Bulletin of Electrical Engineering and Informatics
Vol. 15, No. 2, April 2026, pp. 1879~1900
ISSN: 2302-9285, DOI: 10.11591/eei.v15i2.10723

Cloud security: an upcoming development in hybrid automated storage protection systems

Srividya Bharadwaja VenkataSubbu[1], Smitha Sasi[1], Meharunnisa Sakkar Sab Pakeer Saheb[2], Harikeerthan Mysore Keshava Rao[3], Soumya Seetharamaiah[4]
[1]Department of Electronics and Telecommunication Engineering, Dayananda Sagar College of Engineering, Bengaluru, India
[2]Department of Electronics and Instrumentation Engineering, Dayananda Sagar College of Engineering, Bengaluru, India
[3]Department of Civil Engineering, Dayananda Sagar Academy of Technology and Management, Bengaluru, India
[4]Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, India

Article Info

Article history:

Received May 21, 2025
Revised Oct 6, 2025
Accepted Mar 5, 2026

Keywords:

Advanced encryption standard
Attack
Cloud storage
Elliptic curve cryptography
Hybrid-cryptosystem
Ternary Galois field

ABSTRACT

The need for cloud storage is growing daily, despite its apparent reliability and scalability. Because of the complexity of data security, including privacy and confidentiality, it is difficult to store and manage user data remotely. An untrustworthy insider could be the cloud third party responsible for managing the data security issues occurring within the cloud services. An automated hybrid cryptographic algorithm for cloud data storage is provided by the current research to improve data protection and guarantee information confidentiality without any disruption from cloud third parties. Two encryption methods that are employed to attain high performance and efficiency are enhanced automated concurrent encryption for data blocks (CEDB) and improved automated asynchronous encryption for data streams (AEDS). By converting the fixed S-box in the traditional AES to a modifying S-box using a ternary Galois field (GF) and complementary ternary GF, we present a novel encryption technique in enhanced automated concurrent encryption. Furthermore, the matrix utilized for the AES mix column approach is converted into a dynamic matrix using the ternary GF and complementary ternary GF. This cryptographic approach aims at providing cloud data security, ensuring confidentiality, and integrity for the user's information even in compromised situations.

Corresponding Author:

Srividya Bharadwaja VenkataSubbu
Department of Electronics and Telecommunication Engineering, Dayananda Sagar College of Engineering
Bangalore, Karnataka, India
Email: srividyabv@gmail.com

1. INTRODUCTION

Cloud computing provides services includes storage, servers, and apps that are scalable, affordable, as well as dependable [1], [2]. Cloud data storage, however, possess serious security risks, especially with regard to data availability, confidentiality, and integrity. Unauthorized access, manipulation, and possible insider threats can affect sensitive data stored on cloud platforms, particularly when critical security features are managed by third-party service providers. The effectiveness and confidence in the cloud environment must be preserved while addressing these issues with current procedures.

Cloud security issues and solutions have been examined by a significant number of researchers which are as listed:

- The functions and security implications of cloud services has been clarified by researchers who have classified them into three categories such as: infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) ([3]-[5]).
- Typical security concerns: authors of the article ([4], [6], [7]) emphasizes the risks of unapproved access to data, breaches in data integrity, and problems with availability in cloud data centres. Making data available means enabling authorised people to access and utilise it whenever they need to [8]. The methodology of guaranteeing data integrity is to ensure that information accurateness and reliability, is not lost or altered by any intruders [9].
- Third-Party Hazards, wherein the insider threat issue is one of the weaknesses brought about by assigning security duties to outside parties, as covered in the work of [10]-[13].
- Cryptographic solutions: before uploading data to the cloud, several researchers have suggested the usage of cryptographic techniques such access control and encryption ([14]).

These findings demonstrate that even though cloud computing has many advantages, security risks, particularly those pertaining to confidentiality and insider threats still remain a major concern. It is therefore advised to encrypt data before uploading it.

This research focuses on the automated encryption/decryption system to improve security without compromising on its effectiveness, when there is a threat issue from an insider. Issues of trust with third-party providers and the risk of insider threats are still not well addressed according to the previous study. It's possible that current encryption algorithms aren't optimal for cloud environments in terms of both security and performance. The proposal is to use hybrid cryptographic techniques that combine the advantages of multiple algorithms, such as AES for block data encryption and elliptic curve cryptography (ECC) for stream data encryption as well as key exchange, specifically designed for cloud environments. There is a dearth of reliable, automated encryption/decryption systems that incorporate altered cryptographic methods to improve security without compromising effectiveness.

This article proposes, a system of automated information cryptography, created especially for the protection of cloud data. A new hybrid encryption/decryption technique that combines modified AES with modified ECC to take use of both AES's quick symmetric encryption and ECC's lightweight key management and encryption is introduced. Also, standard ECC and AES algorithms are improved to meet cloud security requirements and lessen third-party and insider threats. The emphasis is on real-world application in cloud platforms that guarantees data availability, confidentiality, and integrity. This strategy seeks to close the gap left by current solutions by offering customized, cloud-optimized hybrid cryptography architecture.

This article is divided into 5 sections. The literature review in section 2 goes into further detail on current cloud security issues, cryptographic fixes, and research gaps. Section 3 elaborates on methodology. Also, describes the hybrid cryptography system's architecture, including the automation procedure and changes made to the ECC and AES algorithms. The explanation of the algorithm and few illustrations are listed. In order to illustrate the efficacy and applicability of the suggested system, section 4 is on results and discussion, which provides performance benchmarks, security analysis, and comparison with current techniques. In section 5, conclusion and future work, contributions are compiled, practical consequences are highlighted, and areas for improvement are suggested.

The AES algorithm was developed by Daemen and Rijmen as a symmetric encryption method. The basis of AES is typically a combination which has both substitution and permutation computations. Wherein a series of interconnected operations, need the replacement of certain outputs for inputs (termed as substitutions), while others demand the permutation of bits. All calculations in the AES algorithm employ bytes rather than bits. In order to process the 128 bits of plain text as a matrix, the 128 bits are divided into 16 bytes and arranged as a 4×4 matrix. The number of rounds in AES can be changed according to the key length. For 128-bit keys, it uses ten rounds; for 192-bit keys, 12 rounds; and for 256-bit keys, 14 rounds.

The following steps are part of the AES encryption process:

- Byte substitution: the basis for these operations is a substitution box (S-box) made specifically for this function. The resulting data matrix is of dimension four by four.
- Row shift: the elements of the rows of the matrix are moved towards the left, and omitted elements which are dropped are replaced on the right side of the row.
- Mix columns: each four-byte column is now modified using a specific mathematical function. The four bytes from the original column are replaced with four new bytes by this function, which accepts four input bytes. This results in the creation of a new matrix with 16 more bytes. In the final round, this phase is skipped.
- Add round key: the 16 bytes of the four by four matrix are now considered as 128 bits, and they are then XOR ed with the 128 bits of the round key. The output will be the cipher text if this is the final round. Otherwise, 16 bytes are created from the 128 bits, and a new cycle starts.

The ECC is an asymmetric cryptographic technique designed and implemented by Neil Koblitz. ECC employs dual keys for the purpose of data encryption and decryption, as is the case with any type of asymmetric method. While the private key is used solely to decrypt the data and must be kept secret, the data are encrypted using a public key that is shared with other users. The computations in the ECC algorithm can be described as follows to represent the enciphering and deciphering processes:

a. The finite field considered is F_p^m and P is the generator point which has the order #E (F_p^m)=n
b. The message to be encrypted is chosen from the cubic polynomial over the finite field F_p^m
c. Key generation:

– User A chooses a unique random integer from [1 to n-1] and keeps it private
– Entity A computes the and publishes it
– Similarly, user B chooses a unique random integer from [1 to n-1] and keeps it private
– Entity B computes the and publishes it

d. The session key is computed using
e. The field element is computed by entity B using the session key
f. Entity B generates the cipher text
g. During decryption, entity A computes the session key using
h. The field element is computed by entity A using the session key
i. A recovers the message

The enhanced advanced encryption standard (AES) and ECC algorithms have been employed in this investigation because of their efficiency. As an effective encryption technique for huge quantities of data, AES is used. However, despite being a stream cipher, the ECC functions as a very sophisticated and powerful algorithm used to protect data. Using a particular encryption algorithm to ensure data privacy and confidentiality in open systems like cloud platform could make it susceptible to any sort of hacking as well as privacy violations, not withstanding the advantages of the encryption methods listed above. As a result, as will be detailed in the subsequent section, we provide unique methodologies based on the amalgamation of these traditional algorithms.

The data needs to be encrypted prior to exporting it on the cloud platform. The following are the proposed module's primary contributions:

– Limiting cloud third party operations and improving data privacy.
– Create blocks and streams of original data.
– Creating encryption keys and their mathematically related counterparts, the decryption keys, in an amount proportional to the quantity of data streams and blocks.
– In order to be used in the decryption process, the shared key and a pair of public-private keys are saved in a certified database.
– Using a modified AES algorithm over a ternary Galois field (GF) to encrypt data blocks. Here, dynamic S-box and dynamic matrix are used during encryption. Ternary is used as each trit encodes around 1.585 bits of information. This reduces the number of digits required for storing the data of same range and thereby saves space.
– Using a modified ECC technique over a ternary GF to encrypt the data stream.
– Comparing the suggested methods with existing cutting-edge encryption techniques, such as AES and ECC, over the binary GF.

2. LITERATURE REVIEW

Several recent research on cloud security have focused on removing the inclusion of the cloud third party or lowering the severe threats due to their involvement. A way to assure that the integrity of information is kept secured on a remote cloud server is with the support of a dependable third-party auditor. This technique was suggested by Chakraborty *et al.* [15] of the paper entitled "Integrity checking using third party auditors in cloud storage". In the data auditing process, the auditor has been established to be an expert agent. The auditor is given with a task of recovering a file tag, verifying the signature, and reporting in case of any invalidity.

More and Chaudhari [16] have suggested a novel approach for protecting the confidentiality and integrity of data. This strategy makes use of third-party cloud computing and user-based data encryption. The owner of the information initially permits certain operations on it, such as dividing it into blocks of data, encrypting and generating a hash value, subsequently concatenating, affixing a signature. This is eventually uploaded on to the remote storage.

