

**IMPROVED 2-LEVEL DATA SECURITY APPROACH FOR UBER
TRANSPORT MANAGEMENT SYSTEM (UTMS)**

BY

**ILIYASU, Mohammed Awwal
MTech/SICT/2018/8832**

**A THESIS SUBMITTED TO THE POSTGRADUATE SCHOOL
FEDERAL UNIVERSITY OF TECHNOLOGY, MINNA, NIGERIA IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
AWARD OF THE DEGREE OF MASTER OF TECHNOLOGY
IN COMPUTER SCIENCE**

SEPTEMBER, 2021

ABSTRACT

Transport Management Systems are application softwares designed to optimize and manage various transport operations, including inbound and outbound transport operations. Transport Management System has a wide range of features and functions, these include booking rides, GPS vehicle trackers used to track the locations of vehicles and their drivers, calculating transportation costs, optimizing travel routes and vehicle loads, freight audits, creating and optimizing shipping plans. Uber is a Transport Management Systems application that connects drivers with riders. Uber's rising travel demand has led to several transportation safety issues and currently, users of Uber Transport Management Systems (UTMS) are more concerned with data secrecy than convenience of transportation services. This research aims to improve on a 2-level data security approach to secure confidential information during communication between two or more clients and data in a database of an Uber Transport Management System. The research adopted DNA cryptography method to store, process and transmit information in a mangled form using Shift Cipher and DNA One Time Pad encryption techniques. To implement the 2 level data security approach, Shift Cipher technique was used as the first security level and DNA One Time Pad (OTP) encryption technique for the second security level. DNA OTP and Shift Cipher encryption approaches were used to secure and unveil the meaning of information processed via Transport System Security module for all operations (Encryption and Decryption). The research findings revealed that Transport System Security module has better processing speed for both encryption and decryption operation with an improve 2 level data security. The experimental result shows that information below 200 characters can be encrypted in 0.02 millisecond while information having less 300 characters can be decrypted in 0.02 millisecond.

TABLE OF CONTENTS

Cover Page	
Title Page	i
Declaration	Error! Bookmark not defined.
Certification	Error! Bookmark not defined.
Acknowledgements	Error! Bookmark not defined.
Abstract	ii
Table of Contents	iii
List of Tables	vii
List of Figures	viii
CHAPTER ONE	1
1.0 INTRODUCTION	1
1.1 Background to the Study	1
1.2 Statement of the Research Problem	3
1.3 Aim and Objectives	4
1.4 Scope of the Study	4
1.5 Significance of the Study	5
CHAPTER TWO	6
2.0 LITERATURE REVIEW	6

2.1	Cryptography	6
2.2	Components of Cryptosystem	7
2.3	Types of cryptosystem	8
2.3.1	Symmetric Key Encryption	8
2.3.2	Asymmetric Key Encryption	8
2.4	Methodologies of cryptography	9
2.4.1	One-Time Pad (OTP)	9
2.4.2	Symmetric Key Cryptography	10
2.4.3	Asymmetric Key Cryptography	12
2.4.4	Steganography	14
2.4.5	Quantum Cryptography	14
2.4.6	DNA Cryptography	14
2.5	Related work	16
CHAPTER THREE		24
3.0	RESEARCH METHODOLOGY	24
3.1	Shift Cipher Encryption Technique	24
3.2	DNA One Time Pad (OTP) Encryption Technique	24
3.3	DNA Digital Coding	25
3.4	Proposed Uber Transport Management System (UTMS) Architecture	26
3.5	Proposed 2-level data security approach	27
3.6	Transport System Security (TSS) Design	28
3.6.1	Use Case Diagram	28

3.7	Performance Evaluation Techniques	29
3.7.1	Computational Complexity of the Algorithm	29
3.7.2	Confidential Level of the Algorithms	29
3.7.2.1	Cryptanalytic Attack	29
3.8	Time Evaluation Functions	29
3.9	Shannon Entropy	30
CHAPTER FOUR		31
4.0	RESULTS AND DISCUSSION	31
4.1	2-Level Data Security Algorithms	31
4.2	Transport System Security (TSS) Module	34
4.2.1	TSS Encryption Result	34
4.2.2	TSS Decryption Result	35
4.3	Performance Evaluation	36
4.3.1	Computational Complexity of the TSS algorithms	36
4.3.2	Security Level	41
4.3.2.1	Cryptanalytic Attacks	41
4.4	Experimental Results	43
4.4.1	Key Generation Result	44
4.4.2	Encryption Result	45
4.4.3	Decryption Experimental Result	46
4.5	Entropy Result	47

CHAPTER FIVE	49
5.0 CONCLUSION AND RECOMMENDATION	49
5.1 Conclusion	49
5.2 Recommendation	49
5.3 Contributions to Knowledge	50
REFERENCES	51
APPENDIX	54

LIST OF TABLES

Table	Title	page
3.1	Transformation of binary code format to a DNA sequence	26
4.1	TSS Encryption Algorithm	31
4.2	TSS Decryption Algorithm	32
4.3	TSS Key Generation Algorithm	33
4.4	Pseudo code of Key Generation, Encryption and Decryption operations	37
4.5	Measurements of the encryption runtime	38
4.6	Measurements of the Decryption Runtime	39
4.7	Key generation experimental result	44
4.8	Encryption experimental result	45
4.9	Decryption result	46
4.10	Shannon Entropy with Frequencies of alphabet symbols	47

LIST OF FIGURES

Figure	Title	Page
2.1	Basic model of cryptosystem (Kumar, 2014)	7
2.2	One-Time Pad (OTP) algorithm (Purusothaman & Saravanan, 2016)	10
3.1	Proposed Uber Transport Management System Architecture	26
3.2	TSS Use Case Diagram	28
4.1	Encryption Experimental Result	34
4.2	Decryption Result Interface	35
4.3	Growing Rate of the Encryption Runtime	39
4.4	Growing Rate of the Encryption Runtime	39
4.5	Growing Rate of the Decryption Runtime A	40
4.6	Growing Rate of the Decryption Runtime	40

CHAPTER ONE

1.0 INTRODUCTION

1.1 Background to the Study

Transportation is a complex system that is a part of any city, state, or nation's critical infrastructure. Any disruption of transport networks from highways to rail lines to rivers and to the sky can have an immediate and significant impact on people and the economy (Mark, 2019). The transport networks have the services needed to meet society's mobility needs. In the end, the transport system's function is to resolve the tension associated with the physical separation between land uses, goods, services and people. Rising travel demand has led to a number of major transportation safety issues. This sector is now a prime target for cybercriminals, hackers and other bad actors (Newton, 2016).

Transport Management Systems (TMS) are application softwares designed to optimize and manage various transport system operations, including inbound and outbound transport operations. TMS have a wide range of features and functions, these include booking rides, GPS vehicle trackers used to track the locations of vehicles and their drivers, calculating transportation costs, optimizing travel routes and vehicle loads, freight audits, creating and optimizing shipping plans and so on (Beecroft, 2019).

Security is important for public transport as it has the ability to influence travel behavior at any stage of a journey from pre-trip preparation to post-trip assessment through undertaking the journey. There are major difficulties in assessing crime and disorder in public transport, making it difficult to assess if there is a difference between the rates of crime perceived and actual. This is a result of the restrictions on the collection of data on real levels of public transport crime (Newton, 2004). It is also unclear to what degree

under-reporting of public transport crime is as an important factor. Hence data reliability and availability in Transport System Applications is considerably unreliable (Findin, 2014).

At present, users of Uber Transport Management System (UTMS) are more concerned with information privacy than convenience. Thanks to the incidence of data breach reports of some TSA companies which includes Taxi, Lyft and Uber and cab riders, among others; although these has limited the patronization of some transport system firms as customers were acquainted of the data breaches in recent years (Stephen, 2019).

Uber is a TMS application that connect drivers with riders. The company had failed to provide fair protections for users' information stored in an Amazon Web Services (AWS) third party cloud storage database called the Amazon Simple Storage System (Amazon S3 Datastore). In May 2014 Uber's Amazon S3 Datastore was attacked using an access key which was posted publicly and given full administrative rights to all data and documents contained in Uber's Amazon S3 Datastore. The hacker accessed one file containing confidential Uber Drivers personal information, including more than 100,000 unencrypted names and driver's license numbers, 215 unencrypted names and their bank details and domestic routing numbers, and 84 unencrypted names and social security numbers (Donald, 2018).

Another Intruder accessed the Uber's Amazon S3 Datastore between October and November 2016 using an AWS access key that was posted to a private GitHub repository. The intruders who perpetrated the intrusion in 2016 claimed they accessed Uber's GitHub website using passwords already revealed in other major data breaches, whereupon they identified the AWS access key they used to access and download files from Uber's Amazon S3 Datastore. The intruders downloaded 16 files containing unencrypted user

personal data relating to the U.S. Riders and drivers which have 25.6 million names and email addresses, 22.1 million names and cell phone numbers, and 607,000 names and license numbers for drivers (Newton, 2016).

Lyft transport organization was previously suspected of both spying on consumers and their own drivers, but now competitor Uber is facing similar accusations. The Uber organization has informed customers that it is investigating an allegation that employees have accessed customer data illegally (Stephen, 2019).

1.2 Statement of the Research Problem

Securing confidential information during communication between two or more clients and data in a database of an Uber Transport Management System (UTMS) is a crucial challenge in recent years. Transport Management Systems are convenient and easy to use for transportation services, but they pose a significant amount of risks and concerns to all active members. The ease of assessing information and services on UTMS has made confidential information prone to cyber-attack.

In recent years, the occurrence of information security breaches in an Uber Transport Management System (UTMS) has become frequent news which has led to financial consequences, as well as identity fraud risk of innocent clients. Therefore, in this research, DNA Cryptography is adopted using Shift Cipher and DNA One Time Pad encryption techniques to secure the confidentiality of information during communication between two or more clients and data in a database of an Uber Transport Management System (UTMS).

1.3 Aim and Objectives

The aim of this research is to improve on a 2-level data security approach for Uber Transport Management System (UTMS).

The research objectives are to:

- Design a 2-level data security algorithm using Shift Cipher and DNA One-Time-Pad (OTP) encryption techniques
- Develop a Transport System Security module for securing the confidentiality of communication between two or more clients
- Evaluate the performance of the developed DNA encryption technique with a Shannon entropy theory

1.4 Scope of the Study

This study focused on developing an improved 2-level data encryption system using DNA One-Time-Pad (OTP) and Shift Cipher encryption method for securing the confidentiality of information during communication between parties (i.e. customers, drivers and administrators) and for the confidentiality of data in archive (i.e. in a database) of an Uber Transport Management System (UTMS). As a proof of concept, a 2-level data encryption system was developed to evaluate the performance of the system. The first security level used shift key value method to shift the plaintext characters' position and the second security level used OTP encryption method to generate the ciphertext of the shifted plaintext obtained from the first security level. The runtime of the developed encryption system was measured using Personal Home Page (PHP) Time Evaluation Function. Shannon entropy theory was used to estimate the average minimum number of bits required to encrypt an information, based on the occurrence of each character in an encrypted information (ciphertext).

