes, the classes are expected to have, on average n_w elements, and around this number be the highest concentration. The probability that there are many more elements in each class, or very few, decreases as one moves away from the mean n_w. If so, the key should appear more often if looking at classes that have elements close to n_w, after the w ciphertexts are grouped into classes. This fact

does not means a weakness in the ciphers but is favored by their stochastic structure, and in any case, the more random encryption is attempted, the better it will approach (3.3).
To check this, $\hat{k}_2 = 4$ was taken, and 19 200 ciphertexts were generated, grouped into 200 tests of 96 texts each. The idea is, in each test, to group the 96 texts into the $2^4 = 16$ classes and check how many elements the class to which the key belongs has. A Laptop PC was used with a processor: Intel(R) Celeron(R) CPU N3050 @1.60GHz (2 CPUs), ~1.6GHz and 4GB of RAM. In these 200 tests, the results are shown in Table 2. The first and third columns indicate the number of elements of the

Num. Elem.	Num. Appearance	Num. Elem.	Num. Appearance
0	2	9	16
1	3	10	5
2	8	11	4
3	21	12	5
4	25	13	0
5	32	14	0
6	25	15	0
7	31	16+	0
8	23	–	–

Table 2: Times the key appears around n_w

class, and the second and fourth the number of times the key was found. As can be seen, the largest number of keys is concentrated around 6 (since $n_w = 96/16 = 6$). For example, in the range of classes with 3 to 9 elements, there are 173 keys, 86.5% of the 200 total.
In this sense, the idea of the Class Elimination Attack (CEA) is to group the w ciphertexts into equivalence classes, then choose an interval,

$$[a, b] \subset \mathbb{N}, \tag{3.4}$$

where a and b represent the amount of ciphertexts in the classes, after being grouped, in a neighborhood of n_w, with $a \leq n_w \leq b$. Such that the length of the interval is less than the number of classes at least once,

$$\eta(b - a + 1) < 2^{\hat{k}_2}, \eta \in \mathbb{N}, \eta \neq 0. \tag{3.5}$$

With (3.5) an *elimination condition* is guaranteed, since it is giving the possibility of traversing η times the $b-a+1$ classes contained in the interval $[a, b]$ as *ideal case*, without even reaching the total $2^{\hat{k}_2}$ in the worst case, and it is the starting point. Note that in the interval $[a, b]$ are the classes with the highest probability of containing the key, hence the idea of traversing those classes η times. Let $\mathcal{G}$ be the set containing the classes of the ciphertexts after grouping, and,

$$C_{[a,b]} = \{\zeta \in \mathcal{G} | \#\zeta \in [a, b]\}, \tag{3.6}$$

we refer to *ideal case* when,

$$\#C_{[a,b]} = b - a + 1. \tag{3.7}$$

It's clear that,

$$1 \leq \#C_{[a,b]} \leq 2^{\hat{k}_2}, \tag{3.8}$$

and the fact that $\#C_{[a,b]}$ approaches 1, or $2^{\hat{k}_2}$, are unwanted cases, the former because it restricts too much the number of classes selected for traverse, being more likely that the key is not found in these. The problem of the second case is that almost all the classes would be selected, being necessary to traverse practically the entire space of the keys. That (3.7) holds is what is desired and the assumption on which the CEA is based.

After choosing the interval, the main idea is to look for the key in the $\#C_{[a,b]}$ classes that are in $C_{[a,b]}$. This search is repeated η times in the interval as long as the key is not found. The search in this interval is done prioritizing from the center n_w inside towards the ends. The stages of the CEA are summarized in the Algorithm 1:

Algorithm 1 Class Elimination Attack

```
Input: $w$ plaintext and ciphertext pairs.
Output: The individual with the highest fitness function as the best solution.
1: Choose $\hat{k}_2$ and calculate $n_w$.
2: Group the $w$ pairs into equivalence classes.
3: Choose $\eta$ and $[a,b]$ satisfying (3.5).
4: while the key is not found or the $\eta$ iterations are not reached do
5:     Find the key with the GA in the classes of $C_{[a,b]}$.
6: end while
```

In the following Definition 3..1 a methodology of how to choose the interval $[a,b]$ is proposed. In the same way, other criteria could be taken for the choice of a and b.

