

AN EFFICIENT IMPLEMENTATION OF THE TLS PROTOCOL

A PROJECT REPORT

Submitted by

Manik , 18BCE2167

CSE3502
Information Security Management

Under the guidance of
Dr. Kakelli Anil Kumar
Associate Professor
SCOPE, VIT, Vellore.



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

May -2021

INDEX

	Page no.
1. Introduction	
1.1. Background	4
1.2. Motivation	5
2. Literature Survey	6
3. Overview of the Work	10
3.1.Problem description	10
3.2.Working model	10
3.3.Design description	11
3.4.Algorithms	16
4. Implementation	16
4.1.Description of Modules/Programs	16
4.2. Algorithms/ Techniques/ Tools/ source code	18
4.3.Execution of the project	30
4.4.Results analysis in the form of graphs and tables	35
5. Conclusion and Future Scope	39
6. References	41

ABSTRACT

The growing level of the computational complexity has enabled us to explore multiple dimensions of the encryption algorithms which are involved in the cryptography , mathematical computations are also optimised in order to generate highly secure encryption keys which can be achieved without compromising application performance and efficiency.

There are a number of broad classification for the cryptographic encryption methods out of which symmetric and asymmetric Cryptography encryption methods are one of the most popular methods which are used for both encryption and decryption of the information that is received by the algorithm. There are a certain number of drawbacks that will be discussed in the upcoming segments by which some intended and unintended attacks can be conducted on the system or by using the brute force attacks there are some aspects of the algorithms that can be unauthorized accessed or scripted to fail.

With the help of our project we have developed a complete ecosystem of cryptography that uses some of the most optimised algorithms with the minimal effect on the performance to generate quasi-random numbers using generators that are specifically designed for this purpose.

TLS Protocol, Complete Cryptosystem, DHA, Symmetric Encryption, Pseudo Random Generator, Stream Cipher, Forward Secrecy.

1. INTRODUCTION

1.1 Background

It is our fundamental tendency as an individual's being to speak with one another. We tend to form networks of individuals around us so as to possess a share of feelings and have conversation with them so as to precise over what we feel and describe them thoroughly. This is often very normal as a personality's being so as to survive but there are many undesirable aspects thereto which are unfortunately very hard to get rid of from the conventional tendency of groups of people. One such behavior is that the explicit disclosure of the thoughts being shared among the those who aren't concerned with the matter but maybe it is a results of its eavesdropping on any possible manner during which this disclosure might happen. There are many of us who are engaged within the study of this possible encryption of the conversation which is able to allow the restrictive flow of communication that we today call cryptography.

In our day-to-day lives, the utilization of cryptography is everywhere. For instance, we use it to securely send passwords over vast networks for online purchases. Bank servers and e-mail clients save your passwords using cryptography similarly. Cryptography is employed to secure all transmitted information in our IoT-connected world, to authenticate people and devices, and devices to other devices. If all of the cryptographic engines/functions stopped working for every day, modern life as we all know it might stop. Bank transactions wouldn't undergo, internet traffic would come to a halt, and cell phones would not function. At this time, all of our important information would be exposed, and it then might be exploited to try and do unimaginable harm to us all.

Cryptography is a necessary way of preventing that from happening. It secures information and communications employing a set of rules that enables only those intended—and nobody else—to receive the knowledge to access and process it.

1.2 Motivation

The motivation of our project is the importance of cryptography in today's world. It uses "security by obscurity" as a way to keep the transmitted information secure. In those cases, the technique used was kept secret from all but a few, hence the term "obscurity." This made the communication secure, but it was not very easy to implement on a wide scale. Classical cryptographic methods are only secure when two parties can communicate in a secure ecosystem.

The primary purpose of an algorithmic approach to the Cryptography is to make everything randomised as possible so that the entire conversation can be encrypted or simply put very hard to be able to be guessed by someone who is not concerned in that conversation. It can only be decrypted using a private piece of material suggesting a shared private key or something similar to it not to have that privilege of decoding that particular information. Modern-day computational capabilities have enabled more computations to be done in a less amount of time and this is also so good to generate the randomness of the parameters that stress upon a hardened Encryption Algorithm. Some of these algorithms are a part of the standard procedures or protocols that are being widely used in defense organisations and companies who want to protect their integral assets and the assets of their loyal customers who use the product for the confidence of security as well as get the work done in predictable time which is often desirable with the security instances.

The concern regarding the security issues is a part of interest for many people who are particularly concerned and many traditional and conventional key encryption methods are being revised from ages in order to provide more security measures against the increasing computational complexity of the modern day computers and Systems but there are certain fundamental drawbacks for the traditional approaches in which some of the systems may be compromised despite of having large-scale research and development on the security measures. In order to prevent those loopholes to occur we are in a process to revise new algorithms in which this particular concern of getting the algorithm compromised by some third party agent is being minimised to the minimum level without impacting the performance of the system to a significant level.

2. Literature review

We have referred to many publications in order to validate the efforts to prove the effectiveness of our algorithm for which summary is provided below.

1. Cryptography and Encryption Algorithms for Information Security

Year of publication -2014

Summary:

This paper revisits the popular encryption algorithms like AES , DES , Blowfish and discovers the in-depth mechanism of their working and put them against different situations in order to have a valid and fair comparison practical situation where there is a need of to balance the band requirement for the algorithm to take place and the performance overhead in order to review the best algorithm among these.

It was concluded in the paper that blowfish algorithm was considered as the best among the algorithms considered for the purpose with the best network utilisation and in various conditions.

2. An Advanced Architecture for Securing Data in Cloud

Year of Publication - 2016

Summary:

In the recent situation our dependency on the cloud services have increased by manyfolds and the security concerns related to the authentication and the access in the cloud storage has been a point of discussion for the moment. In this architecture, the paper is primarily dealing with three stage Encryption Algorithm that consists of Diffe Hellman algo, Base64 encryption and two fish encryption In order to achieve the required level of encryption for better security.

It was concluded that the combination of such algorithms led to robust encryption process that not only help to increase the security level of the cloud service but it also kept the network load to the minimum.

3. Security Issues with Self-Signed SSL Certificates

Year of Publication - 2012

Summary:

The paper highlights the difference between third party signed SSL certificates and self-signed SSL certificates and concludes that third party SSL certificates, though are expensive, but in the long run, they are a better alternative. It also delineates a strategic man-in-the-middle attack on a self-signed SSL certificate, which is carried out by replacing certificates on the fly using ARP poisoning and DNS spoofing to redirect users to malicious sites.

4. TEA, a Tiny Encryption Algorithm

Year of Publication – 2019

Summary:

A short program which can run on most machines and encipher safely. It uses an outsized number of iterations instead of a sophisticated program. It is hoped that it can easily be translated into most languages during a compatible way.

Improvement:

The encryption key is using a predetermined algorithm due to which if a person could guess the algorithm being followed for encryption. That person could decipher the message. Through our algorithm we would replace the pre-existing algorithm with a random encrypted.

5. An Advanced Architecture for Securing Data in Cloud

Year of Publication – 2016

Summary:

Symmetric Encryption algorithms play a primary role in information security. So this paper has surveyed the foremost common algorithms and standards available for the encryption of data within the digital form. An encryption algorithm would be useless if it is secure but takes a long time in execution. The field of cryptography is becoming vital in today's times as information security is of absolute importance. Contemporarily more and more sensitive data is being stored on computers and transmitted over the Internet. We need to ensure security and safety of information.

Improvement:

The techniques above paper describes the one which covers the basic encryption techniques and TLS. And doesn't apply any advanced technique to improve the algorithm. We would tackle this situation by adopting the TLS algorithm we apply while coupling it with also our pseudo random generator. To improve privacy.

