Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 26 (2023), No. 2, pp. 479–505
DOI : 10.1080/09720529.2021.1968573

Varibox encryption algorithm : The new generation of hybrid security measure for the era of quantum computation

Tuhin Karmakar [†]
Department of Computer Science & Engineering
Meghnad Saha Institute of Technology
Nazirabad Rd, Uchhepota
Kolkata 700150
West Bengal
India

Saptarshi Biswas [*]
Department of Computer Science
Iowa State University
Atanasoff Hall, 2434 Osborn Dr
Ames, IA 50011
U.S.A.

Indrajit Das [§]
Department of Information Technology
Meghnad Saha Institute of Technology
Nazirabad Rd, Uchhepota
Kolkata 700150
West Bengal
India

Subhrapratim Nath [‡]
Department of Computer Science & Engineering
Jadavpur University
188, Raja S.C. Mallick Rd
Kolkata 700032
West Bengal
India

[†]*E-mail:* tuhinkarmakar3882@gmail.com
[*]*E-mail:* saptarshi.biswas9@gmail.com (Corresponding Author)
[§]*E-mail:* indrajitdas1979@hotmail.com
[‡]*E-mail:* suvro.n@gmail.com

Abstract

The continual proliferation and evolution of the Internet of Things (IoT) domain is posing serious security threats to the whole network. In IoT, the insecure communication channel between sensors and the Internet can lead to malicious penetrations. Poor computational resources in traditional sensors critically constraint and impact the execution of conventional security algorithms among IoT devices. Consequently, lightweight security algorithms turn out to be a feasible alternative. To keep up with the existent Internet security level, this work proposes a new security scheme that employs a unique methodology based on a one-way counter value hashing table i.e. VariBox. This scheme integrates modular arithmetic applied on Unicode UTF-16 encoding standards coupled with AES and RSA algorithms. The proposed methodology effectively offers i) low computational overhead i.e. overall processing time of 0.1 seconds (VariBox processing time 0.003 seconds) and ii) enhances the overall security of the IoT data transfer model. Once this model is employed, the overall spectrum of potential brute force solutions will be rendered computationally expensive and infeasible to be calculated within a finite time period.

Subject Classification: *00A69, 05-XX, 06F25, 11Axx, 68-XX, 11T71, 11Y40, 14K10.*

Keywords: *IoT, VariBox, AES, RSA, Hashing, Unicode UTF-16, Vigenère Cipher.*

Introduction

Internet of Things (IoT) has been one of the most promising technologies which have paced the digital revolution throughout the globe. Furthermore, Cisco's Internet Business Solutions Group has anticipated that the utilized amount of IoT gadgets will double (50 billion gadgets) by 2020 [1]. The domain of IoT has also found great application in security sectors of industrial personnel for health management and quick rescue [2] which have been proposed and verified [3]. IoT can build up a massive network for creating a connected automated system. Owing to the presence of huge traffic volume, IoT security has consistently remained a pain point of concern and was enlisted as one of the top five security threats in 2015 [4]. So more secure and resilient systems need to be designed that provide enhanced data security in the domain of IoT devices [5].

To overcome the poor security of sensitive information in the network database work has been done with the help of a secondary encryption algorithm based on NCA mapping was proposed for the user's information resource with the spatiotemporal chaotic system. By exploiting the attributes of sensitivity, scrambling and randomness, Hennon mapping was combined with the Logistic mapping in a dual chaotic system with respect to the initial value. The algorithm was then applied to generate the key in order to construct a double encryption of the plaintext. This system

[6] was further improved, and a new spatiotemporal chaotic system based on NCA mapping was constructed [7]. The scrambling of sensitive information was controlled by the NCA mapping while the change of diffusion process was manipulated by the spatiotemporal chaotic sequence to complete the encryption of a user's sensitive information resource in a network database.

The critical domain of IoT security has always demanded continuous attention from multiple researchers' worldwide. F. Amounas et. al. [8] proposed an elliptic curve based on Unicode representation that converted a given text message into an affine point (Pm) of the elliptic curve, over a finite field GF(p). The character was represented by its Unicode, which could be described by 2 digits and differentiated into two values. Further, addition and doubling operations were done with each value of the affine point on the elliptic curve. It made the system more secure and complicated. The use of Unicode resulted in improved system performance. On the contrary, B. Maram et. al. [9] highlighted the fact that most of the security algorithms used in IoT are related to ASCII text and very few of them were used to deal with Unicode characters. However, in the real world, it has been observed that Unicode encodings prove to be significant in low power security processing in IoT systems. As a result, a key-dependent dynamic based S- Box has been designed to handle all the Unicode characters from different languages. For encryption, the Unicode is taken initially as plain text and is converted into Unicode-32 from which its binary form was adapted. Then a 512-bit block is generated on which a randomly generated dynamic S-Box is being operated to carry out the encryption process. Comparative performance of the obtained result was performed between Hamming distance, balanced output, avalanche effect and security analysis of different algorithms. The avalanche effect of the proposed system i.e. Unicode data privacy and security (UDPS) was 73%, which is reportedly more than its relevant counterparts.

Similarly, S. Rajesh et. al. [10] mentioned that modern development in wireless technology could increase the number of devices in the IoT environment. However, the small symmetric encryption algorithm used in this proposal was proven to be one of the effective algorithms for low memory utilization with ease of simplified implementation (both hardware and software). This paper proposes the Novel Tiny Symmetric Encryption Algorithm (NTSA). It provides an enhanced security for the transfer of text files through the IoT network by introducing additional key confusions dynamically for each round of encryption. Furthermore, Tiny Encryption Algorithm (TEA) has lower memory utilization and

ease of implementation on both hardware and software scales. The experimentation was performed to analyze the avalanche effect and time complexity in the IoT environment, which showed that the proposed method was more secure than other algorithms, but avalanche effect and time complexity values did not come up to the expectations.

In this context, R. Das et. al. [11] proposed a combination of lightweight cryptography and steganography for secure information transfer between IoT devices. In the proposed algorithm of this paper, first, the information is being encrypted employing standard encryption algorithms like DES. It is followed by computing its digest using the MD5 algorithm. Then both the encrypted data and the digest is embedded within a randomly selected cover image using a newly proposed algorithm in this paper. This algorithm was a new steganography MSB – LSB algorithm which could embed a maximum of 2 message bits in a pixel with the degree of distortion being 1 that enhanced the security of the IoT data transfer model with relatively low computational complexity. A comparative study was made with the proposed MSB-LSB substitution schemes and a simple LSB substitution. It was observed that the MSB-LSB substitution scheme performed better than the simple LSB substitution counterpart in most cases. Later, R. Das et. al. [12] devised another similar IoT data transfer model employing an integration of steganography algorithm i.e. variable LSB substitution and lightweight cryptography in the IoT environment. To incorporate randomness and better security, a hashing algorithm was proposed where randomly pixel values of the cover image were mapped to different target pixel values through a one-way hash function.

