Int. j. adv. multidisc. res. stud. 2026; 6(4):455-460

Received: 21-05-2026
Accepted: 02-07-2026

International Journal of Advanced Multidisciplinary Research and Studies

ISSN: 2583-049X

Performance Analysis of Secured Efficient Distributed Storage in Cloud (SecDcloud) Encryption Algorithm Based on Avalanche Effect

[1] Anah Hassan Bijik, [2] Barka Fori Tattama, [3] Ibrahim Aliyu Yakubu, [4] Joseph Oladele Aremu, [5] Adelaiye Oluwasegun Ishaya

[1, 2, 3, 4, 5] Department of Computer Science, Bingham University, Karu, Nasarawa State, Nigeria

Corresponding Author: **Anah Hassan Bijik**

Abstract

The avalanche effect is a fundamental property used to measure the cryptographic strength of an encryption algorithm: any single-bit change in the plaintext or encryption key should cause approximately half of the bits in the resulting cipher text to flip unpredictably. An algorithm is considered cryptographically strong only when its avalanche effect exceeds 50%; values below this threshold signal weak diffusion and leave the cipher vulnerable to brute-force, statistical, and other cryptanalytic attacks. Evaluating the avalanche effect of a cloud encryption algorithm is therefore essential, since cloud data is typically stored on servers controlled by third-party providers, making the diffusion strength of the encryption algorithm the primary safeguard for data confidentiality and integrity. This paper determines the avalanche effect of the Secured Efficient Distributed Storage in Cloud (SecDcloud) encryption algorithm. SecDcloud is a formal-methods-based improvement on the Security Aware Efficient Distributed Storage (SA-EDS) algorithm; it partitions user data into four units using sequence head-and-last operations, pairs and encrypts them with a hash-and-salt-generated key via XOR, and distributes the cipher text across separate cloud servers. The results show avalanche effect values consistently above 50% across all ten trials, with an overall average of 54%, confirming that SecDcloud satisfies the avalanche criterion and provides the diffusion strength required of a secure cloud encryption scheme.

Keywords: Avalanche Effect, SecDcloud, Cryptographic Strength, SA-EDS

1. Introduction

The avalanche effect is a foundational concept in cryptography: it requires that a single-bit change in the plaintext or the encryption key should cause approximately 50% of the bits in the resulting cipher text to change unpredictably Dewangan *et al*. (1987) [16] and Bore and Hemanth (2026) [17]. An encryption algorithm that fails to satisfy this property has weak diffusion, meaning an attacker can predict or approximate the cipher text output from a partial knowledge of the input, making the algorithm vulnerable to brute-force, statistical, and differential cryptanalytic attacks. Measuring the avalanche effect is therefore the primary empirical test of how strong any encryption algorithm truly is Kumar and Tiwari (2012) [19] and Bondar *et al*. (2025) [20].

This requirement is important in cloud computing, where organizations store and retrieve sensitive data on shared servers they do not control, relying entirely on encryption to maintain confidentiality and integrity Peter and Grance (2011) [1] and Devi (2024) [2]. Cloud computing delivers on demand, pay-per-use access to a shared pool of computing resources servers, storage, networks, applications, and services with minimal management effort Pansotra and Singh (2015) [3] and Jewargi (2023) [4]; but as enterprises transfer growing volumes of sensitive data to these environments, the access that cloud service providers retain to that stored data becomes a major security concern Potey *et al*. (2016) [5], Ginanjar *et al*. (2024) [6], and Li *et al*. (2017) [7]. In this context, the strength of the encryption algorithm in use as measured by its avalanche effect is the principal line of defence.

Most cloud service providers store data in plaintext format and leave encryption to the user Nadeem *et al*. (2022) [8] and Choubey *et al*. (2022) [9], which means that the quality of the encryption algorithm chosen directly determines how secure cloud data is. Major cloud security challenges unauthorized access, data breaches, privacy violations, loss of confidentiality and integrity are all addressed at the encryption layer. Cryptography, the discipline of transforming plaintext into unreadable cipher text and back, is the core of that layer Maqsood *et al*. (2017) [10] and Jøsang (2024) [11]; and among all the properties a

cryptographic algorithm can possess, the avalanche effect is the one most directly tied to how unpredictable and attack-resistant the cipher text output is.

