Journal of Theoretical and Applied Information Technology
15th August 2024. Vol.102. No. 15
© Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

IMPLEMENTATION AND MANAGEMENT OF SECURITY FOR SENSITIVE DATA IN CLOUD COMPUTING ENVIRONMENT USING ELLIPTICAL CURVE CRYPTOGRAPHY

N. KRISHNAMOORTHY[1] , S.UMARANI[2]

[1]Department of Computer Science and Applications (MCA), Faculty of Science and Humanities, SRM Institute of Science and Technology, Ramapuram, Chennai- 600 089, TamilNadu, India

[2]Department of Computer Science and Applications (BCA), Faculty of Science and Humanities, SRM Institute of Science and Technology, Ramapuram, Chennai- 600 089, TamilNadu, India

E-mail: [1] krishnan@srmist.edu.in, krishnamoorthy72@gmail.com, [2] umaranis@srmist.edu.in

ABSTRACT

When it comes to cloud storage, one of the biggest concerns is keeping data safe. In today's technologically evolved world, cloud attacks are on the rise. Current cloud storage security services rely heavily on symmetric key encryption algorithms, which involve the exchange of secret keys and may be susceptible to attacks from outside parties. In the same way that cloud computing has become indispensable in recent years, so too have the security issues surrounding the cloud model grown in scope and complexity. There has been a dramatic shift in how infrastructure, service delivery, and development models are seen thanks to cloud computing. Each participant in the proposed cloud storage security framework has a specific role to play: the data owner, who is responsible for encrypting the plain text and constructing the access control policy; the attribute authority, who acts as the data owner's trusted agent and stores the attribute-based access control policy used for key generation and to restrict access to authorized users; the cloud storage; and the data users. In Elliptic Curve Cryptography (ECC), improving encoding schemes is a top priority. Despite the obvious advantages of the cloud model, it will have a hard time gaining widespread client acceptance unless issues of privacy and security are resolved. Concerns about the privacy, authenticity, and integrity of cloud-stored data are the focus of this thesis, along with recommendations for implementing those safeguards. In this thesis, we design and develop secure and efficient protocols based on Elliptic Curve Cryptography for multilevel security in a cloud environment, with the goals of protecting the privacy of users' data, ensuring that only authorized users can access their information, and guaranteeing that all data is genuine and unaltered. When it comes to encryption, computational load, and security during cloud data storage, retrieval, and access, the proposed security frameworks have been implemented and compared to existing models. This research work aims at introducing four different novel algorithms and finds out the minimized encoding and decoding durations and thereby reducing uploading and downloading durations.

Keywords: *Elliptic Curve Cryptography, Data Security, Cloud deployment, Private cloud, Public cloud*

1. INTRODUCTION

Data storage and processing can be optimized through the usage of cloud computing. Cloud computing altered the industry's previous service delivery mechanisms, development methodologies, and underlying infrastructure [1]. From simple data storage to advanced online banking, predictive modeling, and data analytics, the applications of cloud computing are vast. There are three main categories of cloud services: public, private, and hybrid [2]. Private cloud storage is more secure due to additional safeguards, while public cloud information is accessible to anybody and could potentially be a privacy risk. The hybrid cloud combines elements of both the public and private cloud [3]. There could be a lot of trouble with the security paradigm for data transfer across cloud architecture.

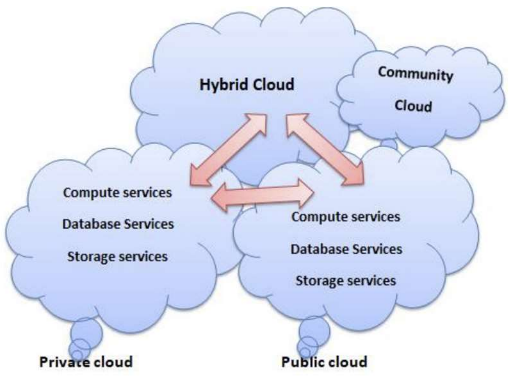

Fig 1: Process of Cloud Deployment Model

This same security problem in a hybrid cloud setting was addressed in the proposed study [4]. When it comes to cloud computing, especially open cloud implementation, security is still the primary worry for most organizations. Since the open cloud is multi-tenant, the cloud provider's secure hardware infrastructure is shared by multiple customers. There is an excessive demand for isolation between consistent process assets, which this scenario necessitates [5-9]. Login credentials are used to restrict who can access open, distributed storage and data. Threats to cloud-based data resources might vary widely across organizations and types of cloud service delivery models. Private clouds, public clouds, and hybrid clouds all exist. The private cloud management is communicated to the client via the company server. Deployment strategies provide cloud flexibility and services while isolating and minimizing disruptions to on-premises IT infrastructure. The cloud services are provided by external cloud providers and are made available online [10-14]. The majority of public cloud services are sold on a per-use or per-hour basis, but some also provide long-term commitments. Customers pay for the services they use, which may include CPU time or data transmission capability. The public cloud includes services from major players like Amazon Web Services (AWS), IBM, Microsoft Azure, and Google Cloud Platform. Hybrid clouds combine public cloud services with private clouds hosted on-premises, with the help of planning and automation. When an enterprise runs its workloads and applications in a private cloud, it can take advantage of the public cloud to handle sudden spikes in burden.