1.5 Significance of the Study

The findings of the study would help the stakeholders of Uber Transport Management in providing security designs and algorithms that will improve the security level of the current transportation system issues and minimize the occurrence of private data breach of Uber Transport Management System (UTMS).

The information of customers, drivers and other related business partners who frequently used UTMS will be secured and confidential.

Uber Transport Management System (UTMS) information will be secured and confidential during communication between the parties and when the data is stored. This will increase the reliability of customer's and driver's private information.

CHAPTER TWO

2.0 LITERATURE REVIEW

2.1 Cryptography

Cryptology is the study of cryptosystem and it is the science for information security which converts ordinary plain text into human unreadable codes i.e. cipher text and vice-versa. There are two subfields of cryptology, which are cryptography and cryptanalysis. Cryptography is the science of using mathematics to encrypt and decrypt data, or it is the ability to transmit information in a mangled format between participants that prevents others from reading it. This allows for versatile storage of confidential information or transmits it through vulnerable networks (such as the internet) so that nobody can read it except the intended user (Darbari & Prakash, 2014).

Kumar (2014) describes cryptography as a technique developed by applying mathematics and logic to store and transmit data in coded and protected form so that it can be read and processed only by the intended recipient. The method of data securement is known as encryption by generating cipher text from plain text. Cryptography protects data from third parties i.e. adversaries and is often used for user authentication. The science or technique for decoding the cipher text is cryptanalysis or decryption. The cryptosystem basic model is depicted in Figure 2.1.

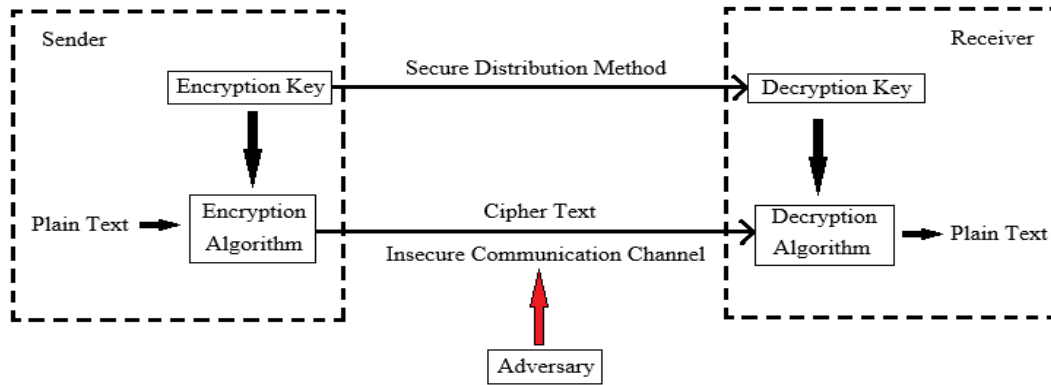


Figure 2.1: Basic model of cryptosystem (Kumar, 2014)

2.2 Components of Cryptosystem

The basic components of cryptosystem are (Gulati & Kalyani, 2016):

- Plain Text: Original data not coded computationally.
- Cipher Text: It is the plain text encoded or encrypted in the form of unreadable human language.
- Encryption Algorithm: This is a mathematical procedure or algorithm which takes plain text as the input and generates cipher text as the output using the encryption key.
- Decryption Algorithm: A mathematical method to transform cipher text to plaintext using decryption key. It is the inverse form of an algorithm for encryption.
- Encryption Key: This is the parameter specifically generated to produce the usable output, i.e. using encryption algorithm to obtain cipher text.
- Decryption Key: This is the parameter needed to convert the encrypted data, i.e. the ciphertext, to its original form, i.e. the plaintext of the intended recipient.

2.3 Types of cryptosystem

Cryptosystem consists mainly of two types, based on encryption and decryption techniques (Babu *et al*, 2016).

- Symmetric key encryption
- Asymmetric key encryption

2.3.1 Symmetric Key Encryption

Symmetric key encryption or hidden key encryption uses the same key for encryption and decryption of information. Encryption key is a mutual secret transmitted between senders and receivers. Due to broad key size, symmetric encryption is difficult to crack and used mainly for bulk encryption. Confidentiality is the only security benefit this methodology makes available. Digital Encryption Standard (DES), Triple-DES (3DES), BLOWFISH and IDEA are just a few examples of common methods for symmetric key encryption (Gambhir & Rakesh, 2019).

2.3.2 Asymmetric Key Encryption

Asymmetric key encryption or public key cryptography is used to encrypt and decrypt information using different keys. Every communication participant has two keys in this methodology; one is public key that is shared with all participants, and the other is private key that is confidential and that is known only to the intended receiver. Though there are apparently different public and private keys, these are mathematically related. Each public key has a private key which corresponds. It will provide honesty, authentication, and non-repudiation. Rivest Shamir Adleman (RSA), Diffie and Elliptic Curve cryptography are just a few common examples of Asymmetric key encryption algorithms (Raj & Panchami, 2015).

2.4 Methodologies of cryptography

Before the modern era, cryptography was only used by the military leaders, spies and diplomats to keep the message secret. Six encryption technologies have been tested over the past years to ensure safe computation, verify validity of documents, authenticate identities of senders and recipients (kazazi & Torkaman, 2015).

Modern cryptography follows solid, scientific methods to design an adversary's potentially unbreakable encryption algorithm. But sometimes, computationally efficient methods are less feasible in practice to do so (Gulati & Kalyani, 2016).

2.4.1 One-Time Pad (OTP)

Towards the end of the 19th century Vernam Cipher developed a one-time pad algorithm. If the key used in OTP is generated randomly and not used more than once, then the algorithm is assumed to be entirely unbreakable. Previously, the randomly generated keys were exchanged as a pad of papers so that after using the key, the sheets could be turn off; thus, the algorithm was called a one-time pad (Gambhir *et al.*, 2019). The algorithm's secret key normally is a string of characters or numbers that is at least as long as the longest message to be encrypted. This algorithm is explained using an example in Figure 2.2.

ENCRYPTION	
Message to be Encoded	DNA
Plain Text (Corresponding Bit String)	01000100 01001110 01000001
Randomly generated Key	11010001 01101000 00101011
Cipher Text (XOR between plain text and key)	10010101 00100110 01101010
DECRYPTION	
Cipher Text	10010101 00100110 01101010
Key	11010001 01101000 00101011
Plain Text (XOR between cipher text and key)	01000100 01001110 01000001
Decoded Message	DNA

Figure 2.2: One-Time Pad (OTP) algorithm (Purusothaman & Saravanan, 2016)

Pseudo-random number generators (PRNGs) are the algorithms that generate sequences of random numbers by using mathematical formulae. Some common PRNGs are; lagged Fibonacci generators, linear congruential generators, linear feedback shift registers etc. Cryptographically Stable Pseudo-Random Number Generators (CSPRNGs) are used for cryptographic applications, and are safer in terms of protection than PRNGs (Purusothaman & Saravanan, 2016).

2.4.2 Symmetric Key Cryptography

Symmetric key cryptography or hidden key cryptography was built mainly for bulk data encryption or data streams. Uses the same key to encrypt and decrypt confidential information. The block cipher is the symmetric encryption algorithm which encrypts a group of bits with a fixed length, i.e. block of plain text at a time. Many types of symmetric encryption algorithms and stream cipher, on the other hand, can encrypt a single bit or byte of data at a time. Block ciphers are typically symmetric encryption algorithms used frequently in recent years. The standard block sizes that can be encoded by the block cipher at a time are 64 bits, 128 bits, and 256 bits. A 128-bit block cipher

usually encrypts 128 bits of plain text at a time, and produces 128-bit cipher text. DES, Triple DES, AES, Blowfish, Twofish etc. are the most common block ciphers having a wide range of applications. Other examples as listed (Sohal, 2018) are:

- DES: Data Encryption Standard (DES) is a common 64-bit block cipher developed in 1975 and standardized by ANSI (American National Standards Institute) in 1981 as ANSI X.3.92. The key used in DES consists of 64 bits, the effective key length is 56 bits. The actual key size 8 bits are used to test parity, and then discarded.
- Triple DES: Triple DES focuses its basic approach on DES. It encrypts the plain text three times. The total key length of Triple DES is 192 bits i.e. this block cipher uses three 64-bit keys. Like DES each key has an effective length of 56 bits. The first encrypted cipher text in Triple DES is again encrypted by the second key, and the third key encrypts the resulting encrypted cipher text. Although this algorithm is much safer in terms of surety than DES, for real-life applications the execution time is too slow.
- Advanced Encryption Standard (AES): AES is the most widely common symmetric block cipher with a wide range of applications in modern security systems, such as financial transactions, e-business, wireless communication, encrypted data storage etc.

In both hardware and software, it's more reliable and faster than triple DES. The number of rounds is variable, as is the duration of the AES keys. 10 rounds are valid for 128-bit key, 12 rounds for 192-bit key and 14 rounds for 256-bit key. For each round different 128-bit round keys are used, determined from the original AES key. AES algorithm handles data block in bytes; i.e., it handles 128 bits of plain text as 16 bytes. The production of AES, a clear successor to DES, started in 1997 by the

National Institute of Standards and Technology (NIST) and was more protected against the attacks by brute force.

- Blowfish: Blowfish is another 64-bit block cipher that supports different key lengths ranging from 32 to 448 bits. This function makes Blowfish algorithm suitable for both domestic and exportable use. This algorithm can be used in software easily and is free for all users, since Blowfish is unpatented and license free.
- Twofish: The 128-bit symmetric key block cipher supports up to 256 bits' key length. Twofish is similar to the block cipher Blowfish mentioned above, but is not as common as Blowfish.

