# Parallel Cryptanalysis Ciphertexts Encrypted by Rotor Machines

Aleksandra Wszeborowska and Łukasz Świerczewski

***Abstract*** — this paper describes the theoretical possibilities of breaking messages encrypted by electro-mechanical rotor machines with the use of parallel algorithms optimized with the capacities of supercomputers in mind. Rotor machines had been used by the military and civil organizations until the 1980s, and their most famous representative is German Enigma.

***Keywords*** **—** Cryptography, parallel cryptanalysis, parallel programming, rotor machines

## I. Introduction

Famous Enigma owes its fame to its enormous complexity and constant attempts to break it made by the Allies during the Second World War. Germans were perfecting their solution all the time, making the construction more and more complicated by means of adding a plug-in switch and further rotors. In the time, when real computers did not exist, a number of possibilities which it was necessary to analyze was so large that breaking a text encrypted by the machine was practically impossible.

In this paper, we are not going to present a possibility of breaking a message encrypted by the original versions of Enigma, but only, in general, rotor machines built of 6 rings, which are based upon those used in the German machine.

Algorithms were implemented with the use of the environment of the C language and the MPI libraries, and also OpenMP, so as to make it possible to observe substantial acceleration on computers provided with a large number of computational units (CPUs) working parallelly. During the compilation, Intel C Compiler (version: 12.x.x) was used; it constructed a possibly effective code, activated on the processors of the Intel company.

## II. Problem Formulation

The idea behind the functioning of rotor machines is mainly based upon rotors, which dynamically change their location in the course of encrypting. It is thanks to the change of their states that an information flow path is always different, and, for example, letter 'A' will be receiving different forms at the outlet. Rotor machines are, in a matter of fact, solutions in which a poly-alphabetic cipher was used in combination with so-called Caesar's shift. The size of the shift is determined by the state of the rotors, which changes after encrypting every single letter. Moreover, the text, entering the inlet, passes through a set of rotors, reaches the reversal rotor (called the reflector), and then travels back through all the rotors. If we assume that our alphabet consists of 26 letters, the number of the possible encryptions of a text on 6 rotors equals $26^6$, i.e. 308915776.

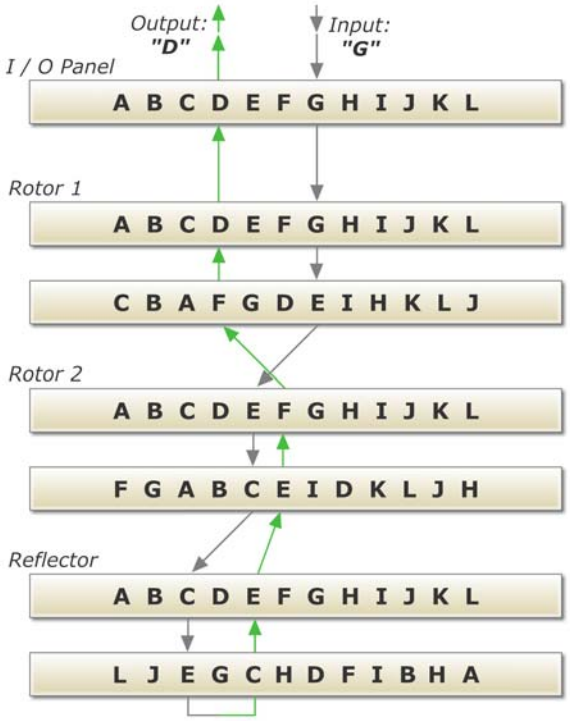


Fig. 1 Course of encrypting a letter on a simplified machine, consisting of two rotors and a reflector

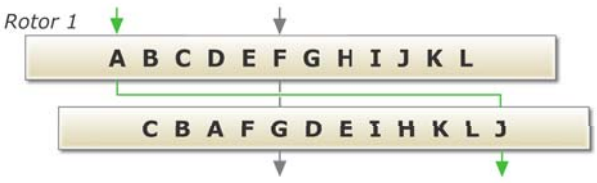


Fig. 2 Using Caesar's shift (value: 1) on a single rotor

Being in possession of the encrypted text, and knowing the construction of a machine in detail, we will only need the initial state of the rotors, which had been determined by the operator prior to encrypting, to decrypt the message. It is possible to conduct the effective parallel analysis of all possible settings and look for candidates which might be the solutions upon the basis of the statistics of a given language, or expressions defined in a dictionary.

