Module Code & Module Title

CC5004NI Security in Computing

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2022 -23 Autumn

Student Name: Sharams Kunwar

London Met ID: 21049701

College ID: np01nt4a210112

Assignment Due Date: 9th January 2023

Assignment Submission Date: 9th January 2023

Word Count (Where Required): 5962

Acknowledgement

I would like to express my deepest gratitude to the Islington College and the module teacher Mrs. Suruchi Shrestha and Mr. Akchayat Bikram Joshi for assisting me in accomplishment of the coursework in the given time frame. Likewise, I would also like to convey my sincere appreciation to all the involved helping hands. The guidance from lecturer Mr. Joshi led to accomplishment of the coursework. I would also like to greatly thank Mrs. Shrestha for assisting me day and night to solve my errors and helping me to conduct research on the topic. I would like to express my sincere gratitude to both of them for proper guidance. At last, I would like to thank my friends and family and seniors who helped me clear my confusions and solve problems I faced throughout the completion of the project.

Sincerely Yours,

Sharams Kunwar.

Table of Contents

Table of Figures

List of Table

Abstract

The report discusses about concepts of security and cryptography, its history, its types based on keys. Also, deep research has been done on Stream Cipher and a proper background of it is discussed in the next section of the report along with encryption/decryption examples. Its pros and cons have also been listed.

Likewise, other half of the report covers the development of a cryptosystem based on stream cipher. Research has been done and modified algorithm has been presented addressing the weak aspects of the cipher. Similarly, everything about the algorithm has been elaborated on the latter part of the report along with flowchart representing the working of algorithm.

Finally, the working of algorithm has been verified using multiple varying test cases in the final section of report. The algorithm has been critically analysed addressing its strengths and weaknesses, also discussing its applicability.

1 Introduction to Cryptography

The most fundamental aspects of the society we aspire to live in are Public Safety and Personal Privacy. One may wonder about finding these two principals at odds. To solve the conflict, we need to redirect ourselves to roots of origins of cryptography.

Cryptography means security. Security refers to strategies used to defend organization’s assets. IT security maintains the Confidentiality, Integrity, and Availability (CIA) of sensitive information, also referred to as CIA triad (CISCO, 2022).

Confidentiality refers to preventing sensitive information from unauthorized access attempts. Integrity involves taking steps to ensure the data remains unaltered and maintain the trustworthiness, accuracy, and consistency of data. Likewise, Availability means maintaining infrastructure which hold and display information for authorized personnel to easily access information (Chai, 2022).

CIA is very important as information and data are properly secured ensuring their stability, availability and security which ultimately assures quality information security. Information is collected from multiple sources which must be managed and protected at every step of the information collection which is ensured by CIA triad (DNV, 2022).

Cryptography addresses all three elements of CIA. It can be used to ensure confidentiality of information, protect its integrity and its availability. It is a key tool that can be used by organizations to secure their assets and defend against threats.

From the Roman Empire to Nazi Germany, cryptography has always been taken in use as a strong tool in the struggle for global dominance. In the present days, though the technology has been improvised a lot, it still holds major significance as it is present in everyone’s life, from e-mail to e-commerce. Understanding the change is very crucial to make sense of the debate between the fundamentals, i.e., Public Safety and Personal Privacy.

If one intends to keep any information secret, there are two possible strategies: hide the information’s existence or make the information unintelligible. The term “Cryptography” itself originates from two Greek words: ‘Krypto’ which means hidden and ‘graphene’ which means writing (geeksforgeeks, 2022). Henceforth, it can be concluded that Cryptography is the art and science of keeping information secure from unintended audiences, of encrypting it (LAITS). In other words, cryptography is an ancient art of enciphering and deciphering encoded messages which has taken many forms over the years. It all started with a simple pen-and-paper done simply with letter substitutions which later evolved into machines being built to carry out the encryption. In the modern times, we have even ditched the physical methods and digital encryption has been adopted which is carried out using computers (Morkel, 2004).

1.1 Brief Overview of the Report

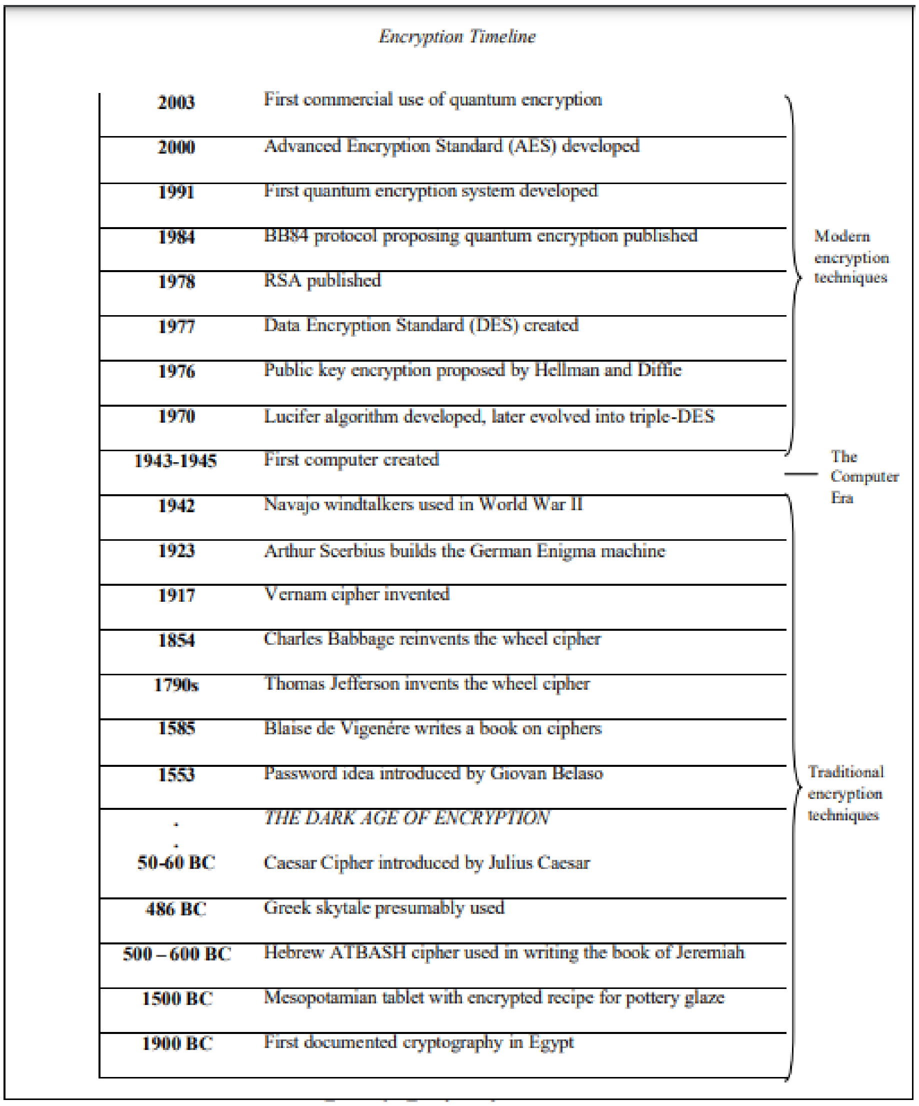
The remainder of this report will focus on briefing on the history of cryptography, the symmetric and asymmetric ciphers. The background of Stream Cipher and the created cryptosystem taking Stream Cipher as a base will follow. In the final section of the report, the created algorithm is tested and is accordingly evaluated with regards to its strengths and weaknesses.