The avalanche effect quantifies how a small change in the input of a cryptographic function propagates to its output Kumar and Tiwari (2012) [19] and Bondar *et al*. (2025) [20]. In practical terms, altering a single bit in the plaintext or encryption key of a secure algorithm should change approximately half of the bits in the resulting cipher text, making the output appear statistically random to anyone who does not hold the key. A cryptographic algorithm is classified as meeting the avalanche criterion when this ratio exceeds 50%; below this threshold, the output is insufficiently unpredictable and the algorithm is considered cryptographically weak Dewangan *et al*. (1987) [16] and Bore and Hemanth (2026) [17]. The study of avalanche effect across block ciphers, S-box constructions, and hash functions consistently uses this 50% benchmark as the pass-or-fail standard Maram and Gnanasekar (2018) [21], Bondar *et al*. (2025) [22], Mohamed *et al*. (2021) [23], and Darshana *et al*. (2022) [24].

Despite the principled design of SecDcloud MSED2 and its adoption of hash-and-salt key generation and data splitting across servers, no study has yet empirically determined its avalanche effect to confirm whether it meets the 50% cryptographic standard. Without this measurement, the algorithm's resistance to brute-force and statistical attacks cannot be formally established, and its suitability as a cloud encryption scheme remains unverified. This gap is the motivation for the present study.

This paper determines the avalanche effect of the SecDcloud MSED2 encryption algorithm. The result is compared against the accepted 50% threshold to establish whether SecDcloud MSED2 provides the diffusion strength required of a secure cloud encryption algorithm. The remainder of this paper is organized as follows: Section 2 reviews related literature on the avalanche effect and cloud encryption algorithms; Section 3 describes the SecDcloud MSED2 methodology; Section 4 presents and discusses the results; and Section 5 draws conclusions.

2. Aim and Objectives

The aim of this study is to determine the avalanche effect of the Secured Efficient Distributed Storage in Cloud (SecDcloud) MSED2 encryption algorithm, in order to establish whether it meets the accepted 50% cryptographic diffusion standard required of a secure cloud encryption scheme.

The specific objectives of this study are to: (i) implement the SecDcloud MSED2 encryption algorithm, including its hash-and-salt key generation, data partitioning, and XOR-based encryption processes; (ii) generate plaintext and single-bit-modified plaintext/key pairs and encrypt them using the SecDcloud MSED2 algorithm across multiple trials; (iii) compute the avalanche effect for each trial by measuring the percentage of cipher text bits that change as a result of the single-bit modification; (iv) compare the computed avalanche effect values against the accepted 50% cryptographic threshold to evaluate the diffusion strength of SecDcloud MSED2; and (v) draw conclusions on the suitability of SecDcloud MSED2 as a secure cloud encryption scheme based on the avalanche effect results obtained.

3. Review of Related Works

In today's internet era, with online transactions occurring almost every second and terabytes of data generated daily, securing information is a constant challenge, and cryptography remains an integral part of modern information security. Several cryptographic algorithms exist, each with its own cost-performance trade-off, and no single algorithm offers low cost and high performance as a one-stop solution; comparing the strengths and weaknesses of candidate algorithms therefore requires a consistent, measurable criterion. The avalanche effect provides exactly that criterion, and the studies reviewed below illustrate how it has been used to evaluate block ciphers, hash functions, and distributed-storage encryption schemes alike.

