Computers, Materials & Continua
DOI: 10.32604/cmc.2023.032882
Article

Tech Science Press

check for updates

An Efficient Technique to Prevent Data Misuse with Matrix Cipher Encryption Algorithms

Muhammad Nadeem[1], Ali Arshad[2,*], Saman Riaz[2], Syeda Wajiha Zahra[1], Ashit Kumar Dutta[3], Moteeb Al Moteri[4] and Sultan Almotairi[5]

[1]Department of Computing, Abasyn University, Islamabad, Pakistan
[2]Department of Computer Science, National University of Technology, Islamabad, Pakistan
[3]Department of Computer Science and Information Systems, College of Applied Sciences, AlMaarefa University, Riyadh, 13713, Kingdom of Saudi Arabia
[4]Department of Management Information Systems, Business Administration College, King Saud University, Riyadh, 11451, Saudi Arabia
[5]Department of Natural and Applied Sciences, Faculty of Community College, Majmaah University, Majmaah, 11952, Saudi Arabia
*Corresponding Author: Ali Arshad. Email: alli.arshad@gmail.com
Received: 01 June 2022; Accepted: 26 August 2022

Abstract: Many symmetric and asymmetric encryption algorithms have been developed in cloud computing to transmit data in a secure form. Cloud cryptography is a data encryption mechanism that consists of different steps and prevents the attacker from misusing the data. This paper has developed an efficient algorithm to protect the data from invaders and secure the data from misuse. If this algorithm is applied to the cloud network, the attacker will not be able to access the data. To encrypt the data, the values of the bytes have been obtained by converting the plain text to ASCII. A key has been generated using the Non-Deterministic Bit Generator (NRBG) mechanism, and the key is XNORed with plain text bits, and then Bit toggling has been implemented. After that, an efficient matrix cipher encryption algorithm has been developed, and this algorithm has been applied to this text. The capability of this algorithm is that with its help, a key has been obtained from the plain text, and only by using this key can the data be decrypted in the first steps. A plain text key will never be used for another plain text. The data has been secured by implementing different mechanisms in both stages, and after that, a ciphertext has been obtained. At the end of the article, the latest technique will be compared with different techniques. There will be a discussion on how the present technique is better than all the other techniques; then, the conclusion will be drawn based on comparative analysis.

Keywords: Symmetric; cryptography; ciphertext; encryption; decryption; cloud security; matrix cipher

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1 Introduction

Cloud computing is derived from two words, "*Cloud*" and "*Computing.*" Cloud means "*Internet,*" and computing means "*Calculating something.*" Cloud computing is an online network-based data storage system that combines different hardware and software and allows all users across the network to communicate online [1]. Cloud computing has gained much popularity over the years due to the collection of various servers, storage systems, databases, routers, switches, hardware, and software. Cloud computing provides many services to end-users, and all of these are available through the internet [2]. Many organizations are moving data to cloud servers to protect them from malware [3], and this data can be accessed from any part of the world. Many organizations offer free services to store data online, including the Amazon Web Service (A.W.S.), Google Cloud Platform, and Microsoft Azure. Sometimes. The network, front-end server, and back-end storage are all components of the cloud computing architecture [4]. Cloud computing is also called distributed system [5] because a distributed system is a system that stores data across multiple locations and servers, and that data can be accessed from anywhere. But as the demand for users increases in cloud computing similarly, the rate of cloud computing attacks is also growing [6].

Various attackers develop multiple algorithms and tools to misuse users' data and gain access to data that can be detrimental to users' data and cloud architecture. Two types of security are required to secure cloud data [7]; Hardware-based security and software-based security. Hardware-based security means protecting all the hardware and software connected to the cloud network and used for communication purposes. In comparison, software-based security means protecting all the applications that fetch data from the cloud server. Cloud servers and user data cannot be secured unless all hardware and software associated with cloud computing are protected [8].

Many organizations are developing different applications to secure cloud servers. Some organizations are working on various algorithms to ensure cloud computing from the outside, but the highest attack rate in cloud computing is from the inner side. When an attacker tries to attack from outside, various techniques can be used to avoid outside attacks [9]. These techniques include Intrusion Detection and Prevention systems (IDPs), firewalls, authentication, authorization, and biometrics. Multiple methods have been developed to prevent internal attacks in cloud computing, including Cloudflare, Anomaly-based Intrusion Detection System (AIDS), and Cryptography.

Cloudflare is one of the techniques used to prevent internal attacks, which is usually done to protect against phishing attacks such as DDoS (Distributed Denial of Service) and malicious bots [10]. Only authentic users and Internet Protocol (I.P) addresses can access cloud servers with this technique. Whenever an invalid user tries to perform malicious activity on the cloud server [11], it can be blocked by Cloudflare. Whenever an attacker tries to change the behavior of cloud data or cloud devices, such intruders can be detected by the Anomaly-Based Intrusion Detection system [12]. Still, the problem with Cloudflare and AIDS techniques is that these techniques can only see and block the attacks within the cloud server, but data cannot be secure through these techniques. Whenever an attacker develops a mechanism to break these techniques, he can easily access and use the data, which is the biggest problem.

The best way to keep data safe is to use cryptographic techniques [13]. Whenever the data is encrypted, the attacker will not be able to read the text, nor will he understand what is written in the text [14]. The most significant advantage of cloud cryptography is that cloud data cannot be accessed unless the exact mechanism is applied to this data as it is designed to convert it into a nonreadable form.

1.1 Cryptography

Cryptography is one of the best techniques to protect data from attackers [15]. Cryptography is an encryption technique in which the data is changed into a nonreadable format in different steps [16]. Nonreadable text is also called ciphertext. The ciphertext is the text that is difficult for any person to understand, whether the person is authentic or unauthentic. Ciphertext consists of symbols, characters, and numeric keys [17]. Whenever the text is converted to a ciphertext and sent to the receiver, it must be converted to a readable form on the receiver side called decryption [18]. In decryption, the ciphertext is converted to readable format using different steps. Whenever a data transmission occurs between the sender and the receiver, the attacker can use any technique to gain access to the data and transmit it to the receiver by altering it [19]. Data leakage and misuse are the most significant problems.

1.2 Encryption and Decryption

When data is converted from readable to nonreadable format to prevent misuse, this process is called Encryption. Various techniques are used to convert data to nonreadable format, which can be used to protect data from attackers. Whenever data is sent to the receiver side, it is necessary to bring this data in a readable format on the receiver side [20], for which different techniques are used, called decryptions.

1.3 Cryptography Algorithms

Cryptography has been categorized into Symmetric Encryption and Asymmetric Encryption [21]. Symmetric means "*same,*" whereas Asymmetric means "*different.*" Data encryption and decryption are performed using the same key in symmetric cryptography [22]. In Asymmetric cryptography, data is encrypted and decrypted using two different keys. The first key is public, and the second is private. A Symmetric cryptography key is also called a secret key [23], and whenever this data is decrypted, it is done using the same key used during Encryption.

1.4 Problem Formulation

Different researchers have developed various algorithms to secure the data. These algorithms generate a key from the key generator in randomized form and use this key to encrypt the data and then get the ciphertext, using the ASCII table, which is predefined. Whenever an attacker tries to decrypt the data, he always uses a predefined ASCII table that can help decrypt the data. Data transmission can be reliable if a predefined ASCII table is replaced by a Customized ASCII table that can only be used between sender and receiver, as done in this paper. In this paper, a matrix cipher encryption algorithm has been developed. A Customized ASCII table will be generated in which each character will be stored in the randomized form. Two keys will be created to store the data, and the procedure for obtaining each key will differ from the other key mechanism. The first key will be obtained from the NRBG, while the second key will be obtained from the plain text to be encrypted. Matrix cipher encryption algorithms will be used in two phases. In the first phase, Key-2 will be obtained by implementing a mechanism that can be used to decrypt only this data. Both sides will secure data in the second phase by implementing a random seed mechanism. The matrix cipher algorithm's most significant efficiency is achieving the second key. The second key (Key-2) is a key that will be unique to each text and can only be used up to that text, while the first key (Key-1) can be used for every text.

