






## Article

# Emergence of Novel WEDEx-Kerberos Cryptographic Framework to Strengthen the Cloud Data Security against Malicious Attacks

Syeda Wajiha Zahra <sup>1</sup>, Muhammad Nadeem <sup>2</sup>, Ali Arshad <sup>3,\*</sup>, Saman Riaz <sup>4</sup>, Waqas Ahmed <sup>5</sup>, Muhammad Abu Bakr <sup>6</sup> and Amerah Alabrah <sup>7</sup>

<sup>1</sup> Department of Computer Science, National University of Technology, Islamabad 44000, Pakistan; syeda.wajia786@gmail.com

<sup>2</sup> Department of Computer Science, University of Science and Technology Beijing, Beijing 100083, China; nadeem72g@gmail.com

<sup>3</sup> Department of Computing, NASTP Institute of Information Technology, Lahore 58810, Pakistan

<sup>4</sup> Department of Computing, Riphah International University, Lahore 54660, Pakistan; samanriaz@hotmail.com

<sup>5</sup> Department of Computer Science, Qurtuba University of Science and Technology, Dera Ismail Khan 29050, Pakistan; waqasahmad9107@gmail.com

<sup>6</sup> Department of Electrical Engineering, National University of Technology, Islamabad 44000, Pakistan; muhammadabubakr@nutech.edu.pk

<sup>7</sup> Department of Information Systems, College of Computer and Information Science, King Saud University, Riyadh 11543, Saudi Arabia; aalobrah@ksu.edu.sa

\* Correspondence: alli.arshad@gmail.com

**Abstract:** Researchers have created cryptography algorithms that encrypt data using a public or private key to secure it from intruders. It is insufficient to protect the data by using such a key. No research article has identified an algorithm capable of protecting both the data and the associated key, nor has any mechanism been developed to determine whether access to the data is permissible or impermissible based on the authentication of the key. This paper presents a WEDEx-Kerberos Framework for data protection, in which a user-defined key is firstly converted to a cipher key using the “Secure Words on Joining Key (SWJK)” algorithm. Subsequently, a WEDEx-Kerberos encryption mechanism is created to protect the data by encrypting it with the cipher key. The first reason for making the WEDEx-Kerberos Framework is to convert the user-defined key into a key that has nothing to do with the original key, and the length of the cipher key is much shorter than the original key. The second reason is that each ciphertext and key value are interlinked. When an intruder utilizes the snatching mechanism to obtain data, the attacker obtains data or a key unrelated to the original data. No matter how efficient the algorithm is, an attacker cannot access the data when these methods and algorithms are used to protect it. Finally, the proposed algorithm is compared to the previous approaches to determine the uniqueness of the algorithm and assess its superiority to the previous algorithms.

**Keywords:** Kerberos; cryptographic; random seed; cloud computing; cyber security; network attacks



**Citation:** Zahra, S.W.; Nadeem, M.; Arshad, A.; Riaz, S.; Ahmed, W.; Abu Bakr, M.; Alabrah, A. Emergence of Novel WEDEx-Kerberos Cryptographic Framework to Strengthen the Cloud Data Security against Malicious Attacks. *Symmetry* **2024**, *16*, 605. <https://doi.org/10.3390/sym16050605>

Received: 7 April 2024

Revised: 8 May 2024

Accepted: 8 May 2024

Published: 13 May 2024



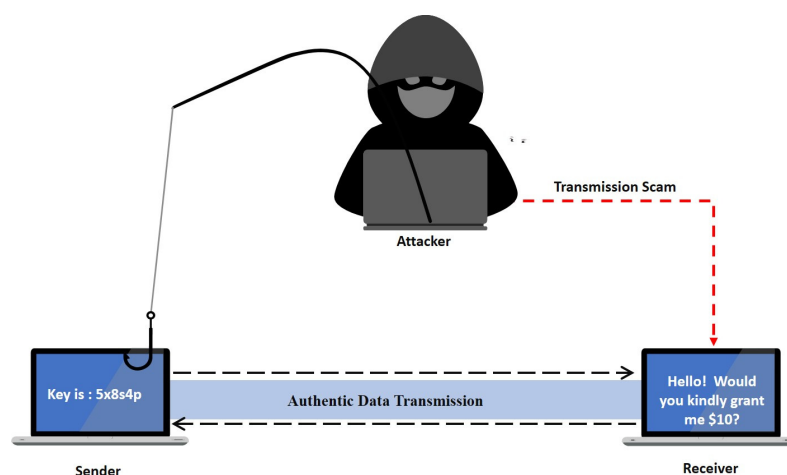
**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Nowadays, attackers use sneaky methods to get into cloud networks, in which attackers always identify weaknesses in the infrastructure of the cloud or faults that people make when using the online service [1]. Cloud providers or users may occasionally commit errors in configuring their systems, exposing them to potential vulnerabilities [2]. Attackers do everything possible to gain access to users’ data, including sending fake emails to users that look like real emails and asking for usernames and passwords [3]. When someone falls for this trick and enters their information, an attacker can use it to sneak into a cloud account. Different attacks target the data stored in a cloud network and aim to retrieve, change,

or destroy it [4]. One typical attack is a “data breach,” in which attackers get access to confidential data saved in a cloud server [5]. Attackers may employ numerous techniques to infiltrate a cloud network [6], including trying to decipher passwords or focusing on vulnerabilities in cloud services. Once attackers access the network, they may acquire any desired information, which can be used for different purposes, such as identity theft, economic deception, or other unlawful endeavors [7].

Cloud attacks on enterprises’ networks have become problematic in the digital world [8]. Cybercriminals are constantly developing innovative methods to exploit weaknesses in cloud settings [9]. When enterprises accidentally expose their data and resources to the public Internet, misconfiguration of cloud services is one typical attack vector that opens doors for attackers [10]. Every time a hacker attempts to access data, their initial effort is to get the original data, for which they make every effort and perform malicious attacks on the encrypted data. Malicious attacks utilize advanced techniques to break encryption and get unauthorized access to encrypted data. Malicious attacks include advanced cryptanalysis, brute force attacks, and fault injection attacks. After obtaining access to the data, the intruder’s next phase is to transmit the recipient’s destructive data rather than the original data to carry out unlawful actions across the entire network [11], as illustrated in Figure 1. Organizations must implement robust security measures and train personnel on best practices to protect against these constantly developing cloud-based risks as cloud usage rises [12].



**Figure 1.** Data snatching mechanism.

### 1.1. Cryptography

Cryptography is used to secure communication and protect information from unauthorized access or malicious attacks [13]. It entails encrypting data (plaintext) so that only authorized users may decode it using a specific key or algorithm. The two main types of cryptographic techniques are symmetric and asymmetric key cryptography [14]. Symmetric cryptography encrypts data using a similar key [15], whereas asymmetric cryptography uses two separate keys to encrypt and decrypt data [16]. This paper will develop a framework in which data are encrypted with a dynamic key while data are decrypted with a static key called a cipher key. Each encrypted ciphertext key will only work on the text from which the key is derived.

### 1.2. Problem Formulation

Researchers have developed several techniques and machine learning algorithms to prevent data misuse by attackers. For encoding data, multiple researchers created static keys with cryptographic methods. Data that have been broadcast over the network and encrypted by researchers may be retrieved with the use of a static key. Data can be encapsulated through all these techniques, but such mechanisms are unreliable for cloud networks. If an attacker uses cryptanalysis, the key can be easily obtained, which is a data

security problem. Malware can be detected with machine learning algorithms, but the data cannot be saved. These issues of cloud networks have yet to be solved entirely in any paper, and no reliable mechanism has been developed for cloud networks.

### 1.3. Proposed Framework

This paper developed a WEDEx-Kerberotic framework to secure data, and various algorithms are used to secure data and keys. The WEDEx-Kerberotic system is made up of two different methods, in which the first method is designed for data encryption and decryption, which is named “WEDEx”. “WEDEx” is made up of different words. The purpose of creating the WEDEx” algorithm is to provide key-based security for exchanging and encrypting large-scale data, which is why “WEDEx” is made up of various words whose full form is “Widely Encrypted Data Exchange”. Another technique is “Kerberotic”, which is a data security mechanism built from “Kerberos Authentication System”. The aim of the Kerberotic system is to provide a complete encryption and key authentication framework. The WEDEx-Kerberotic system aims to develop a framework that thoroughly performs data encryption, decryption, key generation, and verification phases. Along with encrypting and decrypting the data, it is also essential to encrypt the key used on the data. Verification of this key is also necessary when any authentic or inauthentic key is used on encrypted data. When all processes are carried out in parallel, a framework for the cloud can be efficient.

Firstly, a user-defined key  $K[x]$  will be taken, and then a key  $K[j]$  will be obtained by implementing the SWJK (secure words on joining key) algorithm on the key  $K[x]$ .  $K[x]$  is the original key (user-defined key), while  $K[j]$  is the key obtained from the SWJK algorithm. A user-defined key will decrease its length by two times when the SWJK algorithm is applied to it. The most significant advantage of the SWJK key is that the SWJK key will be utterly different from the user-defined key. When cryptanalysis is carried out, no value will be the same, making the cryptanalysis impossible. Applying the SWJK algorithm's key to the WEDEx-Kerberotic method will produce a ciphertext. Each text's value in the WEDEx-Kerberotic system will be connected to the one after it. If the attacker changes even one value, all values will be automatically changed, and the attacker will get data that cannot be easily deciphered and has no relation to the authentic data. The key generated through the SWJK technique will be saved in the Kerberos system. The Kerberos system will act as a database authentication system. The user will first send the key they got from the SWJK method to the Kerberos system in order to decode the data. The Kerberos system will verify the validity of the SWJK key via the KSS method. After verification of the key's validity, the data decryption will automatically use the same key, yielding the plaintext. The peculiarity of this work is that when such methods and algorithms are employed for data security, data leakage or breakage will become impossible if the attacker applies any technique or algorithm to the suggested algorithm.

The remaining paper will be distributed in this manner. In Section 2, the proposed work will be discussed. The suggested algorithm will be detailed in Section 3. The suggested work will be appraised in Section 4. In Section 5, we will compare the suggested work with existing work to see how much better it is. Conclusions and suggestions for further study are included in Section 6.

## 2. Related Work

M. A. Al-Shabi [17] integrated several symmetric and asymmetric methods to assess the effectiveness of various cryptographic algorithms. After gathering the methods, the uniqueness of each one was evaluated, and it was then debated which method was best for data security. Researchers determined that when data are encrypted using symmetric algorithms, greater security may be given for the data than asymmetric algorithms. Researchers also found that symmetric methods are more reliable and efficient than asymmetric ones.

Musa et al. [18] state that man-in-the-middle attacks, in which the attacker does everything they can to destroy data, constitute the majority of assaults against cloud

networks. To address this issue, researchers originated a hill cipher algorithm that takes advantage of all practical methods for securing data against man-in-the-middle attacks. The researchers started with plaintext and applied the standard ASCII table to acquire ASCII values. Afterward, using Hill cipher methods, researchers used the ASCII values to create a ciphertext.

Hossain et al. [19] created a data security method by merging three separate processes: the first was Playfair, the second was stream cipher, and the third was Caesar cipher. The program first used the Caesar cipher approach and acquired a CT1 ciphertext. Next, the Playfair method was used for CT1, resulting in the CT2 ciphertext. Ultimately, a ciphertext was acquired by applying a stream cipher to the cipher values derived from CT2. Each method in data security utilizes three distinct static keys.

According to Akanksha et al. [20], it is pretty simple for an attacker to obtain the data once they have obtained the key using cryptanalysis or a key-base assault. To address this problem, the researchers used a static key on the simple text and spoke about the necessity of repeating the plaintext if the key is shorter than the plaintext. The researchers generated a static key using a key generator that used the complete length of the plaintext rather than using the same key again. Researchers then used this key with the Vigenère cipher technique to generate a ciphertext.

H. Sun and R. Grishman [21] used several strategies to enhance the security of the Hill cipher algorithm and created an improved version known as the Hill cipher chain algorithm. This algorithm is used to secure the primary key. The data were encrypted using the Hill cipher method after encrypting the main key. This encryption process included employing a 2x2 matrix structure, resulting in the generation of a ciphertext.