The end goal of the report is to develop a well-tested cryptographic algorithm by choosing a cryptosystem and further modifying it via inclusion of new mathematical and logical operations. To achieve the end goal, the objectives were completed at prior which are as follows:

- To research in detail in the domain of history of cryptosystem, asymmetric and symmetric cipher,
- To develop a cryptosystem by choosing one and including new mathematical and logical operations on it,
- To test the developed algorithm,
- To critically analyse and evaluate the developed algorithm.

1.2 History of Cryptography

Referencing to the encryption timeline below, the creation of the first computer has brought changes to the encryption techniques. Encryption is the process involved in cryptography which basically is the method to encrypt and decrypt messages. Henceforth, the timeline is divided into two phases, i.e., Classical Cryptography and Modern Cryptography based on encryption techniques used, i.e., Traditional Encryption Techniques and Modern Encryption Techniques (Morkel, 2004).



Encryption Timeline

2003	First commercial use of quantum encryption	
2000	Advanced Encryption Standard (AES) developed	
1991	First quantum encryption system developed	
1984	BB84 protocol proposing quantum encryption published	Modern encryption techniques
1978	RSA published	
1977	Data Encryption Standard (DES) created	
1976	Public key encryption proposed by Hellman and Diffie	
1970	Lucifer algorithm developed, later evolved into triple-DES	
1943-1945	First computer created	The Computer Era
1942	Navajo windtalkers used in World War II	
1923	Arthur Scerbius builds the German Enigma machine	
1917	Vernam cipher invented	
1854	Charles Babbage reinvents the wheel cipher	
1790s	Thomas Jefferson invents the wheel cipher	
1585	Blaise de Vigenére writes a book on ciphers	
1553	Password idea introduced by Giovan Belaso	Traditional encryption techniques
.	*THE DARK AGE OF ENCRYPTION*	
.		
50-60 BC	Caesar Cipher introduced by Julius Caesar	
486 BC	Greek skytale presumably used	
500 – 600 BC	Hebrew ATBASH cipher used in writing the book of Jeremiah	
1500 BC	Mesopotamian tablet with encrypted recipe for pottery glaze	
1900 BC	First documented cryptography in Egypt	

Figure 1 Timeline of Encryption Events (Morkel, 2004)

1.2.1 Classical Cryptography (Traditional Encryption Techniques)

The roots of cryptography dates to Roman and Egyptian civilizations. The earliest known use of cryptography can be dated back to 1900 BCE (geeksforgeeks, 2022). They used Hieroglyphs as a secret form of communication between fellow Egyptians.

Then, in 60 B.C., Julius Caesar invented one of the earliest, simplest, and well-known substitution cipher called Caesar Cipher. In the Caesar Cipher the characters were shifted by three places. It was simple and seemed very effective at that time (THALES, 2022).

The next development in the encryption techniques timeline occurred on 1553 A.D after a long dark age. Giovan Battista Bellaso created a first cipher to use a proper encryption key. It was a great steppingstone in the development of encryption techniques (Morkel, 2004).

The next milestone would be Charles Wheatstone's Playfair Cipher which would encrypt pair of letters making it harder to crack than the ones with single letter encryption (Damico, 2009).

Soon, the machinery phase would start involving physical methods of encryption. In 1917, Edward Hebern, invented electro-mechanical machine which had a rotating disc where the key was embedded, and it encoded a substitution table on the typing of a new character. Similarly, in 1918, Arthur Scherbius, invented the Enigma machine which comprised of many rotors compared to one in the Hebern's machine. It was heavily used by German military to send messages (Damico, 2009).

Then, between 1943-1945, the first computer was built. Also in 1945, Claude E. Shannon of Bell Labs published an article named "A mathematical theory of cryptography". This marked the end of physical encryption and gave rise to the modern cryptography which involved computer-based or digital encryption or can also be called modern encryption techniques (Damico, 2009).

1.2.2 Modern Cryptography (Modern Encryption Techniques)

The invention of the first computer brought a massive update to previously existing encryption techniques. The techniques which were thought to be difficult to crack were absolutely crushed by computers as decrypting them were a matter of few seconds (Morkel, 2004).

Modern encryption techniques focus on binary bits and no longer concern the written alphabet. The modern techniques use standardized algorithm which solved impracticality of the traditional techniques (Morkel, 2004).

The working of the standard algorithm would be publicly announced, and the secrecy of the message would rely on another factor called cryptographic key which would be used while encrypting or decrypting the message and it would be impossible for message to be deciphered without it (Morkel, 2004).

Depending on the secrecy of the key, the two types of cyphers have been derived, symmetric and asymmetric ciphers. The following is a short description of each.

1.2.2.1 Asymmetric Cipher

Asymmetric cipher involves the use of two types of keys to encrypt a plain text, public key, and private key. A public key is freely available to receive messages from anyone. A private key is kept secret. Message encrypted with a private key can be decrypted using public key and vice versa. A public key can be extracted from digital certificates to identify the holders. Asymmetric cipher boosts security of information transmitted during communication. It is widely used over the internet. RSA, DSA, PKCs are examples of it (SSL2BUY, 2022).

1.2.2.2 Symmetric Cipher

Symmetric cipher involves a secret key to cipher and decipher information. Henceforth, it is also called as secret-key information. It requires the sender and receiver to have the same secret key which is required for the encryption and decryption of the message. The key is blended with a plain text of message to obtain the cipher text. Then the cipher text is again decrypted using the same key (Abdallah). Blowfish, AES, RC4, DES, etc. are examples of it. Meanwhile, AES-128,192 and 256 are widely used symmetric ciphers (SSL2BUY, 2022).

Present modern cryptosystems are classified into three categories, block ciphers, stream cipher and hybrid ciphers (Nikita Arora, 2014). In the report, we will next emphasize on background of Stream Cipher, its development and analyse its weaknesses and strengths.

2 Background of Stream Cipher

Stream ciphers were first used back in World War II by the German officials. They possessed a typewriter-looking device called Enigma Machine with the help of which the German officials would send directions to their troops. It took years for the code to be cracked. The war ended, and even after tons of development in the field of cryptography, Stream ciphers never really went out of style (okta, 2022).

Stream cipher are the symmetric cipher where the plain text digits are combined with pseudorandom keystream. Each of the plain text is encrypted at once with the corresponding digit of keystream (Nikita Arora, 2014).

It can encrypt plaintext messages of variable length. Modern Stream cipher operate much like the Vernam’s original cipher or also called one-time pads which led to perfect secrecy as the ciphertext gives no information about the plaintext in it (Christensen).