The avalanche effect has been studied extensively across cryptographic algorithm types block ciphers, S-box constructions, and hash functions as the primary empirical measure of diffusion strength and resistance to cryptanalytic attack. The works reviewed in this section are organized around four themes directly relevant to the current study: (i) the definition and measurement standard for the avalanche effect; (ii) its evaluation in AES and block cipher implementations; (iii) its role in dynamic S-box design and permutation-based algorithms; and (iv) its application to hash functions. Together, these works establish the theoretical basis and the empirical precedent against which the avalanche effect of the SecDcloud MSED2 algorithm is determined.

3.1 Avalanche Effect: Definition and Standard

Kumar and Tiwari (2012) [19] define the avalanche effect as the property by which a slight change in either the plaintext or the encryption key results in a significant and unpredictable change in the cipher text. This property ensures that an attacker cannot predict the cipher text from partial knowledge of the plaintext, and vice versa; an algorithm that does not satisfy it can be breached by a cryptanalyst. The avalanche effect is formally quantified as the ratio of the number of bits changed in the cipher text to the total number of bits in the cipher text, and a cryptographic algorithm is classified as meeting the standard when this value exceeds 50% Kumar and Tiwari (2012) [19] and Bondar *et al*. (2025) [20].

The 50% output-bit-change criterion established in Kumar and Tiwari (2012) [19] and confirmed in Bondar *et al*. (2025) [20] is the benchmark adopted in the present study to evaluate the SecDcloud MSED2 algorithm.

3.2 Avalanche Effect in AES and Block Cipher Implementations

Dewangan *et al*. (1987) [16] studied the avalanche effect in AES using binary codes, demonstrating that single-bit changes in the key or plaintext produce substantial changes in the cipher text, which directly reflects the diffusion strength that AES derives from its substitution-permutation network structure. Maram and Gnanasekar (2018) [21] extended this investigation by implementing AES in MATLAB and using the platform's simulation and visualization capabilities to measure and display the avalanche effect step by step. Their work showed that AES is computationally impossible to break without knowledge of the key, and that its avalanche effect can be reproduced and verified systematically using software simulation.

Bondar *et al*. (2025) [22] further analyzed the avalanche effect of a cryptographic algorithm based on information-controlled permutation operations, providing additional evidence for the use of permutation-based transformations to achieve strong diffusion Maram and Gnanasekar (2018) [21] and Bondar *et al*. (2025) [22].

AES consistently demonstrated an avalanche effect above 50% in both binary-code-based and MATLAB-based evaluations, confirming its suitability as a reference standard for block cipher diffusion and providing a methodological precedent for evaluating the avalanche effect of other algorithms including SecDcloud MSED2.

3.3 Avalanche Effect in Dynamic S-Box and Block Cipher Design

Bore and Hemanth (2026) [17] proposed dynamic S-boxes designed to achieve improved Strict Avalanche Criterion (SAC) values, arguing that the S-box is the primary source of nonlinearity and diffusion in block ciphers and that dynamically generated S-boxes produce stronger avalanche behavior than fixed ones. Mohamed *et al*. (2021) [23] evaluated the key avalanche effect of a proposed block cipher by subjecting 32-bit plaintext to 32 experimental trials, each time changing a single input bit and measuring how many cipher text bits changed as a result. The experiment confirmed that if a block cipher does not exhibit a sufficient avalanche effect, its output is predictable and the algorithm is vulnerable to cryptanalysis.

The findings revealed that the proposed block cipher algorithm met the key avalanche criterion with approximately 50% of the output bits changed in the cipher text across all 32 trials, confirming that the algorithm satisfies the avalanche effect property and achieves improved diffusion. The output was effectively random irrespective of the input, hiding all useful information about the original data and raising the algorithm's overall security potential Mohamed *et al*. (2021) [23] and Darshana *et al*. (2022) [24].

3.4 Avalanche Effect in Hash Functions

The avalanche effect is directly relevant to hash functions because SecDcloud MSED2 uses a hash-and-salt function to generate its encryption key; the diffusion strength of the hash function directly affects the unpredictability of the key and therefore the cipher text.