2.4.3 Asymmetric Key Cryptography

Asymmetric key cryptography or public key cryptography uses varying but mathematically similar keys; widely distributed public encryption keys; and decryption private keys known only to intended recipients. According to Jain *et al.* (2014), examples of asymmetric key cryptography are:

- Diffie-Hellman Key Exchange Protocol: is a scheme for information exchange through a vulnerable public channel. Let, two people Alice and Bob want to exchange data securely via public web, as referred to in cryptographic literature. The main concept for this protocol to create a safe communication network is that there are some hidden details that Alice and Bob know only about. This information is used to deduce an appropriate key for encoding or decoding the data (Taha *et al.*, 2019).
- RSA Algorithm: The reliability of the RSA algorithm relies on the fact that it is too difficult to solve the factorization of a large number, which is a combination of two large prime numbers. This is a trapdoor function that is easy to calculate in one direction but hard to calculate the reverse without any precise details. Let for instance,

$P \times Q = N$, where P and Q are very large prime numbers. If P and Q are given, then N can be easily determined. But when N is given, in polynomial time, the factorization of N can't be determined. But if Q is identified alongside N , then P can be easily determined i.e. $P = N / Q$ (Kumar, 2014).

- **El-Gamal Cryptosystem:** El-Gamal Cryptosystem is an asymmetric key encryption, based on the Discrete Logarithm Problem for this algorithm. There is no current algorithm for a given number that can find its discrete logarithm in polynomial time, but the inverse operation of the power can be efficiently obtained. Another main feature of the El-Gamal cryptosystem is encryption by randomization. This algorithm can create a safe channel for key sharing and generally used as key protocol for authentication. The key size of this algorithm should be in excess of 1024 bits for stability. El-Gamal algorithm's big disadvantage is that it is fairly time-consuming (Jain *et al.*, 2014).
- **Elliptic Curve Cryptography (ECC):** it is the promising future of asymmetric key encryption based on algebraic elliptic curve structure over finite fields. Similar to the ElGamal cryptosystem, ECC protection also depends on the algorithmically difficult, discrete logarithm problem. Although this algorithm follows the same technique as the Diffie-Hellman key exchange protocol and the RSA algorithm, the unique feature of Elliptic-curve cryptography is that the numbers are chosen to form a finite field specified within an elliptic curve. This technique is much smaller in key size; For example, 160-bit key-length ECC ensures the same level of safety as 1024-bit key-length RSA algorithms. Since the processing power consumption and the necessary memory sizes are significantly small, this relatively new algorithm has a huge potential for applications on restricted devices. In real life, ECC has many

applications; for example, safe data transfer, digital signatures, shared authentication (Harsh & Jae, 2018).

2.4.4 Steganography

The term "steganography" comes from the Greek words "stegano," meaning "covered" and "graphie" means "writing." This approach guarantees data protection by ambiguity rather than actual encryption. It transmits data via video, audio, document or image files, by embedding it in an unnoticeable way (Taha *et al.*, 2019).

2.4.5 Quantum Cryptography

Quantum cryptography is a thriving technique of encryption based on the principles of quantum mechanics, literally the principle of Heisenberg Uncertainty, and the theory of photon polarization. This approach takes advantage of the counterintuitive behavior of atomic scale elementary particles, typically the mass of photons. Quantum bit, also called qubit, transmits information in quantum cryptography and is just a single photon particle (Gambhir & Rakesh, 2019).

2.4.6 DNA Cryptography

Another rapidly evolving approach within the cryptographic domain is focused on sequences of DNA. The concept of DNA cryptography is inspired by the DNA molecule which has the capacity to store, process and transmit information. It operates on the DNA computing principle which uses 4 bases to conduct computation, i.e. Adenine (A), Guanine (G), Cytosine (C), and Thymine (T), (Raj & Panchami, 2015).

DNA-based cryptography is the technique employing biological structure to hide data and information. Scientists nowadays work in the area of DNA cryptography; are based on using DNA code to represent binary information in one form or another. DNA encryption

is a method that involve the use of DNA sequence to transform plain text into cipher text.

There are four methods of DNA encryption (Pramanik & Kumar, 2012), these are:

I. DNA random One Time Pad Based

In this technique, a series of randomly ordered non-repeating characters are used to enforce a one-time pad, if an input ciphertext is used once, it is not used again to increase protection. The size of the plain text in this scheme maybe be equivalent to the size of a one-time pad. DNA One-Time Pad method is used to transform the short parts of plain text messages to ciphertext. To substitute the plain text, a random and special codebook is taken into account. Due to hardware limitations this strategy applies only to short messages. The large size of the message makes DNA mapping more complicated (Singh & Kamaljit, 2015).

II. DNA chip-based cryptography

The Microarray is also called the DNA chip. This DNA chip, made of semiconductor-designed nucleic acid and electronic circuit. This technology represents excellent development in the field of cryptography based on DNA. A DNA-chip is used to store, handle and maintain a significant volume of genome and biological information. The text and photographs are encrypted using biochemical processes. The drawback of this strategy is the sudden physical factor shift which use to produces negative outcomes (Kumar, 2014).

III. DNA Fragmentation

This approach is used in the DNA sequence for library building. This splits the DNA sequence into tiny bits. Most encryption algorithms use it as a second protection layer. It's also being applied in key encryption (Darbari & Prakash, 2014).

IV. DNA Steganography

Steganography of the DNA is used to conceal one message inside another. Image, audio, and video are reused to preserve vast volumes of data, but data can be lost due to sudden environmental changes (Anwar *et al.*, 2015).

2.5 Related work

Olga and Borda (2013) have suggested a cryptographic DNA strategy that uses indexing of DNA. The researchers got accessed to the DNA code from the database random, the database comprises information that are genetic in nature as the OTP key and this sequence is transmitted via a protected information transmission medium to a recipient. The plain text was encrypted by converting each character of the plain text to its equivalence ASCII code and then transformed into the binary digits (code), then finally converted to into A (Adenine), T (Thymine), C (Cytosine) and G (Guanine). This DNA sequenced are checked in the main sequence and also noted with the index numbers. The sequences of the obtained decimal values are the required cipher text. The receiver decodes this cipher text via key and index pointer.

Kazazi and Torkaman (2015) designed a five-phase algorithm that is cryptography based on DNA which is used for the encryption of an information. The five stages of this technique includes: preprocessing of data, key generation, and encryption process which is carried out at the source end and encoding or post processing of data at the receiver end. This technique employed the principle of vigenere cipher which offers a double safety layer. This is a hidden cryptographic key technique which takes a huge amount of execution time.

Priyadarshani *et al.* (2014) suggested a system that uses DNA sequence and substitution techniques. Substitution and DNA sequence techniques are used to encode simple text

which was chosen from 55 million DNA sequences. The DNA sequence selection and substitution make the algorithm simple, efficient, secured and unbreakable. The algorithm was developed and applied on an Electronic Medical Record System.

Raj and Panchami (2015) proposed a cryptography technique based on DNA that employed permutation and random key generation algorithm. This approach makes random use of the concept of creating DNA patterns. The input is converted into 7 bits ASCII code then later transformed to its corresponding binary values. Binary Code where represented with A, C, T, and G as the DNA Sequence of 00, 01, 10, 11 respectively. A sequence of DNA is chosen as a key and stored in segments, each segment consists of four characters which is based on the character's positions in the segment table. Finally, the random sequence of DNA picked from this table is translated into an encrypted sequence. The encrypted sequence and the cipher key is send to the receiver via the media communication channel. The sequences of the cipher are decrypted to get back the same text by applying the reverse steps. This algorithm is different from others as it does not use conventional mathematical operations or techniques for the manipulation of the data. This approach cannot therefore be extended for protection at multilevel.

Mahalaxmi and Raj (2016) proposed a symmetric key cryptography based on DNA for safe transmission of data via the channel of communication. Firstly, the input (data, image or text) is converted to ASCII value and the ASCII values were transformed into their corresponding binary digits. The binary digits are converted into DNA codes. The DNA code are randomly allocated with a corresponding private key which further converted to an advance ASCII codes. Finally, the ASCII codes are replaced using DNA sequence table and clinical permutation is tested with private key. This symmetric key algorithm is designed using Java for the transmission of data via a communication channel. The limitation of this algorithm is that chromosome DNA is required.

A technique based on DNA cryptography was proposed by Gulati and Kalyani (2016), using XOR operation and One Time Pad (OTP) for the safe transmission of information. This algorithm is complementary to arithmetic operations, XOR operations, One-Time Pad and DNA principle, it provides three levels of protection. The hacker does not imagine this procedure to be very easy and safe to use because of certain preconditions applied, this technique is not user-friendly, so you need to take the precautionary preconditions when choosing the OTP.

Purusothaman and Saravanan (2016) proposed a technique based on the hidden algorithm of the modified Shamir, and the technique of encryption and decryption based on DNA. A big no. of consumer sits on the end of the receiver. In the algorithm some additional protection is fused. Decryption is only possible when the entire client is active in the decryption process, only then will it decipher the hidden message. Using Mathematical Equations, the message is translated to ASCII values. Converted to DNA bases ASCII value. The transmission of the message is done through the client party, and the information is decrypted DNA encryption principle to improve the confidentiality of information for multicast applications. The proposed technique was developed using Java and Python technologies. The algorithm is very strong, because of the mutual principle, if anyone is not interested in the decryption process, then decryption of the message is impossible. This technique has high execution.

Bhavithara *et al.* (2016) suggested a double-layered, symmetric key algorithm. In this method, plain text at the sender end is encoded twice with a key length 100, and the procedure is reverse to obtain the result of the recipient. The algorithm first stage is encryption where plain text is transformed to a ciphertext and decryption is the algorithm second stage, used to obtain the original plain text. Here plain text is provided to the Field-Programmable Gate Array (FPGA) via the PS2 keyboard that reads ASCII value using

the FPGA. Then, a codon table converts it to the codon, and Vigenere cipher is used for codon encryption. Key distribution is not discussed here; therefore, the algorithm can have a hectic problem.

Darbari and Prakash (2014) proposed a method that involves three stages, i.e. set of proofs, estimate of the reputation factor, and faith in reputation. The trust-based distributed systems are extremely reliable when it comes to managing security features. DNA based cryptography improves distributed system based on confidence in security. This strategy incorporates data post-processing approach to deal with a number of cyber-attacks. This technique is suitable for distributed systems focused solely on trust-based distributed systems but not acceptable for all.

Jain *et al.* (2014) suggested a technique for cryptographic DNA that uses XOR operations. Simple text is selected and encrypted using randomly generated key in this technique. Also a randomized codon list is generated, and the random key is XOR-ed. To encrypt an information in this technique, the DNA mapping operation is applied. The key is made from the properties of DNA and the biotechnology. This technique is high execution time because of the XOR operation, and complexity of the algorithm is $O(\log N)$. But the complexity of space is a big problem to the research, although this can be minimized in the future research.

Babu *et al.* (2016) have suggested an inspired cryptography of the pseudo biotic DNA that uses central biological dogmas. This is unlike DNA cryptography but uses only the terminology and DNA process. In this process, the transcription, splicing, and translation are used for encryption and decryption. These encryption measures are used to make protection easier. In this method the key is obtained randomly to increase the confusion and diffusion degree of the algorithm. This makes the algorithm strong and complex to

break the cipher text. Analysis of robustness was performed to demonstrate the security level of the algorithm which is very secure against attacks. The algorithm implementation requires advanced tech bio-computational laboratories.

