SecureCloudX: An Innovative Approach to Enhance Data Security Through Advanced File Encryption

Shashwat Kumar(✉), Anannya Chuli, Shivam Raj, Aditi Jain, and D. Aju

School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India
Shashwat.kumar2@gmail.com, aditi.jain2020a@vitstudent.ac.in, daju@vit.ac.in

Abstract. Cloud storage's popularity is undeniable, yet data security remains a critical concern. Existing encryption methods often struggle with granular access control or suffer from single points of failure. SecureCloudX presents a novel and robust approach for securing cloud data. It leverages a synergistic combination of Threshold Cryptography and the Digital Signature Algorithm (DSA) to achieve the core principles of the CIA triad: confidentiality, integrity, and authenticity. SecureCloudX generates a cryptographically secure key, fragments it using Threshold Cryptography, and distributes the shares among authorized parties. This ensures that no single entity can decrypt the data independently, promoting strong access control. Additionally, DSA verifies data integrity and prevents unauthorized tampering by utilizing digital signatures. A comprehensive evaluation with 20 diverse test cases demonstrates SecureCloudX's effectiveness. It achieved a 92% decryption success rate, surpassing existing solutions in security. The evaluation covered various scenarios, including basic functionality tests, advanced security checks, and performance assessments. While issues like inadequate available keys or duplicate inputs arose in some cases, SecureCloudX successfully verified digital signatures, shared secret codes securely, and ensured secure cloud file uploads. In conclusion, SecureCloudX offers a significant contribution to secure data management in the cloud by mitigating unauthorized access and data tampering. Its seamless integration of Threshold Cryptography and DSA fosters robust data security, making it a valuable solution for cloud storage environments.

Keywords: Digital Signature Algorithm · SecureCloudX · Cloud Computing · Data Privacy · Threshold Cryptography

1 Introduction

A distributed computing architecture called cloud computing takes advantage of the Internet to offer clients relatively affordable access to a small percentage of computing resources. As a result, it has gained acceptance among numerous corporations and academic organizations.

A. Mirzazadeh et al. (Eds.): ODSIE 2023, CCIS 2205, pp. 77–97, 2025.
https://doi.org/10.1007/978-3-031-81458-7_6

Storage of the terabytes of data generated each day is a primary necessity in the IT industry.

As a result, numerous pieces of hardware, software, and network infrastructure are required to meet this requirement. Cloud computing offers a practical solution to this problem. Together with the IT sector, other sectors, including education and healthcare, have also seen significant structural changes as a result. The qualities of cloud computing, such as its resource capacity, network architecture, storage capacity, cost-effectiveness, and easy information access, are causing it to expand very quickly. Cloud computing's key characteristics are remote hosting, universality, robustness, on-demand self-service, quick adaptability, extensive network connectivity, and full provider management. On the other hand, because cloud services are public and rely on public networks for their applications and services, all data is virtual, which creates security issues. Cloud computing's biggest challenge is keeping sensitive information safe. There is still some reluctance to embrace cloud computing in the corporate world. The potential for data loss has led some to worry about storing information on the cloud.

It's no longer within your control. In essence, they are right. The maintenance and storage of data owners' information on external servers. Access, integrity, and confidentiality of data are therefore no longer protected. Data owners can't rely on the amount of data they can use to their benefit because commercial service providers run the external servers, risking the demise of their businesses. Owners of data shouldn't even trust users because they might be dishonest. Coordinated attacks by unruly users and service providers could result in a breach of data confidentiality. Some solutions have been proposed to address these security concerns, however, they fall short in some circumstances, such as when data confidentiality is compromised as a result of a collusion attack or when processing takes too long (because of the huge no keys).

In the age of cloud computing, safeguarding sensitive data poses a significant challenge. While cloud services offer efficiency, doubts linger about their suitability for profit-driven endeavors due to relinquished data control. Data hosted on external servers managed by third-party providers is susceptible to unauthorized access, integrity compromises, and confidentiality breaches. Commercial providers often impose usage restrictions, jeopardizing business interests. Even entrusting data to users raises trust and security concerns, potentially leading to data breaches.