Uyanik *et al*. (2022) [24] evaluated the avalanche effect of sixteen cryptographic hash functions and two hash-based applications HMAC and PKCS using an automated CrypTool circuit. The study tested two types of input: binary strings with one-bit reflections, and strings built from dictionary data, to satisfy both the Strict Avalanche Criterion (SAC) and the Bit Independence Criterion (BIC). Hash functions were subsequently ranked using a Multi-Criteria Decision Making (MCDM) approach and validated against NIST's fifteen statistical randomness tests. The study is particularly relevant to SecDcloud MSED2 because that algorithm uses a hash function to generate its encryption key; the avalanche properties of the hash directly affect the unpredictability of the key material and, in turn, the diffusion strength of the cipher text.

Simulation results showed that approximately half of the data inputs across the sixteen hash functions satisfied the SAC and BIC properties, while the remaining half failed, indicating that hash length alone does not guarantee strong avalanche behavior. The study confirmed that avalanche-effect testing is a more reliable indicator of resistance to collision, length-extension, and preimage attacks than key length, and recommended it as an essential evaluation step for any cryptographic scheme that incorporates hash functions including hash-based key generation schemes such as the one used in SecDcloud MSED2.

3.5 Summary and Research Gap

The reviewed literature consistently confirms the avalanche effect as the definitive empirical measure of diffusion strength for any cryptographic algorithm, with a 50% output-bit-change threshold universally accepted as the standard for a satisfactory result Dewangan *et al*. (1987) [16], Bore and Hemanth (2026) [17], Kumar and Tiwari (2012) [19], Bondar *et al*. (2025) [20], Maram and Gnanasekar (2018) [21], Bondar *et al*. (2025) [22], Mohamed *et al*. (2021) [23], and Darshana *et al*. (2022) [24]. Studies spanning AES implementations, dynamic S-box block ciphers, and hash function comparisons all use this criterion, and each confirms that algorithms which meet it are resistant to brute-force, differential, statistical, and preimage attacks, while those that fall below are vulnerable. Critically, however, none of the reviewed works evaluate the avalanche effect of an encryption algorithm purpose-built for distributed cloud storage using formal-methods-based data splitting and hash-and-salt key generation. SecDcloud MSED2, as proposed by Anah *et al*. (2023) [18], is precisely such an algorithm: it incorporates multiple diffusion mechanisms formal-methods sequence partitioning, XOR-based encryption, hash-generated and salted keys, and multi-server distribution but its overall avalanche effect has not been empirically determined.

4. Methods

Formal methods mathematical techniques such as logic, sets, sequence, and map operations applied to software design have been used to address security vulnerabilities systematically, including uncovering previously unknown attacks in the Needham-Schroeder public-key authentication protocol through model-checking tools Wing (1998) [12], Bringhenti *et al*. (2023) [13], Gavin (1996) [14], and Needham and Schroeder (1987) [15].

Anah *et al*. (2023) [18] applied this formal-methods approach to the cloud data storage problem, proposing the Secured Efficient Distributed Storage in Cloud (SecDcloud) MSED2 algorithm as an improvement on the earlier Security-Aware Efficient Distributed Storage (SA-EDS) algorithm. In SecDcloud MSED2, user data is partitioned into four functional units in a sequence and subjected to head-and-last pairing operations; a key is generated using a hash function and then salted before being XORed with the paired data units, and the resulting cipher text components are distributed across separate cloud servers to prevent any single party from reconstructing the original data. The formal-methods basis of SecDcloud gives it a principled design; however, design alone does not guarantee cryptographic strength that can only be confirmed by evaluating its avalanche effect.

We consider the existing SA-EDS Algorithm, there exists an input data D, and the data is split into two components, C and R, represented as *D*, C is randomly generated such that component R can be gained by D-C, The original input data D=C+R. Next, the key is generated randomly and XORED

with both C and R. The process is represented as α= *C XOR key* and β = *R XOR key,* The outcome of the encryption process is the Encrypted text α and β and later stored on separate servers, a detailed model is discussed in Jewargi (2023) [4]. The challenge with the approach is the key generation, in which there is a possibility of generating the same key.

