Original Article

Invo-Substitute: Three Layer Encryption For Enhanced E-Commerce Website Security Using Substitution Cipher And Involution Function

Biresh Kumar[1], Sharmistha Roy[1], *Anurag Sinha[2], Vikas Kumar[3] , Ashish kumar verma[4]

[1]Faculty of Computing and Information Technology, Usha Martin University, Ranchi, Jharkhand, India
bireshmtech@gmail.com (B.K.), sharmistha@umu.ac.in (S.R.)

[2] Department of Computer Science, IGNOU, New delhi, India, anuragsinha257@gmail.com
[3]Department of CSE,
KIET GROUP OF INSTITUTION, kvikas1482@gmail.com
[4]Dr Rajendra Prasad kendriya vidyalaya Rashtrapati bhavan
imdigitalashish@gmail.com
*Corresponding Author:-Anurag Sinha, anuragsinha257@gmail.com
DOI: 10.47750/pnr.2023.14.S02.198

Abstract

The use of cryptography, or the art of secret writing, to secure information exchanged via an open medium is an ancient technique, but its significance in the modern world has greatly expanded. It has become necessary to search for ever-better cryptographic primitives that offer a level of security while limiting the time needed to encrypt and decode the data as eavesdroppers' computational power has increased. Each letter in the message is swapped out for a different letter using a substitution cipher that follows a predetermined mapping. A, B, and C are shown as an image of two alphabets with a mapping between them: D, E, and F. the Caesar cipher is an easy illustration of a substitution cipher. When more than one independent layer of encryption is used, it is known as double encryption and it serves as a safeguard against the compromising of any one layer. The dangers associated with encrypting data are reduced by using two levels of encryption. Concurrent Error Detection (CED) for cryptographic chips also offers a huge potential for detecting (intentional) fault injection attacks when errors are injected into a cryptographic chip to break the key. For a family of symmetric block ciphers whose round functions are involutions, we present in this study a temporal redundancy based CED approach that is low cost, low latency, and effective. Nearly little more time is required for this CED approach to identify both ongoing and passing errors. F(F(x))=x indicates that a function F is an involution.

Keywords : E-commerce, security, cipher, cryptography, substitution cipher, involution function

INTRODUCTION

Using the same or a different algorithm, multiple encryption is the process of encrypting a communication that has previously been encrypted once or more. Additionally, it is referred to as multiple encryption, superencipherment, cascade ciphering, and cascade encryption. Superencryption is the term used to describe a multiple encryption's outer-level encryption.

According to certain cryptographers, such as Matthew Green of Johns Hopkins University, multiple encryption addresses an issue that mostly doesn't exist: Ciphers used nowadays are seldom cracked... As opposed to being subjected to a devastating attack on AES, you are much more likely to be affected by malware or an implementation error. The root cause of multiple encryption, bad implementation, is shown in the quotation [1]. It is necessary for both suppliers' products to be compromised for using two distinct crypto modules and keying procedures from two separate vendors. If the keys used for any two cryptosystems are the same, the second cipher may be able to partially or completely decipher the first cipher. This is true for ciphers when the decoding

procedure is precisely the same as the encryption process; the second cipher would entirely undo the first. Assuming the same key is used for all levels, if an attacker were to discover the key through cryptanalysis of the first encryption layer, they could be able to decode all the subsequent layers as well. No cipher has ever been theoretically demonstrated to be impenetrable, with the exception of the one-time pad. The cipher texts produced by the first cipher may also exhibit certain repeating characteristics. Given that the second cipher uses those cipher texts as its plaintexts.

From the NSA's Commercial Solutions for Classified Program, the Rule of Two is a data security tenet (CSfC). [3] For the purpose of protecting data, it provides two totally distinct layers of cryptography. Data, for instance, may be safeguarded by both software and hardware encryption at several layers, starting with the lowest level of hardware encryption. Data encryption and decryption may include the use of two FIPS-validated software cryptomodules from various suppliers. In order to eliminate the chance that two manufacturers or models may share vulnerability, it is crucial for the layers of components to have a variety of vendors and/or models. In this manner, even if one component is broken, the data is still being protected by a whole layer of encryption, whether it is at rest or while being sent. For achieving diversity, the CSfC Program provides two options.

By replacing each letter or symbol in the plaintext with a new symbol as instructed by the key, substitution ciphers encrypt the plaintext. The Caesar cipher, which was employed by Caesar and is named after him, is possibly the most straightforward substitution cipher. The Captain Midnight Code-O-Graph and the hidden decoder rings that even came in Kix cereal boxes have helped modern readers become more familiar with the Caesar cipher [4]. It is acceptable to refer to the Caesar cipher as either a shift cipher or a mono-alphabetic cipher in order to distinguish it from other, trickier substitution ciphers.

The second issue to solve is that of interception, presuming that the devices and the connection are secure. Using packet logging, data can be purposefully intercepted or accidentally lost. In both scenarios, encryption is the next degree of protection. Only the intended recipient can decipher a communication that has been successfully encrypted. Since the encrypted message is encoded in the cover image's non-edge pixels, the edge pixels are used to produce the encryption keys. By using several bits in each of the three colour channels, the method's capacity is increased. Because lower bit-planes are not used during edge detection, the accurate extraction of target data is guaranteed. Even with a larger embedding capacity, the stego media quality is kept within the acceptable PSNR range.

The CED method we provide takes use of the involution property of the round operations of this family of symmetric block ciphers and determines if $x=F(F(x))$ for each of the involutional round functions to find systemic errors. The trade-off between performance, fault detection delay, and area and time overhead is optimized. It may also be used with any involution-based symmetric block ciphers and requires little change to the encryption equipment. Time redundancy-based CED methods often have >100% time overhead but cannot yet identify permanent flaws. Despite being predicated on temporal redundancy, the CED system we suggest has essentially little time overhead. The CED method we provide takes use of the involution property of the round operations of this family of symmetric block ciphers and determines if $x=F(F(x))$ for each of the involutional round functions to find systemic errors. The trade-off between performance, fault detection delay, and area and time overhead is optimized. We provide a low-cost, low-latency CED method for involution-based symmetric block ciphers in this study. As long as $F(F(x))=x$, every function F is an involution. Each of the cipher's round operations is an involution in an involutional symmetric block cipher. In symmetric block ciphers, the decryption process typically diverges dramatically from the encrypting process. Although decryption may be implemented so that it follows the same series of steps as encryption, round level operations like S-Box and other similar ones are different. Every operation in involutional ciphers is an involution. This makes it feasible to implement decryption in a way where the only round keys used differ from those for encryption. An involutional structure also benefits from implementation. It may also be used with any involution-based symmetric block ciphers and requires little change to the encryption equipment. Time redundancy-based CED methods often have >100% time overhead but cannot yet identify permanent flaws. Despite being predicated on temporal redundancy, the CED system we suggest has essentially little time overhead.

The paper is arranged as follows section 1- Introduction, section 2 -Literature review, section 3- Method, section 4- Results, Section 5 conclusion.

