ISSN: 2250–3676

[IJESAT] INTERNATIONAL JOURNAL OF ENGINEERING SCIENCE & ADVANCED TECHNOLOGY Volume-2, Issue-5, 1198 – 1202

AN IMPROVED CAESAR CIPHER (ICC) ALGORITHM

Ochoche Abraham [1], Ganiyu O. Shefiu [2]

[1] *Senior Lecturer, Information & Media Technology Department, Federal University of Technology, Niger State, Nigeria,* ***abochoche@futminna.edu.ng***

[2] *Assitant Lecturer, Information & Media Technology Department, Federal University of Technology, Niger State, Nigeria,* ***gshefiu@yahoo.com***

Abstract

In this paper, we have introduced some novel improvements to the traditional Caesar cipher algorithm, which completely eliminates its fundamental weaknesses. First we have eliminated spaces from the ciphertext and secondly the process of creating the ciphertext by the Improved Caesar Cipher (ICC) now involves two steps; encryption and scrambling of the letters in the ciphertext, so even if it is decrypted the result would be gibberish. This fact was proven by some specific demonstrations.

Index Terms: *Improved, Caesar Cipher, Scramble, Concatenate, Ciphertext, Gibberish.*

---***---

1. INTRODUCTION

Cryptography, from the Greek word 'kryptos' (hidden) and 'graphein' (to write), is the art and science of making communication unintelligible to all except the intended recipients [5]. In the language of cryptography, the message one intends to send is called the plaintext while the message that is actually sent is called the ciphertext. Unless the secret of the encoding system, that is, the key is compromised, intercepting the ciphertext would be a waste of time because they will not be able to discover the original plaintext version of the message [7].

Ciphers make textual communication a mystery to anyone who might unduly intercept it. Hence, a cipher is a method used to encode characters to hide their values [1]. Cipher is employed in design of cryptosystem. A cryptosystem is a system, method, or process that is used to provide encryption and decryption.

One of the simplest examples of a substitution cipher is the *Caesar cipher [8]. It* is said to have been used by Julius Caesar to communicate with his army. Caesar is considered to be one of the first persons to have ever employed encryption for the sake of securing messages. Caesar decided that shifting each letter three places down the alphabet in the message would be his standard algorithm, and so he informed all of his generals of his decision, and was then able to send them secured messages [2]. One of the strengths of the Caesar cipher is its ease of use and this ease of use would be important for Caesar since his soldiers were likely uneducated and not capable of using a complicated coding system.

Further enhancement to original three places shifting of character in Caesar cipher uses modulo twenty six arithmetic for encryption key that is greater than twenty six.

$$E_n(x) = (x + n) mod\ 26$$

The most pressing weakness of this cipher is simplicity of its encryption and decryption algorithms; the system can be deciphered without knowing the encryption key. It is easily broken by reversing encryption process with simple shift of alphabet ordering [6].

$$D_n(x) = (x - n) mod\ 26$$

Another security concern is that, if how one letter should be deciphered is already known, then the shift can be determine and decipher the entire message. A better approach would be to make use of statistical data about English letter frequencies. Correcting these weaknesses of the Caesar cipher so it becomes unbreakable using existing methods, is the reason for this paper.

Caesar cipher requires little computing resources when use for cryptosystem. Improved Caesar Cipher (ICC) strengthens ciphertext generated by Caesar cipher by removing spaces in plaintext and changing positions of character in the original Caesar ciphertext to generate a new ICC ciphertext that cannot be decrypted without integer values representing space locations in plaintext, encryption key and the ICC algorithm.

2. METHODOLOGY

To encrypt a message, ICC requires plaintext and encryption key. The encryption key is an integer value and it determines location of alphabet to be used for substitution. It is based on modulo twenty six arithmetic to ensure that integer value wraps round incase encryption key supplied is more than twenty six. The output from encryption process generates two outputs viz. encrypted text (ciphertext) and space delimited integer value (s) of locations within the plaintext where spaces are available.