The accountability of the cloud third party is later seen in the auditing of public data. The cloud third party verifies the accurateness of the data by creating hash values for each signature that is concatenated and created in the cloud environment, as well as for the blocks of encrypted cipher that the sender has

obtained. Finally, the third party compares both the signatures, just to ensure that the encrypted data has not been altered. This approach has encountered multiple cloud third party issues, and it significantly increases customer workload. The authors of [15], [16], have used cloud third parties, which may lead to vulnerabilities in trust, accountability and overhead on third party dependent systems.

Akhil *et al.* [17] offered another approach to restrict the function of the third party. The authors have created a technology that offers secured data without any third party intrusion. Sending encrypted data to the cloud server for storage. Encryption is achieved using the AES encryption method. With this method, the system's internal encryption and decryption are hidden from the third-party auditor.

Additionally, Sivakumar *et al.* [18] used the AES method to secure data on the Heroku cloud. Encryption and subsequent transferring of data from the use of AES demand strong security as the system uses the concept of shared key for encryption and subsequent decryption of the same. This problem might make it simple to access the original information to retrieve it. The authors of [17], [18] have used shared key cryptosystem, which is very efficient but has issues with the key management.

According to Orobosade *et al.* [19], there is yet another method for presenting a security system that is independent of a third party. These researchers have suggested a hybrid encryption technique that combines cryptographic methods that are shared key based and public-private key based that are developed in order to protect user privacy and security in the cloud. These authors have presented a cryptographic system that involves AES and ECC, to obtain a privacy paradigm. A hybrid framework for cloud data security was created by Hosam and Ahmad [20]. ECC, an asymmetric cryptography technology, and the symmetrical algorithm AES are used to enable safe data interchange and storage [20]. In contrast to the single symmetric encryption method, this research study suggests a hybrid symmetric encryption methodology to provide users' data stored in the cloud with extra protection [21]. Sasikumar and Nagarajan [22], give a comprehensive review on the various cryptographic algorithms that can be utilized for cloud services. The usage of ECC, multilevel storage, split and combines strategies, duplication procedures, and Hash-Solomon coding has been suggested by Ahamed and Duraipandian [23]. This strategy is highly secure. However, advanced cryptography techniques were required.

The authors of [19]-[23] have used a hybrid framework where AES and ECC are integrated together. This strategy is more secured than the preceding ones. Hence a variation of the hybrid framework along with ternary GF, dynamic S-box, varying matrices for mix column approach, splitting the data into blocks and streams for parallel encryption is the highlights of this article.

Shrivastava *et al.* [24] employed the neural network approach, enhanced MD5, and enhanced ECC. This method used less computational power while still providing security. However, it was put into practice in decentralized storage. Asymmetric cryptosystems like RSA and ECC were employed by the authors in [25], who also used a search algorithm to optimize the secret key. The issue of security was resolved. The isolation feature, however, needed to be improved. The association rule mining method was employed as a novel key selection management technique. Decoding and encryption became less complicated. Nevertheless, more effective key generation methods had to be employed [26]. The authors present genetic cryptography approaches. This method performs better than all symmetric key crypto methods in terms of security level, throughput, and time. Nonetheless, genetic cryptography is still in early stages [27]. Saxena and Alam [28] suggest RSA with a new division-based PHE scheme and partial homomorphic encryption. This approach reduced the time needed for encryption and decryption. It was necessary to validate trust-based access controls, though. Gusset [29] elaborates about the adoption of AES-256 in platforms such as AWS Azure, for data security and encryption, during the transit and at rest of the data. This scheme provides 98% data protection with a latency of 5%.

From the preceding discussion, we may draw the following conclusions about the inadequacies of current methods for protecting data confidentiality in cloud environments: i) because symmetric methods only require a single shared key for encryption and decryption, using a unique type of algorithm (non-hybrid approaches) is not sufficient to guarantee higher levels of security and ii) the systems that use the manual, non-automated concepts use cloud third parties to ensure they are reliable and haven't been turned into a perverse insider. It is difficult to ensure this issue in actual practises.

The proposed hybrid protection system offers the following major features when compared with the existing architecture.

- The proposed scheme integrates light weight ECC and AES, which is a faster encryption technique. This hybridization results in both efficiency and robust security.
- Conventional cryptographic techniques work with binary GF, while the proposed system uses ternary GF, which introduces a higher algebraic complexity and resistance against cryptanalytic attacks.
- The adoption of dynamic S-box allocation, substitution and transformation offers protection against differential and linear attacks.

In this work, the authors will discuss the topic of data confidentiality from a different angle than that used in the studies mentioned above. By creating a fully portable cryptographic system for cloud data storage the proposed algorithm intends to reduce the involvement of the cloud third party. The proposed algorithm is based on giving an autonomous system, control over all security operations on the data, starting with the user's upload of the data to the cloud platform, moving through encryption to make it prepared for loading, and concluding with decryption when the receiver requests to recover the data.

3. METHOD

The proposed method protects data by using two different kinds of encryption techniques. The data is split up as blocks and streams before being processed. Also Algorithms 1 and 2 use abstract algebra which is of a special kind, making it more suitable for applications that require data protection.

The hybrid cryptographic technique combines ECC over ternary GFs for stream encryption with dynamically generated S-box for block encryption, improving on conventional AES and ECC. The modified AES strengthens defenses against differential and side-channel attacks by utilizing dynamic replacement Substitution-box, in contrast to the standard AES, which employs a fixed S-box and functions over binary fields. Furthermore, in contrast to ordinary ECC, which usually depends on binary or prime fields, the use of ECC over GF(3^m) and its complementary field introduces new algebraic structures that are less vulnerable to traditional cryptanalytic techniques. By dividing data into parallel streams and blocks that are each encrypted using different techniques, the suggested solution enhances security and performance while providing more resistance to attacks and greater appropriateness for systems that need data protection.

In the proposed system, as shown in Figure 1, the input data is divided into two categories: data blocks and data streams.

- In order to ensure robust and effective encryption for large amounts of data, the data blocks are encrypted in parallel using a modified version of the AES algorithm.
- Following further division of the data streams into Streams 1 and 2, each stream is encrypted using a modified ECC technique. A ternary GF (GF(3^m)) is used to define ECC for the encryption of Stream 1. While the complementing ternary GF is used to define ECC for the encryption of stream 2.

The system's overall security and performance are enhanced by this dual strategy, which combines the mathematical power of ECC-based stream encryption with the speed of block-based AES encryption.

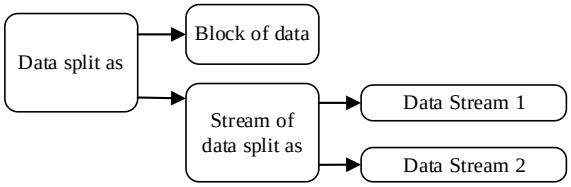

Figure 1. Segmentation of data

The proposed hybrid algorithm offers a novel cryptographic model that combines the proven strength of AES and ECC with enhancements in field structure due to the usage of ternary GF, dynamic S-box and dual ECC streams. The cryptographic algorithms are defined over ternary GF. Hence the following section explains the same.

Ternary GF and its 3's complement representation: this study emphasizes on the p's compliment form representation of the GF elements, denoted as $GF((\tilde{p})^m)$. This approach is used to enhance information security. As an illustration, let us consider GF (3^3). The elements of GF (3^3) are constructed using the primitive polynomial$p(x) = x^3 + 2x^2 + 1$.
Let α be a root of the polynomial p(x).

$p(x = \alpha) = \alpha^3 + 2\alpha^2 + 1 = 0, \alpha^3 = -2\alpha^2 - 1$
Hence $\alpha^3 = \alpha^2 + 2$

As can be seen, Table 1 shows the elements of $GF(3^3)$ constructed using the primitive polynomial $p(x) = x^3 + 2x^2 + 1$. The polynomial representation, the ternary representation, the decimal representation and the proposed novel complimentary representation of the GF elements are depicted in Table 1.

Table 1. The elements of $GF(3^3)$

Elements	Polynomial representation	Ternary and decimal representation weightage=3^0 3^1 3^2	Representation in 3's complement form
0	0	(000)3=(0)	001
1	1	(100)3=(1)	201
α	α	(010)3=(3)	221
α^2	α^2	(001)3=(9)	000
α^3	$2+\alpha^2$	(201)3=(11)	100
α^4	$2+2\alpha+\alpha^2$	(221)3=(17)	010
α^5	$2+2\alpha$	(220)3=(8)	011
α^6	$2\alpha+2\alpha^2$	(022)3=(24)	202
α^7	$1+\alpha^2$	(101)3=(10)	200
α^8	$2+\alpha+\alpha^2$	(211)3=(14)	020
α^9	$2+2\alpha+2\alpha^2$	(222)3=(26)	002
α^{10}	$1+2\alpha+\alpha^2$	(121)3=(16)	110
α^{11}	$2+\alpha$	(210)3=(5)	021
α^{12}	$2\alpha+\alpha^2$	(021)3=(15)	210
α^{13}	2	(200)3=(2)	101
α^{14}	2α	(020)3=(6)	211
α^{15}	$2\alpha^2$	(002)3=(18)	222
α^{16}	$1+2\alpha^2$	(102)3=(19)	122
α^{17}	$1+\alpha+2\alpha^2$	(112)3=(22)	112
α^{18}	$1+\alpha$	(110)3=(4)	121
α^{19}	$\alpha+\alpha^2$	(011)3=(12)	212
α^{20}	$2+2\alpha^2$	(202)3=(20)	022
α^{21}	$1+2\alpha+2\alpha^2$	(122)3=(25)	102
α^{22}	$1+\alpha+\alpha^2$	(111)3=(13)	120
α^{23}	$2+\alpha+2\alpha^2$	(212)3=(23)	012
α^{24}	$1+2\alpha$	(120)3=(7)	111
α^{25}	$\alpha+2\alpha^2$	(012)3=(21)	212
α^{26}	1	(100)3=(1)	001

As an illustration, the element $\alpha^7 = 1+\alpha^2$, represented as (101) in ternary and 10 in decimal. While the same element is represented as (200) in ternary and 2 in decimal. In this article, to obtain the complimentary value, the following calculations are performed. The value 101 in ternary is considered. Every trit of 101 is subtracted from 2. The resulting value is 121. To this 121, $(2)_3$ is added to the right most value, with the carry propagating to the left. This results in 200 in ternary. This approach is denoted as 3's complimentary approach in this article.

$$222 - 101 = 121$$
$$121$$
$$\underline{+2}$$
$$(200)_3$$

In the proposed algorithm, mathematical operations such as multiplication, addition, inverse and the necessary modular arithmetic operations use the complimentary representation of data. For the ease of computation, the values are stored in the form of a look-up-table.

