# ENHANCED ENCRYPTION USING NEXT NEAREST PRIME KEY GENERATION ALGORITHM

**ZAINAB ADAMU ALIYU**
*Computer Science Department, Federal College of Education, Zaria, P.M.B 1041, Kaduna State, Nigeria.*

## ABSTRACT

*Today the usage of internet increases tremendously. So, there is need of increased security in transmitting data. Cryptography is a process of scrambling data into unknown format which provides more security to the data. Modern cryptography is mainly based on mathematical theory and computer science practice. Cryptography process is done with the help of encryption and decryption. The basic two ideas behind the cryptography technique are substitution and transposition. One such techniques was proposed by Kiran et al. for improved data confidentiality and integrity. Although the technique of Kiran et al. has some merits, its efficiency can be enhanced. We propose to improve performance of the next nearest prime key generation algorithm used by Kiran et al. We plan to adopt the Sieve of Eratosthenes algorithm in the framework of Kiran et al. We plan to evaluate, empirically and theoretically, the relative time efficiencies of the original algorithm of Kiran et al. and other enhanced version of it.*

***Keywords:*** *Cryptography, Key generation, Next nearest prime, Sieve of Eratosthenes*

## Introduction

Human being from ages had two inherent needs to communicate and share information and to communicate selectively. These two needs gave rise to the art of coding the messages in such a way that only the intended people could have access to the information. Unauthorized people could not extract any information, even if the scrambled messages fell in their hand (tutorial point, 2015). Due to tremendous growth in communication technology now the security of data is really a big issue. In banking system the data must be fully

secured. Under no circumstances the authentic data should go to hacker. In defense, the security of data is much more prominent. The leakage of data in defense systems can be highly fatal and can cause too much destruction. Due to this security issue different cryptographic methods are used by different organizations and government institutions to protect their data online. But, cryptography hackers are always trying to break the cryptographic methods or retrieve keys by different means such as Exhaustive Key Search Attack, and Related Key Attack etc. For this reason cryptographers are always trying to produce new cryptographic methods to keep the data safe as much as possible (Ajit, et al, 2013).

The art and science of concealing the messages to introduce secrecy in information security is recognized as cryptography. The word 'cryptography' was coined by combining two Greek words, 'Krypto' meaning hidden and 'graphene' meaning writing (Meena & Komathi, 2015).

In cryptography, data that can be read and understood without any special measures is called plaintext or clear text (Kiran, et al, 2016). The method of cover up plaintext in such a way as to hide its substance is called encryption.

In cryptography three types of algorithms are present, Symmetric key algorithm, Asymmetric key algorithm and Hash function. In this work we are focusing on symmetric key algorithm. One of the main advantages of using the symmetric key algorithm is that a single key is used in both encryption and decryption and the computational power to this technique is small (Acharya *et al*, 2014). Cryptographic algorithms play a major role for data security. As the complexity of algorithm is high the risk of breaking the original plaintext from that of cipher text is less. Greater complexity means greater security (Haldankar & Kuwelkar, 2014). Encryption is the process of encoding plain text into cipher text (secure data). Decryption is the revoking of the encryption process by which cipher text is converted to plain text, (Ayushi, 2010).

## Review of Related work

Srikantaswamy & Phaneendra, (2011) in their work an Enhanced One Time Pad Cipher with More Arithmetic and Logical Operations with Flexible Key Generation Algorithm demonstrates that one-time pad can be used as an efficient encryption scheme by involving arithmetic and logical operations. A new key generation technique, to generate a key of any length just by providing

a seed value, was proposed to encrypt the message. The problem generating key value has been solved by the use of key generation algorithm. This approach is applicable to only 127 characters. It is not providing full support for ASCII characters.

B. Bazith, (2013) proposed an Automatic Key Generation of Caesar Cipher in his work he uses combination of Caesar cipher and rail fence cipher, the Caesar cipher is used to generate key automatically. It takes a single round for the encryption/decryption process. This approach has the limitation that; Rail fence cipher offers essentially no communication security and Only a single round for both encryption and decryption which makes the system prone to brute force attack.

Devi & Palanisamy, (2014) present a Multi-Level Encryption using Simplified Data Encryption Standard (SDES) Key Generation Technique with Genetic Algorithm. In this research work SDES algorithm is use for key generation. Encryption algorithm is done in three steps. Caesar cipher substitution, transposition and arithmetic and logical operations are applied to generate cipher text. It involve three rounds. It is applicable only for 94 characters, a look up table is necessary and key length is small.

