Educational Administration: Theory and Practice
2024, 30(5), 3911-3922
ISSN: 2148-2403
https://kuey.net/

Research Article

Analyze the New Data Protection Mechanism to Maximize Data Availability without Having Compromise Data Privacy

Mohamed Azharudheen A[1*] , Dr.Vijayalakshmi V[2]

[1*]Research Scholar, PG & Research Department of Computer Science, Government Arts College(Grade-I),(Affiliated to Bharathidasan University) Ariyalur 621 713, Tamilnadu, India, azhar.scas@gmail.com, 9994478437
[2]Assistant Professor, PG & Research Department of Computer Science, Government Arts College(Grade-I),(Affiliated to Bharathidasan University) Ariyalur 621 713, Tamilnadu, India, v.viji08@yahoo.co.in, 9788170037

Citation: Mohamed Azharudheen A , Dr.Vijayalakshmi V, (2024), Analyze the New Data Protection Mechanism to Maximize Data Availability without Having Compromise Data Privacy, Educational Administration: Theory and Practice, 30(5), 3911-3922
Doi: 10.53555/kuey.v30i5.3548

ARTICLE INFO | **ABSTRACT**

This study delves into innovative data protection mechanisms designed to optimize data availability while maintaining data privacy. In the current era characterized by vast amounts of digital information, finding a delicate balance between accessibility and safeguarding sensitive data is crucial. Conventional approaches to data protection often involve compromises between availability and privacy. This research tackles this challenge by exploring state-of-the-art strategies that aim to improve data availability while upholding strict privacy standards.The research adopts a multidisciplinary approach, integrating perspectives from computer science, cryptography, and privacy engineering. Through a thorough analysis of emerging technologies and methodologies, the study seeks to identify and assess mechanisms that alleviate the inherent tension between data availability and privacy concerns. Special emphasis is placed on advancements such as homomorphic encryption, differential privacy, and federated learning, which show promise in revolutionizing data protection paradigms.Additionally, the research evaluates the practical implications and implementation challenges associated with these novel mechanisms. By scrutinizing real-world scenarios and case studies, the study aims to offer practical recommendations for organizations looking to implement robust data protection strategies. Ultimately, this research contributes to the ongoing discourse on the evolving landscape of data security, providing insights that can guide the development of policies and technologies conducive to maximizing data availability without compromising privacy.

Keywords: Innovative Data Protection; Maximizing Data Availability; Privacy-First Data Mechanism; Enhanced Data Security;Balancing Data Availability and Privacy.

1. INTRODUCTION

The era of pervasive computing has seen a proliferation of smart devices designed with constraints in memory, computing power, and battery supply. Users utilize these devices to interact with cloud storage for data storage, necessitating robust data security measures to ensure efficient storage and retrieval (Abadi *et al.,* 2013).
However, ensuring data security involves tradeoffs for users, as high-level security often comes with increased costs. Depending on the application and data criticality, users must make tradeoffs. Traditional encryption methods pose threats due to potential vulnerabilities in algorithm implementation, relying heavily on the security of encryption and decryption keys. Key management involves tasks such as generation, distribution, storage, revocation, and verification, with side-channel attacks being a common threat (Kolomvatsos*et al.,* 2015).

Side-channel attacks, like Cross Virtual Machine (Cross VM) attacks on the AES algorithm, exploit information from the physical implementation of cryptographic algorithms on computer systems, specifically targeting encryption keys. The generation and management of keys involve various methods, including separate keys for users and Cloud Service Providers (CSPs), single shared keys, and key splitting, all of which entail computational complexity and pose security risks during storage and transmission (Manyika *et al.,* 2011).

To address these challenges, there is motivation to develop new security models using lightweight techniques to reduce overhead and complexities associated with traditional algorithms. The primary objective of encryption algorithms remains ensuring confusion and diffusion properties, achievable through lightweight techniques like shuffling, scrambling, and transformations. The following sections delve into security models employing scrambling techniques and propose a model integrating secure user authentication with data security (Gantz *et al.,* 2011).

In the ever-evolving landscape of data-driven technologies and pervasive connectivity, safeguarding sensitive information is critical. With a surge in smart devices and an unprecedented reliance on cloud storage, the need for a robust data protection mechanism becomes paramount. The research endeavor titled "The New Data Protection Mechanism to Maximize Data Availability Without Having to Compromise Data Privacy" explores innovative approaches aimed at harmonizing conflicting goals of maximizing data availability and preserving data privacy (Tsai *et al.,* 2015).

This study recognizes challenges posed by limited resources in smart devices, such as memory, computing power, and battery supply, originally designed to interact with cloud storage for seamless data storage. Addressing data security traditionally involves tradeoffs, often entailing high costs for heightened security. Moreover, concerns arise from vulnerabilities inherent in encryption algorithm implementation, emphasizing the critical role of encryption keys in maintaining data security (Mehmood *et al.,* 2016).

In response to these challenges, this research aims to explore and propose a new data protection mechanism that transcends conventional tradeoffs. The objective is to maximize data availability without compromising data privacy, introducing a paradigm shift in how we approach the security of our digital assets. By navigating through issues such as key management, encryption vulnerabilities, and intricacies of data storage and transmission, this study aims to contribute innovative solutions redefining the boundaries of data protection in our interconnected world. Through a careful examination of lightweight techniques and advanced encryption methods, the research aims to pave the way for a future where data availability and privacy coexist harmoniously, ushering in a new era of secure and seamless data management (Jain *et al.,* 2013).

2. DESIGN OF A DYNAMIC SHUFFLING (DS) MODEL

The primary objective of this model is to guarantee data security through the application of lightweight techniques. In the suggested model, security is achieved by dynamically executing a shuffling operation based on a secret key. The model's operations, referred to as Dynamic Shuffling (DS), are elaborated upon in the following discussion (Qin *et al.,* 2016).