2. BACKGROUND

The success of any business with an online presence depends on the reliability of its web apps. Due to the global nature of the Internet, web domains can be subject to attacks of variable scale and complexity. The term "web application security" is used to describe the precautions taken to stop hackers from exploiting vulnerabilities in a website or service [15-18]. Security measures such as up-to-date encryption, robust authentication, and continuous patching of newly discovered vulnerabilities are essential for keeping cloud-based web applications safe from incursion. Therefore, it is necessary to design and implement contemporary cryptographic primitives that make use of current protocols to protect users' privacy, authenticity, and the integrity of their communications. Our research group focuses on the mathematical improvement of PKC primitives based on Elliptic Curve Cryptography (ECC). An elliptic curve is defined by the following equation of the form.

$$Y^2=x^3+Ax+B \qquad (1)$$

Points on the elliptic curve are represented by x and y in this equation, whereas the values of a and b are the shape coefficients. For all points on the curve, the left side of the equation (y2) is equal to the right side (x3 + ax + b), describing the relationship between the x and y coordinates [19-22]. Parameters a and b are commonly used to guarantee the curve is non-singular, has no singularities, and does not self-intersect, among other mathematical criteria.

In ECC, the polynomial equation takes several forms, and the Weierstrass equation is simply one of them. Elliptic curves can also be defined in various forms, such as the Edwards, Montgomery, and Hessian forms, which use different polynomial equations. Using mathematical techniques that work on the coordinates of the points, ECC performs cryptographic operations on these elliptic curves, such as adding and doubling points and performing scalar multiplication [23-24]. Elliptic curve cryptography (ECC) employs a variety of coefficients and curve parameters that vary depending on the elliptic curve and cryptographic implementation. For the ECC scheme's safety and performance, the elliptic curve and its parameters are critical. The primary need for such a curve is that it be discriminating.

$$\Delta=4A^3+27B^2\neq0 \qquad (2)$$

The Edward Curve transformation allows ECC to be calculated more quickly. The Edwards curve form is equivalent to the binary field in an elliptical curve on field extension and in the original field. This

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

results in the following division of the number field along an ellipse:

$$x^2+y^2=c^2\,(1+x^2\,y^2) \qquad (3)$$

where, (0, c) is selected as a neutral element using symmetric addition law.

$$(x1,y1)(x2,y2) = \frac{(x1y2+y1x2)}{c(1+x1\,x2\,y1\,y2)},\ \frac{(y1y2+x1x2)}{c(1+x1\,x2\,y1\,y2)} \qquad (4)$$

Similar to how a whole elliptical curve can be converted into Edwards form with the use of some hard-and-fast formulas,

- Addition (X1:Y1:Z1)(X2:Y2:Z2) = (X1:Y1:Z1) + (X2:Y2:Z2) through 1S +10M
- Doubling (X1:Y1:Z1) = 2(X1:Y1:Z1) through 4S + 6M.

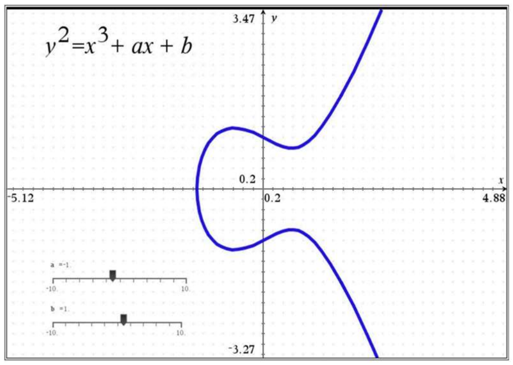

Fig 2: Plot for the elliptic Curve

The group structure specified over a finite field that elliptic curves give is used to create cryptographic protocols. The elliptic curves for eqn 1, with a=-1 and b=1, are graphically represented in fig 2. As can be seen in the accompanying graph, the x-axis symmetry of the curve is crucial to the ECC calculation. The graphic provides visual representations of several types of elliptic curves. For instance, with a prime Galois field of GF(5), the number of pints over the curve is 9 if a=1 and b=1. Elliptic curve points of rank 9 are as follows: (0,1), (0,4), (2,1), (2,4), (3,1), (3,4), (4,3), (4,2),.