6.Design and Implementation of key Generation Algorithm for Secure Image

Year of Publication – 2019

Summary:

The advancements in internet and 5G communication technologies enormously use images as a crucial element for effective digital communication. Most of the digital devices support image communication as a prominent way of communication in digital Era. Most of the unauthorized access of digital Era communications results in serious issues in digital data or digital image communications. Hence, Image Security in data communication becomes an egress area of research in Digital Era. This paper is to style and implement an algorithm with a 96 bits cryptographic system. The result analysis carries call at this paper on input images and presents the comparative analysis of our proposed algorithm with the quality algorithms through standard performance evaluation metrics.

7) A Database Record Encryption Scheme Using then RSA Public Key Cryptosystem and Its Master Keys-2006

Summary:

This paper presents two different database-level encryption. Both systems are using the RSA algorithm. First of all, there is a field-based encryption scheme. In all of the fields that are available in the user's master key. After the write mode, the encryption is presented. It uses a master key. This method is used for the sub-groups, and the groups of numbers. See the system design is one of the most difficult tasks. For this purpose, as a general rule, an asymmetrical encryption methods are used. Basically, the cryptographic keys used to sign the data, which are protected areas. To decrypt the keys to be used to read the information. Therefore, give the permissions for the user. A simple database of the method is RSA public / private key method. Multiple RSA key-master keys, for they are two different keys. Encryption is used to represent the type of operation rights. Direct-reading activities are represented by the decryption keys. Number of buttons supported by the database management systems. All rights to plots of land will be accompanied by an RSA public / private key-a master key.

8)Chip-Secured Data Access: Confidential Data on Untrusted Servers-2011

Summary:The ubiquitous computing is to present the data from anywhere, at any time, and this is the only way. It is used to improve the performance of database access. It also needs to

ensure the confidentiality of the information. The number of malware attacks, and security threats are increasing day by day. So, think of a traditional database is at the same time, is at risk of dropping out. We propose a new method, which is called the C-VIRUS (chip to secure access to the data. This will control the access rights of users and data, and ensures the confidentiality of the data. And, also, to act as an intermediary between the client and the encryption of the database. In this post we are included in a smart card. With this, the hardware and the software. This is a guarantee of the attack. In essence, we're using the requests to methods of the evaluation process.

9)A Framework for Efficient Storage Security in RDBMS-2015

Summary:

Step by step, development of E-business is massively expanded. So everyone ought to know about information security and data set security. Some RDBMS stockpiling models, (for example, the N-ary Storage Model) stores records. Offset table is utilized toward the finish of the page. It is utilized to find the beginning stage of record. If question is more touchy, NSM gives enormous execution. It is utilized to move information to and from optional capacity. This is reasonable for online exchange handling

10)Fast, Secure Encryption for Indexing in a Column- Oriented DBMS-2014

Summary:

This paper manages two significant issues. In the first place, security for the encryption. Then, quick execution of inquiry. There are number of techniques bargains something similar. The current strategy guarantees request saving encryption methods are appropriate for information bases. Contrast with different techniques this one is extremely straightforward and brilliant strategy to construct lists. However, it makes issues on direct assaults. In this paper, another segment situated encryption is proposed. It guarantees quick ordering tasks. Square code is applied to scramble minuscule bytes per page. Two figure writings are looks at from the main byte. It thinks about byte by byte. Even however other square codes scramble unit of 8 bytes or more, here it is feasible to encode byte by byte

Improvement:

In spite of the fact that the above strategy does gives us a brief look at what we are attempting to accomplish however the disadvantage is that it piggybacks on a procedure (5G) whis isn't accessible for masses yet. Yet, this paper likewise shows when quicker web opens up for everybody. How our innovation could likewise be put for non military personnel use.

3. Overview of the Work

3.1 Problem description

To mitigate the vulnerability of possible attacks on the process of extraction of data using the conventional ciphertext (which uses a secret key cryptosystem) using a stream cipher process.

In this project we have developed a complete ecosystem of cryptography that uses some of the most optimised algorithms with the minimal effect on the performance to generate quasi-random numbers using generators that are specifically designed for this purpose.

3.2 Working model

The proposed algorithm is designed to work in the Transport layer of the TCP/IP protocol suite. It uses the Diffie-Hellman method of asymmetric key encryption to first establish a session. A session is more like a digital agreement between two communicating parties that have identified each other. The session dissolves once the communication is done.

Our algorithm uses the standard DHA to get a common key established between the two parties. The common key is generated by using public keys from both the sides, and the public key is generated using a mathematical basis, that involves a private key, which, as its name suggests, is private to either party.

This algorithm ensures secrecy of the communication that is about to take place between the two parties, and prevents anyone without the private key from eavesdropping in the conversation, thereby establishing a secure session.

3.3 Design description

Transport Security Layer

Transport Layer Security (TLS), and its now-deprecated predecessor, Secure Sockets Layer (SSL), are cryptographic protocols designed to provide communications security over a computer network. Several versions of the protocols find widespread use in applications such as web browsing, email, instant messaging, and voice over IP (VoIP). Websites can use TLS to secure all communications between their servers and web browsers. The TLS protocol aims primarily to provide privacy and data integrity between two or more communicating computer applications.[2]:3 When secured by TLS, connections between a client (e.g., a web browser) and a server (e.g., wikipedia.org) should have one or more of the following properties:

- ❖ The connection is private (or secure) because symmetric cryptography is used to encrypt the data transmitted. The keys for this symmetric encryption are generated uniquely for each connection and are based on a shared secret that was negotiated at the start of the session. The server and client negotiate the details of which encryption algorithm and cryptographic keys to use before the first byte of data is transmitted. The negotiation of a shared secret is both secure (the negotiated secret is unavailable to eavesdroppers and cannot be obtained, even by an attacker who places themselves in the middle of the connection) and reliable (no attacker can modify the communications during the negotiation without being detected).

- ❖ The identity of the communicating parties can be authenticated using public-key cryptography. This authentication can be made optional, but is generally required for at least one of the parties (typically the server).
- ❖ The connection is reliable because each message transmitted includes a message integrity check using a message authentication code to prevent undetected loss or alteration of the data during transmission.

TLS & CRYPTOGRAPHY

- Cryptography involves the use of keys, which are strings of bits used pre communication for securely transferring information/data.
- Two types: Symmetric key cryptography and Asymmetric key cryptography
- Symmetric key cryptography uses a single key both on client side and server side; computationally fast but vulnerable.
- Asymmetric key cryptography uses a pair of keys; a public key and a private key both on server side as well as client side. Both the keys are mathematically related; secure but computation wise, it is slow.
- TLS hence uses both types of cryptography; a single key (called session key) for encryption and decryption but to generate this session key it uses asymmetric algorithms and involves a private and public key.