The paper has been distributed something like this. Previous work has been discussed in Section 2. The proposed algorithm is discussed in Section 3. The algorithm has been tested in Section 4. The latest work has been compared with previous work in Section 5, and Section 6 consists of Conclusions.

2 Literature Review

Ahmad et al. [10] developed the hill cipher algorithm. They applied the hill cipher algorithm results to the Strassen algorithm to identify the time complexity of the hill cipher algorithm. The text was first taken in an alphabetical form in the Hill chipper algorithm. This text was equalized with different numeric words with the help of a table, then generated a key, multiplied the numeric values with the key in matrix forms, and obtained the ciphertext by *mod26*. After that, they determined the time complexity by applying this ciphertext to the Strassen algorithm.

According to Al-Shabi [24], Cryptography is the art of converting data to a nonreadable format. It consists of various methods and techniques in which data has to be reversed from nonreadable form to readable format. This article developed a hill cipher algorithm to secure the data. The purpose of this algorithm was to create a technique that could be used to protect data. First, an image was taken as input to encrypt the image data, and, after that, the colors of the image were checked. If the image is color, it was moved to the grayscale and then took each component in pixel form, and each pixel was converted to a matrix. After converting to a matrix, the Hill cipher algorithm was applied and encrypted the data.

In the paper [25], the researchers developed a Triple Pass Protocol (T.P.P.) method and divided data into three phases for Encryption. In the first phase, plain text was taken and converted this plain text to ciphertext-1 with the help of the Square Matrix mechanism. Ciphertext-1 was represented with K_A. In the second phase, the Hill Cipher algorithm was implemented on cipher text-1 (K_A) and obtained ciphertext-2, described with K_B. All the values were inverted in the third step, and a ciphertext was obtained, described with A.B.

Sun et al. [22] modified the Hill cipher algorithm and developed a Hill cipher chain algorithm. This algorithm was implemented on the primary key text and obtained a ciphertext. The first plain text was converted to ASCII, then converted ASCII values to matrix form. After that, a random key was generated, multiplied the matrix with the key, and obtained different results in the matrix form. After that, these results were converted to ciphertext with the help of an ASCII table and stored in place of the primary key. The ciphertext was converted to matrix form to decrypt the data, and the keys were reversed and multiplied with the matrix. After that, plain text was obtained by replacing matrix values with their equivalent ASCII.

In this paper [26], the authors created a rectangular matrix key using the play fair cipher algorithm. The matrix key was created using the Playfair cipher algorithm. Afterward, keywords were created and arranged in the matrix; then, alphabets that do not exist in the keyword were stored in the matrix after the keyword. The ciphertext was then obtained using the Playfair cipher algorithm. The alphabets of this text were converted to their corresponding numerical values to form the rectangular matrix from those values. The pseudo-inverse of the matrix was then taken. The Rectangular matrix was taken as a key matrix for the Hill cipher algorithm, with the help of which encryptions were performed.

In Paper [27], researchers analyzed the various cryptographic algorithms used by cloud service providers and proposed an algorithm for providing security on client-side data. In this algorithm, data was converted to ASCII values for data encryption, and then these ASCII values were converted to binary. After conversion to binary, the bits were completed using 0, the last 4 bits were inverted, and the cipher text was generated. To decrypt the data, the ciphertext was converted to ASCII, the ASCII values were converted to binary, and the bits were completed. After completing the bits, the last 4 bits were inverted, and the binary values were converted to ASCII to obtain the plain text.

In this paper [28], various cryptographic papers were surveyed, and the cryptographic mechanisms proposed in these papers were compared and identified which mechanism is better in terms of security. Then it was discussed that R4 and R5 algorithms are such mathematical solutions that can make it difficult for an attacker to decrypt the data. Finally, it was concluded that the symmetric algorithm performs better than the asymmetric algorithm.

3 Proposed Algorithm

This article will develop a matrix cipher encryption algorithm that will help prevent data from being misused. When an attacker tries to access the data, using this technique will make it difficult for the attacker to access it. The attacker cannot access it to retrieve the original data.

3.1 Work Overflow

Whenever data is transmitted to the receiver side, the attacker makes every effort to gain access to that data and uses every possible algorithm that can be helpful to the attacker. An advanced matrix cipher encryption algorithm has been developed to solve this problem in which data can be saved from attackers using different techniques. The reason for creating a matrix cipher algorithm is that if an attacker attempts to misuse the data, the attacker will not be able to misuse the data due to the matrix cipher encryption algorithm. Only an authoritative person will have access to the data. First, plaintext will be taken to encrypt the data, and then plaintext will be converted to its equivalent ASCII. After that, a key will be generated from an NRBG, which will be XORed with ASCII. After that, a key will be generated from an NRBG, which will be XORed with plaintext ASCII. The bits toggling mechanism will be implemented on the result obtained from XOR, and the result obtained from the bits toggling will be converted to decimal.

After getting the decimal results, the 1C2R metric algorithm will be implemented on the decimal values, and a key will be generated. The key obtained from the 1C2R matrix will be used up to the same plaintext. A key from one plaintext cannot be substituted for a key from another plaintext. A random key will be used to provide security to the data on both sides, and the obtained result will be converted to ASCII, and the cipher text will be obtained. A complete encryption and decryption methodology is shown in Flow charts Figs. 1 and 2.

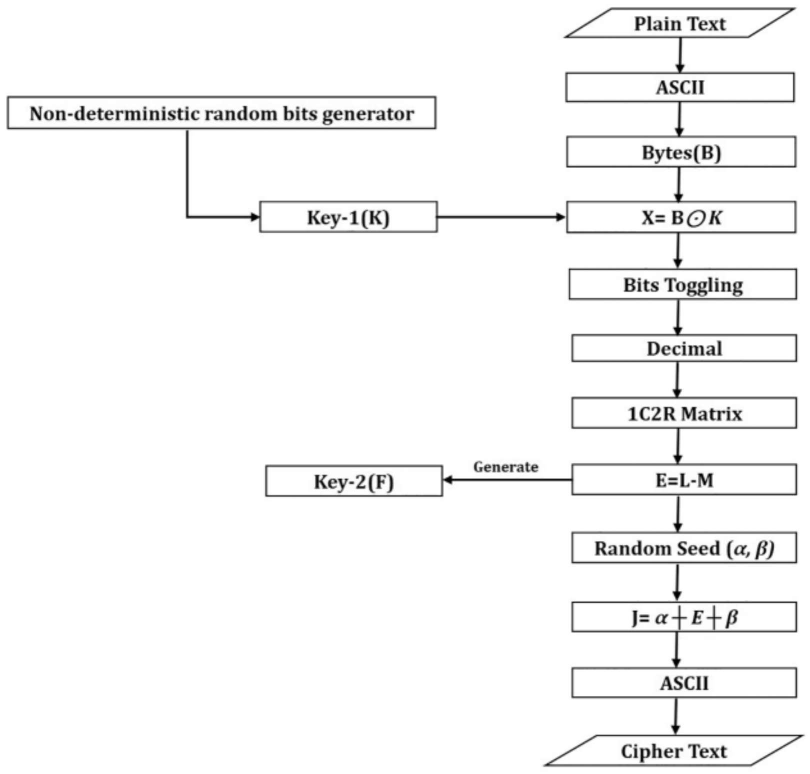


Figure 1: Encryption methodology

3.2 Data Encryption Process

Different steps will be used to convert plain text into ciphertext, in which the first plain text will be converted to its equivalent ASCII. Whenever an attacker tries to access the file, the attacker attempts to use the same ASCII table on the file, which is pre-created. This pre-created ASCII table will give him different data than the original data, which has nothing to do with actual data. In this paper, a customized ASCII table will be made, storing alphabets, symbols, and numeric keys in randomized form. The equivalent values of the predefined ASCII table and the customized ASCII table are different. A customized ASCII table is shown in Fig. 4. The plain text will be converted to the equivalent ASCII and then converted to bytes. A key will be generated using NRBG, represented as *key-1*. *Key-1* will be represented by "*K*," and "*Bytes*" will be represented by "*B*." "*X*" value will be obtained by XNORing the *Key-1* (K) and *Bytes* (B). Bits of all values derived from XNOR will be toggled so that if an attacker uses decryption mechanisms, he will not be able to understand which phase has been toggled. After toggling the bits, each byte will be converted to decimals, as shown in Fig. 3.