On the other hand, concentrating on the preexisting encryption algorithms that have been used for years to provide data security between communicating computing systems brought forward the dominating essence of symmetric block cipher [13] and the public key cryptographic system. One of the most used symmetric ciphers is Advanced Encryption System (AES) [14] and that of the public key cryptographic cipher is RSA [15]. In recent days, the AES has gone through numerous modifications and variations [16], however, the standard AES-256 has been utilised in this proposed work along with the RSA.

In this regard, it was found that the traditional security algorithms suffer from high computational overhead and are rendered unsuitable for IoT devices (for example sensors) because of their constrained resources (low memory and computation power etc.) [17]. Furthermore, the introduction of highly efficient computing devices like quantum computational systems [18] in the recent era of technological revolution

poses a high threat to all the established encryption algorithms in terms of security. To counter this problem, lightweight secure algorithms serve as feasible and inevitable solutions that are highly desirable for securing data transfer in IoT environments. Thus, to effectively combat the aforesaid issue, this paper proposes a security model dependent on one-way counter value hashing tables (VariBox). This scheme utilizes modular arithmetic applied on Unicode UTF-16 encoding which uses a dynamic mapping table that is created on-demand. The outlined scheme offers the following benefits: i) low overhead and power consumption (with minimum processing time) and ii) enhances the overall security of the IoT data transfer model.

The residual paper is structurally oriented as follows. Section 2 provides the preliminary theories related to this work. Section 3 demonstrates the proposed methodology that has been pursued. Section 4 constructs the mathematical foundation related to this work and elucidates the run time complexity and the brute force cryptanalysis. Section 5 outlines the detailed experimental results and their analysis. Finally, Section 6 draws the conclusion of this work.

Preliminaries

Advanced Encryption System (AES)

The Advanced Encryption System (AES) is an iterative block cipher method that principally relies upon two normal strategies for encryption and decryption [19]. These are normally known as substitution and permutation. The AES can manage at least 128 bit or 16 bytes as a fixed plaintext block size which can be presented as a 4x4 matrix where every unit presents 1 byte of the key. AES then works on the plaintext in blocks utilizing this key. There are 3 forms of AES in particular that are utilized to encrypt and decrypt the information. The numeric attribute in the terminology speaks to the size of the key in bits. The norms and key components with respect to various variants of the AES calculation are given in Table 1 whose summarization has been adjusted from [20].

There are three key processes of the AES algorithm which are enrolled as follows:

1. Encryption Process:

   The encryption technique of the AES algorithm depends on a factor named rounds as shown in Table 1, which generally depends on the

Table 1
Quantitative parameters related to different versions of the AES algorithm

Version Name	Number of Rounds	Size of Key (in bits)	Block Size of Plaintext (in bits)
AES-128	10	128	128
AES-192	12	192	
AES-256	14	256	

length of the key. Every one of these rounds includes four stages which are:

i. Substitution Byte Transformation: This stage relies on a nonlinear S-box to substitute a byte in one state with another byte of information. As per Shannon's diffusion and confusion principles for cryptographic algorithm design, it has essential roles which fortify the security of a framework.

ii. ShiftRows Transformation: The essential thought behind this progression is to shift bytes of states cyclically to the left in each row rather than row number zero. In this cycle, the bytes of row number zero remain and do not change. In the principal column, just a single byte is shifted circularly to the left. The subsequent row is moved two bytes to the left. The last line is moved three bytes to the left.

iii. MixColumns Transformation: Here, multiplication is performed on the state. In this stage, each row of matrix transformation must be multiplied with each column of the state. The results of this multiplication are used with XOR to produce four new bytes for the next state.

iv. AddRoundKey Transformation: It is one of the most essential stages in the AES calculation. The key and the information are organized in a 4x4 matrix of bytes. At that point, a bitwise XOR is performed between the current state with an information block and a predefined segment of the key in extended form.

2. Decryption Process:

The decryption process of an AES is nothing but the complement of the encryption where both the sender and receiver have the same key for encryption as well as decryption of the plaintext data. In

contrast, the final round of the decryption comprises three stages namely, InvShiftRows, InvSubBytes, and AddRoundKey.

3. AES Key Expansion:

   This is another vital step in the AES structure. Each round has a new key. The key expansion stage involves the creation of round keys for each word one by one where each word is composed of a four bytes array. It creates $4 \times (N + 1)$ words where N is the total number of rounds. The process is as follows: Initially, the cipher key is utilised to generate the first four words. The size of the key consists of 16 bytes (k_0 to k_{15}) that represent an array. The first four bytes i.e., from k_0 to k_3 represents w_0, the next four bytes i.e., from k_4 to k_7 in the first column represents w_1, and so on.

Vigenère Cipher

It was found that the traditional Caesar Cipher which used a shift of message characters for a certain distance in a circular fashion within the 26 English alphabetic system was inefficient [21]. It produced too few brute force combinations i.e., 26^n, where n is the length of the message, which was easily breakable by a computing system using a brute force attack. Furthermore, the greatest drawback of Caesar Cipher was that one could easily approximate the shift by analysing the most frequent letters occurring in the ciphertext and relating it to the frequently occurring English alphabets. The ciphertext C can be found using the following equation:

$$C = E(k, p) = (p + k) \bmod 26 \tag{1}$$

- where p is the plaintext character and k is the key character which is predefined.

  This drawback was later corrected through the Vigenère Cipher [22], which has gone through various modifications in recent days [23, 24], however, in this work the general Vigenère Cipher has been utilised. In contrast to Caesar Cipher, this method used a reference key text K of length m which is used to encrypt a plaintext M of length n such that $m < n$. The key K is superimposed on a circular basis upon the message character which is then shifted according to the general principle of Caesar Cipher to generate the ciphertext. This proves to be very efficient in restricting the frequency analysis of the ciphertext as different message characters can generate the

same cipher character. The general equation that the Vigenere Cipher follows is provided below:

$$C_i = (p_i + k_{i \bmod m}) \bmod 26 \tag{2}$$

- where C_i is the i[th] character of ciphertext, p_i is the i[th] character of the plaintext and $k_{i \bmod m}$ is the $(i \bmod m)^{\text{th}}$ character of the key superimposed on the plaintext.