2.1. Perfect shuffle Network model

Several methods can be employed for shuffling, and among the various shuffling algorithms, the perfect shuffle technique (Stone 2021) stands out as one of the most popular and widely utilized algorithms. The perfect shuffle technique possesses several interesting properties (John &Hongbing 2022). In recent times, the perfect shuffle has found application in cryptography, as demonstrated by Pandey et al. in 2012. Ernastuti (2014) introduced the Perfect Shuffle Crypto Algorithm (PSCA), utilizing the perfect shuffle algorithm as a security measure for safeguarding messages during transmission (Fong *et al.,* 2016).

Two routing functions shuffle and exchange have served as the foundation for the development of the shuffle exchange network model (Graham & William 2013). Here, the study of the shuffling process is the only goal. An example is provided to clarify how the shuffle exchange network operates. Let us consider an example where N = 8 and we need to shuffle 8 processors or nodes. The perfect shuffling is executed among the 8 processors or nodes, as illustrated in Figure 1(a). The inverse perfect shuffle accomplishes the opposite, restoring the original ordering, as depicted in Figure 1(b) (Middleton *et al.,* 2013).

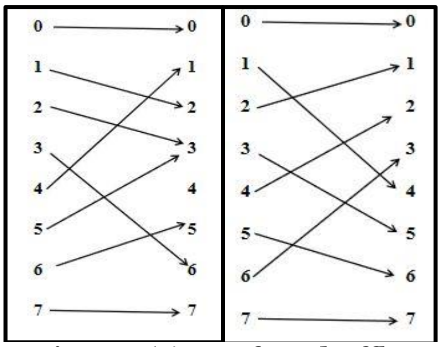

Figure 1(a): Perfect shuffle

2.2. Inverse perfect shuffle

There are two possible implementations for the shuffle and exchanging function: a multistage network and a recirculating network. Figure 2 shows an illustration of a shuffling function that makes use of the recirculating network.

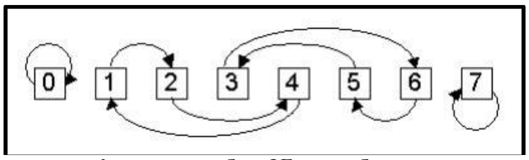

Figure 2: Shuffle exchange rrecirculating network for N=8

The processing is done by the Shuffle Exchange Network in two stages: Shuffle and Exchange. The node in position "i" is shuffled to position "j" by the shuffle component. A fixed permutation process, as shown in Table 1, is used to accomplish this shuffling (Hu *et al.,* 2016).

Table 1 Permutation procedure

p (j) = 2 * i	0<= i<2^n/ 2-1
p (j) = 2*i + 1 - 2^n	Other

Value of 'j' specifies the new position where the node is shifted from its old position 'i'. The computed value of j specifies the position to which the old value is moved.

3. PROPOSED DYNAMIC SHUFFLING MODEL

In the DS model, the entities involved are the CSP and CU. The CU is responsible for ensuring data integrity. The model is comprised of two phases: system initialization and secure data manipulations, which involve file conversion and splitting (Porambage*et al.,* 2016).

3.1. FILE CONVERSION AND SPLITTING

The data must be altered during this phase before it can be sent to cloud storage. This step involves the following processes: conversion, dividing, rearranging, and storage. A series of byte streams are created from the user file. The file is divided into an arbitrary number of 'n' equal-sized fragments during the splitting process; the small subfiles that emerge from the file split are referred to as packets and fragments, respectively. Either the user specifies the number of fragments or a default value is used. The order in which the divided packets are shuffled is exclusive to the CU. To improve the qualities of confusion and diffusion, DS uses the shuffling approach to reorganize user data. The CU uploads the packets to the CSP after rearranging them. Figure 3 shows the DS workflow (Jing et al., 2014).

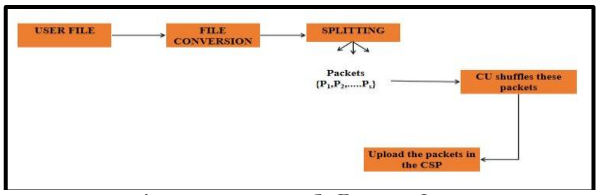
USER FILE
FILE CONVERSION
SPLITTING
Packets {P_1,P_2,.....P_s}
CU shuffles these packets
Upload the packets in the CSP

Figure 3: Workflow of DS

3.2. PROCEDURE FOLLOWED FOR DYNAMIC SHUFFLING

A slight modification has been attempted on the PSCA model to enhance its security further. The model incorporates a dynamic shuffling approach as an addition to the existing static shuffling method. Shuffling is conducted by the user before storing the data and is dynamically performed based on a key. The key serves as a permutation cipher, determining the new positions to which the data fragments are shuffled and placed. This process can be described as a permutation cipher, representing another form of a transposition cipher (Han *et al.,* 2013).

Blocks of letters are used by the permutation cipher instead of the full plaintext or ciphertext. A permutation is a mathematical rule that specifies how to rearrange a set of items. There are two methods for calculating the encryption key, which is used to execute the permutation (Gudipati *et al.,* 2012):

i. The Secure Dynamic Shuffling (SDS) model rearranges the packets using a permutation cipher, and the user provides the encryption key.

ii. The encryption key is divided between the CU and CSP under the Secret Shared Dynamic Shuffling (SSDS) architecture.

Comparing each of the suggested approaches to static shuffling reveals numerous advantages. An example is used to clarify the advantages.