Kiran et al, (2015) proposed a Multi-stage Encryption Using Seeded SDES algorithm in their work they presents a multistage encryption algorithm. At the end of each stage an intermediate cipher is produced and is considered to be an input of the next stage. The key is generated by using SEEDED SDES algorithm. Final cipher text is derived from the local binary pattern (LBP). It limited to 94 characters. Look up table is necessary, key length is small and only column wise retrieval process is proposed.

Kiran et al, (2016) proposed Nearest Prime Cipher for Data Confidentiality and Integrity. This research work generates an 8-bit random sequence for key generation. Key is obtained by performing subtraction between plain text values and its nearest prime values. Several stages are involved in encryption and decryption also 2's compliment, arithmetic and logical operations are used. Our proposal is similar to the work of Kiran et al. because both approaches use prime numbers to generate key, but differ in the method used to generate the prime during implementation.

## Existing Work

Keeping data secret required keeping the key secret, as such, key generation algorithm should be efficient that is why the proposed system improved the performance of the key generation algorithm of Kiran *et al*, by using Sieve of Eratosthenes algorithm to sieve out prime numbers first then placed them in a new array data structure and from the new array the next nearest prime of a corresponding value is retrieved

## Key Generation Process

Figure 3.1 show the steps followed in key generation of the designed system

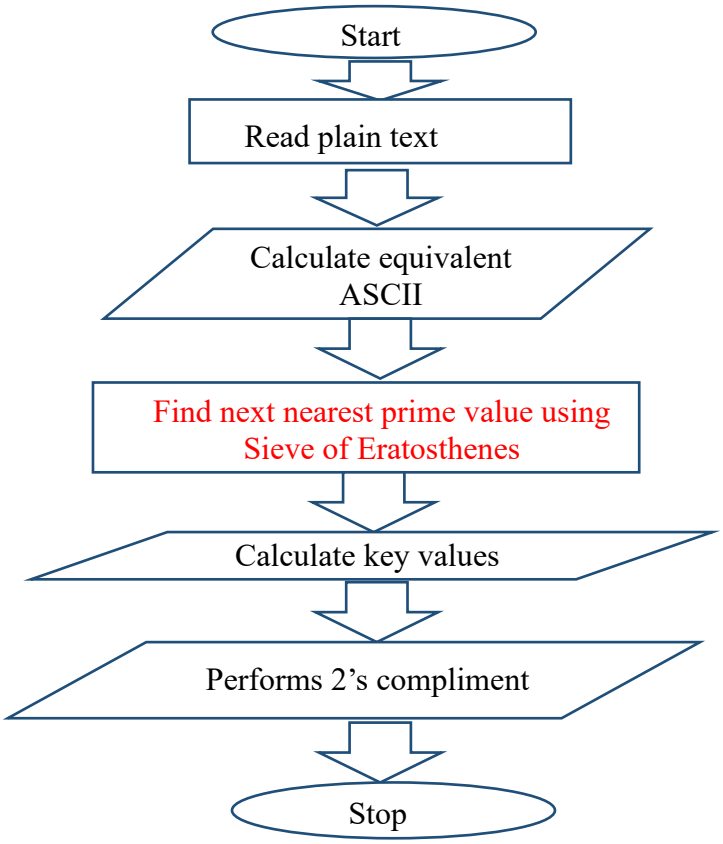


**Figure 3.1:** Key Generation Process

The Key Generation Algorithms steps are as follows:-

Step1: Convert plain text into its equivalent ASCII values.

Step2: Find next nearest prime values for ASCII values using Sieve of Eratosthenes algorithm.

Step3: The key values are obtained from finding difference between ASCII and next nearest prime values.
Step4: Perform 2's complement for generated value.

## ENCRYPTION PROCESS

In the encryption process, Exclusive OR operation is performed on each corresponding bit of the next nearest prime and the key generated, then the result of the Exclusive OR operation is reversed, now the result of the reversed operation is stored in a matrix form Column wise, lastly retrieve values from the matrix and find their equivalent ASCII value which become the cipher text. The following are the encryption process performed in rounds:-
Round1: perform XOR operation between next nearest prime and key.
Round2: reverse the result of round1.
Round3: store it in matrix form (column wise).
Round4: retrieve values from matrix. Find equivalent ASCII and consider to be as final cipher text.
The images below explained each step taken in each round during Encryption.