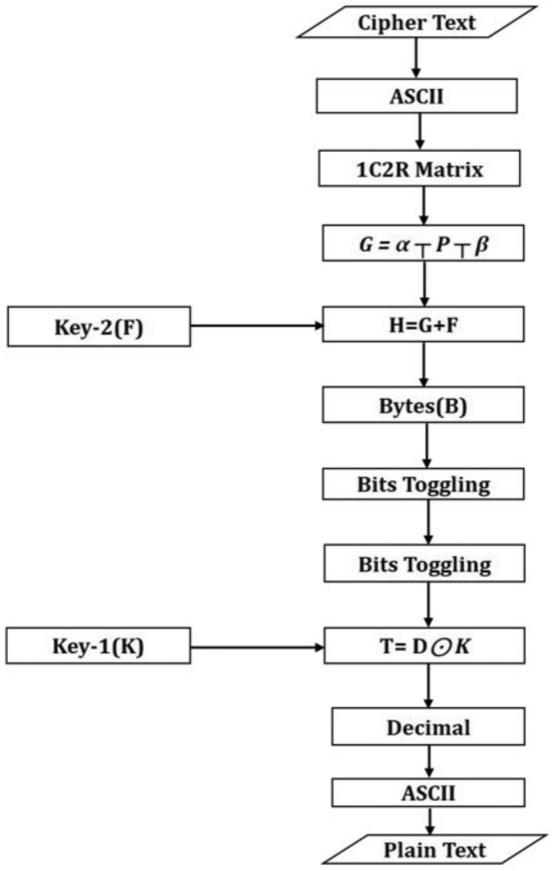


Figure 2: Decryption methodology

Mathematical Notation:

$+$ = Insertion

$\top$ = Elimination

$\odot$ = XNOR

$\bar{\bar{z}}$ = Bits inversion

α = First random seed

β = Second random seed

An efficientalgorithm, "Matrix Cipher Encryption Algorithm," has since been developed, and that decimal value will be implemented in this algorithm. The matrix cipher encryption algorithm is divided into two phases. The first phase is *1C2R-Matrix*, while the second phase is the Random Seed phase. When the decimal values are obtained, these decimal values will be paired and converted to *1C2R-matrix* form. The bits toggle values will be denoted by "*L,*" and the *key-1 (K)* will be represented by "*M.*" Matrix "*L*" and matrix "*M*" will be applied to the formula "$E = L\text{-}M$" and "*E*" value will be obtained. The second phase will generate two random seed pairs in matrix form. The first random seed pair has been denoted by "α."

In contrast, the second random seed pair has been represented by "β." The random seed value "α." will be inserted at the beginning of "E." In contrast, the seed badge "β" will be inserted at the end of "E." The formula "$J = \alpha + E + \beta$" will be used for random seed insertion, and the value of "J" will be obtained. The symbol "+" means data insertion. The value "J" obtained from matrix cipher encryptions will be converted to ASCII and obtained ciphertext, as shown in Fig. 3.

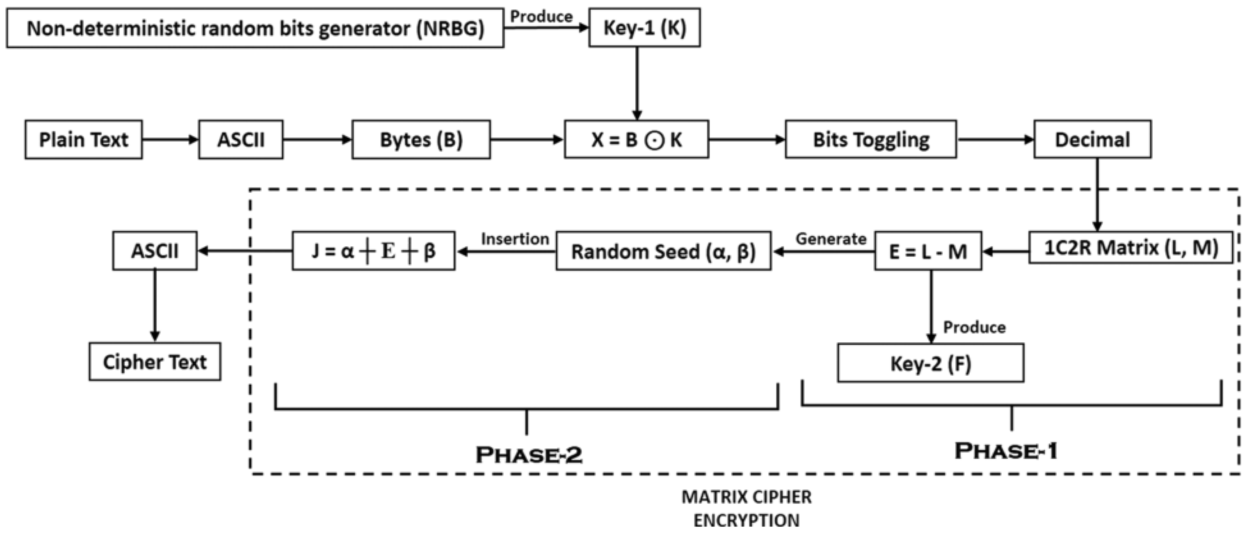


Figure 3: Encryption algorithms

3.2.1 *Non-Deterministic Random Bit Generator (NRBG)*

NRBG is a random bit generator. With the help of which different bits will be generated in randomized form according to converted bits from ASCII. Bits obtained from *NRBG* will be considered as *Key-1*. These bits will be used in two phases. These bits will be XNOR with *ASCII* converted bits in the first phase. *NRBG* bits will be converted to decimal in the second phase and then converted to matrix form. After that, the bits will be subtracted with toggled decimal values.

3.2.2 *Bits Toggling*

Bits toggling means inverting the bits. Whenever bits are inverted, the value of bits is changed from 0 to 1 and from 1 to 0, called bits toggling.

3.2.3 *Matrix*

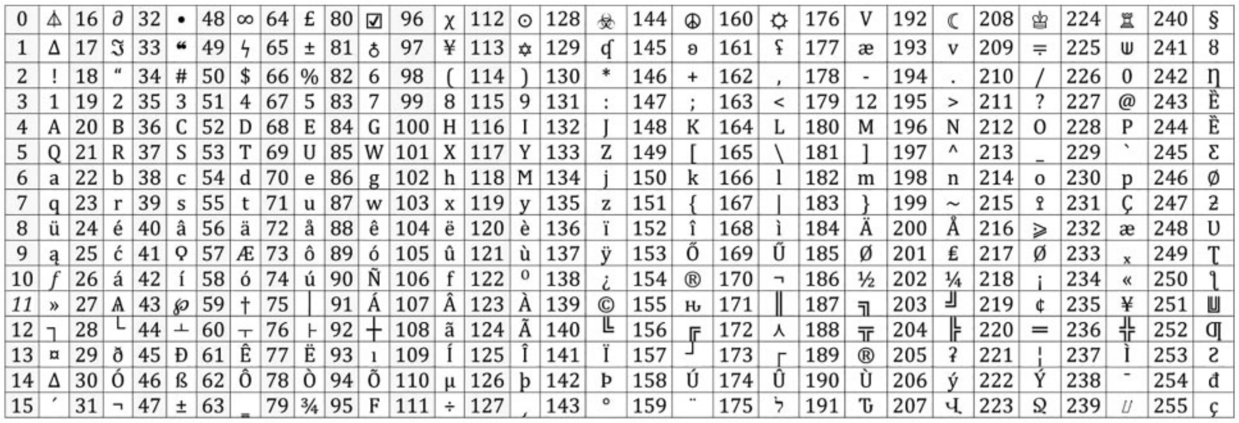
A matrix is a rectangular array in which numbers or expressions are stored in a fixed form. The matrix consists of rows and columns. In the matrix, whenever the values move from the left to the right (vertical), it is called rows, while when the values move from the top to the bottom (horizontal), it is called a column. There are several matrix types, including the Common rows matrix, column matrix, and rectangular matrix. This paperwork is on One-column-two-rows-matrix (1C2R). *1C2R-Matrix* will store two values in one column.

