A Comparative Study of Classical Substitution Ciphers

Anjlee Verma
School of Computer Engineering
Lovely Professional University
Phagwara, Punjab

Navjot Kaur
Department of CSE/IT
CGC Jhanjeri
Mohali, Punjab

***Abstract*— with the rapid development in the technology, the call for security has also raised its pitch and Information Security has become an important issue during last decades. Cryptography; emerged as a solution; has reserved its unvanquishable place in the field of security. The principle objective guiding the design of any cryptographic algorithm must be the security it provides against unauthorized attack. But, the performance and cost implementation of the algorithms are also those factors which we cannot ignore. So, there is always a deemed necessity to analyze, standardize and represent these algorithms to the future researchers and struggling students so that they can learn to design effective and innovative techniques for securing data. In this paper, 7 classical substitution algorithms i.e., Affine, Atbash, Caesar, Modified Caeser Baconian, Polybius square and Letter number ciphers are implemented, and their performance is compared by encoding input files of various sizes on LINUX platform. All the algorithms are implemented in C++ language using QT creator, so that a fair comparison of execution speeds can be done. On the basis of experiments, it is concluded that Caesar cipher the best amongst the algorithms selected for the implementation.**

***Index Terms*— Cryptography, encryption, decryption, substitution cipher.**

I. INTRODUCTION

"The strength of any system is no greater than its weakest link". If we want to protect the data throughout its lifetime, we must ensure that protection mechanisms are implemented on each and every component of the information processing system. Various mechanisms can be commonly adopted in order to provide protection to our resources:

- First attempt can be done by limiting access to the computer system or media.
- Second by creating different profiles or access control mechanisms according to the roles.
- Third level of security can be provided by restricting physical access.

Above approaches can be effective up to a certain, but can be equally disadvantageous and can possess serious shortcomings. So, a more fundamental approach is provided for maintaining data security. This approach is also called Cryptography or Cryptology [1].

Cryptography (also known as cryptology) is a study and practice of hiding information. It is the technique in which a piece of raw data is taken, scrambled into gibberish mathematically, yet allowing for decrypting back into the original plain data. In other words, we can say that it is an art of manipulating messages so that they become more secure. It consists of processes of encoding and decoding. Cryptography includes the techniques for creating various systems, procedures or algorithms for secret writing. Whereas cryptanalysis consists of the techniques of breaking them.[2] Cryptology was well established in ancient times, amongst both Greeks and Romans and both of them used to practice different forms of cryptography.[3] In cryptography, ciphers are classified into various categories on the basis of their functionality. But in this paper we will be covering, implementing, analyzing and comparing one class of ciphers which is 'substitution Ciphers.'

In the field of cryptology, a 'Substitution Cipher' is a way of encrypting in which the units of plaintext replaced with the pre decided cipher text on the basis of a regular system/algorithm; here, the "units" may be taken as single letters (the most common approach), pairs of letters, triplets of letters, combinations of the above, and so forth. The receiver decrypts text by performing the substitution in reverse. The important fact about a substitution cipher is that, in a substitution cipher, the sequence in which units of the plaintext appear is retained in the cipher text, but the units themselves are modified. Further, there are various flavors of substitution ciphers. If the cipher operates on single letter of plaintext, it is known as a 'Simple Substitution Cipher'. On the other hand, if a cipher works on larger groups of letters then it is known to be a 'Polygraphic Substitution Cipher'. More classifications of substitution ciphers also exist in the form of 'Monoalphabetic Ciphers' and 'Polyalphabetic Ciphers'. In a Monoalphabetic cipher, a fixed substitution is used over the entire piece of plain text, whereas a polyalphabetic cipher uses a number of substitutions at different places in the message, where a unit from the plaintext is mapped to one of several possibilities in the ciphertext and vice versa. [4]

In this paper, the level of security an algorithm provides, is not compared. But main focus is kept on comparing some basic classical substitution ciphers on the basis of their performance and ease of implementation.