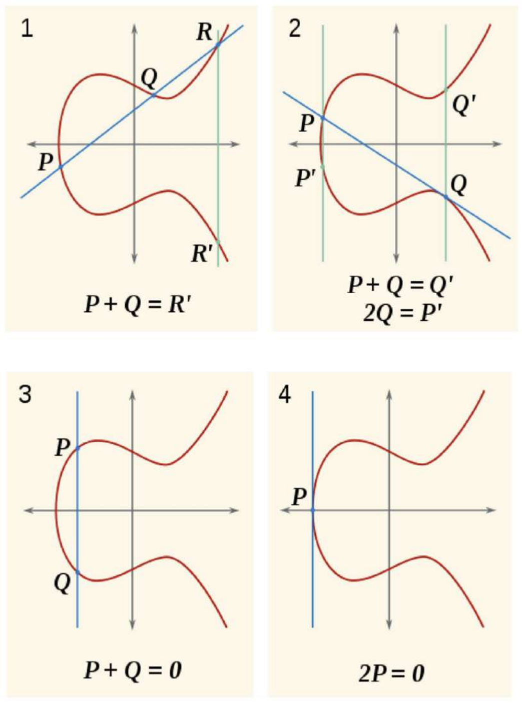

Fig 3: Example of the Elliptic Curve

ECC applies elliptic curve number theoretic problems. The security provided by ECC with a smaller key size is equivalent to that provided by RSA with a higher key size. Therefore, low-power electronics typically use ECC. Another PKC that makes use of the safety afforded by a discrete logarithm defined over a cyclic group ('G') is called Elgammal. Since it yields padded ciphertext, its use is limited to key exchange and not for encrypting plaintext [25-27]. Bitcoin and Ethereum both use elliptic curve points today. Every PKC has a backdoor function that calculates the private key if only the public key is known. Without exception for point addition or doubling, the suggested ECC algorithm's implementation of the Edward curve formula works well on all input point pairs. Elliptic Curve Cryptography is sped up by this technique. Time spent on arithmetic operations such point addition, multiplication, and doubling can be cut do wn by using sequential multiplication. As a result of employing Montgomery multiplier ECC in place of multiplication operations, latency is reduced, and the proposed technique increases operational speed.

3. METHODOLOGY

Key creation, data masking, encryption, and decryption are depicted in fig 4, along with the other procedures taken to develop a secure data sharing cloud system. The sender uploads the material to a centralized cloud storage system. Data obfuscation and encryption methods are implemented into the study to create a double layer of protection [28]. Data obfuscation is the initial layer of security, and it involves masking the data so that it cannot be read. As a result, the obfuscated data file is encrypted using ECC Mechanisms, which is the second layer of security.

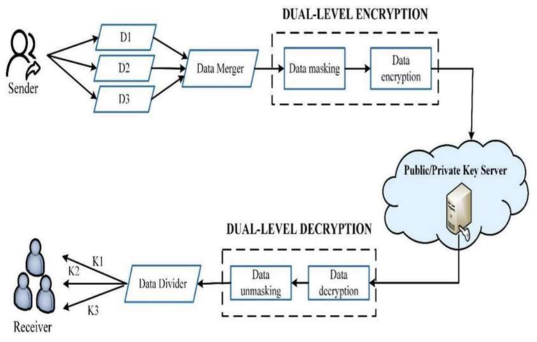

Fig 4: Existing cloud System design

Multiple components and levels of a cloud-based secure storage system's architecture often work together to protect stored information. A cloud-based secure storage system's exact architecture may change depending on the service provider and the needs of the application or business. To keep data safe in the face of evolving threats, it's important to periodically review and upgrade the security procedures in place [29]. Decryption with ECC is based on a shared secret point established between the sender's public key and the recipient's private key. The receiver can decrypt the message by utilizing the identical private key and public key pair that was used to encrypt it. This allows the receiver to calculate the shared secret point and obtain the symmetric encryption key. When transmitting encrypted data, it is crucial to use secure communication channels or encryption algorithms to maintain the data's privacy and integrity. In addition, regular key rotation and safe storage of private keys are essential to preserving the integrity of the decryption procedure. When it comes to protecting sensitive information in the cloud, using ECC for decryption is one of the most reliable methods available.

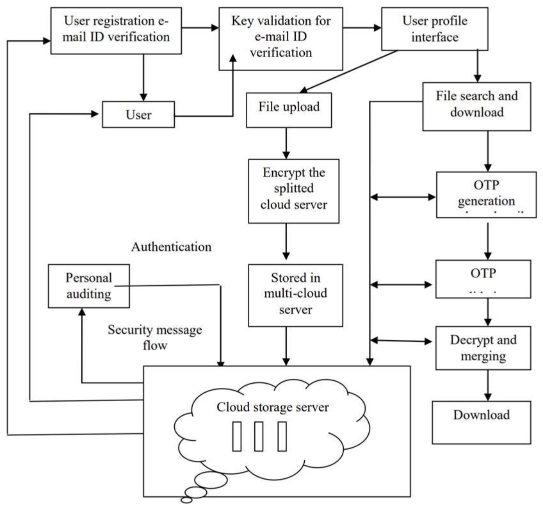

Fig 5: Proposed architecture of cloud based secure storage

3.1 ECC based key Generation using Cloud data security

When it comes to protecting sensitive information and ensuring private conversations, elliptic curve cryptography (ECC) is a popular public-key technique of choice. ECC is well-suited for cloud data security due to its robust security and very tiny key sizes in comparison to other algorithms like RSA. The following is an ECC-based key generation algorithm for use in the context of protecting cloud-stored data:

Algorithm 1: Key Generation using Cloud data security

1. Input: Select an elliptic curve (EC) and its parameters (e.g., prime modulus p, coefficients a and b, base point G, and order n).

2. Generate Private Key:

• Generate a random number k within the range [1, n-1].

• Ensure that k is a valid private key within the defined range.

3. Compute Public Key:

• Compute the elliptic curve point Q = k * G, where * denotes the point multiplication operation.

• Ensure that Q is a valid point on the chosen elliptic curve.

4. Encode Public Key:

• Convert the coordinates of the point Q into a suitable format for storage and transmission (e.g., hexadecimal or base64 encoding).

• Store the encoded public key alongside the corresponding data or in a key repository.

5. Store Private Key:

• Safely store the private key k, ensuring it is securely protected against unauthorized access.

