Module Code & Module Title: CC5009NI

Cyber Security in Computing

Assessment Weightage & Type

40% Individual Coursework

Year and Semester 2024 -25

Autumn Semester

Student Name: LilaRaj Dura

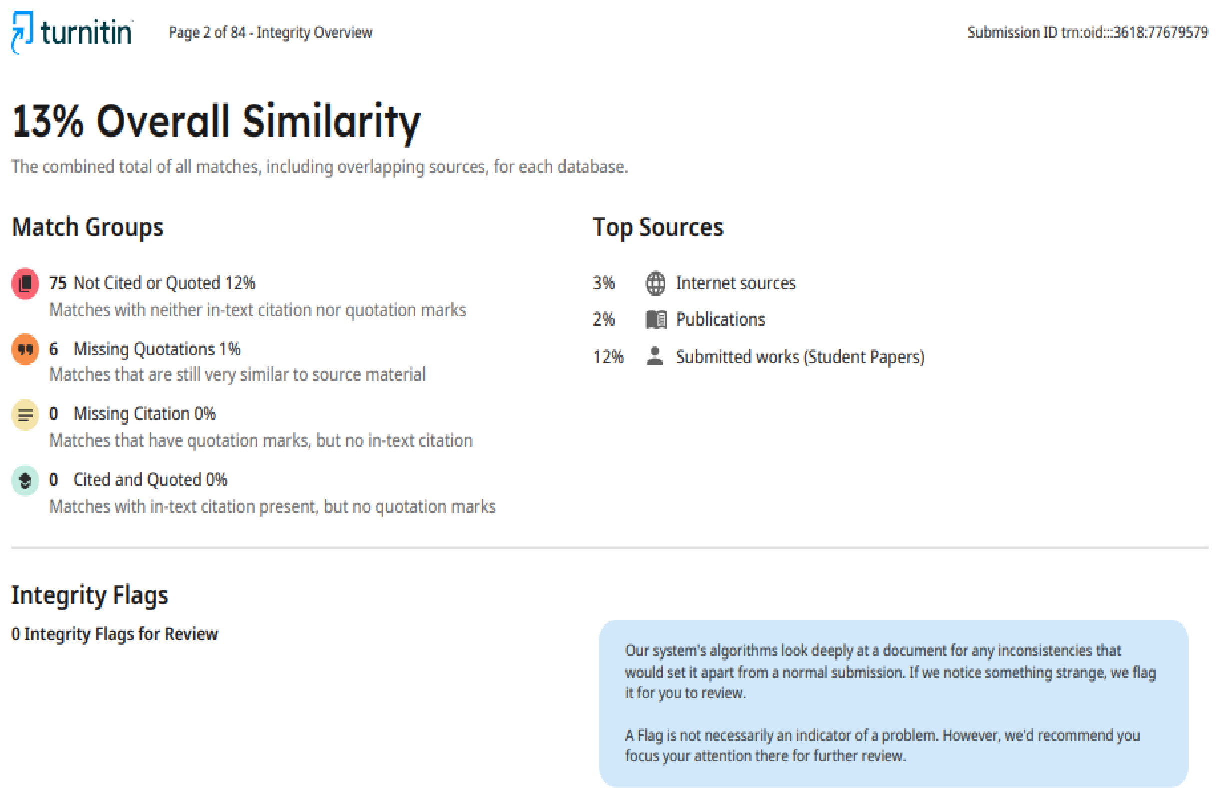
London Met ID:23056178

College ID:NP01NT4S240065

Assignment Due Date: 20/01/2025

Assignment Submission Date: 20/01/2025

Word Count: 9001



turnitin Page 2 of 84 - Integrity Overview Submission ID trn:oid:::3618:77679579

13% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- **75** Not Cited or Quoted 12%
  Matches with neither in-text citation nor quotation marks
- **6** Missing Quotations 1%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

3%	Internet sources
2%	Publications
12%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Table of Contents

Abstract III
Introduction 1
Security 2
CIA Triad 2
Cryptography 3
Ancient cryptography 3
Medieval Cryptography 4
Modern Cryptography 5
Symmetric Encryption 7
Asymmetric Encryption 7
Background 9
1) Caesar cipher 10
2) Row transposition 10
3) Rail fence cipher 11
4) ASCII 12
5) Logical Gate 14
Development 15
Encryption Process 15
Decryption process 25
Testing 30
Test 1 30
Test 2 43
Test 3 51
Test 4 60
Test 5 68
Evaluation 76
Conclusion 77
References 78

Table of Figures

Figure 1: Ancient cryptography 4
Figure 2::Medival Cryptography 5
Figure 3: Modern cryptography 6
Figure 4: Flow chart 9
Figure 5: Rail fence encryption 11
Figure 6: Standard ASCII table 12
Figure 7: Extended ASCII table 13
Figure 8: NOR gate 14

Abstract

Security has always been a critical concern in the field of information technology, encompassing the tools, strategies, and practices designed to safeguard digital and physical assets against threats. At the core of this endeavour lies the CIA triad, which emphasizes three fundamental pillars: Confidentiality, Integrity, and Availability. These principles guide the development of security solutions to ensure that sensitive data remains protected, unaltered, and accessible when needed. Among the various techniques employed in the realm of security, cryptography stands out as one of the most effective and enduring methods.

This report goes into the historical development of cryptography and its contemporary significance in addressing security challenges. A particular focus is placed on the design and implementation of the "LRD Algorithm," a novel encryption method that integrates multiple cryptographic techniques, including the Caesar cipher, Row Transposition cipher, Rail Fence cipher, and logical operations like the NOR gate. The algorithm adopts a layered approach, where the output of one cipher serves as the input for the next, creating a robust encryption process. Additionally, the use of logical operations enhances the complexity and security of the encrypted data. By combining the simplicity of classical ciphers with the sophistication of modern computational techniques, the LRD Algorithm provides a secure and efficient method for protecting sensitive information.

The LRD Algorithm bridges the gap between tradition and contemporary needs. It offers a versatile and reliable solution that addresses the growing demand for robust encryption mechanisms in an increasingly interconnected digital world. This report aims to contribute to the ongoing efforts to enhance information security, fostering a safer and more resilient technological ecosystem.

Introduction

The importance of securing information is increasing rapidly in today's time. From personal communications to sensitive organizational data, the need to protect information from unauthorized access, tampering, and loss has grown exponentially. Security mechanisms are not just a safeguard but a necessity for maintaining trust, privacy, and functionality in modern technology.

At the heart of information security lies the CIA triad Confidentiality, Integrity, and Availability. These three principles provide a comprehensive framework to address the core aspects of security. Confidentiality ensures that sensitive data is accessible only to authorized individuals, integrity guarantees that information remains unaltered, and availability ensures reliable access to resources when needed. Cryptography plays a vital role in achieving these objectives.

Cryptography has a long history, evolving from ancient techniques like substitution and transposition to complex algorithms that secure today's digital communications. It enables the encoding of information in a way that only authorized parties can decipher, ensuring the protection of data both at rest and in transit. However, as technology advances, so do the methods employ by attackers. This necessitates continuous innovation in cryptographic techniques to stay ahead of potential threats.

The "LRD Algorithm" created here provides a new approach to data encryption. By integrating classical ciphers such as Caesar cipher, Rail Fence cipher, and Row Transposition cipher, alongside logical operations like the NOR gate, the algorithm achieves a unique balance between simplicity and security. The layered methodology of the algorithm, where the output of one cipher serves as the input for the next, enhances the robustness of the encryption process. Logical operations further add complexity, making the encrypted data highly resilient to unauthorized decoding attempts.

It explores the historical significance of the cryptographic techniques employed, their role in achieving the goals of the CIA triad, and the practical applications of this encryption method in contemporary security challenges. By using the strengths of traditional methods and modern computational logic, the LRD Algorithm exemplifies how innovative solutions can address the ever-evolving landscape of information security.

Security

Security in IT refers to the practices, tools, and strategies used to protect an organization's digital and physical assets from threats, whether they are accidental or intentional. It focuses on safeguarding devices, data, and services from being accessed, stolen, or exploited by unauthorized individuals, known as threat actors. Security encompasses physical protection, like controlling access and using surveillance, and information security, which includes methods to prevent, detect, and respond to cyberthreats. A strong security approach involves constant updates, testing, and improvement to keep ahead of evolving risks.

CIA Triad

The CIA triad is a core concept in information security, providing a simple yet powerful framework for protecting data and systems. It stands for Confidentiality, Integrity, and Availability three pillars that work together to ensure information stays secure and accessible for those who need it, while keeping it safe from unauthorized access or tampering.

1. Confidentiality

Confidentiality is all about keeping sensitive information private. It ensures that only the right people or systems can access certain data. Think of it like locking your diary or encrypting your messages only someone with the key or password can see the content. Organizations achieve this through techniques like passwords, biometrics, encryption, and strict access controls. For example, when you log in to your bank account, two-factor authentication (like entering a code sent to your phone) ensures that only you can access your money. Confidentiality also involves guarding against tricks like phishing or social engineering, where attackers try to fool people into giving up sensitive information.

2. Integrity

Integrity means keeping data accurate and unaltered. Whether it's a financial report, a medical record, or your online orders, the data should be reliable and reflect the truth. This principle ensures that no one can tamper with or corrupt information either accidentally or on purpose. Techniques like checksums, data validation, and regular backups help maintain data integrity. For instance, when you transfer money between accounts, the bank's systems ensure the exact amount you entered is recorded and reflected in your balance. Integrity is also about spotting changes whether it's a hacked website or a misfiled database entry and restoring the data to its original state.

3. Availability

Availability ensures that information and systems are accessible when needed. It's about making sure systems are up and running, even when there's heavy traffic, technical problems, or attacks like denial-of-service (DoS). For example, imagine trying to withdraw cash from an ATM in the middle of the night it's available because of robust systems that keep it operational 24/7. Organizations use strategies like redundant servers, backups, and disaster recovery plans to maintain availability. They also monitor their systems constantly to quickly address hardware failures, cyberattacks, or unexpected surges in demand. (Fruhlinger, 2024)

Cryptography

Cryptography is a way of securing communication by making data can only be encrypted or decrypted by sender or receiver. Key is a letter or numbers which helps to unencrypted data, encrypted data, and cryptographic algorithms. Encryption transforms plaintext into ciphertext, while decryption reverses the process means from cipher text into plaintext which can be understand by humans. (Eric conrad, 2014)

Cryptography ensures that all the CIA requirements are met. This includes confidentiality which makes sure that secrets remain a secret, integrity includes not getting the data tampered with and authentication involves verifying identities before any data is sent out or given access to.