The rest of this paper is organized as follows:

Section 2 describes the algorithms selected for the implementation. Section 3 tells about the platform, the language, tools used for the implementation and other details. Section 4 discusses the performance results and issues. Lastly section 5 concludes the work.

II. IMPLEMENTED ALGORITHMS

In this work, 7 classical substitution ciphers are implemented and compared on the basis of their performance:

- Affine Cipher
- Atbash Cipher
- Caesar Cipher
- Modified Caesar Cipher
- Baconian Cipher
- Polybius Square Cipher
- Letter-number Cipher

A. *Affine Cipher*

The affine cipher is a type of monoalphabetic substitution cipher, in which every letter of the plaintext is converted to its numeric equivalent, then, the same is encoded by the use of a simple mathematical function, and then converted back into a letter.[5] Means by using the mathematical formula, it is ensured that each letter gets encrypted to another letter. Each letter is enciphered with the function

$$(ax + b) \bmod (26) \qquad \ldots\ldots 1$$

where ‘b’ is the magnitude of the shift. In the implementation values of ‘a’ and ‘b’ are taken to be 5 and 8 respectively.

B. *Atbash Cipher*

Atbash cipher was originally used for the Hebrew alphabet, but can be used for any alphabet. This is a substitution cipher with a specific key where the letters of the alphabet get reversed. That means, all 'A's in the plain text are replaced with letter 'Z's, all 'B's get substituted with 'Y's, and so on. The Atbash cipher is also an Affine cipher with the values of ‘a’ and ‘b’ taken to be 25.

C. *Caesar Cipher*

Caesar's code or Caesarian shift cipher, is one of the easiest and most widely known encryption techniques. The method is named after Julius Caesar, who made and used it in his private correspondence to communicate with his army. This is a type of substitution cipher in which every letter of the plain code is substituted by a letter 3 number of positions down the alphabet. For example, ‘A’ would be substituted by letter ‘D’, letter ‘B’ would become ‘E’, and so on. For the last letters, we consider the alphabet to be looped around in a circle and "wrap them.” V becomes Y, X becomes A, Z becomes C, and Y becomes B. In order to decipher the message back to the plain text, each letter is replaced by the one three positions before it. For example, ‘G’ becomes ‘D’ and ‘D’ becomes ‘A’.

D. *Modified Caesar Cipher*

This is a flavor of Caesar cipher in which key is not fixed to be 3. But it is asked to the user. Now if user enters 5, then ‘A’ will become ‘F’, ‘O’ will become ‘T’.

E. *Baconian Cipher*

Baconian cipher or also called Bacon's cipher is a method of steganography devised by Francis Bacon. In this algorithm, a piece of plain text is encoded into cipher text by replacing each letter of the plaintext by a group of five of the letters 'A' or 'B' as shown below:

a	AAAAA	G	AABBA	n	ABBAA	t	BAABA
b	AAAAB	H	AABBB	o	ABBAB	u-v	BAABB
c	AAABA	i-j	ABAAA	p	ABBBA	w	BABAA
d	AAABB	K	ABAAB	q	ABBBB	x	BABAB
e	AABAA	L	ABABA	r	BAAAA	y	BABBA
f	AABAB	M	ABABB	s	BAAAB	z	BABBB

Table 1. Substitution in Baconian cipher

F. *Polybius Square*

In the field of cryptology, the Polybius square cipher is also famous as the Polybius checkerboard. it is a device invented by the Ancient Greek historian and scholar Polybius. The original square used the Greek alphabet but it can be used with any alphabet. In fact, it has also been used with Japanese hiragana. When used for modern English alphabet, it appears as:

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Table 2. Substitution in PolyBius Square cipher

Now, in order to encrypt a message, Each letter is represented by its coordinates in the grid. For example, "CUT" becomes "13 45 44".

G. *Letter- Number Cipher*

Letter number is a simple substitution cipher in which every letter in the plain text is replaced by its position in number. For example, ‘hey’ is encoded to be ‘8-5-25’.