4. SECURE DYNAMIC SHUFFLING (SDS)

The number of fragments determines which permutation cipher the CU enters. This serves as the key, upon which the pieces of data are rearranged. Since there are eight packets in the example, the CU must provide an eight-valued permutation cipher. The permutation cipher key in Figure 4(a) is 8,4,3,7,6,2,5, 1. The inverse operations can be used to calculate inverse shuffling. The suggested Algorithm 1 can be used to calculate the inverse key, or decryption key. Here, the parameters en_key, len, and de_key indicate the encryption key, length, and generated decryption key, respectively. The encryption key is provided as the algorithm's input in order for it to be executed (Xu et al., 2014).

Algorithm 1: DECODE algorithm for performing the inverse shuffle

Input:en_key, len

Output:de_key

```
1. begin
2. for i ← 0 to len-1do
3. de_key [en_key[i] – 1] = i+1
4. end
5. return de_key
6. End
```

A permutation cipher that the CU supplies as input must have a length equal to the number of nodes. In order to use the input sequence to retrieve the decryption key, the CU must keep it. Together with the input permutation cipher, the CU powers the entire system. The following model addresses these issues and shares the secret key for shuffling and unshuffling(Liu *et al.,* 2011).

5. SECRET SHARED DYNAMIC SHUFFLING (SSDS)

In the initial method, the entire security of the key relies on the user, making it a single point of vulnerability. To address this issue, the proposed SSDS method incorporates the concept of a Secret Sharing Scheme (SSS). In this approach, the encryption key is divided between the CU and the CSP using SSS, with both the user and the CSP possessing only partial keys. By combining both keys, permutation ciphers are calculated, determining the new position values. Retrieving the position values involves calculating the decryption keys in the same manner. Consequently, even if the CSP is compromised and leaks its part of the encryption key, the position values remain secure (Sokolova *et al.,* 2015).

The SSDS model incorporates random numbers in addition to the secret sharing method. Encryption keys are used to create the permutation cipher dynamically. Using pseudo random number generators (PRNG), the SSDS model ensures an unpredictable creation of the permutation cipher. However, when utilizing random number generators, certain issues must be addressed. Random numbers are generated based on a seed value provided by the user, creating a dependency on the seed value. It is essential to safeguard the seed value, and there is also a limitation where the same seed value produces the same sequence of random numbers (Cheng *et al.,* 2015).

The CU does not input the actual seed value directly in order to get around these issues and guarantee a unique seed value. Instead, a pseudo random function (PRF) is used to calculate the seed value. To create random sequences, a random number of a fixed length is generated by the PRF and added to the seed value. The PRF function uses the seed value supplied by the CU in conjunction with the UUID value from the CSP. Figure 4(b) provides an illustration of how SSDS is used (Mell *et al.,* 2019).

Example: Let's say there are N=8 nodes that require connecting and shuffles. Figure 4 (a) shows the nodes' interconnection structure using the perfect shuffle. Using the suggested models, shuffles of the same set of nodes would differ according to the encryption key. In Figure 4, this is shown as below.

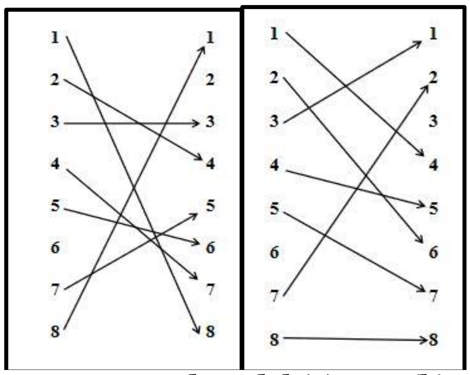

Figure 4: Proposed model (a) SDS (b) SSDS

When the security is concerned, the randomization increases the security of data. The two main benefits are the:
(a) Dynamic permutation of the packets based on the key
(b) Correlation among the neighboring data being very less
When splitting and shuffling user data, it is essential to disperse individual packets as widely as possible. A static shuffle does not contribute to proper randomization and may potentially aid attackers in deducing the packet order, consequently revealing user data. In this model, there may be a computation cost associated with calculating the keys, which can be viewed as a tradeoff when prioritizing the security of the data (Wei *et al.,* 2014).

6. PERFORMANCE ANALYSIS

Three metrics are used to assess the suggested model's performance. Based on the distance measure, the first is completed. After shuffling, it calculates the correlation between the packets. After that, the analysis is completed using the suggested model's security parameter (Xiao *et al.,* 2013).

7. DISTANCE MEASURE

The Euclidean distance metric is applied to the suggested model in order to quantify its randomization property. For both the SDS model and the perfect shuffle, the Euclidean distance is computed. For the SDS model and the perfect shuffle model, it provides the Euclidean distance. There were four distinct encryption keys provided for the SDS model. This information was used to perform encryption and determine the Hamming distance. The perfect shuffle model and SDS model's Euclidean distances are compared on the graph in Figure 5 (Wang *et al.,* 2015).

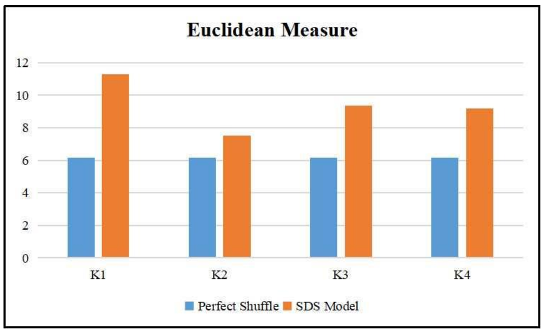

Figure 5: Comparison of the distance measures

8. ANALYSIS OF THE SECURITY FEATURE

Our work's main goal is to provide the CU with a safe storage solution. Attacks against user data may occur when it is being transmitted or when it is stored in the cloud and not in use. The suggested model is made to lessen these two kinds of assaults (Liu *et al.,* 2013).