Ancient cryptography

The origins of cryptography was around on 1900 BCE in ancient civilizations, where the messages need to send secretly due to warfare, diplomacy, and trade. The earliest known form of cryptography comes from Egypt, where non-standard hieroglyphs were used to encode messages on the walls of tombs. This use of secretive symbols was not for general communication but for spiritual, aimed at protecting private information.

Figure 1: Ancient cryptography

(Coinloan, 2023)

By 1500 BCE, the Mesopotamians make cryptography a bit complicated, using clay tablets to encrypt writing. In these tablets are recipes for ceramic glazes, potentially trade secrets. As cryptographic techniques evolved, they became increasingly sophisticated. The earliest known cipher, the scytale, was developed by the Spartans around 650 BCE. This cipher involved writing a message on a strip of parchment wrapped around a wooden box. The text appeared as an decrypt jumble unless the recipient had the same staff to decode it. This was an early example of a transposition cipher, where the order of the characters is rearranged, rather than substituted. (J.SImmons, 2024)

Medieval Cryptography

The credit of rise of cryptography during the medieval period goes to Arab scholars. Al-Kindi, an Arab mathematician in the 9th century, made one of the most significant cryptanalyses by introducing frequency analysis. This technique involves studying the frequency of letters or symbols within a cipher to make educated guesses about the plaintext.

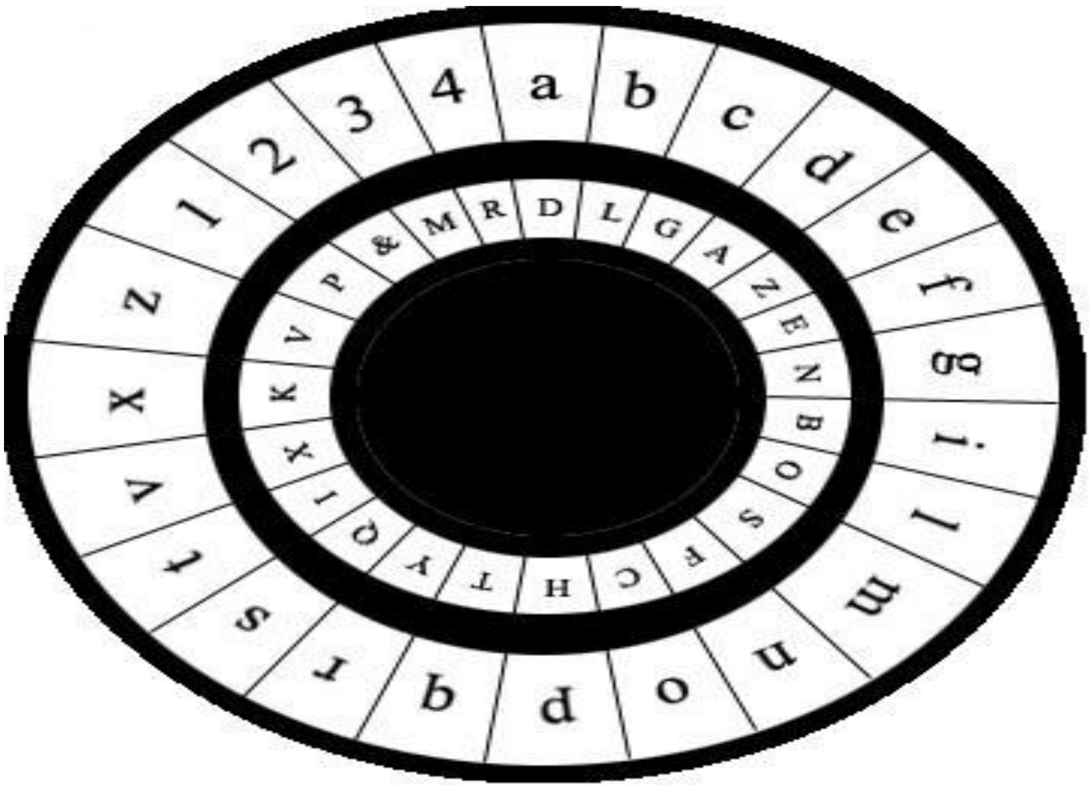


Figure 2::Medieval Cryptography

(servos, 2010)

Further advancements in encryption systems came with the development of polyalphabetic ciphers. Around 1467, Leon Battista Alberti, an Italian cryptographer, introduced the concept of using multiple alphabets in encryption, making it more complex and secure. His system, known as the polyphonic cipher, replaced each letter of the plaintext with one from a different alphabet, thus reducing the likelihood of successful cryptanalysis using frequency analysis. (Sidhpurwala, 2023)

Modern Cryptography

The modern cryptography started in late 20th century, In modern era the cryptography evolve with the computer science and the increased use of encryption in military and governmental communications. During World War I, the need for secure communication became most important thing, leading to a evolution of cryptographic activity. However, the major evolvement in modern cryptography happens during World War II.

The most significant contribution to modern cryptography came from the Enigma Machine. Developed by German cryptologist Arthur Scherbius in 1918, the Enigma used a series of rotors to create a polyalphabetic cipher that could be reset daily. It was used extensively by Nazi Germany during WWII for secure military communication. However, the breaking of the Enigma code by Allied forces, particularly by Alan Turing and his team at Bletchley Park, marked one of the most important events in cryptographic history. Turing's work laid the foundation for computational cryptography and was crucial in the development of computer algorithms.

In 1975, cryptographic theory saw a revolutionary shift with the development of public-key cryptography. Prior to this, all cryptosystems relied on shared private keys between the sender and receiver, a model that was vulnerable if the key was intercepted. The Diffie-Hellman key exchange introduced a new approach where users could establish a shared secret key over an insecure channel without ever directly transmitting the key itself. This laid the groundwork for asymmetric cryptography.

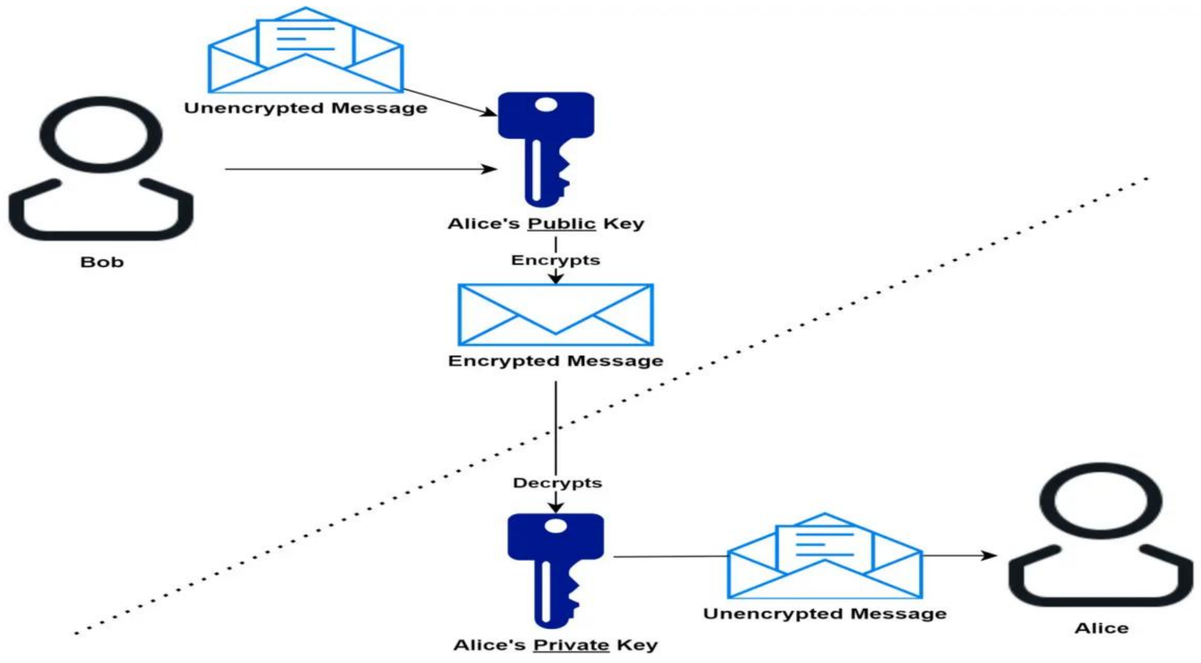


Figure 3: Modern cryptography

(Tyson, 2021)

Another major milestone in modern cryptography occurred in 2001 with the creation of the Advanced Encryption Standard (AES), which in came to use in common instead of Data Encryption Standard (DES). AES, designed to be computationally secure against modern cyber-attacks, these types of symmetric encryption algorithm are widely used in government, finance, and commercial applications departments. Today, cryptography continues to evolve with the technologies like quantum cryptography offering new level of secure communication. Quantum encryption relies on the principles of quantum mechanics, potentially making encryption systems unbreakable by commonly used computers.

Symmetric Encryption

Symmetric encryption is simplest kind of encryption which uses a single secret key for both encryption and decryption. The secret key, which can be a number, word, or string of random characters. The key must be shared between the sender and recipient securely. Examples include AES (128, 192, 256-bit), DES, RC5, and Blowfish. It is widely used in real-time communications like WhatsApp where messages are encrypted and decrypted quickly. In these applications, messages are encrypted on the sender's device using a symmetric key and then sent to the recipient, who uses the same key to decrypt the message.

Advantages of symmetric encryption

1. It is efficient and faster due to the use of a single key for both encryption and decryption.
2. It is easy to manage because only one key is needed for encryption and decryption.
3. It is ideal for bulk encryption tasks, such as encrypting files, databases, and backups.