While comparing the performance of algorithms, the time taken to enter/set up the key by the user is not considered for a fair evaluation.

III. IMPLEMENTATION DETAILS

All the algorithms are coded, in C++ using QT creator running on LINUX platform. QT is a cross-platform complete development framework with tools designed to streamline the creation of stunning native applications and amazing user interfaces for desktop, embedded and mobile platforms.

It is part of the QT Project. QT Creator is a cross-platform C++ integrated development environment which is part of the SDK for the QT GUI Application development framework. QT Creator includes a code editor and integrates QT Designer for designing and building graphical user interfaces (guis) from QT widgets. The code editor in QT Creator supports syntax highlighting for various languages. In addition to that, the code editor can parse code in C++. QT Creator uses the C++ compiler from the GNU Compiler Collection on Linux. There are some tag lines which are famous about QT and are self-explanatory. Few of those are [6]:

- “Power. Beauty. Portability. Target Everything with QT.”
- “Improve Product Lifecycle and Corporate Productivity with QT”
- “If You Can Imagine It, You Can Build It With QT.”

QT is enriched with brilliant features which are preferred, recommended and desired by any programmer to develop tools, projects and GUI applications. Some of them are listed below:

- QT creator is equipped with advanced code editor.
- QT Creator focuses on providing features that help new QT users get up and running faster, and also boost the productivity of experienced QT developers.
- Code editor with C++, QML and ECMA script support.
- Group files together.
- Auto indent selection.
- Add custom build steps.
- Include forms and resource files.
- Specify settings for running applications.
- Parenthesis matching and parenthesis selection modes.
- Display inline error and warning messages.
- Enable to semantically navigate to classes, functions, and symbols.
- Provide you with context-sensitive help on classes, functions, and symbols.
- Rename symbols in an intelligent way, so that other symbols with the same name that belong to other scopes are not renamed.
- QT has rapid code navigation tools.
- Support for source code refactoring.
- Syntax highlighting and code completion.
- Code folding.
- Static code checking and style hints as you type.
- Context sensitive help.
- It has Visual debugger which enables users to interrupt program execution.
- Step through the program line-by-line or instruction-by-instruction.
- Set breakpoints.
- Examine call stack contents, watchers, and local and global variables.
- It has GUI designers which enables users to rapidly design and build widgets and dialogs using on-screen forms using the same widgets that will be used in your application.
- QT enables users to get their source code saved, built and run with one click. And many more.

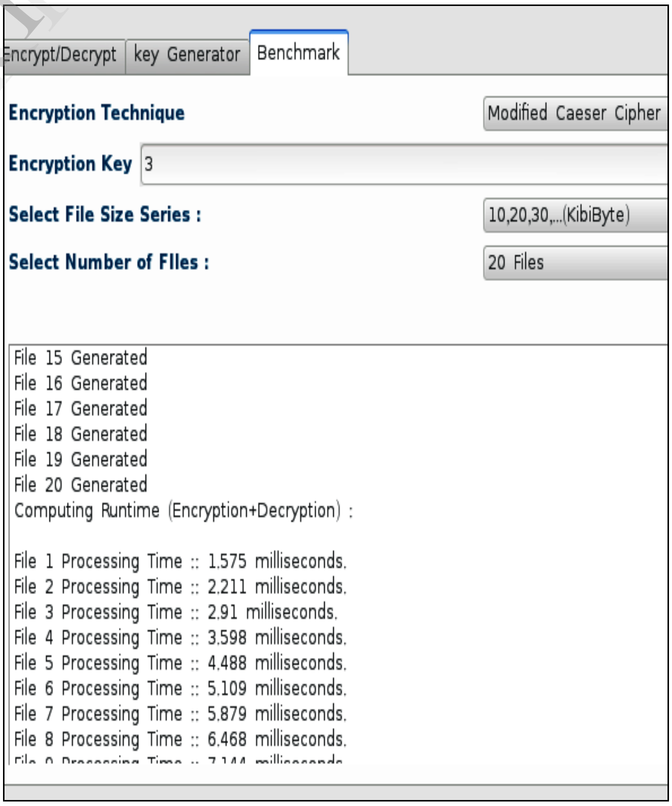
Choosing these platforms may have some disadvantages too, but as mentioned earlier, the main focus of this research was not to give the most efficient way to implement the algorithms but just to compare the performance of these algorithms.