Tan et al. [22] claim that the Ceaser cipher algorithm is readily broken. The researchers created a hybrid method to address this issue by fusing the best Ceaser cipher and Vigenère cipher elements. First, raw text was encrypted using the Caesar cipher method. Then, each character was shifted on a formula. The Caesar cipher algorithm results were then used to implement the effective Vigenère cipher method, yielding a ciphertext.

Several strategies are available for protecting cloud data from attackers, which, when utilized appropriately, can preserve data security. The researchers [23] created an effective solution by combining several ways to address this issue. First, the data were transformed into a decimal format using an ASCII table. To enhance the amount of encrypted text, a radix 64-bit algorithm for encryption was applied to encode decimal values. The ciphertext was then produced using the Hill encryption method. The Hill matrix method was created to make a key that could be used up to the same plaintext from which it was taken.

Singh and Pandey [24] conducted a comprehensive study of many studies to assess the issues presented by replay attacks. In addition, they compiled an extensive collection of techniques, records, and procedures that can be utilized to protect data against replay attacks. Subsequently, the discussion covered other grounds for replay attacks, such as the deficiency of security procedures, the utilization of vulnerable design, and the employment of weak protocols. After conducting a comprehensive analysis of several research articles and gathering diverse approaches, it was concluded that integrating these techniques into a cohesive algorithm makes it possible to enhance the security of cloud data and protect it from replay assaults.

In an extensive review, N. R. Tadapaneni examined several research publications [25] and elucidated that the absence of an initial degree of security in the infrastructure is the main factor behind the occurrence of attacks on cloud networks. Researchers emphasized multiple safety obstacles that could be used to mitigate attacks and fix this issue. Researchers assert that any technology may be divided into two distinct phases. One step is beneficial for ensuring dependable transmission on the cloud network, while the other involves the attacker attempting to obstruct the transmission of users. Subsequently, many strategies and algorithms were integrated and deliberated to ascertain how the intrusion ratio may be minimized using all effective ways on the cloud server.

Amirreza and Behrouz [26] state that protecting data against replay assaults on a control system network is complex. To detect and mitigate replay attacks on the control system network, a linear quadratic Gaussian (LQG) controller system was developed. A robust network structure was first devised using dropout-packet characteristics via the Kullback–Leibler divergence algorithm. It was observed that the occurrence of replay attacks on a cloud network typically results in a larger ratio of successful assaults compared to the rate at which these attacks are detected.

When content is transmitted over a network of clouds, it may be opened for any node linked to the network [27]. This means that if someone who is not authorized wants to intercept and alter the original data packets, they can do so by broadcasting illicit data packets to nodes on the network. Differentiating between illicit packets and genuine ones is a difficult task. Replay attacks were carried out using three separate nodes inside a control area network (CAN) to tackle this problem, and malfunctioning nodes were detected. Two replay attacks were executed to identify malfunctioning nodes, including the interception of the entire message, subsequent modification, and dissemination of this altered message over the network's entirety. Subsequently, the whole communication was collected and subsequently disseminated. During the discussion, it was determined that detecting a partial replay attack message is more uncomplicated than detecting a whole replay attack message when data are made public in the cloud network.

Researchers developed an application [28] that utilizes intrusion detection systems to defend the public cloud infrastructure from outside attacks. The framework was initially designed to ensure the security of the cloud network. Subsequently, the intrusion detection system identified outside attacks occurring on a cloud network. The subsequent discussion focused on strategies to safeguard the cloud network against these attacks. The Cloudflare method, which blocks the attacker's IP address on a static or dynamic basis, was employed to defend the cloud network from internal attacks, and it was discussed if every device in the cloud network was protected. After that, researchers discussed that the attacking rates can be reduced if these security mechanisms are implemented on the network.

Felix and Isaac [29] developed a triple pass protocol (TPP) method to protect data in which data were encrypted in three different phases. First, a plaintext was taken, square matrix encryption was applied to the plaintext, and a ciphertext called ciphertext-1 was obtained. Subsequently, the Hill matrix algorithm was used to ciphertext-1, and new results were obtained called ciphertext-2. Ultimately, the ciphertext-2 values were inverted, resulting in the acquisition of a ciphertext referred to as the original ciphertext. The primary purpose of developing the triple pass protocol was to provide security in three stages that could not be broken.

In 2021 [30], researchers created an algorithm known as the Hill matrix. An image was used as input to accomplish file-based encryption, during which the file was encrypted and the colors were determined. Before coloring, the image underwent a conversion to grayscale. After converting the image to grayscale, it was converted to pixels, and then the conversion was carried out in matrix form. After converting the image to the matrix, the Hill matrix algorithm was implemented, and the file was encrypted.

In a study [31], Elsaeidy et al. first identified several origins of replay and DDoS assaults. They then created a deep learning model with a hybrid approach that could effectively identify attacks inside a smart city environment. After that, researchers discussed how DDoS attacks send out messages without connecting to any data. Meanwhile, replay attacks capture the original data and broadcast malicious data instead of the data with the same infrastructure as the original data. Either kind of attack may harm the cloud server. Three real-time datasets were used to solve this problem: environmental, smart-soil, and smart-river datasets. These datasets were implemented on a hybrid deep learning model and identified the attacks. Following that, it was discussed that replay attacks could not be easily detected due to the same infrastructure as DDoS attacks because of their behavior.

Multiple algorithms and methods [32] have been devised to protect cloud data against replay attacks. However, it is essential to note that cloud data are still vulnerable to

such assaults. The researchers developed an effective encryption method to tackle this problem, which was used to encrypt the raw text data and then upload the encrypted data to the cloud server. The data were validated at several stages after receiving it, and the authenticity of the arriving data were identified. After that, the existing work was compared with the previous work, and it was discussed that the cloud server could be secured from the attacker if this approach is employed to defend against replay attacks.

Bharath and Rajesh [33] gathered several symmetric and asymmetric cryptography methods to assess the effectiveness and accuracy of different algorithms. Researchers have shown that when encryption is performed using symmetric cryptography, a single key is utilized for encryption and decryption. However, if encryption is performed using asymmetric key cryptography, two separate keys will be used for encryption and decryption. Encryption will be performed using one key, while decryption will be performed using another. Subsequently, many algorithms were gathered, and their performance and efficiency were assessed. The discussion concluded that the symmetric method outperforms the asymmetric approach in terms of both performance and efficiency.

### 3. Proposed Algorithm

Three procedures will be devised to preserve and encrypt data from potential intruders. The first is to develop a customized ASCII table that aims to shift each ASCII value to different indexes. When the attacker attempts to access the data from the cloud, their first approach is to employ the Standard ASCII table in their grabbing methods. Using a customized ASCII table rather than the Standard ASCII table makes it challenging for an attacker to get ASCII indexation. When the attacker attempts to decode the data using any method on the cipher data, the attacker will obtain data utterly unrelated to the original data. A Kerberos system will be developed to implement the customized table and store the data, which will be discussed in Section 3.2. To protect the cloud data, it is vital to protect the key associated with it. It does not matter how many secure algorithms are developed for cloud data. Each algorithm works with a key. Once an attacker has access to the key, it will be easy for the attacker to gain access to the data, which is a problem. To solve this problem, the SWJK algorithm will be developed in this paper, which aims to protect user-defined keys by using the SWJK algorithm, and the key has to be encrypted with a key of various symbols, numbers, and characters. When such a key is used to protect data, it will be challenging for an attacker to guess the key or implement cryptanalysis. The discussion on the SWJK algorithm will be carried out in detail in Section 3.2. Table 1 displays many mathematical notations and their respective meanings.

**Table 1.** Mathematical Notations.

Notation	Meaning
$[\delta_1]$	Odd indexes
$[\delta_2]$	Even indexes
$\oint_{[m\ n]}$	m-nth results of SWJK
$\sigma$	Indexes value that will be a test
$*$	Multiplying values from $n$ th to $n$ th
$\oint_{[n]}^{[m]}$	Concatenation results of $\oint_n$ (s)
$\int_{[q]}^{[p]}$	$\int$ is the positional result of row (p) and column (q)
$\gamma_{[r]}$	Concatenation of key-value results.
$\dagger$	Concatenation
$\in$	Belong to
$\perp$	Dependency

#### 3.1. Customized ASCII Table

The most crucial components for data security against hackers or cryptanalysis assaults are ASCII values. When attempting to steal data, an attacker always employs predefined

ASCII tables that are advantageous to the attacker. Data security may be considerably increased when employing a customized ASCII table to store data instead of an ordinary ASCII table. The data for this research have been stored in a modified ASCII table. Characters, numbers, and alphanumeric letters are arranged in a randomly generated fashion in a customized ASCII table. When the values of the provided ASCII table and the customized ASCII table are in separate indexes, it will be challenging for the attacker to obtain the customized ASCII table index data. Figure 2 displays a whole ASCII table.

0	Δ	16	θ	32	•	48	∞	64	£	80	☐	96	χ	112	○	128	※	144	⊗	160	☆	176	ν	192	©	208	⌘	224	⌘	240	\$
1	Δ	17	ϑ	33	•	49	∞	65	£	81	⊖	97	χ	113	⊗	129	⊗	145	⊗	161	☆	177	⊗	193	⊗	209	⊗	225	⊗	241	8
2	!	18	"	34	#	50	\$	66	%	82	⊖	98	(	114	)	130	*	146	+	162	,	178	-	194	.	210	/	226	0	242	η
3	!	19	2	35	3	51	4	67	5	83	7	99	8	115	9	131	:	147	:	163	<	179	=	195	>	211	?	227	@	243	È
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	É
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	Ê
6	a	22	b	38	c	54	d	70	e	86	g	102	h	118	i	134	j	150	k	166	l	182	m	198	n	214	o	230	p	246	Θ
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	⊂
8	ü	24	ë	40	ä	56	ä	72	ä	88	ë	104	e	120	ë	136	i	152	i	168	i	184	Ä	200	Ä	216	232	⊗	248	U	
9	ä	25	ë	41	Q	57	Æ	73	ö	89	ö	105	ü	121	ü	137	ÿ	153	Ö	169	Ü	185	Ø	201	£	217	Ø	233	z	249	Ť
10	f	26	ä	42	i	58	ó	74	ü	90	N	106	f	122	ó	138	¿	154	⊗	170	~	186	½	202	¼	218	j	234	«	250	Ť
11	»	27	Δ	43	⊗	59	†	75	l	91	Ä	107	Ä	123	Ä	139	⊗	155	⊗	171		187	⌘	203	⌘	219	⊗	235	⊗	251	&nbsp;
12	⌘	28	L	44	⌘	60	⌘	76	⌘	92	⌘	108	ä	124	Ä	140	⌘	156	⌘	172	Λ	188	⌘	204	⌘	220	=	236	⌘	252	Q
13	⌘	29	ð	45	Ð	61	È	77	È	93	ı	109	ı	125	ı	141	ı	157	ı	173	ı	189	⊗	205	ı	221		237	ı	253	⊂
14	Δ	30	Ö	46	ß	62	Ö	78	Ö	94	Ö	110	μ	126	4b	142	⊂	158	Ü	174	Ü	190	Ü	206	ÿ	222	ÿ	238	~	254	đ
15	ˆ	31	ˆ	47	±	63	—	79	¼	95	F	111	ˆ	127	,	143	°	159	ˆ	175	ˆ	191	ü	207	ˆ	223	⊂	239	μ	255	⊂

Figure 2. Customized ASCII table.

### 3.2. Kerberotic System

Figure 3 depicts the database system (DS) and authentication system (AS) that comprise the key distribution system known as the Kerberos system. Whenever a user encrypts data, all the keys generated by the SWJK algorithm will be automatically saved in the key database system. All varieties of SWJK keys will be kept in the key database system, which is a centralized database system. All keys used in data encryption will be checked within the framework of the authentication system, which is called the key verification system. If the key is valid, then text decryption will be possible. If the arriving key does not match an existing key, the data decryption process will not be possible. Data decryption is possible only when the arriving key matches any one of the keys that are already stored in the database of the Kerberos system, and the data decryption will be carried out automatically on the same key that has already been used, whether the arriving key will match with encrypted text or not. To implement the data decryption process, the key must match the stored key of the database system. When the key does not match the database, the data decryption process will not be possible.

