Chapter I

INTRODUCTION

A. Background of the Study

The use of cryptosystems in strengthening security services worldwide is todays current trend with rise of cyber criminals and attacks. There are lots of encryption algorithm used to counter these attacks and one of this is an NTRU-based Cryptosystem called, ITRU Encryption Algorithm. This algorithm adapts how NTRU is much faster in generating keys, likewise in terms of encryption and decryption time as compared to other public key cryptosystems. However, it uses a ring of integers, instead of a ring of polynomials, for a simple parameter selection, invertibility, and successful message decryption. Furthermore, the same with NTRU, it is said to be much more compatible with mobile devices and other portable devices.

Since the wide-spread era of smartphones and the rapid expansion of mobile platforms, Instant Messaging (IM) on mobile had been the most current way of communicating online because of its convenience and portability. As IM on mobiles goes in trend, most business organizations and consumers had used this technology on their sole purposes. Security had been the main concern of these entities in terms of their business transactions and processes. Hence, there is a need for better end to end message encryption to provide a secure medium or communication due to profound risks that are caused by online infiltrators.

Instant Messaging on Mobile requires real-time and fast response to security, and because of that, the proper algorithm is needed for this certain process. ITRU Encryption Algorithm surpasses other encryption algorithms, such as AES, DES or Blowfish, in terms of performance. With this capability of ITRU Encryption Algorithm, it is the best algorithm to be used for mobile IM.

Existing Algorithm:

a. ITRU Parameters

Parameter	Description
p’	Small modulus
q’	Large Modulus
f’	Private integer for private key generation
g’	Private random integer for public key generation
r’	Private random integer for cipher-text generation
m’	Decimal representation of the message
Kpr	Private key pair (f’, Fp’)
Kpb	Public key parameter h’
a’	Intermediate parameter
C’	Decrypted message

b. ITRU Parameter Generation

1. Initialize p’ = 1000
2. Randomly select an odd integer. Assign value to f’.
3. Randomly select two integers. Assign values to g’ and r’.
4. Convert m’ into its decimal representation based on the ASCII table values.
5. Set q’ = prime integer > (p’ × r’ × g’ + f’ × m’).

c. Key Creation

1. Compute modular multiplicative inverse $Fp' = f'^{-1} \bmod p'$.
2. Compute modular multiplicative inverse $Fq' = f'^{-1} \bmod q'$.
3. Sender obtains private key pair Kpr = (f’, Fp’).
4. Compute for public key $Kph = h' = (p' \times Fq' \times g') \bmod q'$.

d. Encryption

1. Encrypt message m’ using $e' = ((r' \times h') + m') \bmod q'$.
2. Cipher-text e’ is sent to the intended recipient.

e. Decryption

1. Compute $a' = (f' \times e') \bmod q'$.
2. Recover message using $C' = (Fp' \times a') \bmod p'$.
3. Convert C’ back from decimal back to ASCII encoding character.

Existing Algorithm (With Highlighted Problems):

a. ITRU Parameters

Parameter	Description
p’	Small modulus
q’	Large Modulus
f’	Private integer for private key generation
g’	Private random integer for public key generation
r’	Private random integer for cipher-text generation
m’	Decimal representation of the message
Kpr	Private key pair (f’, Fp’)
Kpb	Public key parameter h’
a’	Intermediate parameter
C’	Decrypted message

b. ITRU Parameter Generation

1. Initialize p’ = 1000
2. Randomly select an odd integer. Assign value to f’.
3. Randomly select two integers. Assign values to g’ and r’. **Problem No. 1**
4. Convert m’ into its decimal representation based on the ASCII table values.
5. Set q’ = prime integer > (p’ × r’ × g’ + f’ × m’).

c. Key Creation

Problem No. 2

1. Compute modular multiplicative inverse $Fp' = f'^{-1} \bmod p'$.
2. Compute modular multiplicative inverse $Fq' = f'^{-1} \bmod q'$.
3. Sender obtains private key pair $Kpr = (f', Fp')$.
4. Compute for public key $Kph = h' = (p' \times Fq' \times g') \bmod q'$.

Problem No. 3

d. Encryption

1. Encrypt message m' using $e' = ((r' \times h') + m') \bmod q'$.
2. Cipher-text e' is sent to the intended recipient.

e. Decryption

1. Compute $a' = (f' \times e') \bmod q'$.
2. Recover message using $C' = (Fp' \times a') \bmod p'$.
3. Convert C' back from decimal back to ASCII encoding character.

B. Statement of the Problems

This study endeavors to provide solutions to the following problems regarding the enhancement of ITRU encryption algorithm applied in mobile instant messaging client applications:

1. **The algorithm is unable to encrypt Unicode characters except for characters from the ASCII table.**

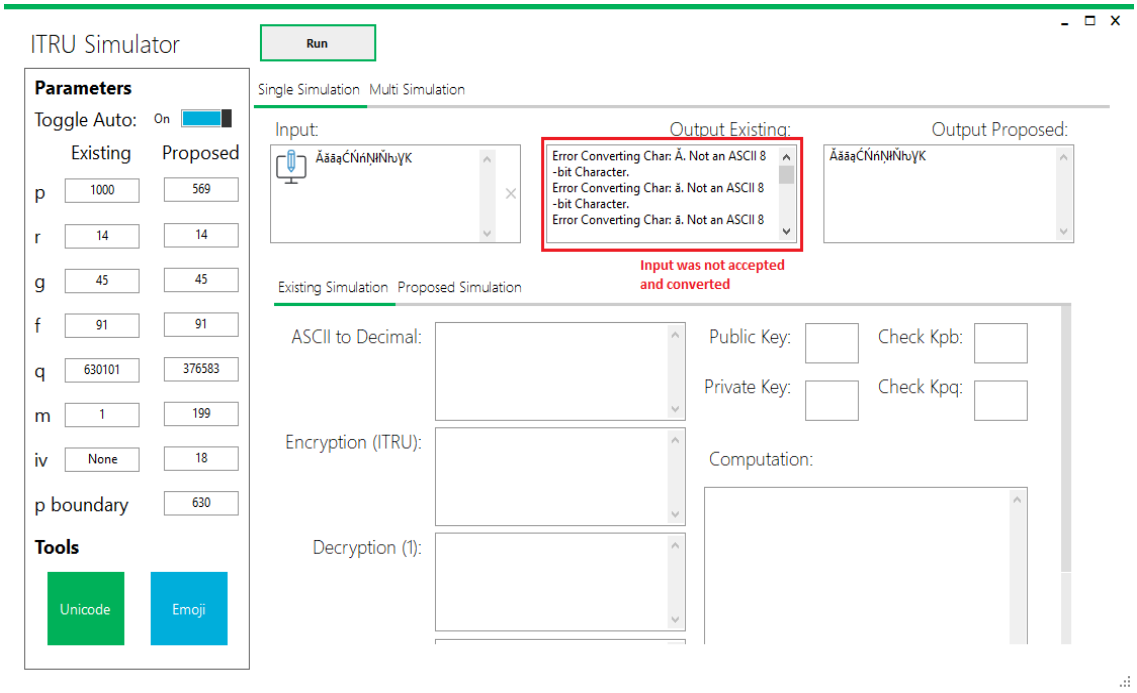


Figure No. 1: Failed Test Simulation Problem No. 1

The figure shows an example of a failed test simulation using a sample sentence with ASCII characters.

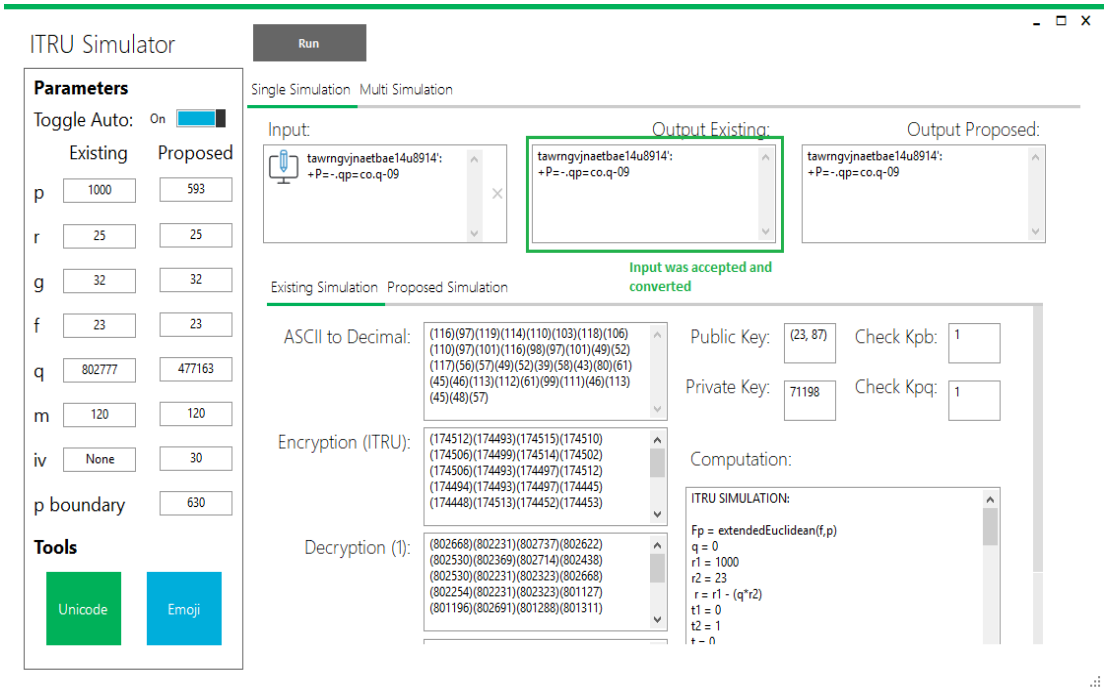


Figure No. 2: Successful Test Simulation for Problem No. 1

The figure shows an example of a successful test simulation using a sample sentence without ASCII characters.

The algorithm converts the message string to its decimal representation using the ASCII standard encoding. The ASCII encoding is an 8-bit encoding scheme from decimal 0 to decimal 255. Other character encodings, such as UTF-8 or Unicode are not accepted by the string-to-decimal conversion scheme. Unicode encoding is dominant than ASCII encoding since it allows different characters outside the ASCII table, such as Latin, Arabic, and Chinese letters.