TLS HANDSHAKE PROCESS

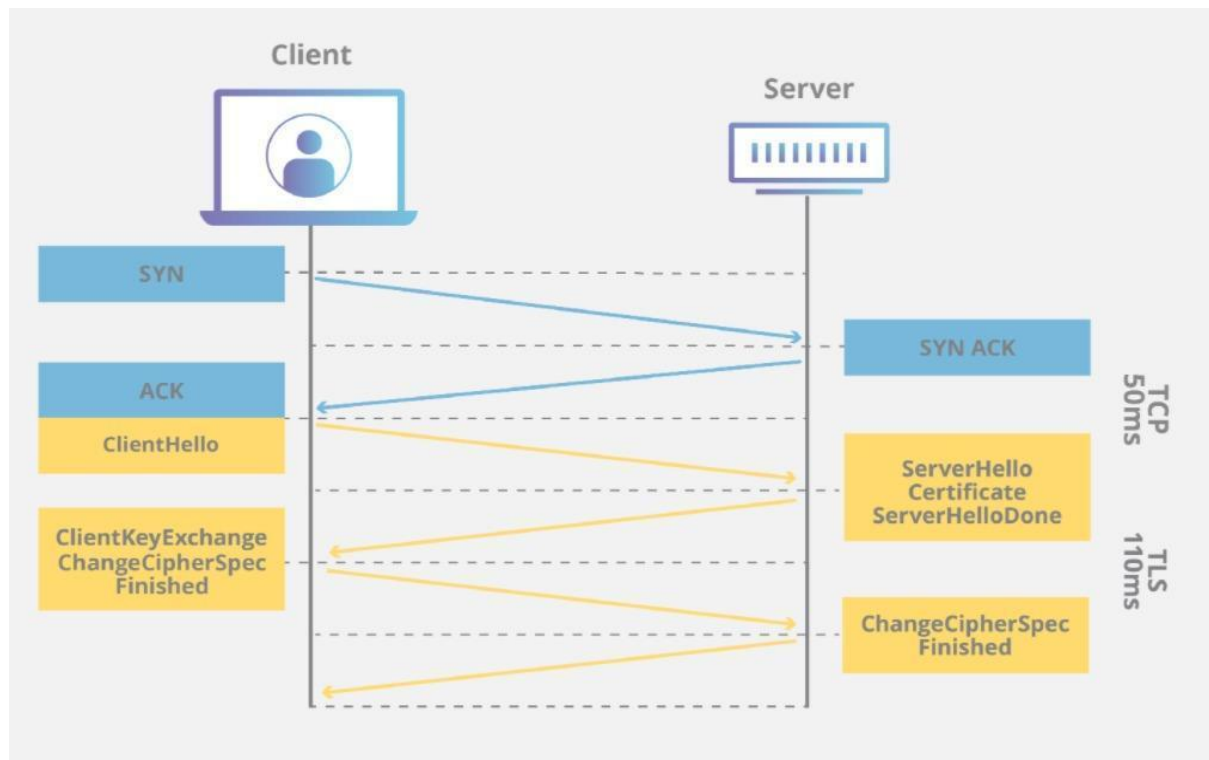


FIG 1 : workflow of TLS Handshake Process

Client-authenticated TLS handshake

This handshake is similar as the essential TLS handshake, however the customer is validated also. The fundamental contrast is that after the worker sends its Certificate message, it likewise sends a Certificate Request message, requesting the customer's endorsement. When the worker is done, the customer sends its authentication in a Certificate message.

The customer at that point sends its Client Key Exchange message, very much like in the fundamental TLS handshake. This is trailed by the Certificate Verify message, which incorporates the customer's computerized signature. Since it is determined from the customer's private key, the worker can confirm the mark utilizing the public key that was sent as a feature of the customer's advanced declaration. The remainder of the Client-confirmed TLS handshake tracks with similar lines as the fundamental TLS handshake.

Abbreviated TLS handshake

When a handshake has occurred, TLS permits a significant part of the cycle to be removed by utilizing a contracted handshake all things considered. These handshakes utilize the meeting ID to interface the new association with the past boundaries.

The customer and worker then both utilize the premaster secret and the irregular numbers that they sent toward the beginning of the correspondence to think of the expert mystery. When the expert key has been determined, it is utilized to concoct either four or six separate keys. These are the:

Customer compose MAC key – This key is utilized by the worker to check the respectability of information that was sent by the customer.

Worker compose MAC key – The worker compose MAC key is utilized by the customer to check the respectability of information that was sent by the worker.

Customer compose encryption key – The worker utilizes this key to encode information that was sent by the customer.

Worker compose encryption key – The customer utilizes this key to scramble information that was sent by the worker.

Customer compose IV key – The customer compose IV key is utilized by the worker in AEAD figures, however not when other key trade calculations are utilized.

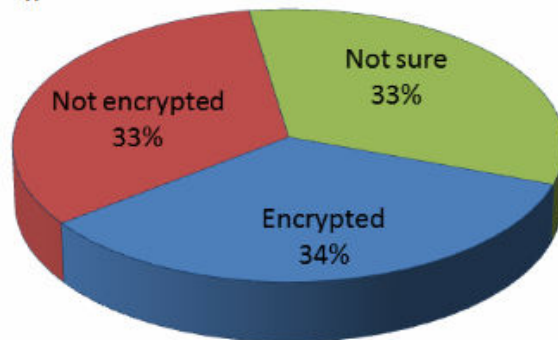
Worker compose IV key – Similarly, this key is utilized by the customer in AEAD figures, however not when other key trade calculations are utilized.

Requirement Analysis

Transport Layer Security (TLS) encodes information sent over the Internet to guarantee that busybodies and programmers can't perceive what you communicate which is especially

helpful for private and touchy data like passwords, charge card numbers, and individual correspondence. This page clarifies what TLS is, the means by which it works, and why you ought to convey it. The essential need of any cutting edge cryptographic calculation is to get or create an adequately irregular boundary which can't be effortlessly decided without the information identified with a specific private snippet of data.

When using your personal electronic device for work, are your organization's data and/or files encrypted?



As you could see a lot of systems we use today are not following any safe encryption which leads to the system being compromised. The need becomes even more apparent when the growth in the computational power available to the eavesdropper is considered. More is the level of haphazardness of this boundary, harder it is for the assailant to create the plaintext from the given ciphertext through a Brute-Force assault on the code.

Functional Requirement

- Pseudo random generator for generating same set of keys on both side.
- Highly Secure.
- Reduce the chance for the capturing of the key.

Non Functional Requirement

- Algorithm should be robust Execution time for Algorithm should be minimal to improve connection speed.

3.4 Algorithms

Once the session is established using DHA, both the communicating parties will have a common key. We use this key for starting a communication.

1. Let the common key be the initial seed value.
2. Maintain the index of communication for further communication as a session variable, incrementing it each time a communication happens.
3. To encode a message, do the following:
 - a. Generate a significantly large number
 - b. Fill an array of constant length (pre decided) with ASCII characters, ensure the randomness of ASCII characters by making sure the number in step 3a is big enough
 - c. For each character in the original message, XOR it with the array items containing ASCII characters, using the array in a cyclic manner.
4. Convert the encrypted characters into binary and store it suitably in a file.
5. This file can be transmitted to the receiver. To decrypt a message, follow the above process in reverse. You would get the original message because of the property of the XOR operation that: $(A \text{ XOR } B) \text{ XOR } B = A$

4. Implementation

4.1 Description of Modules/Programs

Module 1 : Establishing a session before any communication happens

In this section, we use the standard cryptographic placeholder names Alice and Bob to identify the receiver and sender respectively. We first ask the program user to identify themselves as Alice or Bob.

1.1 Alice's identity established as the receiver Upon choosing Alice, we are asked to enter a prime number. The algorithm then checks for a generator in accordance with the DHA. A generator should be able to generate all the numbers between 0 and the prime number (both exclusive). The program then randomly chooses one of the generators from an array that stores the generator from the previous step. We then enter Alice's private key, after which our public key is generated, which is to be shared with Bob, and finally we enter Bob's public key that we receive. This way, Alice's identity is established.

1.2 This way, Alice's identity is established. Bob's identity established as the sender On Bob's side, the process is simpler, all we have to do is enter the prime base, the generator that was chosen on Alice's side. We then choose our private key, and enter Alice's public key received. As evident, we have a common key that is generated independently on both sides. This concludes the session establishing process and the communication can now happen.

Module 2 : The communication

We are then given the option for encryption (send a message), decryption (receive a message), and to exit (kill the session). The following part is common to both the sides.