Decryption follows reverse operations performed during encryption. It requires decryption key, space delimited integer value(s) and of course the encrypted text. The decryption key should be complement of the encryption key so that reverse character substitution can be achieved.

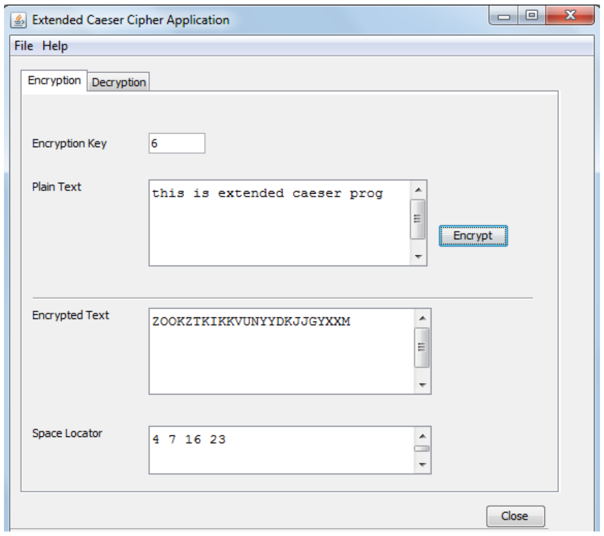
As stated earlier, Caesar cipher simply shifts encrypted character by number of positions specified while leaving spaces in between words of sentence. This is first taken care of in this paper by removing the spaces, thereby generating continuous and single stream of encrypted characters from plain text. The generated stream lacks semantic meaning of each word and entire plaintext. The locations of the removed spaces are extracted during encryption process and should be sent as part of information needed to form each word of the encrypted text and fully comprehend the content of the decrypted message.

Also, characters in the decrypted text are rearranged such that reading meaning to group of characters after spaces are removed becomes more difficult, if not impossible. Even applying dictionary attack may not reveal correct meaning to serially grouped characters. Furthermore, the characters of the encrypted text are scrambled in such a way that if an attempt is made to decrypt the ciphertext generated by ICC using any other Caesar cipher algorithm the result would be gibberish.

3. DEMONSTRATION OF RESULTS

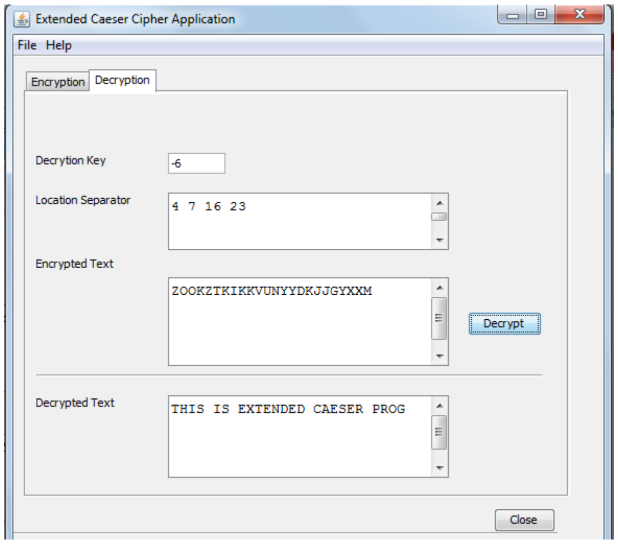
To demonstrate the efficacy of ICC, we tested it using some online programs [3, 4] of Fig-2 and Fig-3 to decrypt the ciphertext generated by it and the result can be seen in Figure 3 and Figure 4. Though each character was decrypted but the outcome was gibberish; no meaning could be deduced from each of the generated decrypted stream because of the improvements that we have made to the Caesar Cipher algorithm.

Another major strength of ICC is that it requires less computer resources when compared to other encryption algorithms. Thus ICC would be ideal for devices with limited computing resources such as mobile devices and Personal Digital Assistance (PDA).