6. Output:

• Return the private key (k) and the encoded public key (Q) as the result of the key generation process.

A cryptographically sound random number generator should be used in the key creation process to guarantee the keys' unpredictability and durability. In addition, the produced keys should be safeguarded against unwanted access and subjected to frequent key rotation as part of sound key management policies designed to assure their security and integrity. Elliptic curve parameters, such as the curve and the base point, should be selected from a predefined set of well-known curves for maximum safety. A cryptographically sound random number generator should be used in the key creation procedure to produce the private key. In addition, the private key should be kept secure and the produced keys should be kept secret by using good key management procedures.

3.2 ECC based key Distribution using Cloud data security

Secure key distribution in cloud data security scenarios is possible with Elliptic Curve Cryptography (ECC). Distributing encryption keys entails sending them from a sender to a receiver in a way that protects their privacy and security while they are in transit. When it comes to protecting sensitive information stored in the cloud, an ECC-based key distribution mechanism is commonly used.

Algorithm 2: key Distribution using Cloud data security

1. Input: Sender S and recipient R identities, sender's private key (kS), sender's public key (QS), recipient's public key (QR).

2. Sender S initiates the key distribution process:

• Sender S encrypts the data D using a symmetric encryption algorithm (e.g., AES) and generates a symmetric encryption key K.

• Sender S generates a random value r within the range [1, n-1] as a ephemeral private key.

• Sender S computes the ephemeral public key R = r * G, where * denotes point multiplication.

• Sender S computes the shared secret point P = r * QR.

• Sender S derives a shared secret key SK using a key derivation function (KDF) with P's x-coordinate or a hash of it as input.

3. Sender S encrypts the symmetric encryption key K:

• Sender S encrypts the symmetric encryption key K using a hybrid encryption scheme such as RSA or AES.

• Sender S uses the derived shared secret key SK as the encryption key for encrypting K.

4. Sender S sends the encrypted symmetric encryption key and the ephemeral public key to the recipient R.

5. Recipient R receives the encrypted symmetric encryption key and the ephemeral public key from sender S.

6. Recipient R computes the shared secret point P:

• Recipient R computes the shared secret point P = kR * R, where kR is the recipient's private key.

7. Recipient R derives the shared secret key SK:

• Recipient R derives the shared secret key SK using the same key derivation function (KDF) used by the sender S, using P's x-coordinate or a hash of it as input.

8. Recipient R decrypts the symmetric encryption key K:

• Recipient R uses the derived shared secret key SK as the decryption key to decrypt the encrypted symmetric encryption key received from sender S.

9. Recipient R decrypts the data D:

• Recipient R uses the decrypted symmetric encryption key K to decrypt the encrypted data D received from sender S.

10. Output: Recipient R obtains the decrypted data D, which can be processed or used for further operations.

Secure key distribution in cloud data security scenarios is possible with Elliptic Curve Cryptography (ECC). Distributing encryption keys entails sending them from a sender to a receiver in a way that protects their privacy and security while they are in transit. When it comes to protecting sensitive information stored in the cloud, an ECC-based key distribution mechanism is commonly used.

3.3 ECC based encryption using Cloud data security

Secure key distribution in cloud data security scenarios is possible with Elliptic Curve Cryptography (ECC). Distributing encryption keys entails sending them from a sender to a receiver in a way that protects their privacy and security while they are in transit. In most cases, a method for ECC-based encryption in the cloud is used to distribute keys for cloud-based data protection.

Algorithm 3: Encryption using Cloud data security

1. Input: Data D to be encrypted, recipient's public key QR.
2. Sender S retrieves the recipient's public key QR from a trusted source or key repository.
3. Sender S generates a random value k within the range [1, n-1] as an ephemeral private key.
4. Sender S computes the ephemeral public key R = k * G, where * denotes point multiplication.
5. Sender S computes the shared secret point P = k * QR, where QR is the recipient's public key.
6. Sender S derives a shared secret key SK using a key derivation function (KDF) with P's x-coordinate or a hash of it as input.
7. Sender S encrypts the data D using a symmetric encryption algorithm (e.g., AES) and a symmetric encryption key K.
8. Sender S securely transfers the encrypted data and the ephemeral public key R to the recipient.
9. Recipient R receives the encrypted data and the ephemeral public key R from the sender.
10. Recipient R computes the shared secret point P = kR * R, where kR is the recipient's private key.
11. Recipient R derives the shared secret key SK using the same key derivation function (KDF) used by the sender, using P's x-coordinate or a hash of it as input.
12. Recipient R decrypts the encrypted data using the derived shared secret key SK to obtain the original data D.
13. Output: Recipient R obtains the decrypted data D, which can be processed or used for further operations.

Elliptic curve cryptography (ECC) is an algorithm for encrypting data that relies on the features of elliptic curves to create a shared secret point and a symmetric encryption key. Forward secrecy is provided by the transient nature of the public key and the random selection of the private key, both of which contribute to the robustness of the encryption procedure. The encrypted data and the temporary public key should be transmitted through a secure channel or encryption mechanism to ensure their privacy and security. In addition to safekeeping private keys in a secure location and rotating encryption keys on a regular basis, good key management methods are essential for keeping encryption keys secure. Forward secrecy and resistance to certain cryptographic attacks are provided by the pair of temporary private and public keys. To further protect the privacy and integrity of the data, it is essential to guarantee the safe transfer of the encrypted data and the temporary public key between the sender and the recipient.

