International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 05 | May-2016 www.irjet.net p-ISSN: 2395-0072

# An Efficient Symmetric Cipher Algorithm for Data Encryption

**Prosper Kandabongee Yeng[1], Joseph KobinaPanford[2], James Ben Hayfron-Acquah[3], Frimpong Twum[4]**

[1] *Head of IT, IT, C and J Medicare Hospital, Accra, Ghana*

[2]*Lecturer, Computer Science, Kwame Nkrumah University of Science and Technology (KNUST), Kumasi, Ghana*

[3]*Senior Lecturer, Computer Science, Kwame Nkrumah University of Science and Technology (KNUST), Kumasi, Ghana*

[4]*Lecturer, Computer Science,Kwame Nkrumah University of Science and Technology (KNUST), Kumasi, Ghana*

---

**Abstract-*This paper presents an efficient algorithm for a symmetric cipher named, "YC1" that employs key spaces of varying lengths to encrypt and decrypt a plain text. Information Technology plays a very pivotal role in our businesses such as accomplishing complex tasks, speedy processing and many others but one very challenging concern today has to do with security in data communications. Data security in databases can be maintained by conventional methods even though malicious attackers can intercept the message or information and use them for ill-intentioned purposes. Therefore there is the need to effectively apply encryption and decryption techniques to enhance security in either stored data or data in transition. Cryptographic components are considered to be highly resistant to cryptanalysis if the right techniques or algorithms are employed.***

***"YC1" elegantly employs the principles of bit rotation in its construction process. The bitwise operators (bit shift left, bit shift right and bitwise OR operators), modular and basic arithmetic which are crucial concepts were also employed. The traditional system development life cycle (SDLC) methodology was adopted in developing YC1.***

***Depending on observations of the results of implementation of the proposed symmetric cipher on a set of real data of several types, all results were registered and analyzed. In this research, suggested data encryption algorithm is presented by using the above mentioned principles. This algorithm was tested on some data, visual and numerical measurements were used to check the strength and performance of this technique. The experiments showed that the suggested technique can be used easily to encrypt textual data.***

***Key Words:S***ymmetric cipher, Data security, Bit rotation, bitwise operators, XOR cipher

## 1.0 INTRODUCTION

The means to ensuring that parties maintain confidentiality and privacy is by means of cryptography, even in the presence of an adversary with access to the communication channel. One of the central goals of cryptography is to maintain privacy. There are other goals of communication security, such as guaranteeing integrity and authenticity of communications and many other sophisticated and fascinating goals. Cryptography is now in widespread use and if a person undertakes or indulges in online transactions using credit cards, cryptography might have indirectly been used. This is to ensure the privacy of credit card numbers. In electronic banking, cryptography is used to ensure that checks cannot be forged. Cryptography remained an art, a game of ad hoc designs and attacks for the larger part of its history. The field of cryptography has retained some of its flavor over the years and has brought in something new. The art of cryptography has now been supplemented with a legitimate science. In this research the focus shall be on that science, which is modern cryptography [1].

Modern cryptography is a cornerstone of computer and communications security with end products that are imminently practical. It is a remarkable discipline that touches on branches of mathematics that may have been considered esoteric, and it brings together fields like number theory, computational-complexity theory, and probability theory [1]. There is also cryptology, a vital area in the field of computer science which deals with cryptography and cryptanalysis. Cryptography refers to the art or science encompassing the principles and methods of converting an intelligible message (plaintext) into one that is unintelligible (cipher text), and then converting that message back to its original form [2]. In its cipher form, a message cannot be read by anyone but the intended

receiver. The act of converting a plain text message to its cipher text form is called enciphering. Reversing that act (i.e., cipher text form to plain text message) is deciphering [2].

Cryptology refers to the art and sciences involved in protecting one's information from unintended recipients and also in exploiting information. In the present age of technology it has both offensive and defensive connotations and there is high needs of components associated with cryptography or cryptology in organizations especially with regards to the security of information. This is to realize the goals of information security namely confidentiality, integrity and non-repudiation [3]. Cryptography provides integrity, authentication, non-repudiation and confidentiality and are employed in several technologies and protocols. These are the goals of cryptography but not all algorithms are able to do effectively. Most of the algorithms are focused mainly on one of the goals. Some cryptographic algorithms (like the Substitution ciphers) have been proven to be insecure over the years leading to their deprecation [4].

Now, the cost of data storage and information processing reduces with every year advancing, in order to retain effective security. Symmetric cryptographic keys must increase in size every 18 months by 1 bit because of Moore's law and a similar empirical law for storage costs. For an encryption system to have a useful shelf life and securely interoperate with other devices throughout its life span, the system should provide security for 10 or more years into the future. In general, cryptographic algorithms, are divided into the following: symmetric or secret key algorithms, asymmetric or public key algorithms, elliptic curve algorithms and hash.[4].

One very challenging concern today has to do with security in data communications. In data communications, security is of primary concern especially in the areas of secure communication channels, data encryption and maintenance of databases. Great attention is accorded to secure transmission of confidential information. Data security in databases can be maintained by conventional methods even though malicious attackers can intercept the message or information and use them for ill-intentioned purposes. Therefore there is the need to effectively apply encryption and decryption techniques to enhance security. Cryptographic components are considered to be highly resistant to cryptanalysis if the right techniques or algorithms are employed.

The cipher being proposed uses the concept of private secret key cryptography making it efficient and difficult to break. However, one major drawback is that the key must be exchanged between the sender and recipient beforehand, raising the issue of how to protect the secrecy of the key [5]. Considerable gaps exist between the target objective with respect to securing data within networks and local computers. In addition to these some of the approaches or techniques even though very effective and efficient has their own weaknesses with some of them being key length, improper implementations, design errors, size of values, reusing random parameters that should never be reused, inadequate protection mechanisms to serve as backup, hardware [6], number of rounds, sub key generation algorithm, round function, fast software encryption/decryption and ease of analysis [2]. YC1 comes to play in this situation and has the tendency to help safeguard the key due to some complexities incorporated in its design and construction process.

This research proposes an algorithm that can be considered secure at present, can scale better and has key sizes that provide adequate security levels.

## 2. LITERATURE REVIEW

Many ciphers have been developed and used over the years with some of them having become standards all over the world. In this paper, emphasis will be placed on the following ciphers. They include: XOR, NC1, DES, AES, Vigenere and Gronsfeld. YC1 (proposed cipher) uses the following concepts in its design. These concepts include: bit rotation, bitwise operators, substitution techniques, modular and basic arithmetic.