2.1 Working of a Stream Cipher

Stream cipher encrypts one byte of data at a time compared to block cipher which encrypts 128 bits of data at once. Steam Cipher uses a keystream for encryption. It being a symmetric encryption, the recipients and the sender must use the same key to encrypt and decrypt the data (Malviya, 2021).

Stream-Cipher operates by using a pseudorandom number generator (PRNG) to generate a keystream, which is then combined with the plaintext using the exclusive-OR operation, producing the ciphertext. Then, the same PRNG key is used to generate the keystream to get the plaintext, after combining the keystream and ciphertext using exclusive-OR operation (Christensen).

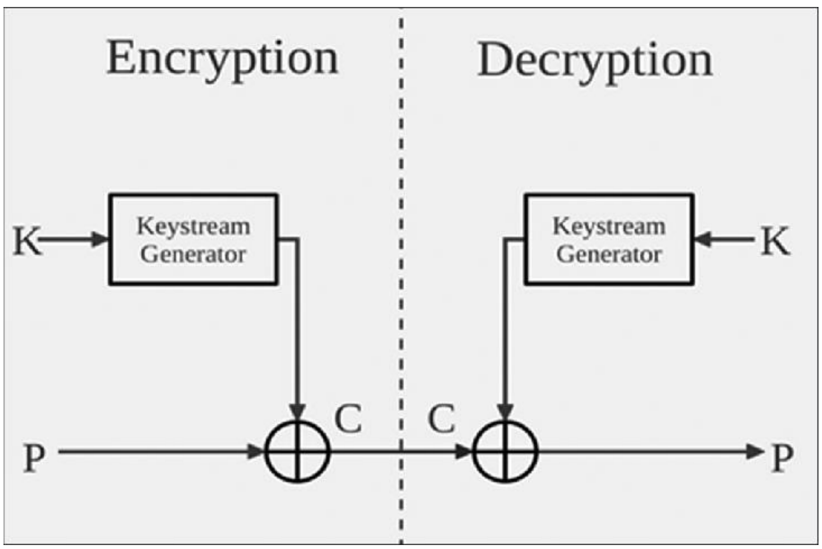


Figure 2 Working Diagram of Stream Cipher (Almufti, 2017)

2.1.1 Keystream

A keystream is generated by supplying a key to pseudorandom bit generator. The output keystream is used in encryption and decryption of the algorithm (Malviya, 2021).

2.1.2 Steps involved in encryption

1. Plaintext and Keystream are essential at prior.
2. Ciphertext is produced using bit-by-bit XOR operation of plaintext and keystream (Malviya, 2021).

Example:

Plaintext: 11011100

Keystream: 11110011

Ciphertext: 00101111

2.1.3 Steps involved in decryption

1. Ciphertext and the keystream used in encryption are used in decryption.
2. Ciphertext is XORed with keystream bit-by-bit to obtain the plaintext (Malviya, 2021).

Example:

Ciphertext: 00101111

Keystream: 11110011

Plaintext: 11011100

2.1.4 Pros and Cons of the Stream Cipher

The stream cipher encrypts data one bit at a time because of which it holds several advantages over other ciphers. Few of the advantages are as follows:

- Because it encrypts data one bit or a byte at a time, large amounts of data can be encrypted quickly without the need of computational resources.
- Similarly, it is easier for it to be implemented in software or hardware due to its quickness and inexpensiveness.
- Stream cipher is pretty-quick as well. It operates in real time as the data gets encrypted and decrypted as it is transmitted and received. The example of this would be encryption during any form of communication over the internet including calls, audio/video streams and many more.
- Only small amount of memory is required to store the key. Henceforth, it can be easily implemented in the hardware (Al-Rasedy & Al-swidi, 2018).

Despite the advantages, stream cipher has its cons as well. It has a price to pay for its simplicity and quickness. Few of the cons are listed below:

- If the key is not complex enough, then the stream cipher is vulnerable to plaintext and replay attacks.
- It may be susceptible to Brute-force attacks if key is short.
- It may be susceptible to dictionary attacks if key is often reused.
- Attackers can analyse patterns and timing of encryption/decryption process to determine the key or plaintext.
- Key may be easy for attacker to guess or predict if it is a word or a phrase.
- In case of large messages, stream cipher is considerably slower than block cipher.
- As only a symmetric key is used in encryption and decryption, if the key is lost, the message can no longer be decrypted.
- Integrity and authenticity are not insured in stream ciphers (Al-Rasedy & Al-swidi, 2018).

3 Development

The algorithm developed as a part of this report has been derived from stream cipher. It is a symmetric-key cipher as well.

Stream-Cipher operates by using a pseudorandom number generator (PRNG) to generate a keystream, which is then combined with the plaintext using the exclusive-OR operation, producing the ciphertext. Then, the same PRNG key is used to generate the keystream to get the plaintext, after combining the keystream and ciphertext using exclusive-OR operation (Christensen).

In the developed algorithm, keystream has been generated using Linear-Feedback Shift Register or LFSR. Multiple taps have been added to plaintext in binary form to operate LFSR in it. Changes have been made in logical part, mostly. Before LFSR operation, the bits in plaintext are substituted. The keystream is finalized after 3 LFSRs and is finalized depending upon its Least Significant Bit (LSB). This makes it immune to different form of exhaustive attacks as well. After that, the encryption and decryption method are basically the same which is conducted using XOR between ciphertext and plaintext or keystream to either encrypt or decrypt.

A new encryption algorithm has been generated addressing the weaknesses of Stream Cipher. The next part of the report elaborates on the background of modifications in algorithm and explanation of its working mechanism, and it is then critically analysed as well.

3.1 Keystream generation using LFSR

LFSR which also stands for Linear-Feedback Shift Register is taken as Pseudo-Random Generators in my algorithm. LFSR manipulates a number by making shifts in position and adding up a new bit in the created space during shifting.

LFSR generates the same number sequence from an initial state which is often referred to as initial seed. The feedback works by initializing bits in certain position as taps and combining them to create a new bit using XOR operation on selected taps which is then inserted into the gap created during shifting.

Example:

4-bit LFSR with taps at 1st and 4th bit:

Position	1	2	3	4
Pre-LFSR	1	0	0	1
Post-LFSR	0	1	0	0

Table 1 LFSR table

Here, 1001 is shifted right and the void is left at 1st bit position which is filled by XOR between 1st bit: 1 and 4th bit: 1, which returns value 0. This gives us the output of LFSR-1 as 0100 and continues up to 15 times and gives 1001 again. The bit, which is shifted out which, in above case, is 4th bit, i.e., 1 is also called Least Significant Bit or LSB.

The unique numbers generated from LFSR can be calculated by formula

: (2^m) – 1, where m stands for no of bits.

In this algorithm we are working with 128-bit seed with the taps of bits at 127, 120, 104, 88, 72, 54, 40, 24, 8. Using 128-bit seed helps to obtain a maximum cycle, meaning it will generate 340,282,366,920,938,463,463,374,607,431,768,211,455 unique numbers without any repetitions. Likewise, before using LFSR on the plaintext, the bits next to taps are also interchanged, i.e., if the 128th bit is 0, then it is changed to 1 and vice-versa. This ensures extra protection.