Basha *et al.* (2015) proposed a secured advanced encryption algorithm which is based on DNA for large data. Unauthorized person can only access the information in its cipher form but cannot understand the meaning of the information. Big data was used to test the performance of the algorithm by encrypting a numerous transaction data. The process of the encryption involves using DNA sequence table and PHP language is used for the implementation of the system. The proposed algorithm is limited to solving of big data confidential problems.

Vidhya and Rathipriya (2018) developed a 2-level text data encryption using DNA encryption cryptography to increase the confidentiality of an information by increasing the security level of the algorithm which leads to increase in complexity of the sequence DNA. The main objective of the research was to provide data with high level of security. The research work was implemented in two security phases. The first converts the plain text into an ASCII code using a shift key and then further converted to binary number. The second phase employs insertion method, which fix some binary digits after segmenting the resulting ASCII code. The new generated ASCII codes are converted to their corresponding binary numbers and then replaced with the DNA sequence representing the cipher text. The recipient has to apply insertion technique to decrypt the information into plaintext. Shannon entropy was used to evaluate the degree of the randomness of the algorithm, the data compression ratio and the algorithm complexity was used to ascertain the average execution time of the two phase algorithm.

Aggarwal and Kanth (2014) suggested a complementary pairing technique that differs from the conventional approach of encoding DNA. The four different DNA sequences A, T, C and G are supplemented A with T, G with C, C with A and T with G in the first step. After the reference sequence of DNA is selected in the second stage and called as sender S and receiver is aware with S. S is then supplemented in the third stage and called as S'. This S' will then be forwarded to the recipient using a stenographic communication medium. The recipient, using S, must decode ciphertext S'. Steganography implementation improves the confidentiality of the proposed algorithm. decrypting the information required a proper guess of the sequence S that is randomly generated. There are about 55 million DNA sequences available in public domain. Hence cracking S will be major. This makes the algorithm sturdy and reliable.

Sohal (2018) introduced a new algorithm approach where data are encrypted from the client-side until they are saved in a cloud database. This algorithm is a symmetric key cryptographic approach that employed DNA sequence. Furthermore, the detailed findings of this algorithm as contrasted with the existing symmetric key algorithms (DNA, AES, DES, and Blowfish) show that this method performs better than the traditional algorithms based on cipher text size, encryption runtime, and confidentiality of information transmitted. This method is effective and executes better but yet has limitation in times of space management.

Yunpeng *et al.* (2017) suggested cryptographic technique which is based on DNA sequence, hamming and a block cipher coding patterns are used to protect the algorithm key. This technique is strong symmetric cryptography used to improve an existing method. Length matching is deployed in the development process to protect data against all kinds of attacks.

Tiwari and Hyung (2018) proposed an ECC algorithm for DNA mapping. This system used DNA code randomly and alphabets of plaintext are dispersed without repetitive feature. The encoding and decoding used the alphabets at both ends. The system was implemented successfully and tested on internet of things applications in real-time.

El-Latif and Moussa (2019) proposed an encryption that execute in two rounds. This technique is similar to the trending technique known as Data Encryption Standard (DES) algorithm. This algorithm converts the plaintext using two different keys. These two keys are constituted on the Gaussian kernel function (GKF) and elliptic curve cryptography (ECC). The second key is generated based on the second characters that are replicated from the first key using a random injection mapping. In the final stage, the DNA sequence obtained is used to arbitrarily hides the message with the use of GKF numbers.

Tanaka *et al.* (2015) presented a DNA cryptography which used public Key. The researchers discussed and depict an algorithm on how to produce two public keys via a solid mixture which was used for key generation, the conversion of the data was based on DNA code and the first key was synthesized public using a synthesizer of DNA, and the encrypted information sequence was generated with second public key. The immobilization procedure obtained the result and while the PCR amplification is completed acquired using secret DNA sequence to decipher the converted DNA sequence.

Anwar *et al.* (2015) proposed a method based on an XOR operation that used a symmetric key approach to encode information. The algorithm is an extension of DNA hybridization which was proposed by the researchers in the previous research domain. It was used to secure transmission medium such as the internet of things applications. This technique is simple and strong for DNA hybridization. The measurement of the matrix is used to

reduce running time of the technique. This algorithm is cost-effective in times of implementation.

CHAPTER THREE

3.0 RESEARCH METHODOLOGY

3.1 Shift Cipher Encryption Technique

The general idea of shifting plaintext by k letters (or adding k modulo 26) is called a shift cipher, with a key of k . Shift Cipher is a Symmetric key cipher where the plain text and the pseudo-random key (key stream) are combined by XOR operation. In the encryption, each byte of plain text is encrypted with the corresponding character of the key stream. An alternate name is the cipher state, where the encryption of each character depends on the current state (Sudhakara *et al.*, 2016). The mathematical notation of Shift Cipher is

$$C \equiv P + k \text{ modulo } 26$$

Where P stands for any number between 1 and 26 that represents a plaintext letter and C stands for a number that represents a ciphertext letter and k is the key encryption.

3.2 DNA One Time Pad (OTP) Encryption Technique

DNA One-Time pad encryption technique is implemented using a set of randomly arranged non repeating characters as the input. This set of random characters works as a pad. It is considered highly secure because if an input cipher text used once, it is never used for any other message, due to which it is named as One-time-pad. The length of the pad should be equal to the length of the plain text in One-time-pad (Hazra *et al.*, 2018). In DNA Cryptography there is a method of DNA one-time pad substitution where One-time-pad encryption process uses a random codebook to convert short segments of original plaintext messages to cipher text, which provide a random mapping. Two important points should be noted about codebooks, first a codebook must be truly random, and second thing it must be used only once for any message.

One-time-pad encryption equation is generalized as

$$C_i = E(P_i, K_i) \text{ for } i=1, 2, 3, \dots, n$$

where, E is the encryption operation, P_i is the i -th character of the plaintext, K_i is the i -th byte of the key used for this particular message, C_i is the i -th character of the resulting ciphertext and n is the length of the key stream. Both the key stream K and the encryption operation must be kept secret. For practical purposes, the key for a One-Tim-Pad cipher is a string of random bits, usually generated by a Cryptographically Strong Pseudo Random Number Generator (CSPRNG). The security of the one-time pad relies on keeping the key 100% secret. It is implemented by using a modular addition (XOR) to combine plaintext elements with key elements. The key used for encryption is also used for decryption (Singh & Kamaljit, 2015).

3.3 DNA Digital Coding

DNA Digital Coding is a mapping technique for the 4 main DNA bases: Adenine (A), Cytosine (C), Guanine (G) and Thymine (T) which are substituted with the combination of 0s and 1s (Hazra *et al.*, 2018).

Using this scheme, the plaintext message or information submitted (entered) via UTMS form interface are encrypted and decrypted in TSS module. Table 3.1 shows the Binary Form to DNA sequence conversion process, and vice versa.

Table 3.1: Transformation of binary code format to a DNA sequence

Binary Form	DNA Sequence
00	A
01	C
10	G
11	T

3.4 Proposed Uber Transport Management System (UTMS) Architecture

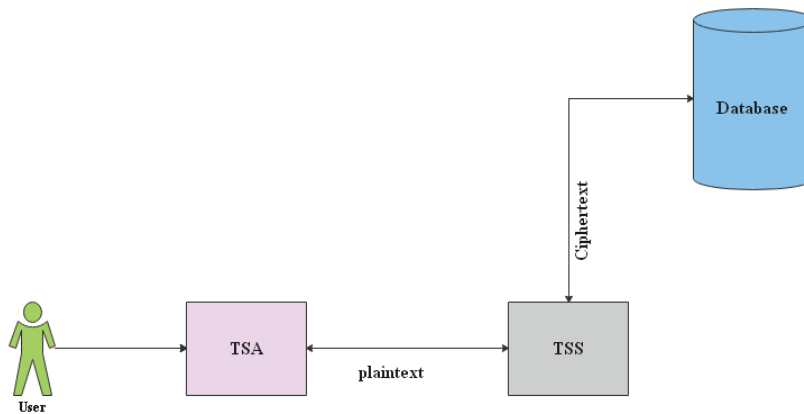


Figure 3.1: Proposed Uber Transport Management System Architecture

The propose UTMS architecture consists of three components. These include: Transport System Application (TSA), Transport System Security (TSS) module and a Database as described in figure 3.1.

Transport System Application (TSA) is a software designed to perform transportation services which include booking, tracking vehicle, navigating routes among other services. It is used in accepting and processing of information submitted by client (user of the software known as customer).

Transport System Security (TSS) module consist of functions and methods of the proposed DNA cryptographic technique used to encrypt and decrypt plaintext and ciphertext respectively. It is used as a middle agent for transmitting information between the TSA and the database. Information retrieved from TSA interface must be process via TSS module before reaching the database and vice versa.

Database is a TSA storage medium used to insert, update, delete and retrieve encrypted information processed and transmitted via TSS.

3.5 Proposed 2-level data security approach

The proposed 2-level data security approach is designed to eliminate the limitations of the existing information security methods of low confidentiality and high execution time.

The 2-level data security design are:

Level 1: Shift Cipher

- Plaintext to shift text with shift key
- Shift text to ASCII number
- ASCII Number to Binary number
- Binary Number to DNA Sequence

Level 2: DNA One Time Pad (OTP) Method

- Plaintext to ASCII Number
- ASCII Number to Binary number
- Binary Number to DNA Sequence

3.6 Transport System Security (TSS) Design

Transport System Security (TSS) design captures two kinds of views: the user point of view and the TSS implementation point of view. It was accomplished through diagrams in violet Unified Modeling Language (UML). UML provides graphical notations for developing software application visual models. It provides the TSS development with architectural structure and description of behaviors. Diagrams generated with this method aid in the interpretation of system to avoid ambiguities.

3.6.1 Use Case Diagram

The capabilities expected from the system are described in Figure 3.2. Use case was used to display typical user-to-system interactions under construction. The purpose was to capture every possible task that a user may perform in a system with the use case. The full functionality of the system is defined in the use case. Figures 3.2 presents TSS module diagram.