RSA Public Key Encryption Technique

RSA is a popular public-key cryptography algorithm that is frequently used nowadays [25]. In this algorithm there exist two different keys namely, the public and the private key, which are related to each other in the views of both the sender and the receiver end. The public key is of the form {e, n} whereas the private key is of the form {d, n} where d is only known by the receiver, however, e is kept public. The message characters are encrypted using the following mathematical operation:

$$C = M^e \bmod n \tag{3}$$

On the other hand, the encrypted message is decrypted using the following equation:

$$M = C^d \bmod n \tag{4}$$

Here $n = p \times q$ where p and q are two prime numbers. The public number e is chosen such that $1 < e < \Phi(n)$ and $\gcd(\Phi(n), e) = 1$ where $\Phi(n)$ is the Euler's Totient function. On the other hand, d is the multiplicative inverse of e mod $\Phi(n)$. This can be expressed as follows:

$$d \equiv e^{-1} \bmod \Phi(n) \tag{5}$$

Prime Number Theorem (PNT)

The PNT [26, 27] states that the total number of prime numbers present within a search range can be approximately estimated through:

$$\pi(N) = \frac{N}{\log(N)} \tag{6}$$

where $\pi(N)$ is the prime counting function, N is the maximum number of search range and $\log(N)$ is the natural logarithm.

Proposed Methodology

The Varibox Encryptor is designed with 5 integrated layers of different encryption algorithms, especially for low power IoT systems. Three of these algorithms are the traditional AES-256, Vigenere Cipher and the RSA while the two newly introduced algorithms are Varibox Processing and Post-processing algorithms. Initially, the AES 256 bit is used to encrypt the raw plaintext message using the block encryption technique. This follows the Vigenere Cipher which is implemented to restrict the frequency analysis of the encrypted message. Then the encrypted message is further exposed to Varibox Encryptor where one layer of encryption is performed in the processing which is followed by a second layer of post-processing encryption. All the important information of the variables used to encrypt a message through all the steps is then encapsulated as a tuple along with the encrypted message which is further passed through an RSA encryption system before transmitting the message publicly.

For sending the information (M) from one IoT device to another, the following proposed methodology was used in this work.

Sender Side

For sending the message (M) from sender to receiver, the proposed methodology is shown below in Fig.1.

Step 1 : The message (M) is passed through a layer of AES Encryption with a key K_1 (256 bit) and an encrypted message M_1 is obtained. This M_1 is now ready to be passed through a VariBox.

The VariBox is a one-way hash function, but instead of creating unique digests, it works as a Many-to-One mapping function, i.e. any value passed through that VariBox, will output a long string consisting of a single unique repeating character. It is a set of dynamic operations, which generates a mapping table based on

TUPLE <Total Unique Characters in the Set, Adder, Modular, Destination Character, Key (K_2)>

where **Total Unique Characters in the Set** is the number of unique characters in a particular language. For example, in ASCII there are a total of 2^7 or 128 values. An **Adder (A)** is an odd number. **Modular (M)** is any number that is greater than the Unicode Value of Destination Character. **Destination Character (D)** is any character from the character set in a particular language. Key (K_2) is an alphanumeric string that is added to the message before performing varibox operations.

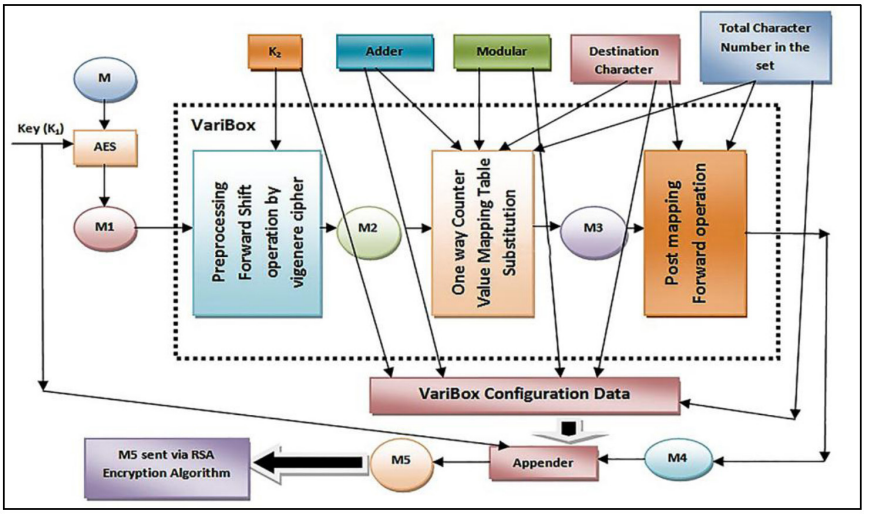

Figure 1

Sender site encryption method

Step 2 : VariBox operation (Encryption)

a) Preprocessing shift operation: The VariBox takes this M_1 and transforms it into M_2 by shifting the M_1 using the key (K_2) with Vigenere cipher methodology, which is used for eliminating frequency analysis.

b) One way Counter Value mapping table Substitution: The M_2 (from step 2. a) is then converted into M_3 which is generated by mapping M_2 using **Counter Value Mapping Table.** Here Adder, Modular, Destination Character, Total Character Number in the set is used.

c) Post mapping operation: Each character M_3 (from part 2. b) is shifted again by the difference of Unicode Value of Destination Character and Unicode Value of Message Character M_3 resulting in the output M_4. The post mapping procedure is described in Fig. 2 below.

Step 3 : An adding circuit is used to concatenate the M_4 along with the K_1, K_2 and the VariBox configuration data. Let the cumulative output be called M_5.

$$M_5 = (M_4, K_1, K_2, \text{VariBox Configuration data}) \quad (7)$$

Step 4 : M_5 is then finally transmitted via the RSA Encryption Scheme.