2. **The algorithm does not trap errors produced when creating the public key using the Extended Euclidean Method.**

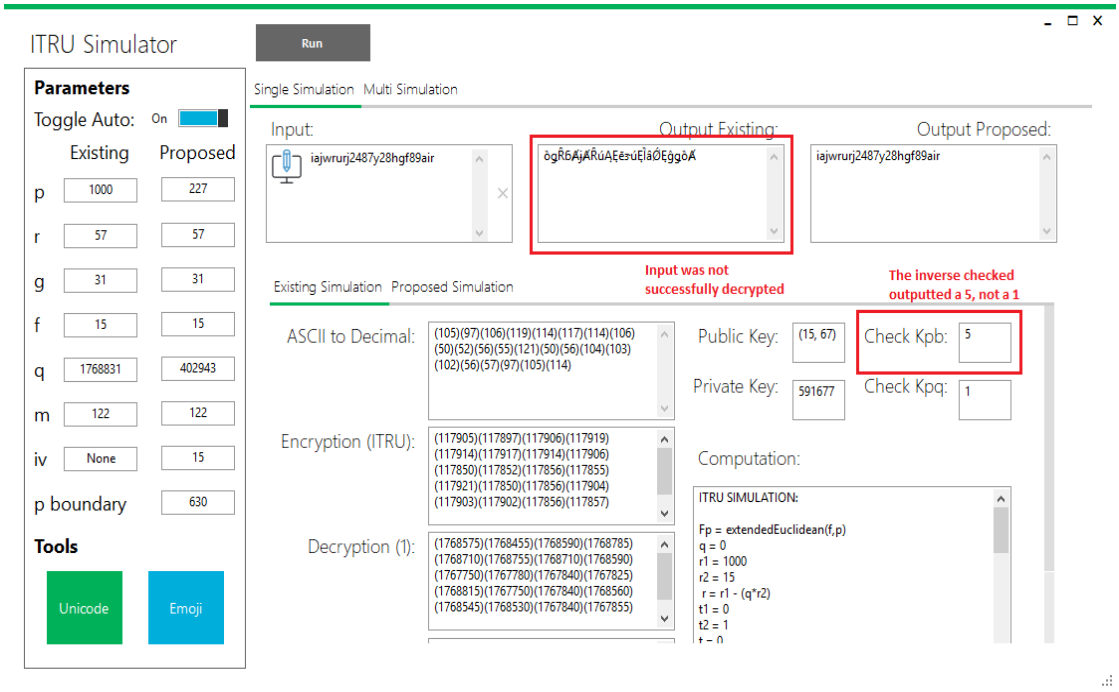


Figure No. 3: Failed Test Simulation for Problem No. 2

The figure shows an example of a failed test simulation.

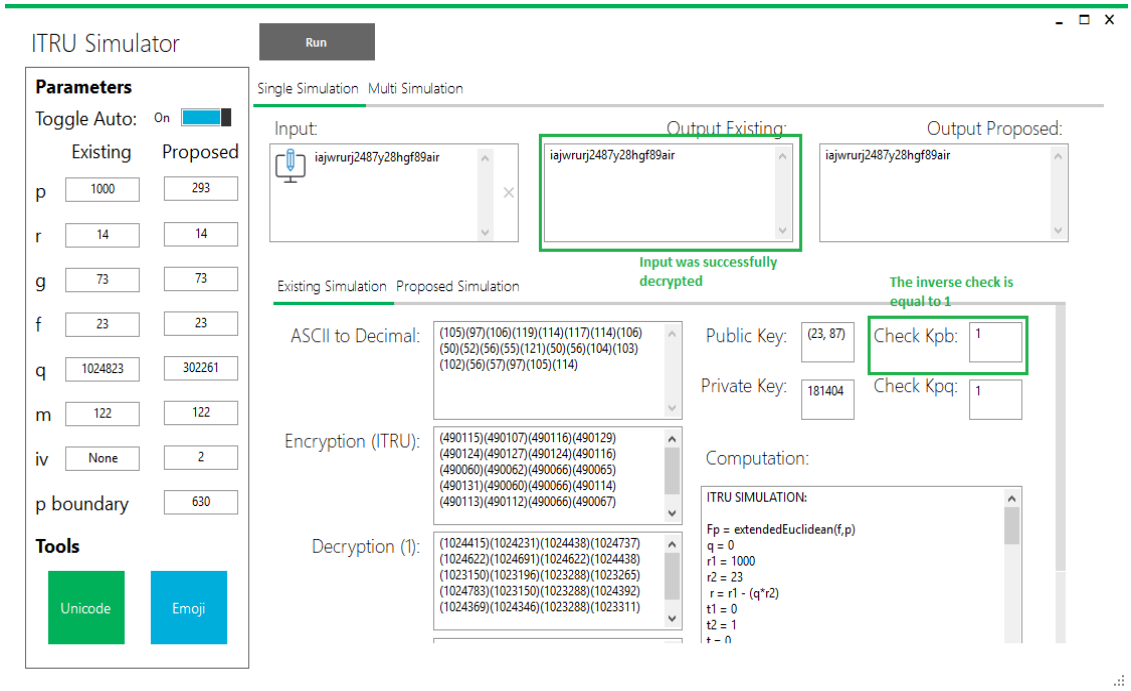


Figure No. 4: Successful Test Simulation for Problem No. 2

The figure shows an example of successful test simulation with the same message as in Figure No. 3.

The algorithm states that a number has a modular multiplicative inverse if the product of the number and its inverse is equal to 1, or in the algorithm's case, f x Fp mod p = f x Fq mod q = 1. Though the algorithm states this, no measures are created to prevent the encryption/decryption of an instance where f x Fp mod p or f x Fp mod q are not equal to 1. Successful decryption will only occur if f x Fp mod p = f x Fq mod q = 1, given the parameters of the ITRU instance.

3. **The algorithm creates an encrypted message with a pattern-like sequence similar to the patterns of the plaintext message.**

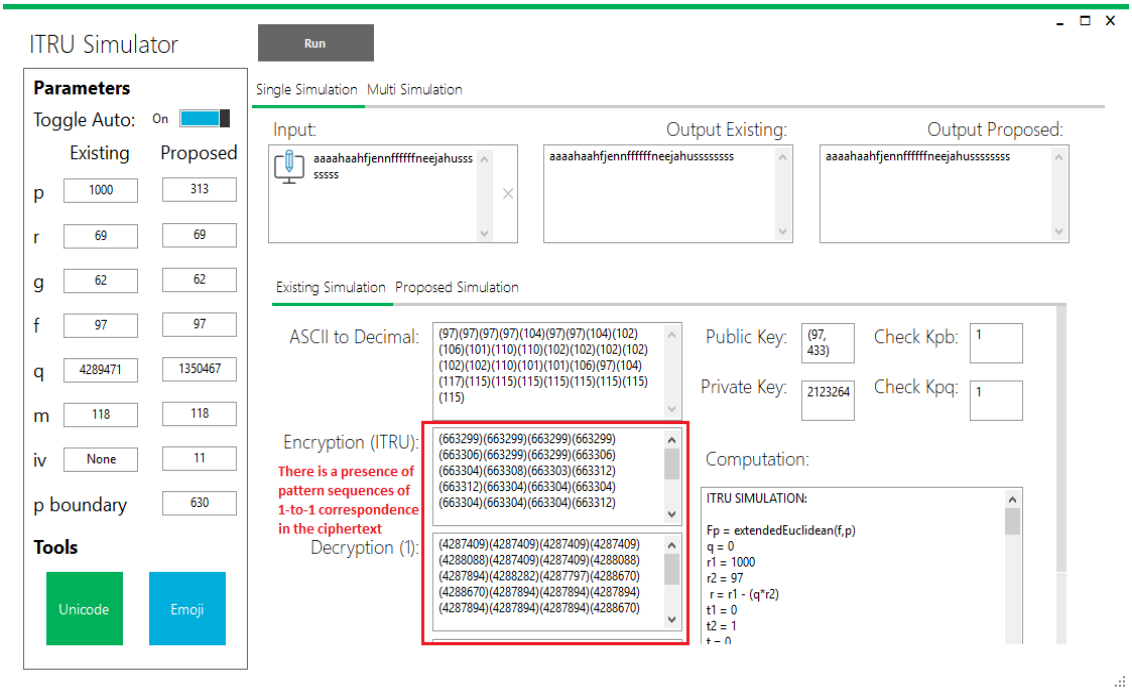


Figure No. 5: Failed Test Simulation for Problem No. 3

An example of a failed test simulation with repeating patterns.

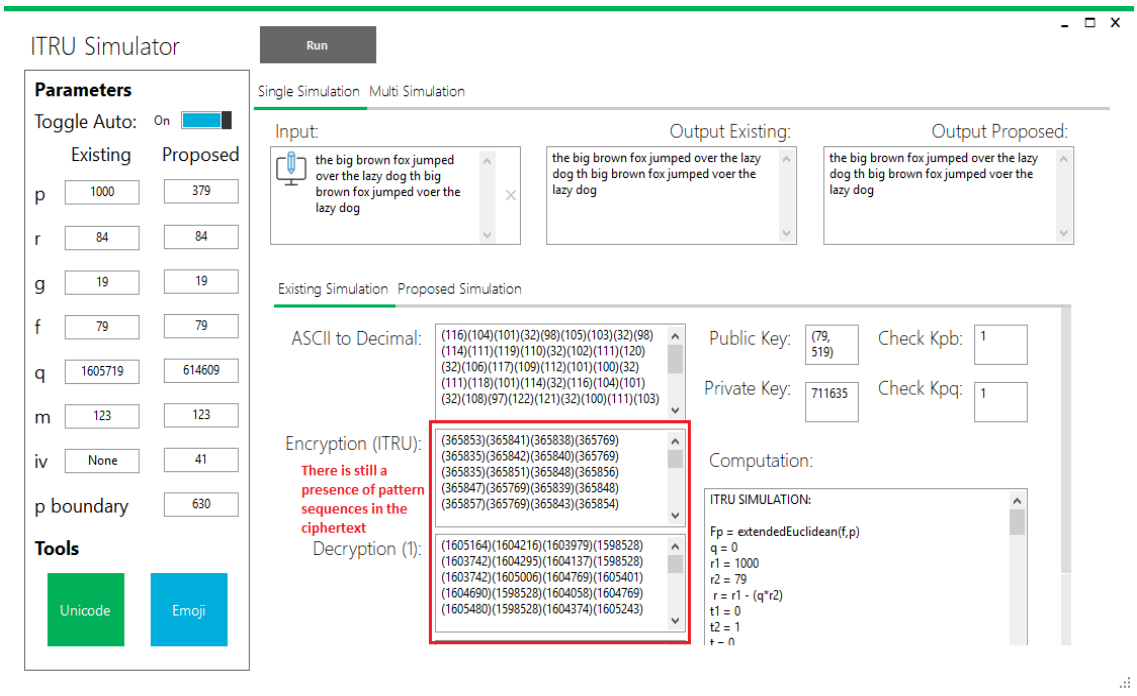


Figure No. 6: Failed Test Simulation for Problem No. 3

An example of a failed test simulation with repeating patterns.