3.1. Proposed hybrid automated cryptosystem for encryption/decryption

Utilizing the advantages of both block-based and stream-based encryption techniques, the suggested cryptography system is built as a hybrid encryption framework, as shown in Figure 2. The hybrid nature preserves flexibility for various data kinds and operating scenarios while guaranteeing strong security. Two fundamental ideas serve as the framework for the architecture:

- Safeguarding user data by making certain that all private data is securely encrypted and impervious to cryptanalytic attacks.
- Despite system vulnerabilities, the security of encryption keys ensures that the keys themselves cannot be deduced or compromised.

Concurrent encryption of data blocks (CEDB): for data stored in blocks, the modified AES algorithm is used. Certain modifications are designed compared to the conventional AES algorithm to introduce a dynamic cryptographic behavior, such as:

- Ternary GF integration, to allow more diverse cryptographic transformations.
- Dynamic non-repeating S-box, which makes it more susceptible to differential attacks

– Varying substitutions and mix-column matrices, which introduce higher algebraic complexity and non-linearity.

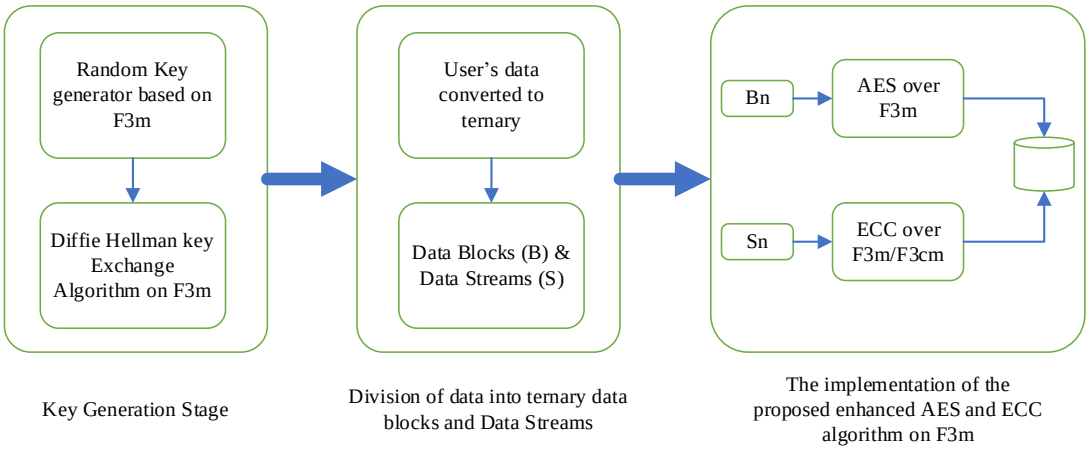

Figure 2. The proposed encryption/decryption system

Asynchronous encryption of data streams (AEDS): for continuous data stream the system uses ECC with ternary GF and complimentary ternary GF structures. Conventional ECC operates over binary fields. Arithmetic complexity enhances over ternary fields. This method enables parallel computation for stream encryption.

The security advantage of this hybrid approach is the layered protection mechanism. Even if one algorithm is compromised, the other algorithm provides protection. Dynamic S-box and variable matrices using ternary algebraic fields, leads to non-linear operations that are more resistant to cryptanalytic attacks.

The suggested work starts with a random key generator, which generates a random key K_i for each block of data B_i and stream of data S_i based on the random function generator, as illustrated in Figure 3. The coding is carried out using R2025b MATLAB online. Figure 3 shows the pseudo code for generating random keys using MATLAB functions.

```
p=3;
m=input ('enter the degree of the Galois field');
primitive_polynomial=gfprimdf (m,p);
Field_elements = gftuple ([-1 :(p^m)-2]',primitive_polynomial,p);
% Field elements pertaining to F3m
key=randperm (p^m-2, 1);
```

Figure 3. Pseudo code for generating random keys

Figure 3, depicts the pseudocode in MATLAB for generating random numbers. Choosing the value of p as 3 implies the prime field is ternary. Ternary GF (3^m) is considered. Also the degree of the field 'm' is taken as an user input. The primitive polynomial is generated based on the prime number 'p' and the degree of the field 'm'. The field elements are generated by considering the prime number 'p', and the primitive polynomial. Subsequently the "randperm" function is used to generate a random key from the chosen ternary GF elements. The value of 'm' and the corresponding primitive polynomial are decided by the transacting parties. In this article, smaller values of 'm' are chosen for the ease of representation and tabulation.

As shown in Figure 4, blocks of ternary data are encrypted and decrypted using the enhanced AES method (described in subsection 3.2, Algorithms 1(a) and 2(a)) with the help of randomly generated keys, which serve as shared keys. In parallel, streams of ternary data are encrypted and decrypted using the enhanced ECC method (explained in subsection 3.3, Algorithms 1(b), 1(c), 2(b), and 2(c)). The Diffie–Hellman key exchange protocol is used to securely exchange these keys. Importantly, each data block employs a unique shared key during encryption.

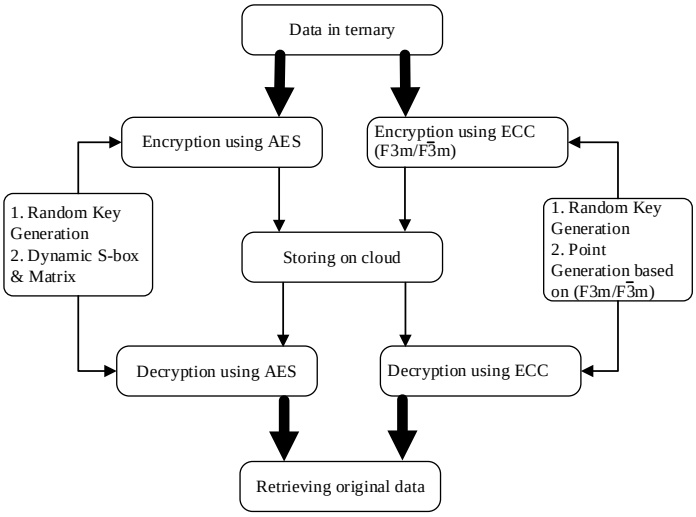

Figure 4. The proposed cryptographic system

The same randomly generated keys are also integrated into the enhanced ECC algorithm for public-key based encryption of data streams. A notable feature of this approach is the automated elimination of the private-key after encryption, which prevents reuse and strengthens security. These random keys ensure that all encryption keys remain secure and inaccessible to unauthorized parties. Finally, the encrypted data blocks are transmitted to the cloud, where they are securely stored and made ready for further access.

3.1. Enhanced automated concurrent encryption for data blocks

Another innovative method for the encryption of data suggested in this framework to improve data privacy and security is the automated random cryptography methodology. This encryption method's unique feature is encrypting the data segments using an enhanced AES. The unique feature of this algorithm is the dynamic S-box and dynamic matrix generated randomly without any repetition for performing the substitution and mix column functions that are a part of the AES algorithm, as shown in Figure 4. During the encryption of each data block, a dynamic random S-box and dynamic random matrix are generated for substitution and mix column operation. The inverse of the dynamic S-box and the matrix is utilized for decryption. The algorithm is briefly described in the following 3 steps.

a. Step 1. Key generation:
- A random master key is taken as input and converted into ternary representation.
- From the master key, six sub-keys (w0–w5) are generated.
- w0 and w1 are extracted directly from the leftmost and rightmost symbols of the master key.
- w2–w5 are derived iteratively using a dynamic S-box defined over GF(3^m).
- The S-box is generated randomly for each round and must be bijective to ensure the existence of an inverse S-box for decryption.

b. Step 2. Encryption process:

Encryption of plaintext is performed in the following steps:
- Convert plaintext: the plaintext is represented in ternary symbols from GF(3^m).
- Initial key addition: plaintext is XORed (or its ternary equivalent operation) with the sub-key.
- Substitution (S-box): each element undergoes substitution using the dynamic S-box.
- Swapping: the 2nd and 4th symbols of the state are swapped to introduce diffusion.
- Mix column transformation: a dynamic matrix over GF(3^m) is generated and multiplied with the state for enhanced diffusion.
- Repeated substitution and swapping: steps of substitution and swapping are repeated to increase complexity.
- Final round operations: after the last mix-column and substitution, the cipher text is obtained.

c. Step 3. Decryption process

Decryption is essentially the reverse of encryption, following,
- Initial key processing with the sub-keys in reverse order.
- Inverse swapping: swap the 2nd and 4th elements back.

- Inverse substitution: apply the inverse S-box (computed as soon as the S-box is generated during encryption).
- Inverse mix column: multiply by the inverse of the dynamic matrix used during encryption.
- Repeat inverse operations in reverse order of encryption steps.
- Recovered plaintext is obtained in its ternary representation, which can then be converted back to binary form.

d. The key features of the proposed algorithm is:

Dynamic components:

- Random non-repeating S-box and its inverse.
- Random matrix for mix-column transformation.
- Unique sub-keys for each encryption round.

The algorithm listed in Step 1 to Step 3 is explained as Algorithms 1(a) and 2(a). It includes the explanation with a suitable illustration. The encryption steps are illustrated in Algorithm 1(a) and decryption processes are elaborated in Algorithm 2(a), with illustrations from the elements of GF(3^3). All the information is represented as symbols from GF (3^3), as shown in Table 1.

Algorithm 1(a). Encryption process

Step 1: Input Random key (referred to as Master key), convert it to Ternary over F_3m

As an Illustration, let us consider the representation of a Plaintext in Binary, Ternary and Ternary Galois field. Binary data has 0s ans 1s. While Ternary data is represented using 0, 1, 2. Ternary Galois field representations consider symbols fromF_3m.

Plaintext in Binary :{ 000110,001001, 101001, 010110}

Plaintext in Ternary :{ 012,021,221,112} over$F_3 3$

Plaintext in F_3m:$\{\alpha^{15}, \alpha^{17}, \alpha^{23}, \alpha^{14}\}$

With the necessity to encrypt, the Plaintext values are represented with symbols from GF $(3m)$.

The cryptographic algorithm and working of the same with an example are as explained. The elements are represented using the symbols which are listed in Table 1.