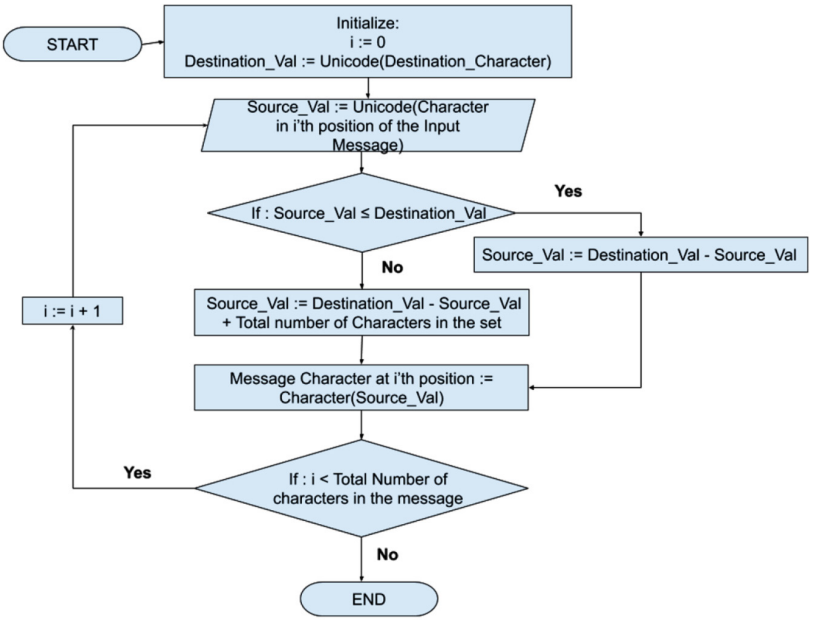

Figure 2

Post Mapping Operation

Receiver Side

After receiving the encrypted message (M_5) the following steps are required, as shown in Fig. 3.

Step 1 : M_5 is decrypted first using conventional RSA decryption procedure and then it passes the data through a splitter and parser section which gives M_4 along with the K_1, K_2 and VariBox Configuration data. The output of splitting M_5 is given below.

$$M_5 = (M_4, K_1, K_2, \text{VariBox Configuration data}) \quad (8)$$

Step 2 : VariBox operation (Decryption)

a) Post mapping operation: The VariBox Configuration Data is then loaded and M_4 is passed through it. Each character in the message is reverse shifted by the positional difference between the Unicode Value of Message character and Unicode Value of Destination Character. This generates M_3. The proof of reversibility of this shift operation has been provided in the supplementary material.

b) **One way counter value mapping table substitution:** M_3 is then mapped back to the M_2 using the **Counter Value Mapping Table** generated by VariBox. This outputs M_2. Here Adder, Modular, Destination Character, Total Character Number in the set is used.

c) **Preprocessing shift operation:** M_2 is then again reverse shifted using the key K_2 and generates M_1.

Step 3 : Finally, M_1 is decrypted using AES and K_1 (256 bit) which retrieves the original message M.

VariBox Counter Value Mapping Table Generation Algorithm

Fig 4. shows the procedure for the counter value mapping table generating algorithm used in VariBox. The algorithm steps are enlisted below:

Step 1 : Take the message M_2.

Step 2 : Split the message M_2, into its individual characters.

Step 3 : Take the Destination Character's Unicode Point Representation.

Step 4 : Set Current Char Position = 1 and Counter = 0

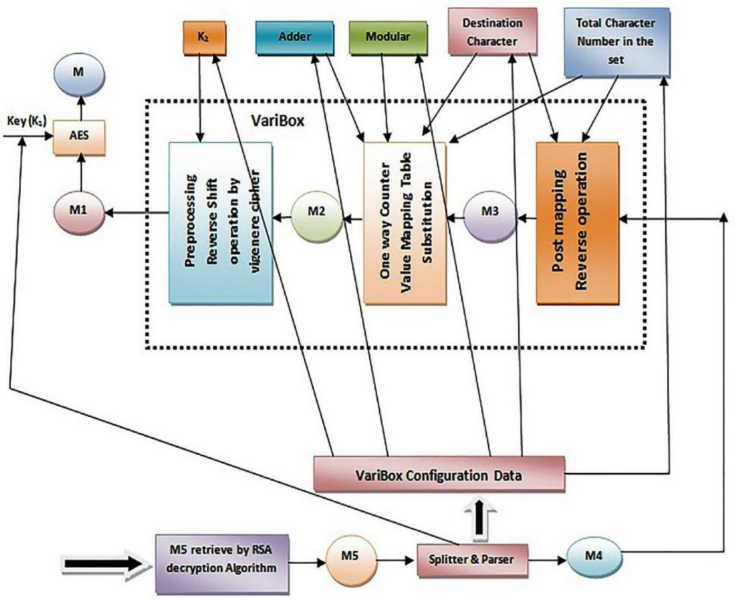

Figure 3

Receiver site Decryption method

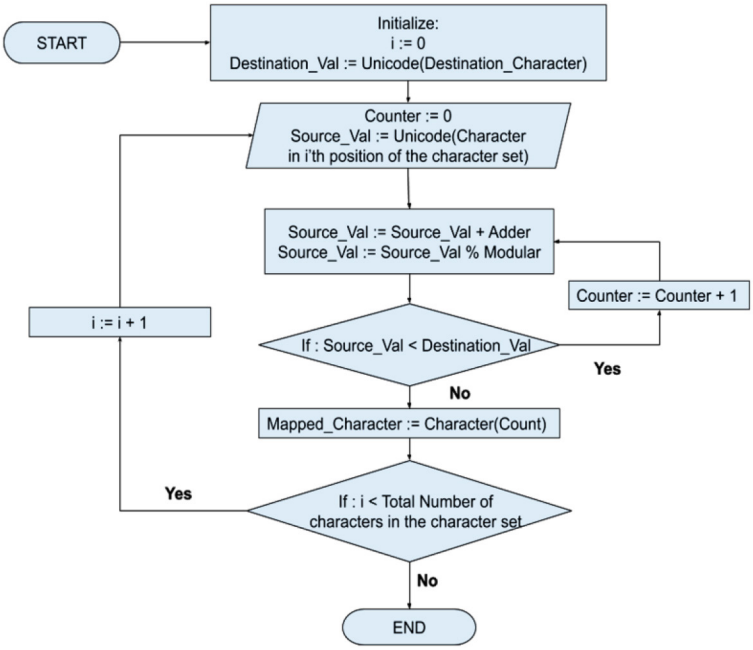

Figure 4

Counter Value Mapping Table Generation

Step 5 : Repeat Step Numbers 6 to 10 until the Current_Char_Position > Total Characters_Set

Step 6 : Take Src_Val = Character_Set [Current_Char_ Position]'s Unicode Point Representation.

Step 7 : Set Src_Val = Src_Val + Adder

Step 8 : Set Src_Val = Src_Val mod Modular.

Step 9 : if
Value of Step 8 = Value of Step 3
then
go to Step 10
else
go to step 7 and increment Counter by 1.