The simulation has shown that the algorithm produces a cipher text with a pattern that is dependent on the pattern of the plaintext itself. Each letter corresponds to a unique value in the cipher text, and as such, thorough analyzation of the patterns of the cipher text can uncover common terms used in the language of the plaintext, and consequently, revealing the plaintext itself. The use of algorithms to examine patterns, such as frequency analysis algorithms, can allow attackers to decode the ciphertext without the private and public keys.

C. Objectives of the Study

The study aims to create an enhancement of the ITRU Encryption Algorithm that will be applied to Mobile Instant Messaging Client Applications. Specifically, the study aims to accomplish the following specific objectives:

1. **To expand the scope of the encoding scheme that the algorithm uses beyond the ASCII table conversion.**

   Using the ASCII table as a basis for decimal conversion limits the amount of characters that can be accepted by the system. If another encoding system is used, such as the UTF-8 encoding system that is widely used in androids and iOS smartphones, characters beyond the ASCII encoding can be accepted a thus be encrypted and decrypted.

2. **To remove the decryption errors caused by the computation of non-inverse keys.**

   Creating an error-detecting and error-correcting module in the algorithm that checks for the inversibility of the keys would decrease the frequency of decryption failure occurrences in the algorithm.

3. **To randomize the cipher text produced by the algorithm so that frequency analysis attacks can be prevented.**

   Randomizing the cipher text produced by the algorithm in such a way that it does not resemble the original structure of the plaintext can increase the security of the encryption. Frequency analyzers will not be as efficient in predicting the frequent words used in the plaintext.

D. Significance of the Study

This study will be very beneficial to the following :

1. **For Security Experts** – this study will help system security experts and professionals in creating a more secure system for their clients. Experts can also use this study's algorithm in their seminars and lectures.

2. **For Developers/Programmers** – this study's algorithm can be of use for developers and programmers, mainly in system desings. Since system security is a key part of a system design, the use of this study's algorithm by the users can be a vital asset to their system design.

3. **For Mobile IM Users** – this study will also be helpful to non-professional communities such as people who always use IM applications on their smartphones. This study will help them in making their personal data more secure and safe.

4. **For Future Researchers** – future researchers who will be studying the field of cryptography and security can use this study as a reference to their own studies. This will also provide them with the enhanced ITRU encryption algorithm documentation when they need to implement it to their researches.

E. Scope and Limitations

The study focuses on the ITRU Encryption Algorithm and its application to the Mobile Instant Messaging Client Application. The scope of the study includes the algorithm of ITRU as stated in the research of Gaithuru, et. al. and the Extended Euclidean Algorithm used in the computation of the modular multiplicative inverse of the encryption keys. The study will also focus on mobile instant messaging applications as its applications in the enhancement of the algorithm. Though the algorithm can be applied to an Android Ice Cream Sandwich version, the study will use an Android Jellybean version and as well as other newer versions since these are the prominent Android OS in use as of today. The study will only be utilizing devices running under an Android Operating System.

The study will only focus on mobile as its application platform. The study will also only analyze the problems occurring due to the existing algorithm, and it will not cover problems that can occur due to technical issues or problems caused by factors other than the algorithm, such as a person intentionally or unintentionally giving

another person their mobile phone. The study limits itself to only Instant Messaging, and excludes other means of communication such as electronic mail, SMS, etc.

F. Definition of Terms

Algorithm. A procedure or formula for solving a problem, based on conducting a sequence of specified actions.

Application. A software designed and written to fulfill a particular purpose of the user.

Byte-Encoding. Encoding of characters using one to four 8-bit blocks.

Ciphertext. Encrypted text.

Client. A person or group that uses the professional advice or services of a lawyer, accountant, advertising agency, architect, etc.

Cryptosystems. An implementation of cryptographic techniques and their accompanying infrastructure to provide information security services.

Decryption. The process of transforming data that has been rendered unreadable through encryption back to its unencrypted form.

Encryption. The transaltion of data into a secret code.

Extended Eucledian Algorithm. A method of computing the greatest common divisor (GCD) of two integers *a* and *b*. It allows computers to do a variety of simple number-theoretic tasks, and also serves as a foundation for more complicated algorithms in number theory.

Greatest Common Divisor (GCD). The highest common divisor (Hardy and Wright 1979, p. 20), of two positive integers a and b is the largest divisor common to a and b.

Integer. A number with no fractional part.

Instant Messaging. The exchange of text messages through a software application in real-time.

ITRU. An NTRU-based cryptosystem that uses a ring of integer instead of a polynomial ring with integer coefficient.

Key. A random string of bits created explicitly fo scrambling and unscrambling data.

Lattice. A regular geometrical arrangement of points or objects over an area or in space.

Message. A verbal, written, or recorded communication sent to or left for a recipient who cannot be contacted directly.

Messaging. The act of sending a message.

Mobile Instant Messaging Client Application. A type of instant messaging held on applications that can be downloaded on smartphones. Generally needs an internet connection and mostly free of download.

Mobile Application. A computer program designed to run on a mobile device such as a smartphone or tablet computer.

Modulo. Uses a modulus of a specified number.

NTRU. Also known as Number Theory Research Unit, is a patented and open source public-key cryptosystem that uses lattice-based cryptography to encrypt and decrypt data.

Plaintext. An ordinary readable text before being encrypted into ciphertext or after being decrypted.

Private Key. Also known as Secret Key, is a variable that is used with an algorithm to encrypt and decrypt code.

Probabilistic. A chance of variation based to a theory of probabilty.

Public Key. A large numerical value that is used to encrypt data.

Quantum Conputers. A computer that makes use of the quantum states of subatomic particles to store information.

Ring of Integer. A set of integers ..., -2, -1, 0, 1, 2, ..., which form a ring.

Shortest Vector Problem. Also known as SVP, refers to the restriction to integer lattices without of loss to generality.

Smartphones. A mobile phone that performs many of the functions of a computer, typically having a touchscreen interface, internet access, and an operating system capable of running downloaded applications.

SMS. Also known as Short Message Services.

UTF-8. Stands for Unicode Transformation Format. A compromise character encoding capable of encoding all 1,112,064 that can be as compact as ASCII but can also contain any Unicode characters

Chapter II

RELATED LITERATURE AND STUDIES

A. Foreign Literature

According to the article "UTF-8 Everywhere", the first version of the Unicode standard had code points limited to 16 bits in length, but it was discovered that the early standard was not enough for Unicode. The UTF-16 encoding was created so that systems can use non-16-bit characters for their operations. Most Unicode code points, the numerical value in Unicode code space, can take the same number of bytes in UTF-8 and UTF-16. This includes the Russian, Hebrew, Greek, and other code points that can take 2 or 4 bytes in encoding. Latin letters and punctuation marks with the rest of the ASCII takes more space in UTF-16, while Asian characters take space in UTF-8. In the comparison of all the UTF variants, UTF-16 takes about 50% more space than UTF-8 on the data, saving 20% for dense Asian texts. The article concluded that the best approach is to use UTF-8 narrow strings everywhere and converting them back and forth when using platform APIs.

In the documentation of Android Developers' "Android Charset Class", a character is defined as a Java character or a Unicode code point in the range of U+0000 to U+FFFF. The most important mapping capable of representing the characters in Android is the Unicode Transformation or UTF charsets, common of

which is the UTF-8 and UTF-16 family. The UTF-8 family encodes characters using 1 to 4 bytes while the UTF-16 family encodes 2 bytes for each character.

The article of Olzak entitled "Chapter 7: The Role of Cryptography in Information Security" discussed the Monoalphabetic Substitution Cipher, a ciphering technique that involves converting plaintext to ciphertext with each letter in the plaintext having an equivalent cipher alphabet. A character in the cipher alphabet is substituted with a character in the plaintext, usually with a shift in the cipher alphabet. The article stated that a Monoalphabetic Substitution Cipher can be broken with frequency analysis, or measuring the number of frequent letters appearing in the cipher text. A cryptanalyst that has knowledge of the origin of the ciphertext can count the occurrence of letters in a sample text and cipher text and then record it to a table, and then rank the number of frequent letters from most frequent to least frequent.

According to the article "The Euclidean Algorithm and the Extended Euclidean Algorithm" from DI Management Home, the Euclidean Algorithm is an efficient method to compute the Greatest Common Divisor or GCD of two integers. The proof uses the division algorithm, which states that for any two integers a and b with $b > 0$, there is a unique pair of integers q and r such that $a = qb + r$ and $0 <= r < b$. The article states that the modular inverse of an integer e modulo n is defined as the value of d such that $ed = 1 \bmod n$, wherein $d = (1/e) \bmod n$ or $d = e^{-1} \bmod n$. It is stated that the inverse exists if and only if $GCD(n, e) = 1$.

The United States Patented invention of Arnold, W. C and et. al, entitled "Searching for Patterns in Encrypted Data", the procedure detects and finds the pattern or patterns that are present in the encrypted data, provided that the encryption technique used is one of a variety of simple methods often used by computer programs such as viruses. The procedure applies an invariance transformation to the chosen pattern to be matched to obtain a "reduced pattern", apply the same reduction to the encrypted data to obtain the "reduced data", check the existence of a pattern match from the reduced pattern and reduced data, corroborate any such matches and providing information about the match.