Example:

If we were to encrypt, acquaintanceship. It is first converted into binary codes.

Character	Binary Code	Character	Binary Code	Character	Binary Code	Character	Binary Code	Character	Binary Code
A	01000001	Q	01010001	g	01100111	w	01110111	-	00101101
B	01000010	R	01010010	h	01101000	x	01111000	.	00101110
C	01000011	S	01010011	i	01101001	y	01111001	/	00101111
D	01000100	T	01010100	j	01101010	z	01111010	0	00110000
E	01000101	U	01010101	k	01101011	!	00100001	1	00110001
F	01000110	V	01010110	l	01101100	"	00100010	2	00110010
G	01000111	W	01010111	m	01101101	#	00100011	3	00110011
H	01001000	X	01011000	n	01101110	$	00100100	4	00110100
I	01001001	Y	01011001	o	01101111	%	00100101	5	00110101
J	01001010	Z	01011010	p	01110000	&	00100110	6	00110110
K	01001011	a	01100001	q	01110001	'	00100111	7	00110111
L	01001100	b	01100010	r	01110010	(	00101000	8	00111000
M	01001101	c	01100011	s	01110011	)	00101001	9	00111001
N	01001110	d	01100100	t	01110100	*	00101010	?	00111111
O	01001111	e	01100101	u	01110101	+	00101011	@	01000000
P	01010000	f	01100110	v	01110110	,	00101100	_	01011111

Figure 3 Binary code Conversion sheet (Zych, 2015)

Plaintext: A C Q U A I N T A N C E S H I P

Using the sheet in the above figure each letter is converted into binary codes.

Plaintext: 0100000**1** 01100011 0101000**1** 01010101 0100000**1** 01001001 0100111**0** 01010100 0100000**1** 01001110 0110001**1** 01000101 0101001**1** 01001000 0100100**1** 010100**0**0

Then the first cipher is obtained by substituting the 0s to 1 and vice versa next to taps (in Bold letters).

Ciphertext I: 0100000**1** *1*1100011 0101000**1** *1*1010101 0100000**1** *1*1001001 0100111**0** *1*1010100 0100000**1** *1*1001110 0110001**1** *1*1000101 0101001**1** *1*1001000 0100100**1** *1*10100**0***1*

Hence, the key to generate keystream is obtained via substitution in the form of Ciphertext I. Then, it is LFSR three times to obtain LFSR-1, LFSR-2, and LFSR-3. After that, is LSB of LFSR-3 is 1, it is used as keystream else LFSR-1 and LFSR-2 are XORed and the output is used as keystream.

LFSR-1: 1010000**0** 11110001 1010100**0** 11101010 1010000**0** 11100100 1010011**1** 01101010 0010000**0** 11100111 0011000**1** 11100010 1010100**1** 11100100 0010010**1** 111010**0**0

LFSR-2: 1101000**0** 01111000 1101010**0** 01110101 0101000**0** 01110010 0101001**1** 10110101 0001000**0** 01110011 1001100**0** 11110001 0101010**0** 11110010 0001001**0** 111101**0**0

LFSR-3: 1110100**0** 00111100 0110101**0** 00111010 1010100**0** 00111001 0010100**1** 11011010 1000100**0** 00111001 1100110**0** 01111000 1010101**0** 01111001 0000100**1** 011110**1**0

Here, LSB of LFSR-3 is 0. Henceforth, key is generated by XOR

(⊕) between LFSR-1 and LFSR-2.

Key: LFSR-1 XOR LFSR-2

Keystream:

1010000**0** 11110001 1010100**0** 11101010 1010000**0** 11100100 1010011**1** 01101010 0010000**0** 11100111 0011000**1** 11100010 1010100**1** 11100100 0010010**1** 111010**0**0

⊕

1110100**0** 00111100 0110101**0** 00111010 1010100**0** 00111001 0010100**1** 11011010 1000100**0** 00111001 1100110**0** 01111000 1010101**0** 01111001 0000100**1** 011110**1**0

Keystream:

01001000 11001101 11000010 11010000 00001000 11011101 10001110

10110000 10011000 11011110 11110101 10011010 00000011 10011101

00101100 10010010

Hence, final keystream is generated which then would be used in encryption process.

3.1.1 Algorithm for Keystream Generation

STEP 1: Plaintext is converted into 128-bits with taps at 127, 120, 104, 88, 72, 54, 40, 24, 8.

STEP 2: The converted bits are then modified by interchanging the bits next to taps, i.e., 0 is changed to 1 and vice versa.

STEP 3: Then obtained ciphertext is LFSRed three times.

STEP 4: If the Least Significant Bit of LFSR-3 is 1, then, go to step 6, else go to step 5

STEP 5: LFSR-1 and LFSR-2 are combined using XOR operation to obtain final keystream.

STEP 6: Use LFSR-3 as final keystream.

3.2 Encryption and Decryption processes

The encryption and decryption processes don't involve any kind of modification and is simple and not complex unlike key generation process. The keystream from above processes is taken in use which would bring about modifications to make steam cipher even more secure.

3.2.1 Algorithm for encryption

STEP1: The number next to taps in plaintext are interchanged, i.e., 0 to 1 and vice versa to obtain Ciphertext I.

STEP 2: Ciphertext I and Keystream generated using LFSR are combined using XOR ($\oplus$) operation.

STEP 3: The Ciphertext I gets encrypted and ciphertext II is obtained which would then be used for decryption process.

Example:

Ciphertext I:

0100000**1** *1*1100011 0101000**1** *1*1010101 0100000**1** *1*1001001 0100111**0** *1*1010100 0100000**1** *1*1001110 0110001**1** *1*1000101 0101001**1** *1*1001000 0100100**1** *1*10100**0***1*

$\oplus$

Keystream:

01001000 11001101 11000010 11010000 00001000 11011101 10001110

10110000 10011000 11011110 11110101 10011010 00000011 10011101

00101100 10010010

Ciphertext II:

11110110 11010001 01101100 11111010 10110110 11101011 00111111 10011011 00100110 11101111 01101001 10100000 10101111 10101010 10011010 10111100

Hence, ciphertext is obtained.

3.2.2 Algorithm for decryption

STEP 1: Ciphertext II obtained from encryption and generated keystream are combined using XOR (⊕) operation.

STEP 2: The Ciphertext II gets decrypted, and Ciphertext I is obtained.

STEP 3: The numbers next to taps are interchanged in Ciphertext I to obtain plaintext.

Example:

Ciphertext II:

11110110 11010001 01101100 11111010 10110110 11101011 00111111 10011011 00100110 11101111 01101001 10100000 10101111 10101010 10011010 10111100

⊕

Keystream:

01001000 11001101 11000010 11010000 00001000 11011101 10001110

10110000 10011000 11011110 11110101 10011010 00000011 10011101

00101100 10010010

Ciphertext I:

0100000**1** *1*1100011 0101000**1** *1*1010101 0100000**1** *1*1001001 0100111**0** *1*1010100 0100000**1** *1*1001110 0110001**1** *1*1000101 0101001**1** *1*1001000 0100100**1** *1*10100**0***1*

Here, Ciphertext I is obtained. Then the numbers next to taps are again interchanged to obtain plaintext.