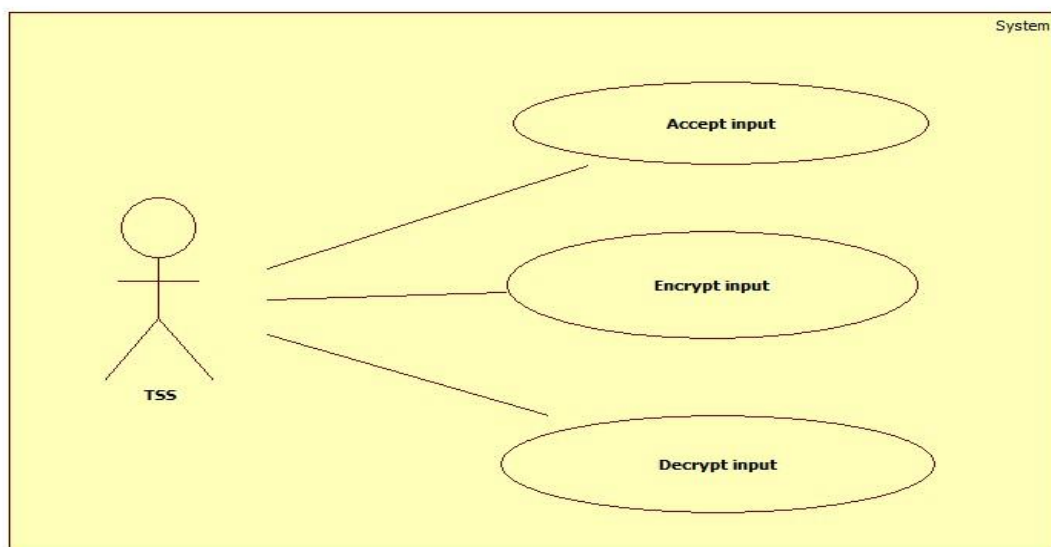


Figure 3.2: TSS Use Case Diagram

3.7 Performance Evaluation Techniques

3.7.1 Computational Complexity of the Algorithm

The analysis of the TSS Encryption, Key Generation and Decryption Algorithms Complexity is important as it revealed the efficiency of the algorithms for real time applications. In this work the computational time was evaluated for the algorithms using methods of complexity theory. The conclusions obtained were tested through the results of the implementation. The complexity of 3 important TSS model algorithm operations was analyzed: key generation computation, encryption, and decryption.

3.7.2 Confidential Level of the Algorithms

Security of the algorithms was analyzed using cryptanalytic attacks.

3.7.2.1 Cryptanalytic Attack

The TSS key generation algorithm was based on the OTP principle to protect it against vulnerability to most conventional attacks.

The resistance of the algorithms to Mathematical Analysis Attack, Only Ciphertext Attack, Known Plaintext Attack, Chosen Plaintext Attack and Chosen Ciphertext Attack are evaluated to ascertain the confidential level of the algorithms.

3.8 Time Evaluation Functions

Implementations were achieved in Sublime Text 2018 IDE using PHP language on the computer with the following properties: Genuine Intel(R) CPU Dual Core 1.79 GHz, 1 GB RAM. In PHP there are different functions available to measure execution time. Below is the chosen function for time measurements on this research.

```
$time_start = microtime_float();
```

```
//sleep for a while
```

```
usleep(100);
```

```
//Encryption or Decryption code
```

```
$time_end = microtime_float();
```

```
$time = $time_end - $time_start;
```

This procedure measures the length of time the process between the "start and end time" has kept the CPU busy. It gives more consistent results as it does not calculate how much time has passed, the time while the code is waiting (sleeping) is not counted, and other processes that consume CPU do not skew the timings.

3.9 Shannon Entropy

Shannon entropy is the volume of information within a variable, which is the main concept for the notation of information. In terms of probabilistic model, entropy is an information theory which states that the modules with much reshuffle has high entropy while systems with few reshuffles has low entropy. The Shannon entropy equation is used to estimate the average minimum number of bits required to encrypt an information, based on the occurrence of each character in an information (Vidhya & Rathipriya, 2018). The Shannon entropy equation is

$$H(X) = - \sum_{i=1}^n P(x_i) \log_b P(x_i)$$

P denotes the Probability of a given symbol while b is the algorithm base.

CHAPTER FOUR

4.0 RESULTS AND DISCUSSION

4.1 2-Level Data Security Algorithms

The 2-level data security algorithms were designed to eliminate the limitations of the existing information security methods of low confidentiality and high execution time. Table 4.1 presents the encryption algorithm using Shift Cipher technique at the first security level and OTP technique for the second security level.

Table 4.1: TSS Encryption Algorithm

Step	Algorithm
	// First security level
1	Enter input P_i (where $i=1, 2, 3, \dots, N$ characters) and input could be (alphabet, numbers and special symbols)
2	Compute the length N of P_i
3	For each P_i , shift P_i using the shift key value and convert to its corresponding decimal value P_{ASCII}
4	Convert P_{ASCII} to its equivalent binary string P_B
5	Compute P_B length L obtained from step 4
6	If L is less than 8-bits, convert P_B to 8-bits string by inserting 0s at the prefix of P_B to generate 8-bit string P_{Bytes} , else goto step 7
	$P_{Byte} = [L / 8] \quad \text{if} \quad L \bmod 8 = 0$ $P_{Byte} = [L / 8] + 0 \quad \text{if} \quad L \bmod 8 \neq 0$
	//Second security level
7	Generate key K_{OTP} using P_{Byte} and a pseudo random number R
8	Apply XOR operation on P_{Bytes} and its respective K_{OTP} obtained from step 7 to get the DNA binary form
9	Convert the result of the DNA binary form to DNA sequence P_{DNA}
	Generate ciphertext C_{DNA}

In Table 4.1, the input P represent the plaintext while I is the I^{th} position of a character (symbol, number, alphabet) in P , each character is converted to its corresponding decimal value, denoted as P_{ASCII} and further converted to binary numbers denoted as P_B respectively. The binary number for each character P_B is transformed to 8 bits, denoted as P_{Byte} . K_{OTP} represent the key variable, it is generated using the key generation function that accepts two parameters, these are: P_{Byte} of each character and a generated pseudo random number. The encryption algorithm applied an XOR operation on the P_{Byte} and K_{OTP} . The binary result obtained is Converted to DNA sequence and the cipher text is generated. Table 4.2 presents the decryption algorithm of the 2-level data security approach.

Table 4.2: TSS Decryption Algorithm

Step	Algorithm
	// Security level I
1	Enter input (DNA sequence C_i , where $i=1, 2, 3, \dots, n$ characters)
2	Calculate the length of the DNA sequence N
3	For $i=1$ to N
4	Shift C_i using shift key value and Convert C_i to its corresponding binary string C_B
5	$C_{\text{Bits}} = \text{concat}(C_{\text{Bits}}, C_B)$
6	Increment i
7	if i less than or equal N , repeat steps 4 to 6, else goto step 8
8	Compute C_{Bits} length L
9	Divide L by 8 to get new length m $m = L / 8$
10	For $j=1$ to m , divide C_B into m groups of 8-bit binary string C_{Bytes} $C_{\text{Bytes}} = \text{substr}(C_B, j, 8)$
	//Security level II
11	Generate key_{OTP} using m and the input corresponding pseudo random number (i.e. obtained from the database)
12	Perform XOR on C_{Bytes} and key_{OTP}
13	Convert XOR 8-bit binary string result to its equivalent decimal value C_{ASCII}
14	Substitute C_{ASCII} to its corresponding plaintext character
15	Increment j
16	if j less than equal m , repeat steps 11 to 15 , else concatenate the plaintext characters (i.e. obtained from step 10) and generate the plaintext P

In Table 4.2 the DNA sequence C is received from the sender and each character of the DNA sequence C_i is transformed using Shift key value then converted into their corresponding binary numbers C_B . K_{OTP} is generated using the length of the resulting binary strings m and the input corresponding pseudo random number (i.e. obtained from the database). The decryption function performs XOR operation on the C_{Bytes} with the K_{OTP} . The resulting binary strings are converted to an ASCII numbers. The numbers are transformed to their corresponding characters as plaintext P . Table 4.3 depicts the algorithm for generation of key.

Table 4.3: TSS Key Generation Algorithm

Step	Algorithm
1	Enter input N and R (where N is the input length and R is a pseudo random number)
2	Convert N and R to their corresponding binary sequence (i.e. N_B and R_B respectively)
3	Compute the length of N_B and R_B
4	Find the difference D of N_B and R_B
5	If D is not equal zero and N_B length is greater than R_B length then add zero(s) of size D to the prefix of R_B else, add zeros of size D to the prefix of N_B
6	Apply XNOR operation for the result of N_B and R_B (i.e. the result obtained in step 5)
7	Generate key K_{OTP}


In Table 4.3, the key generation function requires two parameters to process the key, the first parameter is the length of the input to be encrypted and the second parameter is a generated a pseudo random number. Both inputs/parameters are converted to their corresponding binary strings. The length of both strings are compared by finding the difference and then converted to 8-bit strings. After converting the two strings to 8-bit string, XNOR operation is applied to generate the key denoted as K_{OTP} .

4.2 Transport System Security (TSS) Module

Transport System Security (TSS) Module contains two function for the processing of an information. These functions are encryption and decryption function, each of the function requires an input before the information can be process and the inputs are plaintext and DNA ciphertext respectively. The TSS module was developed using Personal Home Page (PHP).

4.2.1 TSS Encryption Result

The interface presented in figure 4.1 displayed the experimental result for the encryption of “DNA ENCRYPTION FOR TMS” as input.



The interface is displayed on a light blue background. It features an **INPUT** section with a text box containing "DNA ENCRYPTION FOR TMS". Below this is an **OUTPUT** section with a text box displaying the encrypted result: "ATAAATGGATCCCCAATACATGGATCTAGCGAGTCAGCAAGAAATTCATGTATGGCCCA ATAGATGTAGCGCCCAAGAAATGCAGCT". Under the output, the **Execution Time: 0.0019290447235107** is shown in red text. At the bottom, the **DNA Encryption Operations** section contains three buttons: "Encrypt" (orange), "Decrypt" (green), and "Clear" (blue).

Figure 4.1: Encryption Result Interface

The encryption result of the input “DNA ENCRYPTION FOR TMS” was executed in 0.00193 and the output of the encryption process was displayed at the output field as

“ATAAATGGATCCCCCAATACATGGATCTAGCGAGTCAGCAAGAAATTCAT
GTATGGCCCAATAGATGTAGCGCCCAAGAAATGCAGCT”.