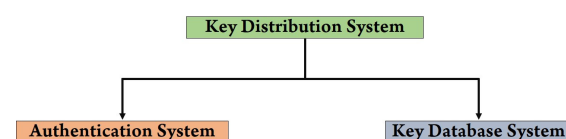


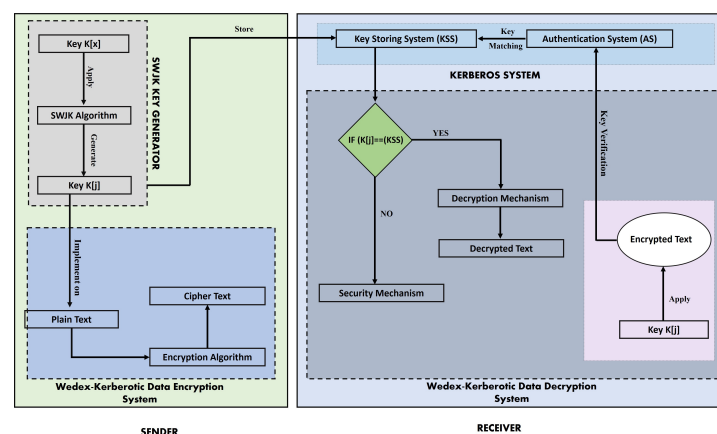
Figure 3. Key distribution system.

When an arriving key is matched with a key stored in the database system, data decryption is possible whether the key is authentic or inauthentic. However, by not using the original key, the attacker will get data that is not authentic and will have no relation to the original data. The main advantage of using such an algorithm would be that the attacker would not have to match the key repeatedly and could not decipher the decrypted data. The original key obtained from the SWJK algorithm must be implemented on the cipher data to get the original data. Otherwise, data decryption will not be possible.

Various researchers used different algorithms to protect data from attackers. Some researchers used user-defined symmetric keys to secure the data. Some researchers have worked on public and private user-defined keys to protect cloud data. Different symmetric

and asymmetric user-defined keys are not the best solution to protect the data entirely from attackers. Each user uses a key to protect the data, but by guessing this key, an attacker can access the cloud data. When the proposed SWJK algorithm is implemented on the user-defined key, a new key (cipher key) containing various symbols, numbers, and characters is obtained. Then, when the data are encrypted with that key, it will be impossible for an attacker to get or access it. The length of the key obtained from the SWJK algorithm will be much less than the original key, and no snatching, guessing, or cryptanalysis mechanism can be used on such a key. Various researchers have used symmetric or asymmetric keys to secure data. If any algorithm has been developed to protect the key, then the length of the key has not been reduced with the help of any algorithm. When the cipher key length is half the user-defined key, no matter how efficiently the algorithm is developed for key generation, the key cannot be accessible.

When the SWJK algorithm is applied on a user-defined key  $K[x]$ , a secure key " $K[j]$ " will be produced. After obtaining the key  $K[j]$ , the key will be implemented on plaintext data, and a ciphertext will be obtained. The key " $K[j]$ " will be stored on the KSS of the Kerberos system, as illustrated in Figure 4. To establish whether the key submitted on the Kerberos system is legitimate or invalid before attempting to decrypt the data, the user's key must first be authenticated. The purpose of an authentication system (AS) is to check the authenticity of the key. When the user uses the key  $K[j]$  for data decryption, the key will first be transmitted to the encryption system, where the key-matching process will be carried out with the help of KSS. Data decryption is possible if the arriving key matches the KSS key. If the arriving key matches the KSS key, but some other data are encrypted with that KSS key, and when the arriving key is applied to the encrypted data, the user will get data that has no relation to the original data. The attacker consistently attempts to get the key, which is why this technique is used. Even after decrypting the encrypted data using a key obtained through the key-matching mechanism, the attacker cannot access the original data because that data have no relation to the original data. The advantage of this will be that the encrypted data will not be attacked and searched repeatedly, which is a better form of data security.



**Figure 4.** Framework of WEDEx Kerberotic system.

Data decryption will be feasible if  $K[j]$  meets KSS. The original data will be decrypted if the incoming key is equal to the one used to encrypt the data. The data will be decrypted but will only be authentic if the key is genuine and matches the encrypted data.

#### SWJK Key Generator Algorithm

The SWJK key generator is an excellent technique for protecting data from intruders. In this technique, the cipher key is generated using a user-defined key, and then the ciphertext is obtained by applying the cipher key to plaintext. It is relatively easy for an attacker to figure out a username-base key, numeric key, or alphabetic key pattern. A typical example

of this is cryptanalysis data mining. However, obtaining that key will be complicated when data are encrypted with a cipher key.

The SWJK technique is created to address this issue, which will firstly generate a user-defined key  $K[x]$ , as illustrated in Figure 5. “ $x$ ” refers to the key’s length, which is maintained constant in the proposed technique  $K[7]$ . Following acquiring the key,  $K[7]$  will be indexed using Equation (1).

$$[K] = [K_0] [K_1] [K_2] \dots [K_p] \quad (1)$$

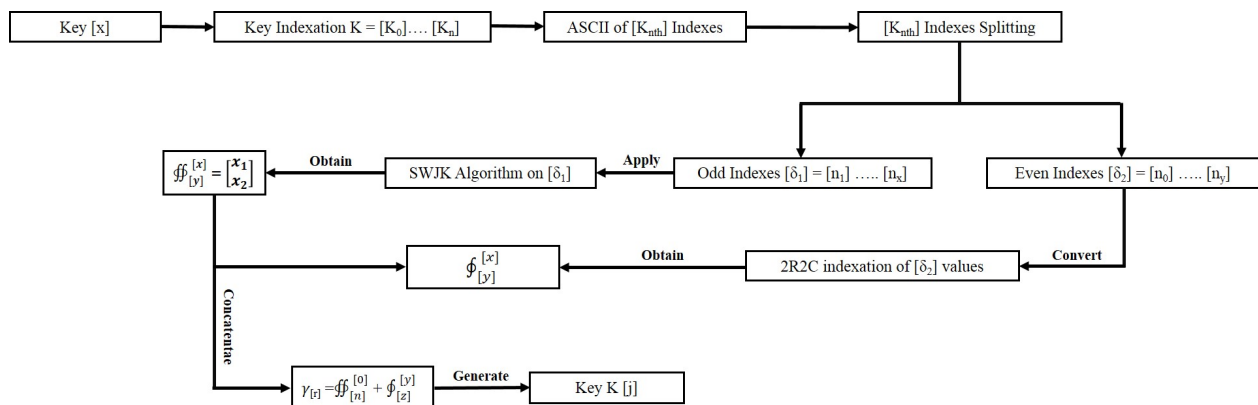


Figure 5. SWJK key generator.

As illustrated in Equation (2), the equivalent ASCII value  $K[n]$  of each key value will be determined after indexing the key  $[K]$ .

$$K[n] = (AS_0[K_0]) (AS_1[K_1]) \dots (AS_n[K_n]) \quad (2)$$

$(AS_0[K_0])$  indicates that the ASCII of Index  $[K_0]$  is  $AS_0$ . Similarly,  $(AS_1[K_0])$  indicates that the ASCII of Index  $[K_1]$  is ASCII  $AS_1$ . After obtaining the key  $K[n]$  index ASCII values, the index values will be divided into two sections, the first of which will be odd indexation  $[\delta_1]$  as illustrated in Equation (2a), and the second, even indexation  $[\delta_2]$  is illustrated in Equation (2b).

$$\text{Odd Indexes } [\delta_1] = (AS_1[K_1]) (AS_3[K_3]) \dots (AS_x[K_x]) \quad (2a)$$

Odd indexation  $[\delta_1]$  will have all the odd values of  $K[n]$ , which start with  $[n_1]$  and end will be on  $[n_x]$ .

$$\text{Even Indexes } [\delta_2] = (AS_0[K_0]) (AS_2[K_2]) \dots (AS_y[K_y]) \quad (2b)$$

Even indexation  $[\delta_2]$  will contain all even values. Even values start from Index  $[n_0]$  and end on  $[n_y]$ . The SWJK algorithm will be applied to the odd index values  $[\delta_1]$ . In the SWJK algorithm, odd indexation  $[\delta_1]$  values will be obtained, and Equation (3) will be applied to the values.

$$\phi_{[m\ n]}(s) = \sigma^* \bmod [x] \quad (3)$$

In eq (ii), “ $\sigma$ ”  $i[1]$  and  $i[3]$  are the values on which testing will be applied. “ $i$ ” represents the index number. “ $\phi$ ” is the result of Equation 3. Index  $[m]$  will contain the values of the test-related indexes  $i[1]$  and  $i[3]$ . First, the first value will be obtained from Index  $i[1]$  and utilized for testing. A second value will be obtained from Index  $i[3]$  for testing. The value of Index  $[n]$  will be the Index  $[5]$  value used in  $\bmod [x]$ .

After getting the two values from *Indexes i[1][3]* and *I [3][5]*, all values will be transformed using Equation (4) into *1x2-matrix form*.

$$\mathbb{F}_{[n]}^{[m]} = \begin{bmatrix} \mathcal{F}_{[1][5]} \\ \mathcal{F}_{[3][5]} \end{bmatrix} \quad (4)$$

After getting the values from Equation (4) index, all values of *even indexation*  $[\delta_2]$  will be converted into the *2x2-matrix form*  $[\delta_{[2x2]}]$  with the help of Equation (5).

$$[\delta_{[2x2]}] = \begin{bmatrix} [i] & [j] \\ [k] & [l] \end{bmatrix} \quad (5)$$

In Equation (5), two rows and two columns are denoted by  $[2x2]$ , while the indexes represent the values that result from  $[\delta_2]$   $[i][j][k][l]$ . The result of Equation (5) can also be expressed as in Equation (6).

$$[\delta_{[2x2]}] = \int_{[q]}^{[p]} \quad (6)$$

In Equation (6),  $[p]$  denotes the length of rows and  $[q]$  the length of columns. All values will be added together, as illustrated in Equation (7), once the results of odd indexation Equation (4) and even indexes Equation (6) have been obtained.

$$\gamma_{[r]} = \int_{[q]}^{[p]} + \mathbb{F}_{[n]}^{[m]} \quad (7)$$

By rules, we know that the values of  $\gamma_{[r]}$  belong to  $\int_{[q]}^{[p]}$  and  $\mathbb{F}_{[n]}^{[m]}$ . So, mathematically for Equation (8), the following can be said:

$$[\gamma_{[r]}] \in \left[ \int_{[q]}^{[p]} \mathbb{F}_{[n]}^{[m]} \right] \quad (8)$$

$[\gamma_{[r]}]$  is the key generated by the SWJK algorithm. After obtaining the values from  $[\gamma_{[r]}]$ , convert the values of  $[\gamma_{[r]}]$  to *1x4-matrix form*. After that, a key  $K[j]$  will be obtained. After generating the key  $K[j]$  from the SWJK algorithm, we can say that the SWJK algorithm can reduce the length of  $K[x]$ . If this algorithm is used to generate the key, no one attacker can guess the key and no one can perform the cryptanalysis attack on the  $[\gamma_{[r]}]$  key. The length of the original key is  $K[7]$ , while the length of the text obtained by the SWJK algorithm is  $K[4]$ , which means that the length of the SWJK key is twice that of the original key. The key values generated by the SWJK algorithm  $[\gamma_{[r]}]$  will be different from the original key  $K[x]$ .

So, mathematically, for Equation (9), the following can be said:

$$[\gamma_{[r]}] \perp K[x] \quad (9)$$

The length and values of  $[\gamma_{[r]}]$  will be determined by  $K[x]$  but the values obtained from  $[\gamma_{[r]}]$  will be different from  $K[x]$ . When the key length of  $K[x]$  is longer, then the length of  $[\gamma_{[r]}]$  will be two times less than the actual key, which makes the SWJK algorithm innovative.