3.2.4 Random Seed

A random seed has been derived from two words. One is "*Random,*" and the other is "*Seed.*" Random is also called pseudorandom. Seed is a technique that is used to generate numbers. A random seed is a starting point from which a random number is started. In this paper, a random seed is used to initialize the random number as desired by the user, the random number starts from where the user wants. In this paper, random seeds are carried in matrix form.

3.2.5 Customized ASCII

Customized ASCII means creating an updated ASCII by modifying the existing ASCII. Whenever an attacker receives data in encrypted form, he tries to implement a predefined ASCII table on the malicious algorithms, which can be a way for one type of attacker to decrypt the data. If the attacker understands the method of encrypting the data, he can quickly create an algorithm to decrypt the data. Still, unless the attacker has the customized ASCII table, as developed in this article, the attacker cannot get the actual data according to this ASCII. If the attacker uses a predefined ASCII table, he will get the values according to the predefined ASCII table. That value will not be equalized to the actual ASCII used in this paper. The customized ASCII table is shown in Fig. 4. If the proposed algorithm is applied to a standard ASCII table, then, in this case, the data can be protected from man-in-the-middle attacks, but the problem with using a standard ASCII table is that whenever an attacker Attempts to decrypt the data, the attacker's first attempt is to use the standard table. Suppose a custom ASCII table is used instead of a standard ASCII table. In that case, it will be difficult for an attacker to create such an ASCII table, which is why a custom ASCII table can provide more security than a standard ASCII table.



0	⚠	16	∂	32	•	48	∞	64	£	80	☑	96	χ	112	⊙	128	☣	144	☮	160	☼	176	V	192	☾	208	♔	224	♜	240	§
1	Δ	17	ℑ	33	❝	49	ϟ	65	±	81	♁	97	¥	113	✡	129	ɖ	145	ɘ	161	ʕ	177	æ	193	v	209	≑	225	ɯ	241	8
2	!	18	"	34	#	50	$	66	%	82	6	98	(	114	)	130	*	146	+	162	,	178	-	194	.	210	/	226	0	242	ƞ
3	1	19	2	35	3	51	4	67	5	83	7	99	8	115	9	131	:	147	;	163	<	179	12	195	>	211	?	227	@	243	Ȅ
4	A	20	B	36	C	52	D	68	E	84	G	100	H	116	I	132	J	148	K	164	L	180	M	196	N	212	O	228	P	244	Ȇ
5	Q	21	R	37	S	53	T	69	U	85	W	101	X	117	Y	133	Z	149	[	165	\	181	]	197	^	213	_	229	`	245	Ɛ
6	a	22	b	38	c	54	d	70	e	86	g	102	h	118	M	134	j	150	k	166	l	182	m	198	n	214	o	230	p	246	ø
7	q	23	r	39	s	55	t	71	u	87	w	103	x	119	y	135	z	151	{	167	|	183	}	199	~	215	♀	231	Ç	247	ƻ
8	ü	24	é	40	â	56	ä	72	å	88	ê	104	ë	120	è	136	ï	152	î	168	ì	184	Ä	200	Å	216	⩾	232	æ	248	Ʊ
9	ą	25	ć	41	Ϙ	57	Æ	73	ô	89	ó	105	û	121	ù	137	ÿ	153	Ő	169	Ű	185	Ø	201	₤	217	Ø	233	ₓ	249	Ʈ
10	ƒ	26	á	42	í	58	ó	74	ú	90	Ñ	106	f	122	⁰	138	¿	154	®	170	¬	186	½	202	¼	218	¡	234	«	250	ʅ
11	»	27	₳	43	℘	59	†	75	│	91	Á	107	Â	123	À	139	©	155	Hь	171	‖	187	╗	203	╝	219	¢	235	¥	251	Ш
12	┐	28	└	44	┴	60	┬	76	├	92	┼	108	ã	124	Ã	140	╚	156	╔	172	ʎ	188	╦	204	╠	220	═	236	╬	252	ɋ
13	¤	29	ð	45	Đ	61	Ê	77	Ë	93	ı	109	Í	125	Î	141	Ï	157	┘	173	┌	189	®	205	ʡ	221	¦	237	Ì	253	ƨ
14	Δ	30	Ó	46	ß	62	Ô	78	Ò	94	Õ	110	µ	126	þ	142	Þ	158	Ú	174	Û	190	Ù	206	ý	222	Ý	238	¯	254	đ
15	´	31	¬	47	±	63	‗	79	¾	95	F	111	÷	127	¸	143	°	159	¨	175	ʖ	191	Ꚍ	207	Ч	223	Ω	239	//	255	ç

Figure 4: Customized ASCII table

3.3 Data Decryption Process

First, the ciphertext will be converted to its equivalent *ASCII*, then *ASCII* values will be converted to *1CR2-matrix* form. Of the values derived from *1C2R*, the values of "α" and "β" will be removed with the help of "$G = \alpha \top E \top \beta$" to obtain the value of "*G*." Symbol "$\top$" means remove the values. After obtaining the value of "*G*," matrix values in the decimal form will be obtained by adding a

"*Key-2*" *(F)* and "*G*" using the formula "$H = G + F$," and each value will be converted to *Bytes*. Bits toggling mechanism will be used after bytes conversion, and then the decimal value will be obtained by XNORing *Key-1 (K)* and bits toggled value *(D)*. When the decimal value is obtained, each decimal value will be replaced by its equivalent *ASCII* and will get the plain text. The decryption process is shown in Fig. 5.