### 2.1 XOR

According to Churchhouse (2002), the simple XOR cipher is a type of additive cipher an encryption algorithm or technique that operated according to the following principles below.

$A \oplus 0 = A,$

$A \oplus A = 0,$

$(A \oplus B) \oplus C = A \oplus (B \oplus C),$

$(B \oplus A) \oplus A = B \oplus 0 = B,$

where $\oplus$ denotes the exclusive disjunction (XOR) operation. This operation is sometimes called modulus 2 addition (or

subtraction, which is identical).With this logic, a string of text can be encrypted by applying the bitwise XOR operator to every character using a given key. To decrypt the output, merely reapplying the XOR function with the key will remove the cipher [7].
For example, the string "Wiki" (01010111 01101001 01101011 01101001 in 8-bit ASCII) can be encrypted with the repeating key 11110011 as follows:

```
  01010111 01101001 01101011 01101001
⊕ 11110011 11110011 11110011 11110011
  ----------------------------------
= 10100100 10011010 10011000 10011010
```

And conversely, for decryption:

```
  10100100 10011010 10011000 10011010
⊕ 11110011 11110011 11110011 11110011
  ----------------------------------
= 01010111 01101001 01101011 01101001
```

The bitwise XOR is one of the bitwise operators used for manipulating numbers at the binary level. This encryption algorithm or cipher can easily be broken by means of frequency analysis for if the content of a message is known the key could be guessed easily. Its primary merit is that it is simple to implement, and that the XOR operation is computationally inexpensive. Another advantage is the fast that it can be used to encrypt data very quickly. A simple repeating XOR (i.e. using the same key for xor operation on the whole data) cipher is therefore sometimes used for hiding information in cases where no particular security is required [7].

The length of the key and its randomness determines the security levels of the cipher. When the keystream is generated by a pseudo-random number generator, the result is a stream cipher. With a key that is truly random, the result is a one-time pad, which is unbreakable even in theory [8].
In any of these ciphers, the XOR operator is vulnerable to a known-plaintext attack, since plaintext ⊕ ciphertext = key.

In 2014, a new approach to improving data security using modified XOR encryption algorithm was proposed. The essence of this new modified approach it to resolve some defects in the traditional XOR encryption by breaking data into blocks to remove pattern recognition. In this modified approach the two way operations of compression and decompression is eliminated although it is the normal way of avoiding pattern recognition in the traditional XOR encryption method, hence processing time is reduced.Other block encryption algorithms are outperformed by this approach as the maximum number of characters in a block is more than what could be handled by its counterparts when keys of the same length are used. The algorithm is implemented in Matrix Laboratory (MATLAB) programming language [9].

**Merits**

The XOR Cipher is a symmetric or secret-key cipher with its strength depending on

- If the key is random and is at least as long as the message, the XOR cipher is much more secure than when there is key repetition within a message.
- It is simple to implement, and that the XOR operation is computationally inexpensive.

**Drawbacks**

- The XOR operator is vulnerable to a known-plaintext attack.
- A simple XOR cipher can trivially be broken using frequency analysis.

## 2.2 NC1

NC1 also known as Nimbe Cipher 1 was developed in 2015 and is an innovative algorithm for encrypting data in a form of text. It is employs positive integers as private key-spaces of varying lengths to encrypt and decrypt data or information. NC1 elegantly employs the Binomial Theorem, Pascal Triangle, Modular and Basic Arithmetic and some concepts of the Substitution ciphers [10].

This symmetric cipher is quite non-susceptible to cryptanalysis intrusions considering its large key-space and the strength of the underlying cipher. Evaluation of key mechanisms, concepts and principles in this cipher was

highlighted and its implementation and applicability shown to demonstrate its importance.

NC1 is a monoalphabetic cipher which consists of substituting every plaintext character for a different cipher text character [10].

It makes use of 95 characters which include alphabets, numbers, special characters and other characters. Here, characters are represented by other characters. Concepts of the Binomial Theorem, Pascal triangle, Substitution ciphers, Modular and Basic arithmetic are employed in its construction. It is symmetric in nature [10].

**Merits**

NC1 is a symmetric or secret-key cipher with its strength depending on

- The nature of mathematical concepts used in its design.
- The numbers of keys that can be used in a single round of encryption.
- 95 Character substitution.
- Therefore makes NC1 computationally secure and very difficult to break using exhaustive key search.

**Drawbacks**

- The cipher becomes very slow if bigger keys are used in the encryption process.
- It also becomes slow if the text is very long thus speed and processing time of the algorithm can be significantly slow.
- The keys 0, 1 and 2 tend to produce a *Zero Division Error* or *float division by zero error* especially in the calculation of the throughput.

## 2.3 AES

The advanced encryption standard also referenced to as Rijndael is the standard for all forms of encryption today [11].It is a specification of electronic data established by the US National Institute of Standards and Technology (NIST) in 2001 [12]. AES is based on the Rijndael cipher developed by 2 Belgian cryptographers, Joan Daemen and Vincent Rijmen. They submitted a proposal to NIST during the AES selection process and were subsequently adopted as AES [13].

The U.S. government has adopted the advanced encryption standard and is now used worldwide. It is now considered and recognized as the standard of encryption and decryption worldwide. It far supersedes the data encryption standard (DES) which is now outdated. The algorithm described by AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data.

*Rijndael is a block cipher having variable key lengths and blocks. The block length and key length can be specified or represented as multiples of 32 bits but then it has a minimum of 128 bits and 256 bits [14].*

**Drawbacks**

- In hardware, the inverse cipher can only partially re-use the circuitry that implements the cipher.
- The inverse cipher is less suited to be implemented on a smart card than the cipher itself: it takes more code and cycles. It is still very fast as compared to other ciphers.

## 2.4 DES

The Data Encryption Standard (DES) is a secret key encryption algorithm or scheme. It was developed in 1977 in USA and has been adopted since then as the standard of encryption and decryption worldwide until recently. From Engelfreit (2005), due to its aging factors and insufficiency to handle data securely there was the need to introduce AES which is now the current standard. It uses a 56-bit key, which is today considered by many to be insufficient as it can with moderate effort be cracked by brute force. DES has quite a number of variants which include 3DES and TDES.

These variants use a longer key and are more secure, but has never become popular. The Advanced Encryption Standard (AES) has now superseded DES (and 3DES) as the standard encryption algorithm [15].