According to the article "Cryptanalysis Hints", there are a lot of different techniques that can be used to decipher a cipher text based on frequencies. Cryptanalysts can focus on identifying common pairs of letters that occurs in the ciphertext. They can also identify the smallest words first by distinguishing the cipher alphabet corresponding to the spaces of the ciphertext. Tailor made frequency tables can also be used to measure frequencies based on the standards and rules followed by the ciphertext and encryption scheme. Cryptanalysts can also play the guessing game by using their wits to try to guess the ciphertext itself based on cultural backgrounds and other attributes of the encrypted cipher text.

B. Foreign Studies

According to the study of Gaithuru and et. Al. (2017) entitiled "ITRU: NTRU-Based Cryptosystem Using Ring of Integers", the ITRU encryption algorithm is an NTRU-based cryptosystem using the ring of integers. The proposed algorithm uses integers as opposed to polynomials, as is the structure of the classical NTRU. The paper described a description of the theoretical model of ITRU, the parameter selection conditions as well as an illustration of its implementation using an example. The performance of ITRU is then contrasted with that of NTRU. The study showed that the selection of parameters in accordance with the described parameter selection algorithm ensures a successful message decryption, thereby eliminating the risk of decryption failure. However, owing to ITRU's integer structure, it is susceptible to similar attacks as those posed on encryption schemes which have integer structures such as RSA. Such attacks include the number field sieve algorithm and Pollard's Rho Method.

According to Hindocha, N. (2003) in his article entitled "Instant Insecurity: Security Issues of the Instant Messaging", one of the threats to Instant Messaging applications is hijacking and impersonation of hackers. To get the account information of users, hackers use password-stealing trojan horses that finds the password for the instant messaging account used by the victim and sends it back to the hacker. Furthermore, since none of the four major instant messaging protocols encrypt their network traffic, attackers can highjack connections using a man-in-the-middle

attack, inserting messages into an ongoing chat-session and impersonating one of the chatting parties.

C. Comparative Analysis

Table No. 1: Comparative Execution Times (in Milliseconds) of Decryption Algorithms with Different Packet Size

The results show the superiority of NTRU algorithm, which is the basis of ITRU algorithm, over other algorithms in terms of the processing time. It can also be observable here that 3DES and other algorithms has low performance in terms of power consumption and throughput when compared with DES algorithm. It requires always more time than DES because of its triple phase encryption characteristics of it.

Input size in(Kbytes)	AES	3DES	DES	BLOW FISH	NTRU
yh49	63	54	50	38	28
59	58	51	42	26	16
100	60	57	57	52	32
247	176	77	72	66	55
321	149	87	74	92	79
694	142	146	120	89	92
899	171	171	152	102	60
963	164	177	157	80	50
5345.28	655	835	782	149	115
7310.336	882	1101	953	140	103
Average Time	242	275	246	83.3	58.2
Throughput (Mega-bytes per sec)	6.174	6.455	6.365	18.892	28.77

Table No. 2: Comparative Evaluation of NTRU and ITRU

Encryption Algorithms

The table above describes the comparison of the differences between the NTRU algorithm versus the algorithm of this study, the ITRU algorithm. It is observed here that the success rate of the ITRU algorithm his higher than that of the NTRU algorithm, and that the failure rate of ITRU is lower. It is also noted that the parameters of the NTRU algorithm is of the ring of polynomials while the ITRU algorithm is under the ring of integers.

FACTORS	NTRU	ITRU
Parameters	Ring of Polynomials	Ring of Integer
Message Encryption	Maximum Number of Repeating Patterns	Minimal Number of Repeating Numbers
Message Decryption	Maximum Number of Repeating Patterns	Minimal Number of Repeating Numbers
Success Rate	Low	High
Failure Rate	High	Low

D. Synthesis

The ITRU Encryption Algorithm is one of the newest encryption schemes to be created in 2017. This encryption scheme is based on the lattice-based encryption scheme, NTRU, and uses some of its elements in its algorithm, but differs in that ITRU focuses on the ring of integers rather than the ring of polynomials. ITRU utilizes Modular Arithmetic for its encryption and decryption processes, and it uses the Modular Multiplicative Inverse, specifically the Extended Euclidean Algorithm, in creating its public and private keys. ITRU focuses on the ASCII encoding scheme, which causes a major flaw since the article "UTF-8 Everywhere" has stated that characters such as Latin and Asian charsets lie outside the limits of the ASCII encoding scheme. Since the algorithm will be applied to Mobile Instant Messaging Client Apps, UTF-8 or UTF-16 are more suitable encoding schemes than ASCII, since the documentation for the Android Charset states that UTF-8 is the most common encoding format for Android characters. The algorithm also creates a structural pattern akin to the structure of the plaintext, and as stated by the article "Cryptanalyst Hints", presence of patterns in cipher text can make the encryption vulnerable to frequency analysis attacks. Mobile Instant Messaging Security focuses heavily on the security of packet transfer from one user to another and is vulnerable to hijacking and impersonation. A proper encryption of the information and messages send by the Instant Messaging App can increase the security and prevent it from threats.

Chapter III

DESIGN AND METHODOLOGY

This chapter presents the design and methods used to test the proposed enhancement in the existing ITRU encryption algorithm. Manual and automated simulations is also presented here for easy understanding of the existing and proposed algorithm.

A. Existing Algorithm

1. ITRU Parameters

Parameter	Description
p’	Small modulus
q’	Large Modulus
f’	Private integer for private key generation
g’	Private random integer for public key generation
r’	Private random integer for cipher-text generation
m’	Decimal representation of the message
Kpr	Private key pair (f’, Fp’)

Kpb	Public key parameter h’
a’	Intermediate parameter
C’	Decrypted message

2. ITRU Parameter Generation

1. Initialize p’ = 1000
2. Randomly select an odd integer. Assign value to f’.
3. Randomly select two integers. Assign values to g’ and r’. **Problem No. 1**
4. Convert m’ into its decimal representation based on the ASCII table values.
5. Set q’ = prime integer > (p’ × r’ × g’ + f’ × m’).

3. Key Creation **Problem No. 2**

1. Compute modular multiplicative inverse $Fp' = f'^{-1} \bmod p'$.
2. Compute modular multiplicative inverse $Fq' = f'^{-1} \bmod q'$.
3. Sender obtains private key pair Kpr = (f’, Fp’).
4. Compute for public key $Kph = h' = (p' \times Fq' \times g') \bmod q'$.

4. Encryption **Problem No. 3**

1. Encrypt message m’ using $e' = ((r' \times h') + m') \bmod q'$.
2. Cipher-text e’ is sent to the intended recipient.

5. Decryption

1. Compute $a' = (f' \times e') \bmod q'$.
2. Recover message using $C' = (Fp' \times a') \bmod p'$.
3. Convert C’ back from decimal back to ASCII encoding character.

B. Enhanced Algorithm

1. ITRU Parameters

Parameter	Description
p	Small modulus
Pb	Upper boundary of p randomization
q	Large Modulus
f	Random odd integer for private key generation
g	Random integer for public key generation
r	Random integer for cipher-text generation
m	Decimal representation of the message
m’	Greatest decimal value in m’ + 1
Kpr	Private key pair (f, Fp)
Kpb	Public key parameter
Kiv	Initializing vector
iv	Length of initializing vector
b	Blocks of message m

Eb	Encrypted blocks of b using ((r x h) + b) mod q
Cb	Encrypted blocks of Eb using XOR Operation
e	Combined blocks of Cb
Cb’	Blocks of cipher text e
Eb’	Decrypted blocks of Cb' using XOR Operation
b’	Decrypted blocks of Eb' using (Fp x ((f x Eb') mod q)) mod p
C’	Combined blocks of b'

2. ITRU Parameter Generation

Solution No. 1

1. Convert string into its byte array encoding format. Assign byte array to m.

2. Select the greatest decimal value in m + 1 and assign the decimal to m'.
3. Randomly select a prime integer and assign to p where m < p < Pb .

Solution No. 2

4. Randomly select an odd integer, and assign the integer to f.
5. Randomly select two integers and assign the integers to g and r respectively.

6. Select a prime integer greater than (p x r x g + f x m), and assign the integer to q.

3. Key Creation

1. Compute modular multiplicative inverse $Fp = f^{-1} \bmod p$.
2. Compute modular multiplicative inverse $Fq = f^{-1} \bmod q$.
3. Sender obtains private key pair Kpr = (f, Fp).
4. Compute for public key $Kph = (p \times Fq \times g) \bmod q$.
5. Randomly select an integer between 1 and the length of m, and assign the integer to iv.
6. Generate initializing vector key Kiv with length iv, where each value of Kiv is a random integer between 1 and 255.

4. Encryption

Solution No. 3

1. Divide m into blocks b such that each block has a length equal to iv.
2. Encrypt b using Eb = ((r x h) + b) mod q.
3. If Eb is the first block, apply XOR Operation with Kiv, and assign it to Cb.
4. Else apply XOR Operation with the previous Cb and assign to a new Cb.
5. Repeat until the last Eb and combine all Cb to create e.

6. Cipher-text e, Kpb and Kiv are sent to the intended recipient.

5. Decryption

Solution No. 3

1. Divide e into blocks Cb' such that each block has a length equal to iv.
2. If Cb' is the first block, apply XOR Operation with Kiv, and assign it to Eb'.
3. Else apply XOR Operation with the previous Eb', and assign to a new Eb'.
4. Recover the message blocks using b' = (Fp x ((f x Eb') mod q)) mod p and convert to byte encoding.
5. Repeat until the last b' and combine all b' to create C.

6. Convert C from byte encoding back to the original message string format.

C. Methodology

1. Problem 1 VS. Solution 1

Instead of converting m' into its decimal representation based on the ASCII table values, the researchers worked on converting the string into byte array encoding format then assign that byte array to m'.

2. Problem 2 VS. Solution 2

Aside from computing for the modular multiplicative inverse Fp' = f -1 mod p' then, computing also for the modular multiplicative inverse Fq' = f -1 mod q', the researchers set m as the greatest decimal value in m' + 1 and randomly selecting a prime integer and assigning it to p' where m > p' > 700.