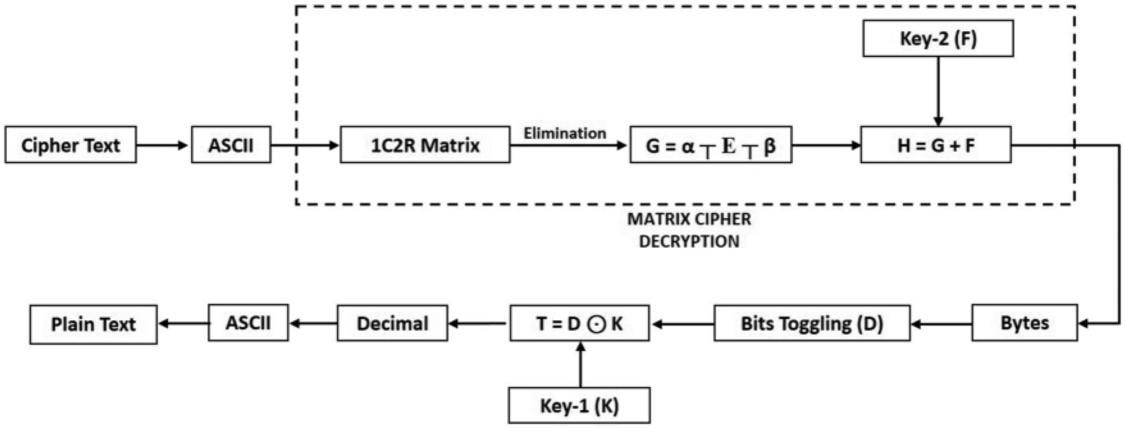


Figure 5: Decryption algorithm

First, a plan text will be taken, and this plan text will be converted to its equivalent *ASCII* then each *ASCII* value will be converted to its bytes. After that, a *Key* will be generated using the NRBG mechanism and *XNOR* this Key with *Bytes* by using the "$X = B \odot K$" formula. Where "X" is the result, "K" is the key, and "X" is bytes generated by the decimal. Then, the bits toggling procedure will be applied to "X" and "$\overline{X}$" will be obtained. T*oggle bytes values* and *key values* will be converted to *decimal* format. *Toggle bytes* $\left(\overline{X}\right)$ will represent with "L" whereas *Key (K)* will be represented by "M". After that, "L" and "M" pairs will be made. If any decimal value remains left single, then make a pair of remaining values with "∅." Now, the *IC2R-matrix mechanism* will be applied to the decimal values "L" and "M," and then the formula "$E = L\text{-}M$" will be applied to it. If the index value of the first matrix *(L)* is less than the second matrix *(M)*, replace that index value of the second matrix with the first matrix value. After this, two values will be obtained, "E" is the result obtained from "L" and "M," and "F" is the *key-2*, obtained after replacing the second matrix with the first matrix value. After that, two pairs of random seed matrices (α, β) will be generated. "α" will insert at the start of "E" and "β" will insert at the end of "E" by using "$J = \alpha + E + \beta$" formula. The "J" matrices will be obtained from step 8. Each metric value will be converted to its ASCII equivalent, and a ciphertext will be generated.

4 Testing

First, a plain text is taken to test the data, and then this plain text is encrypted with the help of a matrix cipher algorithm, and an unreadable text is obtained, called ciphertext. After receiving the ciphertext, various decryption steps have been followed to convert the text from nonreadable format to readable.

Algorithm 1: Encryption

Input: Plain text
Output: Ciphertext

```
1. Create a plain text
2. Convert the plain text to its ASCII values.
3. Convert each ASCII value into bytes $(B)$.
4. Generate a Key (K) using NRBG.
5. XNOR bytes $(B)$ with a generated key $(K)$ using the formula "$X = B \odot K$."
6. Apply bits toggling on step 5.
7. Convert toggle bytes "$\overline{X}$" and Key $(K)$ into decimal format $(L, M)$.
8. Make pairs of "$L$" and "$M$."
9. Convert the values of $(L)$ and Key $(M)$ into 1C2R-matrix form.
10. Apply $E = L–M$ on step 9.
11. Generate two pairs of random seed matrices $(\alpha, \beta)$.
12. Insert the "$\alpha$" matrix at the start of "$E$" and the "$\beta$" matrix at the end of "$E$" by using "$J = \boldsymbol{\alpha} + E + \beta$" formula.
13. Convert each matrix value to equivalent ASCII
14. Generated ciphertext.
```

Algorithm 2: Decryption

Input: Ciphertext
Output: Plain text

```
1. Ciphertext
2. Convert each cipher value to its equivalent ASCII.
3. Generate 1C2R matrix $(P)$ from step-2 ASCII.
4. Eliminate the value of "$\alpha$" and "$\beta$" from "$P$" by using $G = \alpha \top P \top \beta$, where "$\top$" is the elimination.
5. Add the value of "$G$" with the value of Key-2 $(F)$, obtained at the time of Encryption.
   $H = G + F$
6. Convert each value of "$H$" to binary.
7. Apply toggling on bits obtained in step-6.
8. XNOR binary values of step-6 with key-1 $(K)$.
9. Convert binary to decimal.
10. Convert each decimal value to its equivalent ASCII and obtain plain text.
```

4.1 Encryption Algorithm

Step 1: First, *plain text* has been taken, as shown in Fig. 6, and this plain text has been encrypted.

N@deem

Figure 6: Plain text

Step 2: Each *plain text character* has been converted to its equivalent *ASCII*, as shown in Fig. 7.

N = 78, @ = 64, d = 100, e = 101, e = 101, m = 109

Figure 7: ASCII of each character

Step 3: After obtaining the *ASCII* of each character, each *ASCII* value has been converted to *Bytes*, and *Bytes* are represented with "*B,*" as shown in Fig. 8.

B =	78 = 01001110,	64 = 01000000	100 = 01100100
	101 = 01100101	101 = 01100101	109 = 01101101

Figure 8: Byte of each ASCII value

Step 4: When all the values have been converted to *Bytes*, a *Random key* will be generated according to the length of "*B.*" NRBG is used to generate bits. The values obtained from the NRBG are named *Key-1*," as shown in Fig. 9.

K = 11010011 00111001 10011101 01110110 11011001 01101000

Figure 9: Generate key from NRBG

Step 5: After attaining the random bits from the Non-deterministic Random Bit Generator, these *bits* have been XNORed with Step-3 *bytes (B)*, as shown in Fig. 10. The "$X = B \odot K$" formula has been used for *XNOR*. "*K*" is *Key-1* obtained from *NRBG* in step-4, while "*B*" is the value obtained in *step-3*, in which each *ASCII* has been converted into *Bytes*.

$$\begin{array}{r} 01001110\ 01000000\ 01100100\ 01100101\ 01100101\ 01101101 \\ B \odot K = 11010011\ 00111001\ 10011101\ 01110110\ 11011001\ 01101000 \\ \hline X = 01100010\ 10000110\ 00000110\ 11101100\ 01000011\ 11111010 \end{array}$$

Figure 10: XNOR result

Step 6: When *Bytes (B)* will XNORed with *Key-1 (K)*, a value will be obtained, which is represented as "*X.*" Bits toggling has been then implemented on the XNOR result "*X,*" as shown in Fig. 11. In bits toggling, each Bit has been inverted.

$\bar{X}$ = 10011101 01111001 11111001 00010011 10111100 00000101

Figure 11: Bit Toggling on XNOR result

Step 7: The values of step-6 Bits toggling result in"$\overline{X}$" and *step-4 Key-1* result "*K*" has been converted to decimal format. After converting to decimal, the result of *Key-1 (K)* has been represented with "*M,*" as shown in Fig. 12. In contrast, Bits toggling."$\overline{X}$" has been represented with "*L*" after converting to decimal, as shown in Fig. 13.

$$\begin{array}{cccccccc} K = & 11010011 & 00111001 & 10011101 & 01110110 & 11011001 & 01101000 \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ M = & 211 & 57 & 157 & 118 & 217 & 104 \end{array}$$

Figure 12: Decimal conversion result of Key-1

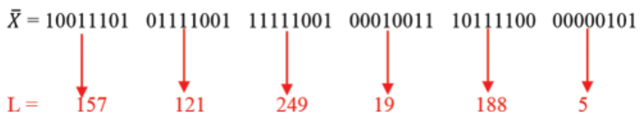

$\bar{X}$ = 10011101 01111001 11111001 00010011 10111100 00000101
L = 157 121 249 19 188 5

Figure 13: Decimal conversion result of Bit toggling

Step 8: After receiving decimal results, step-7 "*L*" and "*M*" has been converted to pairs, as shown in Figs. 14 and 15.

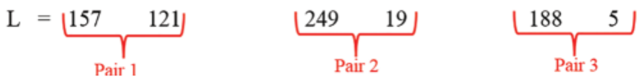


Figure 14: Pairs of "L"

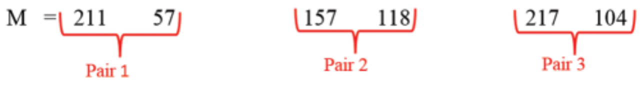


Figure 15: Pairs of "M"

If the last digit is left single, that digit will be paired with "Ø" and the pair will be completed.

Step 9: After making pairs, each pair has been converted into matrix form, as shown in Fig. 16.

$$L = \begin{bmatrix}157\\121\end{bmatrix}\begin{bmatrix}249\\19\end{bmatrix}\begin{bmatrix}188\\5\end{bmatrix}$$

$$M = \begin{bmatrix}211\\57\end{bmatrix}\begin{bmatrix}157\\118\end{bmatrix}\begin{bmatrix}217\\104\end{bmatrix}$$

Figure 16: Matrix conversion results

Step 10: Applied "*E* = *L*–*M*" on step-9, as shown in Fig. 17.

$$E = L - M\left[\begin{bmatrix}157\\121\end{bmatrix} - \begin{bmatrix}157\\57\end{bmatrix}\right]\left[\begin{bmatrix}249\\19\end{bmatrix} - \begin{bmatrix}157\\19\end{bmatrix}\right]\left[\begin{bmatrix}188\\5\end{bmatrix} - \begin{bmatrix}188\\5\end{bmatrix}\right]$$

Figure 17: Matrix formula

If the value of the first matrix is less than the second matrix, then replace the second matrix value with the first matrix, as shown in Fig. 18.