IV. RESULTS

As mentioned above, all the algorithms have been coded in C++ from scratch using the specification documents.

A. Performance measuring interface

For calculating the time taken by algorithms, an interface was developed as shown in fig.1.



Encryption Technique: Modified Caeser Cipher
Encryption Key: 3
Select File Size Series : 10,20,30,..(KibiByte)
Select Number of FIles : 20 Files

```
File 15 Generated
File 16 Generated
File 17 Generated
File 18 Generated
File 19 Generated
File 20 Generated
Computing Runtime (Encryption+Decryption) :

File 1 Processing Time :: 1.575 milliseconds.
File 2 Processing Time :: 2.211 milliseconds.
File 3 Processing Time :: 2.91 milliseconds.
File 4 Processing Time :: 3.598 milliseconds.
File 5 Processing Time :: 4.488 milliseconds.
File 6 Processing Time :: 5.109 milliseconds.
File 7 Processing Time :: 5.879 milliseconds.
File 8 Processing Time :: 6.468 milliseconds.
File 9 Processing Time :: 7.144 milliseconds.
```

Fig 1. Graphical User Interface for performance measurement.

This interface firstly asks users to select the cryptographic scheme which they want to test. User is here provided with a drop-down list of 7 substitution ciphers chosen for the implementation. Then there is a tab “select file size series”. This tab allows users to choose between three options. Which are:

- 10,20,30,40…. Kbs.
- 1,2,3,4………. Mbs.
- 10,20,30,40…..Mbs.

Then last tab helps users to select the number of files. Which could be any one of the following:

- 10 files.
- 20 files.
- 30 files.
- 40 files.
- 50 files.

User can also enter the encryption key in the respective field which is automatically enabled or disabled as per the selected cryptographic scheme. As user presses the “Benchmark start” button, mentioned number of files are created with random content and input as plain text to the algorithm chosen. A text field provided at the bottom of the interface displays the processing steps and time taken by the particular algorithm for encrypting and decrypting all the input files.

B. Calculating execution time

For carrying out the experiment, it was decided to settle on the use of an Intel Core i3 CPU M 350 @ 2.27Ghz X 4 processor Running 64 bit Fedora 20 operating system.

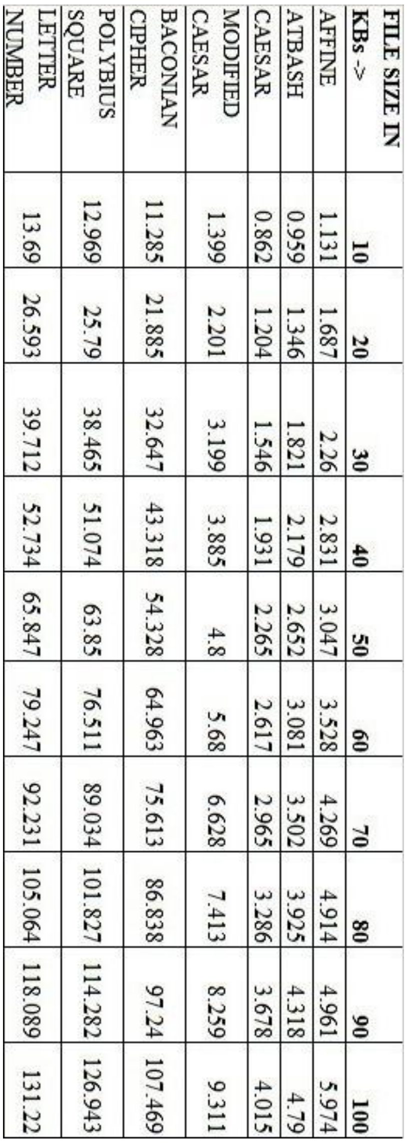
10 number of files with random contents and sizes series 10,20,30…100Kbs, were given as input to these 7 algorithms one by one and the time taken by algorithm for carrying out the whole process of encryption and decryption was recorded. Here we have not included the time taken by the user to enter the key. i.e., the calculation of time is started when the user has entered the key and has pressed the start button. Time is measured in milli-seconds.

