Application of the Tabu Search Algorithm to Cryptography

Zakaria KADDOURI[1]
Laboratory of computer science research
Department of Computer Science
Mohammed V University Abu Dhabi
Abu Dhabi, United Arab Emirates

Fouzia OMARY[2]
Laboratory of computer science research
Department of Computer Science
Mohammed V University – Faculty of Sciences Rabat
Rabat, Morocco

***Abstract*—Tabu search is a powerful algorithm that has been applied with great success to many difficult combinatorial problems. In this paper, we have designed and implemented a symmetrical encryption algorithm whose internal structure is mainly based on Tabu search algorithm. This heuristic performs multiple searches among different solutions and stores the best solutions in an adaptive memory. First of all, we coded the encryption problem by simulating a scheduling problem. Next, we have used an appropriate coding for our problem. Then we used the suitable evaluation function. Through the symmetric key generated by our algorithm, we have illustrated the principle of encryption and decryption. The experimentations of our approach are given at the end of this paper, from which we examined our new system strengths and the elements that could be improved.**

Keywords—Symmetric encryption; heuristic; Tabu search; algorithm; scheduling problem; combinatorial problems

I. Introduction

Since the invention of writing, ***men*** expressed the need to transmit information securely making it unintelligible to anyone outside the exchange. Indeed the messages cannot be understood by the enemy, even if they are intercepted. At the current time, information in all its form circulates in digital format throughout the world in a split second on the networks. This information is exchanged every day from one point to another either by telephone, cable, optical fiber or satellite. It is likely to be read, deleted or falsified. Cryptography is a science in full operation and meets the needs of today's information security [1]. Metaheuristics are a family of optimization algorithms that are designed to solve general classes of mathematical problems by combining research procedures to quickly find the best solution [2]. Tabu search is an example of such a heuristic. It starts from an initial solution and attempts to improve it by transforming it iteratively. At each iteration, the neighborhood of the current solution is generated and the best solution in this neighborhood is chosen. To avoid circling, a Tabu list is defined to prohibit revisiting of the solutions already examined [3], [4].

We will apply the Tabu search in the main phase of cryptography. To start, we have transformed the problem of encrypting a message M to an optimization problem like to Evolutionary Ciphering System [5]-[7]. Then, we coded this problem in a particular way to bring us back to scheduling problems.

This design is based on the enrichment of the search space and the application of a well-defined model during the process of Tabu search. We notice that the random aspect used in our algorithm is crucial to the success of the method, especially if the number of iterations is relatively small.

In the next section of this article, we describe the general algorithm of TABU search and we present a detailed description of our encryption algorithm entitled Symmetrical Tabu Search Ciphering (STSC). Then, we will analyze the security of our approach, and we will compare it with same kind of systems SEC (Symmetrical Evolutionist-based Ciphering) and SMC (Symmetrical Memetic Ciphering).

II. Description of Tabu Search

A. *Definition*

Tabu search was proposed by F.Glover in 1986. The algorithm is called Tabu because there is prohibition from resuming recently visited solutions [8]. Since then, the method has become very popular, thanks to its successes to solve many problems. This algorithm introduces a notion of memory in the strategy of the search space exploration [9]. Tabu search uses local or neighborhood iterative procedures to move from solution x to a solution x' (in the vicinity of x) until the stopping conditions are met [10].

B. *Principle of Tabu Search*

The principle of Tabu search is based on a method of moving on the space of the solutions, while continually seeking to improve the current best solution and by storing in memory the list of previous moves [11], thus guiding the research outside the previously traveled zones.

The basic idea is inspired by the research techniques used in artificial intelligence. That is to keep the track of the past path of the research process in one or more memories and to use this information in order to orient future development. In practice, we will not memorize all the displacement (very costly in memory), but we will prevent only the access to some solutions during a certain number of iterations.

The neighborhood of a solution is defined by an elementary transformation (movement) permitting the switch from a solution to another solution nearby with a slight modification of the structure of the solution.

The Tabu search is based on:

- The use of flexible memory structures (short, medium and long term) allowing the full exploration of the evaluation criteria and the search history.
- A control mechanism based on alternating between the conditions that restrict (restriction Tabu) and those that liberate (aspiration criterion) the search process.
- The incorporation of the strategies of intensification and diversification of the search:
  - Intensification strategy uses the medium term memory, and serves to strengthen the search in the regions of the best solutions found recently.
  - Diversification strategy uses the long term memory, and serves to search in new regions.