In the modified concept coined MSED2, which is also part of the SecDcloud algorithm, User data is partitioned into four functional units in a sequence, and subject to sequence operation [Head, Last]. The Head and the last are paired and encrypted as well middle. The key is generated using a hash function and salted then XORED with the paired input. The encrypted data are then stored on different cloud servers. This method is proposed to prevent cloud data center operators from having access to original user data as shown in Fig 1.

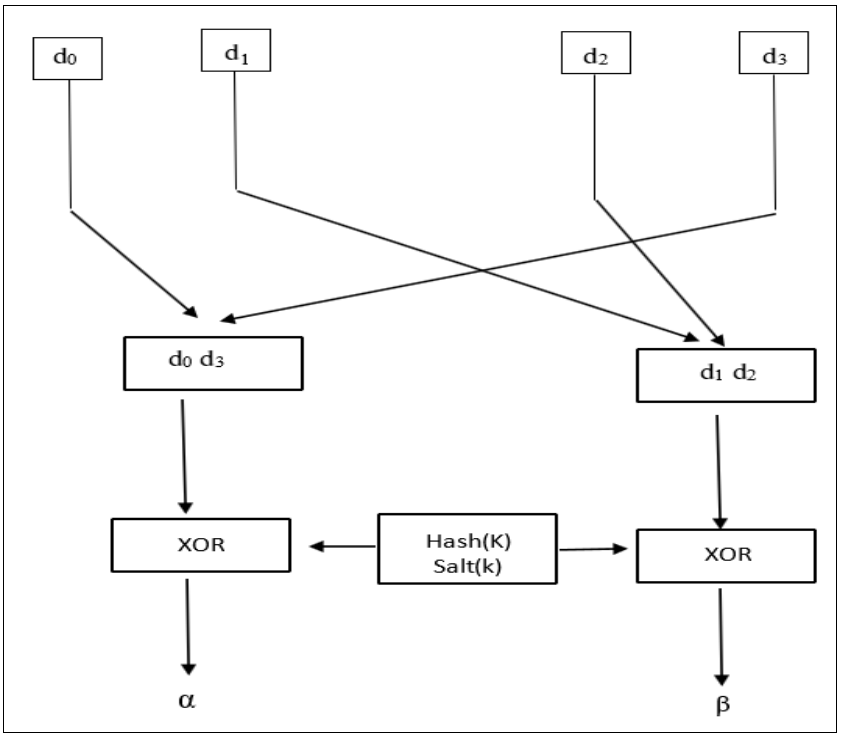

Fig 1: Modified Secured Efficient Data Distribution (MSED2) Model

As illustrated in Fig 1, the SecDcloud MSED2 system operates as a five-stage pipeline that carries the input data from plaintext through to distributed cipher text storage. In the first stage, the input data D is partitioned into four functional units arranged in sequence. In the second stage, a sequence operation, Head and Last, is applied to the partitioned units: the first unit (Head) is paired with the last unit (Last), while the two remaining middle units are paired together, so that every unit is combined with another before encryption rather than being processed in isolation. In the third stage, an encryption key is generated by passing an input through a hash function and then salting the resulting hash value; the salting step ensures that the same input does not repeatedly yield the same key, which resolves the key-collision weakness identified in the original SA-EDS scheme. In the fourth stage, each paired data unit is XORed with the salted, hashed key to produce a corresponding cipher text component, denoted α for the Head-Last pair and β for the middle pair. In the fifth stage, the resulting cipher text components are transmitted to and stored on separate cloud servers rather than a single server, so that no individual cloud data center operator holds a complete cipher text or has access to the information needed to reconstruct the original data. It is the combination of these five stages, sequence-based pairing, hash-and-salt key generation, XOR-based encryption, and multi-server distribution, working together rather than any single stage in isolation, that gives SecDcloud MSED2 its layered diffusion and confidentiality guarantees, and this is the end-to-end flow that Fig 1 depicts.