2.1 Encryption: On choosing encryption, we are asked to enter a message to send. The program then uses a random number generated to encrypt the message and outputs the encoded message. All the symbols in the encoded message are then converted to binary strings that can be transmitted. Here's the message after encryption and binary conversion is shown.

2.2 Decryption: Upon the receipt of the encoded file, if the user chooses the option to decrypt and provides the encoded message, the program successfully decodes and outputs the original message. As evident, we have successfully decoded the initial message, thereby completing the communication.

2.3 The Randomness The security benefit that our algorithm has could be tested in the same session by sending the same message again, only to see the encoded message being different from the first one. The encoded message is different than the one that was previously produced. Upon decoding this message, we get the original message.

4.2 Techniques/ Tools/ source code

Techniques

Complete Cryptosystem: This is an encryption with a public key, there is a set of cryptographic algorithms needed to implement a particular security service, most of all, for achieving confidentiality (encryption). Typically, a cryptosystem consists of three algorithms: one is to generate keys for encryption and another for decryption.

Symmetric Encryption: Symmetric encryption is a type of encryption, which uses only one key (the public key) to encrypt and decrypt electronic data. In fact, it is to communicate using symmetric encryption, it is necessary to share the most important thing is to be used in the decryption process.

DHA : The Diffie-Hellman key exchange is a method of secure communication the encryption of data over a public channel and was one of the first public-key protocols as

conceived by Ralph Merkle and named after Whitfield Diffie and Martin Hellman. DH is one of the earliest practical examples of public key exchange implemented in the field of cryptography. Diffie and Hellman, these being the earliest known piece of paper, which proposed the idea of a private key and a corresponding public key.

Pseudo Random Generator: A pseudorandom number generator, also known as a deterministic random bit generator, is an algorithm for generating a sequence of numbers, properties approximate the properties of sequences of random numbers.

Algorithm for Pseudo-Random Number Generator

1. Take input of a number , that is a seed or key.
2. that seed should be in a sequence of mathematical operations to generate the result. ...
3. Use that resulting random number as the seed for the next iteration.
4. Repeat the process to calculate randomness.

Stream Cipher: A stream cipher is a symmetric key cipher where plaintext digits are combined with a pseudorandom cipher digit stream (keystream). In a stream cipher, each digit in the plaintext is encrypted by the corresponding number of the main stream is to give the digit of the ciphertext stream.

Forward Secrecy: In cryptography, forward secrecy which is also known as PFS, it has a special key negotiation protocol, which provides you with the assurances that session keys are going to be endangered, although, in the long-term secrets to be used for the exchange of session keys will be compromised.

Tools used

Xcode

Xcode is Apple's integrated development environment (IDE) for macOS, used to develop for developing software for macOS, iOS, iPadOS, watchOS, and tvOS. It was first released in 2003, the latest stable release is version 12.5, which was released on April 26, 2021, and is available on the Mac App Store, for free, on macOS, the Big Sur of the users. Registered developers can download an early version of the earlier versions of packages via Apple's developer. Xcode includes command-line tools (CLT), which enables UNIX-style

development via Terminal program on the os. They can also be accepted, but to identify the main Ideas.

Source code

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#define Path "C:/Users/ritwi/Desktop/transmission.txt"
#define MAX_ITERATIONS 50

int no_pr, proots[10], keylen, communication_index = 0;
unsigned long long int a, b, g, private_key, partner_public_key, my_public_key,
common_key, seed;
char binary_buffer[8], key[MAX_ITERATIONS], message[400];

void DHA_for_Alice();
void DHA_for_Bob();
void generate_publickey();
void generate_commonkey();
void compute_primitive_roots();
int is_prime(unsigned long long int);
int make_a_choice();
unsigned long long int order(int);
unsigned long long int exp_mod(unsigned long long int, unsigned long long int);

void begin_symmetric_encryption();
void message_encode();
void message_decode();
```

$$\}$$

```

void DHA_for_Alice()
{
    printf("\n=====: Start of DHA :=====\\n");
    printf("Enter the prime base  : ");
    scanf("%llu", &b);
    if (!is_prime(b))
    {
        printf("-----");
        printf("\\n\\nThe base entered is not a prime number !\\nPlease try again.\\n\\n\\n");
        exit(0);
    }
    compute_primitive_roots();
    system("cls");
    printf("\n=====: Start of DHA :=====\\n");
    printf("Choose a prime base :  %llu\\n", b);
    printf("The generator  :  %llu", g);
    printf("\\n-----< Confidential >-----\\n");
    printf("Enter the private key  : ");
    scanf("%llu", &private_key);
    printf("\\n-----< Confidential >-----\\n");
    generate_publickey();
    generate_commonkey();
    printf("\n=====: End of DHA :=====\\n");
}

```

```

void DHA_for_Bob()
{
    printf("\n=====: Start of DHA
:=====\\n");
    printf("Enter the prime base : ");
    scanf("%llu", &b);

```

```

if (!is_prime(b))
{
    printf("-----");
    printf("\n\nThe base entered is not a prime number! Please try again.\n\n\n");
    exit(0);
}
printf("The generator  : ");
scanf("%llu", &g);
printf("\n-----< Confidential >-----\n");
printf("Enter the private key  : ");
scanf("%llu", &private_key);
printf("\n-----< Confidential >-----\n");
generate_publickey();
generate_commonkey();
printf("\n=====: End of DHA
:=====\n");
}

```

```

int is_prime(unsigned long long int n)
{
    int i;
    if (n <= 1)
        return 0;
    else if (n <= 3)
        return 1;
    if (n % 2 == 0 || n % 3 == 0)
        return 0;
    for (i = 5; i * i <= n; i += 6)
    {
        if (n % i == 0 || n % (i + 2) == 0)
            return 0;
    }
}

```

```

    return 1;
}

void compute_primitive_roots()
{
    int i, j = 0;
    printf("-----");
    printf("\n\nChoice of generator under progress....\nPlease wait...\n");
    for (i = 2; i < 30 && i < b; i++)
        if (!j && i >= 15)
        {
            printf("-----");
            printf("\nThe given prime is not suitable for the algorithm. Please enter another
one.");
            printf("n-----\n\n\n");
            exit(0);
        }
        else if (j < 4 && order(i) == (b - 1))
        {
            proots[j++] = i;
        }
    printf("\n");
    no_pr = j;
    printf("\n-----");
    printf("\nprimitive roots : ");
    for (i = 0; i < no_pr; i++)
    {
        printf("%d ", proots[i]);
    }
    g = proots[make_a_choice()];
    printf("\nThe generator chosen : %llu", g);
    printf("\n-----\n");
}

```



```
}
```

```
void generate_publickey()
```

```
{  
    my_public_key = exp_mod(g, private_key);  
    printf("The public key generated   : %llu", my_public_key);  
}
```

```
void generate_commonkey()
```

```
{  
    printf("\nThe public key received   : ");  
    scanf("%llu", &partner_public_key);  
    common_key = exp_mod(partner_public_key, private_key);  
    printf("the common key generated   : %llu", common_key);  
}
```

```
int make_a_choice()
```

```
{  
    int x;  
    srand((unsigned)time(NULL));  
    x = rand();  
    x %= no_pr;  
    return x;  
}
```

```
unsigned long long int order(int x)
```

```
{  
    int ptr = 1;  
    a = 0;  
    while (ptr)  
    {  
        a++;  
    }
```

```

        if (a >= 300000000)
            break;
        if (exp_mod(x, a) == 1)
            ptr = 0;
    }
    printf("\n\"%02d\" checked.", x);
    return a;
}

```