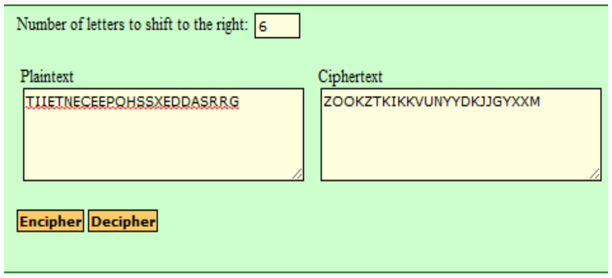
Encryption Key: 6
Plain Text: this is extended caeser prog
Encrypted Text: ZOOKZTKIKKVUNYYDKJJGYXXM
Space Locator: 4 7 16 23

Fig-1: ICC Encryption Interface



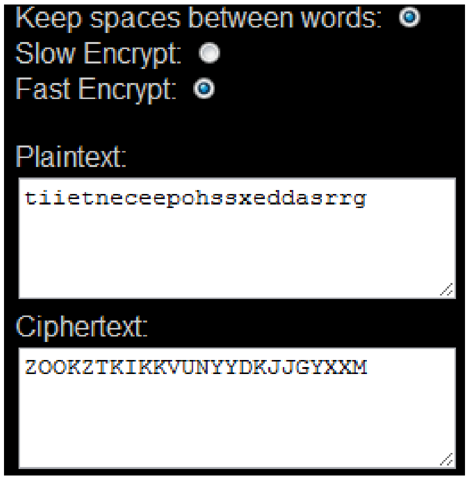
Decryption Key: -6
Location Separator: 4 7 16 23
Encrypted Text: ZOOKZTKIKKVUNYYDKJJGYXXM
Decrypted Text: THIS IS EXTENDED CAESER PROG

Fig-2: ICC Decryption Interface



Number of letters to shift to the right: 6
Plaintext: TIIETNECEEPOHSSXEDDASRRG
Ciphertext: ZOOKZTKIKKVUNYYDKJJGYXXM

Fig-3: Decrypting ICC Ciphertext using Codes and Ciphers: Caesar Ciphers


Keep spaces between words: [✓]
Slow Encrypt: [ ]
Fast Encrypt: [✓]

Plaintext:
tiietneceepohssxeddasrrg

Ciphertext:
ZOOKZTKIKKVUNYYDKJJGYXXM

Fig-4: Decrypting ICC Ciphertext using the Caesar cipher on The Black Chamber website

4. CONCLUSION

In this paper we have shown that Caesar cipher being one of the simplest and widely used encryption techniques can be fortified beyond what common Caesar cipher algorithm can achieve. To achieve the fortification, spaces in between words were removed and encrypted characters were scrambled to form a continuous stream of characters.