3.4 ECC based decryption using Cloud data security

In order to decipher encrypted ciphertext and recover the original data, Elliptic Curve Cryptography (ECC) can be employed for cloud data security. The following is an algorithm for ECC-based decryption, which can be used to ensure the safety of cloud data:

Algorithm 4: Decryption using Cloud data security

1. Input: Encrypted data D_enc, sender's public key QS, recipient's private key kR.
2. Recipient R retrieves the sender's public key QS from a trusted source or key repository.
3. Recipient R computes the shared secret point P = kR * QS, where kR is the recipient's private key.

4. Recipient R derives the shared secret key SK using a key derivation function (KDF) with P's x-coordinate or a hash of it as input.

5. Recipient R decrypts the encrypted data D_enc using the derived shared secret key SK to obtain the original data D.

6. Output: Recipient R obtains the decrypted data D, which can be processed or used for further operations.

Decryption with ECC is based on a shared secret point established between the transmitter's public key and receiver's private key. Here the reciepient can decrypt the message by utilizing the identical private key and public key pair that was used to encrypt it. This allows the receiver to calculate the shared secret point and obtain the symmetric encryption key. When transmitting encrypted data, it is crucial to use secure communication channels or encryption algorithms to maintain the data's privacy and integrity. In addition, regular key rotation and safe storage of private keys are essential to preserving the integrity of the decryption procedure. When it comes to protecting sensitive information in the cloud, using ECC for decryption is one of the most reliable methods available.

4. RESULTS

Calculating the computational cost of encoding and decoding 'n' characters of plain text involves counting the number of scalar multiplications, point additions, and point subtractions related to the points available in the elliptic curve. Suppose 'S' represent the expense of scalar multiplication, 'A' represent the expense of adding points on an elliptic curve, and 'SU' represent the expense of subtracting. The suggested encoding and decoding scheme's number of scalar and addition operations across 'n' number of plain text characters is listed in Table 1. According to the above table, the suggested technique's deciphering cost is lesser when compared with that of enciphering cost in terms of the computational cost of scalar and addition operations on an elliptic curve. Encoding requires 127 scalar multiplications before it can be done, however if the plain text is tiny, most of those multiplications may be pointless. As the proposed scheme's scalar multiplications are size-dependent on the plaintext, smaller inputs incur lower computational costs. Each plaintext character can be converted into its corresponding pair of elliptic curve coordinates by applying ECC to the plaintext. The suggested mapping approach for encryption results in just 'n+1' elliptic curve coordinates for 'n' number of plaintext characters. Therefore, the space complexity of the cryptographic technique is decreased because the ciphertext size is decreased. The elliptic curve P192 is used, and the following values for the experiment's domain are implemented.

gx,gy=(602046282375688656758213480587526111916698976636884684818,174050332293622031404857552280219410364023488927386650641)

a = -3

b=2455155546008943817740293915197451784769108058161191238065

p=6277101735386680763835789423207666416083908700390324961279

q=6277101735386680763835789423176059013767194773182842284

Encoding and decoding times for various sized samples of plain text encrypted using the P192 elliptic curve are listed in Table 1.

Table 1: Results for the Uploading time, Encryption time, Downloading time and Decryption time

Bit size in chars	Uploading Time	Encryption Time	Downloading Time	Decryption Time
3000	109	134	89	117
2500	95	102	74	94
2000	82	95	57	72
1500	61	69	45	53
1000	46	53	32	47
500	23	31	16	24

The experimental results demonstrate that the time needed to encrypt and decrypt a message varies with the size of the plain text, but that both processes still need comparatively tiny time when weighed against the level of defense offerred. The following data show the size of ciphertext generated by the proposed system and by other systems in the ECC NIST-supported P192 curve, as well as the time required for encryption and decryption.

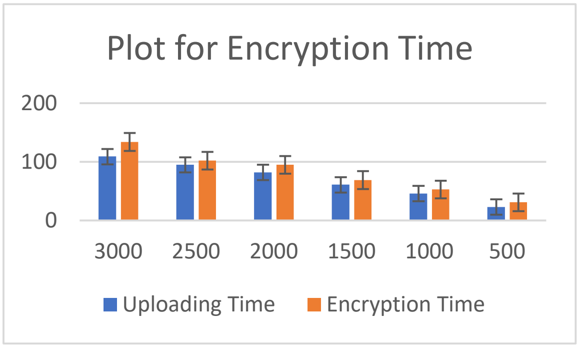

Fig 6: Plot for Encryption time

Figure 6 is a visual illustration of the fact that, in comparison to existing elliptic curve cryptographic methods that use the same encoding and decoding scheme for encryption and decryption, the proposed system creates a significantly smaller quantity of ciphertext.