The inputs include the initial data that consist of user data packets represented as terms in sequence. Sequence and its operations are a part of Model based Approach in adapting formal method which is the basis for this work. The outputs are two separate data packets that will be transmitted to different cloud storage servers. The new generated data packets need to hide user data so that the cloud operators cannot read and understand the information, even though they have the access to the data.

Model Expression

This can be expressed in two folds Storage and Retrieval Storage:

Consider the Sequence D = {d1, d2, d3, d4} Where D is d1, d2, d3, d4 form the terms represent data packets which is in form of plain text Perform sequence operations [head, last] on D as well as middle Such that D1 = d1d4 D2 = d2d3 //When operations are executed and encrypted on the data packets, it generates, Then D1 and D2 are encrypted and sent or stored in Cloud A and Cloud B. Retrieval(Reorganising) Reorganising is performed with mid first, and Decrypted back to its original form Operations: Mid first Decrypt back to original form.

Pseudocode for SecDcloud MSED2

1. For all the terms d1, d2, d3, d4 in sequence D, as sensitive

text (Assume that the plaintext is sensitive)
2. Perform operation [Head, Last] as well as middle such that D1=d1d4, D2=d2d3
3. Generate Key, XOR with D1, D2
4. Store on Cloud A and B
5. End.
***Note that D1 and D2 represent alpha and beta**

Algorithms
Modified Alternative Data Distribution (MAD2) Algorithm
Require: NDP, PNL
Input NDP, PNL

1: For NDP do
2: for each data packet do
3: if ∃ a li ∃ PNL then
Execute MSED2 Algorithm / Algorithm */*
4:
5: else
6: Do XOR operation to the data packet
*7: /*Do XOR operation before the data packet is sent out*/*
8: Generate D xor
9: end if
10: end for
11: Obtain the values of D

Modified Secured Efficient Data Distribution (MSED2) Algorithm
Require: *D //non empty*
Ensure: α, β

1: Input D
2: Initialize $^{\alpha}\leftarrow 0$, $^{\beta}\leftarrow 0$
3: Randomly generate a key K
*5: **for** all input data packets **do***
*6: $^{\alpha} \leftarrow$ d1dn xor K / * head, last*
7: $\beta \leftarrow$d2dn-1 xor K
*8: end **for***
9: output α β

Implementation
The experimental environment was configured as follows. The Proposed Scheme was implemented and tested on a windows machine with a 9th gen corei7 processor, 16GB of RAM, 2Terabyte storage and windows 10 Operating system. Google Server at different locations was used Python was used as language of choice. Table 4 below shows the Avalanche effect of Secdcloud Encryption Algorithm. The Avalanche Effect of Algorithm is 54% on average, which satisfies the avalanche standard. The Avalanche is computed as:

Avalanche Effect = (Number of Changed bit in cipher text) /(Number of bits in cipher text)