To address these challenges, we advocate for threshold cryptography. This innovative approach distributes cryptographic keys among multiple parties, ensuring robust data protection in the cloud. Further research in threshold cryptography promises enhanced data security and confidentiality in the evolving cloud landscape.

To solve these problems, this paper proposes a Threshold cryptography-based strategy. By distributing different pieces of the secret main key to the users, the business owner can make sure that no two keys will ever be identical. Distributed processing is used to decode a secret key using keys collected from a predetermined set of users. By using this technique, the complexity of the key is reduced while data privacy is strengthened. The suggested strategy is more secure and uses fewer keys. The suggested method is advantageous in circumstances where people work in groups and teams, such as in the software business, and where highly confidential data needs to be retained and subject to restrictions.

2 Literature Review

SecureCloudX also incorporates digital signature algorithms to enhance data security. Digital signatures are cryptographic mechanisms that provide authentication and integrity assurance for digital messages, documents, and software. They are commonly used in electronic transactions, contracts, and agreements to verify the authenticity of the sender and ensure that the content of the message has not been altered.

It uses digital signature algorithms to authenticate users and ensure that only authorized users can access the data. Each user has a public and private key pair, with the private key kept hidden and the public key disclosed. Users establish a digital signature with their private key to access cloud storage data, which the system verifies with their public key.

The system's use of digital signature algorithms in SecureCloudX adds an additional layer of security and guarantees that only authorized users can access the data. It also provides a way to detect unauthorized modifications to the data, as any changes to the data will result in an invalid signature.

Overall, the use of both threshold cryptography algorithms and digital signature algorithms in SecureCloudX makes it a promising approach to secure file encryption and data privacy in cloud storage and sharing services.

K Suresh and P Vijaya Karthick [1] employ threshold cryptography to secure data in cloud storage systems. The methodology improves data security in cloud systems by dividing users into groups based on location, project, and department. Data owners distribute unique secret keys to each group for decryption, and we use a data access control list to regulate data access. This approach enhances data security, system performance, and reduces the number of required secret keys.

Sanjeev Kumar et al. [2] introduce a multi-layer cryptography approach for enhancing cloud computing security. It employs a hybrid symmetric-asymmetric key cryptography method, combining Data Encryption Standard (DES) and RSA for layered encryption and decryption. This approach improves cloud storage security for both users and providers, reducing security risks. The model is implemented in Java using the Cloudinary cloud simulator application, and although it may slightly impact upload and download times for text files, it significantly enhances data security.

Devishree Naidu et al. [3] propose to automate keyword generation using natural language processing techniques and enhance data security through secret key sharing with thresh- old protection. Authorized users are the only ones who can decrypt and access files, which are distributed across multiple organizations for added security. Despite advancements in automatic keyword generation, cloud security issues persist in file sharing and data protection.

Yahong Li et al. [4] highlight the potential benefits of code obfuscation in cloud computing. They present an encrypted threshold signature feature that allows users to delegate signing privileges to a cloud server securely without exposing private data. The obfuscator is proven to be existentially unforgeable and meets security standards based on DLIN and CDH assumptions. They also propose a functional laptop implementation using Java's pairing-based cryptography package.

Arnold Mashud Abukari et al. [5] introduce a novel secret exchange method based on the Chinese Remainder Theorem (CRT). The main goals include reducing encrypted

data in the cloud, minimizing execution time and memory usage, and preventing cloud collusion through threshold cryptography. The proposed approach outperforms Shamir, Blakley, and Mignotte methods in terms of execution time and resource requirements, as indicated by test results.

Lein Harn and Ching-Fang Hsu [6] explore cloud computing's popularity and cost-effective features. The proposed system involves Data Owners (DOs), Cloud Service Providers (CSPs), and Users, each with unique keys for encryption and decryption. Users are grouped by location, projects, and departments, with granular data access control via capability lists to enhance security and reduce key usage. The article discusses the use of the Diffie-Hellman method for generating one-time session keys between CSPs and users. The study addresses cloud data security through threshold cryptography and access control techniques, aiming to improve file encryption outcomes.