Step1a: Generation of sub keys w0, w1, w2, w3, w4 and w5 from the Master Key

The sub-keys w0 and w1 are derived from the Master Key

The sub-key w0 (1, 2) is the two left-most symbols of the master key

The sub-key w1 (1, 2) is the two right-most symbols of the master key

The other sub-keys are generated as follows:

$$w2(1,2) = \{(s(w1(1)), x^{i+2}) \oplus (s(w1(2)), \alpha^2)\},$$

Where i=1, for round 1 and α^2 is chosen for computing w2

$$w3(1,2) = \{(s(w0(1)), x^{i+2}) \oplus (s(w0(2)), \alpha^3)\},$$

Where i=1, for round 1 and $\alpha^3$is chosen for computing w3

$$w4(1,2) = \{(s(w2(1)), x^{i+2}) \oplus (s(w2(2)), \alpha^4)\}$$

Where i=2, for round 2 and $\alpha^4$is chosen for computing w4

$$w5(1,2) = \{(s(w3(1)), x^{i+2}) \oplus (s(w3(2)), \alpha^5)\}$$

Where i=2, for round 2 and $\alpha^5$is chosen for computing w5

Illustration:

$$w0(1,2) = \{\alpha^{15}, \alpha^{17}\}\, w1(1,2) = \{\alpha^{23}, \alpha^{14}\}$$

To compute w2, w3, w4 and w5, a dynamic S-box is defined. As an illustration, dynamic s-box over F_3m is defined as follows.

An illustration of a dynamically generated S-box using the elements of GF (3^3) is depicted in Table 2. If the S-box is bijective, only then its inverse exists.

Table 2. S box using ternary GF

F_3m	(00)3	(01) 3	(02) 3	(10) 3	(11) 3	(12) 3	(20) 3	(21) 3	(22) 3
(0) 3	α^{-inf}	α^{23}	α^{12}	α^{17}	α^{6}	α^{9}	α^{1}	α^{4}	α^{18}
(1) 3	α^{5}	α^{10}	α^{11}	α^{2}	α^{16}	α^{14}	α^{22}	α^{20}	α^{8}
(2) 3	α^{19}	α^{15}	α^{13}	α^{24}	α^{3}	α^{7}	α^{25}	α^{21}	α^{0}

$$w2(1,2) = \left\{\left((s(w1(1)), x^{i+2}\right) \oplus \left((s(w1(2)), \alpha^2\right)\right\},$$

$$w2(1,2) = \left\{\left((s(\alpha^{23}), \alpha^3\right) \oplus \left((s(\alpha^{14}), \alpha^2\right)\right\}$$

Determining $s(\alpha^{23})$:
Consider the degree of the finite element whose substitution value needs to be determined. The degree of the element is 23, which is equal to (212)3. The value at the intersection of the row 2 and column 12 is α^7
Hence $w2(1,2) = (\alpha^7, \alpha^3) \oplus (\alpha^{14}, \alpha^2) = \{\alpha^{15}, \alpha^{12}\}$
Similarly:

$$w3(1,2) = \alpha^{25}, \alpha^{17}$$

$$w4(1,2) = (\alpha^{15,}\alpha^{9})$$

$$w5(1,2) = (\alpha, \alpha^7)$$

Step2: Input Plaintext and convert it to Ternary over F_3m
Illustration: Plaintext inF_3m: PT= $\{\alpha^3, \alpha^{25}, \alpha^{18}, \alpha^6\}$
Step3: Encryption: encryption algorithm is elaborated in step 3a to 3h.
Step3a: $E1 = Add(PT, Keys(w0, w1))overF_3m$
Illustration: $E1 = (\alpha^2\alpha^{25}\alpha^{18}\alpha^6) \oplus (\alpha^{15}\alpha^{17}\alpha^{23}\alpha^{14}) = (\alpha^{13}\alpha^{10}\alpha^6\alpha^{25})$
Step3b: E2=S (E1)
Illustration: $E2 = \{S(\alpha^{13})S(\alpha^{10})S(\alpha^6)S(\alpha^{25})\} = \{\alpha^{16}\alpha^{10}\alpha^1\alpha^{21}\}$
Step3c: Swapping the 2nd and 4th position ternary field element of E2
Illustration: $E3 = \{\alpha^{16}\alpha^{21}\alpha^1\alpha^{10}\}$
Step3d: E4=Mix Column
In this step, a dynamic F_3m matrix is generated, and matrix multiplication is performed with the results of E3.

$$E4 = \begin{bmatrix}\alpha^2 & \alpha^4 \\ \alpha^4 & \alpha^2\end{bmatrix}\begin{bmatrix}\alpha^{16} & \alpha^1 \\ \alpha^{21} & \alpha^{10}\end{bmatrix} = \begin{bmatrix}\alpha^{21} & \alpha^4 \\ \alpha^{22} & \alpha^8\end{bmatrix}$$

Step3e: $E5 = Add(E4, sub\_keys(w2, w3))$
Step3f: E6=S (E5)
Step3g: E7=Swapping the 2nd and 4th position ternary field element of E2
Step3h: $CT = E8 = Add(E7, sub\_keys(w4, w5))$
After step3h, the Cipher text obtained is $\{\alpha^{24}\alpha^{23}\alpha^6\alpha^{20}\}$

Algorithm 2(a). Decryption process
Step4: Decryption
Step 4a: $D1 = Subtract(CT, Keys(w4, w5))overF_3m$
Illustration: $D1 = Subtract([\alpha^{24}\alpha^{23}\alpha^6\alpha^{20}], [\alpha^{15}\alpha^9\alpha\alpha^7])over\ F_3m = (\alpha^{10}\alpha^{14}\alpha^{25}\alpha^7)$
Step 4b: D2=Swapping the 2nd and 4th position ternary field element of D1
Illustration $D2 = (\alpha^{10}\alpha^7\alpha^{25}\alpha^{14})$
Step 4c: Inverse S box substitution D3=Inverse Substitution (IS) (D2)
During the encryption process, the S-box is generated randomly. The Inverse S box is computed as soon as the dynamic S-box is randomly generated using the mathematical properties related to F_3m. The Inverse S box IS is shown in Table 3.

Table 3. Inverse S-box using ternary GF

F_3m	(00)3	(01) 3	(02) 3	(10) 3	(11) 3	(12) 3	(20) 3	(21) 3	(22) 3
(0) 3	α^{-inf}	α^{6}	α^{12}	α^{22}	α^{7}	α^{9}	α^{4}	α^{23}	α^{17}
(1) 3	α^{5}	α^{10}	α^{11}	α^{2}	α^{20}	α^{14}	α^{19}	α^{13}	α^{3}
(2) 3	α^{8}	α^{18}	α^{16}	α^{25}	α^{15}	α^{1}	α^{21}	α^{24}	α^{0}

Illustration D3=Inverse S box Substitution=IS (D2)

$$D3 = (\alpha^{10}\alpha^{23}\alpha^{24}\alpha^{14})$$

In this method, S-box is generated dynamically. If the S-box is bijective, only then its inverse exists.
Step 4d: $D4 = Subtract(D3, Keys(w2, w3))overF_3m$
Illustration $D4 = (\alpha^{21}\alpha^{4}\alpha^{22}\alpha^{8}) = \begin{bmatrix}\alpha^{21} & \alpha^{22}\\ \alpha^{4} & \alpha^{8}\end{bmatrix}$
Step 4e: D5= Inverse Mix column;
Matrix Multiplication of the inverse of the random matrix generated during Encryption and the results D4

$$D5 = D4 \times \begin{bmatrix}\alpha & \alpha^{16}\\ \alpha^{16} & \alpha\end{bmatrix} overF_3m$$

$$D5 = \begin{bmatrix}\alpha^{21} & \alpha^{22}\\ \alpha^{4} & \alpha^{8}\end{bmatrix} \times \begin{bmatrix}\alpha & \alpha^{16}\\ \alpha^{16} & \alpha\end{bmatrix} = \begin{bmatrix}\alpha^{16} & \alpha\\ \alpha^{21} & \alpha^{10}\end{bmatrix}$$

$$= [\alpha^{16}, \alpha^{21}, \alpha, \alpha^{10}]$$

Step 4f: D6= Swapping the 2nd and 4th Ternary field digit of D5
Illustration $D6 = [\alpha^{16}, \alpha^{10}, \alpha, \alpha^{21}]$
Step 4g: D7=Inverse S box Substitution= IS (D6)
Illustration $D7 = [\alpha^{13}, \alpha^{10}, \alpha^{6}, \alpha^{25}]$
Step 4h: The recovered Plain text, $D8 = Subtract(D7, Keys(w0, w1))overF_3m$
Illustration: Recovered Plaintext in F_3m: D8= $\{\alpha^{3}, \alpha^{25}, \alpha^{18}, \alpha^{6}\}$
Note: The values considered for illustration purpose is $F_3 3$. Higher values of 'm' are considered for implementation.

3.2. Improved automated asynchronous encryption for data streams (AEDS)

The improved automated AEDS uses ECC over F_3m and the complimentary ternary GF $F_{\bar{3}}m$. The algorithm for encryption and decryption using the ternary GF and the complimentary ternary GF is described in Steps 1 to 3.

a. Step 1 – key generation
- Each user selects a random private key (an integer from the finite field range [1, GF(3^m)], where *m* is the order of the Galois field).
- The public key is computed by multiplying the private key with the curve's generator point P.

b. Step 2 – encryption (sender)
- Message representation: the plaintext is converted into ternary representation and then mapped to a pair of points over GF(3^m).
- Public key lookup: the sender retrieves the recipient's public key from the public directory.
- Random integer selection: the sender chooses a fresh random integer (*ephemeral key*) for each encryption.
- Temporary public key generation: the sender multiplies this random integer with the generator point P to form a temporary public key.
- Shared session key computation: the sender multiplies the random integer with the recipient's public key. This results in a shared secret session key, which both parties can compute independently.
- Field element transformation: the session key values are raised to the order of the field to derive transformation components.
- Cipher text formation: the cipher text is generated using the message components, the session key values, and field operations. The final cipher text consists of the recipient's public key along with the encrypted message pair.
- Transmission: the sender transmits the cipher text and relevant public key components to the recipient.

c. Step 3 – decryption (receiver)

- Session key reconstruction: the receiver computes the session key by multiplying their private key with the sender's transmitted public key. Due to the properties of ECC, this session key matches the sender's computation.
- Field element transformation: the receiver derives the same transformation components from the session key as the sender did during encryption.
- Message recovery: using the cipher text values and the derived session key, the receiver applies inverse field operations to recover the original message components.
- Plaintext conversion: the recovered message components are converted back from GF(3^m) representation into their original ternary and binary form.