Secure data transmission

Numerous attacks occur when data is in transit over the internet. Many CSPs, especially personal clouds, utilize REST API for obtaining customer data. While this functionality offers convenience, it also presents an attractive target for attackers to exploit storage providers. The typical procedure for uploading data to the CSP involves using a web browser, which itself has various vulnerabilities. Several efforts have been made to safeguard data transmitted through web browsers, yet there have been instances of attacks on encrypted

data sent via browsers. Browsers require frequent updates to address security issues and respond to emerging threats (Liu *et al.,* 2014).
For instance, the Google team identified a bug in the SSL 3.0 protocol that could be exploited to intercept supposedly encrypted critical data between the client and server. A notable example is the POODLE (Padding Oracle On Downgraded Legacy Encryption) attack, where a man-in-the-middle attacker decrypted secure HTTP cookies (Jensen *et al.,* 2019).
The suggested model stays away from doing upload and download operations on raw data in order to get around these restrictions. Rather, the information is subjected to a sequence of processes that prevent hackers from reconstructing the initial data. One significant advantage of the splitting operation, which converts data into byte streams and then splits it into packets, is discussed below (Xu *et al.,* 2015):
Let m – be the number of packets to be split
P1, P2 … Pm – be the different individual packets after splitting
For an attacker to reassemble the data, they must have every packet. It becomes hard to obtain all of the packets and reassemble them if m is made really large. The O(m) computational overhead of this method is quite modest, and it becomes insignificant when considering the security parameter (Zhang *et al.,* 2016).

Secure Data Storage

Ensuring the protection of data at its storage location and from internal components is imperative. In the proposed model, two entities handle the data. The CSP may be susceptible to attacks, or it may maliciously target user data, leading to the potential compromise of packets. Despite the attacker gaining access to the data, constructing it in a meaningful form is unfeasible. Reconstruction is thwarted because the packets are shuffled in an order unknown to the CSP but known only to the CU. To prevent the CU from disclosing the permutation cipher, the SSDS model employs a secret shared approach (Li *et al.,* 2017).
In the SSDS model, randomization is introduced through permutation cipher keys, enhancing the confusion and diffusion of the data and reducing correlation among the data. This complexity makes it challenging for attackers to deduce accurate information. In contrast, the fixed shuffle technique lacks randomization, as the shuffling occurs in a predictable manner, susceptible to prediction by attackers. Additionally, the correlation among neighboring data remains constant, creating a vulnerability in determining the correct data order. Conversely, the dynamic shuffling method in SSDS incorporates randomization, thus enhancing security (Machanavajjhala*et al.,* 2016).
The correlation among neighboring data is minimal and varies over time based on the encryption key, as measured by the Euclidean distance. A higher Euclidean distance value signifies low correlation among the data, indicating improved data security.Unlike the perfect shuffle, the dynamic shuffle utilizes an encryption key for the shuffling process. The encryption key is either directly obtained from the user or split between the CU and CSP using the SSS. The computational cost in the dynamic shuffling approach is limited to the computation of encryption and decryption keys. The length of the encryption key is contingent on the number of data fragments requiring shuffling. If the user provides the encryption key, dynamic shuffling is executed based on it. Alternatively, in cases where it is shared, the encryption key is calculated, with the calculation dependent on the number of data fragments. The decryption key is subsequently computed based on the encryption key (Ton *et al.,* 2015).

9. DESIGN OF A SECURE ENCRYPTION MODEL (SEM)

To allow the user to store the data, the cloud storage has its own authentication procedure. Traditionally, the cloud storage stores the login credentials of the user in its server. This might lead to data leakage or attacks over these credentials at the storage side. The various attacks that can happen over the user credentials are listed below (Samarati*et al.,* 2021):
(1) Password guessing attack: attacks that are done for obtaining the user password.
(2) Replay attack: tracks the authentication packet that is transmitted and reproduces the information to unauthorized users.
(3) Man – in – the – middle attack: attacker poses to be a user and tries to acquire the password from the server.
(4) Masquerade attack: attacker pretends to be a verifier whereby retrieving the authentication keys or parameters from the user.
(5) Phishing attack: attacker fakes the emails or the cloud storage websites that demand the user to reveal his password or authentication keys.
(6) Shoulder Surfing attack: attacker secretly observes the password when the user enters it.
To counteract attacks encountered during the user authentication phase, the SEM model is introduced to provide secure user authentication while ensuring data security. In contrast to relying on traditional encryption algorithms, the SEM model utilizes a Fourier transformation technique known as "Hadamard transforms." Hadamard transforms, also known as Walsh Hadamard transforms or Hadamard–Rademacher–Walsh transforms, belong to the generalized class of Fourier transforms (Horadam 2015). Jacques Hadamard invented the Hadamard transforms (Hadamard 2013), and they are commonly employed in signal processing. The technique has applications in various domains, including signal processing

(Gumas& Charles Constantine 2018), data compression (Ouyang & Cham 2010), randomness measures (Kak 2021), and even image encryption (Yan & Pan 2017). Both Hadamard and Fourier transforms are unitary transformations primarily used in image transformations, where the objective is to identify an alternative domain for easier data processing (Samarati*et al.,* 2018).
One advantage of Hadamard transforms over Fourier transforms is that Hadamard transforms involve a series expansion of basis functions over the values of +1 or -1, while Fourier transforms are based on trigonometric terms. As a result, Hadamard functions can be implemented more efficiently in a digital environment compared to the exponential basis functions of Fourier transforms. The proposed model aims to leverage Hadamard transforms to establish a robust security framework (Sweeney *et al.,* 2012).
The security in SEM model is brought into by many levels as listed below
- Secure user authentication
- Rearrangement of the user data
- Encryption being done in a number of rounds with each round having a unique key

The process of introducing a secure user authentication as the first layer of security is demonstrated in the next section. The SEM's intricate design comes next.