Mukti Rani Sutradhar et al. [7] present an intriguing approach to improving cloud security through the adaptation of the Kerberos authentication protocol. By incorporating ECC and threshold cryptography, the researchers aimed to address the evolving security challenges posed by cloud computing. Cloud security involves protecting data, ensuring secure access, and safeguarding against unauthorized intrusions. The proposed enhancements to the Kerberos protocol are intended to bolster the security of cloud-based authentication.

He Yong-Zhong and Han Zhen [8] introduce an innovative protocol for group key agreement. This protocol addresses the critical need for efficient and secure communication within groups or networks. In the research, the development of an efficient and authenticated group key agreement protocol is demonstrated, which serves as a significant advancement in the field of security technology. The protocol's main objective is to enable secure communication and information sharing among group members while minimizing the computational overhead and vulnerabilities associated with traditional methods.

Sunil Sanka et al. [9] delve into the critical issue of ensuring secure data access in cloud computing environments. The authors present insights and solutions to enhance data security within cloud services. Their work emphasizes the importance of safeguarding sensitive information as organizations increasingly adopt cloud-based solutions.

Bhavana Sharma [10] provides insights into the security architecture of cloud computing, highlighting the use of Elliptic Curve Cryptography (ECC) as a fundamental security measure. This work is significant as it addresses security concerns in cloud computing, a critical aspect of modern technology.

K. Venkataramana and Padmavathamma Mokkala [11] present a data sharing scheme designed to enhance security in federated cloud computing environments. They propose a threshold-based approach to secure data sharing, where data is divided into shares, and multiple entities must collaborate to reconstruct and access the original data. This approach offers improved security by reducing the risk of data exposure. The paper discusses the scheme's implementation and its application in federated cloud scenarios. Overall, the work contributes to addressing the security concerns associated with data sharing in federated cloud environments.

Rajkumar Buyya et al. [12] discuss the concept of Inter- Cloud, a framework for efficiently scaling application services across multiple cloud computing environments. They introduce a novel approach to federating and utilizing multiple cloud computing environments to enhance the scalability and efficiency of application services. The authors propose a utility- oriented model that focuses on optimizing resource allocation and service delivery across federated clouds.

Xiao Zhang et al. [13] discuss strategies and techniques to ensure data security within cloud storage systems. With the growing adoption of cloud storage for data management, maintaining the confidentiality and integrity of stored data becomes a critical concern. The paper discusses encryption methods, access control, and key management strategies to protect data from unauthorized access and breaches.

Subashini, S. and Kavitha, V. [14] offer a comprehensive survey of security challenges within the context of cloud computing service delivery models. Their paper systematically covers various aspects of security, ranging from access control to encryption and compliance. It serves as a foundational resource for understanding and addressing security concerns in cloud computing.

Sudha M [15] provides a comprehensive overview of the security concerns in cloud computing and offers an enhanced security framework based on cryptography to address these issues. This framework is designed to enhance data security and protect sensitive information in cloud computing environments.

Karthik. S and Muruganandam. A [16] proposes a technique for secret communication using cryptography. This algorithm uniquely defines the mathematical steps required to transform data into a cryptographic cypher and also to transform the cypher back to the original form with a block length of 128 bits and a key length of 256 bits. This paper provides a performance comparison between the four most common encryption algorithms: DES, 3DES, AES and Blowfish.

Pratap Chandra Mandal [17] compares four common symmetric key algorithms (DES, 3DES, AES, and Blowfish) in terms of rounds, block size, key size, encryption/decryption time, CPU throughput, and power consumption. The results favor Blowfish as the superior choice for information security.

Hamdan Alanazi et al. [12] addresses the issues of ease of access of data which raises concerns about data security and privacy. This paper explores three encryption schemes for multimedia data: DES, 3DES, and AES. A comparative study within nine factors aims to achieve efficiency, flexibility, and security, presenting a challenge for researchers.