5. SAMPLE SOURCE CODE

```
package com.fatech;
public class CaeserEncrypter {
    private String spaceLocations;
    public String encryptCaeser(String str, int key) {
        if (key < 0) {
            key = 26 - (-key % 26);
        }
        String result = "";
        for (int i = 0; i < str.length(); ++i) {
            char ch = encryptCharacter(str.charAt(i), key);
            result += ch;
        }
return result.replaceAll(" ", "");
    }

    public String decryptCaeser(String str, int key) {
        if (key < 0) {
            key = 26 - (-key % 26);
        }
        String result = "";
        for (int i = 0; i < str.length(); ++i) {
            char ch = encryptCharacter(str.charAt(i), key);
            result += ch;
        }
        return result;
    }

    private char encryptCharacter(char ch, int key) {
        if (Character.isLetter(ch)) {
            ch = (char) ('A' + (Character.toUpperCase(ch) - 'A' +
key) % 26);
        }
        return ch;
    }

    public String getSpaceLocations(String pText) {
        pText = pText.trim();
        StringBuilder sb = new StringBuilder();
        char space = ' ';
        for (int i = 0; i < pText.length(); ++i) {
            if (pText.charAt(i) == space) {
                sb.append(i);
                sb.append(" ");
            }
        }
        spaceLocations = sb.toString();
        return spaceLocations;
    }

    public String setSpaceLocations(String enText, String
sLocations) {
        enText = enText.trim();
        sLocations = sLocations.trim();
        String[] spoints = sLocations.split(" ");
        StringBuilder sb = new StringBuilder(enText);
        int location = 0;
        for (int i = 0; i < spoints.length; ++i) {
            location = Integer.parseInt(spoints[i]);
            sb.insert(location, " ");
        }
        return sb.toString();
    }

     public String decryptBlock(String encrypted) {
        StringBuilder processedString = new StringBuilder();
        StringBuilder bufferSpace = new
StringBuilder(encrypted);
        int wordMiddle = 0;
        if (bufferSpace.length() % 2 != 0) {
            bufferSpace.append(" ");
        }
        wordMiddle = bufferSpace.length() / 2;

        for (int i = 0; i < wordMiddle; ++i) {
            processedString.append(bufferSpace.charAt(i));
processedString.append(bufferSpace.charAt(wordMiddle + i));
        }
        return processedString.toString().replaceAll(" ", "");
    }

      public String encryptBlock(String plainTrxt) {
        StringBuilder processedString = new StringBuilder();
        StringBuilder evenChar = new StringBuilder();
        StringBuilder oddChar = new StringBuilder();
        plainTrxt = plainTrxt.trim();
        for (int i = 0; i < plainTrxt.length(); i += 2) {
            evenChar.append(plainTrxt.charAt(i));
            if (i + 1 < plainTrxt.length()) {
                oddChar.append(plainTrxt.charAt(i + 1));
            }
        }
        processedString.append(evenChar).append(oddChar);
        return processedString.toString();
    }
}
```

REFERENCES

[1]. Dulaney E., *CompTIA Security+ Study Guide, Fourth Edition*, Wiley Publishing Inc., Indianapolis, Indiana, 2009.

[2]. http://www.cs.trincoll.edu /~crypto/historical/caesar.html (Savarese, C and Hart, B, The Caesar Cipher, Last updated: 04/26/2010 03:46:57).

[3]. http://en.wikipedia. org/wiki/Caesar_cipher, Caesar Cipher (Last accessed: April 20, 2012).

[4]. http://www.simonsingh. net/The_Black_Chamber/caesar.html (Last accessed: April 23, 2012).

[5]. Luciano D. and Prichett G., "Cryptology: From Caeser Ciphers to Public-Key Cryptosystem", The College Mathematics Journal, vol 18, no 1, pp. 2 -17, 1987.

[6]. Murphy C., "Data Masking with Classical Ciphers", SAS Global Forum 2010, Paper 108-2010, 2010.

[7]. Sahami, M, CS106A: Strings and Ciphers, Handout #26, October 22, 2007 (Assessed March 25, 2011).

[8]. Sinkov A., *Elementary Cryptoanalysis – A mathematical Approach*, Mathematical Association of America, 1966.

BIOGRAPHIES

Dr. Ochoche Abraham is currently a Senior Lecturer and Head of Department of the Department of Information and Media Technology of Federal University of Technology, Minna, Nigeria where he has been since 2005. He received a B. Tech (Mathematics and Computer Science), M. Tech (Mathematics) and Ph.D (in Numerical methods for Ordinary Differential Equations) from Federal University of Technology, Minna, Nigeria. Dr Abraham is a Professional Member of the British Computer Society, a Member of the International Association of Engineers, a Member of the American Mathematical Society, a Member of the Nigerian Mathematical Society, a Senior Member of the International Association of Computer Science and Information Technology, a Member of the Computer Professionals of Nigeria and a Member of The Nigerian Computer Society.

Ganiyu O. Shefiu obtained bachelor degree in Mathematics/Computer science (B.Tech,) and master degree in Information Science (M.Inf.Sc.). He is currently an Assistant Lecturer in the department of Information and Media Technology with wealth of industrial experience and certifications in various aspects of Information Technology among which are information system development (SCJP, SCBCD), Oracle Certified Professional, Sun Certified System Administrator (Solaris 10), CompTIA Project$^+$ and CompTIA Security$^+$.