C. Performance results of substitution ciphers

Table 3 shows the time taken by each algorithm for the process of encoding and decoding. File sizes are varied as 10,20,30,40..100 Kbs and are input to the encryption scheme and time is recorded. Rows represent the names of algorithms and columns represent the file size in Kbs and time taken by the algorithm in milli seconds.

An obvious way to compare the algorithms will be to take average of all the execution times and then rank the algorithms accordingly. Following this criteria, it is clear from Table 3, Fig. 2, and Fig. 3 that the algorithms chosen for the implementation appear in the following order on the basis of their performance:

1. Caesar Cipher (Fastest)
2. Atbash Cipher
3. Affine Cipher
4. Modified Caesar Cipher
5. Baconian Cipher
6. PolyBius Square Cipher
7. Letter Number Cipher (Slowest)



FILE SIZE IN KBs ->	10	20	30	40	50	60	70	80	90	100
AFFINE	1.131	1.687	2.26	2.831	3.047	3.528	4.269	4.914	4.961	5.974
ATBASH	0.959	1.346	1.821	2.179	2.652	3.081	3.502	3.925	4.318	4.79
CAESAR	0.862	1.204	1.546	1.931	2.265	2.617	2.965	3.286	3.678	4.015
MODIFIED CAESAR	1.399	2.201	3.199	3.885	4.8	5.68	6.628	7.413	8.259	9.311
BACONIAN CIPHER	11.285	21.885	32.647	43.318	54.328	64.963	75.613	86.838	97.24	107.469
POLYBIUS SQUARE	12.969	25.79	38.465	51.074	63.85	76.511	89.034	101.827	114.282	126.943
LETTER NUMBER	13.69	26.593	39.712	52.734	65.847	79.247	92.231	105.064	118.089	131.22

Table 3. Time taken by ciphers (in milli seconds)