Disadvantages of symmetric encryption

1. Securely sharing the key between parties poses challenges and risks of key compromise.
2. It is not suitable for large networks or multiple users due to difficulties in securely distributing and managing the key. (Gupta, 2024)

Asymmetric Encryption

Asymmetric encryption uses two types of keys a public key for encryption and a private key for decryption of the share data. The public key is shared openly, while the private key share or kept secretly, ensuring secure communication without the need to exchange keys. This system removes the need for us to share private keys, which enhances security, especially in open networks like the internet. This method is mostly used for encrypting small data, encryption keys or secure connection protocols like SSL/TLS.

Advantages:

It eliminates the need to share a private key, enhancing security.

It supports secure communication with multiple users, making it ideal for large networks.

Disadvantages:

It is slower than symmetric encryption due to the complex operations involved.

It is harder to implement and manage, requiring more resources for effective operation. (okeke, 2022)

Background

The name of new algorithm that created is “LRD algorithm”.

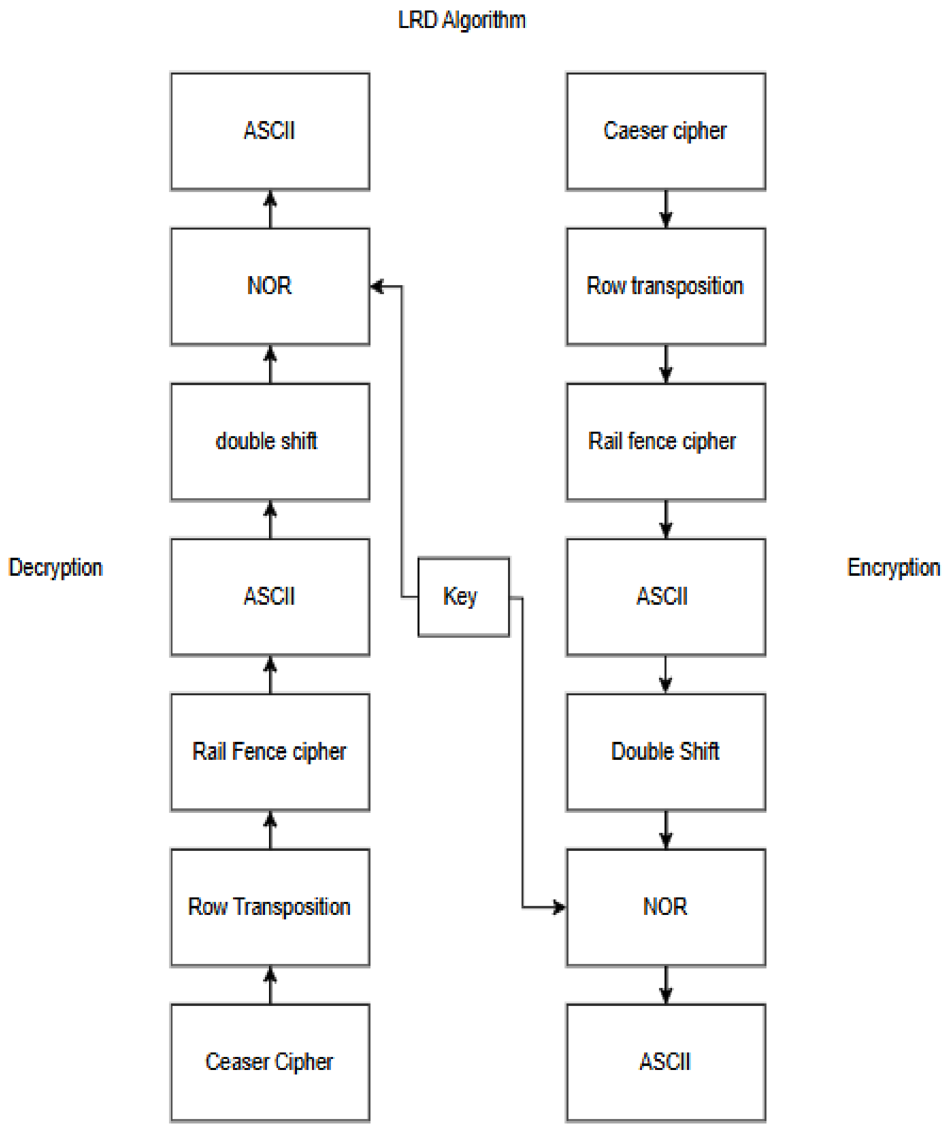


Figure 4: Flow chart

1) Caesar cipher

The Caesar cipher is one of the simplest and oldest encryption techniques. The cease cipher was name after Julius Caesar. The Caesar cipher is done by transposition and shifting of each letter of the plaintext message by a certain number of letters, historically 3. The ciphertext can be decrypted by applying the same number of shifts in the opposite direction. This type of encryption is known as a substitution cipher, due to the substitution of one letter for another in a consistent fashion. (geekforgeeks , 2024)

In this cipher, a shift value of three is used. A shift of 3 means that each letter

in the message is replaced by the letter that comes 3 places after it in the alphabet.

The algorithm can be expressed as:

C =(P+K) mod26

where, K takes the value in the range of 0– 25 for alphabet

• The decryption algorithm:

P =(C-K) mod 26

Example: Plaintext: HELLO

Ciphertext: KHOOR

It is easy and fast to understand, implement, encrypt and decrypt.

2) Row transposition

The Row Transposition Cipher is a classical encryption technique that rearranges the characters of a plaintext message according to a specific pattern defined by a key. It involves writing the plaintext into rows of a grid and then permuting the columns based on the key to generate the ciphertext.

Write Plaintext into Rows: Divide the plaintext into rows according to the length of the key.

Rearrange Columns Using the Key: Rearrange the columns in the order specified by the key.

Generate Ciphertext: Read the characters column by column to form the ciphertext.

Example:

Plaintext: "HELLO WORLD"

Cipher text: “RHOE LWLO DL”

Key: 3142 (length = 4) (Siam, 2023)

3) Rail fence cipher

The Rail Fence Cipher is a type of transposition cipher where plaintext characters are written diagonally across multiple rows in a zigzag or diagonal pattern and then read off row by row to form the ciphertext. This cipher rearranges the characters based on the number of rows chosen.

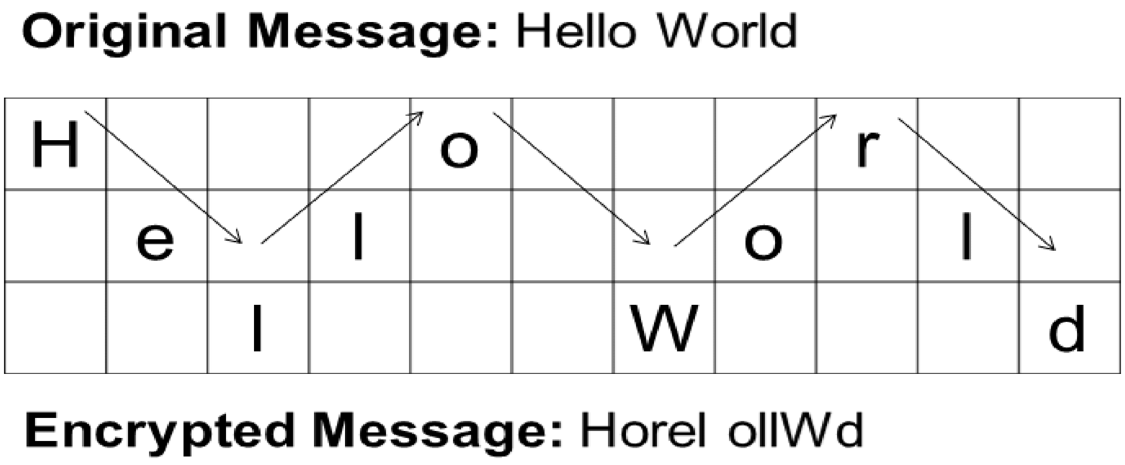

Original Message: Hello World
H
o
r
e
l
o
l
l
W
d
Encrypted Message: Horel ollWd

Figure 5: Rail fence encryption

(Sara farrag, 2019)

Advantages:

- Easy to implement and understand

- More secure than the ceaser cipher

Disadvantages:

- Can be easily cracked.
- Limited security compared to modern ciphers. (Rouse, 2014)

4) ASCII

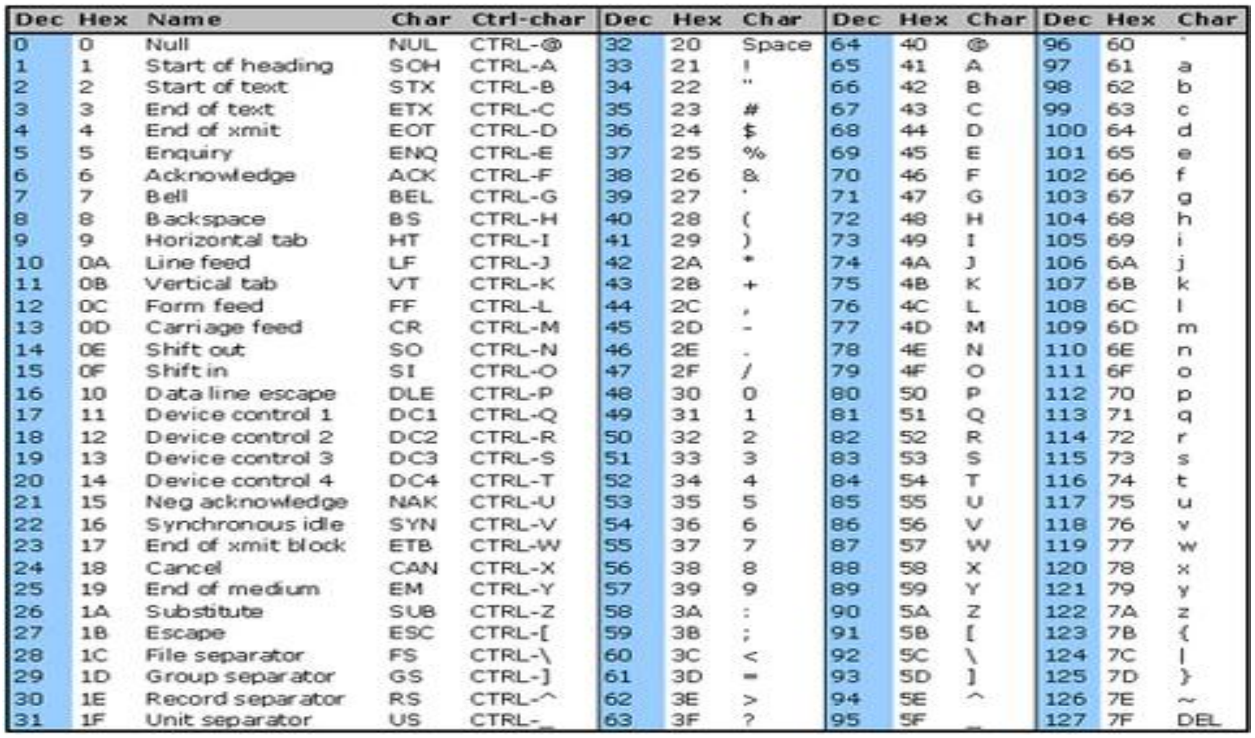
ASCII stands for American Standard Code for Information Interchange, a character encoding standard used to represent text and control commands in computers and communication devices. ASCII assigns a numerical value to each character, allowing computers to store and manipulate text as numbers.