Round 1:

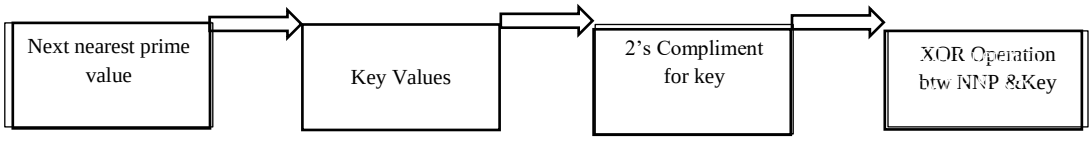


**Figure 3.2:** Steps Involved in Round 1

Round 2:

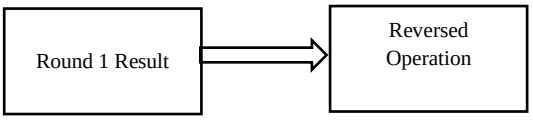


Figure 3.3: Steps Involved in Round 2

Round 3:

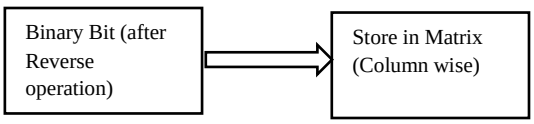

**Figure 3.4:** Steps Involved in Round 3

Round 4:

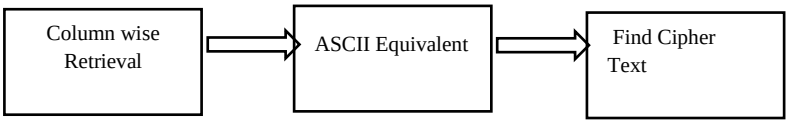


**Figure 3.5:** Steps Involved in Round 4

**Decryption Process**

In the decryption process, the equivalent ASCII values of the cipher text are converted into Binary equivalent, then those bits are rearranged in a matrix form row wise and reversed operation is performed on them. Exclusive OR operation is performed on the reversed values and the key to find the next nearest prime. Lastly find the difference between the next nearest prime values and key and get the equivalent ASCII values of the plaintext. The following are the decryption process performed in rounds: -

Round 1: Rearrange the cipher text information in the form of matrix by finding their binary equivalents.

Round 2: Retrieve row wise binary values and reverse them.

Round 3: Reversed values are XOR with key to find next nearest prime values.

Round 4: Find the original value by subtracting with key value from next nearest prime value.

Figure 3.6 to 3.9 below explained each step taken in each round during decryption.

Round 1:

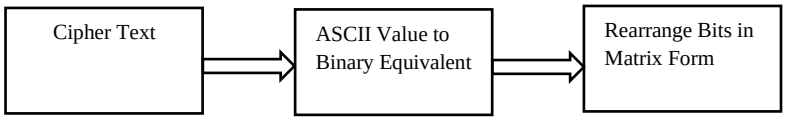


**Figure 3.6:** Steps Involved in Round 1

Round 2:

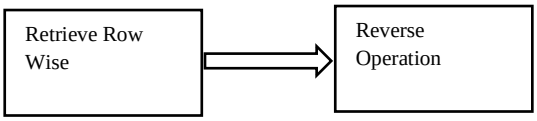


**Figure 3.7**: Steps Involved in Round 2

Round 3:

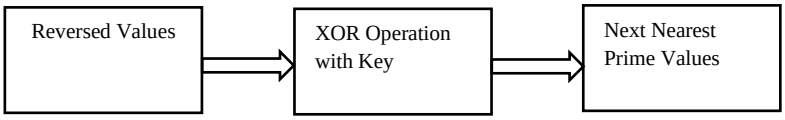

**Figure 3.8:** Steps Involved in Round 3

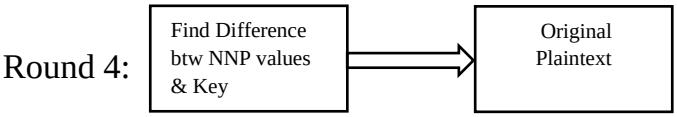


**Figure 3.9:** Steps Involved in Round 4

**Description of the Enhanced Encryption Using Next Nearest Prime Key Generation Algorithm**

The key is generated by sieving out composite numbers from the given plaintext's ASCII values using Sieve of Eratosthenes algorithm, then a new array is provided that will mark index as prime value and zero otherwise, thus the next nearest prime is return if the content of the array is not zero

## RESULT ANALYSIS

### Graphical User Interface System

The proposed system has the following functions: Measured performance section which include number of words, number of characters, size in bytes, key generation time and encryption execution time of proposed and existing system respectively. Plaintext Area, Cipher Text Area, Encryption and Decryption Buttons, Clear Button, and the Quit Button. Figure 4.1 shows the graphical user interface of the designed system.