2. Literature Review

In this [1], various cryptography fundamentals are mentioned. The second unit of cryptography's guiding principles is redundancy. Although all encrypted messages have some redundancy, the message should be encrypted at the sender side. The message is received using a timestamp by the freshness. Every communication that is received is inside the point in time when it is accepted, and the time is prepared for each message. When a communication is received after the period allotted for its receipt, it is rejected. In cryptography, two types are used: bilateral and uneven scientific discipline formula completely separate keys are used on the sender and receiver sides. Encoding transforms plain text into encrypted text. Cryptography systems offer privacy and authentication in communication system. Uneven cryptosystems use two sorts of keys one is public key and another one is non-public key. The key distribution is usually recommended by Diffie and Lillian Hellman. Bilateral cryptosystems use data encryption standard and rotor ciphers. Shyam Nandan Kumar planned review on network security and cryptography [2]. In this paper the categories of security attacks are discussed. Active attacks do some modification information stream. The types of active attacks are modification of message, denial of service, replay and masquerade. In traffic analysis attacks the message is scan by third party. Unharness of message contents attacks the message scan by sender and receiver. Some security services are provided for information transmission, information integrity, Data confidentiality, authenticity, non-repudiation and access management are several the safety services provided by cryptography. Madhumita Panda projected a paper on security in wireless device network victimization cryptological techniques. Data confidentiality, data integrity, information transmission security, authenticity, non-repudiation, and access control are only a few of the security services that cryptography offers. A research on security in wireless device networks that use cryptographic approaches was suggested by Madhumita Panda. In order to generate security for the wireless communication, cryptography is used in this paper's wireless device network. In wireless device networks, security requirements including secrecy, authentication, integrity, and availability are used. Among a restricted resource, some device security barriers are used. A few of the severely constrained resources are limited memory and storage capacity, electricity, and transmission range. One of the challenges, along with faulty transmission, conflicts, and delay, is unreliable communication. Unattended operation is one of the challenges, along with being vulnerable to physical assault, managing remotely, and lacking a centralized control hub. Secret is also known as a secret key. Public key cryptography is a type of unequal encryption that employs only one key on both the transmitter and recipient sides. Both the transmitter and recipient sides utilize completely different keys. Together, Khandoker Abdul Rahad and Sayed Mohsin Reza designed a paper on Vigenere-multiplicative Cipher implementation and network security using cryptography[4]. This essay offers two different sorts of action: encryption and decryption. Network security is treated as a form of communication art. We must encode our information by rearranging our data in order to transmit it. Following the data's encoding. In this research, we construct a formula that combines the MMB encryption technique with the Playfair cipher as key substitution to increase the message's degree of security.

A work on chemical formula encryption utilizing the 7-bit periodic table was organized by Yamuna and Venkata Krishna Pavan Kalahandi [6]. In this study, they designed a new table that takes use of the qualities of the quality table and uses it to encrypt information on any medicine using the drug's name or formula. Sriramulu Ajay Babu writes a paper on the Modification Affine ciphers Algorithms for Cryptography password [7]. In this research, they provide a model of password security using an updated affine cipher approach.

Methodology

Triple layer encryption are more secure no matter what protocols or applications are being communicated, safe transmission is made possible via layer 1 encryption of the OSI model. Voice, data, and video may all be safely encrypted here, along with the Ethernet, Fibre Channel, SDI, and CPRI protocols. Since there is no overhead data while using layer 1 encryption, 100% of the data may be transmitted while the encryption is being processed at line speed. Data connections from 1 Gbps to 200 Gbps full duplex are encrypted using the technologies currently on the market. With layer 2 encryption as opposed to layer 3, there are considerable benefits in data throughput. Data may be sent securely in unicast, multicast, and broadcast modes thanks to encryption at layer 2 of the OSI model. Layer 2 encryption is totally independent from any higher-layer applications and operates transparently over VLAN, CoP, and MPLS connections. When opposed to Layer 3-based encryption, Layer 2 encryption gives a substantial performance benefit due to its reduced overhead. IP-packets may be securely sent across a public transport network thanks to layer 3 of the OSI model's encryption. In the router or the end device, the encryption is often carried out via software. Layer 3 encryption has the highest delay and the most severe throughput limitations because of the way that it is implemented in software. A layer 3 encryption option is one that many system makers, such as those of routers, include as a low-cost alternative in their system technology.

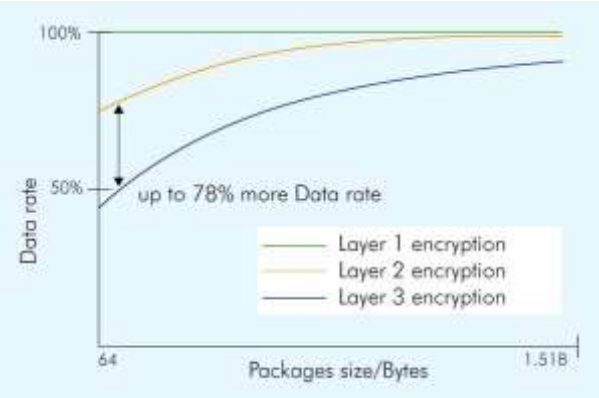


Figure 1: Significant Benefits regarding Data Troughput with L2 Encryption over L3

A Caesar cipher, also known as a Caesar's cipher, a shift cipher, a Caesar's code, or a Caesar shift, is one of the most extensively used and easiest encryption methods. Each letter in the plaintext is changed with a letter that is located a predetermined number of places farther down the alphabet in this sort of substitution cipher. A left shift of 3 would, for instance, cause D to become A, E to become B, and so on. Julius Caesar utilized the technique in his personal communications, thereby earning him the moniker. A substitution cipher is an encryption technique used in cryptography in which the "units" of plaintext are replaced with cipher text in accordance with a regular scheme; the "units" can be single letters (the most frequent), pairs of letters, triplets of letters, combinations of the aforementioned, and so on. The receiver uses an inverse substitution to decode the text Transposition ciphers can be compared to substitution ciphers. The units of the plaintext are kept the same but are rearranged in a different, typically very complex sequence in a transposition cipher. The units of the plaintext are kept in the same order in a substitution cipher, but the units themselves are changed.

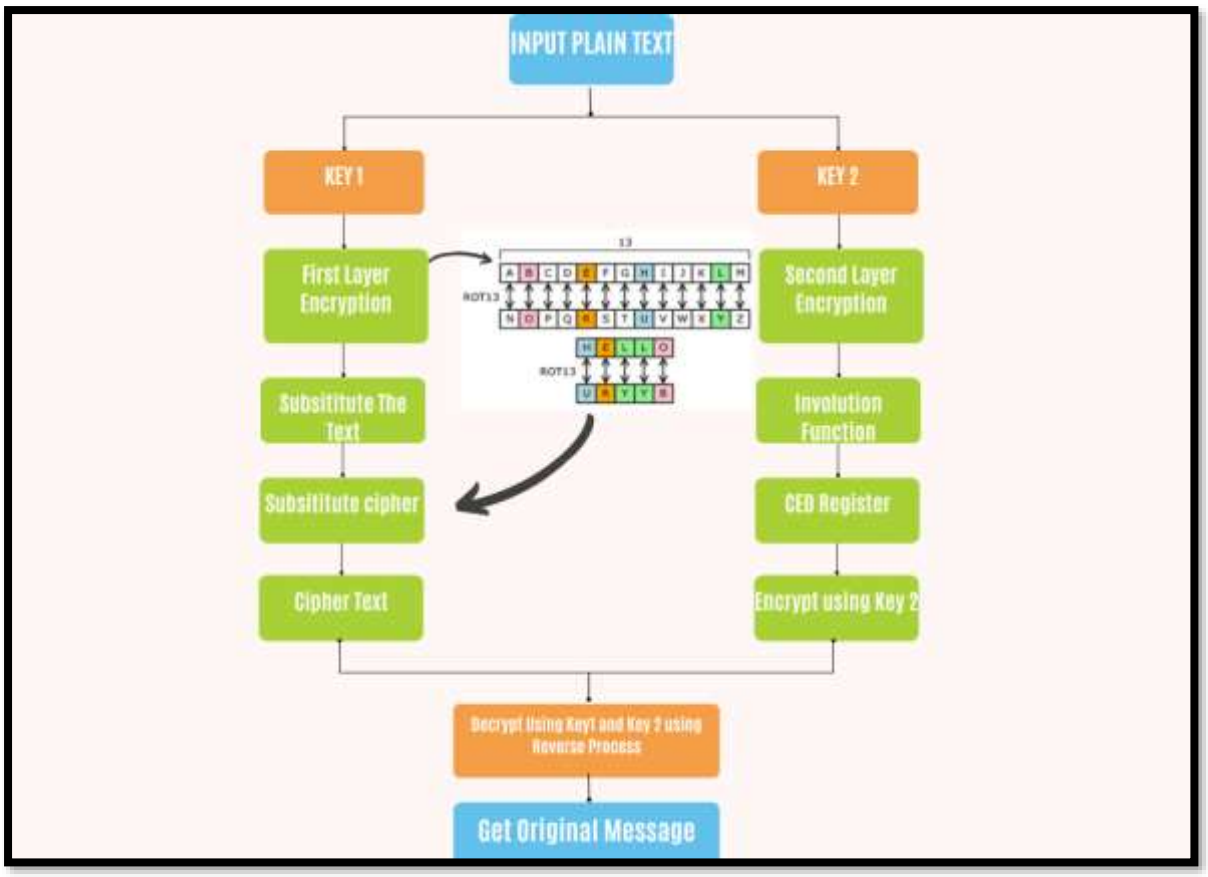


Figure 2: Proposed method of triple layer cryptography using dual substitution na dinvolution function