IBM and the US government jointly developed DES. It operates on blocks of 64 bits using a secret key that is 56 bits long. The original proposal used a secret key that was 64 bits long. It is widely believed that the removal of these 8 bits from the key was done to make it possible for U.S. government agencies to secretly crack messages. DES started out as the "Lucifer" algorithm developed by IBM. The US National Security Agency (NSA) made several modifications, after which it was adopted as Federal Information Processing Standard (FIPS) standard 46-3 and ANSI standard X3.92 [15].

The Feistel cipher structure is used in DES and includes 16 rounds of processing. It also uses a 56-bit encryption key. The key size depended on the processing and memory constraints of a single chip implementations of the DES. The key is specified with 8 bits with one of the bits used as a parity check. Unfortunately, DES was broken by Electronics Frontiers Foundation in 1999 and was subsequently found not to secure any longer. A special-purpose machine was built to accomplish this task and could decrypt a message by trying out all possible keys in less than three days. The machine cost less than $250,000 and searched over 88 billion keys per second. This prompted NIST to issue a directive of TDES (three consecutive applications) as the new standard to be used. Over time, the TDES was also found not to be secure hence prompting NIST to look for a new standard now called AES [16].

**Merits**

- Since DES was designed to run on 1977 hardware, it is fast in hardware and relatively fast in software.
- DES is also an ANSI and ISO standard - anybody can learn the details and implement it.
- DES has been around a long time (since 1977), even now no real weaknesses have been found until most recently. The most efficient attack was brute force.

**Drawbacks**

- DES broken by brute force attack or exhaustive key search.

## 2.5 VIGENERE

Simple substitution ciphers has one of its problems being very vulnerable to frequency analysis. It can easily be broken by mapping the frequency of its letters to the known frequencies in a sufficiently large cipher text. Therefore, to make ciphers more secure, cryptographers have long been interested in developing enciphering techniques that are immune to frequency analysis. One of the most common approaches is to suppress the normal frequency data by using more than one alphabet to encrypt the message. A polyalphabetic substitution cipher involves the use of two or more cipher alphabets. Instead of there being a one-to-one relationship between each letter and its substitute, there is a one-to-many relationship between each letter and its substitute's text [17].

The Vigenere Cipher, proposed by Blaise de Vigenere from the court of Henry III of France in the sixteenth century, is a polyalphabetic substitution based on Table 1:

Table 1: Blaise de Vigenere table of shifted alphabets for enciphering.

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

A AB C D E F G H I J K L M N O P Q R S T U V W X Y Z

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Each row of the table corresponds to a Caesar Cipher. The first row is a shift of 0; the second is a shift of 1; and the last is a shift of 25. The Vigenere cipher uses this table together with a keyword to encipher a message. For example, to encipher the plaintext message:*to be or not to be that is the question* Using the keyword RELATIONS. Begin by writing the keyword, repeated as many times as necessary, above the plaintext message. To derive the ciphertext using the table, for each letter in the plaintext, one finds the intersection of the row given by the corresponding keyword letter and the column given by the plaintext letter itself to pick out the ciphertext letter.

Keyword: RELAT IONSR ELATI ONSRE LATIO NSREL

Plaintext: tobeornottobethatisthequestion

Ciphertext: KSMEH ZBBLK SMEMP OGAJX SEJCS FLZSY

Decipherment of an encrypted message is equally straightforward. One writes the keyword repeatedly above the message:

Keyword: RELAT IONSR ELATI ONSRE LATIO NSREL

Ciphertext: KSMEH ZBBLK SMEMP OGAJX SEJCS FLZSY

Plaintext: tobeornottobethatisthequestion

This time, the keyword letter is being used to pick a row of the table and then traces across the row to get the column containing the cipher text letter. The index of that column is the plaintext letter.

The strength of the Vigenere cipher against frequency analysis can be seen by examining the above cipher text. Note that there are 7 'T's in the plaintext message and that they have been encrypted by 'H,' 'L,' 'K,' 'M,' 'G,' 'X,' and 'L' respectively. This successfully masks the frequency characteristics of the English 'T.' One way of looking at this is to notice that each letter of our keyword RELATIONS picks out 1 of the 26 possible substitution alphabets given in the Vigenere table. Thus, any message encrypted by a Vigenere cipher is a collection of as many simple substitution ciphers as there are letters in the keyword [17].

Drawbacks

- Breakable using the periodicity and frequency analysis

## 2.6 GRONSFELD

In Gronsfeld cipher, according to Morelli (2014), a key number is used instead of a key word as employed in that of the Vigenere cipher. The key does not usually contain repeated digits.

Here is a message written in a Gronsfeld Cipher.

cjifkqywtjioipowovlhncxlopeosggxrkx

baiiqcaguyrxrlqklcoyvewqlnhsutoiddg

qdrapdnfwkowpgwgzlskxlt

Things have been simplified for this problem as follows: only the digits between 0-5 (a-d) to be used in the key [17]. The method for attacking a Gronsfeld cipher involves the following steps:

Step 1: Write the first line of the message, and then write under each of its letter, the letters that precede it in the alphabet. Since this version is known, Gronsfeld uses only numbers between 0-5, (a-f), requiring 6 rows. The rows and columns have been numbered in order for them to be referred to.

0 1 2 3 4 5 6 7

0 c j i f k q y w tjioipowovlhncxlopeosggxrkx( Message)

1 b i h e j p x v sihnhonvnukgmbwknodnrffwqjw

2 a h g d i o w u rhgmgnmumtjflavjmncmqeevpiv

3 z g f c h n v t qgflfmltlsiekzuilmblpdduohu

4 y f e b g m u s pf ekelkskrhdjythklakocctngt

5 x e d a f l t r oedjdkjrjqgcixsgjkzjnbbsmfs

Step 2: Construct all reasonable trigrams using combinations of letters from the first three columns i.e., columns 0-2, taking 1 letter from each column. For example, to get the trigram 'ahe' by picking from rows 2,2,3. The number code for 'ahe' is 223. Since this represents the first word of the message, the trigrams formed should be possible ways to start a word or phrase. In this case, 'ahe' could be the start of 'ahead.' Actually, it is not a very likely trigram, since it repeats the number 2. Make a table of the trigrams, their number codes (which represent a portion of the possible key number) and their frequencies. This is shown in Table 2.