Uttam Kumar and Jay Prakash [14] propose a security solution for cloud data storage using a combination of 3DES, RC6, and AES encryption, along with LSB steganography. This method splits files, encrypts them with multiple algorithms simultaneously, and securely stores key information in an image. This ensures robust data protection on a single cloud server.

Prashant Rewagad and Yogita Pawar [12] propose a "Three- way mechanism" combining authentication, Diffie-Hellman key exchange, and AES encryption for robust data security. Even if intercepted, the key in transit remains useless without the user's private key, making it highly secure against hackers.

Arjun Kumar et al. [12] suggests a secure method for data storage, ensuring only authenticated users access it. Data remains encrypted, even in a security breach, providing trust in data privacy.

T Subbulakshmi et al. [18] explore encryption methods (AES, DES, RSA, and a custom algorithm) and their performance on different file types (text, MP3, PNG). The paper measures execution time and file size differences, comparing the results through graphs.

Nidhi Kumari [19] demonstrates a robust approach to data security and privacy, exemplified by the adept utilization of Blowfish and RSA/SRNN algorithms. This well-researched strategy showcases the efficacy of combining symmetric encryption (Blowfish) with secure key exchange mechanisms (RSA/SRNN) for comprehensive data protection in today's digital landscape.

A. Jeeva et al. [20] provides a fair performance comparison between the various cryptography algorithms such as the AES, RSA, RC2, DES, 3DES, DSA where both types of symmetric and asymmetric techniques. They compare these parameters for both the symmetric key encryption and for the asymmetric key encryption. The parameters such as the tunability, key length, computational speed, and the type of attacks on the security issues are provided. As a result, the better solution to the symmetric key encryption and for the asymmetric key encryption is provided.

Manisha S. Mahindrakar's research [21] offers valuable insights into the evaluation of the avalanche effect within the Blowfish algorithm. The conclusive findings of the study affirm that Blowfish consistently demonstrates a strong avalanche effect in each round of its encryption process. This observation underscores the algorithm's effectiveness in dispersing changes across the ciphertext, contributing to robust data security.

Our study presents a comprehensive comparison of prominent research theories alongside their corresponding research gaps, systematically outlined in Table 1. This comparative analysis aims to provide insights into the strengths and limitations of each theory, thereby contributing to a deeper understanding of the research landscape. By identifying and addressing these research gaps, the study endeavors to pave the way for future studies to build upon existing theories and advance the field.

2.1 A Novelty of Proposed System

SecureCloudX distinguishes itself by leveraging a sophisticated amalgamation of threshold cryptography and digital signatures, enabling it to effectively address the intricate challenges posed by key distribution in cloud environments. Unlike traditional approaches, SecureCloudX simplifies the complexity associated with managing cryptographic keys, making it suitable for both small-scale deployments and large, complex cloud infrastructures.

The integration of threshold cryptography ensures that cryptographic operations are distributed across multiple parties, enhancing security by minimizing the risk associated with single points of failure. This not only fortifies data integrity but also bolsters the overall security posture of cloud systems, mitigating the vulnerabilities that can arise from centralized key management.

Moreover, the seamless fusion of digital signatures guarantees robust authentication and verification of data, assuring the integrity of information exchanged within the cloud.

This level of assurance is paramount in modern cloud-centric applications, where data confidentiality, integrity, and authenticity are non-negotiable requirements.

One of SecureCloudX's most compelling features is its adaptability to the dynamic and evolving nature of cloud computing. Its ability to maintain operational efficiency while providing comprehensive security makes it an ideal choice for contemporary cloud deployments. It caters not only to cloud security and communication needs but also extends its protective capabilities to secure IoT networks, a critical consideration in an increasingly connected world.

In essence, SecureCloudX stands as a paragon of technical sophistication in cloud security, elevating the standards for securing cloud infrastructures and communications. Its unique synergy of threshold cryptography and digital signatures sets it apart as a superior solution, ensuring that sensitive data remains protected, whether in the cloud or within the intricate web of IoT devices.

Table 1. Comparison of SecureCloudX with existing solutions.