The algorithm listed from Steps 1 to 3 is explained with an example. Algorithms 1(b), 2(b) illustrates the encryption and decryption process using the ternary GF, while Algorithms 1(c), 2(c) illustrates the methodology for encrypting and decrypting the data using complimentary ternary GF. All the values considered in this illustration are from GF (3^3), which is listed in Table 1.

Algorithm 1(b). Encryption using F_3m
ECC over finite field $GF(3^3)$
Consider the finite field F_3m, and the generator point $p = \{\alpha^3, \alpha^5\}$ and having the order E (F_3m) = n = 27;
Step 1: -Generation of key
User A carries out the following operation
User A chooses a unique random integer KA = 3 such that $1 \prec K_A \leq n-1$
User A determines the point denoted as $A = KA * P = 3 * (\alpha^3, \alpha^5) = \alpha^{12}, \alpha^{18}$
Step 2: Enciphering process
User B intends to transmit Message to user A. The Message M in Binary and its Ternary equivalent are shown.
Message M = (10100000100) = {(220)3, (020)3)} = (α^5, α^{14})
User B carries out the following operations:

- From the directory of public keys, the User A's public key is fetched. The public key obtained is

  $A = \alpha^{12}, \alpha^{18}$

  The Plaintext M is represented as a pair of points on the x-axis and y-axis$(M1, M2)$
  (M1, M2) = (101000), (00100) in Binary = (220), (020) over$GF(3^m)$
- A random integer denoted as K_B is selected. In this illustration, the value chosen is 2 where in K_B belongs to [1, n-1]
- The public key B is computed using random integer $K_B$and point P. $B = KB * P = 2 * (\alpha^3, \alpha^5) = (\alpha^{18}, \alpha^{24})$
- The shared Key is computed as = $SBA = KB * A = 2 * (\alpha^{12}, \alpha^{18})$

  $(x3, y3) = (\alpha^{20}, \alpha^{22})$

The shared key SBA is treated as the Session Key.

- The field element (x4,y4) is computed using the session key and and raising it to the power of 'm', which is Galois field order.

  $(x4, y4) = ((SBA(x))^m, ((SBA(y))^m$
  $(x4, y4) = ((\alpha^{20})^3, (\alpha^{22})^3)$ over $GF(3^3)$
  $= (\alpha^8, \alpha^{14})$

- B forms the cipher text

  $c1 = ((m1 + x3)(x4)) = \alpha^3$
  $c2 = (m2 + y3)(y4) = \alpha^{21}$

User B transmits the fetched public key of user A along with the computed cipher text to user A;

$(\alpha^{12}, \alpha^{18}, \alpha^3, \alpha^{21})$

Algorithm 2(b). Decryption using F_3m

Step 3: Deciphering is carried out by User A

User A decrypts the cipher $(\alpha^{12}, \alpha^{18}, \alpha^3, \alpha^{21})$ received from the user B by performing the following steps:

User A determines the session key by computing

$SAB = KA * (B) = (2)(\alpha^{18}, \alpha^{24})$

$(x3, y3) = (\alpha^{20}, \alpha^{22})$

A forms $(x4, y4)$ just as B did $x4 = (\alpha^{(20)^3}) = \alpha^5$

$y4 = (\alpha^{(22)^3}) = \alpha^{14}$

A recovers the message (M1, M2) by Computing

$M1 = \frac{C1}{x4} - x3$ overGF(3^m)

$= \frac{\alpha^3}{\alpha^8} - \alpha^{20} = \alpha^5$

$M2 = \frac{C2}{y4} - y3$ overGF(3^m)

$= \frac{\alpha^{21}}{\alpha^{14}} - \alpha^{22} = \alpha^{14}$

(M1, M2) = (α^5, α^{14}) = (220,020) = (101000001000)

ECC over complimentary finite field$GF(\bar{3}^m)$

The finite field considered is $F_{\bar{3}}m$and the generator point $p = \{\alpha^0, \alpha^{19}\}$ having the order E ($F_{\bar{3}}m$)=n=27;

Algorithm 1(c). Encryption using $F_{\bar{3}}m$

Step 1:- Generation of key

User A carries out the following operation

User A chooses a random integer KA which belongs to [1, n-1]. In this illustration, KA= 3

The point 'A' is computed as A=KA $*$ P=$3 * \{\alpha^0, \alpha^{19}\} = (\alpha^5, \alpha^7)$

Step 2: Encryption process

User B carries out the following steps.

Fetches the public key of A from the public key repository: A=(α^5, α^7)

User B intends to transmit Message to User A

Message M= (10100000100)

The message M is represented as a pair of elements $(M1, M2)$

(M1, M2)= (101000), (00100) = (220), (020) over $GF(\bar{3}^m) = (\alpha^{19}, \alpha^8)$ overGF($\bar{3}^m$)

Random integer denoted by K_B ranging from [1, n-1] is chosen. A value of 2 is chosen as an illustration.

Public key is computed as B = KB $*$ P=$2 * (\alpha^0, \alpha^{19})$

$= (\alpha^{17}, \alpha^{17})$

Computes the shared Key =$SBA = KB * A = 2 * (\alpha^5, \alpha^7)$

$(x3, y3) = (\alpha^{14}, \alpha^4)$

The shared key SBA is treated as the Session Key.

The field element (x4, y4) is computed using the session key and and raising it to the power of 'm', which is Galois field order.

$(x4, y4) = ((SBA(x))^m, ((SBA(y))^m$

$(x4, y4) = ((\alpha^{(14)^3}, (\alpha^{(4)}$ over $GF(3^3)$

$= (\alpha^{16}, \alpha^{12})$

B forms the cipher text

$c1 = ((m1 + x3)(x4)) = \alpha^{18}$

$C2 = (m2 + y3)(y4) = \alpha^4$

Entity B transmits the public key of A and the cipher text to A; $(\alpha^5, \alpha^7, \alpha^{18}, \alpha^2)$

Algorithm 2(c). Deciphering using $F_{\bar{3}}m$

Step 3: Deciphering is carried out by User A

User A decrypts the cipher $(\alpha^5, \alpha^7, \alpha^{18}, \alpha^2)$ obtained from user B after performing the following computations:

A performs the following computation to obtain the session

$SAB = KA(B) = (2)(\alpha^{18}, \alpha^{24})$

$(x3, y3) = (\alpha^{14}, \alpha^4)$

A form (x4, y4) just as B did $x4 = (\alpha^{14})^3 = \alpha^{16}$

$y4 = (\alpha^4)^3 = \alpha^{12}$

A recovers the message (M1, M2) using the following equations:

M1=$\frac{C1}{x4} - x3$ overGF($\bar{3}^m$)

$= \frac{\alpha^{18}}{\alpha^{16}} - \alpha^{14} = \alpha^{19}$

M2=$\frac{C2}{y4} - y3$ overGF($3^{\bar{m}}$) =$\frac{\alpha^2}{\alpha^{12}} - \alpha^4 = \alpha^8$

(M1, M2) = (α^{19}, α^8) overGF($\bar{3}^m$) = (220,020) GF($\bar{3}^m$) = (101000001000)

In this approach, point addition over GF($\bar{3}^m$) has been used.

4. RESULTS AND DISCUSSION

Sample results: the results are shown for smaller ternary GF values for simplicity and ease of tabulation. Moreover all the values shown in this section are represented in integer format as per Table 1. However, the proposed algorithm works even for higher GF. The proposed hybrid cryptographic system has been designed, implemented and tested. This section shows the results obtained for various cases. Subsection 4.1 depicts the results obtained for enhanced AES over GF (3^3). Subsection 4.2, shows the results obtained for ECC over GF (3^3) and subsection 4.3 depicts the functional verification of the results obtained for ECC over $GF((\bar{3})^3)$.

4.1. Enhanced AES algorithm with a dynamically generated s-box and inverse s-box, matrix for mix-column over GF (3^3)

Case 1: In this illustration the data considered is 24 samples.
Input data: [25, 15, 4, 1, 3, 19, 2, 21, 23, 12, 17, 22, 14, 13, 6, 10, 8, 9, 20, 16, 7, 11, 24, 5]
Cipher text: [21, 22, 6, 23, 17, 15, 12, 24, 7, 2, 8, 3, 14, 16, 1, 10, 18, 5, 13, 20, 4, 11, 25, 9]
Encryption time: 0.000058s
Recovered plaintext: [25, 15, 4, 1, 3, 19, 2, 21, 23, 12, 17, 22, 14, 13, 6, 10, 8, 9, 20, 16, 7, 11, 24, 5]
Decryption time: 0.000043s

Case 2: Input size: 100352 symbols
Sample input data (first 50 samples is shown):
[4, 23, 10, 10, 9, 24, 23, 2, 22, 8, 12, 16, 11, 1, 24, 18, 22, 21, 16, 24, 21, 19, 11, 20, 4, 11, 7, 23, 5, 0, 17, 15, 14, 22, 2, 10, 23, 18, 21, 8, 13, 22, 20, 2, 18, 12, 7, 25, 4, 23]
Sample cipher text (first 50 samples is shown):
[6, 7, 10, 10, 5, 25, 7, 12, 3, 18, 2, 20, 11, 23, 25, 19, 3, 24, 20, 25, 24, 15, 11, 13, 6, 11, 4, 7, 9, 1, 8, 22, 14, 3, 12, 10, 7, 19, 24, 18, 16, 3, 13, 12, 19, 2, 4, 21, 6, 7]
Sample recovered plaintext (first 50 samples is shown):
[4, 23, 10, 10, 9, 24, 23, 2, 22, 8, 12, 16, 11, 1, 24, 18, 22, 21, 16, 24, 21, 19, 11, 20, 4, 11, 7, 23, 5, 0, 17, 15, 14, 22, 2, 10, 23, 18, 21, 8, 13, 22, 20, 2, 18, 12, 7, 25, 4, 23]
Encryption time: 0.400 ms
Decryption time: 0.401 ms

4.2. Enhanced elliptic curve cryptography algorithm over Galois field (3^m)

In this illustration, the message M=(m1,m2) samples, consists of random values defined over GF(3^3).
m1 = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24]
m2 = [25 15 4 1 3 19 2 21 23 12 17 22 14 13 6 10 8 9 20 16 7 11 24 5]
The cipher text is obtained as C=(c1,c2)
c1 =[2 26 21 9 6 4 19 16 25 13 1 8 23 11 15 3 18 -Inf 24 12 10 7 22 20]
c2 =[23 4 9 5 22 -Inf 19 20 3 24 17 16 1 14 2 7 12
11 26 18 25 21 6 8]
The cipher text C is decrypted as the recovered Message MR=(mr1,mr2)
mr1 =[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24]

mr2 =[25 15 4 1 3 19 2 21 23 12 17 22 14 13 6 10 8 9 20 16 7 11 24 5]
For this illustration, the key derivation time is 1.173019 ms
Time taken to encrypt is 1.508474 ms and decryption time is 9.611368 ms
The recovered message is same as the original message.