Plaintext: 01000001 01100011 01010001 01010101 01000001 01001001 01001110 01010100 01000001 01001110 01100011 01000101 01010011 01001000 01001001 01010000

Hence, this way the plaintext gets encrypted and decrypted using symmetric key.

I would name the newly developed cryptographic algorithm as "**Crackeat**".

3.3 Crackeat: A more secure Stream Cipher

Crackeat is a cryptographic algorithm developed on stream cipher by modifying the key generation process. The key generation process is the most vulnerable aspect of the stream cipher. So, the changes had to made there to increase its security and protection from attackers. The encryption is easy to break if the key is simple and easily predictable. Hence, layers of modification have been applied to strengthen the cipher while keeping its speed and reliability.

3.3.1 New methodology in Crackeat

Following changes have been made to make Crackeat much secure than Stream Cipher:

- The plaintext is modified by interchanging number next to taps of LFSR from 0 to 1 and vice versa to obtain Ciphertext-I.
- 128-bit LFSR is used in the first cipher to generate keystream to be used in encryption/decryption process.
- 3 LFSRs are generated and Determination of LFSR is based upon LSB of LFSR-3, i.e., if LSB of LFSR-3 is 1 it is used as keystream else, LFSR-1 and LFSR-2 are XORed, and the obtained keystream is used.

3.3.2 Encryption in Crackeat

The generated keystream is unique and non-repetitive which is XORed with the Ciphertext-I and not the plaintext to get Ciphertext-II. It adds a layer of protection as even the keystream gets cracked; the plaintext remains safe.

3.3.3 Decryption in Crackeat

The ciphertext-II is again XORed with keystream to decrypt it and obtain Ciphertext-I. The plaintext is then obtained by reversing the interchanged 0s and 1s in the Ciphertext-I.

3.3.4 Need for modification

The major vulnerability of stream cipher is the keystream used in it as the same key is used in both encryption and decryption. If the key is simple and predictable, it can be easily cracked. Here, the plaintext is interchanged first before generating the keystream to ensure the protection of plaintext even if attackers crack the keystream. Also, the entire operation occurs on first cipher rather than plaintext. Hence, the plaintext remains protected.

128-bit LFSR is used here to generate keystream which makes the keystream very complex and keeps it safe from brute-force/ dictionary attacks as the keystream generates 340,282,366,920,938,463,463,374,607,431,768,211,455 unique numbers without repetitions. Only 3 of the LFSRs are used from them, before one is ultimately selected or obtained after XOR operation of the first two LFSRs depending upon the LSB of LFSR-3.

So, there are multiple layers of protection have been added to the cipher as modification to ensure its protection while maintaining its reliability and speed.

3.4 Flowchart

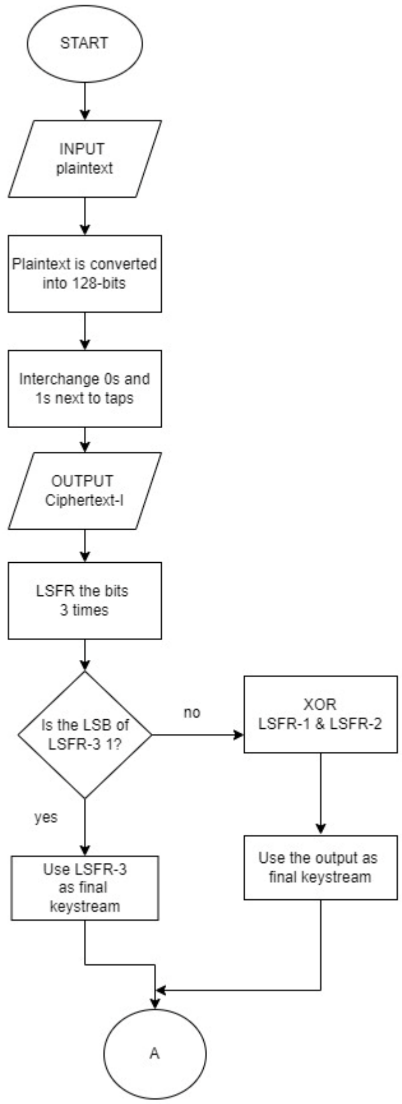


Figure 4 Flowchart for Keystream Generation

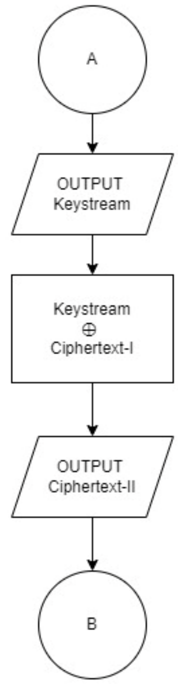


Figure 5 Flowchart for Encryption Process

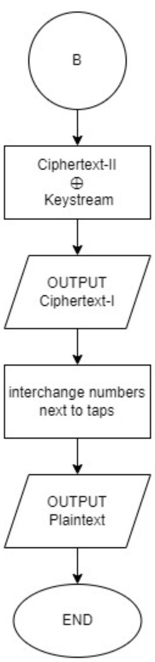


Figure 6 Flowchart for Decryption Process

4 Testing

The developed cryptographic algorithm is tested in this section of the report. All the steps of the algorithm are verified and ensured they work by using multiple different plaintexts for encryption/decryption using the developed cryptographic algorithm: Crackeat.

4.1 Test 1: plaintext with only small letters

Plaintext: t e s t

Plaintext: 0111010**0** 01000101 0111001**1** 01110100 0000000**0** 00000000 0000000**0** 00000000 0000000**0** 00000000 0000000**0** 00000000 0000000**0** 00000000 0000000**0** 000000**0**0

Ciphertext-I: 0111010**0** 11000101 0111001**1** 11110100 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 100000**0**1

LFSR-1: 0011101**0** 01100010 1011100**1** 11111010 0000000**0** 01000000 0000000**0** 01000000 0000000**0** 01000000 0000000**0** 01000000 0000000**0** 01000000 0000000**0** 010000**0**0

LFSR-2: 0001110**1** 00110001 0101110**0** 11111101 0000000**0** 00100000 0000000**0** 00100000 0000000**0** 00100000 0000000**0** 00100000 0000000**0** 00100000 0000000**0** 001000**0**0

LFSR-3: 00001110 10011000 10101110 01111110 10000000 00010000 00000000 00010000 00000000 00010000 00000000 00010000 00000000 00010000 00000000 0001000**0**

Here, LSB of LFSR-3 is 0. So, keystream is generated via XOR of LFSR-1 and LFSR-2.