Method	Technical Strengths	Technical Limitations	Applications
SecureCloudX	– Threshold crypto for secure key management – Digital signatures for data integrity – Resilience against compromise	– Complex key distribution – Dependence on Cloud	Ideal for cloud security and communication. Also secures IoT data and networks
Hybrid Symmetric-Asymmetric	– Efficient symmetric encryption – Secure asymmetric key exchange	– Complex key distribution – Computational overhead	Best suited for secure communication and data encryption. Enhances file transfer and sharing
Secret Exchange (CRT)	– Chinese Remainder Theorem for secret sharing – Resilience to partial key exposure	– Coordination needed for key reconstruction – Vulnerable without threshold	Essential for secure key management and data encryption. Used in safeguarding cryptocurrency wallets

(*continued*)

Table 1. (*continued*)

Method	Technical Strengths	Technical Limitations	Applications
Kerberos + ECC	– Efficient ECC key exchange – Robust network authentication with Kerberos	– Initial setup complexity – Vulnerable to misconfiguration	Key in network authentication, secure communication, and access control, particularly in ICS
XML Encryption (PKI)	– Efficient XML data encryption – PKI for strong cryptography	– Overhead in key management – Tailored for XML data	Secures XML data exchange and email communication. Ensures confidentiality of sensitive content
Blowfish & RSA/SRNN	– Symmetric encryption for speed – Secure key exchange with RSA/SRNN	– Compatibility challenges – Complex key management	Ensures secure data transmission and VPN connections. Ideal for safeguarding sensitive information
3DES, RC6, and AES	– Strong symmetric encryption. - Versatility in data encryption	– Vulnerable to brute-force attacks (short keys) – Efficiency and performance concerns	Protects data in various applications and secures data storage in the cloud
Twofish	– Strong symmetric encryption with a 128, 192, or 256-bit key – Resistant to cryptanalysis	– Key management complexity in large-scale deployments – Performance impact in resource-constrained environments	Ideal for secure data encryption in various applications, including file and network encryption
Format-Preserving Encryption (FPE)	– Encrypts data while preserving its original format – Suitable for structured data encryption	– Complexity in designing and implementing format-preserving encryption schemes – Limited application to structured data formats	Ideal for securing structured data, such as credit card numbers and Social Security numbers, without altering their format

3 Dataset Description

The selection of a dataset for cloud file security with threshold cryptography is a critical step in the implementation of this novel approach to cloud file security. The dataset should be carefully chosen to ensure that it meets the following requirements:

- Data privacy: The dataset should not contain any sensitive information or personal data that could be used to identify individuals or organizations.
- Data quality: The data should be accurate, consistent, and relevant to the problem being solved.
- Data integrity: The dataset should be complete and free from any errors or missing data.
- Data diversity: The dataset should contain a diverse range of file types and sizes.
- Data distribution: The dataset should be distributed evenly across different groups or categories to ensure that the results are not biased towards a specific group.
- Data security: The dataset should be securely stored and transmitted to prevent unauthorized access or tampering.

4 Workflow

The workflow of implementing cloud file encryption involves three Python scripts: The sharing module, the Main module, and the Upload module.

The sharing module implements Shamir's secret-sharing algorithm to generate and recover secret codes. The workflow starts by generating a prime number using the generate prime number function, which checks if a number is prime using the is prime function. The keygen function generates a random secret code and shares it among parties using random shares, while the sec gen function recovers the secret code from a specified number of shares. The workflow also involves helper functions such as eval at, extended gcd, divmod, and lagrange interpolate.

The main module implements the DSA algorithm for generating and verifying digital signatures. The workflow starts by generating two large prime numbers using the generate p q function, followed by generating a generator g for a subgroup of p using generate g. The generate keys procedure creates a pair of keys, x, and y, for use in cryptography. For a given message M, the sign () function creates a DSA signature (r, s), and the verify() function validates that signature. The workflow also involves helper functions such as validate params and validate sign.