3. Problem 3 VS. Solution 3

Aside from encrypting the message m' using e' = ((r' × h') + m') mod q', the researchers also divided e' into blocks b' such that each block's length equals iv'. Next, if b' is the first block, apply XORing operation with Kiv. Assign to Cb'. Then, if not apply XORing operation with the previous Cb' and assign to b'. Repeat this until the last block and combine blocks to form e'. This method had been both used in the Encryption and Decryption phases.

D. Simulations

The simulations presented below our test cases that shows the allowable up to the exact boundary that can be used in Pboundary – p < N:

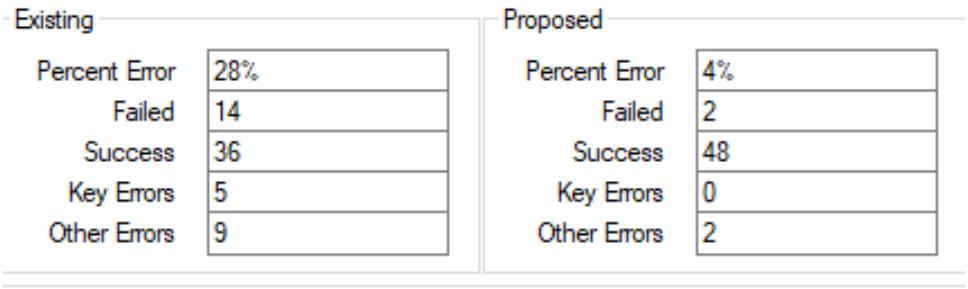
Test Results for Pboundary – p < 1000 (Starting Range)

Test 1:

Input	Output Exist	Status	Output Propose	Status
hi	hi	Success	hi	Success
hi	‘	Fail	hi	Success
hi	` a	Fail	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	` a	Fail	hi	Success
hi	hi	Success	hi	Success
hi	I̋ò	Fail	hi	Success
hi	hi	Success	TU	Fail
hi	` a	Fail	op	Fail
hi	hi	Success	hi	Success
hi	I̋ò	Fail	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	` a	Fail	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	` a	Fail	hi	Success
hi	I̋ò	Fail	hi	Success
hi	I̋ò	Fail	hi	Success
hi	I̋ò	Fail	bc	Fail
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	ɘŋ	Fail	hi	Success
hi	hi	Success	hi	Success
hi	I̋ò	Fail	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success

hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	‘	Fail	hi	Success
hi	‘	Fail	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	̀Ìò	Fail	hi	Success
hi	hi	Success	hi	Success
hi	̀Ìò	Fail	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success

Result:



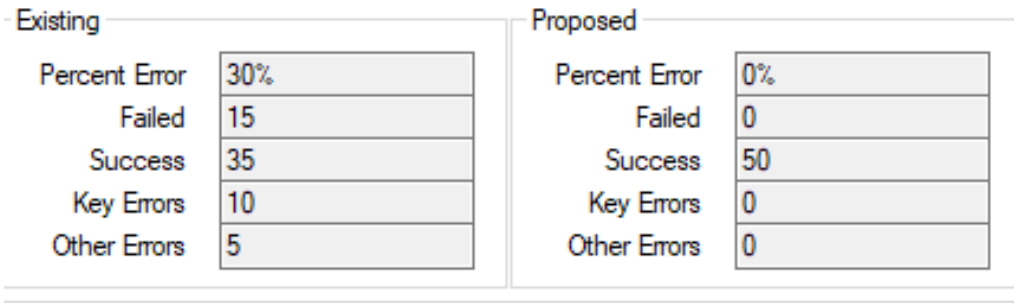
Test Results for Pboundary – p < 700 (Allowable Range)

Test 1:

Input	Output Exist	Status	Output Propose	Status
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	ɘɱ	Fail	hi	Success

hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	‘	Fail	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	l̀ò	Fail	hi	Success
hi	hi	Success	hi	Success
hi	l̀ò	Fail	hi	Success
hi	` a	Fail	hi	Success
hi	hi	Success	hi	Success
hi	àå	Fail	hi	Success
hi	` a	Fail	hi	Success
hi	hi	Success	hi	Success
hi	l̀ò	Fail	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	ɘɱ	Fail	hi	Success
hi	hi	Success	hi	Success
hi	ǐn	Fail	hi	Success
hi	hi	Success	hi	Success
hi	ɘɱ	Fail	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	ɘɱ	Fail	hi	Success
hi	hi	Success	hi	Success
hi	ɤⅱ	Fail	hi	Success
hi	` a	Fail	hi	Success
hi	hi	Success	hi	Success
hi	` a	Fail	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success
hi	hi	Success	hi	Success

Result:



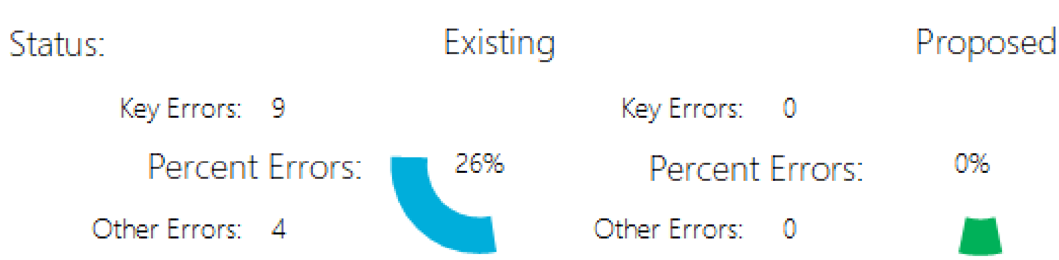
Test Results for Pboundary – p < 630 (Exact Value)

Test 1:

Existing	Status	Params	Proposed	Status	Params
hello world	Success	1000,38,46,11,1749329,120	hello world	Success	439,38,46,11,768727,120,5
hello world	Success	1000,68,39,83,2661961,120	hello world	Success	571,68,39,83,1524253,120,9
hello world	Success	1000,51,44,39,2248681,120	hello world	Success	223,51,44,39,505097,120,7
hello world	Success	1000,14,80,53,1126361,120	hello world	Success	613,14,80,53,692927,120,5
□□□□□/□□□□□	Fail	1000,79,56,35,4428223,120	hello world	Success	271,79,56,35,1203121,120,3
`] d d gɬo g j d \	Fail	1000,99,47,89,4663691,120	hello world	Success	487,99,47,89,2276693,120,7
hello world	Success	1000,34,50,31,1703731,120	hello world	Success	601,34,50,31,1025443,120,8
hello world	Success	1000,37,83,67,3079049,120	hello world	Success	191,37,83,67,594617,120,2
Ìи33õ 6õAЗǴ	Fail	1000,63,35,65,2212817,120	hello world	Success	251,63,35,65,561277,120,8
hello world	Success	1000,66,40,67,2648047,120	hello world	Success	283,66,40,67,755171,120,2
hello world	Success	1000,46,18,19,830293,120	hello world	Success	373,46,18,19,311137,120,7
Ìи33õ 6õAЗǴ	Fail	1000,34,3,5,102607,120	hello world	Success	227,34,3,5,23761,120,3
hello world	Success	1000,42,38,13,1597567,120	hello world	Success	353,42,38,13,564959,120,7
`] d d gɬo g j d \	Fail	1000,92,63,51,5802131,120	hello world	Success	199,92,63,51,1159531,120,10
hello world	Success	1000,54,94,63,5083579,120	hello world	Success	577,54,94,63,2936429,120,6
Ìи33õ 6õAЗǴ	Fail	1000,0,63,15,1801,120	hello world	Success	499,0,63,15,1801,120,1
hello world	Success	1000,59,25,13,1476581,120	hello world	Success	523,59,25,13,772987,120,4
hello world	Success	1000,56,43,21,2410523,120	hello world	Success	139,56,43,21,337261,120,10
□□□□□/□□□□□	Fail	1000,84,41,35,3448243,120	hello world	Success	569,84,41,35,1963873,120,5
hello world	Success	1000,37,35,87,1305449,120	hello world	Success	557,37,35,87,731761,120,4
Ìи33õ 6õAЗǴ	Fail	1000,36,34,15,1225817,120	hello world	Success	587,36,34,15,720289,120,8
hello world	Success	1000,90,58,11,5221327,120	hello world	Success	331,90,58,11,1729141,120,6
hello world	Success	1000,52,76,67,3960049,120	hello world	Success	307,52,76,67,1221373,120,3
hello world	Success	1000,22,86,89,1902737,120	hello world	Success	557,22,86,89,1064533,120,6
hello world	Success	1000,33,50,69,1658291,120	hello world	Success	151,33,50,69,257437,120,3

Existing	Status	Params	Proposed	Status	Params
hello world	Success	1000,41,84,87,3454457,120	hello world	Success	509,41,84,87,1763453,120,1
hello world	Success	1000,25,75,23,1877761,120	hello world	Success	233,25,75,23,439639,120,4
hello world	Success	1000,59,99,17,5843041,120	hello world	Success	281,59,99,17,1643363,120,10
ỵ̸	Fail	1000,62,85,9,5271107,120	hello world	Success	349,62,85,9,1840313,120,2
hello world	Success	1000,11,33,43,368171,120	hello world	Success	449,11,33,43,168151,120,1
hello world	Success	1000,42,5,3,210361,120	hello world	Success	167,42,5,3,35437,120,1
hello world	Success	1000,4,7,43,33161,120	hello world	Success	257,4,7,43,12373,120,9
Ȉǹ33ȫ 6ȫȺ3Ǵ	Fail	1000,56,60,65,3367801,120	hello world	Success	521,56,60,65,1758371,120,3
hello world	Success	1000,50,67,21,3352537,120	hello world	Success	199,50,67,21,669173,120,1
hello world	Success	1000,84,5,47,425641,120	hello world	Success	271,84,5,47,119489,120,2
`] d d gٲo g j d ＼	Fail	1000,64,78,97,5003653,120	hello world	Success	463,64,78,97,2322949,120,6
hello world	Success	1000,60,30,71,1808539,120	hello world	Success	257,60,30,71,471137,120,8
hello world	Success	1000,81,84,73,6812783,120	hello world	Success	433,81,84,73,2954893,120,3
hello world	Success	1000,87,1,1,87121,120	hello world	Success	547,87,1,1,47711,120,10
hello world	Success	1000,56,28,99,1579883,120	hello world	Success	281,56,28,99,452497,120,9
hello world	Success	1000,29,44,9,1277083,120	hello world	Success	307,29,44,9,392827,120,4
hello world	Success	1000,48,34,67,1640053,120	hello world	Success	499,48,34,67,822431,120,7
hello world	Success	1000,90,25,97,2261653,120	hello world	Success	191,90,25,97,441403,120,1
hello world	Success	1000,71,69,81,4908737,120	hello world	Success	179,71,69,81,886643,120,6
hello world	Success	1000,78,3,9,235091,120	hello world	Success	491,78,3,9,115979,120,2
hello world	Success	1000,29,66,31,1917731,120	hello world	Success	353,29,66,31,679363,120,8
hello world	Success	1000,35,76,3,2660377,120	hello world	Success	151,35,76,3,402023,120,3
əȍ’’Ķ̂’Ǵ	Fail	1000,84,75,25,6303001,120	hello world	Success	463,84,75,25,2919907,120,9
Ȉǹ33ȫ 6ȫȺ3Ǵ	Fail	1000,19,15,65,292801,120	hello world	Success	449,19,15,65,135781,120,2
hello world	Success	1000,52,15,11,781321,120	hello world	Success	199,52,15,11,156577,120,8