Hadamard Matrix

The basic matrix for the transformations is a Hadamard matrix, also known as the generating matrix. To carry out the Hadamard transforms, the generator matrix is the Hadamard matrix. For the binary values, refer to the matrix in (2). For values that are not binary, the same Hadamard matrix can be used. In order to achieve this, the key value is used to build the Hadamard matrix, and each negative number that is encountered is substituted with the corresponding modulo number less the corresponding negative value. For example, the number 6 (7 – 1 = 6) is used in place of the -1 after the modulo 7 operation is completed. The temporal complexity is reduced by using modulo operations (Reddy *et al.,* 2010).
The SEM approach uses different keys to accomplish encryption in rounds. The user provides the key values needed to encrypt the data. The number 'n' in the key can have the property that every x in 'n' is prime, and 2x -1 is also prime. The amount of keys that the user defines determines how many rounds there will be. Figure 6 illustrates the encryption procedure that takes place in a single round (Meyerson *et al.,* 2014).

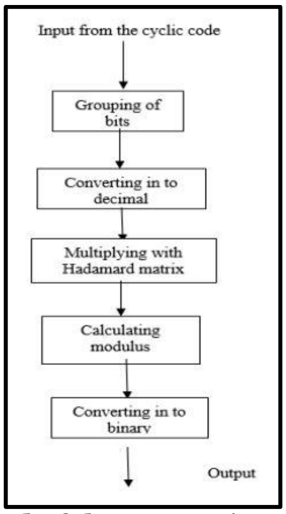

Figure 6: Single round of the encryption process used in SEM

After being sorted into groups according to the key value, the binary bits are translated to the equivalent decimal value. The matching Hadamard matrix is then multiplied by this decimal sequence. Since the Hadamard matrix is a square matrix, the number of decimal sequences should be equal to the size of the matrix. If the size of the decimal sequence is less than the size of the matrix, zeros must be appended at the end of the decimal sequence. The resultant value undergoes a modulus operation. Subsequently, the processes of encryption and decryption will be explained in detail, accompanied by an example (Brederecke*t al.,* 2014).

Round 1

The bits are sorted into three groups for the input sequence (3) at round 1, after which they are transformed to the decimal values 6,2,3,5,7,6,0,3. The matching modulo 7 Hadamard matrix is multiplied by this. When modulo operation is applied to the resultant value, an encrypted sequence with the sequence number 100000011011000100100010 is produced. This serves as the second round's input.

Round 2

The encrypted binary sequence for round 2 is 001101010001111010001110101100001110000O, using key 5 and the Hadamard matrix corresponding to modulo 31.

Round 3

When 7 is used as the key, this input sequence is encrypted to produce 01001100111111010111110011001100010011101110010l, with the Hadamard matrix equal to modulo 127. (6)
A crucial attribute of Hadamard transforms is the duration of both the input and output sequences. The randomness measure will increase if both of these sequences have the same length or a different one. In this case, the encrypted sequence (6) is 56 bits long, while the input sequence (3) was 24 bits long.

DECRYPTION

The key sequence is {7, 5, 3}, and the sequence that needs to be decoded is (6). The modulo multiplicative inverse is computed for decryption. The Hadamard matrix's modulo multiplicative inverse is 16, for modulo 127; it is 4, for modulo 31, and it is 1 for modulo 7(Ko *et al.,* 2011).

Round 1

Groups of every seven bits from the binary sequence (6) are produced in round 1. The decimal values for each group are then obtained; the resulting numbers are 38, 63, 43, 115, 24, 49, 29, and 101. The modulo 127 Hadamard matrix is multiplied by these decimal numbers. The multiplicative inverse of 16 is applied to the resulting numbers, and then a modulo 127 operation is performed. After that, the values are transformed into their binary counterparts. The following round's input is 01101010001111010001110101100001110000O, which is the output of this round.

Round 2

Key 5 and the multiplicative inverse 4 are taken, together with the result from round 1. By following the same procedures, 100000011011000100100010 is the binary sequence that results.

Round 3

The binary sequence that was received in round 2 can be decrypted using key 3 and multiplicative inverse 1, which returns the original binary sequence, 110010011101111110000011.

10. ANALYSIS OF SEM MODEL

SEM model provides security at multiple levels and exhibits better performance. This section presents the benefits of the proposed model in terms of security and performance (Lu *et al.,* 2014).

Security analysis

The model ensures security at various levels. The encryption process does not take the initial input as-is but rearranges or scrambles it, making it challenging for cryptanalysis. Subsequently, encryption is performed over multiple rounds using distinct keys for each level (Paillier*et al.,* 2019).

1. Rearranging of input data

The original input, which is plaintext, is converted to its binary counterpart. The cyclic code technique rearranges the binary sequence in order to counter known-plaintext assaults that target predictable regions of the data. Analyzing both plaintext and ciphertext in order to deduce statistical relationships or patterns is known as a cryptoanalytic attack. In order to prevent these kinds of attacks, the plaintext is first reorganized and then encrypted. As a result, any cryptanalysis that aims to discern patterns between plaintext and ciphertext will find it difficult (Sedayao*et al.,* 2014).