LFSR-1: 0011101**0** 01100010 1011100**1** 11111010 0000000**0** 01000000 0000000**0** 01000000 0000000**0** 01000000 0000000**0** 01000000 0000000**0** 01000000 0000000**0** 010000**0**0

⊕

LFSR-2: 0001110**1** 00110001 0101110**0** 11111101 0000000**0** 00100000 0000000**0** 00100000 0000000**0** 00100000 0000000**0** 00100000 0000000**0** 00100000 0000000**0** 001000**0**0

Keystream:

11011000 10101100 00011010 11111000 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111

Encryption:

Ciphertext-I:

0111010**0** 11000101 0111001**1** 11110100 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 100000**0**1

⊕

Keystream:

11011000 10101100 00011010 11111000 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111

Ciphertext-II:

01010011 10010110 10010110 11110011 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100001

Decryption:

Ciphertext-II:

01010011 10010110 10010110 11110011 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100001

$\oplus$

Keystream:

11011000 10101100 00011010 11111000 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111

Ciphertext-I:

0111010**0** 11000101 0111001**1** 11110100 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 100000**0**1

Plaintext:

01110100 01000101 01110011 01110100 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Plaintext: t e s t

Hence, test is successful.

4.2 Test 2: plaintext with letters (capital and small) and numbers

Plaintext: t E s T 1 2

Plaintext: 0111010**0** 01000101 0111001**1** 01010100 0011000**1** 00110010 0000000**0** 00000000 0000000**0** 00000000 0000000**0** 00000000 0000000**0** 00000000 0000000**0** 000000**0**0

Ciphertext-I: 0111010**0** 11000101 0111001**1** 11010100 0011000**1** 10110010 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 100000**0**1

LFSR-1: 00111010 01100010 10111001 11101010 00011000 11011001 00000000 01000000 00000000 01000000 00000000 01000000 00000000 01000000 00000000 01000000

LFSR-2: 00011101 00110001 01011100 11110101 00001100 01101100 10000000 00100000 00000000 00100000 00000000 00100000 00000000 00100000 00000000 00100000

LFSR-3: 00001110 10011000 10101110 01111010 10000110 00110110 01000000 00010000 00000000 00010000 00000000 00010000 00000000 00010000 00000000 00010000

Here, LSB of LFSR-3 is 0. So, keystream is generated via XOR of LFSR-1 and LFSR-2.

LFSR-1: 00111010 01100010 10111001 11101010 00011000 11011001 00000000 01000000 00000000 01000000 00000000 01000000 00000000 01000000 00000000 01000000

⊕

LFSR-2: 00011101 00110001 01011100 11110101 00001100 01101100 10000000 00100000 00000000 00100000 00000000 00100000 00000000 00100000 00000000 00100000

Keystream:

11011000 10101100 00011010 11100000 11101011 01001010 01111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111

Encryption:

Ciphertext-I:

0111010**0** 11000101 0111001**1** 11010100 0011000**1** 10110010 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 100000**0**1

⊕

Keystream:

11011000 10101100 00011010 11100000 11101011 01001010 01111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111

Ciphertext-II:

01010011 10010110 10010110 11001011 00100101 00000111 10000000 11100000 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100001

Decryption:

Ciphertext-II:

01010011 10010110 10010110 11001011 00100101 00000111 10000000 11100000 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100001

$\oplus$

Keystream:

11011000 10101100 00011010 11100000 11101011 01001010 01111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111

Ciphertext-I:

0111010**0** 11000101 0111001**1** 11010100 0011000**1** 10110010 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 100000**0**1

Plaintext:

01110100 01000101 01110011 01010100 00110001 00110010 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Plaintext: t E s T 1 2

Hence, test is successful.

4.3 Test 3: plaintext with letters and symbol

Plaintext: t e $ t !

Plaintext: 0111010**0** 01100101 0010010**0** 01110100 0010000**1** 00000000 0000000**0** 00000000 0000000**0** 00000000 0000000**0** 00000000 0000000**0** 00000000 0000000**0** 000000**0**0

Ciphertext-I: 0111010**0** 11100101 0010010**0** 11110100 0010000**1** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 100000**0**1

LSFR-1: 00111010 01110010 10010010 01111010 00010000 11000000 00000000 01000000 00000000 01000000 00000000 01000000 00000000 01000000 00000000 01000000

LSFR-2: 10011101 00111001 01001001 00111101 00001000 01100000 00000000 00100000 00000000 00100000 00000000 00100000 00000000 00100000 00000000 00100000

LSFR-3: 01001110 10011100 10100100 10011110 10000100 00110000 00000000 00010000 00000000 00010000 00000000 00010000 00000000 00010000 00000000 00010000

Here, LSB of LFSR-3 is 0. So, keystream is generated via XOR of LFSR-1 and LFSR-2.

LSFR-1: 00111010 01110010 10010010 01111010 00010000 11000000 00000000 01000000 00000000 01000000 00000000 01000000 00000000 01000000 00000000 01000000

⊕

LSFR-2: 10011101 00111001 01001001 00111101 00001000 01100000 00000000 00100000 00000000 00100000 00000000 00100000 00000000 00100000 00000000 00100000

Keystream:

01011000 10110100 00100100 10111000 11100111 01011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111

Encryption:

Ciphertext-I:

0111010**0** 11100101 0010010**0** 11110100 0010000**1** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 100000**0**1

⊕

Keystream:

01011000 10110100 00100100 10111000 11100111 01011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111

Ciphertext-II:

11010011 10101110 11111111 10110011 00111001 00100000 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100001

Decryption:

Ciphertext-II:

11010011 10101110 11111111 10110011 00111001 00100000 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100001

⊕

Keystream:

01011000 10110100 00100100 10111000 11100111 01011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111

Ciphertext-I:

01110100 11100101 00100100 11110100 00100001 10000000 00000000 10000000 00000000 10000000 00000000 10000000 00000000 10000000 00000000 10000001

Plaintext:

01110100 01100101 00100100 01110100 00100001 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Hence, test is successful.

4.4 Test 4: plaintext with letters, symbols, and numbers

Plaintext: t e $ t 1 , 2

Plaintext: 0111010**0** 01100101 0010010**0** 01110100 0011000**1** 00101100 0011001**0** 00000000 0000000**0** 00000000 0000000**0** 000000000 0000000**0** 00000000 0000000**0** 000000**0**0

Ciphertext-I: 0111010**0** 11100101 0010010**0** 11110100 0011000**1** 10101100 0011001**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 100000**0**1

LFSR-1: 00111010 01110010 10010010 01111010 00011000 11010110 00011001 01000000 00000000 01000000 00000000 01000000 00000000 01000000 00000000 01000000

LFSR-2: 00011101 00111001 01001001 00111101 00001100 01101011 00001100 10100000 00000000 00100000 00000000 00100000 00000000 00100000 00000000 00100000

LFSR-3: 00001110 10011100 10100100 10011110 10000110 00110101 10000110 01010000 00000000 00010000 00000000 00010000 00000000 00010000 00000000 00010000

Here, LSB of LFSR-3 is 0. So, keystream is generated via XOR of LFSR-1 and LFSR-2.