Table 2: A table of trigram consisting of respective codes and their frequencies

| Trigram | Code | Trigram Frequency (Table XII in Pratt) |
|---|---|---|
| aid | 215 | 24 ******** |
| age | 234 | 20 ******** |
| aff | 243 | 9 |
| ahe | 224 | 2 |
| agi | 230 | 3 |
| agg | 232 | 3 |
| big | 114 | 4 |
| chi | 010 | 22 ******** repeated numbers |
| che | 024 | 27 ******** |
| cei | 050 052 | 13 |
| bed | 155 | 2 |
| bee | 154 | 32 ******** |
| bei | 150 | 19 ******** |
| bef | 153 | 8 |
| beg | 152 | 5 |

They are all relatively frequent trigrams. They could be used as the prefix of the first word. None of them involves a repeated digit in its number code, which rules out 'chi.'

***Step 4***.

For each of the likely trigrams, apply the number formulas to each succeeding trigram in the message. For example, if 024,is applied to the letters in columns 1,2,3 the trigram gotten is, 'jgb'; if it is applied to the letters in columns 2,3,4 the outcome gotten is 'idg,' and so on. A partial table has been constructed below. Impossible trigrams are marked with (*). Filling in the rows for 'aid' and 'age' is left as an exercise.

Column 1 2 3 4 5

aid 215

age 234

bee 154 idb hag efm* jlu* pts

bei 150 idfhakefq* jly* ptw

che 024 jgb* idgfimkouqws*

Step 5:

Note that in the table above, some of the trigrams for 'bee' and 'bei' are reasonable looking, but they do not combine well with the assumption that 'bee' or 'bei' form the first three letters of the message. For example, 'bee--pts' can be

obtained by combining 'bee' with the trigram that starts in column 5, the first column that has a possible trigram, since 'efm' and 'jlu' are impossible. Similarly, the result 'bei--ptw' can be obtained by combining 'bei' and 'ptw', which also starts in column 5. Neither of these strings ('bee--pts' or 'bei--ptw') look very promising as the start of the clear message. On the other hand, combining 'che' as the prefix with the trigram that begins at column 4 ('kou'), gives the following partial string: 'che-kou.' That looks pretty promising.

***Step 6.***

Now, working with our partial solution, that begins, che-kou, replace the blank with each of the 6 letters from column 3 of the table in step 1. This gives us all possible trigrams for columns 2-3-4 that are consistent with che and kou. This list consists of:

efk, eek, edk, eck, ebk, eak

To eliminate 'efk,' 'edk,' and 'ebk' from this list, leaving Ôeek,ÕÔeckÕ and Ôeak.Õ If these substitutions are made, the outcome gotten are:

Candidate Number Code Comment

cheekou 0241024 Possibly cheek our or cheek out

checkou 0243024 Possible check out or check our

cheakou 0245024 Not very likely

Notice that a cycle is beginning to appear that goes 024-024 and there are now two candidates 02410241 and 02430243. If they are replaced by the 7th letter for each of these candidates the result is:

02410241 = cheekouw Impossible

02430243 = checkout ********* Solution!!!! **********

Step 7:

To complete the decipherment, apply the key number 0243 to the rest of the cipher text. CryptoTool can be used to complete this if the keyword 'aced' is used.

**Drawbacks**

- Gronsfeld cipher is a weaker technique than Vigenere since it only uses 10 substitute alphabets (one per digit 0..9) instead of the 26 used by Vigenere.

**3 Methodology**

In the development of YC1, the traditional system development life cycle (SDLC) was adopted. The various steps in SDLC include: analysis, design, coding, testing and maintenance. Concepts and models which are proven were used in the design process, hence can warranty a stronger cryptographic algorithm. In this paper, secondary data was used basically from journals, literature and websites. Primarily, the concepts of bit rotation and bitwise operators were used in the design phase.

## 3.1 BIT ROTATION

Bit Rotation (or circular shift) is an operation similar to shift except that the bits that fall off at one end are put back to the other end. In left rotation, as indicated inFig. 1, the bits that fall off at left end are put back at right end. In right rotation, the bits that fall off at right end are put back at left end. A circular shift is the operation of rearranging the entries in a tuple, either by moving the final entry to the first position, while shifting all other entries to the next position, or by performing the inverse operation. Rotate instructions are shifts that do not eliminate bits. For a left rotate (rol), as shown in Fig. 1, bits shifted off the left end of a data word fill the vacated positions on the right. Likewise, for a right rotate (ror), bits “falling off” the right end appear in the vacated positions at left.

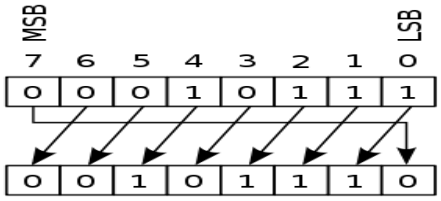


Fig. 1: Rotate Left Diagram

If the bit sequence 0001 0111 were subjected to a circular shift of one bit position; to the left would yield: 1011 1000. Note that no bits are lost.

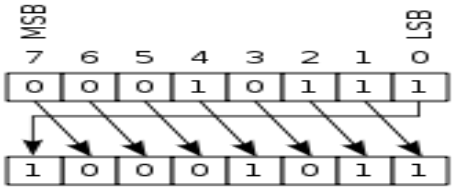


Fig. 2: Rotate Right Diagram

With reference to Fig. 2, if the bit sequence 0001 0111 were subjected to a circular shift of one bit position; to the right would yield: 1110 0010 [18].

The example below will help to make the concept of bit rotation clearer.
If n is using 8 bits. Left rotation of n = 11100101 by 3 makes n = 00101111 (Left shifted by 3 and first 3 bits are put back in last). If n is stored using 16 bits or 32 bits then left rotation of n (000...11100101) becomes 00..00**11100101**000.
Right rotation of n = 11100101 by 3 makes n = 10111100 (Right shifted by 3 and last 3 bits are put back in first) if n is stored using 8 bits. If n is stored using 16 bits or 32 bits then right rotation of n (000...11100101) by 3 becomes **101**000..00**11100**.

## 3.2 BITWISE OPERATORS

C programming language provides for six operators for bit manipulation; these may only be applied to integral operands, that is, char, short, int, and long, whether signed or unsigned [19].These operators are;

& bitwise AND

|bitwise inclusive OR

^ bitwise exclusive OR

<<left shift

>>right shift

~ one's complement (unary)

The bitwise AND operator & is often used to mask off some set of bits, for example n = n & 0177; sets to zero all but the low-order 7 bits of n. The bitwise OR operator | is used to turn bits on: x = x | SET_ON; sets to one in x the bits that are set to one in SET_ON.