Result:



Chapter IV

RESULTS AND DISCUSSIONS

A. Presentation of Data

The following figures show the data used in the application, the existing and proposed simulators, and the comparison between the existing and proposed simulations:

1. Comparison between the 1st problem and objective.

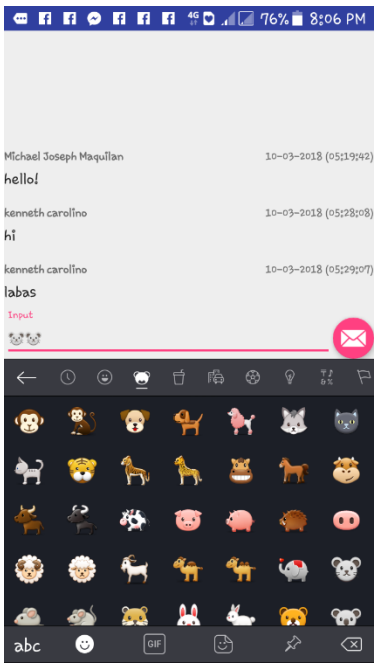


Figure 7: Emoji inputted in the application

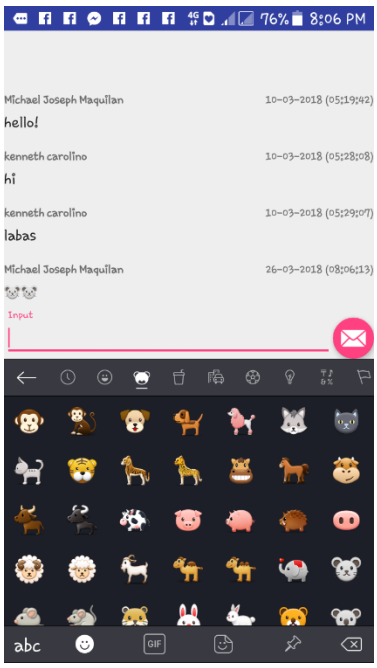


Figure 8: Application accepts emoji

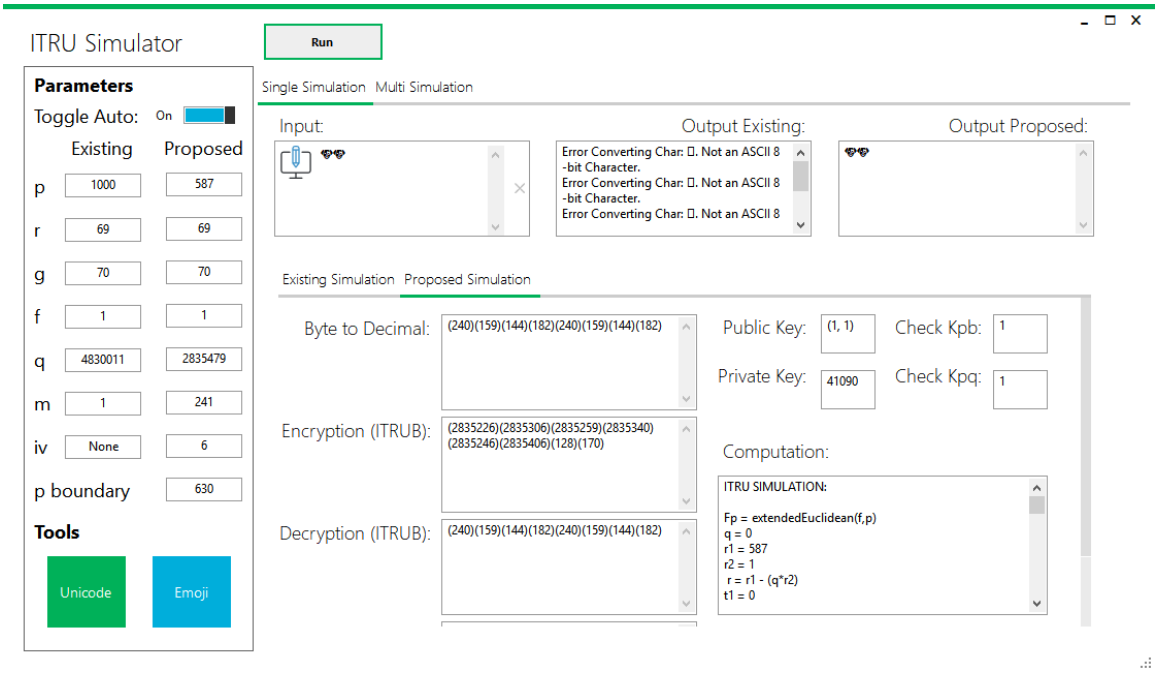


Figure 9: Emoji inputted in the simulator program

For the first problem and objective, the emoji 🐶🐶 is inputted into the simulator and application. The application successfully accepted the emoji, and as presented by the simulator, the proposed algorithm also accepted the emoji string as a valid string. The existing, on the other hand, does not accept the emoji and displays an error.

2. Comparison between the 2nd problem and objective.

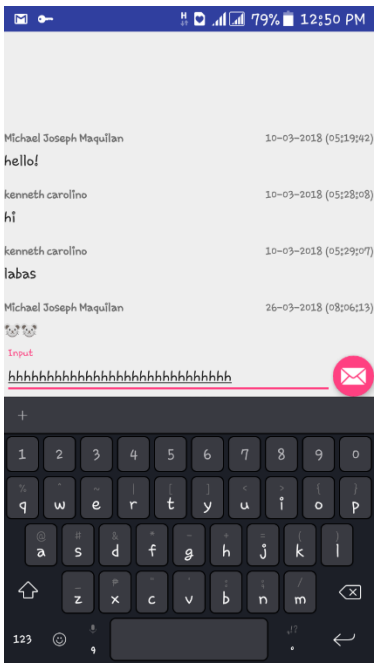


Figure 10: Word inputted in the application

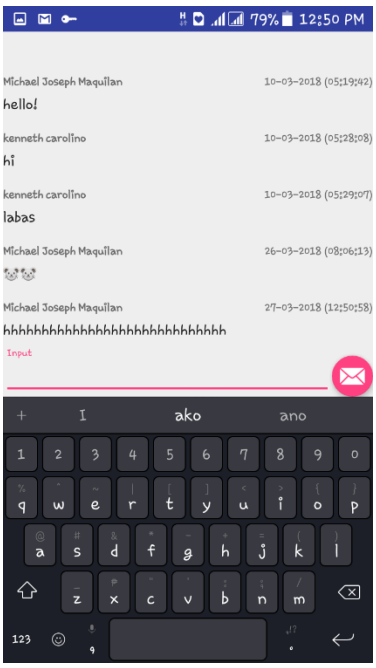


Figure 11: Application accepts word

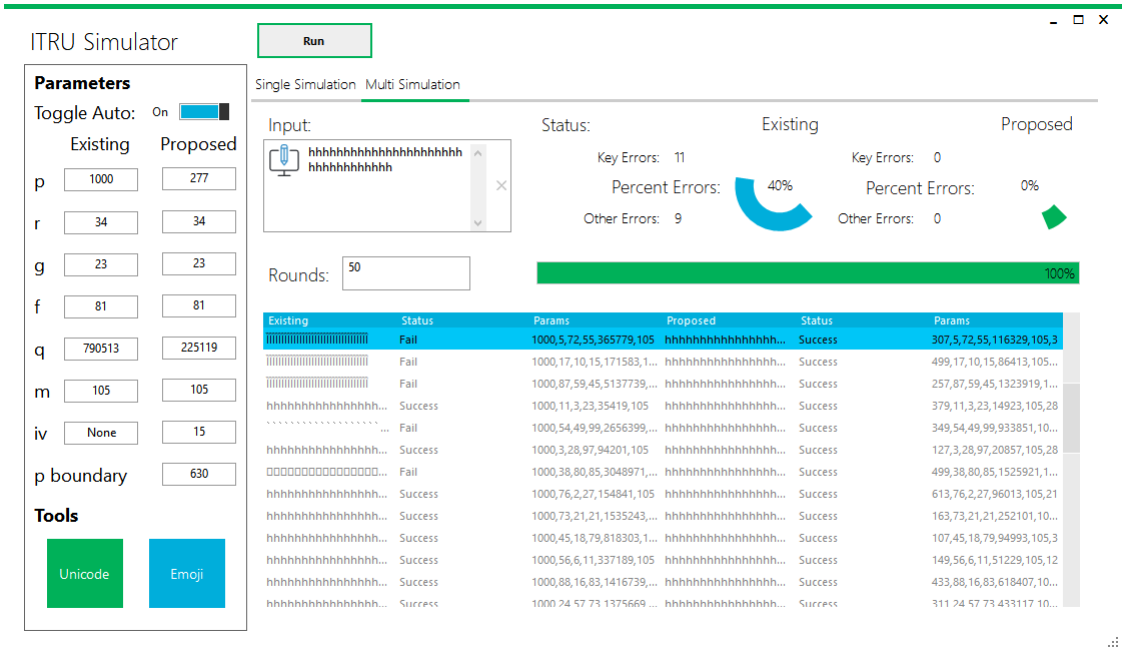


Figure 12: Batch simulation of existing and proposed

For the second problem and objective, a message is inputted into the application and the simulators. The application successfully accepts the message. The simulator was tested through its multiple or batch processing function. The message is inputted, and the cycle was set to 50 rounds. After processing for 50 rounds, the proposed algorithm showed a zero percent error rate, both key-related errors and other errors involved, while the existing algorithm showed a 40% error rate, with 11 key errors and 9 other errors involved.