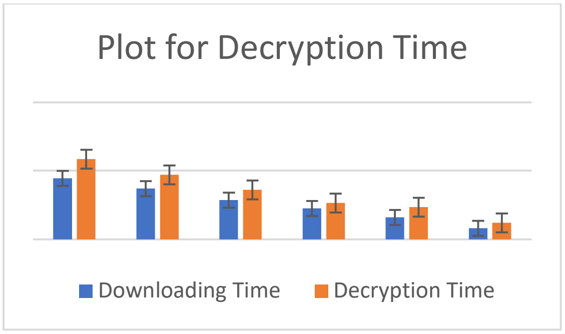

Fig 7: Plot for Decryption Time

Figure 6 and Figure 7 show the time it takes for various techniques to encrypt and decrypt 409 plaintext characters compared to the time it takes for the proposed scheme. Experiments have shown that the suggested approach requires less space for storing ciphertext and a shorter time period for both encrypting and decrypting data.

We now present a hypothesis for the results that can be generalized to 'n' as bit size

Hypothesis: As the bit size of data increases, the time required for both encryption and decryption also increases, but at a faster rate compared to uploading and downloading times.

This hypothesis suggests that there is a correlation between the bit size of data and the time required for encryption and decryption processes. It implies that larger data sizes require more time for encryption and decryption, likely due to increased computational complexity, whereas uploading and downloading times may increase at a slower rate, potentially influenced by factors other than just data size, such as network bandwidth or latency.

To convert the hypothesis into an equation, we can express it in a mathematical form based on the relationship described:

Let's denote:

T_u: Uploading Time
T_e: Encryption Time
T_d: Downloading Time
T_{dec}: Decryption Time
B: Bit size in chars

From the hypothesis it is evident that the Uploading Time and Downloading Time are sub-linearly related with bit size B.

A simple mathematical representation that captures this relationship is presented below:

$$T_e=k_1 \cdot B^a$$
$$T_{dec}=k_2 \cdot B^b \qquad (5)$$

k1 and k2 are constants that scale the encryption and decryption times respectively, and a and b are exponents that determine the rate at which these times increase with B.

For Uploading Time (T_u) and Downloading Time (T_d), if we assume they increase linearly or sub-linearly with B, we can represent them as:

$$T_u=k_3 \cdot B^c$$
$$Td=k_4 \cdot B^d \qquad (6)$$

where k3,k4 are constants and c and d are exponents that govern the rate of increase in uploading and downloading times with B.

This equation set encapsulates the hypothesis that as the bit size B increases, the times T_e, T_{dec} (for encryption and decryption) increase more rapidly compared to T_u and T_d (for uploading and downloading).

In general the same hypothesis can be represented for 'n' bits by the following way

$T_u(n)$: Uploading Time for n bits
$T_e(n)$: Encryption Time for n bits
$T_d(n)$: Downloading Time for n bits
$T_{dec}(n)$: Decryption Time for n bits

We hypothesize the following relationships:

Uploading Time and Downloading Time increase linearly or sub-linearly with n. Encryption Time and Decryption Time increase with n and may follow a more than linear relationship due to computational complexity.

Generalizing from the previous equations, we can write:

$$T_e(n)=k_1\cdot n^a$$
$$T_{dec}(n)=k_2\cdot n^b$$
$$T_u(n)=k_3\cdot n^c$$
$$T_d(n)=k_4\cdot^{nd} \quad (7)$$

where:k1,k2,k3,k4 are constants specific to the system and the environment (like hardware, software, and network characteristics),a,b,c,d are exponents that determine the rate at which the respective times increase with n

These equations provide a generalized form that represents the hypothesis that as the number of bits n increases, the times for encryption and decryption ($T_e(n)$ and $T_{dec}(n)$) increase more rapidly compared to the times for uploading and downloading ($T_u(n)$ and $T_d(n)$

5. CONCLUSIONS

The suggested approach facilitates anonymous and accountable access to outsourced sensor data by employing a modified cryptographic mechanism. By resolving the linear problem connected with the cryptographic algorithm, this technique allows for a thorough examination of security to verify the traceability and altruistic anonymity of the system. The efficiency of data collection and authentication by sensor nodes and mobile sinks is assessed using a complete lightweight protocol designed to enhance sensor-cloud security. The ECC protocol is modeled using a number of network parameters, such as communication range and energy consumption, and the behavior of potential intruders in the network is taken into account. Data gathered by sensor nodes and stored in a sensor cloud is encrypted using the suggested ECC protocol and Abelian group theory. This research prevents intruders from accessing the network, even if the sensor nodes are linked to a limitless trove of data. This paper details the method for improving the safety of ECC by mapping plain text to an elliptic curve point. Different points on the elliptic curve are used to encode the same characters, making the proposed novel mapping approach more safe. There is no necessity for both parties to share a comprehensive look up table. Large values of 'p' on elliptic curves make it impossible for attackers to estimate the result of operations like point addition and scalar multiplication. The time and space complexity of the proposed strategy is reduced, making it quick and effective. With the help of the improved mapping system, ECC became a highly effective cryptographic method. The proposed approach can be evaluated and refined across a variety of web application based cloud service models, including those used in online banking and retail purchasing. This security system has the ability to be utilized / implemented in different cloud models, such as fog and POI (Point of Interest) devices. When it comes to securing data in transit between the cloud and businesses, hybrid cloud architecture and its practical implementation stand out. Communication from the internet to the cloud, communication between apps within the cloud, and lastly communication between multiple clouds are also crucial parts of cloud security to analyze. More in-depth study of the challenges introduced by the combination of hybrid cloud and IaaS can be conducted in the future. For the data to be stored securely in a cloud database, a user interface module will need to analyze the decision manager that selects the most appropriate encryption and decryption technique. With the aid of the location manager, the time manager, and the access control algorithm, it is necessary to conduct additional research into the access control mechanism in order to adapt it to the evolving requirements of the new environment.