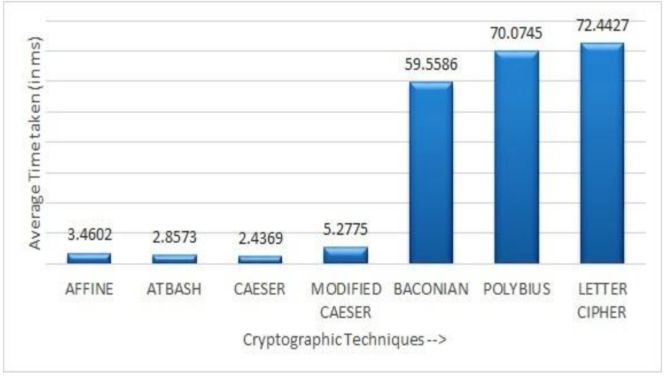


Fig. 2. Average time taken by ciphers

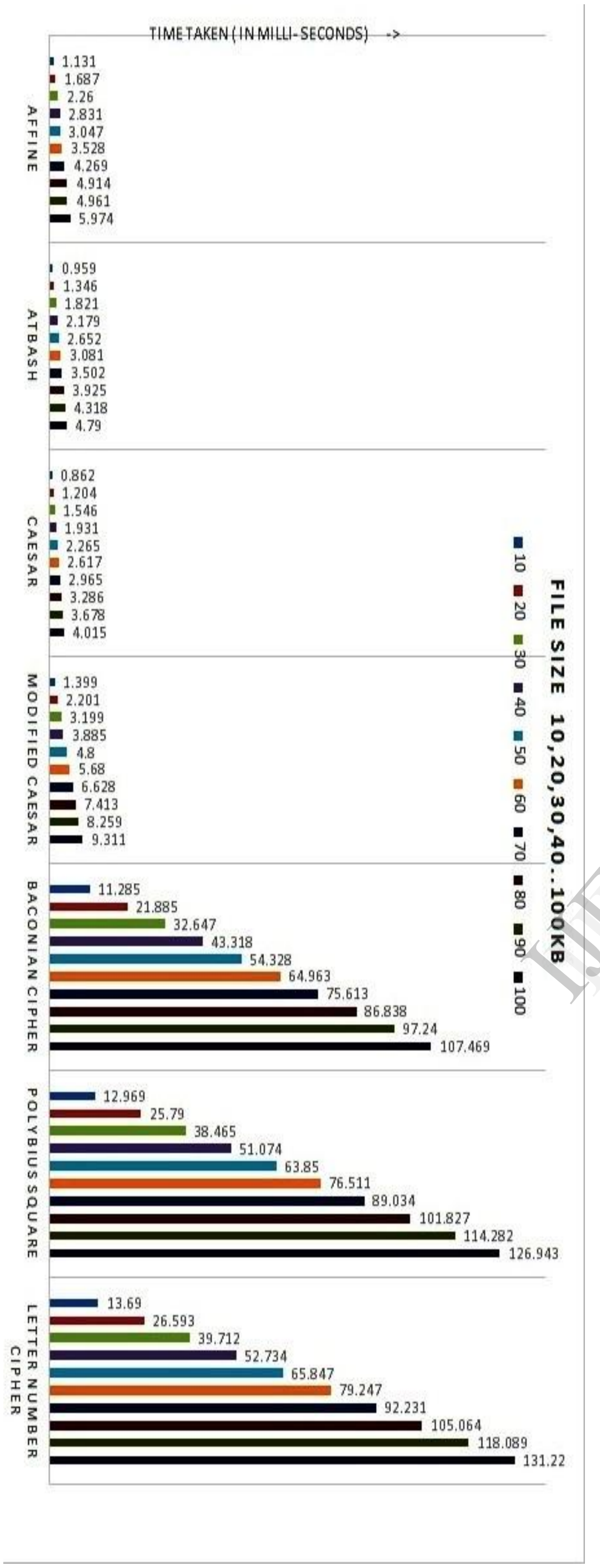


Fig. 3. Graphical representation of time taken by each algorithm.

V. CONCLUSION AND FUTURE WORK

In this paper, some famous classical substitution ciphers including Affine, Atbash, Caesar, Modified Caesar, Baconian, Poly Bius Square, Letter Number cipher have been implemented and their performance has been compared by encoding input files of various sizes. All the algorithms were implemented in a uniform language using the standard specifications. Then all of them were tested on same LINUX platform with the help of QT creator for a fair evaluation. In the end, it is concluded that Caesar cipher is the fastest algorithm followed by Atbash, Affine and Modified Caesar cipher respectively. Then there is a significant difference in the performance of other algorithms which are implemented. The huge difference is because of the reason that in Baconian, Polybius square and Letter number ciphers, each one letter of the alphabet is replaced with a series of letters or numbers. Which makes Baconian, Polybius and letter number ciphers to be the slowest substitution ciphers in the respective orders. A proposed direction for the future work is to compare the algorithms along with transposition ciphers in greater depth considering the performance/ security trade-off scale too.

REFERENCES

[1] http://www.ciphersbyritter.com/LEARNING.HTM

[2] J. F. Dooley, "A Brief History of Cryptology and Cryptographic Algorithms," pp. 1-9, 2013.

[3] J. F. Dooley, "Cryptology Before 1500: A Bit of Magic," pp. 11-17, 2013.

[4] D. Salomon, "Polyalphabetic Substitution Ciphers," pp. 59-92, 2003.

[5] H. N. H. M. G. Manocheher Kazemi, "On the Affine Ciphers in Cryptography," 2011, pp. 185-199.

[6] http://qtproject.org/wiki/Category:Tools::QtCreator