Various algorithms (Algorithms 1–5) have been developed to convert plaintext to ciphertext, as shown in Figure 6. These algorithms aim to derive a cipher key from a user-defined key and then use this cipher key on plaintext to encrypt the data. After that, the data are to be decrypted using the same cipher key. In Algorithm 1, the SWJK key generator algorithm was developed to convert a user-defined key into a cipher key. SWJK will convert the user-defined key into a key with a combination of different numbers, characters, and symbols, and the length of the cipher key will be significantly reduced to the original key length. Algorithm 2 was used to find the mid-point from plaintext using a randomized mid-point algorithm. The reason for finding the mid-point is to interlink the plaintext bits values with each other. A secure data encryption algorithm was developed

in Algorithm 3 to obtain ciphertext from plaintext, cipher key, and mid-point value. In Algorithm 4, text decryption was performed using a cipher key and ciphertext. The key generated by the SWJK algorithm (Algorithm 1) is used to decrypt the ciphertext. After that, the novelty of the key obtained from the SWJK algorithm is determined, for which cryptanalysis (Kasiski test algorithm) is used on cipher keys in Algorithm 5.

---

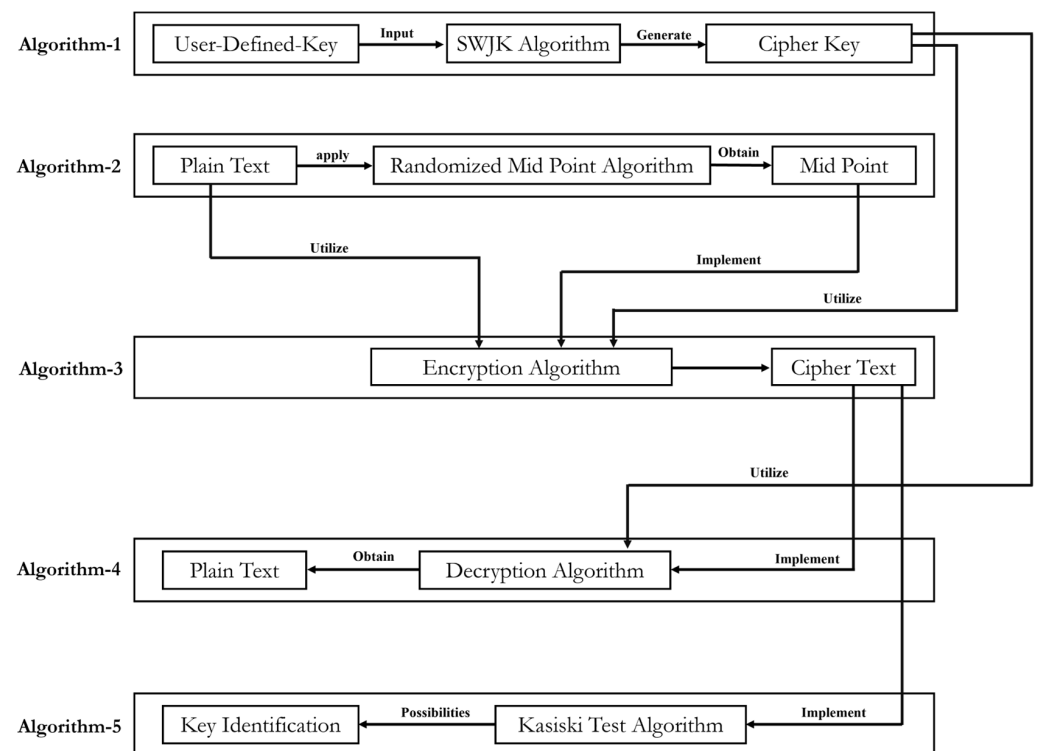
**Algorithm 1: SWJK Key Generator**


---

**Input:** User-Defined Key  $K[x]$

**Output:** SWJK Key  $K[j]$

1. Input a key  $K[x]$  length of 7.
  2. Divide the key  $K[x]$  into index form.  
 $[K] = [K0] [K1] [K2] \dots [Kp]$
  3. Convert each index value of  $[K]$  into ASCII form.  
 $K[n] = (AS0[K0]) (AS1[K1]) \dots (ASn[Kn])$
  4. Split  $K[n]$  indexes into odd  $[\delta_1]$  index and even  $[\delta_2]$  index form.
  5. Apply the SWJK algorithm on  $[\delta_1]$  for the values of Index  $[m]$  and Index  $[n]$ .
    - (a)  $f_{[m,n]}(s) = \sigma^* \bmod [x]$  on  $[\delta_1]$  values and obtained the values of Index  $[m]$  and Index  $[n]$ .
    - (b) Convert  $f_{[m,n]}$  values into 1x2-matrix form by using the equation  $ff_{[n]}^{[m]} = \begin{bmatrix} f_{m_1[n]} \\ f_{m_2[n]} \end{bmatrix}$
  6. Convert Step-4  $[\delta_2]$  ven-indexes into 2x2-matrix form.  $[\delta_{2x2}] = \begin{bmatrix} [i] & [j] \\ [k] & [l] \end{bmatrix}$
  7. Sum the values of Step-5(a) with Step-6 by using an equation  $[\gamma_r] = \int_{[q]}^{[p]} + ff_{[n]}^{[m]}$ .
  8. Convert step-7  $[\gamma_r]$  results in 1x4-matrix form.
  9. A key  $K[j]$  will be obtained.
- 



**Figure 6.** Algorithm flow of WEDEx-Kerberotic encryption and decryption system.

### 3.3. WEDEx-Kerberos Data Encryption System

A system for network authentication called Kerberos is often used to provide trustworthy authorization for users and services across unsafe connections, like the Internet. Kerberos' primary objective is to provide reliable authentication in a networked context so that users can securely authenticate their identities to one another without sending sensitive information over the Internet. It uses the key distribution centre (KDC), a dependable third-party service, and symmetric key cryptography.

The WEDEx-Kerberos encryption system is created to protect both the data and the key concurrently since it is not sufficient to secure either the key or the method for protecting the data. To begin, a key  $K[x]$  with a length of  $K[7]$  will encrypt the data. After that, the key  $K[x]$  will be subjected to the SWJK algorithm to produce the key  $K[j]$ , whose length will be  $K[4]$ , which is two times less than  $K[x]$ . After getting the key  $K[j]$ , the ASCII of plaintext  $P[n]$  and key  $K[j]$  will be determined, and both will be converted to Bit  $[P]$ . After converting  $K[j]$  and  $P[n]$  to Bits  $[P]$ , both will be XORed with each other with the help of Equation (10).

$$E = P \odot K \quad (10)$$

After XORing the values, the binary values  $[E]$  will be obtained, and then the *randomized seed mechanism* will be implemented on these binary values  $[E]$ . The original key  $K[x]$  will be taken for the *randomized seed mechanism*, and the *mid-point* of the key will be determined with the help of the *randomized seed mid-point algorithm*.

---

#### Algorithm 2: Randomized Mid-Point

---

**Input:** Key  $K[x]$

**Output:** Mid-Point

1. Count the key  $K[x]$  length.
  2. Calculate the total number of key  $K[x]$  indexes.  
 $K[x] = K[0] K[1] K[2] \dots K[6]$
  3. Calculate the *mid-point* value by using the equation  
 $[m] = (K[x] + 1)/2$
  4. IF  $[m] == [\text{EVEN\_INTEGER}]$ , THEN  
GOTO Step-5  
ELSE IF  $[m] == [\text{ODD\_INTEGER}]$ , THEN  
Apply  $[m] += [m]$
  5. A *mid-point* value will be obtained.
- 

After finding the *randomized seed mid-point*  $[m]$ ,  $[m]$  will be converted to an 8-bit binary, and then 4-bit splitting will be carried out to obtain " $\alpha$ " and " $\beta$ " values. After obtaining the " $\alpha$ " and " $\beta$ " values, with the help of Equation (11), secure encrypted text will be produced in binary form.

$$L = \alpha \upharpoonright E \upharpoonright \beta \quad (11)$$

A *bit-appending mechanism* will be applied to all binary values obtained from Equation (11). In the *bit-appending mechanism*, the 4 bits of " $\alpha$ " will be merged with the beginning of 4 bits of the XORed Text  $[E]$  to get an 8-bit binary result, as illustrated in Figure 7. Similarly, 4 bits from the previous text and 4 bits from the next text will be combined to form an 8-bit pair, resulting in a secure 8-bit binary text. When bits are concatenated, each bit depends on another bit. The entire ciphertext will be affected if a single value is changed. The unique aspect is that it will change and impact the whole text. When each value of the text is linked, and the length of the encrypted text is greater than the original text, the attacker will consider each value as an encrypted value and try to decrypt the value even though the length of the encrypted text is not the actual length the original text.

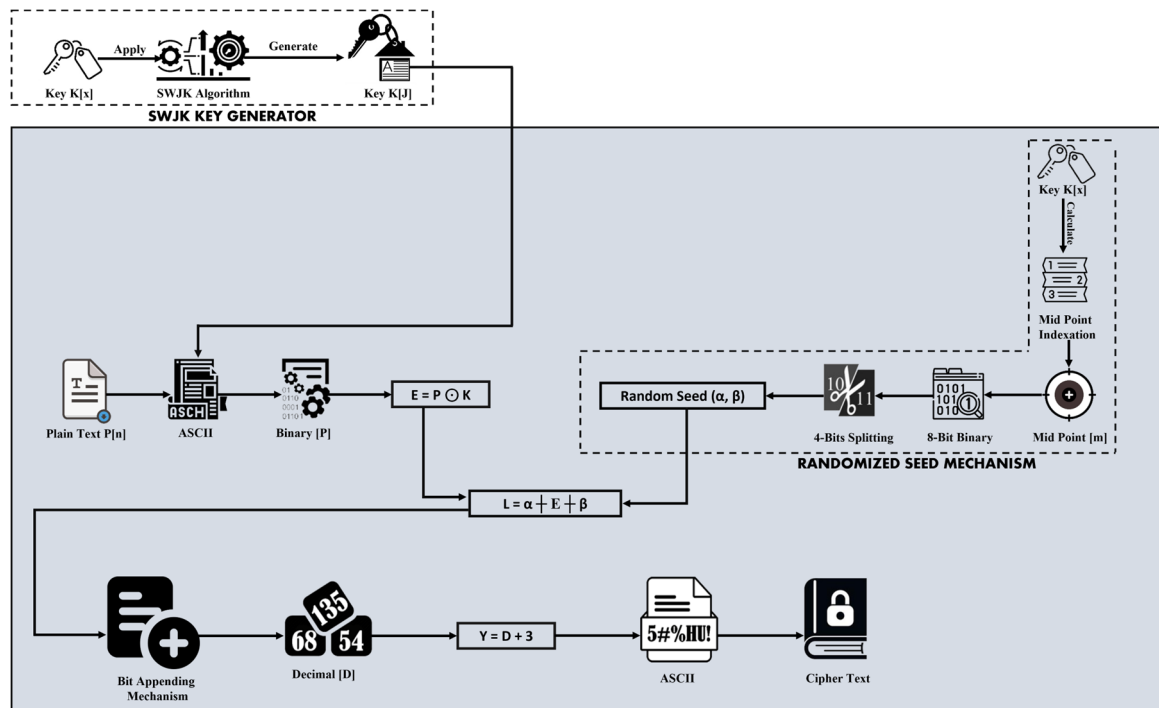


Figure 7. WEDEx-Kerberotic encryption mechanism.

After implementing the *bit appending mechanism*, the *bit appending* results will be converted to *decimal* [D]. ASCII values will be obtained using the formula  $Y = D + 3$  on the *decimal* [D] result. When this formula is applied, the resulting values [L] will be shifted to the *next 3-values* to the original values. When the attacker implements the attacking mechanism on the data, the attacker will consider the shifted values as the original values and try to decrypt the data, which will not be the original values. After getting its different values, all the values will be merged, and a text called ciphertext will be obtained. The key snatching technique cannot be implemented, nor can it be used to compromise the security of the data when the WEDEx-Kerberotic algorithm is used for data security. The key and the data are protected simultaneously with the aid of the suggested method, increasing key and data security.