We also present a brief note on critique the work on key creation, data masking, encryption, and decryption for a secure data sharing cloud system.

Goals:

Security: Ensure that the data is securely stored and shared in the cloud.

Data Obfuscation: Mask data to make it unreadable.

Encryption: Use advanced encryption techniques to protect the data.

Accessibility: Allow authorized users to access the data when needed.

Outcomes:

Data Obfuscation and Encryption: The implementation of a double layer of security using data obfuscation and ECC (Elliptic Curve Cryptography) mechanisms.

Centralized Cloud Storage: Data is uploaded to a centralized cloud storage system.

ECC Mechanisms: Use of ECC for encryption, which is known for providing high security with smaller key sizes compared to other cryptographic methods.

Comparison with State-of-the-Art Solutions

1. Decentralized Storage Systems:

Example: IPFS (InterPlanetary File System) combined with blockchain technology.

Comparison: Decentralized systems offer better resilience against attacks on central servers and

improve data integrity and availability. The system described uses centralized storage, which is more vulnerable.
2. Homomorphic Encryption:

Example: Fully Homomorphic Encryption (FHE) allows computations on encrypted data without decrypting it.
Comparison: FHE provides a higher level of security and privacy since data can be processed without ever being exposed in plain text. The described system requires data to be decrypted for processing, potentially exposing it to risks.

Now we present a overall argument of this work:

The work is based on the argument that a secure data sharing cloud system can be effectively developed by combining data obfuscation and encryption mechanisms. The paper posits that:

Data Security in Cloud Environments: Storing and sharing data in cloud environments pose significant security risks, and therefore, advanced security measures are necessary.
Double Layer of Protection: Implementing a double layer of security—comprising data obfuscation followed by ECC (Elliptic Curve Cryptography) encryption—provides enhanced protection for the data.
Data Obfuscation: The initial layer of security involves masking the data to make it unreadable, which acts as a preliminary barrier against unauthorized access.
ECC Encryption: The obfuscated data is then encrypted using ECC mechanisms, which is known for its strength and efficiency, ensuring that even if obfuscation is compromised, the data remains secure.
Centralized Cloud Storage: The centralized cloud storage system is the chosen platform for data storage and sharing, presumably for ease of access and management.
Questions Raised by the Argument
Vulnerability of Centralized Storage:

Question: How does the system mitigate the risks associated with centralized storage, such as single points of failure or targeted attacks?
Unanswered: The paper does not address whether any redundancy, backup, or additional security measures are in place to protect against these vulnerabilities.

REFERENCES:

[1]. Xie Y, Sun Y, Li Q, Wu L. A dual-core high-performance processor for elliptic curve cryptography in GF(p) over generic Weierstrass curves. IEEE Trans Circuits Syst II Express Briefs. 2022 Nov;69(11):4523-7. doi: 10.1109/TCSII.2022.3194019.

[2]. Ma R, Du L. Attribute-based blind signature scheme based on elliptic curve cryptography. IEEE Access. 2022;10:34221-7. doi: 10.1109/ACCESS.2022.3162231.

[3]. Yeh L-Y, Chen P-J, Pai C-C, Liu T-T. An energy-efficient dual-field elliptic curve cryptography processor for Internet of Things applications. IEEE Trans Circuits Syst II Express Briefs. 2020 Sep;67(9):1614-8.doi: 10.1109/TCSII.2020.3012448.

[4]. Li B, Lei B, Zhang Y, Lei S. A novel and high-performance modular square scheme for elliptic curve cryptography over GF(p). IEEE Trans Circuits Syst II Express Briefs. 2019 Apr;66(4):647-51. doi: 10.1109/TCSII.2018.2867618.

[5]. Amiri R, Elkeelany O. FPGA design of elliptic curve cryptosystem (ECC) for isomorphic transformation and EC ElGamal encryption. IEEE Embed Syst Lett. 2021 Jun;13(2):65-8. doi: 10.1109/LES.2020.3003978.

[6]. Mehrabi MA, Doche C, Jolfaei A. Elliptic curve cryptography point multiplication core for hardware security module. IEEE Trans Comput. 2020 Nov;69(11):1707-18. doi: 10.1109/TC.2020.3013266.

[7]. Hernández Díaz EA, Pérez Meana HM, Silva García VM. Encryption of RGB images by means of a novel cryptosystem using elliptic curves and chaos. IEEE Lat Am Trans. 2020 Aug;18(8):1407-15. doi: 10.1109/TLA.2020.9111676.

[8]. Oudjida AK, Liacha A. Radix-2w arithmetic for scalar multiplication in elliptic curve cryptography. IEEE Trans Circuits Syst I Regul Pap. 2021 May;68(5):1979-89. doi: 10.1109/TCSI.2021.3054781.

[9]. Sowjanya K, Dasgupta M, Ray S, Obaidat MS. An efficient elliptic curve cryptography-based without pairing KPABE for Internet of Things. IEEE Syst J. 2020 Jun;14(2):2154-63. doi: 10.1109/JSYST.2019.2944240.