ASCII Code Range:

Standard ASCII (0 to 127 in Decimal or 00 to 7F in Hexadecimal):

Control Characters (0 to 31): Non-printable characters used for control functions.

Printable Characters (32 to 127): Includes letters ('a'-'z', 'A'-'Z'), digits ('0'-'9'), punctuation marks, and symbols.



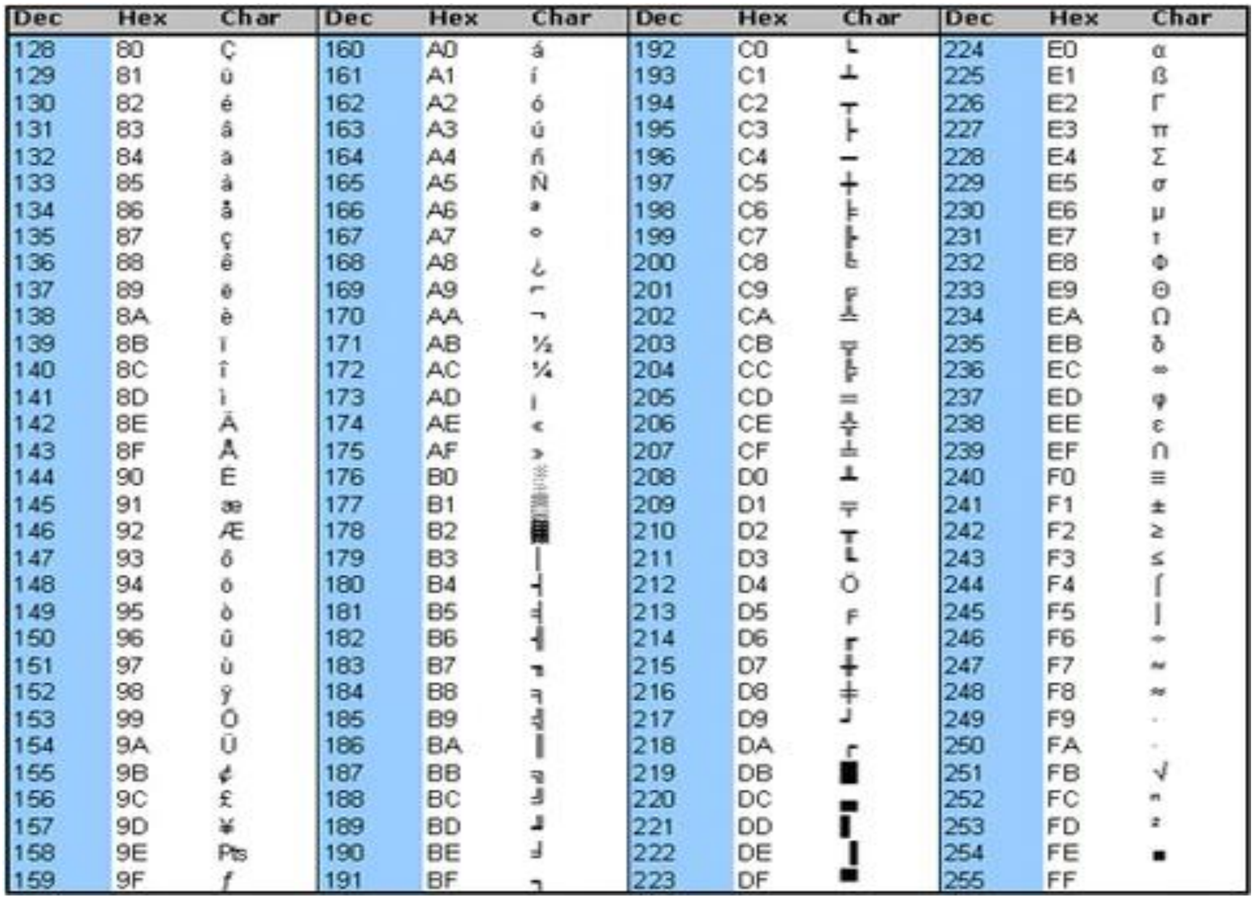
Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

Figure 6: Standard ASCII table

Extended ASCII (128 to 255 in Decimal or 80 to FF in Hexadecimal):

Extends the standard ASCII set to include additional characters, symbols, and graphical elements.

Supports accented characters, box-drawing characters, and symbols required by different languages and systems.



Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	Pts	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

Figure 7: Extended ASCII table

(Commfront , 2024)

5) Logical Gate

A logic gate is an essential building block in digital circuits, used to perform basic logical operations based on Boolean algebra. It operates on one or more binary inputs (0 or 1) and produces a binary output (0 or 1). Logic gates are foundational to modern electronic devices, including computers, smartphones, and memory systems, and are often implemented within integrated circuits. (Gillis, 2023)

NOR and NAND gates are considered as a universal gates because these have property, where any logical Boolean expressions can be formed by implementing these gates properly.

NOR gate: NOR gate is a digital gate that performs logical operation (NOT-OR).

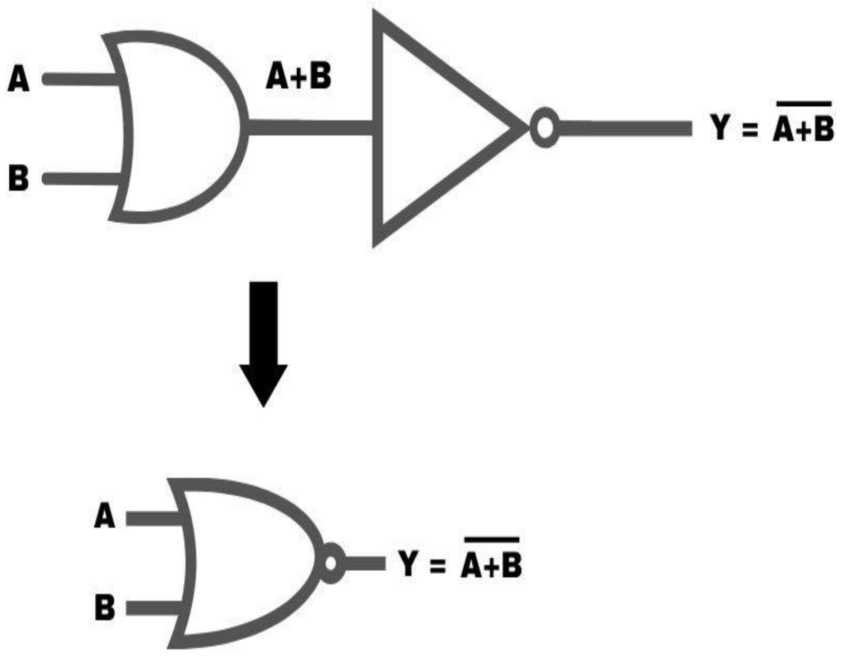


Figure 8: NOR gate

(karan, 2023)

Development

Encryption Process

Using Caesar cipher (+7)

The original Caesar cipher works on shifting the each letter of the give plaintext by +3. Instead of using 3 I will be shifting the letters by 7.

The algorithm can be expressed as:

C =(P+K) mod26

where, K takes the value in the range of 0– 25 for alphabet

• The decryption algorithm:

P =(C-K) mod 26

Example:

Plaintext: logic

Shift each letter by 7 positions (K=7):

Apply the formula C=(P+K) mod 26

L: C= (11+7) mod 26=18 (s)

O: C= (14+7) mod 26=21 (v)

G: C= (6+7) mod 26=13 (n)

I: C= (8+7) mod 26=15 (p)

C: C= (2+7) mod 26=9 (j)

Ciphertext: "svnpj"

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G

This table represents the value after the shifting.

Result:

Plaintext: logic

Ciphertext: svnpj

Using row transposition cipher

In common row transposition cipher the number of rows are 3 but in my algorithm I'm using 4 rows and key as "4132"

Ciphertext (from previous step): svnpj

Key: 4132

Write the text in rows with 4 columns and incomplete rows are left as it is:

s v n p

j

Rearrange columns according to the key order 4132:

Column 4: p

Column 1: s

Column 3: n

Column 2: v

After arrange according to key:

p s n v

j

Read column-wise following the key order 4132:

From column 4: p

From column 1: s

From column 3: n

From column 2: v

Then from the remaining row: j

Final ciphertext: psnvj

Result:

Input: svnpj

Output (encrypted using Row Transposition with key 4132): psnvj

Using Rail fence cipher

In normal rail fence the key word for the depth is 2. I will be using the depth 3.