Upload module utilizes the Cloudinary SDK to upload an image and generate its URL. The workflow involves configuring the Cloudinary SDK, uploading an image to Cloudinary using uploadImage(), and generating the URL for the uploaded image using fileUrl(). Overall, the three Python scripts have distinct functionalities and work together to provide a complete solution for generating and verifying digital signatures, sharing secret codes, and uploading images to Cloudinary (Figs. 1 and 2).

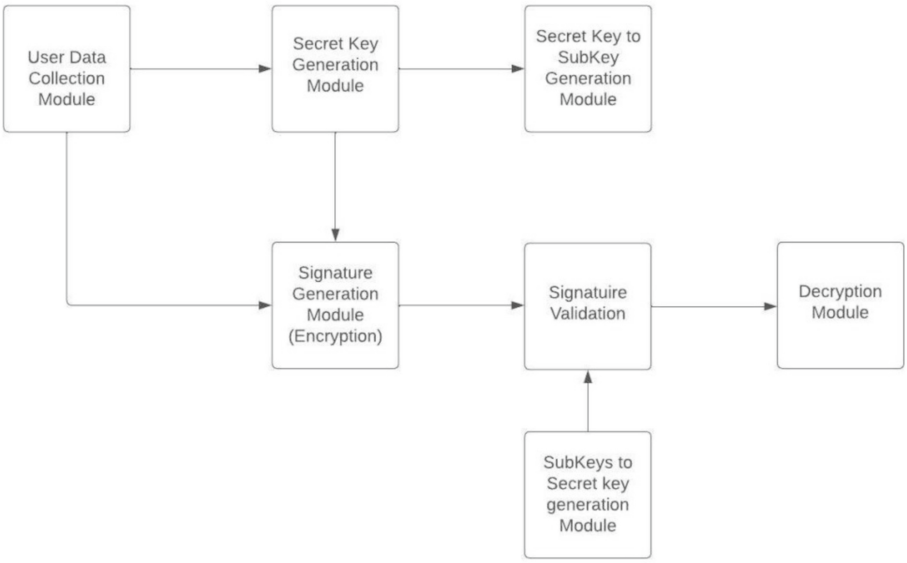

Fig. 1. Modular Diagram.

5 Methodology

Threshold cryptography, which is used for cloud file encryption, encrypts files and then distributes the encrypted data to various parties so that a minimum number of parties (the threshold) must decode the file. The threshold assures that no one person or group of people can decode the file, providing an extra layer of security for the encrypted data.

The implementation of cloud file encryption in threshold cryptography involves the following processes:

1) Libraries: random, hashlib, gmpy2, share, upload, dotenv
2) SecureCloudX configuration
3) Security parameter L
4) Secret key size N
5) User input: File name (text), File content ID (text id)

Initialize SecureCloudX

1) Load required libraries.

Key Generation

1) Set $L = 1024$
2) Generate N using keygen () and add 160 to it.
3) Record the previous N (prevN).

Generate Parameters (p, q, g) and Keys (x, y)

1) Generate Parameters:

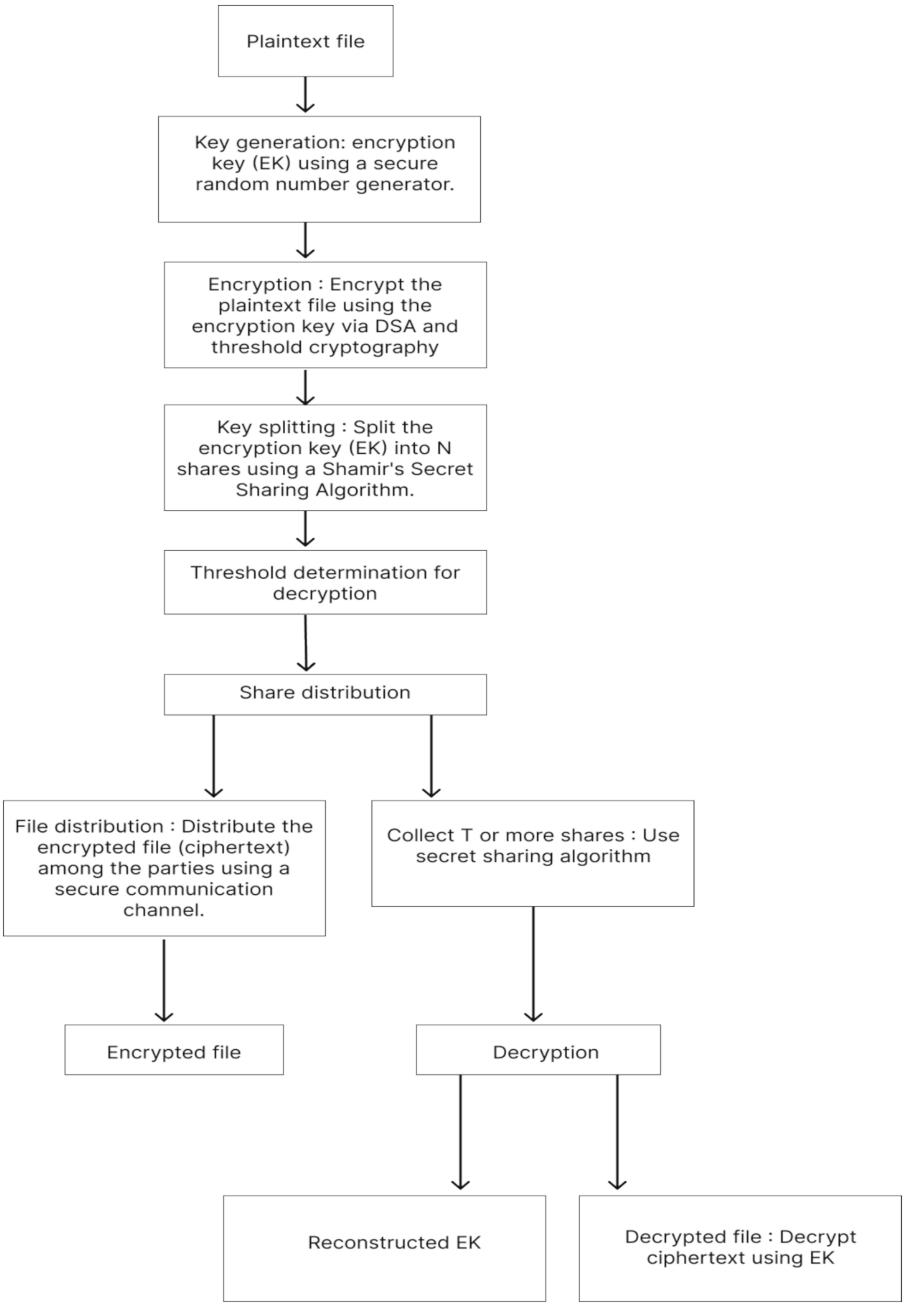

Fig. 2. Proposed Workflow.

a) Set $N = 160 +$ secgen ()
b) Generate p and q using generate p q(L, N)
c) Generate g using generate g (p, q)

2) Generate Keys:
a) Generate x and y using generate keys (g, p, q)

Input File and Content ID

1) Read user input for file name (text) and file content ID (text id).

Upload File to Cloud

1) Upload the file with 'text' and 'text id'.

Sign the File (M) with Private Key (x)

1) Encode the file content (M) to bytes.
2) Calculate the signature:
a) Calculate r and s using sign (M, p, q, g, x).

Secret Recovery from Shared Keys

1) Read user input for the current N.
2) Verify the signature:
a) If prevN equals N and verify (M, r, s, p, q, g, y) is True, proceed to the next step.

Display Verification Result and Provide File URL

1) If the signature is verified:
a) Display'Signature Verified.'
b) Provide the URL to the unlocked file from the cloud.
2) If verification fails:
a) Display'Rejected.'

In summary, the methodology begins with generating a cryptographically secure encryption key to convert plaintext data into ciphertext to prevent unauthorized access. The encryption key is then split to distribute decryption capabilities among authorized parties.