2. Multiple encryption

The number of rounds determines an algorithm's cryptographic strength. An analyst's ability to extract every key is hindered when the number of rounds and keys is increased, reducing the possibility of a brute-force key search assault. When double encryption is used, two keys are used, which are indicated by
$C = E(k_2, E(k_1, P))$
$P = D(k_1, D(k_2, C))$
Where C is the ciphertext, P the plain text and k_1, k_2 are the keys. In the Hadamard transform technique, the key values are prime numbers, which can be of any length. For instance, if the key length were 8 bits, then the key spacewould have a possibility of 2^8 or 256 keys. In addition, for the next level ofencryption, another key is used, which can be of same or different length, soagain256keysarepossible.Togetholdofbothkeystheattackerneedstocrackall the 2^{16} or 65536 possible keys. When the encryption is done in a number ofroundsandwithanumberofdistinctkeys,the securityofthesystem isfurtherimproved(Yong *et al.,* 2016).

Performance Analysis

In addition to ensuring data security, other critical parameters to consider are the storage space required for storing encrypted data and the execution time for the encryption process. Many existing models rely on traditional cryptographic techniques, employing encryption and decryption algorithms to render user data unreadable for protection. However, these encryption algorithms often result in increased processing time, and the encrypted data occupies more storage space. Users also face the overhead of managing encryption and decryption keys. To address these challenges, the primary focus of the thesis has been on leveraging lightweight techniques. These techniques employ mechanisms such as scrambling and shuffling to enhance randomization and, consequently, data protection. In contrast to traditional cryptographic mechanisms, these mechanisms can be termed as lightweight techniques (Oracle, 2012).
To demonstrate the efficiency of lightweight techniques, an analysis was conducted. A set of files was encrypted using traditional mechanisms, and their execution time and storage space requirements were recorded. The same set of files was then encrypted using the proposed model, based on a lightweight approach, and their processing time and storage space utilization were documented. A comparison was made between these two sets of data (Hadoop, 2012).
Furthermore, the proposed model exhibits lower time complexity and satisfies the avalanche effect. The subsequent section provides a detailed discussion of these advantages.

1. Execution time

In the SEM model, text is transformed into binary data and encrypted using Hadamard transforms, a process that requires less time compared to using the ECC encryption algorithm. The algorithms were run on an Intel Core i7 processor with a speed of 3.60 GHz. Files of varying sizes were selected and encrypted using ECC. The identical set of files was also encrypted using SEM. It provides information on file sizes and their respective execution times in seconds (Jung *et al.,* 2014).

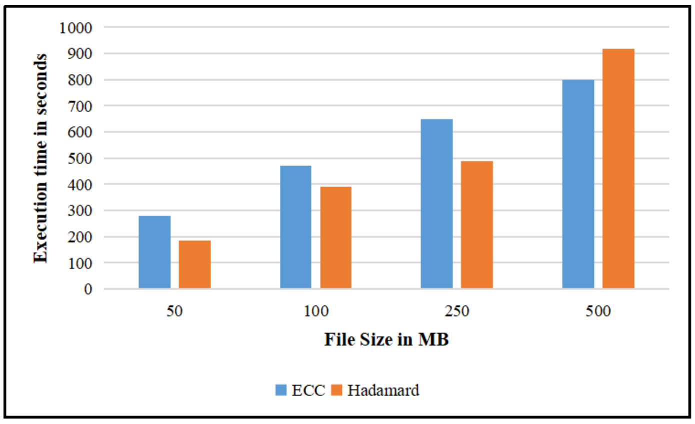

Figure 7: Comparison of execution times needed for the two models

Figure 4.9 displays a comparison of the execution times that were required. The graph shows that for both ways, the execution time increases as the file size increases. Hadamard transformations, on the other hand, have a shorter execution time and do not grow as much as the ECC approach. The average difference in execution times between the two methods when applied to the same collection of files is approximately 25% (Atenieseet *al.,* 2017).

2. Storage Space

The required storage space for encrypted data is a significant factor, especially considering the pricing policies of cloud storage providers. Users typically pay for the storage they utilize, making storage space a crucial cost-cutting consideration for users. Additionally, cloud storage providers may have memory constraints, prompting them to minimize the allocated memory space for each user. To assess the storage space requirements, a comparison is made between the storage space needed when using ECC to encrypt the data and when using the SEM model. It presents the original file size alongside their sizes in the encrypted form. Figure 8 illustrates a comparison of the encrypted file sizes for both models (Verma *et al.,* 2011).

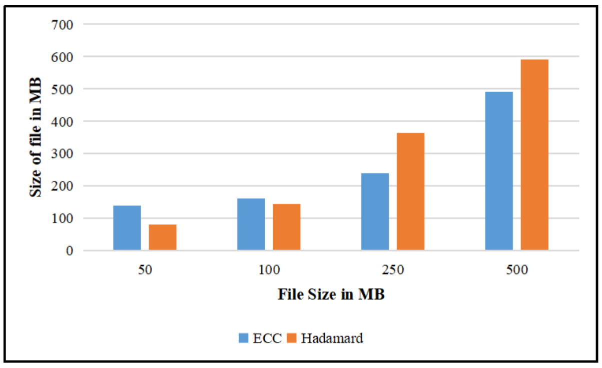

Figure 8: Comparison of encrypted file size for the two models

Compared to encryption using ECC, the SEM model requires around 10% less storage space when the same set of files are performed using the two techniques. This decline would assist in lowering the user's expenses (Sathe, 2019)

3. Time complexity

The quantity and kind of calculations an algorithm completes define its temporal complexity. A point on the elliptic curve is chosen, scalar multiplication is performed, and other computations are part of the ECC method. The selection of a point on the elliptic curve takes the longest of these operations, adding to the overall time complexity. O(nk+1), where n is the bit size and k is dependent on the multiplication algorithm used, is the expression for the worst-case time complexity (Washington 2018). When using ECC encryption, the key is typically longer than when using the SEM model. According to Koblitz (2017), ECC is a public key cryptosystem that uses two keys: a public key and a private key. The method of determining the private key adds difficulty on top of the intricacies inherent in the encryption procedure.The SEM model, in contrast, lowers the overhead and complexity involved in key production and maintenance by using the same key for both encryption and decryption. The SEM model's temporal complexity is O(kn), where k is the number of decimal groups at each level and n is the length of the input sequence. The complexity of one encryption round is O(kn), and the time complexity grows linearly with the number of rounds. Preprocessing stages include creating Hadamard matrices based on the key and figuring out their multiplicative inverses; the complexity of these operations is not taken into account during the main computation. By restricting the output range, the combination of modulo operations and Hadamard matrices significantly minimizes processing complexity (Madrigal, 2012).