Plain text: psnvj

Number of Rails: 4

Write the text row by row across the rails:

Rail 1: p

Rail 2: s

Rail 3: n

Rail 4: v

Combine the characters in order of the rails:

Rail 1: pj

Rail 2: s

Rail 3: n

Rail 4: v

Result in Ciphertext: pjsnv

Steps to convert into binary from letters:

Input Text: pjsnv

Find the ASCII value of each character:

p → ASCII: 112

j → ASCII: 106

s → ASCII: 115

n → ASCII: 110

v → ASCII: 118

Convert ASCII values to binary:

Binary representation of ASCII values is typically 8 bits:

p → 01110000

j → 01101010

s → 01110011

n → 01101110

v → 01110110

Combine the binary values:

01110000 01101010 01110011 01101110 01110110

Result:

The binary representation of "pjsnv" is:

01110000 01101010 01110011 01101110 01110110

Double Left Shift each binary value:

Shift left twice and add 00 at the end:

p → 01110000 → 11000000

j → 01101010 → 10101000

s → 01110011 → 11001100

n → 01101110 → 10111000

v → 01110110 → 11011000

The binary representation of "pjsnv" is:

01110000 01101010 01110011 01101110 01110110

After the double left shift, the binary values are:

11000000 10101000 11001100 10111000 11011000

Using NOR gate

The key for comparing the value is 11110000

Key: 11110000 (constant for comparison)

Binary Values (from the previous step):

p: 11000000

j: 10101000

s: 11001100

n: 10111000

v: 11011000

Apply NOR:

Formula for NOR: NOT (A OR B)

Use A (binary value) and B (key 11110000).

Perform NOR for each:

1. NOR Operation on p (11000000) with 11110000

A (p)	B (key)	A OR B	NOT (A OR B)
1	1	1	0
1	1	1	0
0	1	1	0
0	1	1	0
0	0	0	1
0	0	0	1
0	0	0	1
0	0	0	1

Result: 00001111

2. NOR Operation on j (10101000) with 11110000

A (j)	B (key)	A OR B	NOT (A OR B)
1	1	1	0
0	1	1	0
1	1	1	0
0	1	1	0
1	0	1	0
0	0	0	1
0	0	0	1
0	0	0	1

Result: 00000111

3. NOR Operation on s (11001100) with 11110000

A (s)	B (key)	A OR B	NOT (A OR B)
1	1	1	0
1	1	1	0
0	1	1	0
0	1	1	0
1	0	1	0
1	0	1	0
0	0	0	1
0	0	0	1

Result: 00000011

4. NOR Operation on n (10111000) with 11110000

A (n)	B (key)	A OR B	NOT (A OR B)
1	1	1	0
0	1	1	0
1	1	1	0
1	1	1	0
1	0	1	0
0	0	0	1
0	0	0	1
0	0	0	1

Result: 00000111

5. NOR Operation on v (11011000) with 11110000

A (v)	B (key)	A OR B	NOT (A OR B)
1	1	1	0
1	1	1	0
0	1	1	0
1	1	1	0
1	0	1	0
0	0	0	1
0	0	0	1
0	0	0	1

Result: 00000111

After applying the NOR gate:

p: 00001111

j: 00000111

s: 00000011

n: 00000111

v: 00000111

Combined result: 00001111 00000111 00000011 00000111 00000111

Conversion to Decimal and ASCII:

00001111 (Binary):

Decimal: 15

ASCII: The character corresponding to ASCII value 15

00000111 (Binary):

Decimal: 7

ASCII: The character corresponding to ASCII value 7

00000011 (Binary):

Decimal: 3

ASCII: The character corresponding to ASCII value 3

00000111 (Binary):

Decimal: 7

ASCII: The character corresponding to ASCII value 7

00000111 (Binary):

Decimal: 7

ASCII: The character corresponding to ASCII value 7

Result:

The decimal values are:

15, 7, 3, 7, 7

Original Decimal Values:

15, 7, 3, 7, 7

Add 65 to Each Value:

15 + 65 = 80

7 + 65 = 72

3 + 65 = 68

7 + 65 = 72

7 + 65 = 72

New Decimal Values:

80, 72, 68, 72, 72

Convert to ASCII Characters

80 → P

72 → H

68 → D

72 → H

72 → H

Result:

The corresponding ASCII characters after adding 65 are:

PHDHH

Decryption process

Let's reverse the process to decrypt PHDHH back to the original text, following the same steps used during encryption, but in reverse order.

Ciphertext: PHDHH

Step 1: Convert Characters to Binary (Reversed from ASCII Conversion)

Convert each character of PHDHH to its ASCII value:

P → 80

H → 72

D → 68

H → 72

H → 72

Subtract 65 (reverse of adding 65 during encryption):

80 - 65 = 15

72 - 65 = 7

68 - 65 = 3

72 - 65 = 7

72 - 65 = 7

Convert the results to binary:

15 → 00001111

7 → 00000111

3 → 00000011

7 → 00000111

7 → 00000111

Binary Output:

00001111 00000111 00000011 00000111 00000111

Step 2: Reverse the NOR Operation

Recall the NOR Step:

We had performed a NOR operation with the key 11110000. To reverse this:

Use the original binary value and perform the reverse operation. The reverse of NOR is NOT(A) OR NOT(B).

Perform Reverse NOR:

Using the binary values of above and the key (11110000):

Reverse for each:

00001111 → Original: 11000000

00000111 → Original: 10101000

00000011 → Original: 11001100

00000111 → Original: 10111000

00000111 → Original: 11011000

Output after Reverse NOR:

11000000 10101000 11001100 10111000 11011000

Step 3: Reverse the Double Left Shift

To reverse a double left shift, perform a double right shift:

11000000 → 01110000

10101000 → 00101010

11001100 → 00110011

10111000 → 00101110

11011000 → 00110110

Binary Output after Double Right Shift:

01110000 00101010 00110011 00101110 00110110

Step 4: Convert Binary to Characters

Convert the binary values back to characters using ASCII:

01110000 → p

00101010 → j

00110011 → s

00101110 → n

00110110 → v

Output: pjsnv

Step 5: Reverse Rail Fence Cipher

Rail Fence reverse:

We wrote the text row by row across 4 rails. To reverse this, redistribute the characters back to their respective rails:

Ciphertext: pjsnv

Rails:

Rail 1: p

Rail 2: s

Rail 3: n

Rail 4: v

Rail 1: j

Reconstruct the original sequence:

psnvj

Step 6: Reverse Row Transposition

The row transposition used the key 4132 with length 4. To reverse it:

Reconstruct the transposition table:

p s n v

j

Use the key 4132 to reorder the columns back to their original positions:

Column 4 → 1st position

Column 1 → 2nd position

Column 3 → 3rd position

Column 2 → 4th position

Reconstruct: svnpj

Reverse the shift by subtracting 7 (K=7):

Apply the formula $P=(C-K) \bmod 26$

S: $P= (18-7) \bmod 26=11$ (l)

v: $P= (21-7) \bmod 26=14$ (o)

n: $P= (13-7) \bmod 26=6$ (g)

p: $P= (15-7) \bmod 26=8$ (i)

j: $P= (9-7) \bmod 26=2$ (c)

Combine the resulting plaintext letters:

Plaintext: "logic"

Testing

Test 1

Ceaser cipher

Write the plaintext: logic

Apply the formula C=(P+K) mod 26

L: C= (11+7) mod 26=18 (s)

O: C= (14+7) mod 26=21 (v)

G: C= (6+7) mod 26=13 (n)

I: C= (8+7) mod 26=15 (p)

C: C= (2+7) mod 26=9 (j)

Combine the shifted letters to form the ciphertext: "svnpj"

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G

This table represents the value after the shifting.

Plaintext: logic

Ciphertext: svnpj

Steps for Row Transposition Cipher (Key = 4132):

Ciphertext: svnpj

Key: 4132

Write the text in rows (with 4 columns; incomplete rows are left as is):

s v n p

j

Rearrange columns according to the key order 4132:

Column 4: p

Column 1: s

Column 3: n

Column 2: v

p s n v

j

Read column-wise following the key order 4132:

From column 4: p

From column 1: s

From column 3: n

From column 2: v

Then from the remaining row: j

Final ciphertext: psnvj

Result:

Input: svnpj

Output (encrypted using Row Transposition with key 4132): psnvj

Steps for Rail Fence Cipher:

Input Text: psnvj

Number of Rails: 4

Write the text row by row across the rails:

Rail 1: p

Rail 2: s

Rail 3: n

Rail 4: v

Read Row by Row:

Combine the characters in order of the rails:

Rail 1: pj

Rail 2: s

Rail 3: n

Rail 4: v

Ciphertext: pjsnv

Steps to convert into binary from letters:

Input Text: pjsnv

Find the ASCII value of each character:

p → ASCII: 112

j → ASCII: 106

s → ASCII: 115

n → ASCII: 110

v → ASCII: 118

Convert ASCII values to binary:

Binary representation of ASCII values is typically 8 bits:

p → 01110000

j → 01101010

s → 01110011

n → 01101110

v → 01110110

Combine the binary values:

01110000 01101010 01110011 01101110 01110110

Result:

The binary representation of "pjsnv" is:

01110000 01101010 01110011 01101110 01110110

Double Left Shift each binary value:

Shift left twice and add 00 at the end:

p → 01110000 → 11000000

j → 01101010 → 10101000

s → 01110011 → 11001100

n → 01101110 → 10111000

v → 01110110 → 11011000

The binary representation of "pjsnv" is:

01110000 01101010 01110011 01101110 01110110

After the double left shift, the binary values are:

11000000 10101000 11001100 10111000 11011000

Using NOR gate

The key for comparing the value is 11110000

Key: 11110000 (constant for comparison)

Binary Values:

p: 11000000

j: 10101000

s: 11001100

n: 10111000

v: 11011000

Apply NOR:

Formula for NOR: NOT (A OR B)

Use A (binary value) and B (key 11110000).