#### Algorithm 3: Encryption

**Input:** Plaintext  $P[n]$

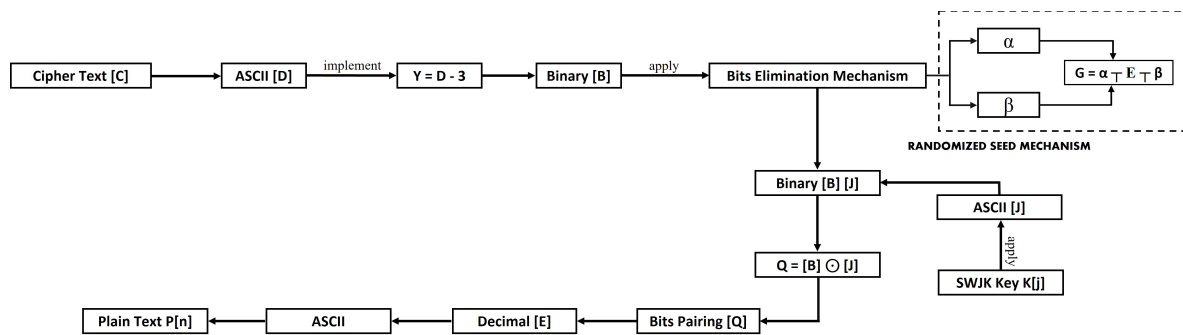
**Output:** Ciphertext

1. Plaintext  $P[n]$  and SWJK key  $K[j]$ .
2. Convert  $P[n]$  and SWJK key  $K[j]$  into ASCII form.
3. Convert each value of  $P[n]$  and  $K[j]$  into an 8-binary value form.
4. Find the random seed  $[\alpha, \beta]$  values using a randomized mid-point Algorithm.
5. Apply *bit appending mechanism* on " $L = \alpha \oplus E \oplus \beta$ ."
6. Convert each *bit appending value* into *decimal* [D] form.
7. Apply the value shifting formula " $Y = D + 3$ " in Step 6.
8. Find the ASCII value of [Y].
9. A ciphertext will be obtained.

#### 3.4. WEDEx-Kerberotic Data Decryption System

The WEDEx-Kerberotic technique encrypts the data, which will first be decrypted using a *ciphertext* [C]. Each character in *ciphertext* [C] will be changed into its corresponding ASCII [D] character. After obtaining the various ASCII values, the bit-reversing process will be used for each ASCII value, for which the formula  $Y = D - 3$  will be utilized. Each

value will be transformed to an 8-bit binary after being obtained through the *bit reversing mechanism*, as illustrated in Figure 8.



**Figure 8.** WEDEx-Kerberotic decryption mechanism.

All decimal numbers will be transformed into binary [B] form after being obtained. After that, a *bit's elimination mechanism* will be applied to these bits. The first 4-bit “α” of the start and 4-bit “β” of the end will be removed with the help of Equation (12).

$$G = \alpha \oplus E \oplus \beta \quad (12)$$

After that, *SWJK key*  $K[j]$  will be taken and converted to *ASCII*. The binary values [G] and [J] will then be obtained. [G] is the value that will be obtained from the ciphertext [C] mechanism, while [J] will be obtained from the *SWJK key*.

Equation (13) will help implement the *XORed mechanism* on *binary* values [B] and [J], and the results [Q] will be obtained in binary form.

$$Q = [G] \odot [J] \quad (13)$$

The binary outcome of the *XOR* will be implemented using a *bits pairing technique*. To acquire equal *ASCII* values, each value derived from *binary* pairings will be transformed into decimals. The result obtained from the equivalent values will be considered plaintext  $P[n]$ . Figure 8 illustrates a complete decryption mechanism.

---

#### Algorithm 4: Decryption

---

**Input:** Ciphertext

**Output:** Plaintext

1. Ciphertext [C]
  2. Transform each cipher (C) text value into ASCII (D) format.
  3. Apply the *bit reversing mechanism* “ $Y = [D] - 3$ ” in Step 2.
  4. Convert each value of *step-3* [Y] into 8-bit *binary form*.
  5. Apply the *bit-elimination mechanism* in Step 4.
  6. Eliminate the values of “α” and “β” using a *randomized seed mechanism*.
  7. *Binary values* [G] and [J] will be obtained from Step 6.
  8. *XOR mechanism* “ $Q = [B] \odot [J]$ ” will be applied to binary values.
  9. Apply the *bit pairing mechanism* in Step 8 and obtain *Decimal* [E] values.
  10. Change each decimal [E] value to its corresponding *ASCII* character.
  11. Plaintext  $P[n]$ .
-

**Algorithm 5:** Kasiski Test**Input:** Ciphertext**Output:** Key Prediction

1. Ciphertext
2. Determine the values of the repeated ciphertext
3. Determine indexes of repeated numbers
4. Calculate the length employing the Kasiski length method

Apply the " $Y = Y1 - Yn$ " formula to calculate the distance between the *first* and *xth* values  
 Determine the *greatest common division* among all distances

5. Calculate the length of the key using Step 4
6. IF  $CIPHER\_TEXT\_LENGTH = KEY\_LENGTH$ , then go to next step  
 ELSE ( $FIND\_KEY$ ) and go to step 9
7. Implement the index of coincidence  
 $Jc(Y) = (Favorable\ cases / Total\ Possible\ cases)$   
 $Jc(Y) = (Hi / Tc)$
8. Kasiski K
9. key
10. Exit

**4. Testing**

The derived key of the SWJK method is initially checked to determine the security of the data and the key. After that, a ciphertext will be produced by applying the key from the SWJK algorithm to the plaintext.

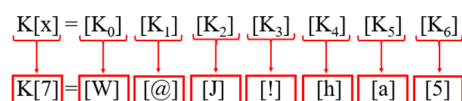
**4.1. SWJK Key Testing**

**Step 1:** First, a key  $K[x]$  is taken, illustrated in Figure 9, whose length is  $K[7]$ .

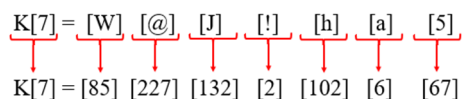
$K[7] = W@J!ha5$

**Figure 9.** Plaintext.

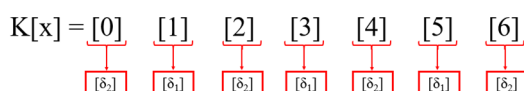
**Step 2:** Each  $K[x]$  value is converted into *index* form, as illustrated in Figure 10.

**Figure 10.** Plaintext indexes.

**Step 3:** After obtaining the *index* values of the *plaintext*, the *customized ASCII* value of each *index* value is obtained, as illustrated in Figure 11.

**Figure 11.** ASCII values.

**Step 4:** After getting the ASCII values, all indexes are transformed into even indexes and odd indexes, as illustrated in Figure 12.

**Figure 12.** Split odd and even index values.

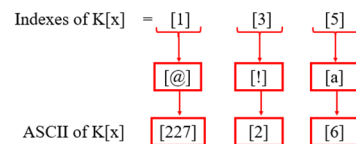
After splitting the values into *odd* and *even* indexes, both index values have to be concatenated separately.

Odd Indexes  $[\delta_1] = [1] [3] [5]$

Even Indexes  $[\delta_2] = [0] [2] [4] [6]$

**Step 5:** After splitting the values into *odd* and *even* indexes, the SWJK algorithm is implemented on *odd indexes*  $[\delta_1]$ .

**Step 5.1:** All values of odd indexes are first converted to the *index character* to identify *parallel* values of indexes, and then the equivalent of each value is determined, as illustrated in Figure 13.



**Figure 13.** Indexes parallel values.

**Step 5.2:** After obtaining the parallel values, Equation (14) will be implemented in step 5.1.

$$\phi_{[m\ n]}(s) = \sigma^* \bmod [x] \quad (14)$$

Where  $[m]$  is the ASCII value of Index  $[1]$  and Index  $[3]$ . Firstly, the Index  $[1]$  ASCII value will be used in the equation to get the *result-1*. After that, Index  $[3]$  ASCII value will be used in *testing 2* by the equation and get the *result-2*. Index  $[m]$  value will be placed in " $\sigma$ ."  $[n]$  is the static value that will be the same for Index  $[m]$  testing. Index  $[n]$  will be placed in " $\bmod[x]$ ." " $s$ " represents several time tests.

**Step 5.2.1:** First, Index  $[1]$  and Index  $[5]$  will be tested and will get the first result value.

$$\begin{aligned} \phi_{[1\ 5]}(1) &= (227)^* \bmod [6] \\ &= 3 \dots \text{result (i)} \end{aligned}$$

**Step 5.2.2:** After testing the Index  $[1]$  and Index  $[5]$ , Index  $[3]$  and Index  $[5]$  will be tested by using Equation (15), and a *second* result will be obtained.

$$\begin{aligned} \phi_{[3\ 5]}(2) &= (2)^* \bmod [6] \\ &= 4 \end{aligned} \quad (15)$$

**Step 5.2.2:** After getting the two values from Index  $[1][5]$  and Index  $[3][5]$ , both results values will be converted into *1x2 matrix form*.

$$\begin{aligned} \mathbb{F}_{[n]}^{[m]} &= \begin{bmatrix} \phi_{[1][5]}^{(1)} \\ \phi_{[3][5]}^{(2)} \end{bmatrix} \\ &= \begin{bmatrix} 3 \\ 4 \end{bmatrix} \end{aligned}$$

**Step 6:** After obtaining the results from *odd indexes*  $[\delta_1]$ , *even indexes* will be converted to *2x2-matrix form*.

$$\begin{aligned} \delta_{[2 \times 2]} &= \begin{bmatrix} [i] & [j] \\ [k] & [l] \end{bmatrix} \\ &= \begin{bmatrix} [0] & [2] \\ [4] & [6] \end{bmatrix} \\ &= \begin{bmatrix} [85] & [132] \\ [102] & [67] \end{bmatrix} \end{aligned}$$

After obtaining the  $2 \times 2$ -matrix values, Matrix  $\begin{bmatrix} \delta_{[2 \times 2]} \end{bmatrix}$ . It can also be represented as the following:

$$\begin{bmatrix} \delta_{[2 \times 2]} \end{bmatrix} = \int_{[q]}^{[p]} \int_{[q]}^{[2]} = \int_{[2]}^{[p]}$$

**Step 7:** The values of Step 6  $\int_{[q]}^{[p]}$  are summed with step-5.2.2 results  $\int_{[n]}^{[m]}$  by using Equation (16).

$$\begin{aligned} \gamma_{[r]} &= \int_{[q]}^{[p]} + \int_{[n]}^{[m]} \\ \gamma_{[r]} &= \begin{bmatrix} [85] & [132] \\ [102] & [67] \end{bmatrix} + \begin{bmatrix} 3 \\ 4 \end{bmatrix} \\ \gamma_{[r]} &= \begin{bmatrix} [88] & [136] \\ [105] & [71] \end{bmatrix} \end{aligned} \quad (16)$$

**Step 8:** Step-7 results  $\gamma_{[r]}$  are converted into  $1 \times 4$ -matrix form.

$$\gamma_{[r]} = [88 \ 136 \ 105 \ 71]$$

**Step 9:** Each value will be converted to its corresponding ASCII after being transformed into a  $1 \times 4$ -matrix form, and a key  $K[j]$  will be obtained in cipher form, as shown in Equation (17).

$$\begin{aligned} \gamma_{[r]} &= [\hat{e} \ \hat{i} \ \hat{u} \ u] \\ \text{Key } K[j] &= \hat{e} \ \hat{i} \ \hat{u} \ u \end{aligned} \quad (17)$$

#### 4.2. WEDEx-Kerberotic Encryption Algorithm Testing

**Step 1:** To encrypt the data, a dynamic key  $K[x]$  will be taken from the user and converted into cipher form using the WEDEx-Kerberotic algorithm.

$$\text{Key } K[x] = W@j!h@5$$

$$\text{Cipher Key } K_j = \hat{e} \ \hat{i} \ \hat{u} \ u$$

**Step 2:** After getting the key  $K[x]$  into cipher form from the SWJK algorithm, named cipher key  $K[j]$ , a simple text  $P[n]$  has then been taken.

$$\text{Plaintext } P[n] = N@DeEm_78$$

**Step 3:** After obtaining the plaintext  $P[n]$  and cipher key  $K[j]$ , as seen in Figure 14, each character of  $P[n]$  and  $K[j]$  is changed to its corresponding ASCII character.