[10]. Almajed HN, Almogren AS. SE-Enc: A secure and efficient encoding scheme using elliptic curve cryptography. IEEE Access. 2019;7:175865-78. doi: 10.1109/ACCESS.2019.2957943.

[11]. Do-Nguyen BK, Pham-Quoc C, Tran N-T, Pham C-K, Hoang T-T. Multi-functional resource-constrained elliptic curve cryptographic processor. IEEE Access. 2023;11:4879-94. doi: 10.1109/ACCESS.2023.3236406.

[12]. Islam MM, Hossain MS, Hasan MK, Shahjalal M, Jang YM. FPGA implementation of high-speed area-efficient processor for elliptic curve point multiplication over prime field. IEEE Access. 2019;7:178811-26. doi: 10.1109/ACCESS.2019.2958491.

[13]. Ibrahim S, Alharbi A. Efficient image encryption scheme using Henon map, dynamic S-Boxes and elliptic curve cryptography. IEEE Access. 2020;8:194289-302. doi: 10.1109/ACCESS.2020.3032403.

[14]. Khalid I, Jamal SS, Shah T, Shah D, Hazzazi MM. A novel scheme of image encryption based on elliptic curves isomorphism and substitution boxes. IEEE Access. 2021;9:77798-810. doi: 10.1109/ACCESS.2021.3083151.

[15]. Oladipupo ET, Misra S, Bhalla V, Maskeliunas R, Damaševičius R. An efficient authenticated elliptic curve cryptography scheme for multicore wireless sensor networks. IEEE Access. 2023;11:1306-23. doi: 10.1109/ACCESS.2022.3233632.

[16]. Shen J, Zhou T, He D, Zhang Y, Sun X, Xiang Y. Block design-based key agreement for group data sharing in cloud computing. IEEE Trans Depend Secure Comput. 2019 Nov-Dec;16(6):996-1010. doi: 10.1109/TDSC.2017.2725953.

[17]. Awaysheh FM, Aladwan MN, Alazab M, Alawadi S, Cabaleiro JC, Pena TF. Security by design for big data frameworks over cloud computing. IEEE Trans Eng Manag. 2022 Dec;69(6):3676-93. doi: 10.1109/TEM.2020.3045661.

[18]. Zhang J, Lu R, Wang B, Wang XA. Comments on “Privacy-preserving public auditing protocol for regenerating-code-based cloud storage”. IEEE Trans Inf Forensics Secur. 2021;16:1288-9. doi: 10.1109/TIFS.2020.3032283.

[19]. Li X, Wang Q, Lan X, Chen X, Zhang N, Chen D. Enhancing cloud-based IoT security through trustworthy cloud service: an integration of security and reputation approach. IEEE Access. 2019;7:9368-83.doi: 10.1109/ACCESS.2018.2890432.

[20]. Thangavel M, Varalakshmi P. Enabling ternary hash tree based integrity verification for secure cloud data storage. IEEE Trans Knowl Data Eng. 2020 Dec;32(12):2351-62. doi: 10.1109/TKDE.2019.2922357.

[21]. Hababeh I, Gharaibeh A, Nofal S, Khalil I. An integrated methodology for big data classification and security for improving cloud systems data mobility. IEEE Access. 2019;7:9153-63. doi: 10.1109/ACCESS.2018.2890099.

[22]. Lan C, Wang C, Li H, Liu L. Comments on “Attribute-based data sharing scheme revisited in cloud computing”. IEEE Trans Inf Forensics Secur. 2021;16:2579-80. doi: 10.1109/TIFS.2021.3058758.

[23]. Wang H, He D, Fu A, Li Q, Wang Q. Provable data possession with outsourced data transfer. IEEE Trans Serv Comput. 2021 Nov-Dec;14(6):1929-39. doi: 10.1109/TSC.2019.2892095.

[24]. Kong F, Zhou Y, Xia B, Pan L, Zhu L. A security reputation model for IoT health data using S-AlexNet and dynamic game theory in cloud computing environment. IEEE Access. 2019;7:161822-30. doi: 10.1109/ACCESS.2019.2950731.

[25]. Yan H, Gui W. Efficient identity-based public integrity auditing of shared data in cloud storage with user privacy preserving. IEEE Access. 2021;9:45822-31. doi: 10.1109/ACCESS.2021.3066497.

[26]. Zhang X, Zhao J, Xu C, Wang H, Zhang Y. DOPIV: post-quantum secure identity-based data outsourcing with public integrity verification in cloud storage. IEEE Trans Serv Comput. 2022 Jan-Feb;15(1):334-45. doi: 10.1109/TSC.2019.2942297.

[27]. Yang C, Liu Y, Tao X, Zhao F. Publicly verifiable and efficient fine-grained data deletion scheme in cloud computing. IEEE Access. 2020;8:99393-403. doi: 10.1109/ACCESS.2020.2997351.

[28]. Zhang K, Jiang Z, Ning J, Huang X. Subversion-resistant and consistent attribute-based keyword search for secure cloud storage. IEEE Trans Inf Forensics Secur. 2022;17:1771-84. doi: 10.1109/TIFS.2022.3172627.

[29]. Gupta R, Saxena D, Gupta I, Singh AK. Differential and TriPhase adaptive learning-based privacy-preserving model for medical data in cloud environment. IEEE Netw Lett. 2022 Dec;4(4):217-21.doi: 10.1109/LNET.2022.3215248.