Step 10 : Map the Character_Set [Current_Char_Position] to the Counter.

Step 11 : Increment the Current_Char_Position.

Step 12 : Initialize the Counter to 0 and go back to step 5

Step 13 :

Encryption:

After the generation of the mapping tables, the input message, M_2, is passed through it character by character and each character gets replaced by its corresponding mapped characters.

Decryption:

After the generation of the mapping tables, the input M_3 is passed through it character by character and each character gets replaced by its corresponding mapped characters. This outputs M_2.

Mathematical foundation of Varibox Encryption Technique

This section provides a brief discussion about the complexities, constraints and cryptanalysis of the varibox encryption technique and provides the mathematical foundation behind them. A detailed description of the mathematical analysis of individual fields and lemmas is provided in the supplementary material. In this context, the mathematical statements use some denotations of the variables used which are described as follows:

(1) Unicode of the Source Character: S
(2) Unicode of the Destination Character: D
(3) Ciphertext Message Character: C
(4) Adder: A
(5) Modular: M
(6) Total number of characters in the character set: T

Complexity Analysis

Since the complete procedure of Varibox encryption percolates through three fundamental predefined standard encryption algorithms namely, the AES-256 bit, Vigenere Cipher and the RSA public-key cryptography algorithm, the time complexity of individual algorithms will have a defined effect on the overall computational performance of

the designed algorithm. The time complexities of these algorithms can be summarized as follows:

AES-256 bit

The AES-256 bit encryption procedure consists of 13 rounds of all four functions and the last round of 3 functions that excludes the AddRoundKey operation. Each of the functions namely, Substitution Byte Transformation, ShiftRows Transformation, MixColumns Transformation and AddRoundKey Transformation, exhibits O (1) operations individually. Taking an approximated net value for all the 14 rounds the time complexity is found to be O(1).

Assuming the total length of the plaintext to be p, where $\frac{p}{128} = n : n \in N$, it can be inferred that the overall time complexity of the complete AES-256 encryption process is O(n).

Vigenère Cipher

This algorithm needs one complete run over the plaintext of length n to convert it to the ciphertext. As a result, the time complexity is found to be $O(n)$.

RSA

The RSA needs iterating over the complete plaintext of length m once raising each plaintext character to its exponent value of e. The most efficient algorithm for raising a number to some power has a time complexity of $O(\log n)$. This ultimately culminates in the resultant time complexity of $O(n \log(n))$.

However, the Varibox encryption mechanism has 2 new procedures namely, the processing and the post-processing mechanism.

Processing

It is assumed that the message consists of m number of characters. Each character is encrypted by iterating through the operation i.e., $S := (S + A) \bmod M$, unless S gets descended to D. This complete process for each character needs n iterations to descend from S to D. This lets to the formulation of the complexity, is $O(m \times n)$ which finally leads to $O(n^2)$.

Post Processing

The post-processing requires an iteration over all the message characters of length n. This generates the complexity to be $O(n)$.

Overall Time Complexity

To wrap up, the overall time complexity is found to be:

$$O(n)+O(n)+O(n\ \log(n))+O(n^2)+O(n)=O(n^2) \tag{9}$$

Lemmas followed by Varibox Encryptor and Constraints for Recursion Chain Termination

Lemma 1 : *The Adder A needs to be a positive odd number, i.e.* $A \equiv 1\ (mod\ 2) and\ A > 0$

Proof : The Adder A is chosen to be odd only because the Adder is added to the Unicode of either the source or any intermediate character. As a result, the addition can generate either an odd value or an even value in an alternating sequence based on the property of the addition of two integers. Due to this property, the Adder is chosen to be an odd value so that it can generate all the numbers i.e. the even as well as the odd numbers, for the source character S to reach to the destination character D. Furthermore, the Adder is taken to be a positive number because the encoding of all the Unicode characters is numerically represented as whole numbers. Thus, to prevent the newly generated intermediate character from going into the area of negative integers, the Adder is taken to be positive to provide an additive phenomenon for generating the intermediate Unicode characters.

Lemma 2 : *The Modular M needs to be greater than the Destination Character D, i.e.,* $M > D$

Proof : The Modular M is used to trim the additive number formed either by the addition of the Source and the Adder or by the addition of the intermediate value and the Adder to its remainder. However, the prime condition of termination of the encryption logic depends on the fact that one intermediate node needs to be generated as Destination Character D. In this context, M is taken to be greater than D as according to the rule of modular arithmetic and the divisibility rule:

$$\begin{aligned} &if\ x = nM + r\ \text{then},\\ &\quad x\ mod\ M = r \end{aligned} \tag{10}$$

where r is the remainder when x is divided by M. Here, it can be concluded that $r < M$. For the recursion chain of the encryption algorithm to terminate $r = D$.

Lemma 3 : *The Adder A should not be perfectly divisible by M, i.e.* $M \nmid A or A \ncong 0 \ (mod\, M)$

Proof : This lemma has been provided with regards to elucidating a boundary condition due to which the encryption algorithm might enter into an infinite looping stage without terminating. This condition occurs when either the value of a source or an intermediate character after being added with an Adder A and then divided with a Modular M generates the same character in return. This condition can be given as the following equation:

$$(C + A) mod\, M = C \text{ where } C \text{ is the Unicode value of a character} \qquad (11)$$

Solving the above equation gives rise to a solution: $n = \frac{A}{M}$

This condition will arise only when $n \in W$ where W represents the whole number. Thus, to avoid this condition, the only possible way is to maintain $n : n \in R^+$ and $n \notin W$ where R^+ is the positive rational number which can be only achieved if $M \nmid A \text{or} A \ncong 0 \ (mod\, M)$.

Brute Force Cryptanalysis

The brute force cryptanalysis has been analysed for the complete encryption technique. This has been provided according to the respective module as follows:

AES

The AES used in the pre-processing step is of a 256-bit key. In this context, the maximum check needed by a brute force attack is 1.158×10^{77}.

Vigenère Cipher

On the other hand, the Vigenère Cipher was also used in the preprocessing stage to restrict the frequency analysis of the message. This contributes a combination of $65536^n - 1$ to the total brute force attack combinations, where n is the length of the plaintext.

Varibox Processing

As it is known that the encrypted character C of the ciphertext depends on the source character S, destination character D, adder A and the modular M. This, in turn, reflects the combinatorial problem about the total number of possible combinations P of these variables that an attacker needs to check in order to perform a brute force attack. Considering the UTF-16 Unicode standard on a standard 64-bit system it was found that the total number of characters that could be encoded can be 65536 different characters.