4.3. Random data values over GF (3^6) are encrypted using the proposed algorithm

The encryption time, decryption time, entropy obtained after conversion from trits to bits and the avalanche effect are tabulated for 128, 256, and 512 message symbols as shown in Figure 5. Entropy quantifies the unpredictability of the output, while avalanche effect indicates the changes in the output for small changes in the input.

```
GF(3^6) simulation starting
Selected irreducible polynomial (lowest-first coefficients): [2, 1, 0, 0, 0, 0, 1]

Results:
 message_size  enc_time_s  dec_time_s  entropy_bits_per_symbol  avalanche_%_single_symbol_change
          128    0.147496    0.036449                 6.863205                          0.781250
          256    0.096151    0.061541                 7.597580                          0.390625
          512    0.272378    0.337127                 8.370865                          0.195312
```

Figure 5. Results of encryption using higher order GF

Having verified the algorithm for its correctness, it becomes essential to determine the strength of the algorithm. Hence randomness test is performed on the algorithm. The algorithm is functional verified for successful encryption and decryption.

4.4. Randomness test

Cryptographic security often relies on generating secret keys, nonces, or unexpected random numbers. If these numbers are not truly random or show observable patterns, security may be jeopardized if attackers can predict or duplicate them. Bytes like keys and cipher texts are tested for randomness to see if they behave like random data. Biases or patterns could indicate inadequate random number generators or bad implementation. Randomization is the foundation of several techniques, such as encryption, key exchange, and signature schemes. Randomness analysis confirms that these internal random values seem to be the same as random noise.

Case 1: randomness test for enhanced-AES over GF (3^m)

Shannon entropy, Chi-squared, frequency test, and runs test is conducted on the enhanced AES-algorithm over GF (3^3). The test was conducted on sample size of 98,034 cipher text symbols, which approximately accounts to 3, 93,216 trits.

Shannon entropy [30]: Shannon entropy provides a measure of unpredictability of information content. According to the survey, maximum possible entropy is 8 bits per byte. Mathematically, it is computed using:

$$H = -\sum_{i=0}^{255} p_i \, log_2(p_i)$$

The Shannon entropy obtained is 4.700270 bits/symbol which are very close to ideal. From this, it can be inferred that the sequences considered for testing produces symbols with a distribution nearer to the uniform randomness.

Chi-squared vs uniform test:

Given observed frequencies O_i for each outcome and expected frequency $E_i$under the null hypothesis of uniformity, the chi-squared statistics is:

$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$, with n-1 degrees of freedom.

Also the expected frequency $E_i = \frac{N}{n}$, where N is the number of samples. ‘n’ is the alphabet size.

With N=98,034 samples and n=26, E_i=3781. The computed χ^2=23.116. Critical value at 95% confidence is approximately equal to 37.65. Since 23.116 < 37.65, cipher text distribution is consistent with uniform random symbols.

Runs test: this test checks for the number of transitions between ones and zeros. For an n-bit binary data, if there are 'n1' ones and 'n0' zeros, then the expected number of runs is given by $\mu = \frac{2n1n0}{n} + 1$

The runs test revealed a count of 24, for a sequence of the same length. From the runs test, it can be inferred that every adjacent pair was not the same. There were no two equal adjacent symbols.

This shows that, if the data pattern and the cipher text cannot be predicted, the brute force attack, a plaintext-cipher text pair attack, is not possible. The security offered is enhanced.

Case 2: randomness test for ECC over GF (3^m)

Randomness tests for partial cipher text c1:

Shannon entropy (bits/symbol): 4.584963 Chi-square vs uniform: chi2=3.000 (df=26) p-value: None Runs test (bits): runs=67, expected=60.183, z=1.267, p=0.205082

Randomness tests for partial cipher text c2:

Shannon entropy (bits/symbol): 4.584963 Chi-square vs uniform: chi2=3.000 (df=26) p-value: None Runs test (bits): runs=60, expected=59.650, z=0.066, p=0.947648

From the results obtained, it can be inferred that the Shannon's entropy test indicates a strong unpredictability of cipher text. Chi-square test suggests there is no-bias in symbol frequencies. The runs test indicates independence among symbols.

Additionally randomness is tested using NIST [31] statistical suite. The NIST suite is designed to test the statistically random nature of the cryptographic ciphers and key generators. There are 15 tests such as frequency test, block frequency test, approximate entropy test, and so on which are performed to verify the randomness. In this approach, the Ternary sequence is converted to binary for testing the randomness using NIST suite. From the output obtained, it can be inferred that 90% of the tests performed using NIST suite have obtained p-value, $p \geq 0.01$, which indicates the randomness of the sequence.

Having verified the correctness of the algorithm and also performing randomness test on the same, it is important to compute the time complexity required for encrypting and decrypting ternary data. Representing the data in ternary format, offers the major advantage of storing more volumes of data, compared to its binary counterpart.

Performance metrics: the proposed encryption methods that work with ternary data are contrasted with those that use binary data. Different sizes of ternary data are used to implement and test the experiments. Each file is broken up into blocks that consist of 128 MB. The second block of 128 MB is divided into smaller data streams and encrypted using enhanced ECC over ternary fields, while the first block of 128 MB is encrypted using the improved AES technique over ternary fields. The following statistical results are determined for all the algorithms used in the comparison to assess the encryption algorithms: the encryption and decryption of files and the throughput determine after the encryption and decryption files.

Simulation time taken for encryption/decryption of data: the total amount of time taken by the system to perform encryption or decryption of all the data blocks of a file that is being selected using a particular algorithm from the commencement of encryption or decryption process till its conclusion.

$$T_S = \sum((End\_time) - (start\_time)), perblock$$

Processing time (PT): it is the total simulation time taken by the system for random key generation and encrypting or decrypting process. It is computed using the equation.

$$PT = keygeneration\_time + T_S$$

Throughput: it is defined as the amount of data that can be processed in a unit time successfully. It is computed using the equation:

$$\text{Throughput (MB/s)} = \text{Size of the selected file (in MB)} /= \text{T (in seconds)}$$

The proposed technique employs unbalanced ternary digits, denoted as 0, 1, and 2. Compared to binary data, ternary data could fit more volumes of data in the same available storage. The comparison table shows how well the modified AES algorithm, enhanced ECC algorithm, and hybrid cryptosystem using binary field and ternary field perform when encrypting and decrypting various files of varying sizes. The files are divided into two blocks of equal size. These equal-sized blocks are encrypted concurrently using the

proposed hybrid cryptographic algorithm. In addition, the ternary field-based hybrid cryptosystem eliminates the third-party problem and achieves high security. The results are tabulated after implementing modified AES (using binary and ternary GF), modified ECC (using binary, ternary, and complimentary ternary GF) and the proposed hybrid algorithm over the Ternary GF. The tabulation displays the total amount of time used to encrypt and decrypt data blocks of various sizes, including the key generation time. The time taken for generating a random S-box, its inverse, random matrix and its inverse are also considered as a part of the key generation time.

According to Table 4 and Figure 6, it can be observed that the overall time taken for random key generation, random S-box generation, random matrix generation and encryption using the proposed hybrid cryptographic algorithm is lesser when compared to the other tested algorithms. Also all the computations using ternary modular GF consumes lesser time compared to binary modular GF. As per the results tabulated and Figure 7, it can be observed that the overall time taken for decryption using the proposed technique is lesser when compared to the other tested algorithms. This overall time includes the inverse S-box computation and inverse matrix generation using ternary GF.

Table 4. Comparison between the proposed algorithm and the other tested algorithms

File size	Overall encryption time					Overall decryption time				
	AES (binary)	ECC	AES	ECC (ternary)	Proposed crypto system	AES (binary)	ECC	AES	ECC (ternary)	Proposed crypto system
98 KB	2.8703	2.619	2.5394	2.2739	1.8381	28.1516	39.8512	26.6206	39.0814	18.3224
196 KB	6.2803	5.7195	5.2281	4.0962	2.6087	56.7898	78.9381	53.5651	72.8808	28.1987
368 KB	11.9245	10.8263	9.8533	8.1337	5.1605	85.5689	110.4939	74.9554	102.0143	56.3405
736 KB	14.3076	12.9821	11.8706	9.7455	8.2016	103.0231	132.5833	89.9287	122.8972	67.8713
3 MB	48.9596	40.2215	41.7146	33.5358	29.9334	488.1295	590.822	456.7043	578.3877	478.1881
6 MB	57.1572	48.2564	51.2228	40.2315	35.9087	586.2389	709.8041	548.7122	694.0559	574.5189
128 MB	191.2465	58.7076	167.2428	50.3156	85.8111	910.9861	972.6982	875.4947	901.2891	859.5789
256 MB	301.649	180.1892	267.535	100.5743	170.2163	1017.15	1120.937	954.3208	1081.537	903.5737

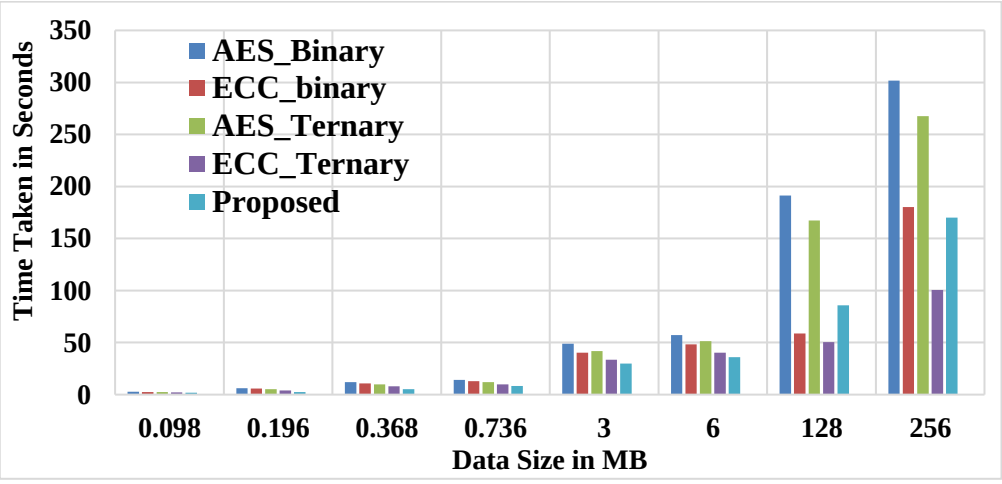

Figure 6. Overall encryption time taken by the various tested algorithms and the proposed algorithm