3. Comparison between the 3rd problem and objective.

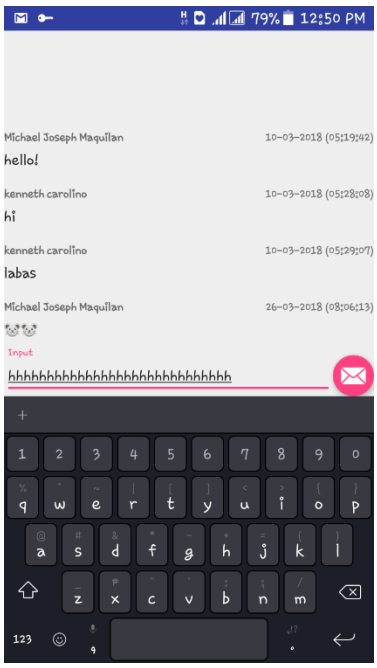


Figure 13: Word inputted in the application

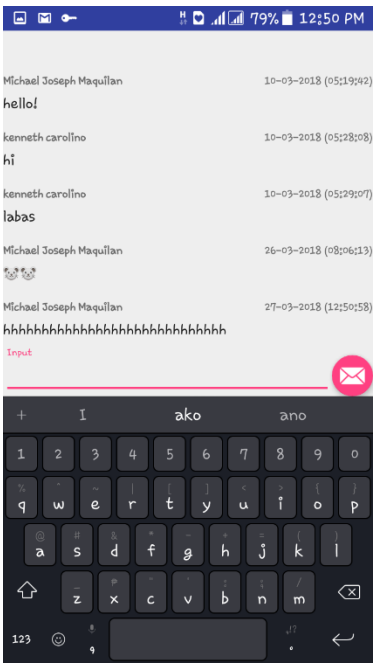


Figure 14: Application accepts word

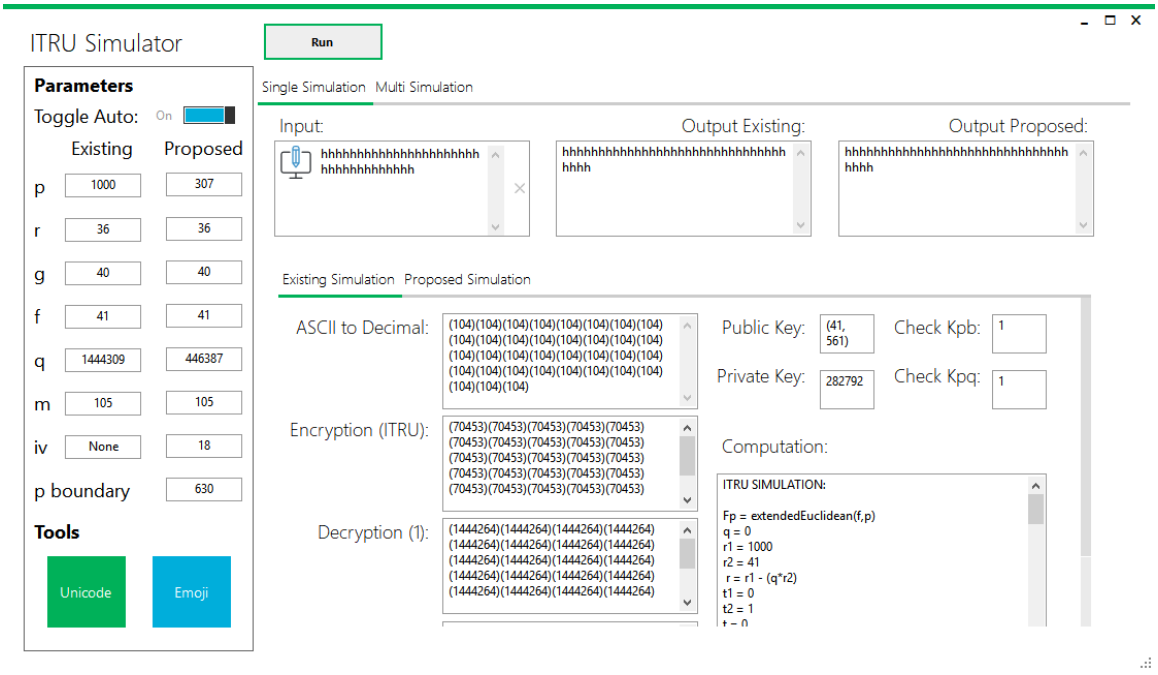


Figure 15: The results of the existing simulation

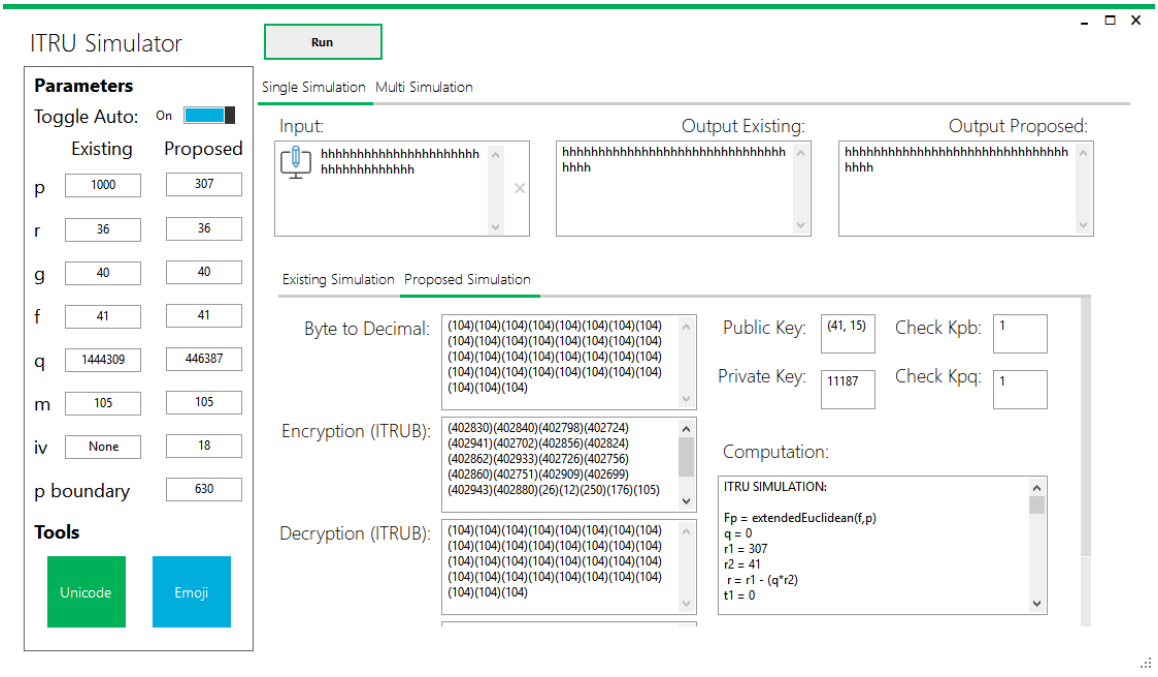


Figure 16: The results of the proposed simulation

For the third and final problem and objective, a message is inputted both in the application and the simulators. The application successfully accepts the message. The simulators presented the encrypted messages or ciphertext after processing the message. For the existing simulator, the ciphertext had a one-to-one correspondence with the plaintext, with each individual ‘h’ has a corresponding ‘70453’ ciphertext. The proposed simulation, on the other hand, had a randomized ciphertext different from the pattern of the plaintext message.

B. Interpretation or Results

Based from the test results, using a Byte equivalence conversion from string to byte made it possible to convert non-ASCII characters to its decimal equivalence as compared to the existing ITRU algorithm that used an ASCII equivalence conversion. The enhanced ITRU algorithm converts each character to its decimal equivalent by dividing the character into its byte representations and stores it into a byte array.

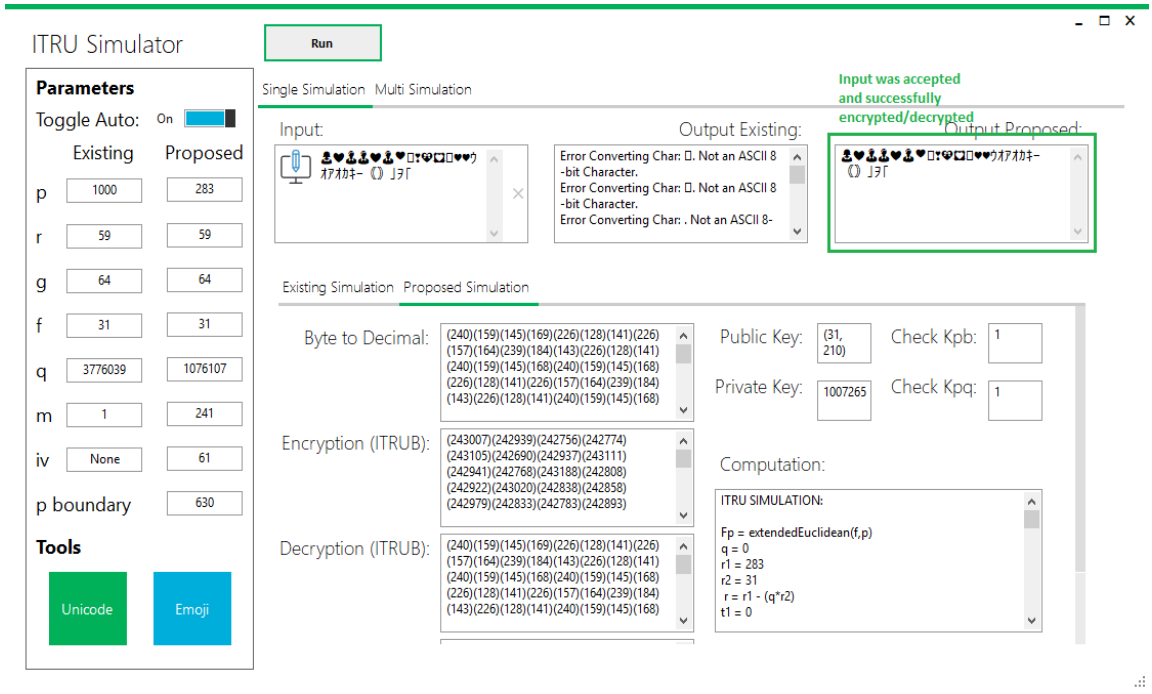


Figure No. 17: Sample Simulation for Objective No. 1

The figure shows that the non-ASCII characters had been successfully converted to decimal using equivalence conversion.