1. NOR Operation on p (11000000) with 11110000

A (p)	B (key)	A OR B	NOT (A OR B)
1	1	1	0
1	1	1	0
0	1	1	0
0	1	1	0
0	0	0	1
0	0	0	1
0	0	0	1
0	0	0	1

Result: 00001111

2. NOR Operation on j (10101000) with 11110000

A (j)	B (key)	A OR B	NOT (A OR B)
1	1	1	0
0	1	1	0
1	1	1	0
0	1	1	0
1	0	1	0
0	0	0	1
0	0	0	1
0	0	0	1

Result: 00000111

3. NOR Operation on s (11001100) with 11110000

A (s)	B (key)	A OR B	NOT (A OR B)
1	1	1	0
1	1	1	0
0	1	1	0
0	1	1	0
1	0	1	0
1	0	1	0
0	0	0	1
0	0	0	1

Result: 00000011

4. NOR Operation on n (10111000) with 11110000

A (n)	B (key)	A OR B	NOT (A OR B)
1	1	1	0
0	1	1	0
1	1	1	0
1	1	1	0
1	0	1	0
0	0	0	1
0	0	0	1
0	0	0	1

Result: 00000111

5. NOR Operation on v (11011000) with 11110000

A (v)	B (key)	A OR B	NOT (A OR B)
1	1	1	0
1	1	1	0
0	1	1	0
1	1	1	0
1	0	1	0
0	0	0	1
0	0	0	1
0	0	0	1

Result: 00000111

After applying the NOR gate:

p: 00001111

j: 00000111

s: 00000011

n: 00000111

v: 00000111

Combined result: 00001111 00000111 00000011 00000111 00000111

Conversion to Decimal and ASCII:

00001111 (Binary):

Decimal: 15

ASCII: The character corresponding to ASCII value 15

00000111 (Binary):

Decimal: 7

ASCII: The character corresponding to ASCII value 7

00000011 (Binary):

Decimal: 3

ASCII: The character corresponding to ASCII value 3

00000111 (Binary):

Decimal: 7

ASCII: The character corresponding to ASCII value 7

00000111 (Binary):

Decimal: 7

ASCII: The character corresponding to ASCII value 7

Result:

The decimal values are:

15, 7, 3, 7, 7

Original Decimal Values:

15, 7, 3, 7, 7

Add 65 to Each Value:

15 + 65 = 80

7 + 65 = 72

3 + 65 = 68

7 + 65 = 72

7 + 65 = 72

New Decimal Values:

80, 72, 68, 72, 72

Convert to ASCII Characters:

80 → P

72 → H

68 → D

72 → H

72 → H

Result:

The corresponding ASCII characters after adding 65 are:

PHDHH

Decryption process

Let's reverse the process to decrypt PHDHH back to the original text, following the same steps used during encryption, but in reverse order.

Ciphertext: PHDHH

Step 1: Convert Characters to Binary (Reversed from ASCII Conversion)

Convert each character of PHDHH to its ASCII value:

P → 80

H → 72

D → 68

H → 72

H → 72

Subtract 65 (reverse of adding 65 during encryption):

80 - 65 = 15

72 - 65 = 7

68 - 65 = 3

72 - 65 = 7

72 - 65 = 7

Convert the results to binary:

15 → 00001111

7 → 00000111

3 → 00000011

7 → 00000111

7 → 00000111

Binary Output:

00001111 00000111 00000011 00000111 00000111

Step 2: Reverse the NOR Operation

Recall the NOR Step:

We had performed a NOR operation with the key 11110000. To reverse this:

Use the original binary value and perform the reverse operation. The reverse of NOR is NOT(A) OR NOT(B).

Perform Reverse NOR:

Using the binary values (00001111, etc.) and the key (11110000):

00001111 → Original: 11000000

00000111 → Original: 10101000

00000011 → Original: 11001100

00000111 → Original: 10111000

00000111 → Original: 11011000

Output after Reverse NOR:

11000000 10101000 11001100 10111000 11011000

Step 3: Reverse the Double Left Shift

To reverse a double left shift, perform a double right shift:

11000000 → 01110000

10101000 → 00101010

11001100 → 00110011

10111000 → 00101110

11011000 → 00110110

Binary Output after Double Right Shift:

01110000 00101010 00110011 00101110 00110110

Step 4: Convert Binary to Characters

Convert the binary values back to characters using ASCII:

01110000 → p

00101010 → j

00110011 → s

00101110 → n

00110110 → v

Output: pjsnv

Step 5: Reverse Rail Fence Cipher

Rail Fence Encoding:

We wrote the text row by row across 4 rails. To reverse this, redistribute the characters back to their respective rails:

Ciphertext: pjsnv

Rails:

Rail 1: p

Rail 2: s

Rail 3: n

Rail 4: v

Rail 1: j

Reconstruct the original sequence:

psnvj

Step 6: Reverse Row Transposition

The row transposition used the key 4132 with length 4. To reverse it:

Reconstruct the transposition table:

p s n v

j

Use the key 4132 to reorder the columns back to their original positions:

Column 4 → 1st position

Column 1 → 2nd position

Column 3 → 3rd position

Column 2 → 4th position

Reconstruct: svnpj

Reverse the shift by subtracting 7 (K=7):

Apply the formula P=(C−K) mod 26

S: P= (18-7) mod 26=11 (l)

v: P= (21-7) mod 26=14 (o)

n: P= (13-7) mod 26=6 (g)

p: P= (15-7) mod 26=8 (i)

j: P= (9−7) mod 26=2 (c)

Combine the resulting plaintext letters:

Plaintext: "logic"

Test 2

Input: "tell"

Step 1: Caesar Cipher Encryption

Shift each letter 7 positions to the right in the alphabet:

Apply the formula C=(P+K)mod 26 with K=7:

T : C= (19+7) mod 26=0 (a)

E : C= (4+7) mod 26=11 (l)

L: C= (11+7) mod 26=18 (s)

L: C= (11+7) mod 26=18 (s)

Result after Caesar Cipher: "**alss**"

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G

This table represents the value after the shifting.

Step 2: Row Transposition Cipher

Key: 4132 (Key length = 4)

Write "alss" into rows of 4:

a l s s

Rearrange columns based on the key 4132:

Column 4 → 1st position: s

Column 1 → 2nd position: a

Column 3 → 3rd position: s

Column 2 → 4th position: l

Result after Row Transposition: "sasl"

Step 3: Rail Fence Cipher

Use 4 rails

Write row by row across the 4 rails:

Rail 1: `s`

Rail 2: `a`

Rail 3: `s`

Rail 4: `l`

Result after Rail Fence Cipher: "sasl"

Step 4: Convert to Binary (ASCII)

Convert each character to its binary ASCII representation:

s → 01110011

a → 01100001

s → 01110011

l → 01101100

Binary Output: 01110011 01100001 01110011 01101100

Step 5: Double Left Shift

Perform a double left shift on each binary value:

01110011 → 11001100

01100001 → 10000110

01110011 → 11001100

01101100 → 10110000

Binary Output after Double Left Shift:

11001100 10000110 11001100 10110000

Step 6: NOR Logical Gate

Perform the NOR operation with the constant key 11110000:

NOR Truth Table:

For each bit in the binary string and the key:

A = binary value, B = key

NOR = NOT (A OR B)

Apply NOR for each binary value:

11001100 NOR 11110000 → 00000011

A (11001100)	B (11110000)	A OR B	NOT (A OR B)
1	1	1	0
1	1	1	0
0	1	1	0
0	1	1	0
1	0	1	0
1	0	1	0
0	0	0	1
0	0	0	1

Result: 00000011

10000110 NOR 11110000 → 00000001

Bit Position	A (10000110)	B (11110000)	A OR B	NOT (A OR B)
1	1	1	1	0
2	0	1	1	0
3	0	1	1	0
4	0	1	1	0
5	1	0	1	0
6	1	0	1	0
7	0	0	0	1
8	0	0	0	1

Result: 00000001

11001100 NOR 11110000 → 00000011

Bit Position	A (11001100)	B (11110000)	A OR B	NOT (A OR B)
1	1	1	1	0
2	1	1	1	0
3	0	1	1	0
4	0	1	1	0
5	1	0	1	0
6	1	0	1	0
7	0	0	0	1
8	0	0	0	1

Result: 00000011

10110000 NOR 11110000 → 00000000

Bit Position	A (10110000)	B (11110000)	A OR B	NOT (A OR B)
1	1	1	1	0
2	0	1	1	0
3	1	1	1	0
4	1	1	1	0
5	0	0	0	1
6	0	0	0	1
7	0	0	0	1
8	0	0	0	1

Result: 00000000

Binary Output after NOR:

00000011 00000001 00000011 00000000

Step 7: Convert to Decimal and Add 65

Convert each binary value to decimal:

00000011 → 3

00000001 → 1

00000011 → 3

00000000 → 0

Add 65 to each decimal value:

3 + 65 = 68

1 + 65 = 66

3 + 65 = 68

0 + 65 = 65

Convert back to ASCII characters:

68 → D

66 → B

68 → D

65 → A

Result after Adding 65: "DBDA"

Final Encrypted Text: "DBDA"

Decryption Steps

Ciphertext: "DBDA"

Step 1: Reverse Adding 65

Convert each character to ASCII values:

D → 68

B → 66

D → 68

A → 65

Subtract 65:

68 - 65 = 3

66 - 65 = 1

68 - 65 = 3

65 - 65 = 0

Convert to binary:

3 → 00000011

1 → 00000001

3 → 00000011

0 → 00000000

Binary Output: 00000011 00000001 00000011 00000000

Step 2: Reverse NOR Logical Gate

Perform the reverse NOR operation to retrieve the original binary values:

00000011 → 11001100

00000001 → 10000110

00000011 → 11001100

00000000 → 10110000

Binary Output after Reverse NOR:

11001100 10000110 11001100 10110000

Step 3: Reverse Double Left Shift

Perform a double right shift:

11001100 → 01110011

10000110 → 01100001

11001100 → 01110011

10110000 → 01101100

Binary Output after Double Right Shift:

01110011 01100001 01110011 01101100

Step 4: Convert Binary to Characters

Convert the binary values to ASCII characters:

01110011 → s

01100001 → a

01110011 → s

01101100 → l

Result: "sasl"

Step 5: Reverse Rail Fence Cipher

Rail 1: `s`

Rail 2: `a`

Rail 3: `s`

Rail 4: `l`

Reconstruct row by row: "sasl"

Step 6: Reverse Row Transposition

Using the key 4132, reverse the column rearrangement:

Rearrange to original positions: "alss"

Step 7: Reverse Caesar Cipher

Apply the formula $P=(C-K) \bmod 26$ with $K=7$:

A : $P= (0-7) \bmod 26=19$ (t)

L : $P= (11-7) \bmod 26=4$ (e)

S: $P= (18-7) \bmod 26=11$ (l)

S: $P= (18-7) \bmod 26=11$ (l)

Final Decrypted Text: **"tell"**

Test 3

Input Word: "lila"

1. Caesar Cipher

For each character in "lila", shift by 7 positions.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G

This table represents the value after the shifting.