Figures 6 and 7 shows the time taken to encrypt and decrypt data of various sizes using the tested algorithms such as modified AES (using binary and ternary GF), modified ECC (using binary, ternary, and complimentary ternary GF) and the proposed hybrid algorithm over ternary GF. It is observed that the proposed algorithm has a lesser encryption and decryption time as the data is processed concurrently. From

Table 4, it can be observed that the proposed hybrid encryption/decryption technique executes at a faster rate, compared to the enhanced AES technique over binary and ternary GF as well as the enhanced ECC technique over binary and ternary GF. The reason is due to the fact, that data is divided into blocks and each block is concurrently executed by the symmetric key based enhanced AES and asymmetric key based enhanced ECC algorithms. Further it can be observed that the representation of data in ternary and computations in ternary enhances the speed of execution compared to data in binary. In contrast, the ternary data-based hybrid cryptosystem took less time to process big file sizes. These methods outperformed the competition in terms of time savings despite having more difficult operations during the decryption process and taking longer than during encryption. They guarantee improved security. Additionally, the hybrid cryptosystem using ternary field and the upgraded AES encryption/decryption process both use dynamic S boxes and dynamic matrices for each every data block, increasing security.

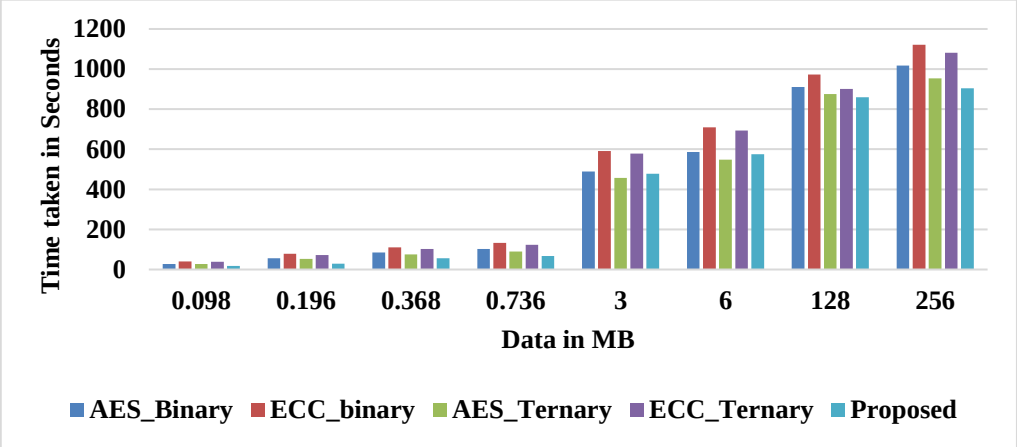

Figure 7. Overall decryption time taken by the various tested algorithm and the proposed algorithm

The proposed hybrid approach may benefit from faster encryption and decryption times, when compared to the existing algorithms. Security is enhanced as data is segmented into blocks and streams. Blocks of data are encrypted using enhanced AES and data streams are encrypted alternatively using ECC over GF (3m) and ECC$GF(\overline{3})^{m}$. The algorithms are functionally verified for the correctness of the results. Randomness test reveals random cipher text and keys in sample space. Insider attack in the cloud environment becomes infeasible, as 50% of the data is encrypted using ECC over ternary GF and complimentary ternary GF, as the algorithm is based on the curve parameters.

As an illustration, the proposed cryptographic system is integrated with the cloud environment for seamless transferring of secured ECG data. At the API gateway, requests are validated and the backend services, securely retrieves the key for decryption. For the purpose of analysis, as shown in Figure 8, randomized data containing heart rate, RR interval, and QRS duration has been used. This data has been encrypted using conventional benchmarked algorithms such as AES, ECC and also the proposed algorithm for analysis. According to the results of Table 5, the proposed algorithm is competitive with ECC showing slightly better encryption efficiency in terms of energy consumption and speed.

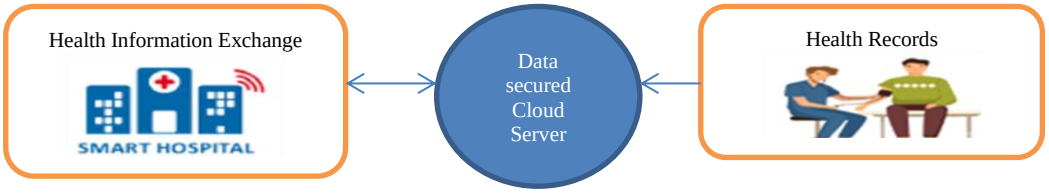

Figure 8. Smart hospital with secured cloud server

Table 5. Comparison between the proposed algorithm and the other tested algorithms for health care data

Algorithm	Encryption time(s)			Energy spent (mJ)			Decryption time(s)		
Nodes	N1	N2	N3	N1	N2	N3	N1	N2	N3
Proposed algorithm	0.000119	0.000216	0.000086	0.0595	0.1078	0.0429	0.000053	0.000092	0.000044
ECC over GF (3^m)	0.000095	0.000094	0.000083	0.0476	0.0470	0.0414	0.000052	0.000058	0.000048
AES over GF (2^m)	0.006364	0.000154	0.000171	3.1819	0.0772	0.0857	0.000133	0.000095	0.000086

5. CONCLUSION

Since multiple users share the same infrastructure, data isolation becomes very crucial in cloud. As the volume of data increases, users are now storing their data on distant servers. Outsourced data is vulnerable to illegal, immoral use and is no longer in the user's control. For this suggested approach, we have developed an automated hybrid cryptography system to guarantee data security and reduce the effort required by users to safeguard their own data. This system functions independently of cloud third parties. In this method, a hybrid cryptographic system is used. ternary field data, as opposed to conventional binary data, makes the hybrid cryptographic system more secure. More data can be kept in the allotted area since ternary is used to represent the data. By employing concurrent processing, data is divided into equal-sized blocks & streams. Data blocks are encrypted using enhanced AES over ternary GF and data streams are encrypted using enhanced ECC over ternary GF in order to decrease PT overall. The hybrid cryptographic system also employs a random dynamic S-box and a dynamic matrix for the substitution and mix column approaches for every data block in order to boost security. The proposed algorithm is tested for its functional correctness for datasets, of size 96 KB to 256 MB. Encryption and decryption produced the exact recovery of original data. The randomness test indicates the absence of any predictable patterns, and it provides resistance to statistical cryptanalysis. The algorithm offers resistance to insider attacks, as the ECC stream protects half of the data, even if one scheme is compromised, such as if the block cipher is cracked. Furthermore, the session keys are not stored. They are produced using elliptic curve-like mathematics, which restricts exposure. The attacker would have to access ephemeral session secrets in addition to knowledge of the hybrid scheme, which makes this approach complex.

The proposed algorithm provides functional correctness, strong randomness, reduced time complexity and improved resistance to insider attacks. Hence the proposed algorithm is suitable for cloud environment. Each node ensures that every shared data is independently encrypted. New storages can be added seamlessly without compromising on the existing infrastructure.

Standard APIs can be used for seamless deployment by integrating the cryptographic algorithms for automatically encrypting the data before upload and decrypting after authorized retrieval. The proposed hybrid encryption system can be integrated into the DevSecOps pipeline at the build stage, ensuring that all data handling applications are automatically encrypted and remains protected throughout deployment and monitoring in the cloud environment.

As a future scope for improvement, multi-round dynamic S-box can be implemented, dynamic partitioning of data to be implemented, and this information security method to be applied to any embedded platform as a potential area for further development.

ACKNOWLEDGMENTS

We, the authors of this article, sincerely thank the Management of Dayananda Sagar College of Engineering, for facilitating with the infrastructure required to successfully implement this research work. This work is not supported by any research grant or contract.

FUNDING INFORMATION

This work is not funded by any funding agency.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Srividya Bharadwaja VenkataSubbu	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓	
Smitha Sasi	✓		✓	✓	✓		✓			✓			✓	
Meharunnisa Sakkar Sab Pakeer Saheb	✓				✓		✓			✓		✓		
Harikeerthan Mysore Keshava Rao		✓			✓			✓		✓	✓	✓		
Soumya Seetharamaiah		✓				✓			✓	✓	✓			

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The authors confirm that the data supporting the findings of this study are available within the article.

REFERENCES

[1] N. E. El-Attar, D. S. El-Morshedy, and W. A. Awad, "A New Hybrid Automated Security Framework to Cloud Storage System," *Cryptography*, vol. 5, no. 4, pp. 1–20, 2021, doi: 10.3390/cryptography5040037.

[2] D. R, T. K, and K. S, "A Decentralized Accountability Framework For Enhancing Secure Data Sharing Through Icm In Cloud," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 10, pp. 1-7, 2019, doi: 10.35940/ijitee.J1026.0881019.

[3] S. Shahzadi, M. Iqbal, Z. U. Qayyum, and T. Dagiuklas, "Infrastructure as a service (IaaS): A comparative performance analysis of open-source cloud platforms," in *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD*, 2017, pp. 19–21, doi: 10.1109/CAMAD.2017.8031522.

[4] R. Jathanna and D. Jagli, "Cloud Computing and Security Issues," *International Journal of Engineering Research and Applications*, vol. 07, no. 06, pp. 31–38, 2017, doi: 10.9790/9622-0706053138.

[5] G. Kulkarni, R. Waghmare, R. Palwe, V. Waykule, H. Bankar, and K. Koli, "Cloud storage architecture," in *2012 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, 2022, pp. 76–81.

[6] N. Vurukonda and B. T. Rao, "A Study on Data Storage Security Issues in Cloud Computing," *Procedia Computer Science*, vol. 92, pp. 128–135, 2016, doi: 10.1016/j.procs.2016.07.335.

[7] A. Bentajer, M. Hedabou, K. Abouelmehdi, and S. Elfezazi, "CS-IBE: a data confidentiality system in public cloud storage system," *Procedia Computer Science*, vol. 141, pp. 559–564, 2018, doi: 0.1016/j.procs.2018.10.126.