Randomizing p as a prime number greater than m' and less than 1000 decreases the decryption errors caused by the computation of the public key using the Extended Euclidean algorithm. There is a trend in the percentage error of decryptions when the upper boundary of p increases from 700.

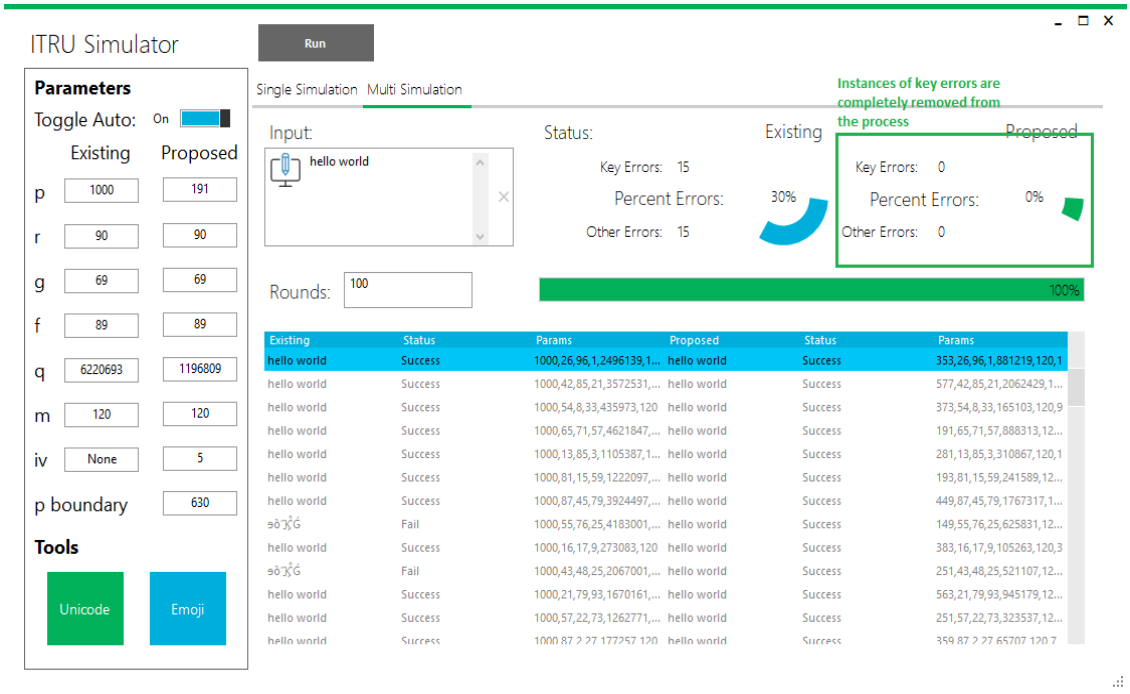


Figure No. 18: Sample Simulation for Objective No. 2

The figure shows that the decryption phase is now successful when p is prime.

The existing algorithm produces a cipher text that has repeating patterns, which can make the algorithm vulnerable. Using Block Cipher Method, the enhanced algorithm filters the cipher text by repeated XOR operations of blocks of the cipher text, reducing the number of patterns that the algorithm produces.

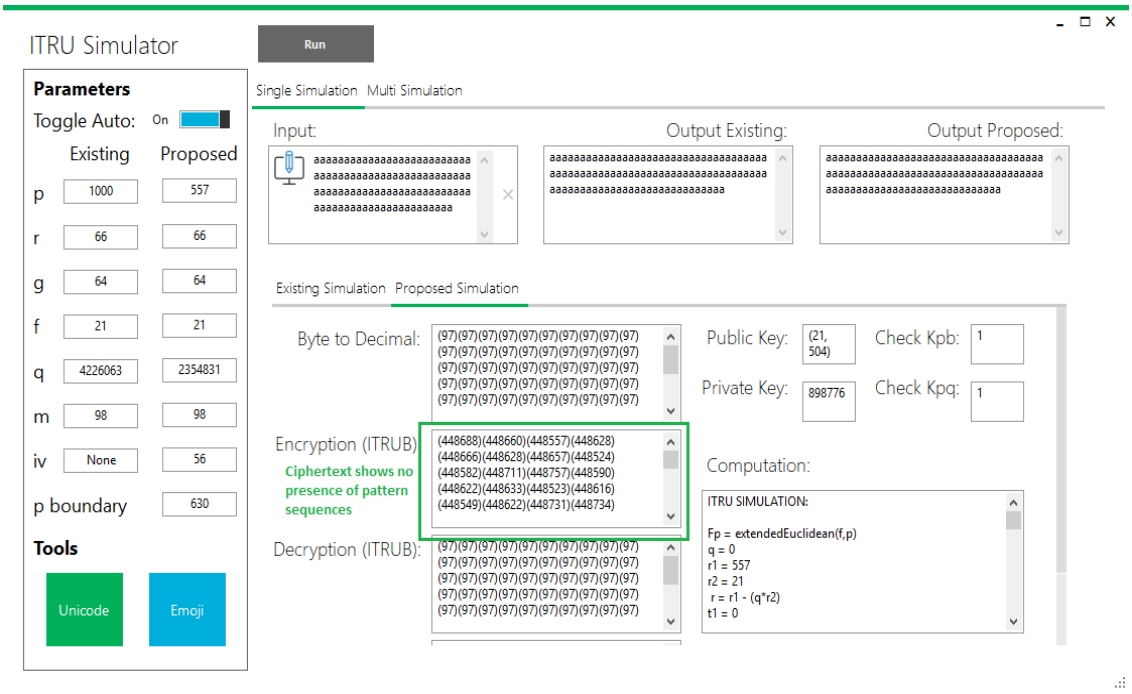


Figure No. 19: Sample Simulation for Objective No. 3

The figure shows about how the Cipher Block Chaining adds diffusion to remove the pattern in the cipher text.

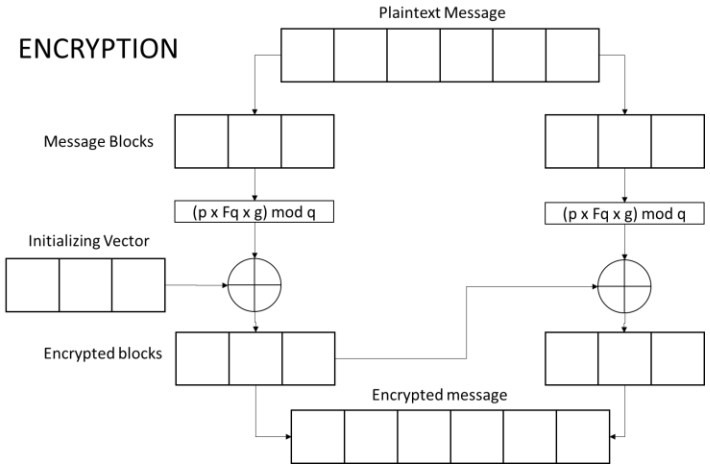


Figure No. 20: Visualization of the Proposed Encryption Scheme

The figure shows about how the Cipher Block Method is applied on the Encryption part of the algorithm.

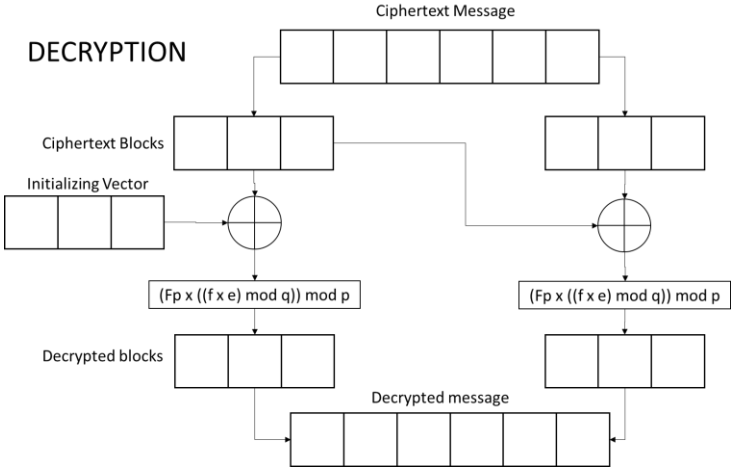


Figure No. 21: Visualization of the Proposed Decryption Scheme

The figure shows about how the Cipher Block Method is applied on the Decryption part of the algorithm.

Chapter V

CONCLUSIONS AND RECOMMENDATIONS

A. Conclusions

Today in the world of cryptosystems today, this research study is useful and beneficial to many. In this research, we have the described the newly introduced encryption algorithm ITRU Algorithm. Not only it is secured with attacks, it also practices optimization. Characters that were not been decrypted before can now be decrypted. Its strength is security among mobile android phones wherein it uses and Integral Method of encryption/decryption. The application is applied particularly in mobile instant messaging apps wherein client users can already feel safe while doing private transaction through chat. The importance of this study is about the security and system performance.

B. Recommendations

The researchers recommend the use of ITRU Encryption Algorithm for current Instant Messaging Applications like WeChat, Kakao Talk, Viber, Messenger etc. and in other platforms such as IOS and Windows. Stand-alone messaging that uses offline services would also be a great application in terms of benefit and comfortability of the user. They can also combine this with other algorithms to improve this innovative cryptosystem on communications though messaging/chat.