4. Avalanche effect

As mentioned in chapter 3, the avalanche effect can be computed by formula (3.5).An example of a plaintext bit change and its impact on the ciphertext can be found in the following example. The input sequence (4.3) remains unchanged, and altering the 9th bit value from 1 to 0 also modifies the decimal equivalent:

Binary	110	010	010	101	111	110	000	011
Decimalequivalent:6		2	2	5	7	6	0	3

Afterperformingtheencryptionsteps,ityieldsthefollowingencryptedsequence:
0010010100110100010110110001110110110111
Thissequenceisdifferentfromtheencryptedsequencethatwasobtainedearlier.
Original encrypted text:01001100111111010101111100110011000011000100111011100101
Changedencryptedtext:11001100101111110111101110100010001101101100010110 01

Theavalancheeffectforthisexamplecalculatedaspertheformula(3.5)wouldbe
Avalancheeffect=(20 /56)*100=38%

The ciphertext is affected by the avalanche effect as well. When the same sequence is decrypted, a single bit change in the ciphertext sequence also has an impact.Despite the ciphertext merely changing by one bit, the decrypted sequence has undergone a significant alteration. The decrypted sequence's length fluctuates in addition to the bits being altered, which increases the randomness.the suggested SEM maintains lower execution times and storage requirements while offering several layers of security(Matthan *et al.,* 2017).

11. CONCLUSION

In conclusion, the quest to enhance data availability while safeguarding privacy is pivotal in our data-driven era. The emergence of novel data protection mechanisms marks a significant stride in this direction. By leveraging cutting-edge encryption techniques, decentralized storage solutions, and advanced access control protocols, organizations can optimize data availability without sacrificing privacy. Moreover, embracing a proactive approach towards compliance with stringent data protection regulations ensures a robust framework for safeguarding sensitive information. However, it's imperative to acknowledge that achieving the delicate balance between data availability and privacy necessitates ongoing evaluation and adaptation.

Continuous refinement of strategies, coupled with heightened awareness of emerging threats, is essential to stay ahead in this evolving landscape. Ultimately, by embracing innovative technologies and adopting a privacy-by-design mindset, businesses can foster trust among stakeholders while maximizing the utility of data assets in a responsible and ethical manner.

Conflict of Interest:

The authors declare no conflict of interest.

Acknowledgment:
I am extremely grateful to my supervisors, Dr. V. Vijayalakshmi, for her invaluable advice, continuous support, and patience during my study. Her immense knowledge and plentiful experience have encouraged me in all the time of my academic research.

REFERENCES

1. Abadi DJ, Carney D, Cetintemel U, Cherniack M, Convey C, Lee S, Stone-braker M, Tatbul N, Zdonik SB. Aurora: a new model and architecture for data stream management. VLDB J. 2013;12(2):120–39.
2. Kolomvatsos K, Anagnostopoulos C, Hadjiefthymiades S. An efficient time optimized scheme for progressive analytics in big data. Big Data Res. 2015;2(4):155–65.
3. Big data at the speed of business, [online]. http://www-01.ibm.com/soft-ware/data/bigdata/2012.
4. Manyika J, Chui M, Brown B, Bughin J, Dobbs R, Roxburgh C, Byers A. Big data: the next frontier for innovation, competition, and productivity. New York: Mickensy Global Institute; 2011. p. 1–137.
5. Gantz J, Reinsel D. Extracting value from chaos. In: Proc on IDC IView. 2011. p. 1–12.
6. Tsai C-W, Lai C-F, Chao H-C, Vasilakos AV. Big data analytics: a survey. J Big Data Springer Open J. 2015.
7. Mehmood A, Natgunanathan I, Xiang Y, Hua G, Guo S. Protection of big data privacy. In: IEEE translations and content mining are permitted for academic research. 2016.
8. Jain P, Pathak N, Tapashetti P, Umesh AS. Privacy preserving processing of data decision tree based on sample selection and singular value decomposition. In: 39th international conference on information assurance and security (lAS). 2013.
9. Qin Y, et al. When things matter: a survey on data-centric internet of things. J Netw Comp Appl. 2016;64:137–53.
10. Fong S, Wong R, Vasilakos AV. Accelerated PSO swarm search feature selection for data stream mining big data. In: IEEE transactions on services computing, vol. 9, no. 1. 2016.
11. Middleton P, Kjeldsen P, Tully J. Forecast: the internet of things, worldwide. Stamford: Gartner; 2013.
12. Hu J, Vasilakos AV. Energy Big data analytics and security: challenges and opportunities. IEEE Trans Smart Grid. 2016;7(5):2423–36.
13. Porambage P, et al. The quest for privacy in the internet of things. IEEE Cloud Comp. 2016;3(2):36–45.
14. Jing Q, et al. Security of the internet of things: perspectives and challenges. WirelNetw. 2014;20(8):2481–501.
15. Han J, Ishii M, Makino H. A hadoop performance model for multi-rack clusters. In: IEEE 5th international conference on computer science and information technology (CSIT). 2013. p. 265–74.
16. Gudipati M, Rao S, Mohan ND, Gajja NK. Big data: testing approach to overcome quality challenges. Data Eng. 2012:23–31.
17. Xu L, Jiang C, Wang J, Yuan J, Ren Y. Information security in big data: privacy and data mining. IEEE Access. 2014;2:1149–76.
18. Liu S. Exploring the future of computing. IT Prof. 2011;15(1):2–3.
19. Sokolova M, Matwin S. Personal privacy protection in time of big data. Berlin: Springer; 2015.
20. Cheng H, Rong C, Hwang K, Wang W, Li Y. Secure big data storage and sharing scheme for cloud tenants. China Commun. 2015;12(6):106–15.
21. Mell P, Grance T. The NIST definition of cloud computing. Natl Inst Stand Technol. 2019;53(6):50.
22. Wei L, Zhu H, Cao Z, Dong X, Jia W, Chen Y, Vasilakos AV. Security and privacy for storage and computation in cloud computing. Inf Sci. 2014;258:371–86.
23. Xiao Z, Xiao Y. Security and privacy in cloud computing. In: IEEE Trans on communications surveys and tutorials, vol 15, no. 2, 2013. p. 843–59.
24. Wang C, Wang Q, Ren K, Lou W. Privacy-preserving public auditing for data storage security in cloud computing. In: Proc. of IEEE Int. Conf. on INFOCOM. 2015. p. 1–9.
25. Liu C, Ranjan R, Zhang X, Yang C, Georgakopoulos D, Chen J. Public auditing for big data storage in cloud computing a survey. In: Proc. of IEEE Int. Conf. on computational science and engineering. 2013. p. 1128–35.
26. Liu C, Chen J, Yang LT, Zhang X, Yang C, Ranjan R, Rao K. Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates. In: IEEE trans. on parallel and distributed systems, vol 25, no. 9. 2014. p. 2234–44