Layer 1 and 2

You undoubtedly observed a pattern in how letters from the original message related to letters in the decoded one when you examined the Caesar cipher in the preceding section, cracked it, and (hopefully) deciphered what it stated. The original message's letters could all be decrypted to the letter that was located ten alphabetical positions ahead of them. You should have underlined this in your conversion table. This table shows the correspondences between the letters, with the letter "K" standing in for a "A". Your conversion table can translate "A" to "K" rather than "K" to "A," as long as it does it in the opposite direction. Go return to the previous section if you were unable to decipher the Caesar cipher there.

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
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

Because keys in the Encryption key are a number between 1 and 25 (consider carefully why we wouldn't want a key of 26!), we will declare that the key in this case is 10. Keys in the Caesar cipher determine how much the alphabet should be rotated. The conversion table might look something like this if we were to use a key of 8. Before, we examined how to decipher the Casear cipher without being given the key beforehand, obtaining the plaintext from the ciphertext. When we know the key, decrypting the Caesar cipher is significantly simpler. In reality, a strong encryption system makes sure that the plaintext cannot be extracted from the ciphertext without the key; in other words, it may be cracked but not decoded.

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
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

```
ZNK WAOIQ HXUCT LUD PASVY UBKX ZNK RGFE JUM
```

We can remove six spaces from each character in the cipher text since we know the key is six, which is. As an illustration, the letters "T," "H," and "E" are each placed six letters before the letters "Z," "N," and "K," respectively. It follows that "THE" must be the initial word. We can finally obtain the plaintext of: by systematically navigating the whole cipher text in this manner.

```
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
```

```
alphabets = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

input_str = raw_input("Enter message that you would like to encrypt using caesar ciphe
shift = int(raw_input("Enter a shift value:"))

no_of_itr = len(input_str)
output_str = ""

for i in range(no_of_itr):
    current = input_str[i]
    loc = alphabets.find(current)
    if location < 0:
        output_str += input_str[i]
    else:
        newLocation = (location + shift)%26
        output_str += alphabets[newLoc]

print "ciphertext: ", output_str
```

The set of all potential keys for a straightforward substitution cipher is the set of potential permutations. As a result, there are 26 keys (26 factorial) for the English alphabet, or around 403*1024. This makes the brute force method impossible to apply if you wish to decipher the text without knowing the key.But since it is susceptible to cryptoanalysis, the basic substitution cipher is regarded as a weak encryption. First off, if you have a sizable piece of enciphered text and are familiar with the language it was written in, you may perform frequency analysis because replacement does not alter the letter frequencies. For instance, the most frequent letter in the English alphabet is E, making it the most frequent letter.

```
for i in range(26):
    output_str = ""

    for j in range(n):
        c = input_str[j]
        location = alphabets.find(c)
        if location < 0:
            output_str += input_str[j]
        else:
            new_location = (location - i)%26
            output_str += alphabets[new_location]
    data = output_str.split(" ", 20)
    score = int(getScore(data[:20]))
    results.append(Result(score,i,output_str))

results.sort(key=operator.attrgetter('score'))
results.reverse()

print "Plain text from your guess: ", base_output_str, " - with shift: ", shift, "
```

```
Plain text from your guess:  THIS IS MY MESSAGE  - with shift:  3  - the score is 32.0

Below are the potential plain text sorted by the probability of being correct in descending order!
Potential plain text:  THIS IS MY MESSAGE  - with shift:  3  - the score is 32
Potential plain text:  XLMW MW QC QIWWEKI  - with shift:  25  - the score is 0
Potential plain text:  YMNX NX RD RJXXFLJ  - with shift:  24  - the score is 0
Potential plain text:  ZNOY OY SE SKYYGMK  - with shift:  23  - the score is 0
Potential plain text:  AOPZ PZ TF TLZZHNL  - with shift:  22  - the score is 0
Potential plain text:  BPQA QA UG UMAAIOM  - with shift:  21  - the score is 0
Potential plain text:  CQRB RB VH VNBBJPN  - with shift:  20  - the score is 0
Potential plain text:  DRSC SC WI WOCCKQO  - with shift:  19  - the score is 0
Potential plain text:  ESTD TD XJ XPDDLRP  - with shift:  18  - the score is 0
Potential plain text:  FTUE UE YK YQEEMSQ  - with shift:  17  - the score is 0
Potential plain text:  GUVF VF ZL ZRFFNTR  - with shift:  16  - the score is 0
Potential plain text:  HVWG WG AM ASGGOUS  - with shift:  15  - the score is 0
Potential plain text:  IWXH XH BN BTHHPVT  - with shift:  14  - the score is 0
Potential plain text:  JXYI YI CO CUIIQWU  - with shift:  13  - the score is 0
Potential plain text:  KYZJ ZJ DP DVJJRXV  - with shift:  12  - the score is 0
Potential plain text:  LZAK AK EQ EWKKSYW  - with shift:  11  - the score is 0
Potential plain text:  MABL BL FR FXLLTZX  - with shift:  10  - the score is 0
Potential plain text:  NBCM CM GS GYMMUAY  - with shift:  9  - the score is 0
Potential plain text:  OCDN DN HT HZNNVBZ  - with shift:  8  - the score is 0
Potential plain text:  PDEO EO IU IAOOWCA  - with shift:  7  - the score is 0
Potential plain text:  QEFP FP JV JBPPXDB  - with shift:  6  - the score is 0
Potential plain text:  RFGQ GQ KW KCQQYEC  - with shift:  5  - the score is 0
Potential plain text:  SGHR HR LX LDRRZFD  - with shift:  4  - the score is 0
Potential plain text:  UIJT JT NZ NFTTBHF  - with shift:  2  - the score is 0
Potential plain text:  VJKU KU OA OGUUCIG  - with shift:  1  - the score is 0
Potential plain text:  WKLV LV PB PHVVDJH  - with shift:  0  - the score is 0
```

Consider a situation where you intercept a communication and you are aware that the sender is employing a Caesar cipher but are unaware of the shift. The message starts with EQZP. How challenging is it to decipher this message?

Solution

We would only need to test 25 different options to see which one yields results that make sense because there are only 25 potential shifts. While it would be laborious, this could easily be done by hand in a short period of time by one person. A contemporary computer could quickly test every scenario. They examine them to see if they can decipher any words and phrases included in the encrypted text.

Shift	Message	Shift	Message	Shift	Message	Shift	Message
1	DPYO	7	XJSI	13	RDMC	19	LXGW
2	COXN	8	WIRH	14	QCLB	20	KWFV
3	BNWM	9	VHQG	15	PBKA	21	JVEU
4	AMVL	10	UGPF	16	OAJZ	22	IUDT
5	ZLUK	11	TFOE	17	NZIY	23	HTCS
6	YKTJ	**12**	**SEND**	18	MYHX	24	GSBR
						25	FRAQ

Use the substitution mapping below to encrypt the message "March 12 0300"

Original: $mathrm{ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789}$

Maps to: 2BQF5WRTD8IJ6HLCOSUVK3A0X9YZN1G4ME7P

Solution

Using the mapping, the message would encrypt to 62SQT ZN Y1YY

Pseudocode of Substitution cipher

If Plaintext = CNS IS MY FAV SUBJECT and Key = 3

then, by applying the numerical equivalent of each letter algorithm

Ciphertext = FQV LV PB IDY VXEMFS

Algorithm for the numerical equivalent of each letter

A=0

B=1

C=2

.