**Figure 4.10:** Graphical User Interface of Designed System

### Key Generation and Plaintext Encryption Process

During the encryption stage, the user enter the plain text in the plain text area, then press the encrypt button in order to encrypt the plain text message to get

the cipher text. Thus from the plaintext the key is generated (one-time pad cryptography) the size of plaintext is the same as the size of key, i.e. the plaintext character is exclusive or bitwise with the key.

The following figures show the data encryption process: Figure 4.2 Shows the encryption mode and result of key generation time and encryption execution time of both existing and proposed
method.


**Figure4.2:** Encryption Mode

Cipher Text Decryption Process During the decryption stage, the user presses the decrypt button and the cipher text (i.e. result of the encrypted plain text) in the cipher text area, will now be decrypted to get the original plain text. Figure 4.3 shows the cipher text to be decrypted.

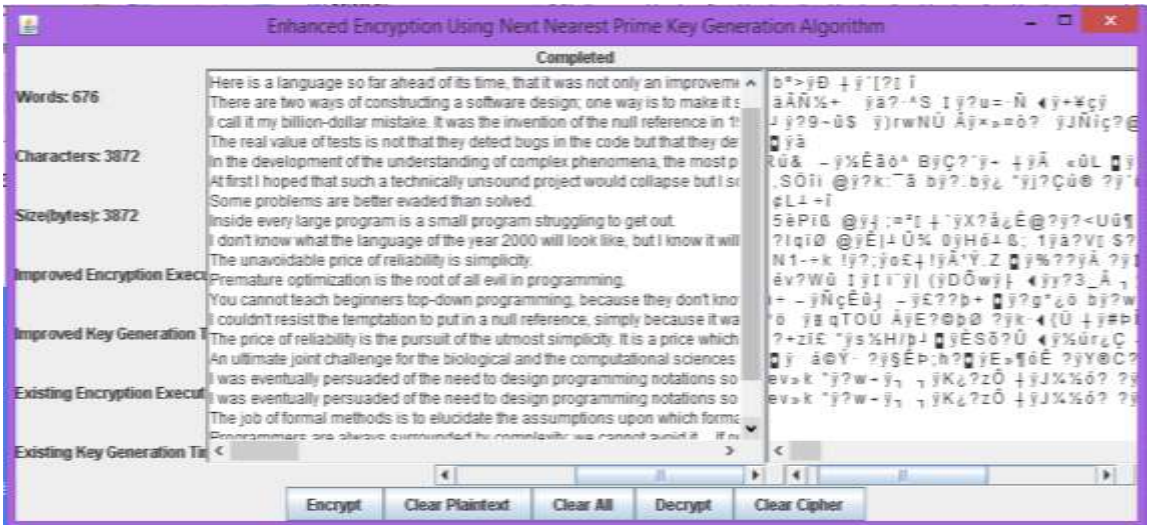


**Figure 4.3:** Cipher text

## Results and Discussion

During this stage, the comparison of the proposed key generation algorithm and the key generation algorithm implemented by (Kiran *et al*, 2016) were performed under the same platform. The performance of the two systems were tested with various samples of plaintext (standard data set) as shown in the Table 4.1 below and compared with the work of (Kiran *et al*, 2016) in line with the stated objectives of our research. The comparative performance was based on the following parameters: number of words, number of characters, size in bytes, key generation time and encryption execution time of proposed and existing system respectively. The results are presented in graph and table.