C. *General Algorithm of Tabu Search*

We present below the general algorithm of Tabu search:

1) Get an initial solution (initialization).

2) Create a list of candidates' movements.

3) Choose the best candidate. This choice is based on Tabu restrictions and the aspiration criteria.

This provides an alternative, which will not be registered only if it is better than the previous solution [12].

4) Apply the stopping criterion.

- Continue: change the candidates of eligibility (Tabu restriction and aspiration criterion). Go to 2.
- Stop: Go to strategies of intensification and diversification.

The flowchart of Tabu search method is shown in Fig. 1.

The general algorithm can be represented with the following pseudo-code:

Let *NT (s)* be all candidate solutions, *T* the tabu list, *N(s)* the neighborhood of solutions and s* the current optimal solution:

$NT(s) = \{s' \in N(s) \text{ such as } s' \notin T \text{ or } f(s') < f(s^*)\}$

```
Process Tabu_method (initial solution s)
    Put $T \leftarrow \varnothing$ and $s^* \leftarrow s$;
    Repeat
            Choose s' that minimizes $f(s')$ in $N_T(s)$
            If $f(s') < f(s^*)$ then put $s^* \leftarrow s'$
            Put $s \leftarrow s'$ and update $T$
    Until the termination criterion is satisfied
End
```

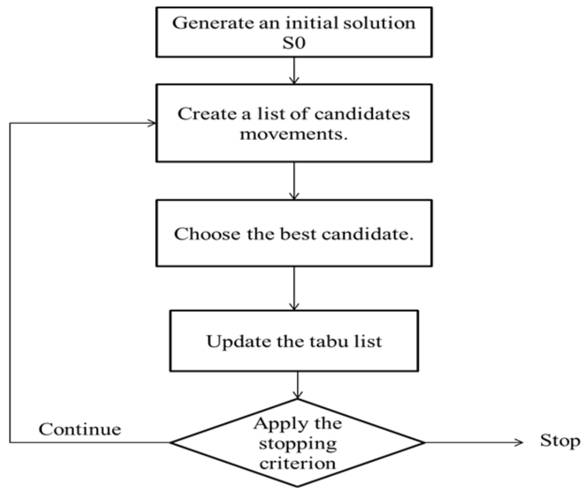
Generate an initial solution S0
Create a list of candidates movements.
Choose the best candidate.
Update the tabu list
Apply the stopping criterion
Continue
Stop

Fig. 1. Flowchart of Tabu search method.

III. Description of our Algorithm

A. *Problem formalization*

We denote by M, the binary encoding of the message M_0:

We represent the message to be encrypted by the lists which are the elements of a partition of the set $\{1, 2, ..., m\}$. The lists are composed by the different positions of each binary block. Let $B_1, B_2, ..., B_m$ the different blocks of M.

Let L_i $(1 \le i \le m)$ a list containing the different positions of the block B_i and $card(L_i)$ the number of occurrences of B_i.

Note. This breakdown only takes place for larger size messages.

We note that $L_i \cap L_j = \emptyset$ if $i \neq j$, $\forall i, j \in \{1,2,..., m\}$.

The message M may be represented by the following vector:

(B1,L1)	(B2,L2)	…	(Bm,Lm)

Our algorithm seeks to create a maximum disorder in the positions of the blocks. For this, we iteratively change the distribution of lists L_i $(1 \le i \le m)$ on the different blocks of B (without changing the content of the lists) so that the difference between the cardinal of the new list assigned to each block B_i and the cardinal of the original list L_i is maximal [13]. Therefore, we are faced with a problem of optimization and we can use the Tabu search method, including that used in scheduling problems. The latter has several versions, the most used is the one described below:

Definition of variables

- *i*: the current solution
- *i'*: the next solution achieved (neighbor solution)
- *N(i):* the space of neighboring solutions at i (the set of i')
- *m*: movement from *i* to *i'*
- *Best_Sol*: the global optimal solution that minimizes the objective function *f(i).*
- *i*:* the current optimal solution *f(i*)*
- *T*: list of Tabu movements. There can be multiple lists simultaneously. The elements of the list are *t(i, m).*
- *a (i, m):* criterion for aspiration. Determines when it is advantageous to undertake m, despite its status Tabu.