· Aleksandra Wszeborowska is with the Computer Science and Automation Institute, College of Computer Science and Business Administration in Lomza (e-mail: olcia90.prv@wp.pl).

Łukasz Świerczewski is with the Computer Science and Automation Institute, College of Computer Science and Business Administration in Lomza (e-mail: lswierczewski@pwsip.edu.pl).

## III. Sequential Algorithm

The sequential version of the algorithm is comparatively straightforward. Commencing from the moment when all the settings of rings equal 0, we attempt to decrypt the text. After decrypting at every configuration of the machine, we analyze the text which we have received in linguistic terms – of the frequency of the occurrence of certain letters in the alphabet. In the research, we used English texts, therefore, linguistic statistics also had to be appropriate for the English language. Moreover, the text was searched in terms of the occurrence of correct words defined by the dictionary. In the course of the research, we used a list of the most commonly occurring words published by Project Gutenberg on 16th April, 2006. The entire analysis of the text was conducted in accordance with Fig. 3.

The generated text was assigned a score, which determined the valence of a given candidate as the one which could be correct. The higher the score, the closer the received text to the theoretical ideal. The score may be defined with the use of the following formula:

$$(1)\quad score = 2w + \frac{n}{100}(3 - \sum_{i=1}^{26} |f_i - f_i'|) - \frac{n}{50}|1.73 - IC|$$

where:

w – a number of words which were found in the text upon the basis of the dictionary
n – a number of characters, determining the length of the text
$f_i$ – the frequency of the occurrence of a certain letter of alphabet in a natural text
$f'_i$ – the frequency of the occurrence of a certain letter of alphabet in the examined text
IC – the Index of Coincidence of the text

TABLE I
Statistics of the frequency of the occurrence of the initial letters of alphabet for the English language

| A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|
| 8.167% | 1.492% | 2.782% | 4.253% | 12.702% | 2.228% | 2.015% |

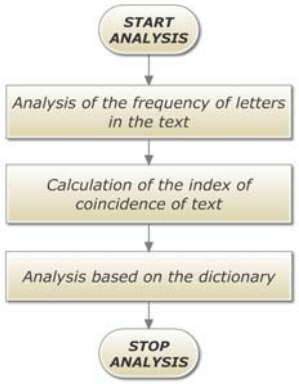


Fig. 3 Course of the analysis of text

The results are recorded in the table of structures: result_table[], the type of which was defined as follows:

```
struct candidat_text {
   char text[501];
   double score;
   int rotors_config[6];
}
```

In the text[] table, a decrypted text which is a candidate is located. The field: score determines the result received during linguistic analysis, and in the table: rotors_config[], the setting of the rotors at which the message was decrypted were recorded.

The table: result_table[] stores only such a number of the best solutions as was determined in advance. After concluding the analysis, the program records proposed solutions in a file on a hard disk. Those results already require being analyzed by a man because there is no entirely safe method, making it possible to find the correct solution without taking a risk of removing it from the set of candidates.

