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 experiments 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.

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 searching 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:

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

Process Tabu_method (initial solution $s$)

- Generate an initial solution $S_0$
- Create a list of candidates movements.
- Choose the best candidate.
- Update the tabu list
- Apply the stopping criterion
- Put $T \leftarrow \emptyset$ and $s^* \leftarrow s$;
- Repeat
  - Choose $s'$ that minimizes $f\ (s')$ in $NT\ (s)$
  - If $f\ (s') < f\ (s^*)$ then put $s^* \leftarrow s'$
  - Put $s \leftarrow s'$ and update $T$
- Until the termination criterion is satisfied
- End

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 $L_i (1 \leq i \leq m)$ 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 $Li \cap Lj = \emptyset$ if $i \neq j, \forall i, j \in \{1,2, ..., m\}$.

The message $M$ may be represented by the following vector:

$$[B_1,L_1] [B_2,L_2] ... [B_m,L_m]$$

Our algorithm seeks to create a maximum disorder in the positions of the blocks. For this, we iteratively change the distribution of lists $L_i\ (1 \leq i \leq m)$ 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:

---

www.ijacsa.thesai.org
• 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 \( L_i \) 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:

\[ f(i') = \sum_{i=1}^{m} |\text{card}(L_{ji}') - \text{card}(L_{ji})| \]

**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.

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.
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 \( n! \).

![Graphical representation of the appearance frequencies of different characters in the plaintext and the cipher text with STSC.](image)

TABLE I. RECAPITULATIVE OF THE RESULTS OF THE CONVERGENCE SYSTEM

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

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

<table>
<thead>
<tr>
<th>Size of blocks</th>
<th>( NL )</th>
<th>( KN )</th>
<th>( PA )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>( 2^5 )</td>
<td>32!</td>
<td>3.80 ( \times ) 10^{-4}</td>
</tr>
<tr>
<td>7</td>
<td>( 2^7 )</td>
<td>128!</td>
<td>2.59 ( \times ) 10^{-6}</td>
</tr>
<tr>
<td>9</td>
<td>( 2^9 )</td>
<td>512!</td>
<td>2.87 ( \times ) 10^{-7}</td>
</tr>
<tr>
<td>10</td>
<td>( 2^{10} )</td>
<td>1024!</td>
<td>1.84 ( \times ) 10^{-8}</td>
</tr>
<tr>
<td>11</td>
<td>( 2^{11} )</td>
<td>2048!</td>
<td>5.97 ( \times ) 10^{-9}</td>
</tr>
</tbody>
</table>
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. III. Recapitulative Presentation of the Results of the Keys Size Generated by STSC and the Complexity of the Brute-Force Attack**

<table>
<thead>
<tr>
<th>Size of Plaintext</th>
<th>Data</th>
<th>Size of blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>1000 characters</td>
<td>NDB</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>STSC Key</td>
<td>216</td>
</tr>
<tr>
<td></td>
<td>Complexity of BFA</td>
<td>$2^{216}$</td>
</tr>
<tr>
<td>3000 characters</td>
<td>NDB</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>STSC Key</td>
<td>224</td>
</tr>
<tr>
<td></td>
<td>Complexity of BFA</td>
<td>$2^{224}$</td>
</tr>
<tr>
<td>6000 characters</td>
<td>NDB</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>STSC Key</td>
<td>248</td>
</tr>
<tr>
<td></td>
<td>Complexity of BFA</td>
<td>$2^{248}$</td>
</tr>
</tbody>
</table>

---

**TABLE. IV. Values of Parameters Relating to SEC, SMC and STSC Systems**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>SEC</th>
<th>SMC</th>
<th>STSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>30</td>
<td>30</td>
<td>-</td>
</tr>
<tr>
<td>Probability of crossover</td>
<td>0.7</td>
<td>0.7</td>
<td>-</td>
</tr>
<tr>
<td>Probability of mutation</td>
<td>0.03</td>
<td>0.03</td>
<td>-</td>
</tr>
<tr>
<td>Size of neighborhood</td>
<td>-</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>Size of the tabu list</td>
<td>-</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>Aspiration Criterion</td>
<td>-</td>
<td>-</td>
<td>50</td>
</tr>
<tr>
<td>Nombre d’itérations Max</td>
<td>100</td>
<td>100</td>
<td>200</td>
</tr>
</tbody>
</table>

---

**TABLE. V. Simulation Results Obtained by SEC, SMC and STSC Systems**

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

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.**
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