Definition 3..1 *Given $\hat{k}_2, k_d, n_w, k_1, \eta \in \mathbb{N}$, $\hat{k}_2, k_d \leq k_1$; $\mathcal{P} \in \mathbb{R}_+^*$, such that, $0 < \mathcal{P} \leq 100$. Let's suppose that in the CEA we want to traverse the $\mathcal{P}$ percent of the total classes as an ideal case. Then, the extremes a and b, and therefore the interval $[a,b] \subset \mathbb{N}$, are defined as follows,*

$$[a,b] = [\lfloor n_w - r \rfloor, \lceil n_w + r \rceil], \tag{3.9}$$

where,

$$r = \frac{\frac{\mathcal{P}2^{\hat{k}_2}}{100\eta} - 1}{2}, \tag{3.10}$$

or,

$$r = \frac{\frac{\mathcal{P}2^{k_d}}{100\eta} - 1}{2}, \tag{3.11}$$

and, if $a < 0$, then $a = 0$, since $a \in \mathbb{N}$.

Experiments were performed applying the CEA to the HTC on the same PC Laptop for the results of the Table 2 and the data that has been treated: $w = 100$, $\hat{k}_2 = 4$, $2^{\hat{k}_2} = 16$ classes, $\eta = 2$ and the interval $[3,9]$ was chosen, such that $2 * (9 - 3 + 1) = 14 < 2^4 = 16$. The choice of the interval, and therefore, of a and b, was made assuming that we wanted to cover a certain percentage of the total classes as an ideal case: 85% of the total classes $2^{\hat{k}_2}$ =16, which is 13.6; dividing by $\eta = 2$ we have an approximate value for $b - a + 1 = 6.8$, from which we have $b - a = 5.8$, but $b - a$ is the length of the interval $[a, b]$, in the center of which should be $n_w = 6$, and taking as radius $r = \frac{b-a}{2} = 2.9$, then, we have,

$$a = \lfloor n_w - r \rfloor = 3, \tag{3.12}$$

and,

$$b = \lceil n_w + r \rceil = 9. \tag{3.13}$$

The fitness function F was used. For which 40 runs were made, giving a total of 4000 pairs of plain and encrypted texts. The results are as follows: the key was found on average in 60 % of the pairs, in approximately 114.26 seconds and 18.999 generations, going through 8.83 classes of the 16 total. Notice how the number of classes needed to find the key is reduced on average. Only half of the classes needed to be traversed on average, 8.83 out of 16. Some classes had to be traversed more than once, since,

$$b - a + 1 = 9 - 3 + 1 = 7.$$

3.3. ABOUT THE CEA AND THE CHOICE OF PARAMETERS

It is know that the estimated time, t_e, that the GA should take in n_g generations, is:

$$t_e = t_m n_g. \tag{3.14}$$

where t_m is the average time obtained in experiments for a generation (see [21]). At this point, regarding the estimation of time, $\hat{k}_2$ ($\hat{k}_d$) and the CEA with the TBB methodology, we have the following definition:

Definition 3..2 (Reference Identity in TBB) *Let $k_1, \hat{k}_2$, $\hat{k}_d \in \mathbb{N}^*$ be given, such that, $\hat{k}_2 \le k_1$, and, $\hat{k}_d = k_1 - \hat{k}_2$; $\eta, a, b \in \mathbb{N}$, such that the elimination condition* (3.5) *is satisfied; $t_m, t_e \in \mathbb{R}_+^*$ the average time that the GA takes to perform a generation (iteration) and the estimated time for a certain number of generations, respectively; and, m the number of individuals in the population in the GA. So, the Reference Identity (RI) in the TBB methodology is defined as,*

$$\frac{t_m \eta (b - a + 1) 2^{\hat{k}_d}}{m t_e} \overset{\text{def}}{=} 1. \tag{3.15}$$

In an equivalent way, the RI is defined for the BBM methodology, it is only necessary to take into account the dual function of the parameters in both partitions and change $\hat{k}_d$ by k_2:

Definition 3..3 (Reference Identity in BBM) *Let $k_1, k_2 \in \mathbb{N}^*$ be given, such that, $k_2 \le k_1$; $\eta, a, b \in \mathbb{N}$, such that the elimination condition* (3.5) *is satisfied; $t_m, t_e \in \mathbb{R}_+^*$ the average time that the GA takes to perform a generation (iteration) and the estimated time for a certain number of generations, respectively; and, m the number of individuals in the population in the GA. So, the Reference Identity in the BBM methodology is defined as,*

$$\frac{t_m \eta (b - a + 1) 2^{k_2}}{m t_e} \overset{\text{def}}{=} 1. \tag{3.16}$$