.

.

Z=25

So,

Encryption algorithm for the above example is

C = E(K,P)=(P+K) mod 26

where K = 1 to 25

and

Decryption algorithm for the above example is

P=D(K,C)=(C-K) mod 26

where, P = Plaintext

C= Ciphertext

K= Key

Consider another example where

Ciphertext=PHHW PH is given and plaintext needs to be found using the decryption algorithm but no key is given.

Key not given we have to calculate it & get the plaintext for which we need to check it with all 26 keys available set of keys with us (A= 0 to Z= 25) till a meaningful message is obtained.

Decryption:

With Key 0

Plaintext = PHHW PH , which is not meaningful so take next key

for Key 1, Plaintext= nffu nf , which is again not meaningful so take next key.

Layer 3

An involution is a function f from a set to itself such that ff is identity (i.e. using the function f on the outcome of the function f returns the original element, i.e. f=f1) Remember that is the composition of functions. A permutation is a function f from a set to itself, such that a function g from that set to itself exists, such that gf is identity (i.e., using the function g on the function f's outcome returns the original element). It is, in essence, a bijection from that set to itself. Every involution is a permutation as a result. Additionally, f is a permutation and is an involution if and only if f f is identity. One example of this is a random involution or permutation.

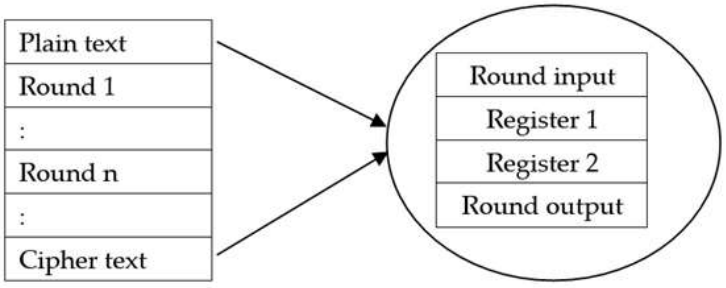


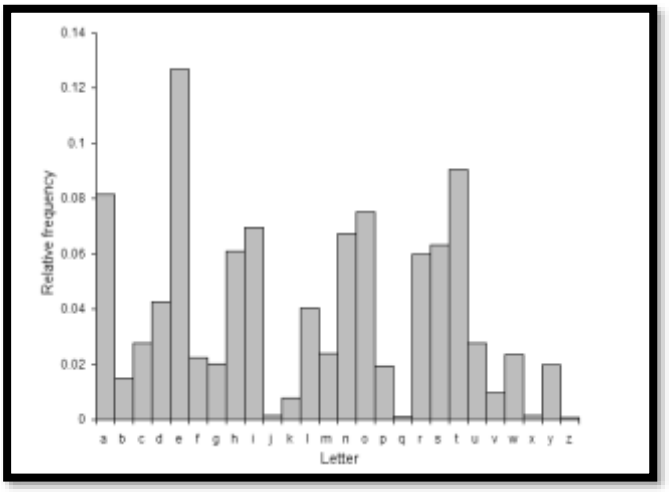
Figure 3: Involution function mapping

In the event that a hardware module implements an involution F, errors in this module can be found by seeing if x = F(F(x)). This fundamental concept may be shown in Fig. 3. The initial input x for the evolution F is in a register, buffered. The output of the module F(x) is stored in a different register before being sent back to the input of the module F using a multiplexer that is controlled by the CED control signal.A discrepancy between the output of this second computation, F(F(x)), and the original input x indicates an error. We will now use involutional substitution-permutation-network (SPN) block ciphers to apply the suggested CED approach to involutional functions from the domain of cryptography. Each round of an SPN symmetric block cipher is made up of a key-mixing function, a linear diffusion function, and a nonlinear replacement function. The linear diffusion function makes sure that after a certain number of cycles, all the output bits depend on all the input bits. This dependency is made complicated and nonlinear by the nonlinear function. The round keys are added by the key-mixing function and are created using the secret key that the user provides.

In this research, we provide a low-cost, low-latency CED method for symmetric block ciphers based on involution. Involution occurs when F(F(x))=x for any function F. Every round operation of an involutional symmetric block cipher entails an involution. The decryption method often differs greatly from encryption in symmetric block ciphers. While it is technically feasible to perform decryption so that it follows the same set of steps as encryption, round level operations like S-Box and other similar ones are different. The operations used in involutional ciphers are all involutions. As a result, it is now possible to construct decryption in a manner where the only round keys that are different from those used for encryption are those used for decryption. An involutional structure also offers the benefit of implementation.

Result and discussion

It is quite straightforward to crack the basic substitution encryption. Even though there are around 288.4 keys—a very large number—there is a lot of redundancy in English text and other statistical qualities that make it simple to choose a key that works quite well. The frequency distribution of the letters in the encrypted text must be calculated first. This involves keeping track of how frequently each letter occurs. Natural English text has a fairly distinctive distribution that can be utilised to decipher codes. This distribution is as follows:



English Letter Frequencies'

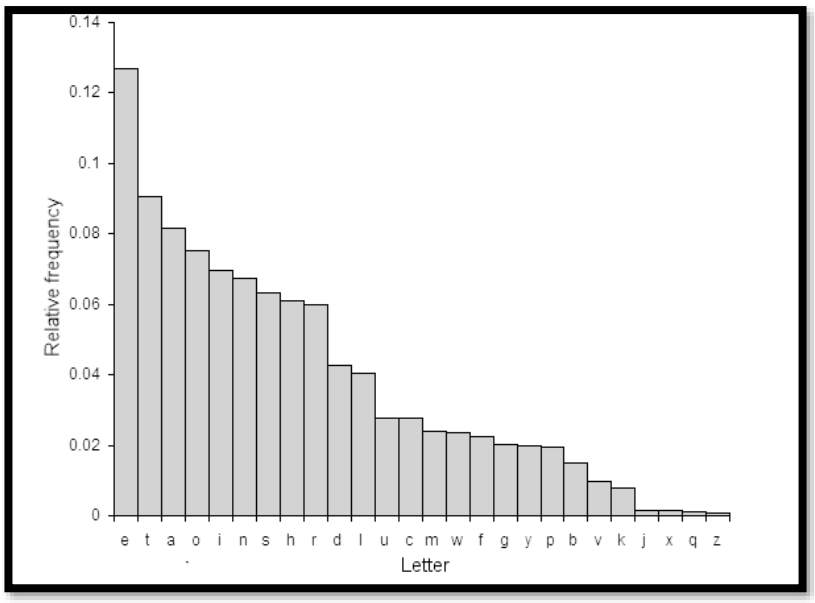


Figure 4: Distribution of English alphabets for substitution cipher

The most frequent letter, then, is 'e,' which appears about 13% of the time. By contrast, 'z' only appears less than 1% of the time. While using the basic substitution cipher, these letter frequencies are only slightly jumbled (in the example above, the letter "e" is encoded as the letter I making I the most frequent character in the cipher text). Finding the mapping for each character is necessary for a cryptanalyst to determine the key that was used to encrypt the message. It is feasible to simply substitute the most prevalent cipher text character with "e," the second-most prevalent cipher text character with "t," etc. for each character (change "e" with "t" for each character in relatively big amounts of text (several hundred characters)).