27. Xu K, et al. Privacy-preserving machine learning algorithms for big data systems. In: Distributed computing systems (ICDCS) IEEE 35th international conference; 2015.
28. Zhang Y, Cao T, Li S, Tian X, Yuan L, Jia H, Vasilakos AV. Parallel processing systems for big data: a survey. In: Proceedings of the IEEE. 2016.
29. Li N, et al. t-Closeness: privacy beyond k-anonymity and L-diversity. In: Data engineering (ICDE) IEEE 23rd international conference; 2017.
30. Machanavajjhala A, Gehrke J, Kifer D, Venkitasubramaniam M. L-diversity: privacy beyond k-anonymity. In: Proc. 22nd international conference data engineering (ICDE); 2016. p. 24.
31. Ton A, Saravanan M. Ericsson research. [Online]. http://www.ericsson.com/research-blog/data-knowledge/big-data-privacy-reservation/2015.
32. Samarati P. Protecting respondent's privacy in microdata release. IEEE Trans Knowl Data Eng. 2021;13(6):1010–27.
33. Samarati P, Sweeney L. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical Report SRI-CSL-98-04, SRI Computer Science Laboratory; 2018.
34. Sweeney L. K-anonymity: a model for protecting privacy. Int J Uncertain Fuzz. 2012;10(5):557–70.
35. Meyerson A, Williams R. On the complexity of optimal k-anonymity. In: Proc. of the ACM Symp. on principles of database systems. 2014.
36. Bredereck R, Nichterlein A, Niedermeier R, Philip G. The effect of homogeneity on the complexity of k-anonymity. In: FCT; 2011. p. 53–64.
37. Ko SY, Jeon K, Morales R. The HybrEx model for confidentiality and privacy in cloud computing. In: 3rd USENIX workshop on hot topics in cloud computing, HotCloud'11, Portland; 2011.
38. Lu R, Zhu H, Liu X, Liu JK, Shao J. Toward efficient and privacy-preserving computing in big data era. IEEE Netw. 2014;28:46–50.
39. Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In: EUROCRYPT. 2019. p. 223–38.
40. Microsoft differential privacy for everyone, [online]. 2015. http://download.microsoft.com/.../Differential_Privacy_ for_Everyone.pdf.
41. Sedayao J, Bhardwaj R. Making big data, privacy, and anonymization work together in the enterprise: experiences and issues. Big Data Congress; 2014.
42. Yong Yu, et al. Cloud data integrity checking with an identity-based auditing mechanism from RSA. Future Gener Comp Syst. 2016;62:85–91.
43. Oracle Big Data for the Enterprise, 2012. [online]. http://www.oracle.com/ca-en/technoloqies/biq-doto.
44. Hadoop Tutorials. 2012. https://developer.yahoo.com/hadoop/tutorial.
45. Jung K, Park S, Park S. Hiding a needle in a haystack: privacy preserving Apriori algorithm in MapReduce framework PSBD'14, Shanghai; 2014. p. 11–17.
46. Ateniese G, Johns RB, Curtmola R, Herring J, Kissner L, Peterson Z, Song D. Provable data possession at untrusted stores. In: Proc. of int. conf. of ACM on computer and communications security. 2017. p. 598–609.
47. Verma A, Cherkasova L, Campbell RH. Play it again, SimMR!. In: Proc. IEEE Int'l conf. cluster computing (Cluster'11); 2011.
48. Sathe, Gopal. 2019. "How Sai Baba Was Made to Spy on Your Phone for Credit Ratings." HuffPost blog post, 7 August.
49. Madrigal, Alexis C. 2012. "Reading the Privacy Policies You Encounter in a Year Would Take 76 Work Days."
50. Matthan, Rahul. 2017. "Beyond Consent: A New Paradigm for Data Protection." Discussion Document 2017-03. Bangalore: Takshashila Institution.