The bitwise exclusive OR operator ^ sets a one in each bit position where its operands have different bits, and zero where they are the same. One must distinguish the bitwise operators & and | from the logical operators && and ||, which imply left-to-right evaluation of a truth value. For example, if x is 1 and y is 2, then x & y is zero while x && y is one.

The shift operators << and >> perform left and right shifts of their left operand by the number of bit positions given by the right operand, which must be non-negative. Thus x << 2 shifts the value of x by two positions, filling vacated bits with zero; this is equivalent to multiplication by 4. Right shifting an unsigned quantity always fits the vacated bits with zero. Right shifting a signed quantity will fill with bit signs (``arithmetic shift'') on some machines and with 0-bits (``logical shift'') on others.

The unary operator ~ yields the one's complement of an integer; that is, it converts each 1-bit into a 0-bit and vice versa. For example x = x & ~077 sets the last six bits of x to zero. Note that x & ~077 is independent of word length, and is thus preferable to, for example, x & 0177700, which assumes that x is a 16-bit quantity. The portable form involves no extra cost; since ~077 is a constant expression that can be evaluated at compile time [19].

### 3.3 YC1

YC1 is a symmetric cipher which uses substitution techniques in pairing or mapping one plaintext element to a cipher text element thereby making it a monoalphabetic cipher. It uses the concepts of bit rotation and bitwise operators.The encryption algorithm has 2 functions namely *right_rotate*, and *main* whereas the decryption algorithm has 2 functions namely *left_rotate* and *main.*

***C++ Implementation:***

***Encryption***

```
#include<iostream>

#include<fstream>

using namespace std;

ofstream output;


//array used to store characters

string
Alphabet[]={"A","B","C","D","E","F","G","H","I","J","K","L","M"
,"N","O",

"P","Q","R","S","T","U","V","W","X","Y","Z","1","2","3","4",


"5","6","7","8","9","0","~","`","!","@","#","$","%","^","&",

            "*","(",")","_","-","+","=","{","}","[","]","|","?","<",">",

            ",",".","'","\"","\\","\/","\;",":","\ ","a","b","c","d","e",


"f","g","h","i","j","k","l","m","n","o","p","q","r","s","t",

            "u","v","w","x","y","z"};


/*Function to right rotate n by d bits*/

intrightRotate(unsigned int n, unsigned int d)

{

 while(d--){

        n= (((n>>1)&(INT_MAX))|((n&1)<<(sizeof(int) * 8 -
1)));

  }

  return n;

}

/* Driver program to test above functions */

int main()

{

int d, enc;

  long n;


  //declaration of variables to store plainText and
cipherText

  string pText,cText;


  //accepting the plain text

cout<<"Please enter a plain text to encrypt:"<<flush;

getline(cin, pText);


cout<<"Please enter the number of shifts: ";

cin>>d;


output.open("C:/ContraPositive.txt");//open output file for
//apppend/additions


output.setf(ios::left); //set numbers to be left justified


  //encrypting the plain text to form cipher text

  for(inti=0;i<pText.size();i++)

  {

    for(int j=0;j<95;j++)

    {

     if(pText.substr(i,1)==Alphabet[j])

     {

       n =  rightRotate(j,d);

enc = n % 95;


        output<<n<<endl;
```

```
cout<<"\nRight Rotation of "<<pText.substr(i,1)<<" by"
<<d<<" is "<<n<<endl;

        if(enc>=0)
cText = cText + Alphabet[enc];
        else
cText = cText + Alphabet[enc*-1];
      }
    }
  }

output.close();
cout<<"\nThe encrypted text: "<<cText<<"\n"<<endl;
 system("pause");
 return 0;
}
```

***Decryption***

```
   #include<iostream>
   #define INT_BITS 32
   #include<fstream>
   using namespace std;
   ifstream input;

   //array used to store characters
   string
Alphabet[]={"A","B","C","D","E","F","G","H","I","J","K","L","
M","N","O",

"P","Q","R","S","T","U","V","W","X","Y","Z","1","2","3","4",

"5","6","7","8","9","0","~","`","!","@","#","$","%","^","&",
            "*","(",")","_","-
","+","=","{","}","[","]","|","?","<",">",
            ",",".","'","\"","\\","\/","\;",":","\
","a","b","c","d","e",

"f","g","h","i","j","k","l","m","n","o","p","q","r","s","t",
            "u","v","w","x","y","z"};

/*Function to left rotate n by d bits*/
intleftRotate(int n, unsigned int d){
  while(d--){
       (n= (n<<1)|((n>>(sizeof(int) * 8 - 1))&1));
  }
  return n;
}

/* Driver program to test above functions */
int main()
{
int d, dec;
  long n, num;

  //declaration of variables to store plainText and
cipherText
  string pText,cText;
```

```
   //accepting the cipher text
cout<<"Please enter a cipher text to decrypt:"<<flush;
getline(cin, cText);


cout<<"Please enter the number of shifts: ";
cin>>d;


input.open("C:/ContraPositive.txt");
   if (input.fail())
   {
cout<<"Input file opening failed";
      system("pause");
      exit(1);
   }


   //encrypting the plain text to form cipher text
   for(inti=0;i<cText.size();i++)
   {
     for(int j=0;j<95;j++)
     {
       if(cText.substr(i,1)==Alphabet[j])
       {
          input>>num;
           while(j<num)
           {
              j= j+95;
           }
           n =  leftRotate(j,d);
dec =  n % 95;


cout<<"\nLeft Rotation of "<<cText.substr(i,1)<<" by "<<d<<" is "<<n<<endl;


pText = pText + Alphabet[dec];
        }
      }
    }


input.close();
cout<<"\nThe decrypted text: "<<pText<<"\n"<<endl;
 system("pause");
 return 0;
}
```

3.4 Cryptosystem

The cryptosystem of YC1 is shown in Fig. 3

Let $P=C=Z_{95}$ and let $K_j = \{0< K_j\}$

For a key K, define

$$e_K(x) = d((b(p) \gg (Kj))) \% 95$$

and

$$d_K(y) = d((b(c) \ll (Kj))) \% 95$$

Fig. 3: Cryptosystem of YC1

### 3.5 INTERPRETATION AND USAGE

To encrypt the text ALPHA AND OMEGA using the new cryptographic algorithm and assume the key to be used in the encryption process is 70000. The key 70000 is chosen due to the fact that, a higher key provides higher security due to the number of shifts of bits one would have to make. A higher key make it difficult for prediction or guess. This tends to increase the complexity of YC1.