A password can be hashed using a method that creates a new string of characters known as a hashed password from a single password. The hashed password is often one-way, meaning we can't move from the hashed password to the original password. So the question is why we needed to utilise hashing to perform all of this, why go the extra mile when we simply keep our passwords in the database as a simple string. Because hackers don't steal login information from our valued site, we are taking all of these steps only to increase security. To safeguard our passwords when building websites and keeping our database, we thus utilise a variety of hashing techniques. There are several popular cryptographic methods in PHP, including md5.

```
    // if (Hash::check($password, $userHasedPassword)) {
    //     return true;
    // } else {
    //     return false;
    // }
    // if (count(users::where("email", $email)->where("password", $password)->get()) != 0) {
    //     return true;
    // } else {
    //     return ["message"=>"$email $password"];
    // }
});
```

```
if(count($validation->errors())==0) {

    $email = request("email");
    $password = request("password");
    if (count(users::where("email", $email)->get())!=0){
        if (users::where("email", $email)->first()->activate) {
            $userHasedPassword = users::select("password")->where("email", $email)->first();
            if (Hash::check($password, $userHasedPassword->password)) {
                return true;
            } else {
                return ["message"=>"Invalid Credentials"];
            }
        } else {
            return ["message"=>"Your account is deactivated"];
        }
    } else {
        return ["message"=>"No such account exists"];
    }
```

Figure 5 : sample of implemented model on E-commerce website enabled hashing algorithm

A permutation of a set S called an involution is one in which there are no permutation cycles with a length greater than two (i.e., it consists exclusively of fixed points and transpositions). Self-conjugate permutations and involutions have a one-to-one connection (i.e., permutations that are their own inverse permutation). For instance, the only involution permutation for an element with one digit is 1, the involution permutations for two digits are 1,2 and 2, and for three digits, 1,2,3, 1,3,2, and 3,2, respectively.

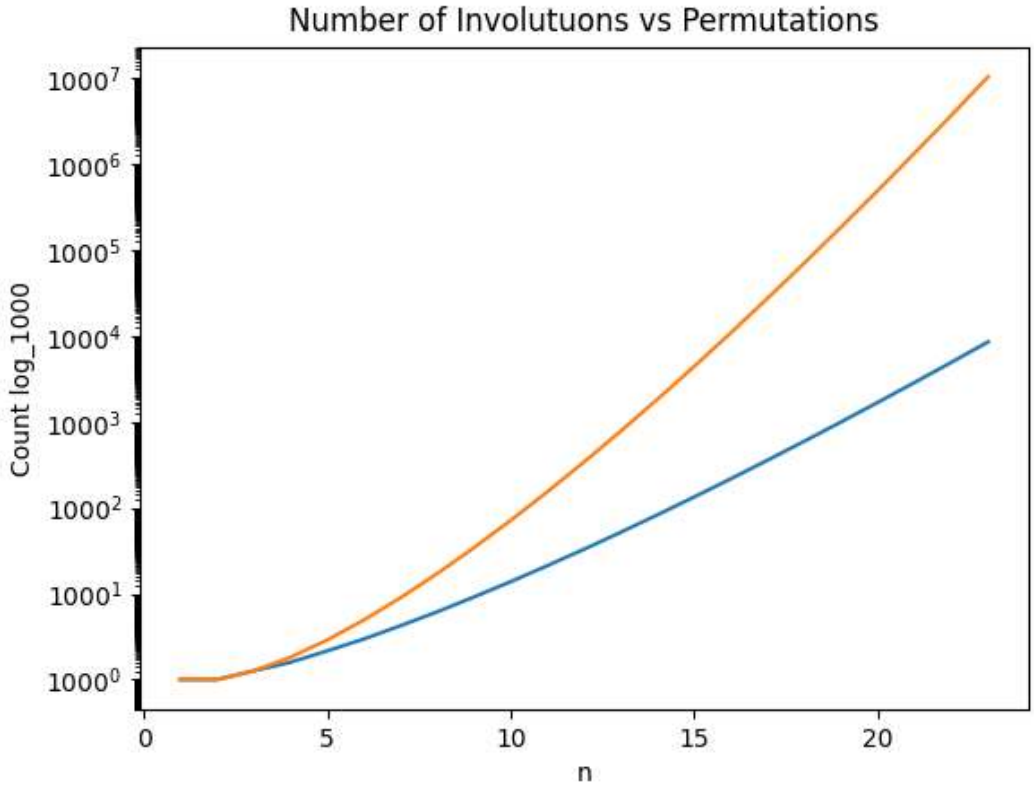


Figure 6: Number of cycles of permuting layers of encryption in involution layer encryption

The decryption I will be attempting is called substitution cipher, where each letter of the alphabet corresponds to another letter (possibly the same one). This is the rule that you must follow for the cryptogram game too. There are 26! = 4e26 possible mappings of the letters of the alphabet to one another. This seems like a hopeless problem to solve, but I will show you that a relatively simple solution can be found as long as your reference text is large enough and the text you are trying to decipher is also large enough. The strategy is to use a reference text to create transition probabilities from each letter to the next. This can be stored in a 26 by 26 matrix, where the ith row and jth column is the probability of the jth letter, given the ith letter preceded it. For example, the given the previous letter was a Q, U almost always follows it, so the Q-row and U-column would be close to probability of 1. Assuming these one step transition probabilities are what matter, the likelihood of any mapping is the product of the transition probabilities observed. To construct transition probabilities from one letter to the next, the approach involves using a reference text. The probability of the jth letter, assuming the ith letter came before it, may be recorded in a 26 by 26 matrix, where ith row and jth column are. For instance, if the preceding letter was a Q, U usually invariably follows it, making the likelihood of the Q-row and U-column to be close to one. The likelihood of any mapping is the total of the observed transition probabilities, if these one-step transition probabilities are what are important.

```
def encrypt(plaintext, plaintext_alphabet ,cipher_alphabet, blacklist={}):
    assert set(list(plaintext)).difference(blacklist).issubset(set(list(plaintext_alphabet))), "Plaintext must only contain characters in Plaintext alphabet"
    assert len(plaintext_alphabet) == len(cipher_alphabet), "Cipher alphabet must have same number of characters as Plaintext alphabet"

    enc_key = dict(zip(list(plaintext_alphabet),list(cipher_alphabet)))
    acc = []
    for s in plaintext:
        if s in blacklist:
            acc += s
        else:
            acc += [enc_key[s]]
    ciphertext = "".join(acc)

    return {'cipher_alphabet' : cipher_alphabet,
            'plaintext_alphabet' : plaintext_alphabet,
            'ciphertext' : ciphertext,
           }

def decrypt(ciphertext, plaintext_alphabet, cipher_alphabet, blacklist={}):
    dec_key = dict(zip(list(cipher_alphabet),list(plaintext_alphabet)))
    acc = []
    for s in ciphertext:
```

test_pattern = '[^A-Z]'

pattern = re.compile(test_pattern)

message = "ENTER HAMLET TO BE OR NOT TO BE THAT IS THE QUESTION WHETHER TIS NOBLER IN THE MIND TO SUFFER THE SLINGS AND ARROWS OF OUTRAGEOUS FORTUNE OR TO TAKE ARMS AGAINST A SEA OF TROUBLES AND BY OPPOSING END"

message_cleaned = pattern.sub('',message.upper())

en_alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ "

tmp = list(en_alphabet)

random.shuffle(tmp)

cipher_alphabet = "".join(tmp)

message_enc = encrypt(message_cleaned,en_alphabet,cipher_alphabet,)

ciphertext = message_enc['ciphertext']

```
print(message_enc)

print(decrypt(message_enc['ciphertext'],en_alphabet, cipher_alphabet,))

print((en_alphabet,cipher_alphabet))

results = process_text('credit_card.txt',regex_ignore=test_pattern)

P = results['transition_matrix']

F = results['bigram_freq_matrix']

m = P.shape[0]

i_c_map = results['index_character_map']

c_i_map = results['character_index_map']
```

To ensure that everything seems proper, I want to view my data. The transition matrix is displayed below. The empty space is a non-letter character. This narrative offers a wealth of revelations. Q-relationship U's becomes apparent. T is the letter that begins words the most frequently. E is a common letter to come after several others. Y often completes words. The next step is to suggest a new mapping by randomly exchanging 2 of the characters. Therefore, if you swapped the two, A would map to Z and L to G. Similarly, if you switched the two, L would map to G. By dividing the likelihood by the likelihood of using this new mapping.