$P[n]$	=	N	@	D	e	E	m	7	8
		196	227	52	70	68	182	213	83
		99							
$K[j]$	=	$\hat{e}$	$\hat{i}$	$\hat{u}$	u				
		88	136	105	71				

**Figure 14.** ASCII of plaintext and key.

**Step 4:** Each value is transformed to an 8-bit binary after being given the ASCII values for the plaintext  $P[n]$  and key  $K[j]$ , as illustrated in Figure 15.

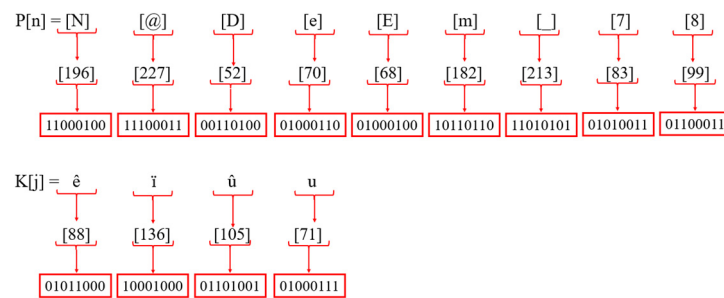


Figure 15. Binary values of plaintext and SWJK key.

**Step 5:** After obtaining the *binary* values, the *plaintext*  $P[n]$  and the *SWJK key*  $K[j]$  are *XORed* with each other using an equation “ $E = P \odot K$ ” to increase the data security, as illustrated in Figure 16.

$$\begin{aligned}
 P[n] &= 110001001110001100110100010001100100010010110110110101010101001101100011 \\
 K[j] &= 010110001000100001101001010001110101100010001000011010010100011101011000 \\
 X = P \odot K &= 100111000110101101011101000000010001110000111110101111000001010000111011
 \end{aligned}$$

Figure 16. XOR results.

**Step 6:** After obtaining the XORed results, the random key values ( $\alpha$ ,  $\beta$ ) have been obtained from the randomized key mechanism, for which the indexes of the key taken from the user are determined first.

$$\begin{aligned}
 K[x] &= [K_0] [K_1] [K_2] [K_3] [K_4] [K_5] [K_6] \\
 &= [0] [1] [2] [3] [4] [5] [6]
 \end{aligned}$$

There is a total of seven indexes  $K[x] = 7$  from  $[K_0]$  to  $[K_6]$ . After finding the indexes, the mid-point value is determined using Equation (18).

$$\begin{aligned}
 [m] &= ((K[x] + 1)/2) \\
 &= 4
 \end{aligned} \tag{18}$$

So, the *mid-point Index*  $[m] = 4$ , and the value of *Index*  $[4]$  is “!”. After finding the *mid-point* value  $[m]$  of *user-defined key*  $K[x]$ , the *ASCII* value of “!” is determined using a *customized ASCII* table, which is  $[2]$ .

$$[!] = [2]$$

After finding the *ASCII* value, an *8-bit binary* of *ASCII* value of “2” is determined.

$$\text{Binary of } 2 = [00000010]$$

After obtaining the *8-bit binary*, it is *split* into *two parts*. The first *four bits* are equated to “ $\alpha$ ,” while the last *four bits* are equated to “ $\beta$ .”

$$\alpha = 0000$$

$$\beta = 0010$$

**Step 7:** After getting the *random seed* ( $\alpha$ ,  $\beta$ ) values, *random seed* “ $\alpha$ ” is concatenated to the beginning of the XORed result  $[X]$ , while *Random Seed* “ $\beta$ ” is concatenated to the end of the XORed result  $[X]$ , as illustrated in Figure 17.

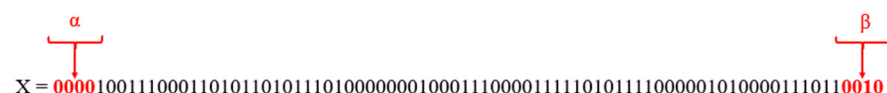


Figure 17. Random seed concatenation results.

**Step 8:** After concatenating the random seed values, the bit appending mechanism is implemented in step 7. In the bit appending mechanism, the 4 bits of the random seed “ $\alpha$ ” have been concatenated with the following 4 bits to form a pair of 8 bits. Similarly, an 8-bit pair is made by merging the previous four binary bits and the following 4. Each value will be linked using the bit-appending mechanism, as illustrated in Figure 18. If an attacker tampers with the values, then due to changing the values, the original text will be changed into a text that is entirely different from the original text.



Figure 18. Bit appending results.

Due to the random seed technique, the amount of encrypted text produced will be more than the original text, and each value will be linked to the others. The attacker will attempt to use the encryption technique on each cipher value while attempting to decode the data, but the characters of the cipher values will not have any link with the plaintext character.

**Step 9:** The 8-bit values obtained by the bit appending mechanism are transformed to decimal form, as illustrated in Figure 19.

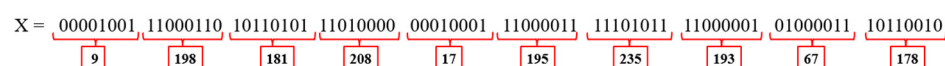


Figure 19. Decimal values.

**Step 10:** As demonstrated in Figure 20, after the different decimal values are collected, all values are swapped out for the following three values to separate the encrypted data from the original data, which is carried out using the formula  $D = D + 3$  to the step-9 decimal values.

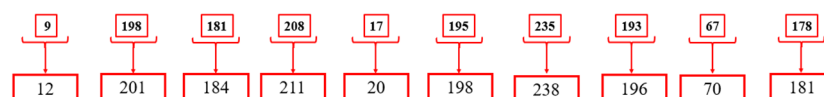


Figure 20. Shifted value result.

**Step 11:** As seen in Figure 21, all decimal values are changed to identical ASCII.

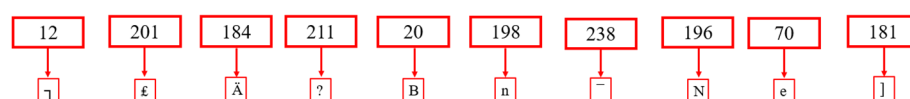


Figure 21. ASCII value of each decimal.

**Step 12:** After acquiring various ASCII values, all values are concatenated to create encrypted text.

$$\text{Ciphertext [C]} = \text{£Å?Bn¯Ne}$$

#### 4.3. WEDEx-Kerberotic Decryption Algorithm Testing

The data are decrypted using various procedures and methods, converting the Ciphertext to plaintext.

**Step 1:** A ciphertext [C] is taken to decrypt the data.

$$\text{Ciphertext [C]} = \text{£Å?Bn¯Ne}$$

**Step 2:** Following the ciphertext [C] collection, each character is converted into corresponding ASCII using a customized ASCII table, as illustrated in Figure 22.

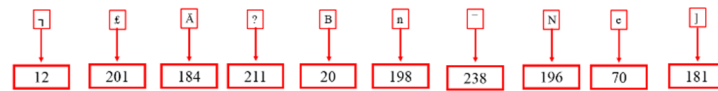


Figure 22. ASCII of each ciphertext.

**Step 3:** After acquiring the different ASCII values, the actual values are produced by applying a bit-shifting method [Y] to them.

$$Y = D - 3$$

Decimal values are replaced by previous *third index values* with the help of [Y], as illustrated in Figure 23.

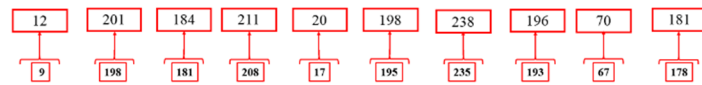


Figure 23. Values shifting.

**Step 4:** Each value acquired by the values shifting technique is transformed to an 8-bit binary, as demonstrated in Figure 24.

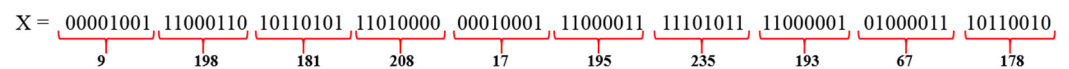


Figure 24. Binary of each decimal.

**Step 5:** After obtaining the 8-bit binary values, a *bit-elimination mechanism* [G] is applied to eliminate the random seed [G] values, as illustrated in Figure 25.

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

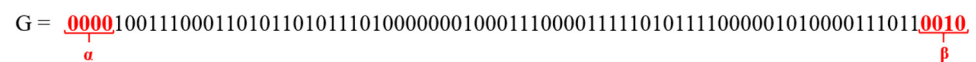


Figure 25. Random seed elimination.

**Step 6:** After eliminating the random seed values, the SWJK key  $K[j]$  is taken, which is converted to ASCII and converted to binary form, as illustrated in Figure 26.

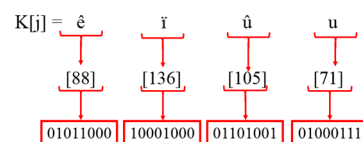


Figure 26. Binary values of SWJK key.

**Step 7:** Following acquiring the binary values, key  $K[j]$  and text [G] are XORed to produce a binary form text, as illustrated in Figure 27.

$$\begin{aligned} [G] &= 110001001110001100110100010001100100010010110110101010101001101100011 \\ K[j] &= 01011000100010000110100101000111010110001000100001010010100011101011000 \\ Q = [B] \odot K[j] &= 10011100011010110101110100000010001110000111110101111000001010000111011 \end{aligned}$$

Figure 27. XOR results of key and text.

**Step 8:** Different decimal values [E] are obtained by creating 8-bit pairs of the bit values obtained from the XORed result, as illustrated in Figure 28.



```

[WED_KER_SERVER]-KEY: N@d$38

[DATABASE_AUTHENTICATION]# File Searching...
[DATABASE_AUTHENTICATION]# Almost Done...
[Status]# Please Wait...
[Status]# Invalid Key...

[WED_KAR_AGAIN]_KEY: ç2fÜBE3Ó÷
[Status]# Matched Successfully...

[WED_KER_SYS]_CIPHER-TEXT:> D×κiOuĖĖÅκτ£εEf□85öO~D#È%£ûRα|íôι|ø□Ð-6£|εQ
WED_KER_SYS]_TEXT:> &ÜJ uĖĖÅκJ OuZĖwöεEf□85öO6£|

```

**Figure 31.** Inauthentic key implementation result.

Data security will be divided into two phases when an algorithm like this is utilized. The first phase is based on key matching, while the second is based on cryptographic algorithms. Data stored in the cloud may be protected against any number of threats using such an algorithm.

#### 4.6. Testing Results

Table 2 displays different results obtained after the development of the tool. Firstly, various simple texts are taken that are represented with [P], and the length of the plaintext is determined. Plaintext length is represented with P[n]. After obtaining the plaintext, various user-defined keys are obtained, represented by [K], and then the length of the user-defined key K[x] is determined. The reason for finding the key length used on different plaintexts is to compare the lengths when the plaintext and key are applied to the algorithm and to find out how much difference there is between the original length and the length obtained from the algorithm. When a user-defined key [K] is applied to the SWJK algorithm, cipher keys of different lengths are obtained, denoted by length K[j]. The length of the cipher key is much less than the length of the user-defined key, up to 40%. This means that when the user-defined key is applied to the SWJK algorithm. The SWJK algorithm will reduce the length of the user-defined key [K]. When the user gets a key, it cannot be guessed or retrieved using snatching mechanisms like cryptanalysis. After obtaining the different cipher keys, the key is applied to the plaintext [P], and the different ciphertext is obtained. The length of plaintext [P] is less than that of ciphertext [N]. This means that when we implement plaintext in the proposed algorithm, the length obtained by the proposed algorithm will also change, and the original length will be different from the ciphertext length. When the length of the plaintext is different from the ciphertext and the length of the user-defined key is also different from the cipher key, it means that it is a better innovation, and this framework can be used to protect data.