B. *Skeleton of the Algorithm*

To represent this, we have:

- Tabu List: contains prohibited movements.
- Movement (to go from one solution to another) is to swap randomly the positions of the two lists of the current solution
- The function f to minimize: eliminates solutions for which only a minority of list values have changed over initial solution.

Step 1: Choose an initial solution i in S (the set of solutions) that we call *Original-Sol*

A solution is a vector v of size m. The content of v is the list L_i $(1 \leq i \leq m)$ of position blocks. L_j is the j^{th} list which contains the new positions that will take the block B_j.

We apply a random permutation algorithm on the initial solution.

$i^* = i$

$k = 0$

$T = 0$

Step 2: $k = k + 1$ and generate a neighborhood of solutions in $N(i, k)$:

The neighborhood of the solutions will be generated by the application of permutations on the positions of the lists. Precisely, we apply random permutation on the positions of the current solution in order to generate neighboring solutions.

- The Tabu movements are not selected.
- An aspiration criteria $a(i,m)$ is applicable.

Step 3: Choose the best solution i' from the set of neighboring

solutions $N(i, k)$

$i = i'$

Let i' be a solution of $N(i, k)$ in which the lists are L'_{j1}, L'_{j2} ... L'_{jm}, *and* let f be the evaluation function on the set of solutions i'

by:

$$f(i') = -\sum_{i=1}^{m} |card(L'_{ji}) - card(L_i)|$$

Step 4: If $f(i) \leq f(i^*)$, a better solution was found

$i^* = i$

Step 5: Update the list T and the criteria aspiration.

Add the best solution in Tabu list (it is Tabu for the next r iterations).

Step 6: If a stop condition is reached, stop.

Alternatively, return to Step 2.

The flowchart of *STSC* algorithm is shown in Fig. 2.

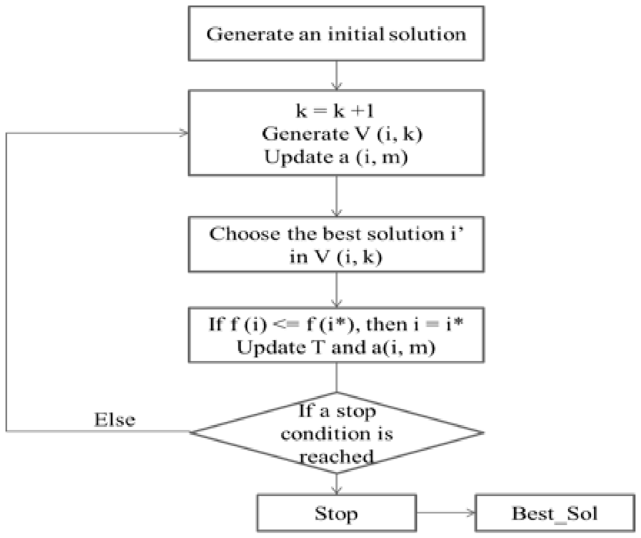
Generate an initial solution
k = k +1
Generate V (i, k)
Update a (i, m)
Choose the best solution i' in V (i, k)
If f (i) <= f (i*), then i = i*
Update T and a(i, m)
If a stop condition is reached
Else
Stop
Best_Sol

Fig. 2. Flowchart of STSC algorithm.

Let *Best_Sol* (the global optimal solution) the final solution given by STSC. We create our encryption key (*Tabu-key*) from *Original_Sol* and *Best_Sol*, Then, using the *Tabu-key* generated by our algorithm we set the corresponding cipher text block by changing the distribution lists on the various characters of the message M. Then, we concatenate the encrypted blocks (obtained by different processes). Thus, we obtain the encrypted message M from the original message M_0.

C. Decryption

Decryption must begin by looking for the reciprocal operation of the last encryption one. The message M will be broken down again into m blocks B_i that have the same size. Because of the *Tabu-key* the blocks are going to recover their lists of corresponding positions [22].

The principle of encryption and decryption can be summarized by the scheme in Fig. 3.