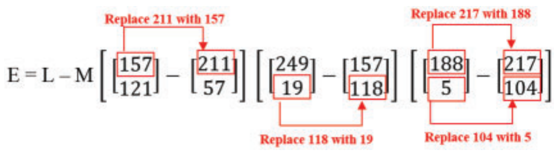

Replace 211 with 157
Replace 217 with 188
E = L − M [[157 121] − [211 57]] [[249 19] − [157 118]] [[188 5] − [217 104]]
Replace 118 with 19
Replace 104 with 5

Figure 18: Replacement of second index value with the first index

New results have been obtained after replacing the second index's greatest value with the smallest value, as shown in Fig. 19.

$$E = L - M\left[\begin{bmatrix}157\\121\end{bmatrix} - \begin{bmatrix}157\\57\end{bmatrix}\right]\left[\begin{bmatrix}249\\19\end{bmatrix} - \begin{bmatrix}157\\19\end{bmatrix}\right]\left[\begin{bmatrix}188\\5\end{bmatrix} - \begin{bmatrix}188\\5\end{bmatrix}\right]$$

Figure 19: Values replacement results

After replacing the values, "*M*" index values have been used as *Key-2* in decryption. The *Key-2* value has been represented with "*F,*" as shown in Fig. 20. The difference between *Key-1* and *Key-2* is that *Key-1* is randomized bits obtained from *NRBG* while *Key-2* is the key generated in matrix cipher phase-1. Even if the attacker tries to decrypt the data through the master key, it will be difficult for the attacker to use the key generated from the text.

$$F = \begin{bmatrix}157\\57\end{bmatrix}\begin{bmatrix}157\\19\end{bmatrix}\begin{bmatrix}188\\5\end{bmatrix}$$

Figure 20: Generated key-2

After that, by using the "$E = L–M$" formula, the value of "*E*" is obtained, as shown in Fig. 21.

$$E = L\text{-}\ M\begin{bmatrix}0\\64\end{bmatrix}\begin{bmatrix}92\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}$$

Figure 21: Matrix cipher phase-1 result

"*E*" is the result that has been obtained from "$E = L–M$," whereas "*F*" is the latest key that has been obtained from phase-1 of the matrix cipher encryption algorithm.

Step 11: After generating *Key-2* and obtaining the result of "*E,*" Two random seed pairs of the matrix have been developed, as shown in Fig. 22, so that "E" text can be provided left and right-side security.

$$\text{First pair of Random Seed matrix} = \alpha = \begin{bmatrix}151\\62\end{bmatrix}\begin{bmatrix}67\\217\end{bmatrix}$$

$$\text{Second pair of Random Seed matrix} = \beta = \begin{bmatrix}96\\128\end{bmatrix}\begin{bmatrix}121\\167\end{bmatrix}$$

Figure 22: Matrix cipher phase-2 result

Step 12: The first pair of Random Seed matrix "α" has been inserted at the start, and the second pair of Random Seed matrix "β" has been inserted at the end of "*E*" by using this equation.

$$J = \alpha + E + \beta$$

"*J*" is the result, which has been obtained after the insertion of "α" and "β," as shown in Fig. 23, while the "+" symbol represents the insertion.

$$J = \begin{bmatrix}151\\62\end{bmatrix}\begin{bmatrix}67\\217\end{bmatrix}\begin{bmatrix}0\\64\end{bmatrix}\begin{bmatrix}91\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}96\\128\end{bmatrix}\begin{bmatrix}121\\167\end{bmatrix}$$

Figure 23: Matrix cipher phase-2 result

Step 13: Each matrix value has been converted to its equivalent *ASCII* using Fig. 4. The *ASCII* value of each matrix is shown in Fig. 24.

151 = {	62 = Ô	67 = 5	217 = Ø	0 = ⍋	64 = £	92 = ┼
0 = ⍋	0 = ⍋	0 = ⍋	96 = χ	128 = ☣	121 = ù	167 = |

Figure 24: ASCII values of each matrix

Step 14: Generated the ciphertext as shown in Fig. 25.

{ Ô 5 Ø ⍋ ┼ ⍋ ⍋ ⍋ χ ☣ ù |

Figure 25: Ciphertext

4.2 Decryption Algorithm

Different steps have been used to convert data from nonreadable format to readable format.

Step 1: A ciphertext has been taken first, as shown in Fig. 26.

{ Ô 5 Ø ⍋ ┼ ⍋ ⍋ ⍋ χ ☣ ù |

Figure 26: Ciphertext

Step 2: Each ciphertext value has been converted to its equivalent *ASCII*, as shown in Fig. 27.

151 = {	62 = Ô	67 = 5	217 = Ø	0 = ⍋	64 = £	92 = ┼
0 = ⍋	0 = ⍋	0 = ⍋	96 = χ	128 = ☣	121 = ù	167 = |

Figure 27: ASCII values of each matrix

Step 3: After converting each cipher value to ASCII, each value has been converted into *One-column-two-rows-matrix (1C2R)* form, as shown in Fig. 28. The matrix derived from *1C2R* has been represented with *"P."*

$$P = \begin{bmatrix}151\\62\end{bmatrix}\begin{bmatrix}67\\217\end{bmatrix}\begin{bmatrix}0\\64\end{bmatrix}\begin{bmatrix}91\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\begin{bmatrix}96\\128\end{bmatrix}\begin{bmatrix}121\\167\end{bmatrix}$$

Figure 28: 1C2R matrix result

Step 4: After converting to *One-column-two-rows-matrix (1C2R), "α"* and *"β"* will be eliminated using the following equation.

$G = \alpha \top E \top \beta$

"α" is the value that has been inserted at the start of "*E,*" while "*β*" is the value that has been inserted at the end of "*E.*" Symbol "⊤" means to eliminate values. Elimination values of "*α*" and "*β*" have been shown in Fig. 29.

$$P = \underbrace{\begin{bmatrix}151\\62\end{bmatrix}\begin{bmatrix}67\\217\end{bmatrix}}_{\text{“}\alpha\text{”}}\begin{bmatrix}0\\64\end{bmatrix}\begin{bmatrix}91\\0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}\underbrace{\begin{bmatrix}96\\128\end{bmatrix}\begin{bmatrix}121\\167\end{bmatrix}}_{\text{“}\beta\text{”}}$$

Figure 29: Elimination of "*α*" and "*β*" values

After eliminating values, different results have been obtained, as shown in Fig. 30.

$$\mathrm{G} = \begin{bmatrix} 0 \\ 64 \end{bmatrix} \begin{bmatrix} 91 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Figure 30: Elimination results of “α” and “β”

Step 5: After eliminating “α,” and “β,” the result “G” has been added with the *Key-2* key using the equation “$H = G + F$.” *Key-2* is the key obtained from plain text in Encryption.