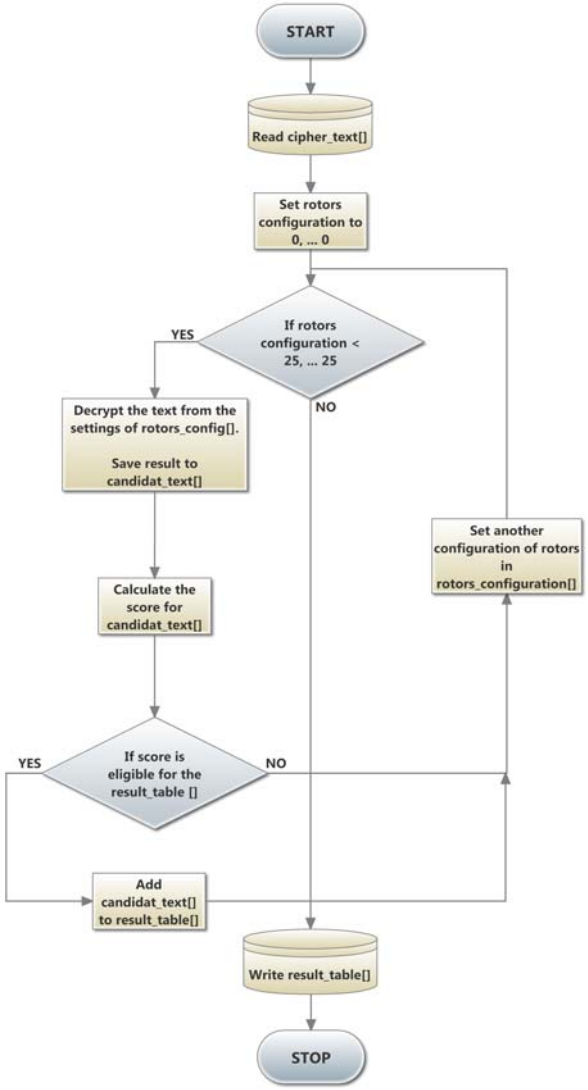


Fig. 4 Block diagram of the sequential algorithm, breaking the text encrypted by a rotor machine

The encrypted quotation: “Hope is the pillar that holds up the world” at the following initial settings of the rotors: 14, 8, 1, 9, 9, 0, has the following form:

TSTXDWXRQETVQVIGUOSFXGPMXKHZHKWTIG

Part of the received result after the cryptanalysis looks as follows:

0 26.765929 22 12 19 23 24 21
INOMEMNFHVMJAMRWENTOOKBIGORLVUKJOF
1 26.761781 23 12 24 19 19 8
HNORZCDOIINSONAMSIANDKKITOKCINRURK
2 24.870784 20 2 5 8 8 17
ANORWRSZBSFDDGOTHEMEAHEEWLPASSOGCF
3 24.848705 19 16 8 8 7 8
AJKFANOTBYNKDGOTHEMPDWESIAORIRGMYR
4 24.845878 14 8 1 9 9 0
HOPEISTHEPILLARTHATHOLDSUPTHEWORLD
5 24.833928 20 8 7 17 22 21
ANORMRSZBYXDDGOTHYMEKHESWLXAQAOKCF

Every result occupies two verses. The first number in the first verse determines the number of a solution (numbered from zero), next – the score, and subsequent digits determine the settings of the rotors at which decrypting occurred. In the second verse, there is a text decrypted at given settings.

As it can be seen, the correct solution is located only as far as on the fourth position, and the result is 24.845878. The encrypting was performed with the initial setting of rotors equal to: 14, 8, 1, 9, 9, 0. The analyzed text is composed of only 34 characters, therefore, it is difficult to receive a good result here. In case of longer cipher-texts, one may single out the correct candidate in a much better manner. The working time of the sequential algorithm analyzing this example on the processor Intel Core 2 Quad CPU Q8200 amounted to 315 minutes and 20 seconds.

## IV. Parallel Algorithm

A parallel algorithm is the extension of the sequential version. Parallelization occurs here at the level of crypt-analysis which is performed parallelly on several computational units (CPUs).