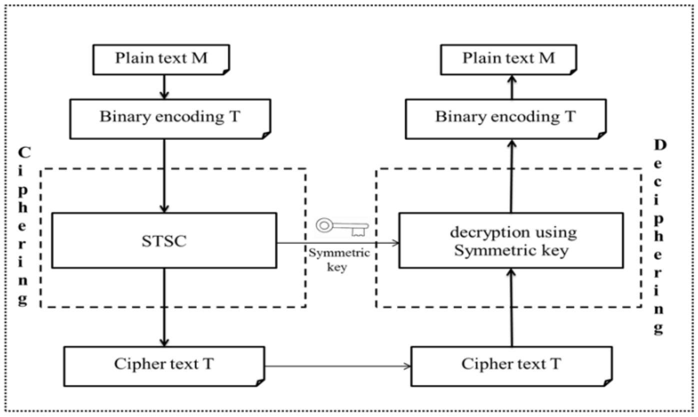

Fig. 3. Scheme of our encryption system.

IV. Experiment Results

We used our algorithm to encrypt and decrypt files such as document, image, audio, etc. The experimental environment is as follows: Processor: Intel® Core ™ i5-2450M (2.50 GHz), 4 Go of RAM; Operating System: Windows 7-x64; Programming language: Java.

A. *Comparison of the Frequencies Analysis*

Comparing the frequency analysis is a very important indicator in cryptography [14]-[16]. To illustrate the performance of the new system *STSC*, we tested our program on several messages with different sizes.

The results are shown graphically in Fig. 4.

In fact, due to the binary coding and the implementation of the system encryption *STSC*, the frequencies of the characters are no longer recognized. Therefore, cryptanalysis based on the study of the occurrence frequency cannot rely on incorrect statistics.

B. *Configuration*

We test our system on several texts of different sizes, and for each one of them, we try to find the best parameters to achieve the optimal solution in an ideal time. For this, we record the results on the number of iterations needed for the convergence of the system. Table 1 shows these results.

We can see that in the case where the size of the binary blocks is $k = 5$ or $k = 6$, our system converges and generates the encryption key in less operations compared to the other cases.

C. *Security Analysis*

1) Security key

Let $X_n = \{1, 2, ..., n\}$ be a permutation of n separate lists. For $n \in \mathbb{N}^*$, denote by E_n the set of the possible permutations of X_n, $E_n = <X_1; X_2; X_3 ; X_4 ; ...... ; X_n >$.

Counting the permutations of X back to enumerate all n-tuples formed of integers from *1* to *n* in some order. There are *n* choices for the first term of the permutation. Then for each of these first choice, there are *n–1* possibilities for the second choice, *n-2* to the third, and so on. Finally, according to this principle the cardinal of E_n is *n!*

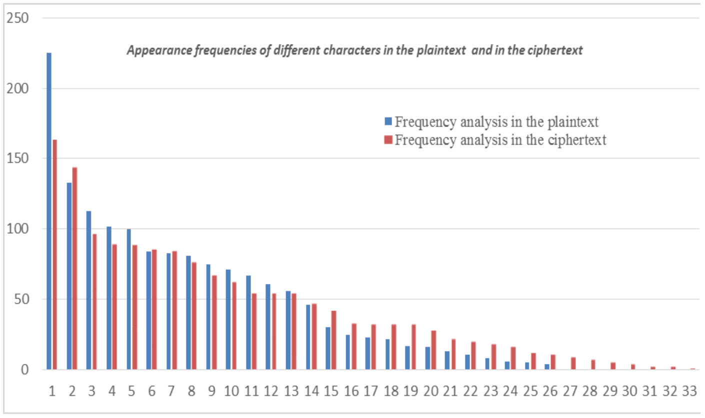

Fig. 4. Graphical representation of the appearance frequencies of different characters in the plaintext and the cipher text with STSC.