**Table 2.** All testing results.

Testing	Original Text		Proposed Algorithm Results					
	Plaintext [P]	Plaintext Length P[n]	User-Defined Key [K]	User-Defined Key Length K[x]	Cipher Key	Cipher Key Length K[j]	Ciphertext [C]	Ciphertext Length C[n]
1	Th!s Research @rticle is WriTTen bY Z@hra	41	N@deeM852	9	&\$7Fp	5	D×κiOuĖĖÅκτ£εEf□85öO~D#È%£ûRα íôι ø□Ð-6£ εQ	42
2	mY_b@ck B0ne !s mY POWeR	24	@!Arsh@d	9	ÊΔáG7	5	TÉOiOuS8rwùαG/Æ TUS/Æ3wÍ2D\$	25
3	@ Secure Crypt0logy P@per	25	Muh@MmaD NadeEM	15	ç2fÜBE3 Ó÷	9	&ÜVJ OuZĖwöε£2UĖ OrτĖtø/Æò£κ}	25
4	W@j!h@ Z@hRa	12	Z@iN!	5	_Ė4CsÑ	3	&ÜJ OG1SQ&ĖrJ Q	13

When the proposed algorithm is used for data security, it will have two significant advantages. The first advantage is that the length of the plaintext will differ from that of the ciphertext. When any snatching algorithm is applied to the ciphertext, a value parallel to

each cipher value will be completely irrelevant. The second advantage is that the proposed algorithm will significantly reduce the length of the original key, and the resulting cipher key will be completely unrelated to the original key. When key snatching or guessing is carried out using snatching techniques like cryptanalysis, the attacker gets keys that have no relation to the proposed algorithm key. When the proposed paper algorithm is used for cloud data security, no matter how efficiently the attacker develops the algorithm, it will be impossible to decrypt the data and the key.

#### 4.7. Cryptanalysis

A cryptanalysis method has been used to determine the novelty of the proposed algorithm, which first used “Cryptanalysis by Plaintext and Ciphertext” and then “Key Cryptanalysis”.

##### 4.7.1. Plaintext and Ciphertext Cryptanalysis

The efficacy of the proposed method is evaluated by cryptanalysis of plaintext and ciphertext, where the lengths of plaintext  $P[n]$  and ciphertext  $[C]$  are first ascertained. The key is accessible using the cryptanalysis approach when the plaintext and ciphertext have the same length. If the sizes of both are not the same, it is impossible to implement any cryptanalysis method, and the key cannot be anticipated. If both lengths are the same, it is easy to estimate the key accurately.

The suggested technique has been applied to these values, as illustrated in Figure 32, and various key results have been achieved, as illustrated in Table 3. The cryptanalysis of plaintext and ciphertext uses distinct plaintext values.

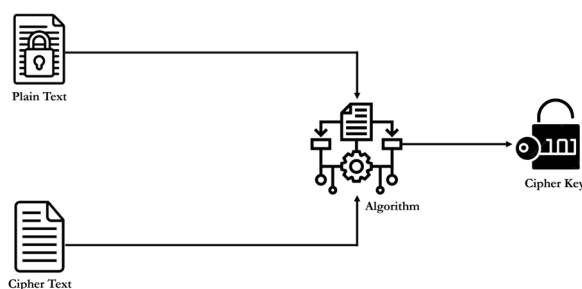


Figure 32. Plaintext and ciphertext cryptanalysis.

Table 3. Results of plaintext and ciphertext cryptanalysis.

Sr#	Plaintext	Text Length $P[n]$	Ciphertext Length $[C]$	Length Equalization $(P[n], [C])$	Cryptanalysis Algorithm	Key Prediction Possibilities
1	Th!s Research @rticle is WriTTen bY Z@hra	41	42	$T \neq X$	No	No
2	mY_b@ck B0ne !s mY POWeR	24	25	$T \neq X$	No	No
3	@ Secure Crypt0logy P@per	25	26	$T \neq X$	No	No
4	W@j!h@ Z@hRa	12	13	$T \neq X$	No	No

The length of the ciphertext  $[C]$  and plaintext  $P[n]$  have been determined after obtaining different results. The length of the plaintext and the ciphertext cannot be identical since the size of the ciphertext exceeds the length of the plaintext owing to the implementation of the suggested method. If the sizes differ, neither the key prediction nor the cryptanalysis procedure may be used. The length of the plaintext and the encrypted text must match to forecast the key.

#### 4.7.2. Key Cryptanalysis

A cryptanalysis algorithm called the Kasiski test is implemented to generate the key from the data. The Kasiski test, as shown in Figure 33, may assist in obtaining the Kky, and the snatching algorithm is employed.

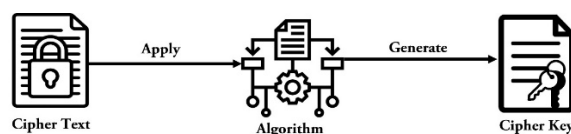


Figure 33. Ciphertext cryptanalysis.

A cipher key  $K[j]$  and user-defined key  $K[x]$  are first taken to get the key from the Kasiski test, and then all the values in the ciphertext used repeatedly in the ciphertext are found. After determining the repeatable values, the index of each repeatable value is indexed. Table 4 shows the results of applying the Kasiski length method to the indexation of the repeated value, by which the distance from the first value to the  $x$ th value is determined. Distance is represented by “ $y$ .” After finding the distance, GCD (greatest common division) is implemented on “ $y$ .” The value obtained from GCD is considered the *key length*. As per cryptanalysis, key length can be possible when the cipher and original key lengths are the same. According to the SWJK algorithm, the length of the cipher and original keys is different.

Table 4. Cryptanalysis of ciphertext.

Proposed Algorithm					Kasiski Test		
Testing	User-Defined Key Length $K[x]$	Cipherkey Length $K[j]$	Repeatable Values Distance $[y]$	GCD Possibilities	Length Equalization ( $K[x], K[j]$ )	Index of Coincidence Algorithm	Key Identification
1	7	4	2	Not	$V \neq K$	No	No
2	10	7	4	Not	$V \neq K$	No	No
3	12	7	4	Yes	$V \neq K$	No	No
4	8	5	3	Not	$V \neq K$	No	No

An index of coincidence can be implemented if the key length and ciphertext length are equal. Key identification is impossible if the key and ciphertext lengths are unequal. Favorable cases and possible cases are identified in the index of coincidence. Favorable cases mean the prediction of equal random numbers, whereas total cases refer to the number of feasible key length calculations. The ciphertexts generated by the proposed approach are all entirely distinct from the input. Because a technique was used to obtain the ciphertext, replacing the original values with other values and linking all the words together, if a value is repeated repeatedly, it does not mean that the key prediction is possible. The key can only be derived from the ciphertext when each value is replaced with an ASCII value. The originality of the technique provided in this paper is that it is impossible to exploit when the values are modified, no matter how proficiently the attacker creates a cryptanalysis program.

#### 4.8. Comparative Cost Analysis

A WEDEx-Kerberotic system has been developed to secure data from attackers. The novelty of the proposed algorithm is that the key and data are encrypted simultaneously. The cipher key is obtained using the SWJK algorithm. Each encrypted ciphertext value obtained from the proposed algorithm is interconnected. Tampering with any single value can affect the structure of the entire ciphertext. The key obtained from each text is unique, but its use is limited and can only be used on that text. Despite using techniques like cryptanalysis, decrypting the ciphertext obtained from the proposed algorithm will not be easy.

The proposed tool has been implemented on multiple servers to evaluate the algorithm's efficacy, as illustrated in Table 5. Testing is performed across various operating systems (OS) on each server. The initial stage involved finding the lengths of the plaintext, key, and ciphertext. After this, the memory allocation of plain and encrypted text is determined. After determining memory allocation, the next step involves calculating the time allocation required for the encryption and decryption of the data.

**Table 5.** Cost analysis.

Servers	O.S	Performance			Space Complexity		Time Complexity	
		Plaintext Length	Cipher Key Length	Ciphertext Length	Plaintext Memory Allocation	Ciphertext Memory Allocation	Data Encryption Time (sec)	Data Decryption Time (sec)
1	MsWindow 7	41	5	42	30.17 KB	32.27 KB	10.35	12.41
2	MsWindow 8	24	5	25	28.52 KB	30.12 KB	07.21	08.12
3	MsWindow 8	25	9	25	30.17 KB	32.81 KB	10.24	22.85
4	MsWindow 10	12	3	13	21.14 KB	23.55 KB	04.58	05.11
5	MsWindow 10	37	6	38	27.40 KB	29.57 KB	09.37	11.19
6	MsWindow 10	26	4	27	24.43 KB	26.35 KB	08.53	10.14
7	MsWindow 11	65	4	66	40.31 KB	42.48 KB	13.25	15.12
8	MsWindow 11	47	5	48	32.49 KB	34.29 KB	11.19	13.45

## 5. Comparative Analysis

Various academics have devised unique algorithms to keep sensitive information safe. Researchers reviewed the literature and debated the most effective methods. A hybrid algorithm was created by academics who tweaked existing algorithms.

M. A. Al-Shabi [17] reviewed cryptographic algorithms, compared their methodologies, and found that symmetric algorithms are more secure and reliable than asymmetric ones. This research compared the algorithms, and the best strategies were presented. However, instead of using the best techniques for cloud data security, a safe algorithm was not constructed.

Musa et al. [18] devised a Hill cipher technique to secure data against man-in-the-middle attacks that transformed plaintext to ciphertext using a standard ASCII table. This paper's issue is data decryption employing the user-defined key and Hill cipher method.

Hossaim et al. [19] used three data security techniques to create a ciphertext using three static keys. This paper's issue is that each key must use the same technique to decrypt data. Key use is hard to memorize. Instead of three keys, one safe key may protect data.

In Paper [20], the attacker may easily access data after cryptanalyzing the key. The researchers secured the data using a static key on plaintext. Instead of using a key, researchers should build a key-securing method and apply it to the Vigenère cipher technique to get a ciphertext.

In [21], the researchers made alterations to a Hill cipher algorithm and created a modified Hill cipher chain method. The main objective of this modification was to encode only the main key. The problem of this paper is that the security of the primary key alone is not enough unless the key is protected as well as the data.

Tan et al. [22] examined Ceaser and Vigenère encryption algorithms to protect data and created a hybrid algorithm by combining the best methods. To protect the data, a static key was applied to the Ceaser cipher algorithm to get a cipher result, which was then applied to the Vigenère cipher using the same key to get a ciphertext. Instead of applying multiple algorithms' outputs on each other, an efficient and trustworthy method might boost data security, which is this paper's challenge.

### *The Novelty of Proposed Work*

The proposed paper algorithm can be used to protect data from attackers and can also help users to keep data safe. In papers [18–22], the researchers developed several security algorithms in which the data were encrypted and decrypted using a single key, as shown in Table 6, and whether the key was static or dynamic. No article has developed such an algorithm to simultaneously protect key and text data. The data are encrypted and

decrypted in the proposed work using a cipher key. Each user-defined key generates a cipher key that can only be used for that text. When the key is additionally protected, data security can be further increased. When data security is performed using a customized ASCII table instead of a standard ASCII table, which is only done in the proposed work, it will be difficult for an attacker to discover the parallel indexes of each ciphertext. It is not enough to create a new algorithm using best practices from several publications to protect data. Cloud data cannot be secured unless a method is devised to encrypt both the ciphertext and the key simultaneously. Just as the ciphertext algorithm must be secure to protect the data, the key used on the ciphertext is also essential to be secure. Once an attacker has access to a user-defined key, the attacker attempts to spoof that key on any connected data. When the key used in each data is different and is a combination of different characters, key accessibility will be impossible, nor can one key be used on another data, which can be a complete and reliable security mechanism for cloud security.