The relation (3.15) can be used to estimate the different parameters depending on the desired purpose and computational capacity. In particular, an approximate value for $\hat{k}_d$ (and thus for $\hat{k}_2$) can be calculated. Clearing $\hat{k}_d$ from (3.15), and taking into account that it must be an integer value, we have,

$$\hat{k}_d \approx \left\lfloor \log_2 \frac{mt_e}{t_m \eta (b-a+1)} \right\rfloor, \tag{3.17}$$

where t_e would be chosen based on a desired time. The relation (3.15) can be obtained from (3.14). The GA has two stopping conditions, if it finds the key, or otherwise, if it performs a certain number of generations, which depends on the number of elements of the class to traverse, $2^{\hat{k}_d}$, and the number of individuals in the population, m, being equal to,

$$\frac{2^{\hat{k}_d}}{m}, \tag{3.18}$$

but on average, in the CEA it is expected to traverse $\eta(b-a+1)$ classes, so the total number of generations to traverse would be,

$$n_g = \frac{\eta(b-a+1)2^{\hat{k}_d}}{m}, \tag{3.19}$$

Now assuming that t_m is the average time that the GA takes to go through one generation, then, substituting (3.14), we have that the estimated total time that it will take to go through all the classes is,

$$t_e = t_m n_g = t_m \frac{\eta(b-a+1)2^{\hat{k}_d}}{m}, \tag{3.20}$$

from where, we arrive at (3.15).
Note that the RI is useful to calculate an approximate value of t_e, when it is desired to estimate the time that the GA in the CEA will take to go through the $b-a+1$ classes that are expected on average. It is a general formula for when the ciphertexts are unknown, and the calculations are done assuming that you have a certain amount w, from which you get a, b, and $\hat{k}_d$. It is a way of making in advances estimates for decision making. However, if the w pairs of plaintext and ciphertext are known, then $\mathcal{G}$ and $C_{[a,b]}$ are computed. And it is known that in the CEA the $\#C_{[a,b]}$ classes that are in $C_{[a,b]}$ will be traversed. Therefore, in this case, a more precise relation to estimate t_e would be,

$$t_e = t_m \frac{\eta 2^{\hat{k}_d} \#C_{[a,b]}}{m}. \tag{3.21}$$

Note that the equation (3.21) is not a general case of (3.15). The main difference is that with (3.15) the value of several parameters can be estimated assuming other known values; however, (3.21) only serves to estimate the time more precisely, since knowledge of the w ciphertexts fixes the values for a, b and $\hat{k}_d$, in the very fact of calculating $\#C_{[a,b]}$, which in practice, can be different from $b-a+1$.
Finally, we have the following theorem,

Theorem 3..1 *Let w, n_w, m, t_e, t_m, η, a, b and k_1, as previously defined, then, we have the following General Reference Identity (GRI),*

$$\frac{wmt_e}{n_w t_m \eta (b-a+1)2^{k_1}} = 1, \tag{3.22}$$

furthermore, the GRI is independent of the key space partition methodology (BBM or TBB).

Proof. It is known that,

$$k_1 = \hat{k}_2 + \hat{k}_d, \tag{3.23}$$

solving $\hat{k}_2 = \log_2 \frac{w}{n_w}$ from (2.6), $\hat{k}_d$ from (3.15) in RI, and substituting both values in (3.23), we have,

$$k_1 = \log_2 \frac{w}{n_w} + \log_2 \frac{mt_e}{t_m \eta (b-a+1)}, \tag{3.24}$$

i. e.,

$$k_1 = \log_2 \frac{wmt_e}{n_w t_m \eta (b-a+1)}, \tag{3.25}$$

where we arrive at (3.22). In the case of the BBM methodology, it starts from,

$$k_1 = k_2 + k_d, \tag{3.26}$$

where, clearing k_2 from (3.16) in the Definition 3..3, k_d from (2.7) and substituting in (3.26), with the same procedure, (3.22) is arrived at in the same way, reflecting the independence of the space partition methodology. ■

In general, the RI and the GRI provide a way to make estimates of each of the parameters involved in the attack on block ciphers through the GA, depending on the information available. An interesting note in this part is that the problem of choosing k_2 (in both methodologies) is solved. In summary, choosing k_2 is not a uniqueness problem, but rather depends on various factors such as the time available for carry out the attack, the calculation capacity, the information that is known about the encrypted texts, etc. In other words, the optimal value of k_2 is chosen based on what is known and what is wanted to be done. These results make it possible to have better criteria for making decisions in advances depending on the calculation capacity and the time available for the attack.