TABLE. I. Recapitulative of the results of the convergence system

Size of plaintext	Neighbors	Size of Blocks						
		5	*6*	*7*	*9*	*10*	*11*	*12*
1000 Characters	20	47	57	44	55	64	79	91
	30	59	60	78	71	60	75	89
	40	67	54	90	78	76	89	98
3000 Characters	20	42	59	76	63	98	86	73
	30	56	48	79	98	120	93	112
	40	66	54	96	102	80	115	104
6000 Characters	20	46	66	59	75	92	84	106
	30	61	63	72	88	90	81	95
	40	71	72	86	99	93	82	99
10000 Characters	20	56	61	77	73	93	107	90
	30	58	67	81	70	83	85	94
	40	53	51	80	79	76	87	111

Denote by NL the maximum number of different lists constituting our key, *KN* the maximum number of different keys generated by STSC and by *PA* the probability to encounter the right key is equal to the inverse of *KN*.

Table 2 summarizes the relation between the number of different blocks and the security of our approach.

It is noted that even in the case where the size of the text desired to be encrypted is small and the number of different blocks is reduced, the probability to fall on the correct key is very small or almost nil. We also note that the security of our approach clearly increases along with the number of different blocks.

2) Complexity of Brute Force Attack

The key length is an important security parameter [17]. Generally, the level of security of the encryption system is based on the size of the keys used (the longer the key size, the more robust the encryption system). The key to our system is composed of two elements: the Tabu-key and the block size '*k*'.

- The *Tabu-key* size is the product of the number of different blocks (NDB) and 8 bits.

TABLE. II. Summary Showing the Relation Between the Number of Different Blocks and the Security of our Approach

Size of blocks	*NL*	*KN*	*PA*
5	2^5	32!	3 ,80 e^{-36}
7	2^7	128!	2,59 e^{-216}
9	2^9	512!	2,87 e^{-1167}
10	2^{10}	1024!	1,84 e^{-2640}
11	2^{11}	2048!	5,97 e^{-5895}

TABLE. III. RECAPITULATIVE PRESENTION OF THE RESULTS OF THE KEYS SIZE GENERATED BY STSC AND THE COMPLEXITY OF THE BRUTE-FORCE ATTACK

Size of Plaintext	Data	Size of blocks					
		5	6	7	9	10	11
1000 characters	NDB	26	55	109	297	310	485
	STSC Key (bit)	216	448	880	2384	2488	3888
	Complexity of BFA	2^{216}	2^{448}	2^{880}	2^{2384}	2^{2488}	2^{3888}
3000 characters	NDB	27	59	115	315	324	599
	STSC Key (bit)	224	480	928	2528	2600	4800
	Complexity of BFA	2^{224}	2^{480}	2^{928}	2^{2528}	2^{2600}	2^{4800}
6000 characters	NDB	30	68	116	335	378	629
	STSC Key (bit)	248	552	936	2688	3032	5040
	Complexity of BFA	2^{248}	2^{552}	2^{936}	2^{2688}	2^{3032}	2^{5040}

We calculate the number of different blocks existing in the texts to determine the size of the encryption STSC key.

In the case of keys used by the system STSC the length is given in bits. In this case, the number of possibility to explore for a brute force attack (BFA) is in the order of 2^N where N is key length in bit, since the key is randomly generated.

Table 3 summarizes the results of the key size generated by STSC and the complexity of the brute-force attack.

With the current technology a 128-bit key length is already a limit impossible to achieve. If we compare the minimum key length of our system with the recommended size for symmetric systems that ensure a basic security, we can deduce that our system is able to resist against the brute-force attacks more than most other existing systems. The attacker must consider other cryptanalysis strategies if they exist. It should nevertheless take into account that the power of computers is increasing every day and an indecipherable message today can be decipherable in the future.

D. *Comparison Between the Performances of STSC System with Existing Systems*

To illustrate the effectiveness and performance of the new STSC system compared to older systems, we tested our program on multiple messages with different sizes and we recorded the number of iterations required for the system to converge.

The comparison is based not only on the quality of the results but also on the speed of convergence and computation.

Table 4 summarizes the values of the parameters used by the SEC, SMC and STSC systems to encrypt a text whose size is 6000 characters.

Table 5 presents the results obtained by applying the three precited algorithms.

Fig. 5 to 7 shows graphs of different values of the evaluation function.