It was further noticed that although the value of A and M can be theoretically deduced as a positive infinite, however, in the actual scenario the maximum numerical value that can be held by a 64-bit system is 1.8×10^{308}. On the other hand, both the S and D can be chosen as any of the characters within the complete character set of the UTF-16 which leads to the total number of possibilities to be 65536 for each of S and D. This leads to a formulation of the net combinatorial possibilities of the processing step to be 6.957×10^{625}.

Varibox Post Processing

In the post-processing step, it was analysed that on an average case scenario where it was assumed that the encrypted data found in the processing stage is uniformly divided on both the sides of the destination character D. This resulted in a probability distribution of 0.5 to be assigned for each decision branch that is taken in the post-processing stage for the final layer of message conversion within the Varibox encryptor.

In this regard, according to the post-processing encryption algorithm, the attacker needs to trace out the destination character D from among all the 65536 characters if the ciphertext message character C is found to be lesser than or equals to D. On the other hand, if the C is found to be greater than D then the attacker needs to find out not only the D but also the total number of characters in the character set T. This finally formulated in a total number of possibilities to be $32768.5 + \sum_{32768}^{32768} C_i^{65536}$.

RSA

A brute force attack is a generic attack that can be performed on an RSA cryptosystem to find the private key (n, p, d). Brute force attack on RSA can be classified into two categories: padding scheme based and private key based. Padding scheme based brute force attack requires a

large amount of memory to build a table listing all possible plaintext, message value, and its corresponding ciphertext. Private key focused brute force attacks can be further classified into two types: p, q-oriented and d-oriented. In a d-oriented attack, the major problem is the ability to identify the message decrypted is exactly the same as the message sent from the sender, since the RSA cryptosystem is mainly used to encrypt the key of other cryptosystems, such as AES encryption. p and q oriented brute force attack is more focused on integer factorization that tries to factorize from the n value in the public key. When one of the p or q values is found, the whole private key will have to be revealed. This analysis finally led to the formulation of 1.88×10^{302} possible brute force combinations.

Overall Analysis

Wrapping up the complete analysis of the individual stages it can be summarized that the net combinations of the brute force attack are the product of all the combinations of every stage. This finally formulates a value of:

$$Total\ Possibilities = 15.146 \times (65536^{n} - 1) \times \left(32768.5 + \sum_{i=1}^{32768} C_{i}^{65536}\right) \times 10^{1004} \quad (12)$$

- where *n* represents the total length of the message characters.

Experimental Results

The experiment was performed using Python IDE 3.6.4 interpreter on Windows 8.1 operating system running on AMD A6-9200 Dual-Core @ 2 GHz with 8GB RAM. All the results were plotted and visualized with the help of the plotting library of Python (Matplotlib).

Assume Message (M) is "**Attack in the Night**"

Encryption and Decryption process

Step 1 : Using AES-256 & Our Secret Key (K_1): GDJTO37578309!@$(&%^)&*(&392DKgi & Method = CBC

Output(M_1):

EXoSkPcJZD3xtiaD75shttnWMxz3khY2x/p78yx5sow=

Step 2 : Using VariBox, the Configuration is given below:

Total Character Number in the Set = 138

Adder = 47

Modular = 139

Destination Character = "@"

$\mathbf{K_2}$ = qgzhfbalcjhialplcjscuzas

Output(M_2):

E(@!%&^(>#s@gmz%fav_ab(DM(~s%~Hh?1n`wb$d&ndo

Step 3 : The M_2 is converted into M_3 using the VariBox Mapping Table.

Output (M_3):

>$\x00tn'^$\x03q\x84\x00\x0b\x024nR\x14:\x17\x14X$\x852$.\
x84n.\x7fOG\\F[~X*U'FU\x8a

Step 4 : Lastly, the M_3 is shifted to a string in reference to the Destination Character, i.e. "@" in the post mapping operation.

Output (M_4):

'\x02\x1c@NT\x19d\x1c=Q>@5>\x0cTp,\x06),j\x1c=\x0e\x1c\
x12>T\x12Cs{f|gDj\x16m\x19|m8'

Step 5 : The M_4, K_1, K_2 and VariBox Configuration are then appended using the adder circuit. This results in M_5.

Step 6 : The M_5 is then sent using RSA Encryption.

Step 7 : The receiver executes the aforesaid steps in reverse order including AES decryption.

VariBox Mapping Table

VariBox mapping table configuration (Table II) is given below for the above experiment having Adder value 47, Character Number in the set 138, Modular 139 and Destination Character @. Here the mapping of

138 characters is shown with the help of VariBox because the Character Number in the set should be always a minimum of 1 less than Modular. The Unicode characters used for this work use the UTF-16 encoding.

Table 2
Varibox Mapping Table

Char. No	Original Character	Counter Value	Encoded Character	Char. No	Original Character	Counter Value	Encoded Character
0	\x00′	96	` ′	69	‘E′	62	‘>′
1	‘\x01′	25	‘\x19′	70	‘F′	130	‘\x82′
2	‘\x02′	93	‘]′	71	‘G′	59	‘;′
3	‘\x03′	22	‘\x16′	72	‘H′	127	‘\x7f′
4	‘\x04′	90	‘Z′	73	‘I′	56	‘8′
5	‘\x05′	19	‘\x13′	74	‘J′	124	‘	′
6	‘\x06′	87	‘W′	75	‘K′	53	‘5′
7	‘\x07′	16	‘\x10′,	76	‘L′	121	‘y′
8	‘\x08′	84	‘T′	77	‘M′	50	‘2′
9	‘\t′	13	‘\r′	78	‘N′	118	‘v′
10	‘\n′	81	‘Q′	79	‘O′	47	‘/′
11	‘\x0b′	10	‘\n′	80	‘P′	115	‘s′
12	‘\x0c′	78	‘N′	81	‘Q′	44	‘,′
13	‘\r′	7	‘\x07′	82	‘R′	112	‘p′
14	‘\x0e′	75	‘K′	83	‘S′	41	‘)′
15	‘\x0f′	4	‘\x04′	84	‘T′	109	‘m′
16	‘\x10′	72	‘H′	85	‘U′	38	‘&′
17	‘\x11′	1	‘\x01′	86	‘V′	106	‘j′
18	‘\x12′	69	‘E′	87	‘W′	35	‘#′
19	‘\x13′	137	‘\x89′	88	‘X′	103	‘g′
20	‘\x14′	66	‘B′	89	‘Y′	32	‘ ′
21	‘\x15′	134	‘\x86′	90	‘Z′	100	‘d′
22	‘\x16′	63	‘?′	91	‘[′	29	‘\x1d′
23	‘\x17′	131	‘\x83′	92	‘\\′	97	‘a′
24	‘\x18′	60	‘<′	93	‘]′	26	‘\x1a′
25	‘\x19′	128	‘\x80′	94	‘^′	94	‘^′
26	‘\x1a′	57	‘9′	95	‘_′	23	‘\x17′
27	‘\x1b′	125	‘}′	96	‘`′	91	‘[′

Contd...