Input Word: "lila"

For l: C= (11+7) mod 26=18 (s)

For i: C= (8+7) mod 26=15 (p)

For l: C= (11+7) mod 26=18 (s)

For a: C= (0+7) mod 26=7 (h)

Caesar Cipher Result: "spsh"

2. Row Transposition Cipher (Key = 4132)

Now, apply the Row Transposition Cipher with a key length of 4 and key 4132.

Write the text "spsh" into rows of 4:

s p s h

Rearrange columns based on the key 4132:

Column 4 → 1st position: h

Column 1 → 2nd position: s

Column 3 → 3rd position: s

Column 2 → 4th position: p

Row Transposition Result: "hssp"

3. Rail Fence Cipher (4 Rails,)

Write each character of "hssp" across 4 rails:

Write characters directly into rails:

Rail 1: `h`

Rail 2: `s`

Rail 3: `s`

Rail 4: `p`

Result: "hssp"

Rail Fence Cipher Result: "hssp"

4. Convert to Binary (ASCII)

Convert each character of "hssp" to its binary ASCII representation:

h → 01101000

s → 01110011

s → 01110011

p → 01110000

Binary Output:

01101000 01110011 01110011 01110000

5. **Double Left Shift**

Shift each binary value two positions to the left:

01101000 → 11010000

01110011 → 11100110

01110011 → 11100110

01110000 → 11100000

Binary Output after Left Shift:

11010000 11100110 11100110 11100000

6. **Apply NOR Logical Gate**

The key is 11110000. Perform a NOR operation for each binary value:

NOR = NOT (A OR B).

Calculation:

1. 11010000 NOR 11110000 → 00001111

A (Input)	B (Key)	A OR B	NOT (A OR B)
1	1	1	0
1	1	1	0
0	1	1	0
1	1	1	0
0	0	0	1
0	0	0	1
0	0	0	1
0	0	0	1
Result:		00001111	

2. 11100110 NOR 11110000 → 00001100

A (Input)	B (Key)	A OR B	NOT (A OR B)
1	1	1	0
1	1	1	0
1	1	1	0
0	1	1	0
0	0	0	1
1	0	1	0
1	0	1	0
0	0	0	1
Result:		00001100	

3. 11100110 NOR 11110000 → 00001100

A (Input)	B (Key)	A OR B	NOT (A OR B)
1	1	1	0
1	1	1	0
1	1	1	0
0	1	1	0
0	0	0	1
1	0	1	0
1	0	1	0
0	0	0	1
Result:		00001100	

4. 11100000 NOR 11110000 → 00001111

A (Input)	B (Key)	A OR B	NOT (A OR B)
1	1	1	0
1	1	1	0
1	1	1	0
0	1	1	0
0	0	0	1
0	0	0	1
0	0	0	1
0	0	0	1
Result:	00001111		

Binary Output after NOR:

00001111 00001100 00001100 00001111

7. Convert Binary to Decimal and Add 65

Convert each binary value to decimal:

00001111 → 15

00001100 → 12

00001100 → 12

00001111 → 15

Add 65 to each decimal value:

15 + 65 = 80

12 + 65 = 77

12 + 65 = 77

15 + 65 = 80

Convert to characters:

80 → P

77 → M

77 → M

80 → P

Final Encrypted Text: "PMPM"

Decryption Process

Ciphertext: "PMPM"

1. Reverse Adding 65

Convert each character to its ASCII value:

P → 80

M → 77

M → 77

P → 80

Subtract 65 from each value:

80 - 65 = 15

77 - 65 = 12

77 - 65 = 12

80 - 65 = 15

Convert to binary:

15 → 00001111

12 → 00001100

12 → 00001100

15 → 00001111

Binary Output:

00001111 00001100 00001100 00001111

2. **Reverse NOR Logical Gate**

Reverse the NOR operation using the key 11110000:

Calculation:

00001111 → 11010000

00001100 → 11100110

00001100 → 11100110

00001111 → 11010000

Binary Output after Reverse NOR:

11010000 11100110 11100110 11010000

3. Reverse Double Left Shift

Perform a double right shift:

11010000 → 01101000

11100110 → 01110011

11100110 → 01110011

11010000 → 01101000

Output after Double Right Shift: 01101000 01110011 01110011 01101000

4. Convert Binary to Characters

Convert binary to characters:

01101000 → h

01110011 → s

01110011 → s

01101000 → h

Decrypted Text: "hssh"

5. Reverse Rail Fence Cipher

Rebuild the text across 4 rails:

Rail 1: `h`

Rail 2: `s`

Rail 3: `s`

Rail 4: `h`

Reconstruct: "hssh"

6. Reverse Row Transposition

Rearrange columns back to original order (key 4132):

Rearrange: "hssp"

Final:spsh

7. Reverse Caesar Cipher

Apply a 7-position:

For S:

$P = (18 - 7) \bmod 26 = 11$ (l)

For P:

P= (15−7) mod 26=8 (i)

For S:

P= (18−7) mod 26=11 (l)

For H:

P= (7−7) mod 26=0 (a)

Decrypted Text: "lila"

Test 4

Caesar Cipher with 7-position shift

Input Word: "DURA"

For d:

C= (3+7) mod 26=10 (k)

For u:

C= (20+7) mod 26=1 (b)

For r:

C= (17+7) mod 26=24 (y)

For a:

C= (0+7) mod 26=7 (h)

Caesar Cipher Result: "KBYH"

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G

This table represents the value after the shifting.

2. Row Transposition Cipher (Key = 4132)

Write "KBYH" into rows of length 4: K B Y H

Rearrange columns based on the key 4132:

Column 4 → 1st position: H

Column 1 → 2nd position: K

Column 3 → 3rd position: Y

Column 2 → 4th position: B

Row Transposition Result: "HKYB"

3. Rail Fence Cipher (4 Rails)

Distribute the text "HKYB" across 4 rails sequentially:

Rail 1: H

Rail 2: K

Rail 3: Y

Rail 4: B

Result: "HKYB"

4. Convert to Binary (ASCII)

Convert each character of "HKYB" to its binary ASCII representation:

H → 01001000

K → 01001011

Y → 01011001

B → 01000010

Binary Output:

01001000 01001011 01011001 01000010

5. Double Left Shift

Shift each binary value two positions to the left:

01001000 → 00100000

01001011 → 00101100

01011001 → 01100100

01000010 → 00001000

Output after Left Shift:

00100000 00101100 01100100 00001000

6. Apply NOR Logical Gate

The key is 11110000. Perform a NOR operation for each binary value:

NOR = NOT (A OR B).

Calculation:

1. 00100000 NOR 11110000 → 00001111

A (Input)	B (Key)	A OR B	NOT (A OR B)
0	1	1	0
0	1	1	0
1	1	1	0
0	1	1	0
0	0	0	1
0	0	0	1
0	0	0	1
0	0	0	1
Result:		00001111	

2. 00101100 NOR 11110000 → 00000011

A (Input)	B (Key)	A OR B	NOT (A OR B)
0	1	1	0
0	1	1	0
1	1	1	0
0	1	1	0
1	0	1	0
1	0	1	0
0	0	0	1
0	0	0	1

Result: 00000011

3. 01100100 NOR 11110000 → 00000011

A (Input)	B (Key)	A OR B	NOT (A OR B)
0	1	1	0
1	1	1	0
1	1	1	0
0	1	1	0
0	0	0	1
1	0	1	0
0	0	0	1
0	0	0	1

Result: 00000011

4. 00001000 NOR 11110000 → 00000011

A (Input)	B (Key)	A OR B	NOT (A OR B)
0	1	1	0
0	1	1	0
0	1	1	0
0	1	1	0
1	0	1	0
0	0	0	1
0	0	0	1
0	0	0	1
Result:	00000011		

Binary Output after NOR:

00001111 00000011 00000011 00000011

7. Convert Binary to Decimal and Add 65

Convert each binary value to decimal:

00001111 → 15

00000011 → 3

00000011 → 3

00000011 → 3

Add 65 to each decimal value

15 + 65 = 80

3 + 65 = 68

3 + 65 = 68

3 + 65 = 68

Convert to characters:

80 → P

68 → D

68 → D

68 → D

Final Encrypted Text: "PDDD"

Decryption Process

Ciphertext: "PDDD"

1. Reverse of Adding 65

Convert each character to its ASCII value:

P → 80

D → 68

D → 68

D → 68

Subtract 65 from each value:

80 - 65 = 15

68 - 65 = 3

68 - 65 = 3

68 - 65 = 3

Convert to binary:

15 → 00001111

3 → 00000011

3 → 00000011

3 → 00000011

Binary Output:

00001111 00000011 00000011 00000011

2. Reverse NOR Logical Gate

Reverse the NOR operation using the key 11110000:

00001111 → 00100000

00000011 → 00101100

00000011 → 01100100

00000011 → 00001000

Binary Output after Reverse NOR:

00100000 00101100 01100100 00001000

3. Reverse Double Left Shift

Perform a double right shift:

00100000 → 01001000

00101100 → 01001011

01100100 → 01011001

00001000 → 01000010

Binary Output after Double Right Shift:

01001000 01001011 01011001 01000010

4. Convert Binary to Characters

Convert binary to characters:

01001000 → H

01001011 → K

01011001 → Y

01000010 → B

Decrypted Text: "HKYB"

5. Reverse Rail Fence Cipher

Rebuild the text across 4 rails:

Rail 1: \`H\`

Rail 2: \`K\`

Rail 3: \`Y\`

Rail 4: \`B\`

Reconstruct: "HKYB"

6. Reverse Row Transposition

Rearrange columns back to original order (key 4132):

Rearrange: "KBYH"

7. Reverse Caesar Cipher

For k: P= (10−7) mod 26=3 (d)

For b: P= (1−7) mod 26=20 (u)

For y: P= (24−7) mod 26=17 (r)

For h: P= (7−7) mod 26=0 (a)

Decrypted Text: "DURA"

Test 5

Caesar Cipher with a 7-position shift

Input Word: "ROOM"

For r:

C= (17+7) mod 26=24 (y)

For o:

C= (14+7) mod 26=21 (v)

For o:

C= (14+7) mod 26=21 (v)

For m:

C= (12+7) mod 26=19 (t)

Caesar Cipher Result: "YVVT"

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G

This table represents the value after the shifting.