The algorithm was implemented in two different manners. In the first one, the OpenMP library was used. Parallelization consists in the division of the analyzed settings of rotors of the machine into separate processors. The simplest solution for computers with shared memory has, however, one critical section, which might be a significant limitation in case of calculations on a larger number of processors. In this section, the possible addition of a new text being the candidate for the correct variant to be recorded in the result_table[] located in the memory common for all threads. The threads may not modify this table parallelly; at a given moment, only one of them may perform a determined part of the code.

```
#pragma omp critical
{
   add_to_result_table(result_table, size_result_table,
rotors_value, score, open_tekst);

   if(size_result_table < MAX_RESULT_TABLE)
      size_result_table++;
}
```

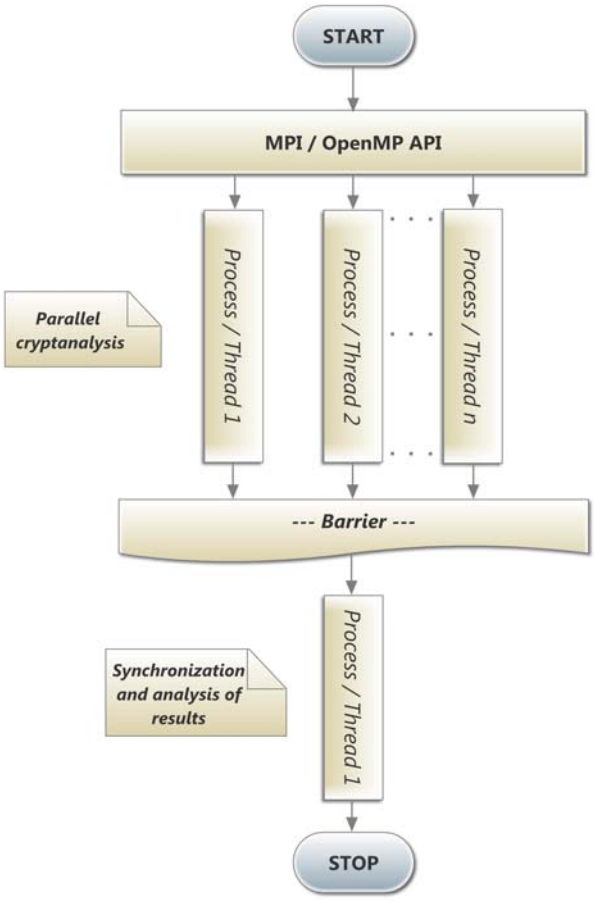


Fig. 5 Scheme of performing a parallel program

Another considered possibility is entire parallellization with the use of MPI. In this case, the data are processed in a completely independent manner, and synchronization occurs only at the moment of concluding the analysis by all the processes. It is at that stage that the best solutions calculated by separate processors are collected, and the table of the final results is collated. This stage lasts for a very short time because small quantities of information are sent between the processes; the table of structure is of a candidate_text type, one field of which covers 533 bytes. This concept is presented in Fig. 5. This solution may well be implemented with the use of OpenMP, too.

## V. Results

The prologue of Act I of the drama *Romeo and Juliet* by William Shakespeare in the English language version was subjected to the analysis. It consists of exactly 500 characters. Dictionaries of various sizes were used, thanks to which it was possible to conduct faster, but less precise, cryptanalysis, or a slower, but taking under consideration more of the words available in the dictionary, one. For very small dictionaries, the received results may be comparatively weak. However, taking under consideration more than 700 words may not bring additional benefits, either, and only additionally increase the calculation-related outlays. A good compromise is choosing a dictionary consisting of 300 to 600 most popular words, which can be observed in Fig. 6. The analysis of the dictionary constitutes an additional linear burden on the entire algorithm – it is presented in Fig. 7.