28	'\x1c'	54	'6'	97	'a'	20	'\x14'
29	'\x1d'	122	'z'	98	'b'	88	'X'
30	'\x1e'	51	'3'	99	'c'	17	'\x11'
31	'\x1f'	119	'w'	100	'd'	85	'U'
32	' '	48	'0'	101	'e'	14	'\x0e'
33	'!'	116	't'	102	'f'	82	'R'
34	'"'	45	'-'	103	'g'	11	'\x0b'
35	'#'	113	'q'	104	'h'	79	'O'
36	'$'	42	'*'	105	'i'	8	'\x08'
37	'%'	110	'n'	106	'j'	76	'L'
38	'&'	39	'"'	107	'k'	5	'\x05'
39	'"'	107	'k'	108	'l'	73	'I'
40	'('	36	'$'	109	'm'	2	'\x02'
41	')'	104	'h'	110	'n'	70	'F'
42	'*'	33	'!'	111	'o'	138	'\x8a'
43	'+'	101	'e'	112	'p'	67	'C'
44	','	30	'\x1e'	113	'q'	135	'\x87'
45	'-'	98	'b'	114	'r'	64	'@'
46	'.'	27	'\x1b'	115	's'	132	'\x84'
47	'/'	95	'_'	116	't'	61	'='
48	'0'	24	'\x18'	117	'u'	129	'\x81'
49	'1'	92	'\\'	118	'v'	58	':'
50	'2'	21	'\x15'	119	'w'	126	'~'
51	'3'	89	'Y'	120	'x'	55	'7'
52	'4'	18	'\x12'	121	'y'	123	'{'
53	'5'	86	'V'	122	'z'	52	'4'
54	'6'	15	'\x0f'	123	'{'	120	'x'
55	'7'	83	'S'	124	'|'	49	'1'
56	'8'	12	'\x0c'	125	'}'	117	'u'
57	'9'	80	'P'	126	'~'	46	'.'
58	':'	9	'\t'	127	'\x7f'	114	'r'
59	';'	77	'M'	128	'\x80'	43	'+'
60	'<'	6	'\x06'	129	'\x81'	111	'o'
61	'='	74	'J'	130	'\x82'	40	'('
62	'>'	3	'\x03'	131	'\x83'	108	'l'
63	'?'	71	'G'	132	'\x84'	37	'%'

Contd...

64	'@'	0	'\x00'	133	'\x85'	105	'i'
65	'A'	68	'D'	134	'\x86'	34	'"'
66	'B'	136	'\x88'	135	'\x87'	102	'f'
67	'C'	65	'A'	136	'\x88'	31	'\x1f'
68	'D'	133	'\x85'	137	'\x89'	99	'c'

Performance Analysis

Analysing the well-established symmetric and asymmetric algorithms, it was found that the Varibox encryption algorithm alone could provide stronger data protection with respect to the brute force attack when compared with the pre-existing algorithms. It was found that a symmetric encryption technique like AES-256 could produce a brute force combination at a range of 10^{77}. On the other hand, the asymmetric encryption algorithm, namely, RSA, provided a brute force combination on the scale of 10^{302}. In comparison to these the Varibox processing and postprocessing alone produced a brute force combination in the order of at least 10^{625} for a single encrypted character of the ciphertext. Furthermore, when this algorithm is integrated with AES-256 and Vigenère Cipher in the preprocessing stage and RSA in the last layer post-processing stage finally germinated to a net cryptanalytic brute force combination in the scale of at least 10^{1004}.

The total time required for conducting this experiment depends on the total character sets used in the experiment which is shown in Fig. 5 below. Here, the experiment was performed with 9000 characters and the maximum time required was found to be approximately 0.1 seconds. Fig. 6 defines the total time required for VariBox operation for 9000 characters which is around 0.003 seconds. The fluctuation of the value is primarily caused due to the internal input-output delay. This clearly depicts that the time required to use the proposed algorithm in the real-world scenario is extremely low. It further implies that the algorithm can be efficiently implemented in systems with very limited power supply and computational resources. These features make the Varibox encryption algorithm suitable for usage in the communication of low power IoT devices.

Conclusion

This proposed work thus offers a novel method of information encryption scheme for providing enhanced secure networks for IoT data

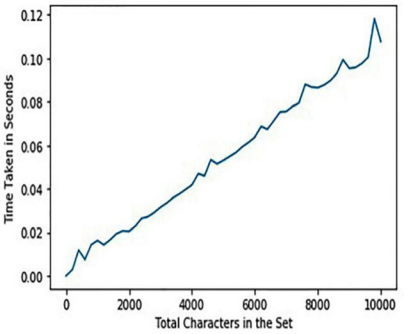

Figure 5

Total Execution time

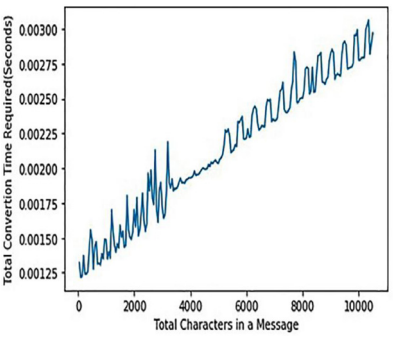

Figure 6

VariBox execution time

transfer models designed for over-air transmissions. This newly designed encryption methodology was implemented on a message of length of 9000 characters based on the UTF-16 Unicode encoding standard. The experimental results also showed promising results related to the computational time for executing such secure, lightweight algorithms in the real-time scenarios (which was found to be approximately 0.1 seconds and VariBox processing time of 0.003 seconds). The low execution time offers lower power consumption in computationally constrained IoT devices. It was analyzed that the change in even a single encryption parameter causes an exponential surge in the total number of possible brute force solutions, thereby making it potentially hard for the attackers and malicious minds to launch attacks in our devised IoT security model.