**4.1 Table 4.1:** Key Generation Time (Milliseconds)

| Plaintext | Key Generation Time of Existing Method (msec) | Key Generation Time of Proposed Method (msec) | % Gain |
|---|---|---|---|
| Sample1:(Dijkstra's Quote)<br>Word = 621<br>Character = 3738<br>Size(bytes) = 3738 | 78.2 | 48.6 | 38 |
| Sample 2: (Hoare's Quote)<br>Word = 676<br>Character = 3872<br>Size(bytes) = 3872 | 78.3 | 46.7 | 40 |
| Sample 3: (My Code)<br>Word = 4071<br>Character =18227<br>Size(bytes) =18227 | 360.9 | 250 | 31 |
| Sample 4: (Merged Samples)<br>Word = 5374<br>Character = 25843<br>Size(bytes) = 25843 | 493.9 | 328.3 | 34 |

Both implementations are run concurrently using each of the data size samples shown in Table 4.1, decrease in the key generation running time of the proposed algorithm suggested the improvement as compared with the algorithm implemented by (Kiran *et al*, 2016) by about 38%, 40%, 31% and 34% on 621 words, 676 words, 4389 and 5690 words respectively.

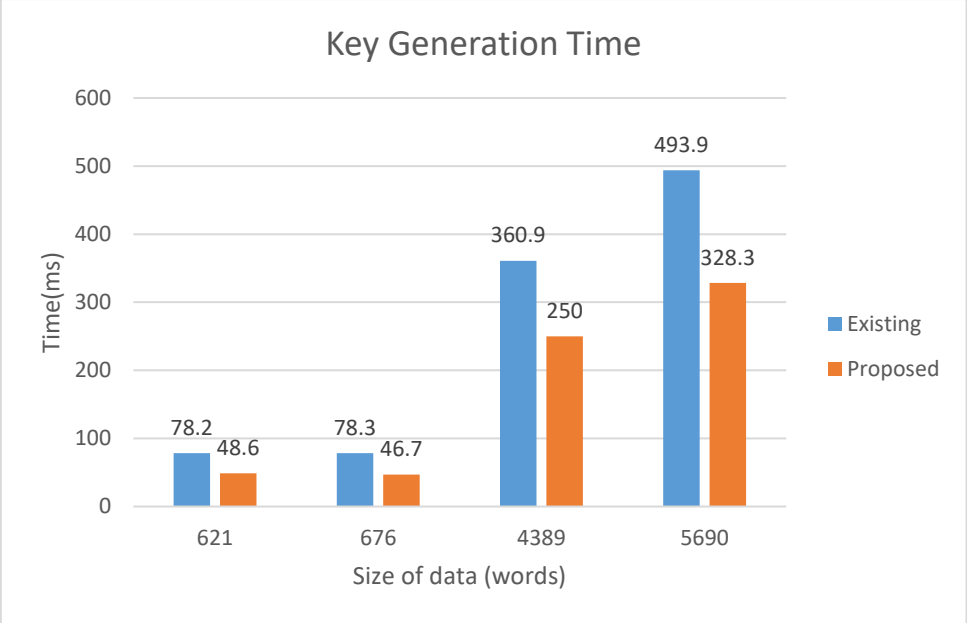


**Figure 4.4:** Comparison Based on Key Generation Time

From the analysis it is seen that the proposed method has a less key generation running time on different set of data when compared with the work of (Kiran *et al*, 2016) which has a higher key generation running time. This is because the design of the proposed next nearest prime algorithm used the sieve of Eratosthenes algorithm to sieve out the prime numbers first then placed those prime numbers in a new array data structure, and the next nearest primes of the corresponding ASCII values are retrieved from the array. Although not in every input that the proposed algorithm performs better, sometimes in smaller input (Kiran *et al*, 2016) algorithm performs better because the time it will take sieve algorithm to remove multiples of the composite numbers (Kiran *et al*, 2016) algorithm must outperform the proposed algorithm. In larger input the proposed algorithm always perform better since prime numbers are already sieved only for the algorithm to retrieve the next nearest prime from the array.

## SUMMARY

This dissertation proposed and implemented an enhanced Encryption using Next Nearest Prime in key generation algorithm, in this research work, the use of appropriate data structure was employed alongside with sieve of Eratosthenes algorithm for generating next nearest prime to improve the performance of the work of Kiran et al. (2016). The result of the implementation shows that there is a great improvement in the performance of key generation algorithm with about 38%, 40%, 31% and 34% on 621 words, 676 words, 4389 and 5690 words respectively of tested data.