The addition results of matrix “H” are shown in Figs. 31 and 32.

$$\mathrm{H} = \mathrm{G} + \mathrm{F} \left[\begin{bmatrix} 0 \\ 64 \end{bmatrix} + \begin{bmatrix} 157 \\ 57 \end{bmatrix} \right] \left[\begin{bmatrix} 92 \\ 0 \end{bmatrix} + \begin{bmatrix} 157 \\ 19 \end{bmatrix} \right] \left[\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 188 \\ 5 \end{bmatrix} \right]$$

Figure 31: Addition of key-2 and “G”

$$\mathrm{H} = \begin{bmatrix} 157 \\ 121 \end{bmatrix} \begin{bmatrix} 249 \\ 19 \end{bmatrix} \begin{bmatrix} 188 \\ 5 \end{bmatrix}$$

Figure 32: Matrix addition results

Step 6: Each value of “H” has been converted to *binary*. The converted results of each value are shown in Fig. 33.

$$\mathrm{H} = \begin{array}{lll} 157 = 10011101, & 121 = 01111001 & 249 = 11111001 \\ 19 = 00010011 & 188 = 10111100 & 5 = 00000101 \end{array}$$

Figure 33: Conversion of each of the values in binary form

Step 7: Apply toggling on bits obtained in step-6. *Bits toggling* result is shown in Fig. 34, and the result obtained from bits toggling has been represented with “D.”

$$\mathrm{D} = 01100010\ \ 10000110\ \ 00000110\ \ 11101100\ \ 01000011\ \ 11111010$$

Figure 34: Conversion of each of the values into binary form

Step 8: After bits toggling step-6, result “D” has been XNORed with *Key-1 (K),* as shown in Fig. 35. *Key-1* is the key obtained from *NRGB* in Encryption.

$$\begin{array}{rl} & 01100010\ 10000110\ 00000110\ 11101100\ 01000011\ 11111010 \\ D \odot K = & 11010011\ 00111001\ 10011101\ 01110110\ 11011001\ 01101000 \\ \hline \mathrm{X} = & 01001110\ 01000000\ 01100100\ 01100101\ 01100101\ 01101101 \end{array}$$

Figure 35: XNOR result

Step 9: Each binary of “X” has been converted to its decimal. The conversion result is shown in Fig. 36.

$$\mathrm{D} = \begin{array}{lll} 01001110 = 78 & 01000000 = 64 & 01100100 = 100 \\ 01100101 = 101 & 01100101 = 101 & 01101101 = 109 \end{array}$$

Figure 36: Binary to decimal conversion results

Step 10: Each decimal value has been converted to its equivalent *ASCII* and obtained the plain text. The plain text is shown in Fig. 37.

78 = N 64 = @ 100 = d 101 = e 101 = e 109 = m

Figure 37: Plain text

5 Comparative Analysis

Researchers have surveyed various cryptographic mechanisms and developed new mechanisms using existing ones. Some researchers modified the Hill cipher algorithm and introduced a new algorithm by adding some new features. Some researchers provided encryption through triple-time encryption. Some researchers tried to provide security only on client-side data. None of the papers developed any algorithm that would provide both sides security for cipher data, nor has any technique been used in any paper that would make data decryption impossible for an attacker. Data in cloud computing cannot be secure unless efficient encryption techniques are implemented on that data. It will be effortless for an attacker to break a single technique, but when a different technique is implemented in each phase, then data leakage or data interruption will be impossible for the attacker, as has been done in this paper. A complete comparative analysis is shown in Table 1.

Table 1: Comparative analysis

Sr#	1	2	3	4	5	Proposed work
Year	2020	2021	2021	2022	2021	
Reference	[10]	[24]	[25]	[22]	[27]	
Proposed algorithm	Advanced hill cipher algorithm	Advanced hill cipher algorithm	Using of hill cipher with triple pass protocol	Hill cipher chain algorithm	Client-Side cryptography algorithm	Matrix cipher encryption algorithm
ASCII Table	Standard	✗	Standard	Standard	Standard	Customized
Encryption key	Random key	✗	✗	Random Key	✗	Random Key
Decryption key	Random key	✗	✗	Random Key	✗	Plaintext generated key, random key
Additional security mechanism	✗	Convert image pixels to matrix	✗	✗	✗	Random seed (α β)

(Continued)

Table 1: Continued

Sr#	1	2	3	4	5	Proposed work
Novelty	Identification of Hill Cipher algorithm time complexity	Encrypted grayscale image	Three phases encryption	Primary key encryption		Customized ASCII, Generated key by text, Generate Key by NRBG, Random seed
Research gaps	Used one random key for decryption	Encryption done by single formula	Random seed can provide better security instead of bits reversing.	Suitable for encrypting the primary key number	Used predefined techniques to provide data security.	Identified all existing gaps
Proposed paper solution	Used two different keys for decryption.	Encryption done by different formulas	Random seed is used to provide security on both sides of cipher text.	Suitable for all types of text	Developed 1C2R matrix technique	Developed an efficient algorithm that can provide ciphertext security on both sides.

In 2020 [10], researchers developed a hill cipher algorithm, in which they multiplied the plaintext by a key equal to different values and obtained a ciphertext. Then, implemented the Strassen algorithm on it to identify the time complexity. In this paper, encryption was performed using a static key. If an attacker gets the static key through any algorithm, the attacker can obtain all the data that is encrypted with that key, which is the drawback of this paper.

In 2021 [24], to encrypt data, an image was taken and converted to grayscale, then converted to pixels, and then the pixels were converted to a matrix. After that, a hill cipher formula was applied, and data encryption was performed. This paper does not use any technique or key that could provide additional security to the data. If the attacker gains access to this formula through some mechanism, he can easily decrypt the data, which is the drawback of this paper.

In 2021 [25], researchers developed a three-phase triple-pass protocol that encrypts plaintext. In the first phase, plaintext was converted to ciphertext-1 with the help of the Square Matrix mechanism. In the second phase, ciphertext-1 was converted to ciphertext-2 with the help of the hill cipher algorithm, and in the third phase, ciphertext A.B. was obtained by inverting all the values through a mechanism. This paper's algorithm did not use any technique that could further prevent data from being misused. Data can be further preserved if a random seed is used instead of inverting the values in phase 3, which is the drawback of this paper.

In 2022 [22], researchers modified the Hill cipher algorithm, named the Hill cipher chain algorithm. The main purpose of which was to encrypt the primary key. Various steps were taken to encrypt the primary key, and a ciphertext was obtained. The problem with this paper is that this algorithm is only suitable for encrypting keys of numbers.

In the paper [27], The researcher developed an algorithm to secure the data on the client side using various techniques such as ASCII conversion, encryption by static key and bits conversion. The last four bits are inverted to protect the data further, and a ciphertext was obtained. The problem with this paper is that this algorithm did not develop any state-of-the-art techniques and only used techniques that already existed. Instead of using existing techniques, developing advanced techniques can further protect data.

6 Novelty of Proposed Work

The working method of this paper is different from all other papers. In various papers, researchers have modified the hill cipher algorithm, multiplicated the matrix form with the help of keys, used standard ASCII tables and implemented algorithms equalizing plain text with different values, which is not enough to encrypt the data completely. This paper uses techniques that can be decrypted only if the algorithm is implemented exactly as it is for each phase. Whenever cloud data is encrypted, it should be encrypted in different stages so that if an attacker decrypt one of the techniques, the second phase technique cannot be decrypted. Various researchers have worked on predefined ASCII tables to obtain cipher values unsuitable for data security. A customized ASCII table should be generated to solve this problem, as has been done in this paper. The matrix cipher encryption algorithm has been used to secure the data, developing a key created from the exact plain text that could prevent data from being misused.

7 Conclusion

After comparing the encryption techniques and algorithms used in different papers, it is concluded that the Matrix Cipher Encryption Algorithm is an efficient and secure algorithm that encrypts data using various methods at different stages. Whenever an attacker tries to break this algorithm, the data cannot be misused due to using different techniques in different phases. The security of each phase of this paper is very different from all other phases. To break each phase, it is essential to use the same key used during Encryption, and the exact mechanism must be used, which is developed for decrypting the data. This paper develops a hill matrix algorithm to encrypt the data. In this algorithm, Security is provided in two phases. In the first step, a key is generated, which can be used in the same plaintext from which the key is generated. A key of one text cannot work in another text, which is the novelty of this paper. The second phase of the matrix cipher algorithm uses a random seed mechanism that provides left-right security, making it difficult for an attacker to decipher the plaintext. Cloud server data can be protected entirely from attackers if the Hill Matrix algorithm is used while encrypting the data. When all the techniques used in Hill Matrix Algorithm are used effectively, it does not matter how many attacks an attacker attacks the cloud server because he can never access the original data, which can further reduce the chances of attack.