```

unsigned long long int exp_mod(unsigned long long int base, unsigned long long int exp)
{
    unsigned long long int ans = 1;
    while (exp != 0)
    {
        if (exp % 2 == 1)
        {
            ans *= base;
            ans %= b;
        }
        base *= base;
        base %= b;
        exp /= 2;
    }
    return ans;
}

```

```

void begin_symmetric_encryption()
{

    int choice;
    system("cls");
    while (1)

```

```

{
    printf("\n===== Symmetric Encryption =====\n");
    printf("Enter your choice:\n1. \tEncryption\n2. \tDecryption\n3. \tExit\n");
    scanf("%d", &choice);
    switch (choice)
    {
        case 1:
            message_encode();
            break;
        case 2:
            message_decode();
            break;
        case 3:

printf("=====
=====");
            exit(0);
            break;
        default:
            printf("\nEnter a valid choice");
            break;
    }
    simulate_delay(3);
}
}

void message_encode()
{
    int i, j = 0;
    char ch;
    FILE *fp = fopen(Path, "w");
    printf("=====: Encryption :=====\n");

```

```

printf("Enter the message:\n");
printf("-----\n");
scanf(" %[^\n]s", message);
get_a_random_key();
printf("-----");
printf("\nEncoded Message:\n");
for (i = 0; message[i] != '\0'; i++)
{
    ch = message[i] ^ key[i % (keylen / sizeof(char))];
    printf("%c", ch);
    convert_char_to_binary(ch);
    fputs(binary_buffer, fp);
    fputs("\n", fp);
}
printf("\n");
printf("-----\n");
fclose(fp);
}

void message_decode()
{
    int ctr = 0;
    FILE *fp = fopen(Path, "r");
    char ch, str[8];
    printf("=====: Decryption :=====\n");
    while (fgets(str, 9, fp) != NULL)
    {
        ch = convert_binary_to_decimal(str);
        printf("%c", ch);
        ch = ch ^ key[ctr % (keylen / sizeof(char))];
        message[ctr] = ch;
        ctr++;
    }
}

```

```

    }
    printf("\n-----");
    printf("\nDecoded Message :\n");
    printf("%s\n", message);
    printf("-----\n");
    fclose(fp);
}

void get_a_random_key()
{
    int i = 0;
    a = 0;
    while (i < MAX_ITERATIONS - 1)
    {
        if (!a)
        {
            b = get_a_random_number(++communication_index, b);
            a = b;
        }
        key[i++] = a % 100;
        a /= 100;
    }
    key[i++] = a % 100;
    keylen = i;
}

unsigned long long int get_a_random_number(int i, unsigned long long int P)
{
    unsigned long long int next, result;
    seed = seed % 1000000000000;
    next = seed ^ i;
    result = P;

```

```

    next <<= 7;
    next = next % 1000000000000;
    result ^= next;
    result += i;
    result = result % 1000000000000;
    seed = next;
    return result;
}

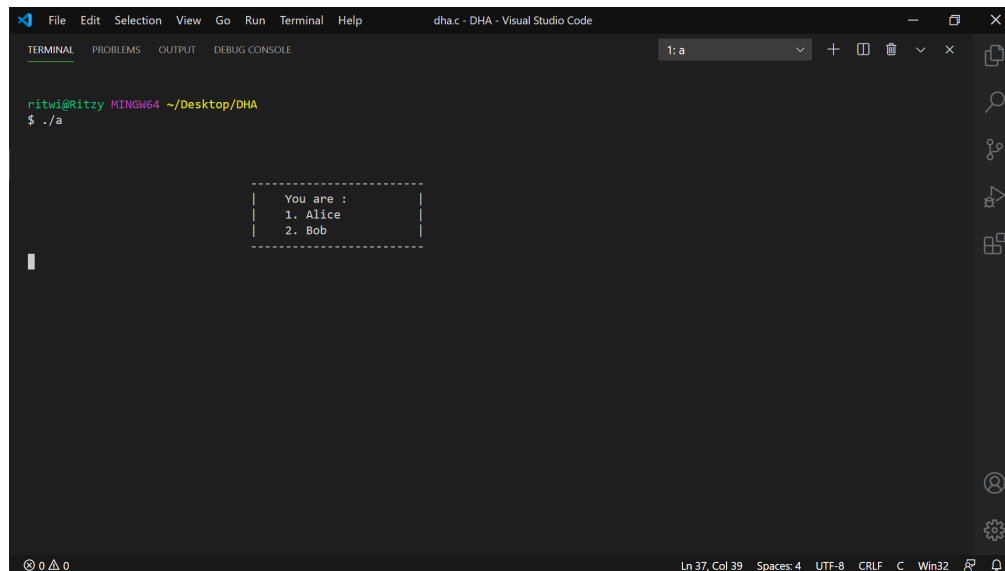
void convert_char_to_binary(char c)
{
    int j;
    for (j = 6; j >= 0; j--)
        binary_buffer[6 - j] = (c & (1 << j)) ? '1' : '0';
}

int convert_binary_to_decimal(char str[])
{
    int i, d = 0;
    for (i = 0; i < 7; i++)
        d += (str[i] - '0') * pow(2, (6 - i));
    return d;
}

void simulate_delay(int seconds)
{
    int ms = 1000 * seconds;
    clock_t start_time = clock();
    while (clock() < start_time + ms);
}

```

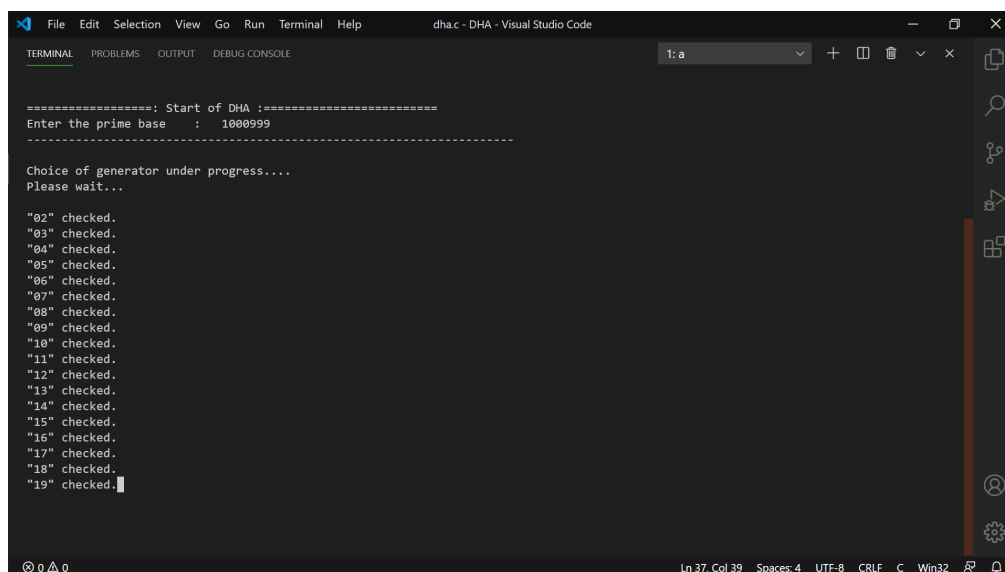
4.3 Execution of the project