Step 2: Row Transposition Cipher (Key = 4132)

Write "YVVT" into rows of length 4:

Y V V T

Rearrange columns based on the key 4132:

Column 4 → 1st position: T

Column 1 → 2nd position: Y

Column 3 → 3rd position: V

Column 2 → 4th position: V

Row Transposition Result: "TYVV"

Step 3: Rail Fence Cipher (4 Rails)

Distribute the text "TYVV" across 4 rails sequentially:

Rail 1: T

Rail 2: Y

Rail 3: V

Rail 4: V

Result: "TYVV"

Rail Fence Cipher Result: "TYVV"

Step 4: Convert to Binary (ASCII)

Convert each character of "TYVV" to its binary ASCII representation:

T → 01010100

Y → 01011001

V → 01010110

V → 01010110

Binary Output:

01010100 01011001 01010110 01010110

Step 5: Double Left Shift

Shift each binary value two positions to the left:

01010100 → 10101000

01011001 → 01100100

01010110 → 01011000

01010110 → 01011000

Binary Output after Left Shift:

10101000 01100100 01011000 01011000

Step 6: Apply NOR Logical Gate

The key is 11110000. Perform a NOR operation for each binary value:

NOR = NOT (A OR B).

Calculation:

1. 10101000 NOR 11110000 → 00000001

A (Input)	B (Key)	A OR B	NOT (A OR B)
1	1	1	0
0	1	1	0
1	1	1	0
0	1	1	0
1	0	1	0
0	0	0	1
0	0	0	1
0	0	0	1
Result:		00000001	

2. 01100100 NOR 11110000 → 00000011

A (Input)	B (Key)	A OR B	NOT (A OR B)
0	1	1	0
1	1	1	0
1	1	1	0
0	1	1	0
0	0	0	1
1	0	1	0
0	0	0	1
0	0	0	1
Result:		00000011	

3. 01011000 NOR 11110000 → 00000011

A (Input)	B (Key)	A OR B	NOT (A OR B)
0	1	1	0
1	1	1	0
0	1	1	0
1	1	1	0
1	0	1	0
0	0	0	1
0	0	0	1
0	0	0	1

Result: 00000011

4. 01011000 NOR 11110000 → 00000011

A (Input)	B (Key)	A OR B	NOT (A OR B)
0	1	1	0
1	1	1	0
0	1	1	0
1	1	1	0
1	0	1	0
0	0	0	1
0	0	0	1
0	0	0	1

Result: 00000011

Binary Output after NOR:

00000001 00000011 00000011 00000011

Step 7: Convert Binary to Decimal and Add 65

Convert each binary value to decimal:

00000001 → 1

00000011 → 3

00000011 → 3

00000011 → 3

Add 65 to each decimal value:

1 + 65 = 66

3 + 65 = 68

3 + 65 = 68

3 + 65 = 68

Convert to characters:

66 → B

68 → D

68 → D

68 → D

Final Encrypted Text: "BDDD"

Decryption Process

Ciphertext: "BDDD"

Step 1: Reverse Adding 65

Convert each character to its ASCII value:

B → 66

D → 68

D → 68

D → 68

Subtract 65 from each value

66 - 65 = 1

68 - 65 = 3

68 - 65 = 3

68 - 65 = 3

Convert to binary:

1 → 00000001

3 → 00000011

3 → 00000011

3 → 00000011

Binary Output:

00000001 00000011 00000011 00000011

Step 2: Reverse NOR Logical Gate

Reverse the NOR operation using the key 11110000:

Calculation:

00000001 → 10101000

00000011 → 01100100

00000011 → 01011000

00000011 → 01011000

Binary Output after Reverse NOR:

10101000 01100100 01011000 01011000

Step 3: Reverse Double Left Shift

Perform a double right shift:

10101000 → 01010100

01100100 → 01011001

01011000 → 01010110

01011000 → 01010110

Binary Output after Double Right Shift:

01010100 01011001 01010110 01010110

Step 4: Convert Binary to Characters

Convert binary to characters:

01010100 → T

01011001 → Y

01010110 → V

01010110 → V

Decrypted Text: "TYVV"

Step 5: Reverse Rail Fence Cipher

Rebuild the text across 4 rails:

Rail 1: `T`

Rail 2: `Y`

Rail 3: `V`

Rail 4: `V`

Reconstruct: "TYVV"

Step 6: Reverse Row Transposition

Rearrange columns back to original order (key 4132):

Rearrange: "YVVT"

Step 7: Reverse Caesar Cipher

Apply a 7-position shift:

for y:

P= (24−7) mod 26=17 (r)

For v:

P= (21−7) mod 26=14 (o)

For v:

P= (21−7) mod 26=14 (o)

For t:

P= (19−7) mod 26=12 (m)

Decrypted Text: "ROOM"

Evaluation

Strengths of this algorithm

- Using multiple encryption steps makes it tough for hackers to crack.
- Adding a NOR gate step makes it unique and harder to guess.
- You can easily tweak the settings to make it more personalized.
- Turning letters into binary adds an extra layer of confusion for attackers.
- Without knowing all the steps, decoding it is nearly impossible.

Weaknesses of this algorithm

- It's complicated to understand and easy to mess up.
- If someone gets hold of your keys, they can decrypt everything.
- It takes a lot of time and effort for larger messages.
- Some parts, like the rail fence and transposition, follow predictable patterns.
- If there's a small mistake or data error, you might not be able to decode it.

Application area:

1. Learning About Encryption: It's a good tool for students and beginners to understand how different encryption techniques work together.
2. Protecting Important Files: It's great for locking up sensitive files before sending or storing them, keeping them away from hackers.
3. Securing Passwords: This can help store passwords safely, making it harder for attackers to figure them out.
4. Government and Military Use: It's useful for sending top-secret messages that need to stay confidential.
5. Smart Devices Security: Works well in small gadgets like smart home devices, keeping their data safe without using too much power.
6. Cloud Storage Safety: Before uploading files to the cloud, this can add an extra security step to keep data private.

Conclusion

In a world where data is a vital asset, security has never been more critical. The ancient cryptographic methods like the scytale to modern advancements such as quantum cryptography underscores humanity's constant quest to protect information. Cryptography serves as the backbone of security, meeting the CIA triad's principles of Confidentiality, Integrity, and Availability.

This study introduced the "LRD Algorithm," a novel approach that combines classic ciphers, binary operations, and logical gates to create a highly secure encryption method. By leveraging the simplicity of techniques like the Caesar cipher and the complexity of modern cryptographic operations, the LRD algorithm strikes a balance between efficiency and robustness. The layered structure of the algorithm ensures that each step contributes to overall security, making it a valuable tool in today's cybersecurity landscape.

As technology advances, the threats to data security grow, but so do the opportunities to innovate and strengthen defenses. The LRD algorithm is a testament to the enduring importance of cryptography, demonstrating that even age-old principles can inspire modern solutions. This work not only highlights the significance of securing data but also underscores the need for continued research and adaptation in the ever-evolving field of information security.

References

Coinloan, 2023. *coinloan.* [Online]
Available at: https://coinloan.io/blog/history-of-cryptography/
[Accessed 10 12 2024].

Commfront , 2024. *commfront.* [Online]
Available at: https://www.commfront.com/pages/ascii-chart
[Accessed 06 01 2025].

Dottx, 2023. *medium.* [Online]
Available at: https://medium.com/@np01nt4s220042/simplified-data-encryption-standard-8ab7061eaa3c
[Accessed 10 12 2024].

Eric conrad, j. f., 2014 . Cryptography. *Eleventh hour CISSP,* Issue 2013, pp. 77-93.

Fruhlinger, J., 2024. *CSO.* [Online]
Available at: https://www.csoonline.com/article/568917/the-cia-triad-definition-components-and-examples.html
[Accessed 16 01 2025].

geekforgeeks , 2024. *geekforgeeks.* [Online]
Available at: https://www.geeksforgeeks.org/caesar-cipher-in-cryptography/
[Accessed 10 12 2024].

Gillis, A. S., 2023. *techtarget.* [Online]
Available at: https://www.techtarget.com/whatis/definition/logic-gate-AND-OR-XOR-NOT-NAND-NOR-and-XNOR
[Accessed 06 01 2025].

Gupta, N., 2024. *SSl2buy.* [Online]
Available at: https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences
[Accessed 10 12 2024].

J.SImmons, G., 2024. *Britannica.* [Online]
Available at: https://www.britannica.com/topic/cryptology/History-of-cryptology
[Accessed 10 12 2024].

karan, R., 2023. *shikshaonline.* [Online]
Available at: https://www.shiksha.com/online-courses/articles/nor-gate/
[Accessed 06 01 2025].

okeke, F., 2022. *TechRepublic.* [Online]
Available at: https://www.techrepublic.com/article/asymmetric-vs-symmetric-encryption/
[Accessed 10 12 2024].

Rouse, M., 2014. *Techopedia.* [Online]
Available at: https://www.techopedia.com/definition/29767/rail-fence-cipher
[Accessed 10 12 2024].

Sara farrag, w. A., 2019. *researchgate.* [Online]
Available at: https://www.researchgate.net/figure/Encrypting-using-Rail-Fence-Cipher5_fig5_333480277
[Accessed 10 12 2024].

Schneider, J., 2024. *IBM.* [Online]
Available at: https://www.ibm.com/think/topics/cryptography-history
[Accessed 05 01 2024].

servos, W., 2010. *trincoll.* [Online]
Available at: https://www.cs.trincoll.edu/~crypto/historical/alberti.html
[Accessed 10 12 2024].

Siam, M. M. A., 2023. *Medium.* [Online]
Available at: https://mahfuzulsiam.medium.com/using-row-transposition-cipher-for-encryption-and-decryption-4a3c753b7a43
[Accessed 06 01 2025].

Sidhpurwala, H., 2023. *Red Hat.* [Online]
Available at: https://www.redhat.com/en/blog/brief-history-cryptography
[Accessed 10 12 2024].

Tyson, M., 2021. *infoworld.* [Online]
Available at: https://www.infoworld.com/article/2270982/a-quick-guide-to-modern-cryptography.html
[Accessed 10 12 2024].