[8] L. A. Tawalbeh and G. Saldamli, "Reconsidering big data security and privacy in cloud and mobile cloud systems," *Journal of King Saud University - Computer and Information Sciences*, vol. 33, no. 7, pp. 810–819, 2021, doi: 10.1016/j.jksuci.2019.05.007.

[9] D. Das, "Secure cloud computing algorithm using homomorphic encryption and multi-party computation," in *International Conference on Information Networking*, pp. 391–396, 2018, doi: 10.1109/ICOIN.2018.8343147.

[10] A. Mohta and L. K. Awasti, "Cloud data security while using third party auditor," *International Journal of Scientific & Engineering Research*, vol. 3, pp. 1–9, 2012.

[11] Z. M. Yusop and J. H. Abawajy, "Analysis of Insiders Attack Mitigation Strategies," *Procedia - Social and Behavioral Sciences*, vol. 129, pp. 611–618, 2014, doi: 10.1016/j.sbspro.2014.06.002.

[12] S. Singh and S. Thokchom, "Public integrity auditing for shared dynamic cloud data," *Procedia Computer Science*, vol. 125, pp. 698–708, 2018, doi: 10.1016/j.procs.2017.12.090.

[13] M. M. Potey, C. A. Dhote, and D. H. Sharma, "Homomorphic Encryption for Security of Cloud Data," *Procedia Computer Science*, vol. 79, pp. 175–181, 2016, doi: 10.1016/j.procs.2016.03.023.

[14] N. E. El-Attar, W. A. Awad, and F. A. Omara, "Empirical assessment for security risk and availability in public Cloud frameworks," in *Proceedings of 2016 11th International Conference on Computer Engineering and Systems, ICCES 2016*, 2017, pp. 17–25, doi: 10.1109/ICCES.2016.7821969.

[15] S. Chakraborty, S. Singh, and S. Thokchom, "Integrity Checking Using Third Party Auditor in Cloud Storage," in *2018 Eleventh International Conference on Contemporary Computing (IC3)*, 2018, pp. 1–6, doi: 10.1109/IC3.2018.8530649.

[16] S. More and S. Chaudhari, "Third Party Public Auditing Scheme for Cloud Storage," *Procedia Computer Science*, vol. 79, pp. 69–76, 2016, doi: 10.1016/j.procs.2016.03.010.

[17] K. M. Akhil, M. P. Kumar, and B. R. Pushpa, "Enhanced cloud data security using AES algorithm," in *2017 International Conference on Intelligent Computing and Control (I2C2)*, 2018, pp. 1–5, doi: 10.1109/I2C2.2017.8321820.

[18] P. Sivakumar, M. Nandhakumar, R. Jayaraj, and A. S. Kumaran, "Securing data and reducing the time traffic using AES encryption with dual cloud," in *2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)*, 2019, pp. 1–5, doi: 10.1109/ICSCAN.2019.8878749.
[19] A. Orobosade, T. Aderonke, A. Boniface, and A. J., "Cloud Application Security using Hybrid Encryption," *Communications on Applied Electronics*, vol. 7, no. 33, pp. 25–31, 2020, doi: 10.5120/cae2020652866.
[20] O. Hosam and M. H. Ahmad, "Hybrid design for cloud data security using combination of AES, ECC and LSB steganography," *International Journal of Computational Science and Engineering*, vol. 19, no. 2, pp. 153–161, 2019, doi: 10.1504/IJCSE.2019.100236.
[21] S. Gadde, J. Amutharaj, and S. Usha, "A Hybrid Cryptography Technique for Cloud Data Security," *International Journal of Engineering Trends and Technology*, vol. 70, no. 11, pp. 258–267, 2022, doi: 10.14445/22315381/IJETT-V70I11P228.
[22] K. Sasikumar and S. Nagarajan, "Comprehensive Review and Analysis of Cryptography Techniques in Cloud Computing," *IEEE Access*, vol. 12, pp. 52325-52351, 2024, doi: 10.1109/ACCESS.2024.3385449.
[23] N. N. Ahamed and N. Duraipandian, "Secured data storage using deduplication in cloud computing based on elliptic curve cryptography," *Computer Systems Science and Engineering*, vol. 41, no. 1, pp. 83–94, 2022, doi: 10.32604/csse.2022.020071.
[24] P. Shrivastava, B. Alam, and M. Alam, "Security enhancement using blockchain based modified infinite chaotic elliptic cryptography in cloud," *Cluster Computing*, vol. 26, no. 6, pp. 3673–3688, 2023, doi: 10.1007/s10586-022-03777-y.
[25] A. Malik, M. Aggarwal, B. Sharma, A. Singh, and K. K. Singh, "Optimal Elliptic Curve Cryptography-Based Effective Approach for Secure Data Storage in Clouds," *International Journal of Knowledge and Systems Science*, vol. 11, no. 4, pp. 65–81, 2020, doi: 10.4018/IJKSS.2020100105.
[26] P. Suthanthiramani, M. Sannasy, G. Sannasi, and K. Arputharaj, "Secured data storage and retrieval using elliptic curve cryptography in cloud," *International Arab Journal of Information Technology*, vol. 18, no. 1, pp. 56–66, 2021, doi: 10.34028/iajit/18/1/7.
[27] F. Thabit, S. Alhomdy, and S. Jagtap, "A new data security algorithm for the cloud computing based on genetics techniques and logical-mathematical functions," *International Journal of Intelligent Networks*, vol. 2, pp. 18–33, 2021, doi: 10.1016/j.ijin.2021.03.001.
[28] U. R. Saxena and T. Alam, "Role-based access using partial homomorphic encryption for securing cloud data," *International Journal of System Assurance Engineering and Management*, vol. 14, no. 3, pp. 950–966, 2023, doi: 10.1007/s13198-023-01896-2.
[29] M. Gusset, "Defense in Depth in a Hybrid Cloud," Master of Philosophy, Capitol Technology University, Maryland, 2025.
[30] Y. Wu, J. P. Noonan, and S. Agaian, "Shannon Entropy based Randomness Measurement and Test for Image Encryption," *arXiv preprint*, 2011, doi: 10.1016/j.ins.2012.07.049.
[31] Z. Mengdi, Z. Xiaojuan, Z. Yayun, and M. Siwei, "Overview of Randomness Test on Cryptographic Algorithms," in *The 5th International Workshop on Advanced Algorithms and Control Engineering (IWAACE 2021)*, vol. 1861, no. 1, 2021, doi: 10.1088/1742-6596/1861/1/012009.

BIOGRAPHIES OF AUTHORS

Srividya Bharadwaja VenkataSubbu received the Engineering degree in Electronics and Communication Engineering from PES Institute of Technology, Bangalore, Karnataka, India in the year 2000. She received the Master degree in VLSI design and Embedded system from Visvesvaraya Technological University in 2009. She was awarded Ph.D. for defending her thesis on "Implementation of Cryptography and Error correction codes in wireless sensor networks" from Visvesvaraya Technological University in 2018. Currently she is working as an Associate Professor in the Department of Electronics and Telecommunication Engineering, Dayananda Sagar College of Engineering, Bangalore, Karnataka, India. Her research interest includes VLSI design, cryptography, error control coding, cyber security, and machine learning. She has a patent granted to her credit and has authored/co-authored around 70 technical research papers. She can be contacted at email: srividyabv@gmail.com.

Smitha Sasi is working as an Associate Professor in the Department of Electronics and Telecommunication Engineering (ETE), Dayananda Sagar College of Engineering (DSCE). She has about 19 years of teaching experience. She received her B.Tech. degree in Information Technology from Calicut University and master's degree in Digital Communication and Networking from Visvesvaraya Technological University. She received her doctoral degree in cryptography from Visvesvaraya Technological University in 2018. She has published many research papers in referred journals and in the proceedings of various international conferences. She has received research funds from various funding agencies in cryptography and network security domain. She is in receipt of several patents, through her efforts of setting up Space Technology Cell in the Department of Electronics and Telecommunication Engineering, DSCE in collaboration with ISRO. She has authored various book chapters in the domain of cryptography. She can be contacted at email: smitha-tce@dayanandasagar.edu.

Meharunnisa Sakkar Sab Pakeer Saheb graduated from Siddaganga Institute of Technology, Tumkur in the year 1999 and later pursued M.E. from University of Visvesvaraya College of Engineering (UVCE), Bengaluru and Ph.D. from Visvesvaraya Technological University (VTU), Belagavi. She has 14 research publications; 3 patents and 56 related papers published in international and national journals and conferences. She has received a research grant Rs 10 Lakhs for modernization of Microcontroller laboratory to Embedded Laboratory with IoT & Multimedia applications from AICTE, New Delhi. Her special interests are in the domains of instrumentation techniques, analog circuit design, embedded systems, AI, IoT, IIoT, signal, and medical image processing. She has 16 years of passionate experience in teaching and research and 6 years in industry, worked in Honeywell Technology Solutions Lab and Bharath Electronics Ltd (BEL), Bengaluru and is currently serving Dayananda Sagar College of Engineering, Bengaluru, as an Associate Professor in the Department of Electronics and Instrumentation Engineering. She can be contacted at email: meharunnisa@dayanandasagar.edu.

Harikeerthan Mysore Keshava Rao received the Engineer degree in Civil Engineering from BMS College of Engineering of Bangalore University, Bangalore in 1999 and the Master of Technology in Highway Technology from the BMS College of Engineering of Visvesvaraya Technological University, Belagavi, in 2004. Has obtained Ph.D. in Highway Engineering under Visvesvaraya Technological University in 2019. Currently he is an Associate Professor in the Department of Civil Engineering, Dayananda Sagar Academy of Technology and Management, An Autonomous Institute under Visvesvaraya Technological University, Bangalore, India. His research interests include pavement deterioration modelling, intelligent transport systems, prediction modelling, urban planning, artificial intelligence, renewable energy, and vehicle sensors. He can be contacted at email: harikeerthan@dsatm.edu.in.

Ms. Soumya Seetharamaiah received her B.E. degree in Telecommunication Engineering from CMR Institute of Technology, VTU, and a master's degree in Digital Electronics and Communication from the Dayananda Sagar College of Engineering. She is currently a faculty member in the Department of Electronics and Communication Engineering in Manipal Institute of Technology, Manipal, Manipal Academy of Higher Education. Her research interests include communication networks, cryptography, and embedded systems. She can be contacted at email: soumya.s@manipal.edu.