TABLE. IV. VALUES OF PARAMETERS RELATING TO SEC, SMC AND STSC SYSTEMS

Parameters	SEC	SMC	STSC
Population size	30	30	-
Probability of crossover	0.7	0.7	-
Probability of mutation	0.03	0.03	-
Size of neighborhood	-	10	50
Size of the tabu list	-	-	100
Aspiration Criterion	-	-	50
Nombre d'itérations Max	100	100	200

TABLE. V. SIMULATION RESULTS OBTAINED BY SEC, SMC AND STSC SYSTEMS

Optimization methods	Evolutionary algorithm (SEC)	Memetic algorithm (SMC)	Tabu search (STSC)
Number of iterations	49	28	88
Optimum Global	5	20	50
Execution time (ms)	69	55	47
Time for a single iteration (ms)	1,40	1,96	0,53

They actually present the best configuration for which the global minimum is obtained.

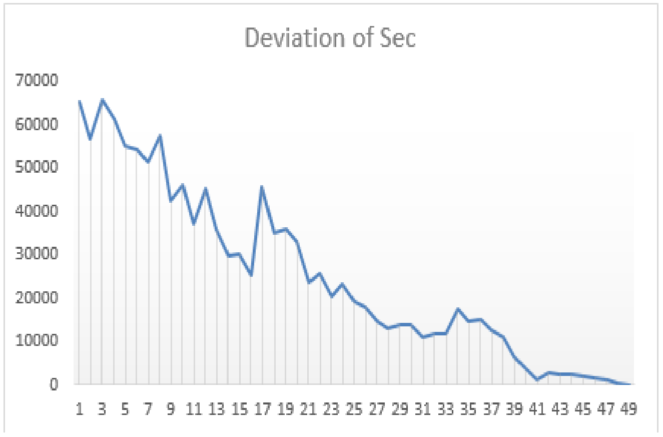

Fig. 5. Evolution of the cost by applying the SEC algorithm.

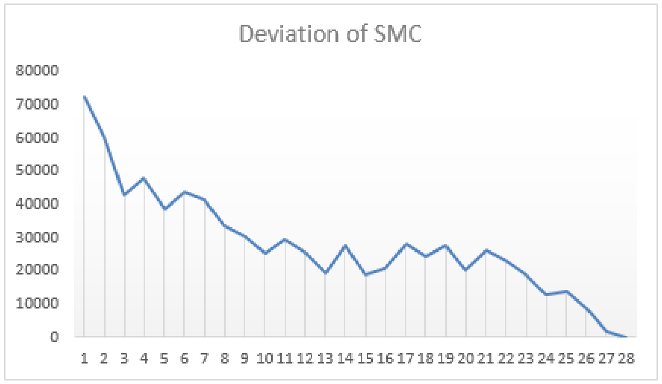

Fig. 6. Evolution of the cost by applying the SMC algorithm.

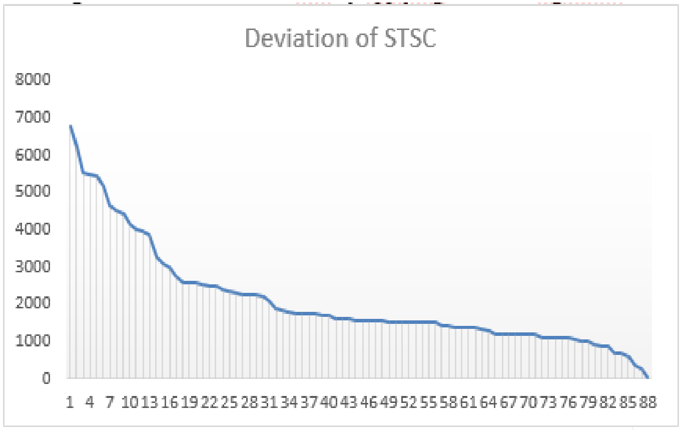

Fig. 7. Evolution of the cost by applying the STSC algorithm.

From Fig. 5, we can notice that the convergence of the SEC system is achieved in the 49th iteration to a global minimum equal to 5. From Fig. 6, the SMC system stops at the 28th iteration with an optimum value of 20 and from Fig. 7 we can see that the best configuration for the STSC system, giving the optimum overall equal to 50, is reached after 88 iterations.

From Table 5, the encryption system STSC turns out faster in terms of computation time than the other two SEC and SMC systems.

V. CONCLUSION

Tabu search is a very efficient new meta-heuristic; it can solve a wide range of problems.