3.4. TIME ESTIMATION IN THE HTC CIPHER USING THE CEA

This section presents an example of how to use RI to estimate the time, t_e, in HTC encryption, assuming different values of the number of ciphertexts, w. The goal is to get an idea of how long it will take for the GA to perform the attack using the CEA.

We will assume the parameters $\eta = 2$ to be fixed, the percentage of the total number of keys to be traversed is $\mathcal{P} = 85$, with experiments carried out on HTC encryption and using the function F the average time in a generation in minutes it is $t_m = 0.00582$ approximately. Two values will be taken for n_w, 6 and 20.

With the above data, $\hat{k}_2$ will be calculated from (2.6), taking into account that it must be an integer value, therefore,

$$\hat{k}_2 \approx \left\lfloor \log_2 \frac{w}{n_w} \right\rfloor. \tag{3.27}$$

We will work on the TBB methodology, so in the tables, $2^{\hat{k}_2}$ represents the total number of equivalence classes for that value of $\hat{k}_2$. Similarly, a and b are obtained from the definition 3..1. The results are presented in Tables 3 and 4. The times, t_e, of the tables are in seconds. Note that the values of $\hat{k}_2$, a and b from Table 4 for $w = 100$, coincide with those obtained in Section 3.2.. On the other hand, it seems that $t_e = 200.25$ does not offer a good estimation of the time. However, in Tables 3 and 4 t_e

w	$\hat{k}_2$	a	b	t_e
100	2	19	21	343.28
1000	5	13	27	214.55
10000	8	0	74	134.09
100000	12	0	890	99.56

Table 3: Calculation of t_e with $n_w = 20$

w	$\hat{k}_2$	a	b	t_e
100	4	3	9	200.25
1000	7	0	33	121.58
10000	10	0	224	100.57
100000	14	0	3488	97.47

Table 4: Calculation of t_e with $n_w = 6$

is estimated assuming that the $\eta(b-a+1)$ classes contained in the interval will be traversed $[a,b]$ as an ideal case. While in the results of Section 3.2., an average of 8.83 classes of the 14 expected were covered. The interesting thing is that if $\eta(b-a+1)$ is replaced by 8.83 in the Reference Identity, the estimated time is $t_e = 126.298$, which is a better approximation to the actual time 114.26.
In both tables, the time decreases as the number of known ciphertexts increases, even though the number of classes and the size of the interval $[a,b]$ increases. Instead, $\hat{k}_2$ is increasing as well, and therefore each class will have fewer elements to traverse. This corroborates the fact that the CEA will be more successful as more ciphertexts are known.

4. CONCLUSIONS

A class elimination methodology was designed to attack block ciphers using the GA in the present work. An attack methodology is proposed where some classes are discarded, reducing the total number that would be necessary to go through the Class Elimination Attack (CEA). Regarding the GA and the CEA algorithm, it is defined: a way to choose the interval $[a,b]$; the Reference Identity (RI) for both space partition methodologies and a necessary condition where the General Reference Identity (GRI) is obtained. In general, the RI and the GRI provide a way to make estimates of each of the parameters involved in the attack based on the information available. An interesting note is that the problem of choosing the parameter that determines the balance between the number of classes in which the space is divided, and the number of elements within each one of them is solved. Consequently, choosing this parameter is not a problem of uniqueness, but rather, it depends on several factors such as the time available to carry out the attack, the calculation capacity, the information that is known about the encrypted texts, etc. These results make it possible to have better criteria for making decisions in advance. Future research will work on applying the results obtained to other families of block ciphers.

ACKNOWLEDGMENTS: The research associated with the results presented in this publication received funds from the International Funds and Projects Management Office under the code PN223LH006-015.

RECEIVED: AUGUST, 2022.
REVISED: FEBRUARY, 2023.

REFERENCES

[1] ABDULJABBAR, R., HAMID, O., AND ALHYANI, N. (2021): Features of genetic algorithm for plain text encryption **International Journal of Electrical and Computer Engineering**, 11(1):434–441.

[2] ALI, F. H. AND JAWAD, R. N. (2020): Using evolving algorithms to cryptanalysis nonlinear cryptosystems **Baghdad Science Journal**, 17(2):682–688.

[3] BAGANE, P. AND KOTRAPPA, D. S. (2021a): Comparison Between Traditional Cryptographic Methods and Genetic Algorithm Based Method Towards Cyber Security **International Journal of Advanced Research in Engineering and Technology**, 12(2):676–682.