**Table 6.** Comparative analysis.

Sr#	1	2	3	4	5	Proposed Work
Reference No.	[18]	[19]	[20]	[21]	[22]	
Used Keys	No	Static	Static	Static	Dynamic	Cipher key
Key Generating Mechanism	No	No	No	No	generated key from Hill matrix	SWJK algorithm
Encryption Time	No	No	No	No	No	Yes
Decryption time	No	No	No	No	No	Yes
ASCII table	Standard	Standard	Standard	Standard	Standard	Customized
Proposed Algorithm	Hill cipher algorithm	Merge Caesar cipher, Stream cipher, and Playfair	Modified Vigenère cipher algorithm	Modified Hill cipher algorithm and developed Hill cipher chain algorithm	Radix-64 Bit and Hill matrix	WEDEX-Kerberotic system
Novelty	Prevent data from man-in-the-middle attacks	Secured data with three different keys	Prevent the key from cryptanalysis attacks	Provided security to primary key	Twice ciphertext than the actual text	Cipher key, data authentication system, data encryption by WEDEX-Kerberotic system
Gaps	The algorithm did not use a key and can be easily decrypt	Difficulty in implementing the same key on the algorithm where the key is required	Secure key can prevent the cloud from cryptanalysis attacks	Only primary key security is not sufficient for data encryption.	Cryptanalysis possibility on Hill matrix algorithm	Identified all existing problems
Proposed Paper Solutions	Used SWJK Cipher key, not easy to decrypt	Secure data with a single cipher key	Cipher key generate from SWJK algorithm	Secure key and data at the same time.	Cryptanalysis is not possible	Solved all problems

When an attacker attempts to steal data, the attacker's initial step is to get the key, for which the attacker uses snatching methods like cryptanalysis. In this article, the key has been secured with the help of the SWJK method, the objective of which is to obtain a key that had no connection to the original key and to reduce the length of the cipher key by two times from the length of the original key. After securing the key, the data are encrypted, for which the WEDEX-Kerberotic system is developed. The purpose of that is to provide authenticity to the cipher data. When an attacker receives the cipher data, the attacker tries to perform pattern-matching attacks on the data. Data security can be increased if the validity of the key is checked before applying the keys to the ciphertext because the keys can be applied to the encrypted data only when the key is valid. When the incoming

key is the same as the KSS key, the key is automatically applied to the ciphertext, whether the incoming key is valid for that ciphertext or not. Data may be confidential when such a technique is used for cloud data; no matter how efficient the algorithm is, the security of the data cannot be broken, which is the novelty of the proposed algorithm.

## 6. Conclusions

After developing the WEDEx-Kerberotic framework, it is determined that when the proposed framework is used to secure cloud data, all user-defined keys and data can be secured simultaneously. The length of the ciphertext obtained by the algorithm is less than that of the original plaintext, which is more than 10%. The user-defined key is secured by applying the SWJK algorithm to it. The length of the key obtained by the SWJK algorithm is less than 40% of the original key and is a combination of different letters, symbols, and numbers. When the cipher key is much shorter than the original key and the cipher key is composed of a combination of different characters, it will be impossible for an attacker to access it. Cryptanalysis is applied to different cipher keys to determine the key's authenticity. It is determined that no snatching techniques, such as cryptanalysis, can be implemented on the key obtained from the proposed algorithm, nor can accessing the key be possible. When the proposed algorithm is used to secure the cloud data, the key can be secured along with the data. When an attacker tries to attack the data, they can neither access the key nor retrieve the data in its original form, which can be a secure algorithm for cloud data.

In the future, a semi-supervised clustering algorithm will be developed to secure the data. This algorithm will be used to assess the authenticity of the users. It will transmit the incoming user packets to label and unlabeled clustering based on authenticity and apply a security mechanism. After that, we will also work on game theory for more robust analysis and security probabilistic testing.

**Author Contributions:** S.W.Z., conceptualization; M.N., methodology; A.A. (Ali Arshad), investigation and supervision; S.R., review; W.A., editing; M.A.B., software; A.A. (Amerah Alabrah), funding and editing. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by the Researchers Supporting Project number (RSP2024R476), King Saud University, Riyadh, Saudi Arabia.

**Data Availability Statement:** Data are contained within the article.

**Acknowledgments:** This research was supported by the NASTP Institute of Information Technology, National University of Technology and King Saud University.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Gundu, S.R.; Panem, C.; Vijaylaxmi, J. *A Glance View on Cloud Infrastructures Security and Solutions*. *Conversational Artificial Intelligence*; Wiley: Hoboken, NJ, USA, 2024; pp. 1–15. [\[CrossRef\]](#)
2. Pratyush, K.; Prasad, V.K.; Mehta, R.; Bhavsar, M. A Secure Mechanism for Safeguarding Cloud Infrastructure. In Proceedings of the International Conference on Advancements in Smart Computing and Information Security, Rajkot, India, 1–2 December 2023; Springer Nature: Cham, Switzerland, 2022; pp. 144–158.
3. Alazaidah, R.; Al-Shaikh, A.; Al-Mousa, M.R.; Khafajah, H.; Samara, G.; Alzyoud, M.; Al-Shanableh, N.; Almatarneh, S. Website phishing detection using machine learning techniques. *J. Stat. Appl. Probab.* **2024**, *13*, 119–129.
4. Jangjou, M.; Sohrabi, M.K. A comprehensive survey on security challenges in different network layers in cloud computing. *Arch. Comput. Methods Eng.* **2022**, *29*, 3587–3608. [\[CrossRef\]](#)
5. Arunkumar, M.; Ashokkumar, K. A review on cloud computing security challenges, attacks and its countermeasures. *AIP Conf. Proc.* **2024**, *3037*, 020047.
6. Jimmy, F.N.U. Cyber security Vulnerabilities and Remediation Through Cloud Security Tools. *J. Artif. Intell. Gen. Sci. (JAIGS) ISSN* **2024**, *3006*, 196–233.
7. Zargar, S.T.; Joshi, J.; Tipper, D. A survey of defense mechanisms against distributed denial of service (DDOS) flooding attacks. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 2046–2069. [\[CrossRef\]](#)

8. Gu, Y.; Li, K.; Guo, Z.; Wang, Y. Semi-supervised K-means DDoS detection method using hybrid feature selection algorithm. *IEEE Access* **2019**, *7*, 64351–64365. [\[CrossRef\]](#)
9. Abdulhamid, S.M.; Shuaib, M.; Osho, O. Comparative Analysis of Classification Algorithms for Email Spam Detection. *Int. J. Comput. Netw. Inf. Secur.* **2018**, *1*, 60–67. [\[CrossRef\]](#)
10. Mohammed, C.M.; Zeebaree, S.R. Sufficient comparison among cloud computing services: IaaS, PaaS, and SaaS: A review. *Int. J. Sci. Bus.* **2021**, *5*, 17–30.
11. Ali, M.; Jung, L.T.; Sodhro, A.H.; Laghari, A.A.; Belhaouari, S.B.; Gillani, Z. A Confidentiality-based data Classification-as-a-Service (C2aaS) for cloud security. *Alex. Eng. J.* **2023**, *64*, 749–760. [\[CrossRef\]](#)
12. Butt, U.A.; Amin, R.; Mehmood, M.; Aldabbas, H.; Alharbi, M.T.; Albaqami, N. Cloud Security Threats and Solutions: A Survey. *Wirel. Pers. Commun.* **2023**, *128*, 387–413. [\[CrossRef\]](#)
13. Aoudni, Y.; Donald, C.; Farouk, A.; Sahay, K.B.; Babu, D.V.; Tripathi, V.; Dhabliya, D. Cloud security based attack detection using transductive learning integrated with Hidden Markov Model. *Pattern Recognit. Lett.* **2022**, *157*, 16–26. [\[CrossRef\]](#)
14. Palanisamy, C.; Kumaresan, T.; Varalakshmi, S. Combined techniques for detecting email spam using negative selection and particle swarm optimization. *Int. J. Adv. Res. Trends Eng. Technol.* **2016**, *3*, 1102.
15. Upadhyay, D.; Zaman, M.; Joshi, R.; Sampalli, S. An efficient key management and multi-layered security framework for SCADA systems. *IEEE Trans. Netw. Serv. Manag.* **2021**, *19*, 642–660. [\[CrossRef\]](#)
16. Newman, S. Under the radar: The danger of stealthy DDoS attacks. *Netw. Secur.* **2019**, *2*, 18–19. [\[CrossRef\]](#)
17. Al-Shabi, M.A. A Survey on Symmetric and Asymmetric Cryptography Algorithms in information Security. *Int. J. Sci. Res. Publ. (IJSRP)* **2019**, *9*, 576–589. [\[CrossRef\]](#)
18. Musa, A.; Mahmood, A. Client-side cryptography based security for cloud computing system. In Proceedings of the 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), Coimbatore, India, 25–27 March 2021; pp. 594–600.
19. Hossain, M.E. Enhancing the security of caesar cipher algorithm by designing a hybrid cryptography system. *Int. J. Comput. Appl.* **2021**, *183*, 55–57. [\[CrossRef\]](#)
20. Akanksha, D.; Samreen, R.; Niharika, G.S.; Shruthi, A.; Kiran, M.J.; Venkatramulu, S. A hybrid cryptosystem based on modified vigenere cipher and polybius cipher. *EPRA Int. J. Res. Dev.* **2022**, *7*, 2455–7838.
21. Sun, H.; Grishman, R. Lexicalized dependency paths based supervised learning for relation extraction. *Comput. Syst. Sci. Eng.* **2022**, *43*, 861–870. [\[CrossRef\]](#)
22. Tan, C.M.S.; Arada, G.P.; Abad, A.C.; Magsino, E.R. A hybrid encryption and decryption algorithm using caesar and vigenere cipher. *J. Phys. Conf. Ser.* **2021**, *1997*, 012021. [\[CrossRef\]](#)
23. Arshad, A.; Nadeem, M.; Riaz, S.; Zahra, S.; Dutta, A.; Alzaid, Z.; Alabdan, R.; Almutairi, B.; Alaybani, S. Hill Matrix and Radix-64 Bit Algorithm to Preserve Data Confidentiality. *Comput. Mater. Contin.* **2023**, *75*, 3065–3089. [\[CrossRef\]](#)
24. Singh, V.; Pandey, S.K. Revisiting cloud security threats: Replay attack. In Proceedings of the 2018 4th International Conference on Computing Communication and Automation (ICCCA), Greater Noida, India, 14–15 December 2018; pp. 1–6.
25. Tadapaneni, N.R. Cloud computing security challenges. *Int. J. Innov. Eng. Res. Technol.* **2020**, *7*, 1–6.
26. Zaman, A.; Safarinejadian, B.; Birk, W. Security Analysis and Fault Detection Against Stealthy Replay Attacks. *Int. J. Control* **2022**, *95*, 1562–1575. [\[CrossRef\]](#)
27. Thirumavalavasethurayar, P.; Ravi, T. Implementation of Replay Attack in Controller Area Network Bus using Universal Verification Methodology. In Proceedings of the 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), Coimbatore, India, 25–27 March 2021; pp. 1142–1146. [\[CrossRef\]](#)
28. Nadeem, M.; Arshad, A.; Riaz, S.; Band, S.S.; Mosavi, A. Intercept the Cloud Network From Brute Force and DDoS Attacks via Intrusion Detection and Prevention System. *IEEE Access* **2021**, *9*, 152300–152309. [\[CrossRef\]](#)
29. Bentil, F.; Lartey, I. Cloud Cryptography—A Security Aspect. *Int. J. Eng. Res. Technol. (IJERT)* **2021**, *10*, 2278–0181.
30. Supiyanto; Mandowen, S. Advanced hill cipher algorithm for security image data with the involutory key matrix. *J. Phys. Conf. Ser.* **2021**, *1899*, 012116. [\[CrossRef\]](#)
31. Elsaedy, A.; Jamalipour, A.; Munasinghe, K. A Hybrid Deep Learning Approach for Replay and DDoS Attack Detection in a Smart City. *IEEE Access* **2021**, *9*, 154864–154875. [\[CrossRef\]](#)
32. Nadeem, M.; Arshad, A.; Riaz, S.; Zahra, S.; Dutta, A.; Almutairi, S. A Secure Architecture to Protect the Network from Replay Attacks during Client-to-Client Data Transmission. *Appl. Sci.* **2022**, *12*, 8143. [\[CrossRef\]](#)
33. Bharath, K.P.; Kumar, M.R. New Replay Attack Detection Using Iterative Adaptive Inverse Filtering and High Frequency Band. *Expert Syst. Appl.* **2022**, *195*, 116597. [\[CrossRef\]](#)

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.