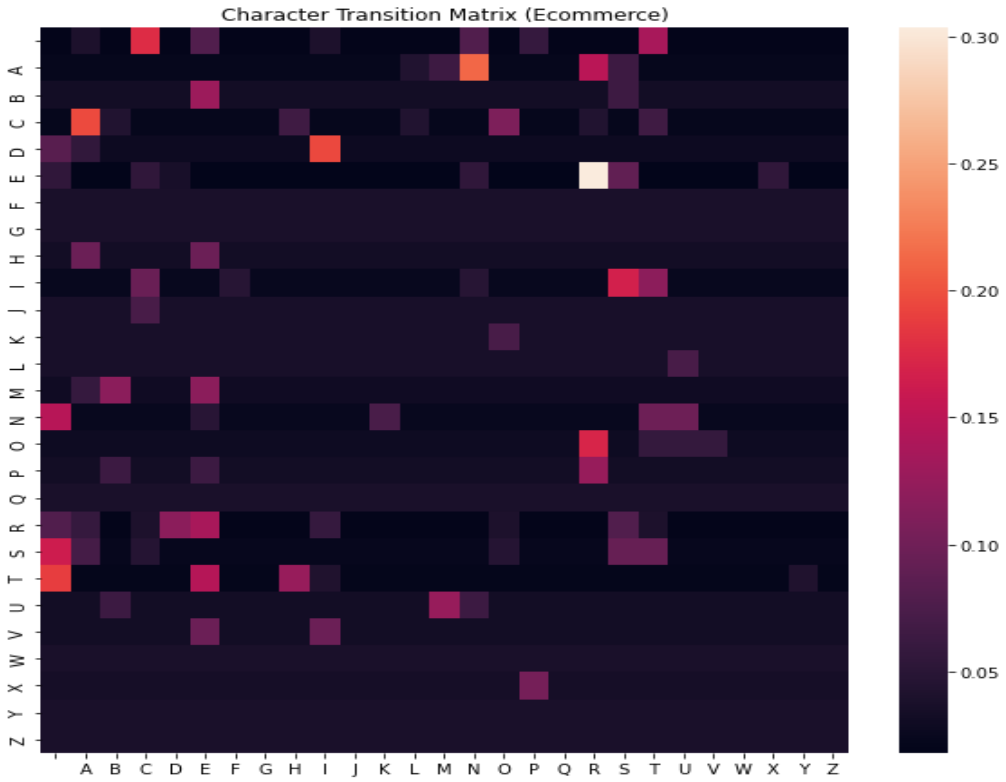


Figure 7 : Heatmap of matrix distribution of alphabets

ciphertext:
UWIUMOBNLPUIOIQODUOQMOWQIOIQODUOIBNIOAZOIBUOCXUZIAQWOHBUIBUMOIAZOWQ
DPUMOAWOIBUOLAWEOIQOZXRRUMOIBUOZPAWZONWEONMMQHZOQROQXIMN
UQXZORQMIXWUOQMOIQOINJUONMLZON
NAWZIONOZUNOQROIMQXDPUZONWEODTOQKKQZAW OUWE

iter: 0
TZGTDLIMAQTGLGYLVTLYDLZYGLGYLVTLGIMGLNULGITLPWTUGNYZLKITGITDLGNULZYV
QTDLNZLGITLANZFLGYLUWCCTDLGITLUQ
iter: 500 C UCONITZSCUNUKNFCNKON KUNUKNFCNUITUNRANUICNPWCAURK
NDICUICONURAN KFSCONR NUICNZR ENUKNAWLLCONUICNAS
iter: 1000 C UCONITGZCUNUKNXCNKON KUNUKNXCNUITUNRANUICNQLCAURK
NDICUICONURAN KXZCONR NUICNGR ENUKNALSSCONUICNAZ
iter: 1500 S KSONITVESKNKUNBSNUON UKNKUNBSNKITKNRANKISNGMSAKRU
NCISKISONKRAN UBESONR NKISNVR PNKUNAMFFSONKISNAE
iter: 2000 S USONITHESUNUKNBSNKON KUNUKNBSNUITUNRANUISNLMSAURK
NDISUISONURAN KBESONR NUISNHR CNUKNAMWWSONUISNAE
iter: 2500 S KSONITHESKNKUNBSNUON UKNKUNBSNKITKNRANKISNVMSAKRU
NDISKISONKRAN UBESONR NKISNHR PNKUNAMXXSONKISNAE
iter: 3000 S KSONITQESKNKUNBSNUON UKNKUNBSNKITKNRANKISNYMSAKRU
NDISKISONKRAN UBESONR NKISNQR PNKUNAMZZSONKISNAE
iter: 3500 S KSONITVESKNKUNHSNUON UKNKUNHSNKITKNRANKISNYJSAKRU
NMISKISONKRAN UHESONR NKISNVR PNKUNAJBBSONKISNAE
iter: 4000 S KSONITMESKNKUNXSNUON UKNKUNXSNKITKNRANKISNZLSAKRU
NDISKISONKRAN UXESONR NKISNMR CNKUNALBBSONKISNAE
iter: 4500 S KSONITHESKNKUNBSNUON UKNKUNBSNKITKNRANKISNYJSAKRU
NXISKISONKRAN UBESONR NKISNHR CNKUNAJMMSONKISNAE
total acceptances: 1040
score: -586.8260850370232

attempted decryption:
S KSONITHESKNKUNBSNUON UKNKUNBSNKITKNRANKISNZMSAKRU NDISKISONKRAN
UBESONR NKISNHR PNKUNAMLLSONKISNAER CANT
PNTOOUDANULNUMKOTCSUMANLUOKM SNUONKUNKTGSNTOHANTCTR
AKNTNASTNULNKOUMBESANT PNBWNUYYUAR CNS P

original message:
ENTER HAMLET TO BE OR NOT TO BE THAT IS THE QUESTION WHETHER TIS NOBLER IN THE MIND TO SUFFER THE SLINGS AND ARROWS OF OUTRAGEOUS FORTUNE OR TO TAKE ARMS AGAINST A SEA OF TROUBLES AND BY OPPOSING END