LFSR-1: 00111010 01110010 10010010 01111010 00011000 11010110 00011001 01000000 00000000 01000000 00000000 01000000 00000000 01000000 00000000 01000000

$\oplus$

LFSR-2: 00011101 00111001 01001001 00111101 00001100 01101011 00001100 10100000 00000000 00100000 00000000 00100000 00000000 00100000 00000000 00100000

Keystream:

11011000 10110100 00100100 10111000 11100000 01000010 11101010 00011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111

Encryption:

Ciphertext-I: 01110100 11100101 00100100 11110100 00110001 10101100 00110010 10000000 00000000 10000000 00000000 10000000 00000000 10000000 00000000 10000001

$\oplus$

Keystream: 11011000 10110100 00100100 10111000 11100000 01000010 11101010 00011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111

Ciphertext-II:

01010011 10101110 11111111 10110011 0010110 00010001 00100111 01100000 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100001

Decryption:

Ciphertext-II:

01010011 10101110 11111111 10110011 0010110 00010001 00100111 01100000 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100001

$\oplus$

Keystream:

11011000 10110100 00100100 10111000 11100000 01000010 11101010 00011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111

Ciphertext-I:

01110100 11100101 00100100 11110100 00110001 10101100 00110010 10000000 00000000 10000000 00000000 10000000 00000000 10000000 00000000 10000001

Plaintext:

01110100 01100101 00100100 01110100 00110001 00101100 00110010 00000000 00000000 00000000 00000000 000000000 00000000 00000000 00000000 00000000

4.5 Test 5: plaintext with symbols and numbers

Plaintext: @ , 7 ? 9

Plaintext: 0100000**0** 00101100 0011011**1** 00111111 0011100**1** 00000000 0000000**0** 00000000 0000000**0** 00000000 0000000**0** 00000000 0000000**0** 00000000 0000000**0** 000000**0**0

Ciphertext-I: 0100000**0** 10101100 0011011**1** 10111111 0011100**1** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 10000000 0000000**0** 100000**0**1

LFSR-1: 00100000 01010110 00011011 11011111 10011100 11000000 00000000 01000000 00000000 01000000 00000000 01000000 00000000 01000000 00000000 01000000

LFSR-2: 00010000 00101011 00001101 11101111 11001110 01100000 00000000 00100000 00000000 00100000 00000000 00100000 00000000 00100000 00000000 00100000

LFSR-3: 00001000 00010101 10000110 11110111 11100111 00110000 00000000 00010000 00000000 00010000 00000000 00010000 00000000 00010000 00000000 00010000

Here, LSB of LFSR-3 is 0. So, keystream is generated via XOR of LFSR-1 and LFSR-2.

LFSR-1: 00100000 01010110 00011011 11011111 10011100 11000000 00000000 01000000 00000000 01000000 00000000 01000000 00000000 01000000 00000000 01000000

$\oplus$

LFSR-2: 00010000 00101011 00001101 11101111 11001110 01100000 00000000 00100000 00000000 00100000 00000000 00100000 00000000 00100000 00000000 00100000

Keystream: 11001111 10000010 11101001 11001111 10101101 01011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111

Encryption:

Ciphertext-I: 01000000 10101100 00110111 10111111 00111001 10000000 00000000 10000000 00000000 10000000 00000000 10000000 00000000 10000000 00000000 10000001

$\oplus$

Keystream: 11001111 10000010 11101001 11001111 10101101 01011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111

Ciphertext-II:

01110000 11010001 00100001 10001111 01101011 00100000 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100001

Decryption:

Ciphertext-II:

01110000 11010001 00100001 10001111 01101011 00100000 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100000 00000000 11100001

$\oplus$

Keystream:

11001111 10000010 11101001 11001111 10101101 01011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111 11111111 10011111

Ciphertext-I:

01000000 10101100 00110111 10111111 00111001 10000000 00000000 10000000 00000000 10000000 00000000 10000000 00000000 10000000 00000000 10000001

Plaintext:

01000000 00101100 00110111 00111111 00111001 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Hence, test is successful.

5 Evaluation

The developed cryptographic algorithm, Crackeat is evaluated and critically analysed based upon its strengths and weaknesses in this section of the report. The changes were made to modify the stream cipher with the intent of increasing its security but not all the aspects were covered in the algorithm and few weaknesses remains there. Similarly, a lot of the aspects of it have been strengthen as well. Crackeat certainly has become a better version of the stream cipher but not completely ideal as well. Everything has been analysed here in this section of the report by briefing on both the strengths and weaknesses and its area of application.

5.1 Strengths of Crackeat

The improvements have been done on the stream cipher's security and integrity which accounts to the strengths of Crackeat. They are listed below:

- The generated keystream is of 128-bits and doesn't repeat often. So, it is immune to plaintext and replay attacks to some extent as key is longer in length and of many possibilities.
- 340,282,366,920,938,463,463,374,607,431,768,211,455 unique numbers are generated by the keystream without repetitions which would tire out exhaustive brute force/ dictionary attacks and key would be rarely reused.
- Even if the keystream is cracked, the attackers would have hard time reversing back to the plaintext, as the keystream is chosen from three different keystreams based on one of the keystream's Least Significant Bit.
- The attackers would not be able to get the plaintext even after they get their hands on keystream, as they would only land on Ciphertext-I and not the actual plaintext.
- The keystream generates tons of multiple unique numbers without repetitions and various patterns. So, the attackers wouldn't be able to analyse the patterns of encryption/decryption and crack the algorithm.
- Here the keystream is generated based on plaintext and the key wouldn't be reused.

5.2 Weaknesses of Crackeat

There have been some significant improvements in integrity and security, and everything sounds perfect and flawless. It isn't the case though. There are multiple flaws in the developed algorithm which account to weaknesses of Crackeat. Few of them are listed below:

- LFSR is easy to implement in hardware. But here we are using multiplexed LFSR which reduces the efficiency of encryption.
- If the LFSRs happen to have all 0s, then the encryption wouldn't occur, which is extremely unlikely, and the plaintext will leak through.
- It would require larger memory and processing to generate 128-bit unique numbers every time.
- The message can't be decrypted if the key is lost as, it is a symmetric key encryption.
- Keystream is very long and can't be easily remembered. A slight mistake in handling and inputting of the keystream leads to failure in decryption.

5.3 Application Area of Crackeat

The application area of Crackeat would be no different than the Stream Cipher. It may be used in communication, DVD players as it involves real-time encryption. Other area of its application would be in wireless connections and instant message applications where the size of plaintext is not known. It may be of purpose in military as well and can be used in devices like radio set. Likewise, it can be used in websites as well and organizational task as well.

6 Conclusion

The report is finally concluded after extensive research on concepts of security, CIA, information security and cryptography. Additional research was done on cryptography and its history, its types based on symmetric and asymmetric keys. Then, stream cipher was selected as a base to develop a new algorithm on. Research was done on the cipher and a proper background was written including its advantages and disadvantages providing a proper example. Then, a new algorithm was developed using stream cipher as a base algorithm improving on the weaknesses of the stream cipher. Similarly, the newly developed algorithm was critically analysed addressing its pros and cons.

At the completion of the report, doubts regarding cryptography and types of keys were cleared. The report enlightens on the importance of information security as well. It enlightens on the history of cryptography and stream cipher. It also highlights the pros and cons of stream cipher and ways to improvise the algorithm addressing the weaknesses.