In this article, the first adaptation of the meta-heuristic "Tabu search" cryptography was presented. The proposed algorithm uses a variable-length encoding to represent a symbol of the data input, which allows the encryption of any kind of information (text, image, sound, etc.).

Our system generates a secret key that we call "*Tabu-key*" which has the essential qualities to be efficient and able to resist against the brute-force attacks. According to the results of the occurrence frequencies of the characters obtained, we have shown that this method blocks the way against all the attacks which are based on the study of the occurrence frequencies of characters in a cipher text. We can also increase the security of our system by combining it with another encryption method such as [18]-[22].

REFERENCES

[1] Delfs H, Knebl H. Introduction to Cryptography: Principles and Applications. Second edition. Springer. United States of America, March 2007.

[2] Colin B. Reeves. Modern Heuristic Techniques for Combinatorial Problems. JohnWiley & Sons. Great Britain, 1993.

[3] Glover F, Laguna M. Tabu Search. Kluwer Academic Publishers, Norwell, MA, 1997.

[4] Glover F, Kelly J.P, Laguna M. Genetic Algorithms and Tabu Search: Hybrids for Optimization. Computers and Operations Research 1995; 22: 111 – 134.

[5] Omary F. Applications des algorithmes évolutionnistes à la cryptographie. University of science, Rabat, Morocco, 2006.

[6] Omary F, Tragha A, Lbekkouri A, Bellaachia A, Mouloudi A. An Evolutionist Algorithm to Cryptography. Brill Academic Publishers – Lecture Series and Computational Sciences 2005; 4: 1749-1752.

[7] Omary F, Tragha A, Bellaachia A, Mouloudi A. Design and Evaluation of Two Symmetrical Evolutionist-Based Ciphering Algorithms. International Journal of Computer Science and Network Security (IJCSNS) 2007; 7: 181-190.

[8] Glover F, Tabu search, part I. ORSA. Journal of Computing 1989; 1: 190-206.

[9] Ji M, Tang H. Global Optimizations and Tabu Search Based on Memory. Applied Mathematics and Computation 2004; 159: 449 - 457.

[10] Hanafi S. On the Convergence of Tabu Search. Journal of Heuristics 2001; 7: 47-58.

[11] Hertz A, Taillard E, Werra D. A Tutorial on Tabu Search. Proc. of Giornate di Lavoro AIRO 1995; 95: 13-24.

[12] Hertz A, Widmer M. Guidelines for the use of meta-heuristics in combinatorial optimization. European Journal of Operational Research 2003; 151: 247-252.

[13] Mouloudi A, Omary F, Tragha A, Bellaachia A. An Extension of evolutionary Ciphering System. International Conference on hybrid Information Technology 2006; 1: 314-321.

[14] Labouret G. Introduction à la cryptographie. Support du cours du cabinet Hervé Schauer (HSC). France, 2001.

[15] Stinson D. cryptography theory and practice. Discrete Mathematics and Its Applications. Canada, 2003.

[16] Schneier B. Applied Cryptography. John Wiley & Sons, France, 1996.

[17] Florin G, Natkin S. Les techniques de la cryptographie. CNAM. France, 2002. http://deptinfo.cnam.fr/Enseignement/DESS/surete/securite/courcry.pdf

[18] Omary F, Tragha A, Lbekkouri A, Bellaachia A, Mouloudi A. A New Ciphering Method Associated with Evolutionary Algorithm. Computational Science and Its Applications; ICCSA 2006; Lecture Notes in Computer Science 2006; 3984: 346-354.

[19] Kaddouri Z, Mise en œuvre de nouvelles techniques pour la securite informatique basees sur les algorithmes evolutionnistes et les fonctions de hachage. University of science, Rabat, Morocco, 2014.

[20] Kaddouri Z, Omary F, Abouchouar A. Binary Fusion Process to the Ciphering System "Sec Extension to Binary Blocks". Journal of Theoretical and Applied Information Technology 2013; 48: 067-075.

[21] Kaddouri Z, Omary F, Abouchouar A, Daari M. Balancing Process to the Ciphering System Sec. Journal of Theoretical and Applied Information Technology 2013; 52: 092-093.

[22] Kaddouri Z, Omary F. New Symmetrical Ciphering Approach Based on Memetic Algorithm. International Journal of Engineering and Technology 2014; 6: 2728-2737.