Table 4: Avalanche effect of SecdCloud Encryption

Key	Cipher text	Avalanche Effect		
$2a$09$4r0TB2hToCS1CgILneEUKu0gqvWIZ8FpzGiWsgGEHpp8UB7fiOieK	S□2w□r□□A□h□ڜ0□y__ g□□OA□>o{□			
$2a$09$yRwB94BqOwJQjgZr76hOwem1w7S0mBx6IcHX2.xfOM32zQNH1xMFS	□Ys□□|□□4 쬡 iqm□X□s□ 譚□□|qD-	74	0.5632	
$2a$09$eYIJeT9WDZpghk81YxuzEO1NsWxuGbzwU8llVfoRpQBej3ILf6Z3G	h<□□□q□W□□□-□□□□□□b1□ L□□□	65	0.5048	
$2a$09$pFqN7dQjqMER0/RpJrD1hOszvDFVNUPcjUzACJ41mFSF6rCqMpwH6	□cF□yF□S□□□G□9□□g'□□(□X□□□□□	72	0.5172	
$2a$09$AEEKtUliYAHqzFA5fG7wB.3SETWv0z6uqAhhlez3FbCUh.7P01dDm	□□□□□7□A□□□□&H>□□□$□□iXOZ□□	66	0.56	
$2a$09$9AQaxHtjBY1NlMdM5i4WmeLgOHQ8y9fc.OahUGYgQVNbxQelVCMve	□□□x□D□3B□-□|6□P□□z□□w□□`□	70	0.5636	
$2a$09$ijwQByIUCRjTXf55R1n8nuPxSABYJnJJDQj70SoklyIcDzl.RbMO6	(□□S□n□U^]T□□□□gSꞏqg□d□□□	65	0.5248	
$2a$09$ayjmBaIVsYIFNnZizQKju.R8IJsCkrQzieo6sSf1ycxKpJilbgGBa	□!C□□y□□2□_□□e2Yn□l□□X□	66	0.5616	
$2a$09$QLm4tivSteEye9.A7usuJepqik25WnM06jE3XifPZLy7lpxlGYIRa	B.IT□□di□X□c-□□;□zP}□□(□?	65	0.568	
$2a$09$9TvO23Y28hheBL39yI9C7OIU0jNn.PZ6Zo6dSQ3nH8C7r9Xu0F5DC	□5(□□@□□5□A□□?W0v□AW□bh	68	0.5616	
$2a$09$YJh5xf69jaQCu9D8hIDqLOibVAUqfnvGnqoO9wvWQeNd9gxNb7/He	□□□x□□-□%□□*ϬL□d□□□6q□□P\□l□□	73	0.568	
			Average Avalanche Effect = 54%	0.54928

5. Conclusion
This paper presents the result of the Avalanche effect of the SedDcloud Encryption Algorithm, The avalanche effect is one of the desirable properties of Cryptographic algorithms, The study shows that the average key Avalanche effect is 54% which implies it will be hard to break the encrypted text, the reasons are that the input data is split, and paired using sequence and operations [head, last] sequence, secondly the key is generated using hash functions are difficult to break and Thirdly the generated hashed key is

salted which makes it more difficult to crack. Finally, the storing on different cloud servers will make it difficult to gain the original data, since the other part is stored differently. This method when adopted might go a long way in mitigating the security challenge in the cloud. A slight change in either the key or the plain text should result in a significant change in the cipher text. This property is termed as avalanche effect. In simple words, it quantifies the effect on the cipher text with respect to the small change made in plain text or the key.