Personally, I enjoyed a lot doing the coursework. It was challenging and totally tested my researching abilities. Collecting data from various sources, compiling them, and including them into the creation of my report was extremely tedious but at the same time very satisfactory as well. The coursework helped me sharpen my research skills and analytical skills as well. I was able to put forward my opinions and learnings in the report, of which, I am very grateful. The guidance I received form module teacher would be the most crucial element which led to successful completion of this coursework.

7 References

Abdallah, A. E. (n.d.). *Introduction to Symmetric and Asymmetric Cryptography.* Retrieved from bcuassets: https://bcuassets.blob.core.windows.net/docs/aacyberessential3cryptography-131067585699078884.pdf

Almufti, S. M. (2017). *ResearchGate.* Retrieved from Stream cipher diagram. : https://www.researchgate.net/figure/Stream-cipher-diagram_fig2_318517979

Al-Rasedy, A. S., & Al-swidi, A. A. (2018). *An advantages and Dis Advantages of Block and Stream Cipher.* Retrieved from becm-iq: http://becm-iq.com/papers/uobj_paper_2018_111013694.pdf

Chai, W. (2022). *confidentiality, integrity and availability (CIA triad).* Retrieved from techtarget: https://www.techtarget.com/whatis/definition/Confidentiality-integrity-and-availability-CIA#:~:text=Confidentiality%2C%20integrity%20and%20availability%2C%20also,with%20the%20Central%20Intelligence%20Agency.

Christensen, C. (n.d.). *Stream Ciphers.* Retrieved from nku: https://www.nku.edu/~christensen/Stream%20ciphers.pdf

CISCO. (2022). *What Is IT Security?* Retrieved from CISCO: https://www.cisco.com/c/en/us/products/security/what-is-it-security.html

Damico, T. M. (2009). A Brief History of Cryptography. *CRYPTOGRAPHY, 1*(11), 1. Retrieved from http://www.inquiriesjournal.com/articles/1698/a-brief-history-of-cryptography

DNV. (2022). *The three-pillar approach to cyber security: Data and information protection.* Retrieved from DNV: https://www.dnv.com/article/the-three-pillar-approach-to-cyber-security-data-and-information-protection-165683

geeksforgeeks. (2022, October 06). Retrieved from History of Cryptography: https://www.geeksforgeeks.org/history-of-cryptography/

LAITS. (n.d.). *Cryptography Defined/Brief History.* Retrieved from LAITS: https://www.laits.utexas.edu/~anorman/BUS.FOR/course.mat/SSim/history.html

Malviya, N. (2021, 11 January). *Understanding stream ciphers in cryptography*. Retrieved from infosecinstitute: https://resources.infosecinstitute.com/topic/stream-ciphers/

Morkel, T. (2004). *ENCRYPTION TECHNIQUES: A TIMELINE APPROACH.* Retrieved from DIGITAL FORENSIC SCIENCE: https://digifors.cs.up.ac.za/issa/2004/Proceedings/Research/062.pdf

Nikita Arora, Y. G. (2014). Block and Stream Cipher Based Cryptographic Algorithms: A Survey. *International Journal of Information and Computation Technology. , 4*(2), 8. Retrieved from https://www.ripublication.com/irph/ijict_spl/ijictv4n2spl_13.pdf

okta. (2022). *Stream Cipher 101: Definition, Usage & Comparisons*. Retrieved from okta: https://www.okta.com/identity-101/stream-cipher/

SSL2BUY. (2022). *Symmetric vs. Asymmetric Encryption – What are differences?* Retrieved from SSL2BUY: https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences#:~:text=Symmetric%20encryption%20uses%20a%20single,and%20decrypt%20messages%20when%20communicating.

THALES. (2022, Nov 21). *A BRIEF HISTORY OF ENCRYPTION (AND CRYPTOGRAPHY)*. Retrieved from THALES: https://www.thalesgroup.com/en/markets/digital-identity-and-security/magazine/brief-history-encryption

Zych, A. (2015). *Write Your Name In Binary Code.* Retrieved from Sciencefriday: https://www.sciencefriday.com/educational-resources/write-your-name-in-binary-code/

8 Bibliography

Anderson, R. (2014). *On Fibonacci Keystream Generators.* Retrieved from cam: https://www.cl.cam.ac.uk/~rja14/Papers/fibonacci.pdf

Chetry, M. K., & Kandaswamy, W. B. (2010). A Note On Self-Shrinking Lagged Fibonacci. *International Journal of Network Security, 11*, 58-60. Retrieved from http://ijns.jalaxy.com.tw/contents/ijns-v11-n1/ijns-2010-v11-n1-p58-60.pdf

Cohen, F. (2007). *New World Encyclopedia*. Retrieved from A short history of cryptography: http://www.all.net/books/ip/Chap2-1.html

Cryptography. (n.d.). Retrieved from New World Encyclopedia: http://www.newworldencyclopedia.org/entry/Cryptography

Dooley, J. F. (2013). *A Brief History of Cryptology and Cryptographic Algorithms.* Springer Cham.

Klein, A. (2013). *Stream Ciphers.*

Menzes, A., Oorschot, P. V., & Vanstone, S. (1996). *Handbook of Applied Cryptography.*

Paul, G., & Maitra, S. (2011). *RC4 Stream Cipher and Its Variants (Discrete Mathematics and Its Applications) .*

Pawlan, Monica. (1998). *Cryptography: the ancient art of secret messages*. Retrieved from Pawlan: http://www.pawlan.com/Monica/crypto/

Rubian. (2008). *Vigenere Cipher*. Retrieved from juliantrubin: http://www.juliantrubin.com/encyclopedia/mathematics/vigenere_cipher.html

SIMMONS, G. J. (1979, August). Symmetric and Asymmetric Encryption. *Computing Surveys, 11*(4), 26. Retrieved from princeton: https://www.princeton.edu/~rblee/ELE572Papers/CSurveys_SymmAsymEncrypt-simmons.pdf

Taylor. (2002). *Number theory 1*. Retrieved from usask: http://math.usask.ca/encryption/lessons/lesson00/page1.html

9 Appendix

Originality report

Originality report

COURSE NAME
CC5004NI - Security in Computing

STUDENT NAME
Sharams Kunwar Networking

FILE NAME
21049701 SHARAMS KUNWAR

REPORT CREATED
9 Jan 2023

Summary

Flagged passages	0	0%
Cited/quoted passages	22	4%

Web matches

up.ac.za	6	0.8%
utexas.edu	2	0.4%
ripublication.com	2	0.4%
infosecinstitute.com	2	0.4%
github.com	2	0.4%
ssl2buy.com	2	0.3%
studypool.com	1	0.3%
techtarget.com	2	0.2%
springer.com	1	0.2%
synopsys.com	1	0.1%
ebin.pub	1	0.1%

1 of 22 passages

Student passage CITED

Confidentiality refers **to** preventing **sensitive information from unauthorized access attempts**. Integrity involves taking steps to ensure the data remains…

Top web match