Now to encrypt the text, the numerical values of the characters must be taken into consideration. These numerical values are the positions of the characters in the xCharacter array. Table 3 shows the various characters and their positional values in the xCharacter array.

Table 3: xCharacter array with characters at various positions

| Character | Value | Character | Value |
|---|---|---|---|
| A | 0 | _ | 48 |
| B | 1 | - | 49 |
| C | 2 | + | 50 |
| D | 3 | = | 51 |
| E | 4 | { | 52 |
| F | 5 | } | 53 |
| G | 6 | [ | 54 |
| H | 7 | ] | 55 |
| I | 8 | \| | 56 |
| J | 9 | ? | 57 |
| K | 10 | < | 58 |
| L | 11 | > | 59 |
| M | 12 | , | 60 |
| N | 13 | . | 61 |
| O | 14 | ‘ | 62 |
| P | 15 | “ | 63 |
| Q | 16 | \ | 64 |
| R | 17 | / | 65 |
| S | 18 | ; | 66 |
| T | 19 | : | 67 |
| U | 20 | space | 68 |
| V | 21 | a | 69 |
| W | 22 | b | 70 |
| X | 23 | c | 71 |
| Y | 24 | d | 72 |
| Z | 25 | e | 73 |
| 1 | 26 | f | 74 |
| 2 | 27 | g | 75 |
| 3 | 28 | H | 76 |
| 4 | 29 | I | 77 |
| 5 | 30 | J | 78 |
| 6 | 31 | K | 79 |
| 7 | 32 | L | 80 |
| 8 | 33 | m | 81 |
| 9 | 34 | n | 82 |
| 0 | 35 | o | 83 |
| ~ | 36 | p | 84 |
| ` | 37 | q | 85 |
| ! | 38 | r | 86 |
| @ | 39 | s | 87 |
| # | 40 | t | 88 |
| $ | 41 | u | 89 |
| % | 42 | v | 90 |

| | | | |
|---|---|---|---|
| ^ | 43 | w | 91 |
| & | 44 | x | 92 |
| * | 45 | y | 93 |
| ( | 46 | z | 94 |
| ) | 47 | | |

**To encrypt the character 'A':**

The positional value of A is 0.

Converting 0 to binary is 0.

Right rotating it by 70000 shifts results in a 0.

Finding 0 % 95 gives an answer of 0.

Looking at the xCharacter array, the character at position 0 is 'A'.

**To encrypt the character 'L':**

The positional value of L is 11.

Converting 11 to binary is 1011.

Left rotating it by 70000 shifts results in a 720896.

Finding 720896 % 95 gives an answer of 36.

Looking at the xCharacter array, the character at position 36 is '~'.

**To encrypt the character 'P' :**

The positional value of P is 15.

Converting 15 to binary is 1111.

Left rotating it by 70000 shifts results in a 983040.

Finding 983040 % 95 gives an answer of 75.

Looking at the xCharacter array, the character at position 36 is 'g'.

Applying the same calculations to the other characters gives the complete list as illustrated in Table 4.

Table 4: Plaintext characters and their encrypted equivalents after a key 70000 is used

| Character | Application of YC1 | Encrypted Character |
|---|---|---|
| A | 0 | A |
| L | 36 | ~ |
| P | 75 | G |
| H | 92 | X |
| space | 93 | Y |
| N | 8 | I |
| D | 53 | } |
| O | 89 | U |
| M | 22 | W |
| E | 39 | @ |
| G | 11 | L |

The encrypted text is A~gxAyAI}yuW@LA.

With reference to Fig. 4 using Code::blocks;

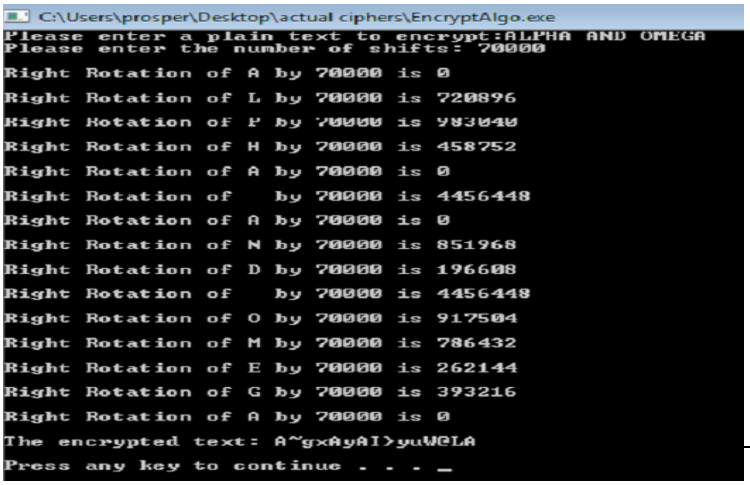


Fig. 4: C++ Console screen showing a cipher text after plaintext was provided as input.

Decrypting the text A~gxAyAI}yuW@LA

**To decrypt the character 'A':**

The positional value of A is 0.

Converting 0 to binary is 0.

Left rotating it by 70000 shifts results in a 0.

Finding 0 % 95 gives an answer of 0.

Looking at the xCharacter array, the character at position 0 is 'A'.

**To decrypt the character '~':**

The positional value of ~ is 36.

Converting 36 to binary is 100100.

Left rotating it by 70000 shifts results in 11.

Finding 11% 95 gives an answer of 11.

Looking at the xCharacter array, the character at position is 'L'.

**To decrypt the character 'g':**

The positional value of g is 75.

Converting 75 to binary is 1001011.

Left rotating it by 70000 shifts results in a 15.

Finding 15% 95 gives an answer of .

Looking at the xCharacter array, the character at position is 'P'.

Applying the same calculations to the other characters gives the complete list as shown in Table 5.

Table 5: Ciphertext characters and their decrypted equivalents after a key 70000 is used

| Character | Application of YC1 | Decrypted Character |
|---|---|---|
| A | 0 | A |
| ~ | 11 | L |
| G | 15 | P |
| X | 7 | H |
| Y | 68 | space |
| I | 13 | N |
| } | 3 | D |
| U | 14 | O |
| W | 12 | M |
| @ | 4 | E |
| L | 6 | G |

The decrypted text is ALPHA AND OMEGA.