In the future, the latest Hill cipher algorithm will be modified, and different techniques will be implemented on this algorithm to secure all kinds of cloud server data, whatever the format of data is. The papers will then be compared in detail with the different articles and will be discussed to see how the present technique is better than all the other techniques. After that, an algorithm will be developed to test the performance of the Hill cipher algorithm.

Acknowledgement: The authors deeply acknowledge the Researchers supporting program (TUMA-Project-2021-27) Almaarefa University, Riyadh, Saudi Arabia for supporting steps of this work.

Author Contributions: M.N. Conceptualization; A.A. Software, Writing—review & editing; S.R Review & editing; S.W.Z Methodology; A.K.D Editing; M.A.M Review; S.A. funding acquisition.

Funding Statement: This research was supported by the Researchers Supporting Program (TUMA-Project-2021-27) Almaarefa University, Riyadh, Saudi Arabia.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] W. Ahmad, A. Rasool, A. R. Javed, T. Baker and Z. Jalil, "Cyber security in IoT-based cloud computing: A comprehensive survey," *Electronics*, vol. 11, no. 1, pp. 1–34, 2022.
[2] S. Ustebay, Z. Turgut and M. A. Aydin, "Intrusion detection system with recursive feature elimination by using random forest and deep learning classifier," in *2018 Int. Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, Ankara, Turkey, pp. 71–76, 2018.
[3] V. Vishal and K. Vasudha, "Dos/ddos attack detection using machine learning: A review," in *Proc. of the Int. Conf. on Innovative Computing & Communication (ICICC)*, Delhi, India, 2021.
[4] A. Mohiyuddin, A. R. Javed, C. Chakraborty, M. Rizwan, M. Shabbir *et al.*, "Secure cloud storage for medical iot data using adaptive neuro-fuzzy inference system," *International Journal of Fuzzy Systems*, vol. 24, pp. 1203–1215, 2022.
[5] M. M. Islam, M. Z. Hasan and R. A. Shaon, "A novel approach for clientside encryption in cloud computing," in *Int. Conf. on Electrical, Computer and Communication Engineering (ECCE)*, Cox'sBazar, Bangladesh, pp. 1–6, 2019.
[6] D. Boland, "Securing Amazon web services (A.W.S.) and simple storage service (Amazon S3) security", *An article regarding amazon simple storage service security, available at* http://www.infosecwriters.com, June 2020.
[7] M. F. Mushtaq, S. Jamel, A. H. Disina, Z. A. Pindar, S. A. Shakir *et al.*, "A survey on the cryptographic encryption algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 11, pp. 333–44, 2017.
[8] S. Tayal, N. Gupta, P. Gupta, D. Goyal and M. Goyal, "A review paper on network security and cryptography," *Advances in Computational Sciences and Technology*, vol. 10, no. 5, pp. 763–770, 2017.
[9] M. Hasan, N. Ariffin and N. Sani, "A review of cryptographic impact in cybersecurity on smart grid: Threat, challenges and countermeasures," *Journal of Theoretical and Applied Information Technology*, vol. 99, no. 10, pp. 2458–2472, 2021.
[10] S. A. Ahmad and A. B. Garko, "Hybrid cryptography algorithms in cloud computing: A review," in *2019 15th Int. Conf. on Electronics, Computer and Computation (ICECCO)*, Abuja, Nigeria, pp. 1–6, 2019.
[11] S. Pandey, G. N. Purohit and U. M. Munshi, "Data security in cloud-based applications," In Munshi U., Verma N. (Eds.) in *Data Science Landscape. Studies in Big Data*, vol. 38. Singapore: Springer, pp. 1–6, 2018.
[12] B. -H. Lee, E. K. Dewi and M. F. Wajdi, "Data security in cloud computing using A.E.S. under HEROKU cloud," in *2018 27th Wireless and Optical Communication Conf. (WOCC)*, Hualien, Taiwan, pp. 1–5, 2018.

[13] C. Biswas, U. D. Gupta and M. M. Haque, "An efficient algorithm for confidentiality, integrity, and authentication using hybrid cryptography and steganography," in *2019 Int. Conf. on Electrical, Computer and Communication Engineering (ECCE)*, Cox'sBazar, Bangladesh, pp. 1–5, 2019.

[14] Y. Sharma, H. Gupta and S. K. Khatri, "A security model for the enhancement of data privacy in cloud computing," in *Amity Int. Conf. on Artificial Intelligence*, Dubai, United Arab Emirates, pp. 898–902, 2019.

[15] M. Z. Abdullah and Z. J. Khaleefah, "Design and implement of a hybrid cryptography textual system," in *Int. Conf. on Engineering and Technology*, Antalya, Turkey, pp. 1–6, 2017.

[16] M. Harini, K. P. Gowri, C. Pavithra and M. P. Selvarani, "A novel security mechanism using hybrid cryptography algorithms," in *IEEE Int. Conf. on Electrical, Instrumentation and Communication Engineering*, Karur, India, pp. 1–4, 2017.

[17] C. Ponnusamy and P. Deepalakshmi, "Design of secure storage for health-care cloud using hybrid cryptography," in *2018 Second Int. Conf. on Inventive Communication and Computational Technologies*, Coimbatore, India, pp. 1717–1720, 2018.

[18] V. K. Soman and V. Natarajan, "An enhanced hybrid data security algorithm for cloud," in *Int. Conf. on Networks & Advances in Computational Technologies*, Thiruvananthapuram, India, pp. 416–419, 2017.

[19] B. Swathi, S. D. Bhaludra and S. Raveendranadh, "Secure file storage in cloud computing using hybrid cryptography algorithm," *International Journal of Advance Research in Science and Engineering*, vol. 6, no. 3, pp. 70–77, 2017.

[20] D. S. Dahanukar and D. S. Shelke, "A review paper on cryptography and network security," *International Journal of Advanced Research in Science, Communication, and Technology*, vol. 12, no. 1, pp. 108–114, 2021.

[21] K. P. Karule and N. V. Nagrale, "Comparative analysis of encryption algorithms for various types of data files for data security," *International Journal of Scientific Engineering and Applied Science*, vol. 2, no. 2, pp. 495–498, 2016.

[22] H. Sun and R. Grishman, "Lexicalized dependency paths based supervised learning for relation extraction," *Computer Systems Science and Engineering*, vol. 43, no. 3, pp. 861–870, 2022.

[23] S. Singh and S. Kumar, "Analysis of various cryptographic algorithms," *Int. J. Advanced Research in Science, Engineering and Technology*, vol. 5, no. 3, pp. 5341–5348, 2018.

[24] M. A. Al-Shabi, "Advanced hill cipher algorithm for security image data with the involutory key matrix," *Journal of Physics: Conference Series*, Makassar, Indonesia, vol. 1899, pp. 1–8, 2021.

[25] M. Nadeem, A. Arshad, S. Riaz, S. S. Band and A. Mosavi, "Intercept the cloud network rom brute force and ddos attacks via intrusion detection and prevention system," *IEEE Access*, vol. 9, pp. 152300–152309, 2021.

[26] T. Alawiyah, A. B. Hikmah, W. Wiguna, M. Kusmira, H. Sutisna *et al.,* "Generation of rectangular matrix key for hill cipher algorithm using playfair cipher," *Journal of Physics: Conference Series*, Voronezh, Russia, vol. 1641, pp. 1–5, 2020.

[27] A. Musa and A. Mahmood, "Client-side cryptography based security for cloud computing system," in *2021 Int. Conf. on Artificial Intelligence and Smart Systems (ICAIS)*, pp. 594–600, 2021.

[28] M. A. Al-Shabi, "A survey on symmetric and asymmetric cryptography algorithms in information security," *International Journal of Scientific and Research Publications (IJSRP)*, Coimbatore, India, vol. 9, no. 3, pp. 576–589, 2019.