score of true key: -591.0435306815182
similarity score: 0.02512562814070352

```
print('score of true key:', ground_truth_score['score'])
print('similarity score:', similarity(message_cleaned, soln['plaintext']))
```

```
ciphertext:
UWIUMOBNLPUIOIQODUOQMOWQIOIQODUOIBNIOAZOIBUOCKUZIAQWOHBUIBUMOIAZOWQDPUMOAWOIBUOLAWEOIQOZKRRUMOIBUOZPAW ZOWWEONMMQHZOQROQXIMN UQXZORQMIXMUOQMOIQOINJ

iter: 0 TZGTDLIMAQTGLGYLVTLYDLZYGLGYLVTLGIMGLNULGITLPWTUGNYZLKITGITDLGNULZYVQTDLNZLGITLANZFLGYLUWCCTDLGITLUQ
iter: 500 C UCONITZSCUNUKNFCNKON KUNUKNFCNUITUNRANUICNPWCAURK NDICUICONURAN KFSCONR NUICNZR ENUKNAWLLCONUICNAS
iter: 1000 C UCONITGZCUNUKNXCNKON KUNUKNXCNUITUNRANUICNQLCAURK NDICUICONURAN KXZCONR NUICNGR ENUKNALSSCONUICNAZ
iter: 1500 S KSONITVESKNKUNBSNUON UKNKUNBSNKITKNRANKISNGMSAKRU NCISKISONKRAN UBESONR NKISNVR PNKUNAMFFSONKISNAE
iter: 2000 S USONITHESUNUKNBSNKON KUNUKNBSNUITUNRANUISNLMSAURK NDISUISONURAN KBESONR NUISNHR CNUKNAMWWSONUISNAE
iter: 2500 S KSONITHESKNKUNBSNUON UKNKUNBSNKITKNRANKISNVMSAKRU NDISKISONKRAN UBESONR NKISNHR PNKUNAMXXSONKISNAE
iter: 3000 S KSONITQESKNKUNBSNUON UKNKUNBSNKITKNRANKISNYMSAKRU NDISKISONKRAN UBESONR NKISNQR PNKUNAMZZSONKISNAE
iter: 3500 S KSONITVESKNKUNHSNUON UKNKUNHSNKITKNRANKISNYJSAKRU NMISKISONKRAN UHESONR NKISNVR PNKUNAJBBSONKISNAE
iter: 4000 S KSONITMESKNKUNXSNUON UKNKUNXSNKITKNRANKISNZLSAKRU NDISKISONKRAN UXESONR NKISNMR CNKUNALBBSONKISNAE
iter: 4500 S KSONITHESKNKUNBSNUON UKNKUNBSNKITKNRANKISNYJSAKRU NXISKISONKRAN UBESONR NKISNHR CNKUNAJMMSONKISNAE
total acceptances:  1040
score: -586.8260850370232

attempted decryption:
S KSONITHESKNKUNBSNUON UKNKUNBSNKITKNRANKISNZMSAKRU NDISKISONKRAN UBESONR NKISNHR PNKUNAMLLSONKISNAER CANT PNTOOUDANULNUMKOTCSUMANLUOKM SNUONKUNKTGS

original message:
ENTER HAMLET TO BE OR NOT TO BE THAT IS THE QUESTION WHETHER TIS NOBLER IN THE MIND TO SUFFER THE SLINGS AND ARROWS OF OUTRAGEOUS FORTUNE OR TO TAKE

score of true key: -591.0435306815182
similarity score: 0.02512562814070352
```

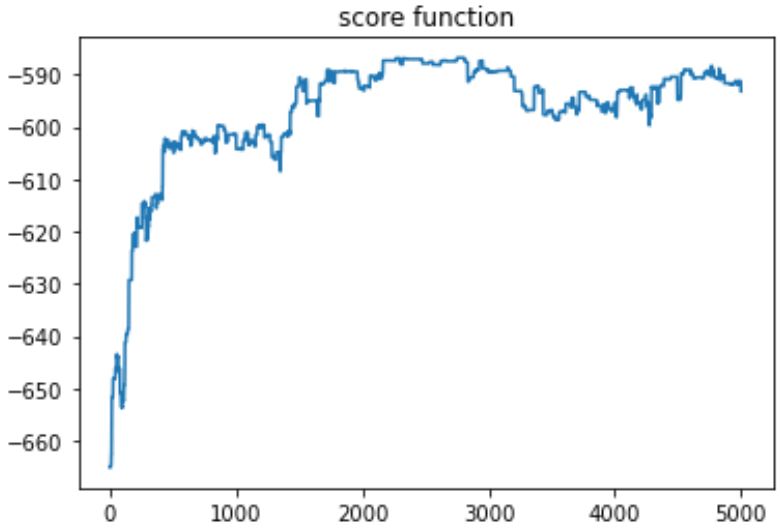
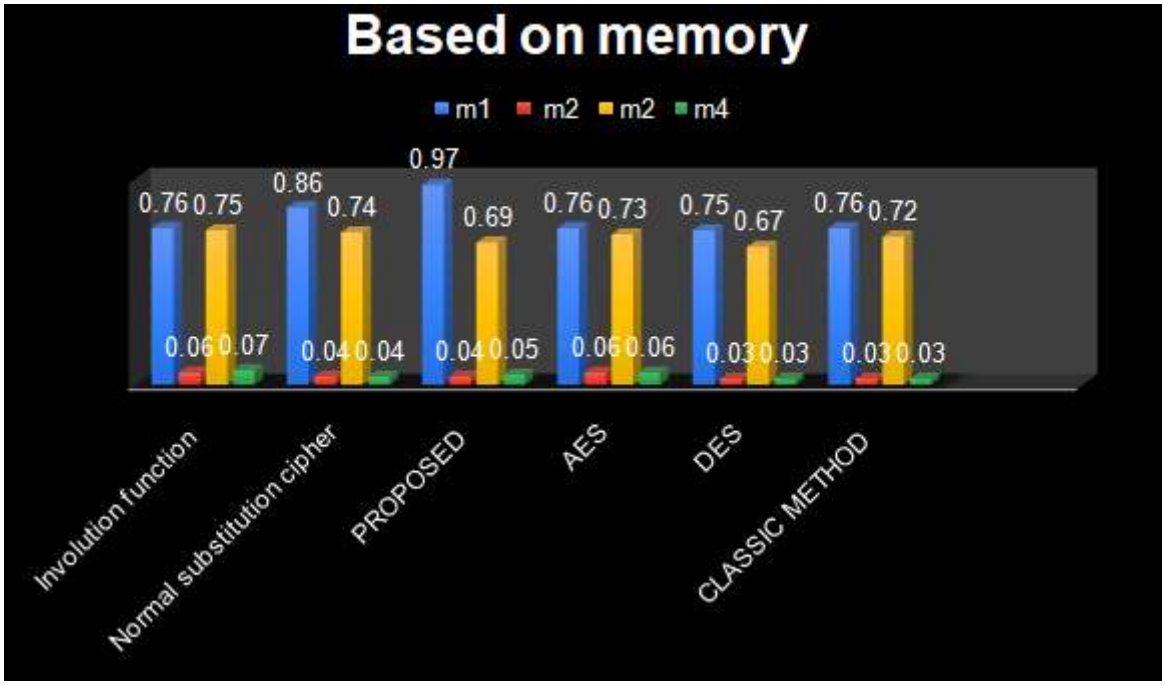


Figure 8 ; function score curve

In figure 9 (a), the comparison of proposed method is shown to higher and more secure than other method like AES, DES based on throughput parameter. In general, we assess communication and computation complexity when evaluating cryptographic algorithms: (Memory, Processing time, communication overhead, energy consumption,etc). Some cryptosystems, especially asymmetrical ones, are not size-preserving. Not all cryptography algorithms rely on mathematical complexity. When quantum cryptography algorithms are analyzed, due to the different nature of the problem to be solved, different metrics are used (Eavesdropper equivocation, or success rate for example). The question posted is of very general type, so the spectrum of answers exists, without possibility for unique answer, given the state of the Cryptography science.