With reference to Fig. 5 using Code::blocks;

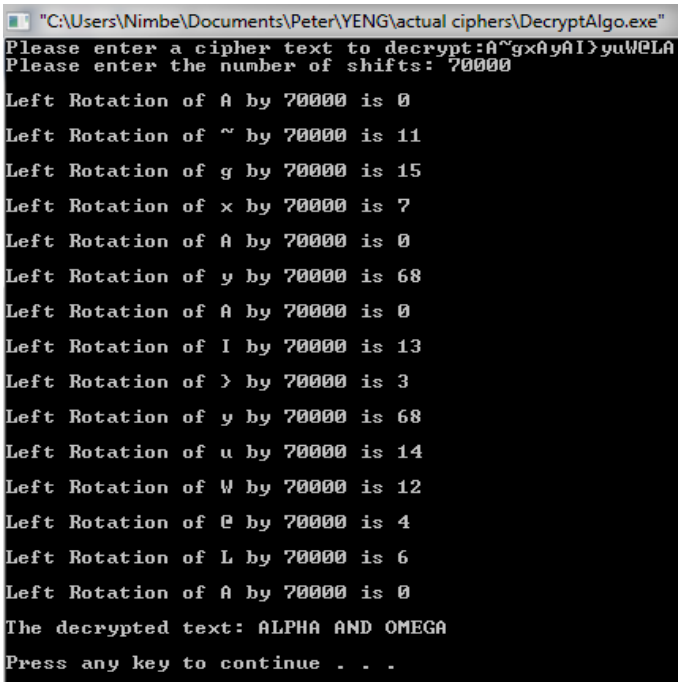


Fig. 5.C++ Console screen showing a plaintext after cipher text was provided as input.

## 4 RESULTS

4.1 Big-O-Notation

Using the worst case scenario of the running times, the big-o-notation is defined as follows:

*Encryption Algorithm*

$O(t(n)) = O(17n^2 + 23n + 101) = O(n^2)$

*Decryption Algorithm*

$O(t(n)) = O(3n^3 + 13n^2 + 19n + 103) = O(n^3)$

### 4.2 EXHAUSTIVE KEY SEARCH

YC1 is a symmetric or secret-key cipher with its strength depending on

- The key space
- Bit Rotation and Bitwise principles
- Operations used to convert plaintext into cipher

text and vice-versa

- 95 Character substitution
- Principles of Files

Just like other ciphers, YC1 has its own associated problems and advantages too.

Recovery of the plaintext of a single cipher text is not the objective of attacking a system but rather the recovery of the key. The nature of the algorithm is very critical when performing cryptanalytic attacks. This research focuses on exhaustive key search.

Every possible key is tried to deduce a meaningful or intelligible text or message. On an average basis, half of all possible keys must be tried to achieve success. Security is compromised if exhaustive key search successfully deduces the key and this is usually the case if the key space is small.

For a large key space it becomes impractical to decipher the cipher text thus the attacker must resort to statistical tests or other attacks which include cryptanalytic attacks like linear and differential cryptanalysis. Security is of paramount concern in the design of YC1. It strives to be secure even though there is no such thing as a perfect cipher. In the design of YC1, the time required breaking the cipher and the cost of breaking is taken into consideration.

YC1 is computationally secure. Table 6 shows the average time required for exhaustive key search.

Table 6: Average Time Required for Exhaustive Key Search [20]

| Key size (bits) | Number of alternative keys | Time required at 1 decryption/ µs | Time required at $10^6$ decryption/ µs |
|---|---|---|---|
| 32 | $2^{32}$ = 4.3 x $10^9$ | $2^{31}$µs = 35.8 minutes | 2.15 milliseconds |
| 56 | $2^{56}$ = 7.2 x $10^{16}$ | $2^{55}$µs = 1142 years | 10.01 hours |
| 128 | $2^{128}$ = 3.4 x $10^{38}$ | $2^{127}$µs = 5.4 x $10^{24}$ years | 5.4 x $10^{18}$ years |
| 168 | $2^{168}$ = 3.7 x $10^{50}$ | $2^{167}$µs = 5.9 x $10^{36}$ years | 5.9 x $10^{30}$ years |
| 26 Characters (permutation) | 26! = 4 x $10^{26}$ | 2 x $10^{26}$µs = 6.4 x $10^{12}$ years | 6.4 x $10^6$ years |

Now, since YC1 uses 95 character permutations, the number of alternative keys, time required at 1 decryption/ µs and time required at 106 decryption/ µs is computed as follows:

Number of Alternative Keys: 95! = 1.0 x $10^{148}$

Time required at 1 decryption/ µs: 2 x $10^{95}$ µs = ((2 x $10^{95}$)/(60 x $10^6$ x 525600) = 6.3 x $10^{81}$ years

Time required at 106 decryption/ µs: 6.3 x $10^{75}$ years

YC1 is computationally secure and very difficult to break using exhaustive key search due to its large key space and design. There is no limit as to the number of keys that can be used in the encryption process.

According to Stallings, an exhaustive key search involves trying every possible key until an intelligible translation of the ciphertext into plaintext is obtained and this is what Table 7 seeks to achieve. Table 7 shows how much time is required for various key spaces. The results of 4 binary key sizes are shown in Table 6. The 56-bit key size is used with the DES (Data Encryption Standard) algorithm, and 168-bit key size is used for triple DES.

The minimum key size specified for AES (Advanced Encryption Standard) is 128 bits. Results are also shown for substitution codes that use 26-character key in which all possible permutations of the 26 characters serve as keys. Results for YC1 (a substitution code) that uses 95-character key is shown as well. Even though YC1 at some point involves substitution, it also has some complexities built in it.

In performing key guessing, the length of the key used in the cipher as well as the number of keys that can be used in the encryption process determines the practical feasibility of conducting an exhaustive key.

Other complexities that have been introduced in the design of YC1 include:

- Nature of bit rotation.
- Bitwise principles employed.

### 4.3 COMPARATIVE ANALYSIS

Table 7: Comparative analysis of reviewed algorithms and proposed algorithm

| Algorithm | Key Size (bits) | Character type | No. of characters/numbers | Time required at 1 decryption(µs) |
|---|---|---|---|---|
| NC1 | 95 characters(permutation) | Alphanumeric Special Characters | 95 | $6.3 \times 10^{81}$ years |
| DES | 56 | Alphanumeric | - | 1142 years |
| AES | 128 | Alphanumeric | - | $5.4 \times 10^{24}$ years |
| Vigenere | 26 characters (permutation) | Alphabet | 26 | $4 \times 10^{26}$ years |
| Gronsfeld | 10 numbers (0 to 9) | Numeric | 10 | $6.3 \times 10^{-4}$years |
| XOR | 8 – 32 | Alphanumeric | Binary (0/1) | 35.8 minutes |
| YC1 | 95 characters(permutation) | Alphanumeric Special Characters | 95 | $6.3 \times 10^{81}$ years |

With reference to the Table 7, YC1 has one of the highest durations ($6.3 \times 10^{81}$ years) of time (1decryption (µs)) to decrypt. Due to the bit shift rotation method implemented, it also has unlimited key space making it comparatively stronger.