4.2.2 TSS Decryption Result

Figure 4.2 depicts the decryption result of an input (DNA sequence) which was processed using the decrypt button and the result was displayed in the output filed as plaintext information. The decryption details as seen in Figure 4.2 are:

Decryption Input:

ATAAATGGATCCCCCAATACATGGATCTAGCGAGTCAGCAAGAAATTCATG
TATGGCCCAATAGATGTAGCGCCCAAGAAATGCAGCT

Decryption Output: DNA ENCRYPTION FOR TMS

The screenshot displays a web interface for DNA encryption and decryption. It features a light blue background. At the top, under the heading "INPUT", there is a text box containing the DNA sequence: "ATAAATGGATCCCCCAATACATGGATCTAGCGAGTCAGCAAGAAATTCATGTATGGCCCAATAGATGTAGCGCCCAAGAAATGCAGCT". Below this, under the heading "OUTPUT", is a text box displaying the result: "DNA ENCRYPTION FOR TMS". Underneath the output box, the "Execution Time: 0.0012228488922119" is shown in red text. At the bottom, under the heading "DNA Encryption Operations", there are three horizontal buttons: an orange "Encrypt" button, a green "Decrypt" button, and a blue "Clear" button.

Figure 4.2: Decryption Result Interface

4.3 Performance Evaluation

4.3.1 Computational Complexity of the TSS algorithms

An algorithm's execution time is deemed to be the sum of all the operations. The number of operations can be either constant or variable and depend on input parameters. According to the approximations from complexity theory, the smallest possible class of functions is used to describe the increasing rate of the algorithm's runtime.

The researcher evaluated the complexity of the TSS algorithm in 3 essential operations: key generation computation, encryption, and decryption. The key generation is calculated in $2 \times 256 \times n$ operations, where 256 is the number of possible values for a byte, and n is the length of the key sequence. Encryption and decryption are carried out in m operations, where m is the number of plaintext and ciphertext characters respectively. Taking the smallest class of functions, the key generation computation complexity is $O(n)$, and $O(m)$ for the encryption and decryption methods respectively. This means that the computational time increases linearly, depending on the size of the input. The pseudo code for those operations is provided in Table 4.4.

Table 4.4: Pseudo code of Key Generation, Encryption and Decryption operations

Key Generation	Encryption	Decryption
for (\$i=0; \$i < \$str_length; \$i++) {	for (\$count=0; \$count < \$string_length; \$count++) {	for (\$i=0; \$i < \$Cbin_string_length; \$i+=8) {
\$p_var = substr(\$a, \$i, 1);	\$DNA_Nucleotide = DNA_encryption(\$xor_bin_res);	\$character_bin = substr(\$DNA_bin, \$i, 8);
\$q_var = substr(\$b, \$i, 1);		\$c_XOR_bin8 = xor_binary(\$c_key_bin8, \$character_bin);
if (\$p_var <> \$q_var) {		
\$res_val = '0';		
\$XNOR_result = \$XNOR_result.\$res_val;	\$DNA_result = \$DNA_result.\$DNA_Nucleotide;	\$Decryption_res = \$Decryption_res.\$decrypt_char;
}	}	}
return \$XNOR_result_8bit;	return \$DNA_result;	return \$Decryption_res;

The experimental findings have proved the estimated complexity to be accurate. The software was experimented at different, progressively increasing values of n and m in order to see the progression of the runtime. Figure 4.3 to 4.6 present graphics of the runtime growing rate for key generation, encryption and decryption processes while the measurements of the execution time are described in Tables 4.5 and Table 4.6.

Table 4.5: Measurements of the encryption runtime

Plaintext size (chars)	Runtime (ms)
3000	0.42
2500	0.37
2000	0.27
1500	0.19
1000	0.14
500	0.08
400	0.06
300	0.05
200	0.03
100	0.02

Table 4.5 present the measurements of the encryption runtime for plaintext characters of size 100 to 3000 characters. Plaintext with 100 characters are executed within 0.02 milliseconds and plaintext with 3000 characters are executed within 0.42 milliseconds. The result shows that the higher the number of characters of a plaintext the higher the execution. The growing rate of the encryption runtime is presented in Figure 4.3 and Figure 4.4 respectively

.

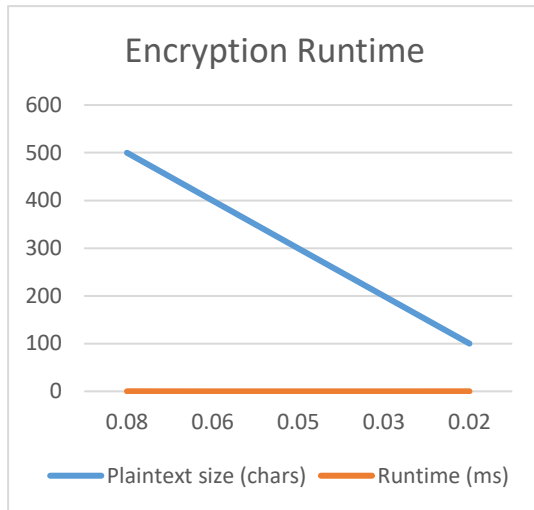


Figure 4.4: Growing Rate of the Encryption Runtime

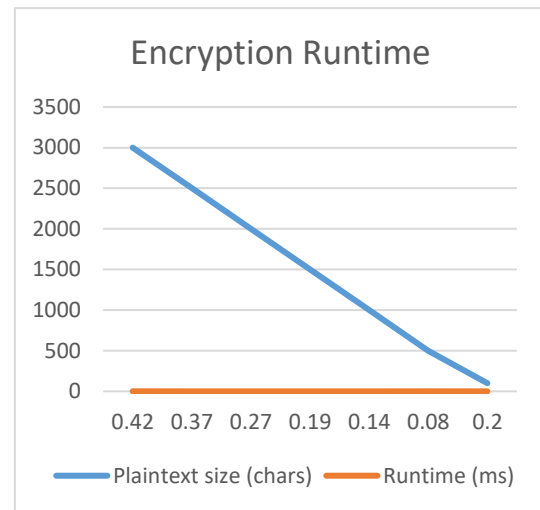


Figure 4.3: Growing Rate of the Encryption Runtime

Figure 4.3 presents an encryption runtime of the plaintext characters between 0 to 500 characters. While Figure 4.4 presents an encryption runtime of the plaintext characters between 500 to 3000 characters. Both Figure 4.3 and Figure 4.4 shows that the encryption runtime against the plaintext character size are linear. Table 4.6 depicts the measurement of the decryption runtime of the experimental result

Table 4.6: Measurements of the Decryption Runtime

Ciphertext size (chars)	Runtime (ms)
3000	0.19
2500	0.14
2000	0.12
1500	0.09
1000	0.06
500	0.03
400	0.03
300	0.03
200	0.02
100	0.02

Table 4.6 presents the measurements of the decryption runtime for plaintext characters of size 100 to 3000 characters. Plaintext with 100 characters' length were encrypted within 0.02 milliseconds and plaintext with 3000 characters' length were executed within 0.19 milliseconds. The result shows that the higher the number of characters of a DNA. ciphertext the higher the execution time. Figure 4.6 presents the decryption runtime of the plaintext characters between 0 to 500 characters. While Figure 4.5 presents a decryption runtime of the plaintext characters between 500 to 3000 characters.

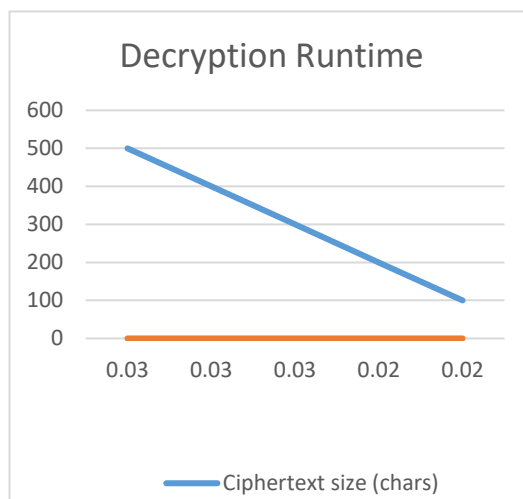


Figure 4.5: Growing Rate of the Decryption Runtime

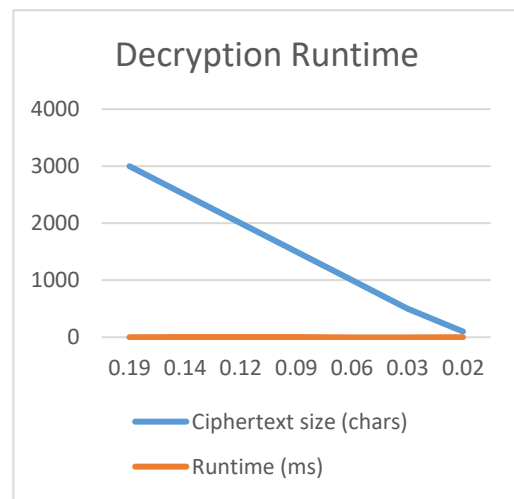


Figure 4.6: Growing Rate of the Decryption Runtime A

As depicted in Figure 4.3 and Figure 4.4, both result shows that the decryption runtime against the plaintext characters' size are linear.

4.3.2 Security Level

4.3.2.1 Cryptanalytic Attacks

In the developed 2-level data encryption technique, OTP technique produced a set of randomly organized non-repeating characters used for implementing one-time-pad because if an input ciphertext is used once it is not used again to increase the security.

In this scheme, the size of the plain text is equivalent to a One-Time-Pad. To convert the short segments of a plain text messages to ciphertext, DNA One-Time-Pad process is used. A random and unique codebook is taken into account for replacing the plain text.

This Resists to Mathematical Analysis Attack: Attackers can't break the ciphertext by solving it mathematically because of its uncertainty level in key generation. The data encryption algorithm in 2-level data encryption includes not only the XOR operation from binary plaintext to binary ciphertext, but also the confounding operation of Shift cipher value to DNA binary sequence and to the DNA ciphertext. The whole process is random and does not rely solely on mathematical calculation. Moreover, the parameter encryption algorithm simulates the process of biological genetic information flowing from DNA to protein without relying on mathematical difficulties and cryptographic characteristics. Multistep confounding operations make the cryptographic algorithm more random. Therefore, even if the attacker obtains the DNA ciphertext sequence, the security parameters cannot be decrypted by mathematical analysis. Moreover, plaintext data cannot be decrypted without security parameters. Assuming that the attacker knows all encryption algorithms, the attack types of encryption system can be divided into four types according to the attacker's mastery of data resources such as plaintext and ciphertext. In addition to the analysis of chosen plaintext attack, we also make a detailed analysis of the other three types of attacks.