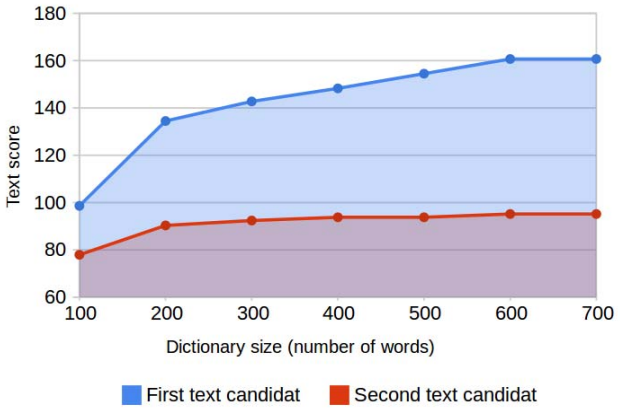


Fig. 6 Increase in the difference in the score of the first (correct) text, and, subsequently, in the score of (erroneous) one received thanks to using a dictionary of a defined size

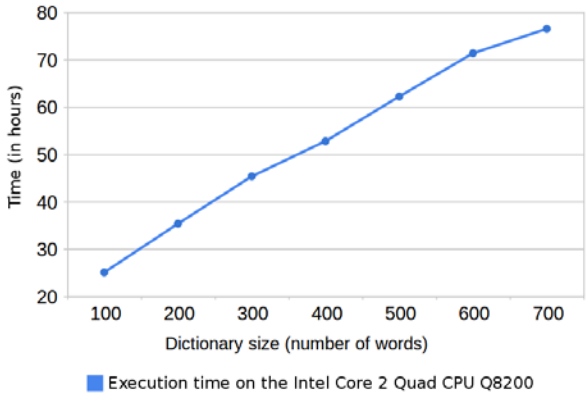


Fig. 7 Increase in time of performing program on a single core of processor Intel Core 2 Quad Q8200 after using a dictionary of a defined size

The implementation of the algorithm with the use of OpenMP is very simple and, for less demanding applications, it may prove to be completely sufficient. The acceleration in the order of 3.5 is received here for newer processors having four cores – Core i7. In case of a larger number of cores, a dominating problem may be the size of the cache memory, which, in most cases, may cause slowing down the work of a parallel algorithm, due to competition of the processors in terms of access to data. On a machine consisting of four 10-core processors Intel Xeon E7- 4860, it was possible to receive the acceleration of 28.62 (dictionary = 500 words) and 38,34 (dictionary = 100 words), which seems to be a good result. The received results were presented in Fig. 8 and in Table II.

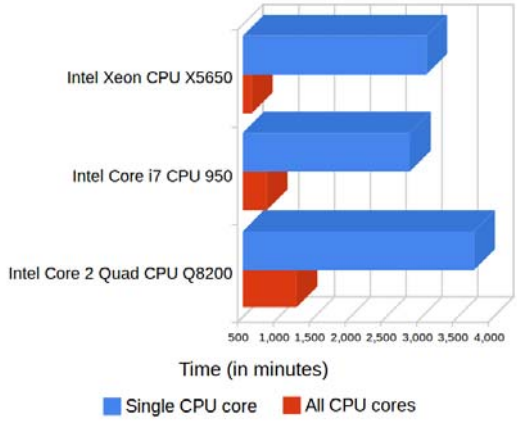


Fig. 8 Visualization of the increase in efficiency received on single multi-core processors (with the use of OpenMP)