### 4.4 MERITS AND DEMERITS OF YC1

Merits

1. It has an unlimited key space.
2. It has a 95 character permutation feature making exhaustive search virtually impossible.
3. There are more throughputs with increase in the size of the text.
4. Execution of program statements is fast and hence has a good execution time.

Demerits

1. Execution time increases with increase in the size of the text.
2. Has some weak keys or elements.
3. Can't seem to find the limit of key space.

## 5 CONCLUSION

A symmetric key algorithm is presented in this paper and simulation results indicate that the new algorithm, YC1, is good in terms of security and performance even though it has some weaknesses just like other algorithms do. In future the work may be extended by performing cryptanalytic attacks like linear and differential cryptanalysis and performance analysis in comparison with some other ciphers or algorithms.

## REFERENCES

[1] M. Bellare, P. Rogaway, "Introduction to Modern Cryptography", 2005. [Online]Available: http://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf

[2] W. Stallings, "Introduction to Cryptography", 1996. [Online] Available: http://williamstallings.com/Extras/Security-Notes/lectures/classical.html

[3] J.V. Boone, "A brief History of cryptology", 2005, Naval Institute Press

[4] Cisco Security Intelligence Operations, ""Next generation encryption", April 2012. [Online] Available: http:// www.cisco.com/web/about/security/intelligence/nextgen_crypto.html

[5] C. Ishrat, S. Anushree, K.F. Saba, S. Granthali, "An Improved Symmetric Key Cryptography With DNA Based Strong Cipher", 2009, University of Mumbai

[6] B. Schneier, "The blowfish encryption algorithm", 1993. [Online] Available: https://www.schneier.com/blowfish.html

[7] R. Churchhouse, "Codes and Ciphers: Julius Caesar the Enigma and the internet ", 2002, Cambridge University Press, [Online] Available: http://en.wikipedia.org/wiki/XOR_cipher

[8] W.T. Tutte, "Fish and I", Transcript of a lecture, 1998 University of Waterloo [Online] Available:http://cryptocellar.web.cern.ch/crytocellar/tutte.pdf

[9] E.T. Oladipupo, O.A. Alade, "An Approach to Improve Data Security using Modified XOR Encryption Algorithm", 2014, International Journal of Core Research In Communication Engineering, Volume No. 1, Issue No. 2

[10] P. Nimbe, J.B. Hayfron-Acquah, B. A. Weyori, "An Improved Symmetric Cipher Encryption for Securing Data", Asian Journal of Mathematics and Computer Research, 2015.

[11] J. Daemen, V. Rijmen, "AES Proposal: Rijndael", 2003, National Institute of Standards and Technology [Online] Available: http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf

[12] FIBS, "Announcing the Advanced Encryption Standard (AES)", 2011, Federal Information Processing Standards Publication 197, United States National Institute of Standards and Technology (NIST)

[13] H.B. Westland, "NIST reports measurable success of Advanced Encryption Standard", 2002, Journal of Research of the National Institute of Standards and Technology

[14] J. Daemen, V. Rijmen, "The design of Rijndael, AES - The Advanced Encryption Standard", 2002, Springer-Verlag Berlin Heidelberg

[15] A. Engelfreit, "The Des encryption algorithm", 2005 [Online] Available: http://www.iusmentis.com/technology/encryption/des/

[16] A. Kak, "Lecture Notes on Computer and Network Security", 2015, Purdue University [Online] Available: https://engineering.purdue.edu/kak/compsec/Lectures.html

[17] R. Morelli, "The Vigenere Cipher", 2014 [Online] Available: http://www.cs.trincoll.edu/~crypto/historical/vigenere.html

[18] N.B. Dodge, "Shift and Rotate Instruction", 2012 [Online]Available: http://www.utdallas.edu/~dodge/EE2310/lec14.pdf

[19] W. Kernighan, M. Ritchie, "The C Programming Language", 1978

[20] W. Stallings, "Cryptography and network security", Principles and Practice, 5th edition, Boca Raton, Florida: Chapman & Hall/CRC 2011 [Online] Available: http://faculty.mu.edu.sa/public/uploads/1360993259.0858Cryptography and Network Security Principles and Practice, 5th Edition.pdf

## BIOGRAPHIES

**Prosper K. Yeng** is the Head of IT for C& J Medicare Hospital, Accra, Ghana. He holds Bsc. Computer Science from Garden City University College, Kumasi Ghana.
He is a Postgraduate Student (MSc Information Technology), KNUST, Kumasi, Ghana, and also holds Project Management Professional (PMP), and ITIL(IT infrastructure Library). His research interests are Information and Network Security, Cryptographic Algorithm Analysis and database management and analysis.

**J. K. Panford**received his BSc degree in Computer Science from the Kwame Nkrumah University of Science and Technology (KNUST), Kumasi, Ghana, his MSc Software Technology degree from Stuttgart University, Stuttgart, Germany He is currently a Lecturer and pursing his PhD in Computer Science at the Department of Computer Science, KNUST. He has over 15 publications to his credit. His research areas include Cloud Computing, Networking, Image Processing and Computer Security, Software technology and Embedded Systems.

**Dr J. B. Hayfron-Acquah**received the BSc degree in Computer Science from the Kwame Nkrumah University of Science and Technology (KNUST), Kumasi, Ghana, his MSc Computer Science and Applications degree from Shanghai University of Science and Technology, Shanghai, China and his PhD from theSouthampton University, Southampton, England. He is currently a Senior Lecturer at the Department of Computer Science, KNUST. He has over 40 publications to his credit. His research areas include Biometrics, Cloud Computing, Networking, Image Processing and Computer Security.

**F.Twum**received his B.Sc.(Hons) degree in Electrical and Electronic Engineering and MSc. Internet and Multimedia Engineering from London South Bank University in 2004, and 2007 respectively. He also received MSc. Degree in Information System from Roehampton University, London in 2011. He is currently pursuing his Ph.D at the Department of Computer Science, KNUST, where he also works as a Lecturer. His areas of research interest include: Computer Networks and Security, Cloud Computing, E-Commerce, Software Engineering.