References

[1] J. Brandt, "50 billion connected IoT devices by 2020". Available: http://www.smartgridnews.com/story/50-billionconnected-iot-devices-2020/2015-04-21

[2] S. Rizvi, A. Kurtz, J. Pfeffer, M. Rizvi "Securing the Internet of Things (IoT): A Security Taxonomy for IoT", 17 th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/ 12th IEEE International Conference on Big Data Science and Engineering(TrustCom/ BigDataSE), Aug,2018, pp-163-168.

[3] S. Nath, A. Dey, P. Pachal, J. K. Sing and S. K. Sarkar, "Nano Structured Gas Sensing Device and Its Application in Underground Mines", *IEEE Electron Device Society Kolkata Conference,* 2018, pp. 445-449.

[4] S. Nath, A. Dey, P. Pachal, J. K. Sing and S. K. Sarkar, "Performance analysis of gas sensing device and corresponding IoT framework in mines", Microsystem Technologies: Micro- and Nanosystems Information Storage and Processing Systems, 2019, pp. 1-9.

[5] H. Garg, M. Dave, "Securing IoT Devices and Securely Connecting the Dots using REST API and Middleware", 4th International conference on Internet of Things: Smart Innovation and Usages (IoT – SIU), April, 2019 , pp. 1-6.

[6] B. Song, Q. Ding, "Comparison of Typical Discrete Logistic Map and Henon Map", Intelligent Data analysis and its Applications, Volume 1, *Advances in Intelligent Systems and Computing,* Volume: 297, 2014, pp. 267-275.

[7] B. Zhao, G. Guo, "An encryption algorithm algorithm for user's sensitive information resources in network database", *Journal of Interdisciplinary Mathematics*, Volume: 21 Issue: 5, 2018, pp. 1255-1260. doi: 10.1080/09720502.2018.1495399.

[8] F. Amounas, L.E. Bermi, "An Enhanced Approach of Elliptic Curve Cryptosystem based Unicode Representation", *International Journal on Recent and Innovation Trends in Computing and Communication,* Volume: 4 Issue: 12, Dec, 2016, pp-126 -131.

[9] B. Maram, J. Gnanasekar, G. Manogaran, B. Muthu, " Intelligent security algorithm for Unicode data privacy and security of IoT", *Service oriented computing and application,* Volume: 13 Issue: 1, Nov, 2018, pp. 3-15.

[10] S. Rajesh, V. Paul, V G. Menon, M R Khosravi, "A Secure and Efficient Lightweight Symmetric Encryption Scheme for Transfer of Text Files between Embedded IoT Devices", Symmetry, Volume: 11 Issue: 2, Feb, 2019, pp. 293.

[11] R. Das, I.Das, "Secure Data Transfer in IoT environment: adopting both Cryptography and Steganography techniques", 2nd International conference on Research in Computational Intelligence and Communication Networks (ICRCICN)", Sept, 2016.

[12] R. Das, P. Chatterjee, "Securing Data Transfer in IoT Employing an Integrated Approach of Cryptography & Steganography",

"International Conference on High Performance Compilation, Computing and Communications", March 2017, (pp. 17-22). ACM.

[13] C. Easttom, "A generalized methodology for designing non-linear elements in symmetric cryptographic primitives," 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, 2018, pp. 444-449, doi: 10.1109/CCWC.2018.8301643.

[14] Alamsyah, A. Bejo and T. B. Adji, "AES S-box construction using different irreducible polynomial and constant 8-bit vector," 2017 IEEE Conference on Dependable and Secure Computing, Taipei, 2017, pp. 366-369, doi: 10.1109/DESEC.2017.8073857.

[15] V. Shende, G. Sudi and M. Kulkarni, "Fast cryptanalysis of RSA encrypted data using a combination of mathematical and brute force attack in distributed computing environment," 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Chennai, 2017, pp. 2446-2449, doi: 10.1109/ICPCSI.2017.8392156.

[16] F. J. D'souza and D. Panchal, "Advanced encryption standard (AES) security enhancement using hybrid approach," 2017 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, 2017, pp. 647-652, doi: 10.1109/CCAA.2017.8229881.

[17] C. Easttom, "An Examination of Inefficiencies in Key Dependent Variations of the Rijndael S-Box," Electrical Engineering (ICEE), Iranian Conference on, Mashhad, 2018, pp. 1658-1663, doi: 10.1109/ICEE.2018.8472462.

[18] M. Möller and C. Vuik, "On the impact of quantum computing technology on future developments in high-performance scientific computing," Ethics and Information Technology, Vol. 19, pp. 253-269, Aug 2017.

[19] J. Daemen and V. Rijmen, The Design of Rijndael, Springer-Verlag, Berlin, Heidelberg, 2002. doi: http://doi.org/10.1007/978-3-662-04722-4

[20] S. More and R. Bansode, "Implementation of AES with Time Complexity Measurement for Various Input," *Global Journal of Computer Science and Technology: E Network, Web & Security,* Vol. 15, No. 4, pp. 11-20, 2015.

[21] W. Stallings, "Classical Encryption Techniques," Cryptography and Network Security Principles and Practice, Seventh Edition, Pearson India Education Services Pvt. Ltd, Noida, India, pp. 92-94, 2017.

[22] R. S. Kartha and V. Paul, "Survey: Recent Modifications in Vigenere Cipher," *IOSR Journal of Computer Engineering*, Vol. 16, No. 2, pp. 49-53, Mar 2014.

[23] B. Triandi, E. Ekadiansyah, R. Puspasari, L. T. Iwan and F. Rahmad, "Improve Security Algorithm Cryptography Vigenere Cipher Using Chaos Functions," 2018 6th International Conference on Cyber and IT Service Management (CITSM), Parapat, Indonesia, 2018, pp. 1-5, doi: 10.1109/CITSM.2018.8674376.

[24] Q-A Kester, "A cryptosystem based on Vigenère cipher with varying key," *International Journal of Advanced Research in Computer Engineering & Technology,* Vol. 1, No. 10, Dec 2012.

[25] R. L. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, Vol. 21, No. 2, pp. 120-126, Feb 1978. doi: https://doi.org/10.1145/359340.359342

[26] A. R. C. D. V. Gunasekara, A. A. C. A. Jayathilake and A. A. I. Perera, "Survey on Prime Numbers," *Elixir Applied Mathematics,* Vol. 88, pp. 36296-36301, Nov 2015.

[27] C. Cornaros and C. Dimitracopoulos, "The prime number theorem and fragments of PA," *Archive for Mathematical Logic*, Vol. 33, pp. 265-281, Aug 1994.

Received December, 2020

Revised May, 2021