[4] BAGANE, P. AND KOTRAPPA, S. (2021b): Comparison Between Traditional Cryptographic Methods and Genetic Algorithm Based Method Towards Cyber Security **International Journal of Advanced Research in Engineering and Technology (IJARET)**, 12(2):676–682.

[5] BLACKLEDGE, J. AND MOSOLA, N. (2020): Applications of artificial intelligence to cryptography **Technological University Dublin**, 8(3):21–60.

[6] BORGES-TRENARD, M., BORGES-QUINTANA, M., AND MONIER-COLUMBIÉ, L. (2019): An application of genetic algorithm to cryptanalysis of block ciphers by partitioning the key space **J. Discret. Math. Sci. Cryptogr.**, 25(2):325–334.

[7] DIN, M., PAL, S. K., MUTTOO, S. K., AND MADAN, S. (2020): A Hybrid Computational Intelligence-based Technique for Automatic Cryptanalysis of Playfair Ciphers **Defence Science Journal**, 70(6):612–618.

[8] GRARI, H., AZOUAOUI, A., AND ZINE-DINE, K. (2019): A cryptanalytic attack of simplified-AES using ant colony optimization **International Journal of Electrical and Computer Engineering**, 9(5):4287–4295.

[9] GÜRFIDAN, R. AND ERSOY, M. (2020): A new hybrid encryption approach for secure communication: GenComPass **International Journal Computer Network and Information Security**, 12(4):1–10.

[10] HEGDE, A. P. AND SUDHA, D. K. L. (2021): Image Cryptography using Chaos Map, Genetic Operator and AES Algorithm **International Research Journal of Modernization in Engineering, Technology and Science**, 3(9):957–964.

[11] HEYS, H. M. (2002): A tutorial on linear and differential cryptanalysis **Cryptologia**, 26(3):189–221.

[12] JAYARAJ, N. AND KUMAR, K. (2018): Cryptanalysis on Chaotic Mapping using Genetic Algorithm **International Journal of Pure and Applied Mathematics**, 119(7):1077–1083.

[13] KADDOURI, Z. AND OMARY, F. (2017): Application of the tabu search algorithm to cryptography **International Journal of Advanced Computer Science and Applications**, 8(7):82–87.

[14] LEE, T. R., TEH, J. S., YAN, J. L. S., JAMIL, N., AND YEOH, W.-Z. (2020): A machine learning approach to predicting block cipher security In **Proceedings of the 7th International Cryptology and Information Security Conference 2020, CRYPTOLOGY 2020**, pages 122–132.

[15] MEZAAL, Y. S. AND ABDULKAREEM, S. F. (2017): Affine cipher cryptanalysis using genetic algorithms **JP Journal of Algebra, Number Theory and Applications**, 39(5):785–802.

[16] QOBBI, Y., JARJAR, A., ESSAID, M., AND BENAZZI, A. (2022): Image Encryption Algorithm based on Genetic Crossover and Chaotic DNA Encoding **Soft Computing**, 26(12):5823–5832.

[17] SABONCHI, A. AND AKAY, B. (2021): A survey on the Metaheuristics for Cryptanalysis of Substitution and Transposition Ciphers **Computer Systems Science & Engineering**, 39(1):87–106.

[18] SAMANTA, P., KARFORMA, S., AND BHOWMIK, A. (2021): ECC with Soft Computing: Design of a Cryptosystem for Data Security in E-Learning System **Vidyabharati International Interdisciplinary Research Journal**, 13(1):667–682.

[19] SO, J. (2020): Deep Learning-Based Cryptanalysis of Lightweight Block Ciphers **Security and Communication Networks**, 2020:1–11.

[20] TITO-CORRIOSO, O. (2021): On Cryptanalysis to Block Ciphers using the Genetic Algorithm **Master's Thesis in Mathematical Sciences. Univ. of Havana, Cuba (In Spanish)**.

[21] TITO-CORRIOSO, O., BORGES-TRENARD, M., BORGES-QUINTANA, M., ROJAS, O., AND SOSA-GÓMEZ, G. (2021): Study of Parameters in the Genetic Algorithm for the Attack on Block Ciphers **Symmetry**, 13(806):1–12.

[22] TIWARI, M., PINHEIRO, D., SHUKLA, S., POPTANI, S., AND NATARAJAN, D. (2020): Cryptanalysis using genetic algorithm **International Research Journal of Advanced Engineering and Science**, 5(2):128–131.