Resists Only Ciphertext Attack: The attacker can only analyze the intercepted ciphertext to obtain plaintext or key. In this algorithm, the publicly transmitted ciphertext consists of OTP key and a Shift Cipher value as parameters of the ciphertext. ciphertext is generated using confounding encryption parameters, where the selection probability provided by the encoding rules are determined based on the plaintext size and a pseudo random number, to obtain the plaintext is completely impossible in this case because the attacker has to analyze the cipher text before he can obtain the plaintext to enable him use the plaintext size for the cryptanalysis. Therefore, the probability of cracking the DNA ciphertext without considering the encryption algorithm is very small. It would also be nearly impossible for an attacker to decipher DNA sequences. First, the attacker needs to know the connected information of the DNA sequence. In addition, Shift Cipher value is required and without knowing the Shift value it would be impossible to obtain the plaintext from the DNA ciphertext.

Resists Known Plaintext Attack: The attacker intercepts some pairs of plaintexts and ciphertext to break the key or algorithm. In this case that the attacker obtains some pairs of plaintext data and corresponding DNA ciphertext sequences. Because data encryption takes different security parameters in every session, each step of the plaintext data generation DNA ciphertext sequence provides security. Therefore, it is unrealistic to crack encryption algorithms and keys based on some corresponding plaintext data and DNA ciphertext sequences. It is assumed that the attacker has obtained some pairs of security parameter plaintexts and corresponding DNA cipher texts. Since each security parameter encryption use different encoding mapping parameters key1, key2 for OTP encryption level and Shift Cipher encryption level respectively, it not only has One-Time-Pad characteristics, but also simulates the biological operation process to make the algorithm more random. Therefore, it is relatively safe for known plaintext attacks.

Resists Chosen Plaintext Attack: Besides getting some corresponding ciphertext, the attacker has analyzed and obtained more information related to the key. According to the comparison of DNA ciphertext sequence and plaintext data, the operation processes of data encryption and decryption algorithm are obtained. It is unable to get encryption mapping parameters (OTP key and Shift Cipher value) through security parameters because the parameters completely depend on the plaintext parameters which are generated through a pseudo random number key as the OTP and a shift key value, therefore it is difficult to decipher the DNA ciphertext sequence during the next decryption.

Resists Chosen Ciphertext Attack: The attacker can select some ciphertext and get the corresponding plaintext. It is assumed that the attacker has mastered the data encryption algorithm, the selected DNA ciphertext sequence, the decrypted plaintext, and the key of the selected part. This one-time-pad algorithm uses different Logistic map parameters to generate the key for each encryption process. Even if the key is cracked this time, it cannot be used for the next decryption. Besides the key, the mapping parameter key0 are also different in every session and cannot be used in the next decryption. In addition, DNA ciphertext sequences were segmented and primers were added at both ends. If you don't know the index order, you can't splice the information correctly, which is a big obstacle to decipher the DNA ciphertext.

4.4 Experimental Results

Validation for the encryption and decryption of the proposed method is demonstrated below. The entire testing was done using the developed TSS module.

4.4.1 Key Generation Result

In the proposed algorithm, key K depend on plaintext P and pseudo rand number R . For the experiment, a pseudo random number R is generated, the value of R obtained is 230 and the plaintext used for the experiment is “**House 49**”, i.e. a sample address. The plaintext “House 49” has 8 characters. Therefore, the plaintext length $N=8$ and the pseudo random number $R = 230$. Table 4.7 depicts the key generation experimental result.

Table 4.7: Key generation experimental result

Operations	Input N	Input R
Enter input N and R respectively	8	230
Convert to binary N_B and R_B	1000	11100110
Compute the resulted binary lengths P and Q respectively	4	8
Find the difference $D = P - Q$	$D = \text{abs}(4-8) = 4$	
$D \neq 0 \ \&\& \ P < Q$	Yes	
Add zero(s) of size D to the prefix of N_B	00001000	11100110
$K_{XNOR} = N_B \odot R_B$	$ \begin{array}{r} 00001000 \\ 11100110 \\ \hline 11101110 \end{array} $	
Generate Key K_{OTP}	11101110	

4.4.2 Encryption Result

Table 4.8 depicts the encryption experimental result for the plaintext “House 49”. The plaintext is used as a sample address to perform the experiment.

Table 4.8: Encryption experimental result

Operation	Experiment Result							
Enter input P	H	o	u	s	e	4	9	
Shift Text $S_K = 5$	M	t	z	x	j	4	9	
Calculate P length i					i = 8			
Convert P_i to ASCII value P_{ASCII}	77	111	122	120	106	32	52	57
Convert P_{ASCII} to binary P_B	1001 101	1101 111	1111 010	1111 000	1101 010	1000 00	1101 00	111001
Compute P_B length L	7	7	7	7	7	6	6	6
$L \leq 8$	yes	yes	yes	yes	yes	yes	yes	yes
Add 0s to the prefix of P_B until it's 8 digit length Generate / Retrieve Key K_{OTP}	0100 1101	0110 1111	0111 1010	0111 1000	0110 1010	0010 0000	0011 0100	0011 1001
					11101110			
$P_{XOR} = P_B \oplus$ K_{OTP}	1010 0011	1000 0001	1001 0100	1001 0110	1000 0100	1100 1110	1101 1010	1101 0111
Convert P_{XOR} to DNA sequence C_{DNA}	GGA T	GAA C	GCC A	GCC G	GAC A	TAT G	TCG G	TCCT
Concatenate C_{DNA}	GGATGAACGCCAGCCGGACATATGTCGGTCCT							

As presented in Table 4.8, the DNA ciphertext C_{DNA} “GGATGAACGCCAGCCGGACATATGTCGGTCCT” is the experimental result of the plaintext “House 49”.

4.4.3 Decryption Experimental Result

Table 4.9 depicts the decryption experimental result for the ciphertext “GGATGAACG
CCAGCCGGAC ATATGTCGGTCCT”.

Table 4.9: Decryption result

Operation	Operation Result								
Enter input C_{DNA}	GGATGAACGCCAGCCGGACATATGTCGGTCCT								
C_{DNA} length N	N = 32								
Convert C_{DNA} to binary value C_B	101000111000000110010100100101101000010011001110110 1101011010111								
Segment C_B into 8bits	1010 0011	1000 0001	1001 0100	1001 0110	1000 0100	1100 1110	1101 1010	1101 0111	
Generate / Retrieve Key K_{OTP}	11101110								
$C_{XOR} = C_B \oplus$ K_{OTP}	1001 101	1101 111	1111 010	1111 000	1101 010	1000 00	1101 00	1110 01	
Compute C_B length L	7	7	7	7	7	6	6	6	
$L \leq 8$	yes	yes	yes	yes	yes	yes	yes	yes	
Add 0s to the prefix of P_B	0100 1101	0110 1111	0111 1010	0111 1000	0110 1010	0010 0000	0011 0100	0011 1001	
Convert C_{XOR} to ASCII value C_{ASCII}	77	111	122	120	106	32	52	57	
	M	t	z	x	j		4	9	
Shift Text (inverse) $S_K = C_{ASCII} -$ 5	72	106	117	115	101	32	52	57	
Convert to plaintext characters P_i	H	o	u	s	e		4	9	

In Table 4.9, the decryption technique transformed the ciphertext C_{DNA} to plaintext. The decryption function used the K_{OTP} and the C_{DNA} to decrypt the information. Therefore;

$K_{OTP} = 11101110$ and $C_{DNA} = GGATGAACGCCAGCCGGACATATGTCGGTCCT$.

The C_{DNA} “GGCGGAACGCGTGCTCGAGTTATGTCGGTCCT” decryption result is “House 49D” as depicted in table 4.9.

4.5 Entropy Result

Table 4.7 depicts the comparative analysis on Two Level Text Data encryption (Vidhya & Rathipriya, 2018) and improved 2-Level Data Security approach.

Table 4.10: Shannon Entropy with Frequencies of alphabet symbols

Two Level Text Data encryption						Improved 2-Level Data Security approach				
Number of characters	Entropy Value	Frequencies of Alphabet Symbols				Entropy Value	Frequencies of Alphabet Symbols			
		A	C	T	G		A	C	T	G
3	1.88	0.19	0.38	0.13	0.31	1.73	0.08	0.17	0.25	0.50
101	1.90	0.28	0.29	0.13	0.29	1.96	0.25	0.23	0.17	0.35
482	1.92	0.32	0.27	0.12	0.29	1.95	0.25	0.23	0.16	0.35
599	1.93	0.31	0.25	0.13	0.30	1.95	0.25	0.23	0.16	0.35
601	1.94	0.32	0.25	0.14	0.29	1.95	0.25	0.24	0.16	0.35
1809	1.93	0.32	0.27	0.13	0.29	1.95	0.25	0.24	0.16	0.35

Table 4.10 presents the comparative analysis on Two Level Text Data encryption (Vidhya & Rathipriya, 2018) and an improved 2-Level Data Security approach. One measure quality of cryptography is to measure the entropy of the output. The entropy result in Table 4.10 shows that the improved 2-Level Data Security approach has high entropy

compared to the entropy values of Two Level Text Data encryption (Vidhya & Rathipriya, 2018). Therefore, having and using data with high entropy is completely random and no meaningful patterns can be found.

CHAPTER FIVE

5.0 CONCLUSION AND RECOMMENDATION

5.1 Conclusion

In the current era of technology, protecting the confidentiality of stored information as well as transmitted data is extremely crucial. The concept of DNA cryptography is inspired by DNA molecules which have the capacity to store, process and transmit information in a mangled form.

This research used Shift Cipher in reshuffling the position of the text and OTP technique in generating a Pseudo random number encryption key which makes the technique strong enough to protect information from all attacks including brute force attacks. The XNOR and XOR operation concepts used in generating key and encrypting information has make the algorithm efficient and less complex.

Furthermore, the experimental result of the encryption and decryption runtime indicates that the developed 2-level data security approach has high speed in times of operation and this will open new horizons for researchers in the field of DNA computing and information security to leverage upon.

5.2 Recommendation

Transport System Security (TSS) module can be deploy on a different server because it has Object Oriented features. It contains functions and classes that can be invoke from any platform once the script is linked. Therefore, the researcher recommends deployment of Transport System Application and TSS module to be hosted on different server to increase security of the encryption and decryption algorithm, confidentiality of

information and to prevent unwanted theft by unauthorized users who might get access to the TSA server.

5.3 Contributions to Knowledge

One Time Pad (OTP) and XNOR concept used in generating key makes the TSS module more secure and easy to use and the invention of TSS module for Uber Transport Management Systems opens new horizons for other researchers in the field of DNA computing and information security.

The result of the encryption and decryption algorithm tests indicates that the developed 2 level data encryption approach has good execution time compare to other researches on 2 level data security techniques.

The developed 2-level data security approach has proffered good algorithm for securing the confidentiality of information between two or more clients during communication using a symmetric encryption key.