TABLE II
Detailed results for a solution based upon OpenMP

| CPU name | Frequency | Cores | Speedup | Time (in minutes) |
|---|---|---|---|---|
| Ciphertext = 500 chars, Dictionary = 500 words | | | | |
| Intel Core 2 Quad Q8200 | 2,33 GHz | 1 | 1 | 3737 |
| | | 4 | **2,98** | 1250 |
| Intel Core i7 950 | 3,07 GHz | 1 | 1 | 2831 |
| | | 4 | **3,42** | 826 |
| Intel Xeon X5650 | 2,67 GHz | 1 | 1 | 3060 |
| | | 6 | **4,93** | 620 |
| Intel Xeon E7- 4860 | 2,27 GHz | 1 | 1 | 4380 |
| | | 40 (4x CPU) | **28,62** | 153 |
| Ciphertext = 500 chars, Dictionary = 100 words | | | | |
| Intel Core 2 Quad Q8200 | 2,33 GHz | 1 | 1 | 1505 |
| | | 4 | **2,78** | 541 |
| Intel Core i7 950 | 3,07 GHz | 1 | 1 | 1224 |
| | | 4 | **3,52** | 347 |
| Intel Xeon X5650 | 2,67 GHz | 1 | 1 | 1360 |
| | | 6 | **3,87** | 351 |
| Intel Xeon E7- 4860 | 2,27 GHz | 1 | 1 | 1879 |
| | | 40 (4x CPU) | **38,34** | 49 |

Thanks to a solution based upon MPI, we can receive nearly ideal acceleration. Of course, even in spite of using the same processors and quick communication interface between them, it will never be perfect. The acceleration presented in Fig. 9 shows, however, that in this case we do not encounter limitations which, at certain moments, are a significant burden upon parallel programming.

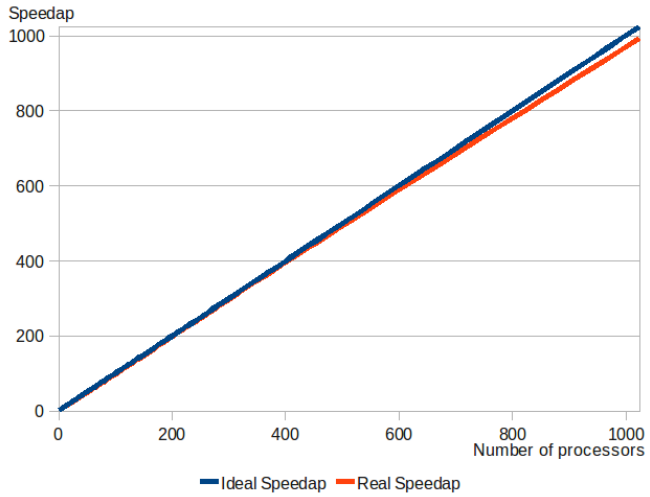


Fig. 9 Acceleration received thanks to using the MPI library and also processors Intel Xeon X5650

## VI. Conclusion

Today, it is possible to attempt to break the safeguards received by electro-mechanical machines used only as recently as 30 years ago. However, one should remember that the classical six-rotor machine presented in the paper is simplified in comparison to, for example, Enigma. The applied solutions may, however, very easily be transferred onto more complicated rotor machines.

An interesting possibility may be that of using graphic processors (nVidia CUDA, AMD FireStream, OpenCL), which, in some cases, make it possible to receive much better results.

## Acknowledgment

This research was supported in part by PL-Grid Infrastructure.

The work has been prepared using the supercomputer resources provided by the Faculty of Mathematics, Physics and Computer Science of the Maria Curie-Skłodowska University in Lublin.

## References

[1] F. L. Bauer, "An error in the history of rotor encryption devices", Cryptologia 23(3), July 1999, p. 206
[2] K. de Leeuw, "The Dutch invention of the rotor machine, 1915 - 1923." Cryptologia 27(1), January 2003, pp. 73–94.
[3] H. Beker, F. Piper, "Cipher Systems: The Protection of Communications", Wiley-Interscience, 1982, pp. 397.
[4] R. Lewand, "Cryptological Mathematics", The Mathematical Association of America, 2000, pp. 36.
[5] W. F. Friedman, The index of coincidence and its applications in cryptology. Department of Ciphers. Publ 22, 1922
[6] W. F. Friedman, L. D. Callimahos, Military Cryptanalytics, Part I – Volume 2, 1956