6. References

1. Peter M, Tim Grance. The NIST definition of cloud computing, 2011.
2. Devi P. Literature Review on Cloud Computing: A Paradigm Combining Service-Oriented Architecture with Internet-Based Solutions. IJARSCT, Sept 2024, 178-184. Doi: 10.48175/ijarsct-19629
3. Pansotra EA, Singh ESP. Cloud Security Algorithms. International Journal of Security and its Applications. 2015; 9:353-360.
4. Jewargi K. What is Cloud computing Benfits and challenges of Cloud. GJCS, Mar 2023; 13(1):28-33. Doi: 10.18844/gjcs.v13i1.8622
5. Potey MM, Dhote C, Sharma DH. Homomorphic Encryption for Security of Cloud Data. Procedia Computer Science. 2016; 79:175-181.
6. Ginanjar MG, Lubis M, Ramadani L, Dwi Handayani DO. Enhancing Security and Privacy in Cloud Computing: Challenges and Solutions in the Digital Age. Institute of Electrical Electronics Engineers, Aug 2024, 1-6. Doi: 10.1109/iccit62134.2024.10701125
7. Li Y, Gai K, Qui LM, Zhao H. Intelligent cryptography approach for secure distributed big data storage in cloud computing. Information Sciences. 2017; 387:103-115.
8. Nadeem M, Arshad A, Riaz S, Zahra SW, Dutta AK, Alruban A, *et al*. Two-Layer Security Algorithms to Prevent Attacks on Data in Cyberspace. Applied Sciences. 2022; 12(19):1-21.
9. Choubey J, Pandey D, Patel I, Bairagi N. Security Challenges and Solutions In Cloud Computing: A Critical Review. International Journal of Innovative Research in Computer and Communication Engineering. 2022; 10(11):8545-8550. Doi: https://doi.org/10.15680/ijircce.2022.1011049
10. Maqsood F, Ahmed M, Ali MM, Shah MA. Cryptography: A comparative analysis for modern techniques. International Journal of Advanced Computer Science and Applications. 2017; 8(6):442-448.
11. Jøsang A. Cryptography. In *Cybersecurity*, Springer Nature, 2024, 63-97. Doi: 10.1007/978-3-031-68483-8_4
12. Jeannette M Wing. A symbiotic Relationship Between Formal Methods and Security. In Proceedings of the ONR/SNF Workshop on Computer Security, Dependability, and Assurance: From Needs to Solution, Washinton DC, USA, December 1998, 26-38. Also available as Carnegie Mellon University report CMU-CS-98-188.
13. Bringhenti D, Sisto R, Valenza F, Yusupov J. Introduction to Formal Methods for the Analysis and Design of Cryptographic Protocols, Crc, 2023, 57-104. Doi: 10.1201/9781003090052-2
14. Gavin L. An Attack on the Needham-Schroeder Public - key Protocol using FDR. Proceedings of the 2nd International Workshop on Tools and Algorithms for the Construction and Analysis of Systems. (TACAS'96), Passau, Germany. 1996; 1055:147-166.
15. Needham R, Schroeder M. Authentication Revisited. Operating System Review. 1987; 21(1).
16. Dewangan CP, Agrawal S, Mandal AK, Tiwari A. Study of avalanche effect in AES using binary codes. IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT). IEEE, 1987, 183-187.
17. Bore GSB, Hemanth KAR. Construction of dynamic S-boxes with improved strict avalanche criteria, Crc, 2026, 516-522. Doi: 10.1201/9781003773504-88
18. Anah HB, Boukari S, Gital AY, Abdulhamid M. An Encryption Approach Based on Formal Method for Securing Distributed Big Data Storage in Cloud Environment. Indonesian Journal of Computer Science. 2023; 12(1).
19. Amish Kumar, Namita Tiwari. Effective Implementation and avalanche effect of AES. International Journal of Security, Privacy and Trust Management (IJSPTM). 2012; 1(3-4).
20. Bondar V, Myroniuk T, Lavdanskyi A, Babenko V. Research on the Avalanche Effect of a Cryptographic Algorithm Based on Information-Controlled Permutation Operations. Institute of Electrical Electronics Engineers, Sept 2025, 1-6. Doi: 10.1109/idaacs68557.2025.11322399
21. Balajee Maram, Gnanasekar JM. A Block Cipher Algorithm to Enhance the Avalanche Effect Using Dynamic Key Dependent S-Box and Genetic Operations. International Journal of Pure and Applied Mathematics. 2018; 119(10):399-418. ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version) url: http://www.ijpam.eu Special Issue
22. Bondar V, Myroniuk T, Lavdanskyi A, Babenko V. Research on the Avalanche Effect of a Cryptographic Algorithm Based on Information-Controlled Permutation Operations. Institute of Electrical Electronics Engineers, Sept 2025, 1-6. Doi: 10.1109/idaacs68557.2025.11322399
23. Kamsiah Mohamed, Mohd Nazran Mohammed Pauzi, Fakariah Hani Hj Mohd Ali, Suriyani Ariffin. Analysis on Avalanche effect in cryptographic. ICONSPADU International Conference on Sustainable Practices, Development and Urbanisation, 2021.
24. Darshana U, Nupur G, Marzia Z, Srinivas S. Investigating the Avalanche Effect of Various Cryptographically Secure Hash Functions and Hash-Based Applications. Research article Received. Digital Object Identifier. Doi: 10.1109/ACCESS.2022.3215778