Beside the achievement in improving performance of (Kiran *et al*, 2016). The research also shows that (Kiran *et al*, 2016) algorithm sometimes outperform the proposed algorithm in smaller input due to the reason that the proposed algorithm waste time in eliminating composite numbers by removing their multiples starting from the smallest number 2.

## CONCLUSION

The new system produces a better algorithm for finding the next nearest prime during key generation, the algorithm was successfully implemented in Java language and Netbeans IDE. A comparison analysis carried out between the existing system and the proposed system shows that the proposed gives a better performance in measuring time complexity.

## RECOMMENDATION

Further studies can be conducted as a compliment to this study or as a follow up by improving on the algorithm to reduce space complexity. Supplementing authentication security service to the nearest prime Encryption algorithm. Also, the algorithm can be improved to encrypt and decrypt Unicode characters.

## REFERENCES

Acharya, K., Sajwan, M., & Bhargava, S. ( 2014). Analysis of Cryptographic Algorithms for Network Security . *International Journal of Computer Applications Technology and Research*, Volume 3– Issue 2, 130 - 135, .

Aized, A. S., Irfan, R., & Rasheed, U. (2016). An Enhanced Veginere Cipher for Data Security. *International Journal for Scientific & Technology Research*. 5(3):141-145.

Ajit, D., Manjula, G. R., & Najak, R. (2013). Data Encryption by Excluding Repetitive Character in Cipher. *International journal of Innovation in Engineering and Technology*., 3(2) 49-53.

Andrew Green, M. W. (2012). *Guide to Network Security.* USA: Cengage Learning. edition?

Asiya, A., & Ruba, M. (2015). Vertically Scrambled Caesar Cipher Method. *International Journal of Computer Applications*. 118(21):9-13

Bazith, M. B. (2013). Automatic Key Generation of Ceasar Cipher. *International Journal of Engineering Trends and Technology*, 6(6):337-229

Devi, S., & Palanisamy, V. (August 2014). Multi Level Encryption using SDES Key Generation Technique with Genetic Algorithm. *International Journal of Engineering and Computer Science*, 3(8): 7596-7600.

Elaine, B. A. (2012). *Recommendation for Cryptographic Key Generation* . United State of America: NIST Special Publication 800-1

Kiran, S., Predeep, R., VenKata, J., & Naga, D. (2015). Multi Stage Encryption using Seeded SDES. *International Journal for Advance Networking and Application*, 7(2):2694-2699.

Kiran, S., Subramanyan, N., Suna, Y., & Haripriya, K. (2016). Next Nearest Prime Cipher for Data Confidentiality and Integrity. *International Journal for Advance Networking and Application*, 7(6):2944-2948.

Levitin, A. (2012). *Introduction to the Design and Analysis of Algorithms.* NewJersy: Pearson Education, Inc ISBN 13:978-0-13-231681-1.

LI, D., & WANG, Y. (2012). An Optimazation Algorithm for RSA Key Generation in Embedded System. *Journal of Theoretical and Applied Information Technology* , pg84-87 ISSN: 1992-8645 .

Meena, M., & Komathi, A. (2015). A Study and Comparative Analysis of Cryptographic Algorithms for Various File Formats . *International Journal of Science and Research (IJSR)*, 991-996.

Michael, A. B., Rezaul, C., & Conway, A. e. (2016). The I/O Complexity of Computing Prime Tables. *Latin American Theoretical Infor-matics Symposium*, pp.192-206, 2016, LNCS.

Neha, T. B., Shalender, G., & Sangeeta, D. (2016). Analysis of Various Crypyographic Technicques. *International Journal for Security and Application*, 4(4), pg59-92.

Saylor. (2012). *Prime numbers*. Retrieved on 15 April 2021 from Saylor.org: https://www.saylor.org/site/wp-content/uploads/2012/06/MA111_Wikipedia_Prime-Number_6.8.2012.pdf

Srinkantasamy, S. G., & Phaneendra, H. D. (2011). Enhanced One Time Pad Cipher with more Arithmetic and Logical Oerations with flexible Key Generation Algorithm. *International Journal of Network Security and Applications (IJNSA)*, 3(6):243-248.

Stallings, W. (2010). *Cryptography and Network Security Principles and practices ThirdEdition.* Pearson Education.

Zeaniah, B. E. (2016). An Analysis of Encryption and Decryption Application by using One Time Pad Algorithm. *International Journal of Advanced Computer Science and Applications,* , 6(9):292-298.