REFERENCES

- Aggarwal, A., & Kanth. P. (2014). Secure data transmission using DNA encryption. *International Journal of Advanced Research in Computer Science*, vol. 5, no. 6, pp. 57-61.
- Anwar, T., Kumar, A., & Paul, S. (2015). DNA cryptography based on symmetric key exchange. *International Journal of Engineering and Technology (IJET'15)*, vol. 7, no. 3, pp. 938-950.
- Beecroft, M. (2019). The future security of travel by public transport: A review of evidence. *Journal Research in Transportation Business & Management*. ISBN: 2210-5395.
- Babu, E. S., Prasad, Mahit, M. K., & Raju, C. N. (2016). Inspired pseudo biotic DNA based cryptographic mechanism against adaptive cryptographic attacks. *International Journal of Network Security*, vol. 18, no. 2, pp. 291-303.
- Bhavithara, M., Bhrinta, A. P., & Kamaraj, A. (2016). DNA based encryption and decryption using FPGA. *International Journal of Current Research and Modern Education (IJCRME'16)*, pp. 89-94.
- Basha, S. S., Emerson, I. A., & Kannadasan R. (2015). Survey on molecular cryptographic network DNA (MCND) using big data. *2nd International Conference of Computer Science on Big Data and Cloud Computing (ISBCC'15)*, vol. 50, pp. 3-9.
- Donald S. C. (2018). Analysis of Proposed Consent Order to Aid Public Comment in the Matter of Uber Technologies. Inc., File No. 1523054.
<https://www.federalregister.gov/documents/2018/04/25/2018-08600/uber-technologies-inc-analysis-to-aid-public-comment>. Retrieved on 09/03/2020
- Darbari, M., & Prakash, V. (2014). A new framework of distributed system security using DNA cryptography and trust based approach. *International Journal of Advancements in Research and Technology*, vol. 3, no. 3, pp. 1-4.
- El-Latif & Moussa, M. I. (2019). Information hiding using artificial DNA sequences based on Gaussian kernel function. *Journal of Information and Optimization Sciences* ISSN: 0252-2667 (Print) 2169-0103 (Online).
- Findin M. (2014). Security in Transportation: Preparing Defenses for a New Connected Era. Talos, Security and Trust Organization, Active Threat Analytics, and Security Research and Operations. <https://www.cisco.com/c/dam/assets/docs/transportation-security.pdf>. Retrieved on 14/04/2020
- Gambhir, S., & Rakesh, K. Y. (2019). DNA Based Cryptography Techniques with Applications and Limitations. *International Journal of Engineering and Advanced Technology (IJEAT)* ISSN: 2249 – 8958, Volume-8 Issue-6, August 2019.
- Gulati, N., Kalyani, S. (2016). Pseudo DNA cryptography technique using OTP key for secure data transfer. *International Journal of Engineering Science and Computing*, vol. 6, no. 5, pp. 5657-5663.
- Harsh, D. T., & Jae, H. K. (2018). Novel Method for DNA Based Elliptic Curve Cryptography for IoT Devices. *ETRI Journal*, Volume 40, Number 3, June 2018.

- Hazra, A., Ghosh, S., & Jash, S. (2018). A Review on DNA Based Cryptographic Techniques. 20(6), International Journal of Engineering and Advanced Technology (IJEAT) 1093–1104. <https://doi.org/10.6633/IJNS.201811>
- Jain, S., Rani, M. & Asha N. (2014). Enhancing asymmetric encryption using DNA-based cryptography. International Journal of Computer Science Trends and Technology (IJCSST'14), vol. 2, no. 3, pp. 7-11.
- Kumar S. R. (2014). Review on DNA Cryptography. Research at Electronics and Communication Science Unit Indian Statistical Institute. International Journal of Engineering and Technology (IJET'15).
- Kazazi, N. S., & Torkaman M. R. N. (2015). A method to encrypt information with DNA-based cryptography. International Journal of Cyber Security and Digital Forensics (IJCSDF'15), vol. 4, no. 3, pp. 417-426.
- Mark B. (2019). The future security of travel by public transport: A review of evidence. International Journal of Transportation Business and Management, vol. 32, 100388.
- Mahalaxmi, T., Raj, B. B., & Vijay, J. F. (2016). Secure data transfer through DNA cryptography using a symmetric algorithm. International Journal of Computer Applications, vol. 133, no. 2, pp. 19-23.
- Newton, A. (2004). Crime on public transport: “Static” and “non-static” (moving) crime events. Western Criminology Review, 5(3), 25–42. Retrieved on 17/07/2020 https://www.westerncriminology.org/documents/WCR/v05n3/article_pds/newton.pdf.
- Newton, Andrew D. (2016) Crime, Transport and Technology. In: The Routledge Handbook of Technology, Crime and Justice. Routledge, London, UK, pp. 281-294. ISBN 9781138820135. Retrieved on 25/08/2020. <https://coeminna.jspnode.com.ng/>
- Olga, T., & Borda, E. M. (2013). Security and complexity of a DNA-based cipher, In Roedunet International Conference (RoEduNet), 11th IEEE International Conference, pp. 1-5.
- Priyadarshani, K., Bama, R., & Deivanai, S. (2014). Secure data transmission using DNA sequencing. IOSR Journal of Computer Engineering (IOSRJCE'14), vol. 16, no. 2, pp. 19-22.
- Pramanik, S., & Kumar, S. S. (2012). DNA cryptography. Electrical & Computer Engineering 7th IEEE International Conference, pp. 551-554.
- Purusothaman, T., Saravanan, K. (2016). DNA-based secret sharing algorithm for a multicast group. Asian Journal of Information Technology, vol. 15, no. 15, pp. 2699-2701.
- Raj, B. B., & Panchami, V. (2015). DNA-based cryptography using permutation and random key generation method. International Journal of Innovative Research in Science, Engineering and Technology, vol. 3, no. 5, pp. 263-267.
- Singh, G. & Kamaljit K. (2015). A Review to an Invincible Cryptographic Approach: DNA Cryptography. 4(1), 327–331. <https://doi.org/10.17148/IJARCCE.2015.4175>
- Sohal S. (2018). BDNA-A DNA inspired symmetric key cryptographic technique to secure cloud computing. Journal of King Saud University of Computer and Information Sciences.

- Stephen, E. (2019). Lyft Investigates Claim That Employees Improperly Accessed Customer Data.
<https://www.thedrive.com/tech/17993/lyft-investigates-claim-that-employees-improperly-accessed-customer-data>. Retrieved on 09/03/2020.
- Sudhakara, P. T., Aditi S., Chahat, K. & Prantik, B. (2016). An Extended Hybridization of Vigenere and Caesar Cipher Techniques for Secure Communication. 2nd International Conference on Intelligent Computing, Communication & Convergence (ICCC-2016).
- Tanaka, K. Okamoto, A. and Saito I. (2015), Public-key system using DNA as a one-way function for key distribution. *Journal of Biosystems Engineering* 81, 1, pp. 25-29
- Taha, M. S., Shafry, M., & Rahim, M. (2019). Combination of Steganography and Cryptography: A short Survey. *IOP Conference Series: Materials Science and Engineering*.
<https://doi.org/10.1088/1757-899X/518/5/052003>.
- Tiwari, H. D., & Hyung, J. K. (2018). Novel Method for DNA-Based Elliptic Curve Cryptography for IoT Devices. *ETRI Journal*, Volume 40(3), 396–409.
<https://doi.org/10.4218/etrij.2017-0220>
- Vidhya, E., & Rathipriya, R. (2018). Two Level Text Data Encryption using DNA Cryptography. *International Journal of Computational Intelligence and Informatics*, Vol. 8: No. 3, 8(3), 106–118.
- Yunpeng, Z., Xin L., Yongqiang M., & Liang-Chieh, C. (2017). An Optimized DNA Based Encryption Scheme with Enforced Secure Key Distribution. Springer Science Business Media, LLC. DOI 10.1007/s10586-017-1009-y

APPENDIX

TSS module Source Code

```
<?php

include("key_generator.php");

include("xor_conversion.php");

include("DNA_encryption.php");

include("DNA_decryption.php");

include("ASCII_bin.php");


function TSS_encrypt($string_var, $rand_val) {

$string_length = strlen($string_var);

$count = '0';

$DNA_result = "";

//$id = random_bytes(8);

    for ($count=0; $count < $string_length; $count++) {

        //selection of char one at a time from the submitted string

        $character = substr($string_var, $count, 1);

        //conversion of character to ASCII code
```

```

$char_ASCII = ord($character);

//conversion of ASCII code to binary

$ASCII_bin = decbin($char_ASCII);

//convert each character ASCII_bin to base 8

$char_bin8 = ASCII_bin8($ASCII_bin);

//generate key for each character from the input string

$k_bin8 = TSS_Key_Generator($string_length, $rand_val);

//copuute the XOR for each character binary and it's key

$xor_bin_res = xor_binary($k_bin8, $char_bin8);

//convert to DNA Neoclode (ATCG)

$DNA_Nucleotide = DNA_encryption($xor_bin_res);

//consolidate each character DNA Neoclode (ATCG)

$DNA_result = $DNA_result.$DNA_Nucleotide;

}

```

```
return $DNA_result;
```

```
}
```

```
//Decryption Code
```

```
function TSS_decrypt($DNA_result, $rand_val){
```

```
//Assignment statement
```

```
$DNA_cipher = $DNA_result;
```

```
//compute cipher text string length
```

```
$DNA_string_length = strlen($DNA_cipher);
```

```
//convert DNA cipher text to binary
```

```
$DNA_bin = DNA_decryption($DNA_cipher, $DNA_string_length);
```

```
//compute string length of ciphertext (i.e in binary form)
```

```
$Cbin_string_length = strlen($DNA_bin);
```



```

//assign fix binary size to the variable "bit_size"

$bit_size = '8';

//initialization

$c_XOR_bin8 = "";

//initialization

$Decryption_res = "";

//divide cipher length obtained from the binary form by base 8

$Cbin_length = $Cbin_string_length / $bit_size;

//generate cipher text key

$c_key_bin8 = TSS_Key_Generator($Cbin_length, $rand_val);

for ($i=0; $i < $Cbin_string_length; $i+=8) {

    $character_bin = substr($DNA_bin, $i, 8);

    $c_XOR_bin8 = xor_binary($c_key_bin8, $character_bin);

    //convert each binary digits in base 8 to decimal values

    $ascii_code = bindec($c_XOR_bin8);

```

```

//convert each decimal value to it's corresponding character (plaintext)

$decrypt_char = chr($ascii_code);


//concatenate the characters

$Decryption_res = $Decryption_res.$decrypt_char;


//echo $c_XOR_bin8.'<br>';

}


//return plaintext result

return $Decryption_res;


}

?>

```