```
ritwi@Ritzy MINGW64 ~/Desktop/DHA
$ ./a

-----
You are :
1. Alice
2. Bob
-----
```

Fig 4.3.1. Establishing session

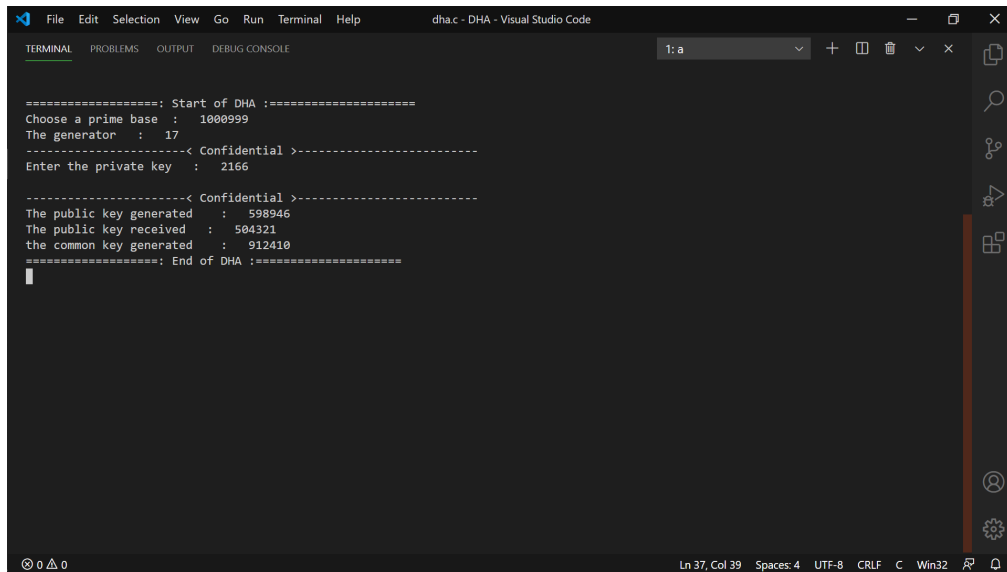


```
===== Start of DHA :=====
Enter the prime base : 1000999
-----

Choice of generator under progress....
Please wait...

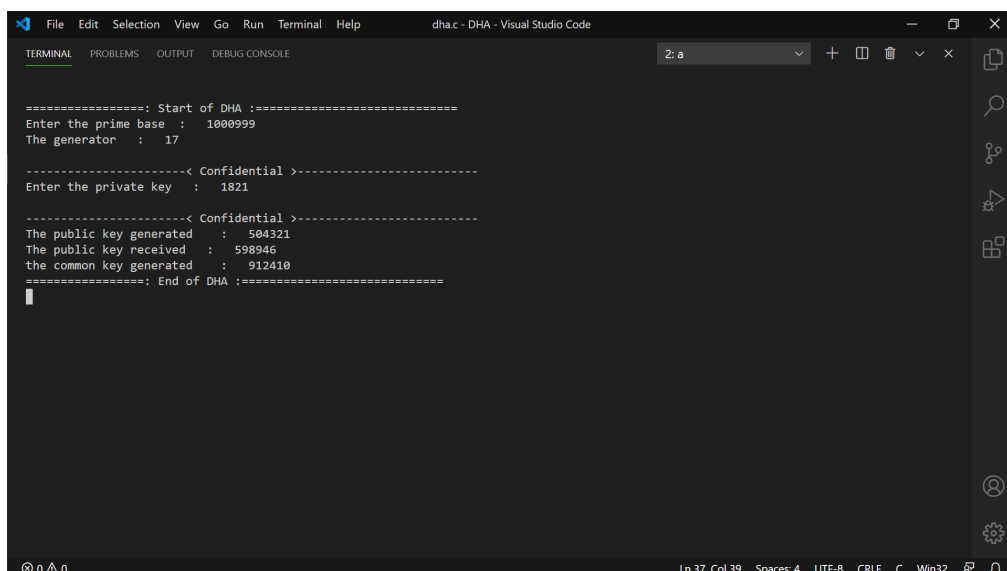
"02" checked.
"03" checked.
"04" checked.
"05" checked.
"06" checked.
"07" checked.
"08" checked.
"09" checked.
"10" checked.
"11" checked.
"12" checked.
"13" checked.
"14" checked.
"15" checked.
"16" checked.
"17" checked.
"18" checked.
"19" checked.
```

Fig 4.3.2 Receiver side identity check



```
File Edit Selection View Go Run Terminal Help dha.c - DHA - Visual Studio Code
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE 1: a
=====: Start of DHA :=====
Choose a prime base : 1000999
The generator : 17
-----< Confidential >-----
Enter the private key : 2166
-----< Confidential >-----
The public key generated : 598946
The public key received : 504321
the common key generated : 912410
=====: End of DHA :=====
```

Fig 4.3.3 Data entry initiated for process



```
File Edit Selection View Go Run Terminal Help dha.c - DHA - Visual Studio Code
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE 2: a
=====: Start of DHA :=====
Enter the prime base : 1000999
The generator : 17
-----< Confidential >-----
Enter the private key : 1821
-----< Confidential >-----
The public key generated : 504321
The public key received : 598946
the common key generated : 912410
=====: End of DHA :=====
```

Fig 4.3.4. Sender side details

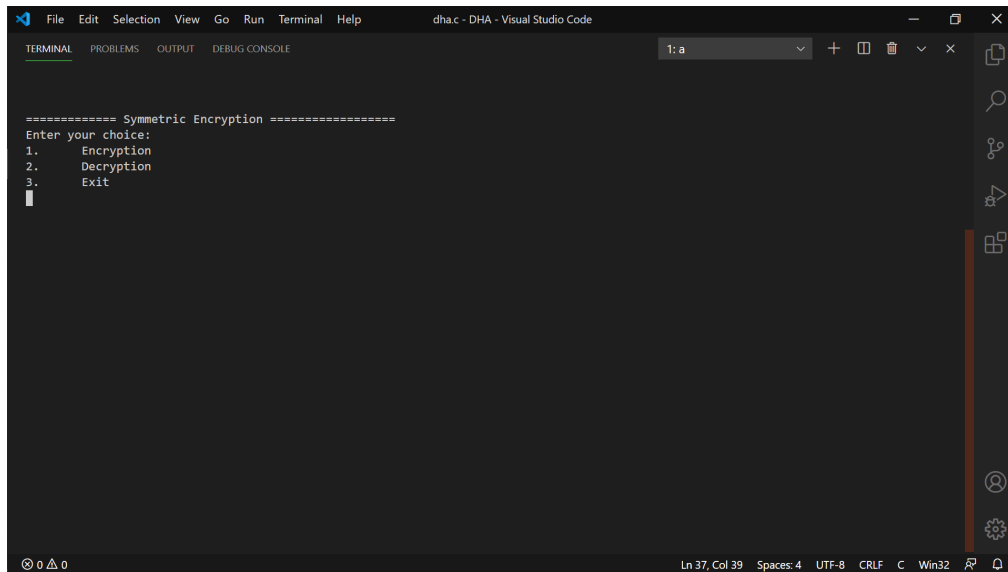


Fig 4.3.5 Communication channel overview

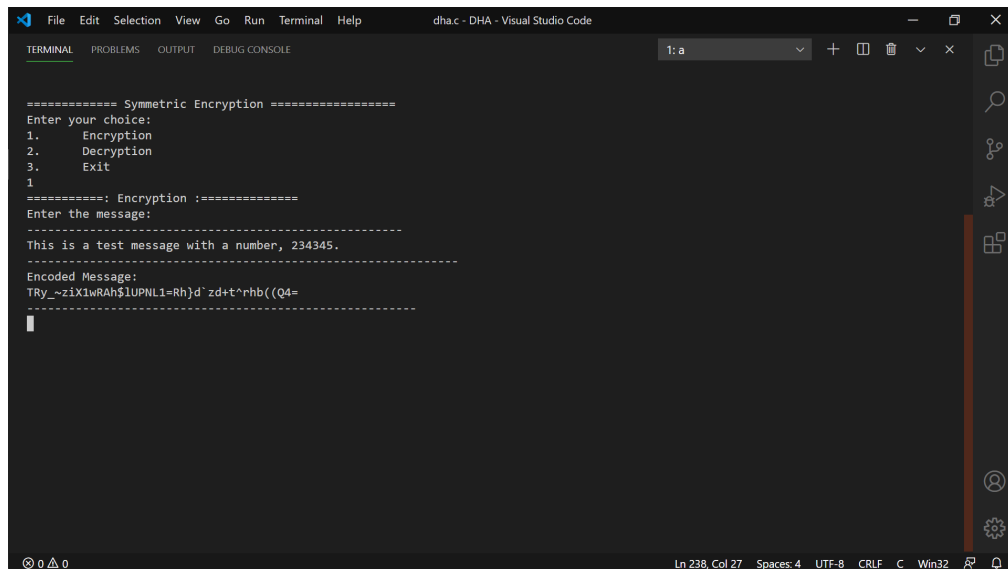


Fig 4.3.6 Encryption of message