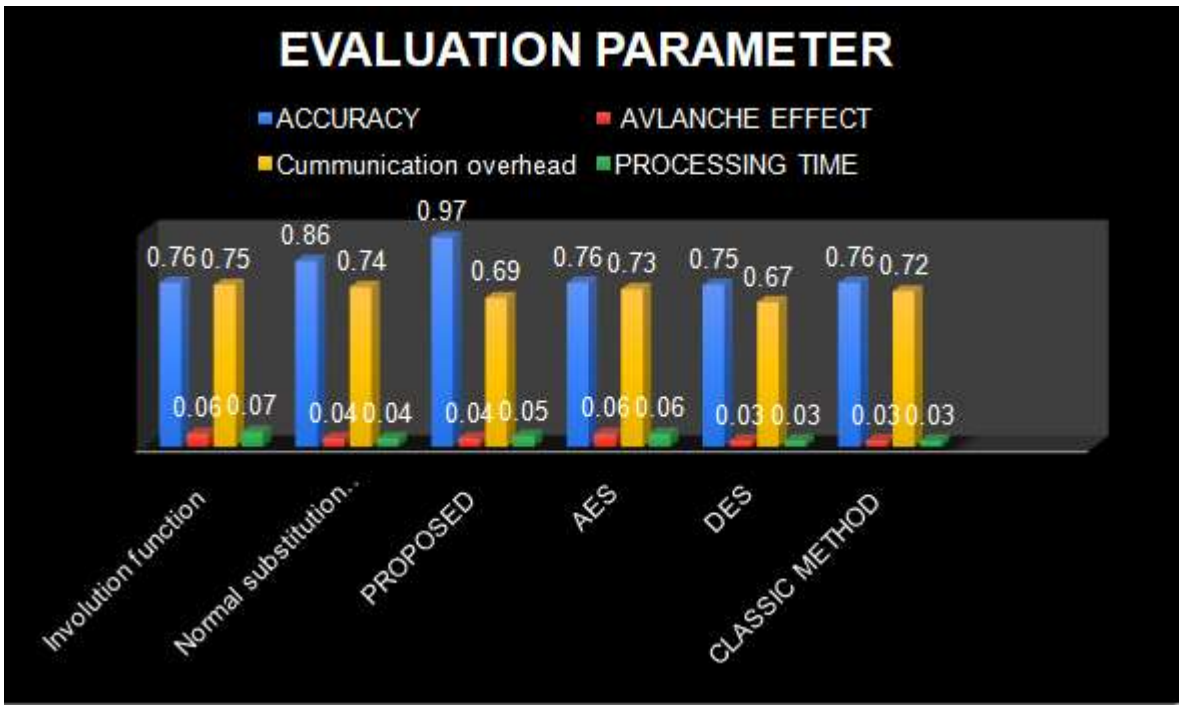


Figure 9 : comparison and performance evaluation of different algorithm with proposed one

Conclusion

We put forth a low-cost CED method for involutional ciphers that takes use of the cipher's involution characteristic. The suggested method requires an extra 23.8% silicon area for the involutional encryption and reduces throughput by less than 10%The time overhead for this method is 4.5% (however it may be decreased to 0% if the CED is only disregarded for the layer in the final cycle of encryption/decryption). The proposed CED approach identifies all single-bit faults and about 99% of all multiple-bit faults, according to the fault injection-based simulation. The simplest substitution ciphers are those in which the cipher alphabet is just a cyclical shift of the plaintext alphabet. The Caesar cipher, which Julius Caesar employed, is the most well-known of them and encrypts letters A through F as D, B as E, and so on. Cyclic-shift substitution ciphers are not safe, and neither are any other monoalphabetic substitution ciphers when a given plaintext symbol is always encrypted into the same ciphertext symbol, as many a schoolboy has learned to his humiliation. Monoalphabetic substitution ciphers are a common source for amusing cryptograms due to the redundancy of the English language and the fact that just roughly 25 symbols of ciphertext are necessary to allow cryptanalysis. This flaw can be explained by the fact that the frequency. As implied by its name, hybrid approaches combine a number of different data-coverage techniques. We're going to examine cryptography. Combining the two will produce a hybrid outcome. It has the idea of hiding the message but not the idea of altering it through cryptography engineering, whereas cryptography has the problem of disclosing data encryption or encoding. Therefore, when we combine the two approaches, their shortcomings are no longer an issue. We refer to the practice of having numerous layers, or rather processes, that are each independently secured as multiple layer encryption.

References

[1] P. Chodowiec and K. Gaj, "Very Compact FPGA Implementation of the AES Algorithm," in Cryptographic Hardware and Embedded Systems - CHES 2003, vol. 2779, C. D. Walter, Ç. K. Koç, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 319–333. doi: 10.1007/978-3-540-45238-6_26.

[2 Department of Computer Science, Maharaja Surajmal Institute of Technology C4E Janakpuri, New Delhi (110058), India and H. Gupta, "Twin Key Implementation in Aes," IOSRJCE, vol. 16, no. 5, pp. 01–05, 2014, doi: 10.9790/0661-16580105.

[3] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, "Strong Authentication for RFID Systems Using the AES Algorithm," in Cryptographic Hardware and Embedded Systems - CHES 2004, vol. 3156, M. Joye and J.-J. Quisquater, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 357–370. doi: 10.1007/978-3-540-28632-5_26.

[4] R. Jain, R. Jejurkar, S. Chopade, S. Vaidya, and M. Sanap, "AES Algorithm Using 512 Bit Key Implementation for Secure Communication," vol. 2, no. 3, p. 7, 2007.

[5 N. Mathur and R. Bansode, "AES Based Text Encryption Using 12 Rounds with Dynamic Key Selection," Procedia Computer Science, vol. 79, pp. 1036–1043, 2016, doi: 10.1016/j.procs.2016.03.131.

[6] R. Padate and A. Patel, "IMAGE ENCRYPTION AND DECRYPTION USING AES ALGORITHM," International Journal of Electronics and Communication Engineering, vol. 6, no. 1, p. 8.

[7] P. Patil, P. Narayankar, Narayan D.G., and Meena S.M., "A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish," Procedia Computer Science, vol. 78, pp. 617–624, 2016, doi: 10.1016/j.procs.2016.02.108.

[8] O. C. Abikoye, A. D. Haruna, A. Abubakar, N. O. Akande, and E. O. Asani, "Modified Advanced Encryption Standard Algorithm for Information Security," Symmetry, vol. 11, no. 12, p. 1484, Dec. 2019, doi: 10.3390/sym11121484.

[9] M.-L. Akkar and C. Giraud, "An Implementation of DES and AES, Secure against Some Attacks," in Cryptographic Hardware and Embedded Systems — CHES 2001, vol. 2162, Ç. K. Koç, D. Naccache, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 309–318. doi: 10.1007/3-540-44709-1_26.

[10] P. Kumar and S. B. Rana, "Development of modified AES algorithm for data security," Optik, vol. 127, no. 4, pp. 2341–2345, Feb. 2016, doi: 10.1016/j.ijleo.2015.11.188.

[11] A. Razaque, M. B. H. Frej, B. Alotaibi, and M. Alotaibi, "Privacy Preservation Models for Third-Party Auditor over Cloud Computing: A Survey," Electronics, vol. 10, no. 21, p. 2721, Nov. 2021, doi: 10.3390/electronics10212721.

[12] N. S. S. Srinivas, "FPGA Based Hardware Implementation of AES Rijndael Algorithm for Encryption and Decryption," p. 8.

[13] X. Zhang and K. K. Parhi, "On the Optimum Constructions of Composite Field for the AES Algorithm," IEEE Trans. Circuits Syst. II, vol. 53, no. 10, pp. 1153–1157, Oct. 2006, doi: 10.1109/TCSII.2006.882217.

[14] Q. Zheng, N. Liu, and F. Wang, "An Adaptive Embedding Strength Watermarking Algorithm Based on Shearlets' Capture Directional Features," Mathematics, vol. 8, no. 8, p. 1377, Aug. 2020, doi: 10.3390/math8081377.

[15] "FIPS 197, Advanced Encryption Standard (AES)," p. 51.

[16] P. Patil, P. Narayankar, Narayan D.G., and Meena S.M., "A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish," Procedia Computer Science, vol. 78, pp. 617–624, 2016, doi: 10.1016/j.procs.2016.02.108.

[17] T. S. Messerges, "Securing the AES Finalists Against Power Analysis Attacks," in Fast Software Encryption, vol. 1978, G. Goos, J. Hartmanis, J. van Leeuwen, and B. Schneier, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 150–164. doi: 10.1007/3-540-44706-7_11.

Appendex

Used combination of plain text key of credit Card

Credit Card Type

Credit Card Number

American Express

378282246310005

American Express

371449635398431

American Express Corporate

378734493671000

Australian BankCard

5610591081018250

Diners Club

30569309025904

Diners Club

38520000023237

Discover

6011111111111117

Discover

6011000990139424

JCB

3530111333300000

JCB

3566002020360505

MasterCard

5555555555554444

MasterCard

5105105105105100

Visa

4111111111111111

Visa

4012888888881881

Visa

4222222222222

Note : Even though this number has a different character count Qsuantity than the other test numbers, it is the correct and functional number zero.

Processor-specific Cards

Dankort (PBS)

76009244561

Dankort (PBS)

5019717010103742

Switch/Solo (Paymentech)

6331101999990016