```
File Edit Selection View Go Run Terminal Help dha.c - DHA - Visual Studio Code
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE 1: a
===== Symmetric Encryption =====
Enter your choice:
1. Encryption
2. Decryption
3. Exit
2
=====: Decryption :=====
Try_~ziX1wRAh$1UPNLI=Rh}d' zd+t^rhb((Q4=
-----
Decoded Message :
This is a test message with a number, 234345.
-----
Ln 238, Col 27 Spaces: 4 UTF-8 CRLF C Win32
```

Fig 4.3.7 Decryption of encrypted message

```
File Edit Selection View Go Run Terminal Help dha.c - DHA - Visual Studio Code
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE 1: a
===== Symmetric Encryption =====
Enter your choice:
1. Encryption
2. Decryption
3. Exit
1
=====: Encryption :=====
Enter the message:
-----
This is a test message with a number, 234345.
-----
Encoded Message:
7mj$Hw|w+5;d/Q5T6'de&h
TE3Ddj0}b
-----
Ln 238, Col 27 Spaces: 4 UTF-8 CRLF C Win32
```

Fig 4.3.8 Randomness

```

===== Symmetric Encryption =====
Enter your choice:
1. Encryption
2. Decryption
3. Exit
2
===== Decryption :=====
7mj$Hw|w+5;d/Q5T6'de&h
-----
Decoded Message :
This is a test message with a number, 234345.
-----

```

Fig 4.3.9 Decryption process authenticity with randomness

4.4 Results analysis in the form of graphs and table

In the following discussion, we analysed the results obtained from a basic 'C' implementation of the algorithm described above. The XOR cipher used here works on the binary ASCII values of the characters present within the message or the key. it's possible here that the generated character, which belongs to the key or the result, might not be a printable entity thanks to the character of the assigned character. Thus, rather than just printing the characters of the key or the ciphertext directly, we also provide their corresponding decimal ASCII value. For the characters that can't be printed, we've used standard 3 alphabet codes for representation.

At the very beginning, the 'seed' value was set to the common key established during a demo session. For the first instance of communication, the value of the communication index is 1. Sender calls the `message_encode` function with the message mentioned below

“Mayday,mayday come quick”

Following this, a call was made to the function ‘get_a_random_key’. It in-turn made repetitive calls to the function ‘get_a_random_number’, whenever ‘a’ became zero. Each call returned an outsized random number, which was then decomposed into two-digit numbers. Each of those two-digit numbers were then went to obtain the corresponding ASCII characters. This was done until the length of the compiled key was shorter than the required length in ‘MAX_ITERATIONS’.

The encrypted message that was produced was

0101101

1110100

0101011

1101101

1111101

1001100

0011101

0001101

1000010

0110000

1000101

1010010

1111010

0001110

1100001

1101011

0111000

0100110

0100000

1111001

1101001

1110010

1010111

0111000

Using the decryption algorithm, the result produced at the other side was

“Mayday,mayday come quick”

The following results that are observed as the output of the encryption and decryption algorithm assure that there is a complete attention of the data during the entire process and the performance of the algorithm was optimised according to the latest benchmarking tools that are being used in order to compute the running time and the analysis of the complexity of that program that has been carefully designed for the most optimised balance possible for a better algorithmic computation and performance. It has to be noted that the algorithm handles the multiple request regarding the encryption of the data that is fed into the system by the user in the most secure manner possible. To eliminate any chance of compromise of the data that is being supplied into the algorithm, every time user gives the input for the encryption algorithm to work it changes the encryption codes that are previously generated in an attempt to keep the entire process as randomised as possible in order to eliminate any possibility of tampering of the data by any third party intervention during the entire process of the computation initiated by the algorithm.

The obtained key was then used for encryption, by performing a character-wise xor operation between the key and the input message in a cyclic fashion, until all the characters of the message were encoded. the generated ciphertext was as follows:

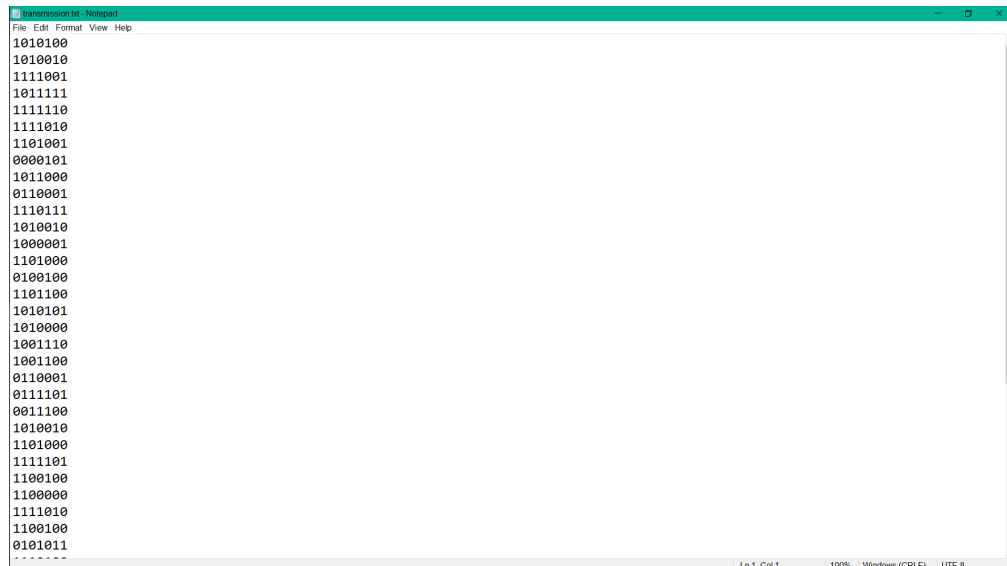
Characters	N	c	-	NUL	p	O	B	s	(.	b	K	l
ASCII	78	99	45	00	112	111	66	115	40	46	98	75	108

Characters	d	w	f	w	<	~	DC3	/	h	SP	B	x	BS
ASCII	100	119	102	119	60	126	19	47	104	32	66	120	08

Characters	\$	NUL	RS	e	5	NUL	#	t	w	>	B	BEL	f
ASCII	36	00	30	101	53	00	35	116	119	62	66	07	115

Characters	U	;	E]	L	X	,						
ASCII	85	59	69	93	76	120	44						

This ciphertext was then stored in a file named ‘transmission.txt’, in the binary format. The contents, as stored in the file were:



```

transmission.txt - Notepad
File Edit Format View Help
1010100
1010010
1111001
1011111
1111110
1111010
1101001
0000101
1011000
0110001
1110111
1010010
1000001
1101000
0100100
1101100
1010101
1010000
1001110
1001100
0110001
0111101
0011100
1010010
1101000
1111101
1100100
1100000
1111010
1100100
0101011

```

This resulted in a successful decryption of the message and the following plaintext was obtained:

Characters	_	ETB	/	-	%	SOH	M	SP	L	@		ACK	G
ASCII	95	23	47	45	37	01	77	32	76	64	93	06	71

Characters	SO	EM	CR	DLE	STX	EM	6	F	CAN	9	ACK	'	+
ASCII	14	25	13	16	02	25	54	70	24	57	06	39	43

Characters	"	F	%	STX	CR	SYN	SI	P	FS	BEL	SP	NAK	7
ASCII	34	70	37	02	13	22	15	80	28	07	32	21	55

Characters	US	STX	HT	4	CAN	W	CR	BEL	_	3	CR	NUL	
ASCII	31	02	09	52	24	87	13	07	95	51	13	00	

The corresponding ciphertext generated after encryption was:

Characters	BS	r	SI	L	W	D	m	L	#	#	<	r	"
ASCII	08	114	15	76	87	100	109	76	35	35	60	114	34

Characters	j	9	l	d	"	l	EOT	t	6	FF	H	VT	GS
ASCII	106	57	108	100	34	49	04	116	54	12	72	11	29

Characters	ETB	h	GS	G	\$	8	/	DC3	s	j	E	5	F
ASCII	23	104	29	71	36	56	47	19	115	106	69	53	70

Characters	j	k	j	_	t	.	,						
ASCII	106	107	106	95	116	46	44						

5. Conclusion and Future Scope

Conclusion

The performance of our algorithm along with the assumptions for the possible trade off balance has yielded quite good results which in our consideration is very reasonable and justifiable. The secure transmission of the encrypted message over the algorithm is able to be

decoded at the other side using the secure parametric transmission through our representation of the algorithms that are well established by the industry has successfully concluded that the algorithm can be used in a production environment as far as the production performance requirements are concerned and the end user will be guaranteed of the any possible tempering of the message in the medium of connection or at the end of the receiver or the sender side. The original implementation of the protocol was robust enough in order to provide a basic Framework for the security of the data that are being transmitted through the medium but it needed certain augmentation to the implementation we should always ensure that the transmission has to be guaranteed by a certain encoding and decoding algorithm that is reasonably described in our situation which ensures the data to be secured from any man in the middle or any of the attacks that are well established.

Future Scope

The project has been made keeping in mind that the components that are involved in it can be scaled up according to the requirement of a particular system and the different protocols that can be used in tandem to it. Secure transmission of the messages is one of the crucial requirement of any secured facility that involve such kind of communications like defence organisation softwares, encrypted messaging platforms and a lot more of applications that use message passing or the conversations that includes sensitive information which should not be e conveyed to anyone other than the sender and the receiver who are involved in the entire process. With this preposition in mind we have created the platform in order to integrate it in a flexible manner of whether it can be a standalone application that involves integrating the algorithm as one of the component in the pipeline or it can be integrated in a system as a module component that can be tweaked in a manner possible to control the behaviour of the algorithm according to the requirement of the situation. We have a plenty of real time applications that we see today including the social messaging platforms and integrated chatbots that involve such kind of tunneling of the virtual communication lines that required encrypted pipelines in order to convey the message in a secured way which makes our algorithm perfect to be integrated with such real-time platforms along with other algorithms that support for the same cause. The output of the system can be used as a starting point of any other system that utilizes such encrypted messages in order to be passed among the

components of the various systems and it can be easily recorded by our algorithm in a reasonable amount of time.

6.References

- [1] Chin-Chen Chang and Chao-Wen Chan, A database record encryption scheme using the RSA public key cryptosystem and its master keys, ICCNMC '03: Proceedings of the 2003 International Conference on Computer Networks and Mobile Computing (Washington, DC, USA), IEEE Computer Society
- [2] Luc Bouganim and Philippe Pucheral, Chip-secured data access: confidential data on untrusted servers, VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases, VLDB Endowment, 2002, pp. 131–142.
- [3] BalaIyer, Sharad Mehrotra², Einar Mykletun, GeneTsudik, and Yonghua Wu, A Framework for Efficient Storage Security in RDBMS, Advances in Database Technology - EDBT 2004 Volume 2992 of the series Lecture Notes in Computer Science pp 147-164
- [4] Tingjian Ge and S. Zdonik, Fast, secure encryption for exing in a column-oriented DBMS, Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on, 2007, pp. 676–685.
- [5] Alan O. Freier, Philip Karlton, and Paul C. Kocher, The SSL protocol version 3.02, 1996.
- [6] T. Dierks and E. Rescorla, The TLS protocol version 1.2, 2006.

- [7] Kumaravel, A., Dutta, P., Application of Pca for context selection for collaborative filtering, Middle - East Journal of Scientific Research, v-20, i-1, pp-88-93, 2014.
- [8] BrinthaRajakumari, S., Nalini, C., An efficient data mining dataset preparation using aggregation in relational database, Indian Journal of Science and Technology, v-7, i-, pp-44-46, 2014.
- [9] Udayakumar, R., Khanaa, V., Saravanan, T., Saritha, G., Retinal image analysis using curvelet transform and multistructure elements morphology by reconstruction, Middle - East Journal of Scientific Research, v-16, i- 12, pp-1781-1785, 2013.
- [10] Khanaa, V., Thooyamani, K.P., Using triangular shaped stepped impedance resonators design of compact microstrip quad-band, Middle - East Journal of Scientific Research, v-18, i-12, pp-1842-1844, 2013.
- [11] Thamotharan, C., Prabhakar, S., Vanangamudi, S., Anbazhagan, R., Anti-lock braking system in two wheelers, Middle - East Journal of Scientific Research, v-20, i-12, pp-2274-2278, 2014.
- [12] Vanangamudi, S., Prabhakar, S., Thamotharan, C., Anbazhagan, R., Design and fabrication of dual clutch, Middle - East Journal of Scientific Research, v-20, i- 12, pp-1816-1818, 2014.
- [13] Vanangamudi, S., Prabhakar, S., Thamotharan, C., Anbazhagan, R., Design and calculation with fabrication of an aero hydraulic clutch, Middle - East Journal of Scientific Research, v-20, i-12, pp-1796-1798, 2014.
- [14] Saravanan, T., Raj, M.S., Gopalakrishnan, K., VLSI based 1-D ICT processor for image coding, Middle - East Journal of Scientific Research, v-20, i-11, pp- 1511-1516, 2014.
- [15] Hasan Kadhemi, Toshiyuki Amagasa, Hiroyuki Kitagawa, A Novel Framework for Database Security based on Mixed Cryptography, Fourth International Conference on Internet and Web Applications and Services 2009
- [16] Authors: Samba Sesay, Zongkai Yang, Jingwen Chen and Du Xu, A secure database encryption scheme, Consumer Communications and Networking Conference, 2005. CCNC. 2005 Second IEEE, Pg:49 - 53
- [17] Einar Mykletun and Gene Tsudik, Incorporating a Secure Coprocessor in the Database-as-a-Service Model, Innovative Architecture for Future Generation High-Performance Processors and Systems, 2005

- [18] Udayakumar R., Kaliyamurthie K.P., Khanaa, Thooyamani K.P., Data mining a boon: Predictive system for university topper women in academia, World Applied Sciences Journal, v-29, i-14, pp-86-90, 2014.
- [19] Kaliyamurthie K.P., Parameswari D., Udayakumar R., QOS aware privacy preserving location monitoring in wireless sensor network, Indian Journal of Science and Technology, v-6, i-SUPPL5, pp-4648-4652, 2013.
- [20] Brintha Rajakumari S., Nalini C., An efficient cost model for data storage with horizontal layout in the cloud, Indian Journal of Science and Technology, v-7, i-, pp-45-46, 2014.