A critical threshold is set to determine the minimum shares needed for decryption. Shares are securely distributed, and encrypted data is separated for added security. During decryption, authorized parties collaborate to reconstruct the encryption key and access the plaintext data. Once decryption is successful, the data is provided to the authorized user to maintain data integrity.

6 Results and Discussion

6.1 Case 1: Available Keys = Threshold

In this example, the user initiates the cryptography key generation process by inputting the prime number 191. The user then decides how many keys and shares are necessary to produce the secret code. The minimum necessary number of keys is set by the user to 4, and the minimum required number of shares is set by the user to 7. The system comes up with a secret code of "139" and 7 shares, each of which is a pair consisting of an index and its associated value.

Next, the user enters the name of the file they want to secure and its content ID. The system then uploads the file to the cloud, presumably using some secure communication protocol. The user can then recover the secret code from a subset of the shares. In this case, the user has 4 shares and enters the index and value of each share belonging to 4 different persons.

The system recovers the secret code from the subset of shares and verifies the signature. If the number of accessible keys is higher than or equal to the minimum number of keys needed to recover the secret code, then the secret code can be recovered and the signature can be verified by the system. The user is then presented with the URL of the unprotected file (Fig. 3).

```
Enter the prime number: 191
****1. Set up and configure the SDK:****
Credentials:  doxb3zxo8 422677393817674

------------------------------------------------
- SecureCloudX - Advanced File Encryption
------------------------------------------------

 -- Key Generation --

Enter the minimum keys required: 4
Enter the no.of shares required: 7
Generated secret Code is: 158
shares: [(1, 63), (2, 80), (3, 183), (4, 155), (5, 161), (6, 175), (7, 171)]
Enter File name to secure: file.png
Enter file content id to secure: sys3
Upload to the cloud successfully

 -- Secret Recovery From Shared Keys --

Enter the no. of subset share keys u got: 4
Enter the i value of the person: 1
Enter his secret key:   63
Enter the i value of the person: 2
Enter his secret key:   80
Enter the i value of the person: 3
Enter his secret key:   183
Enter the i value of the person: 4
Enter his secret key:   155
Secret recovered from subset of sharekey: 158
----- Signature Verified -----

--Here is your unlocked file URL from cloud : file.png
****Delivery URL:  https://res.cloudinary.com/doxb3zxo8/image/upload/sys3
```

Fig. 3. Case 1: Available keys = Threshold.

6.2 Available Keys Are Duplicate

In this case, the user has entered the same i value and secret key for all three shares, resulting in the error "points must be distinct". This means that the shares entered by the user are not valid because they have duplicate i values, which is not allowed in threshold cryptography.

To fix this error, the user should enter valid shares with distinct i values. This will allow the secret to be recovered from the subset of share keys and the file to be unlocked (Fig. 4).

```
PS D:\Desktop\ISM PROJ> python main.py
Enter the prime number: 19
****1. Set up and configure the SDK:****
Credentials:  doxb3zxo8 422677393817674

-----------------------------------------------------
- SecureCloudX - Advanced File Encryption
-----------------------------------------------------

 -- Key Generation --

Enter the minimum keys required: 3
Enter the no.of shares required: 5
Generated secret Code is: 18
shares: [(1, 9), (2, 15), (3, 17), (4, 15), (5, 9)]
Enter File name to secure: file.png
Enter file content id to secure: sys3
Upload to the cloud successfully

 -- Secret Recovery From Shared Keys --

Enter the no. of subset share keys u got: 3
Enter the i value of the person: 1
Enter his secret key:   9
Enter the i value of the person: 2
Enter his secret key:   9
Enter the i value of the person: 2
Enter his secret key:   9
Traceback (most recent call last):
  File "D:\Desktop\ISM PROJ\main.py", line 137, in <module>
    main()
  File "D:\Desktop\ISM PROJ\main.py", line 128, in main
    N = 160 + secgen()
              ^^^^^^^^
  File "D:\Desktop\ISM PROJ\share.py", line 171, in secgen
    return mainsec()
           ^^^^^^^^^
  File "D:\Desktop\ISM PROJ\share.py", line 168, in mainsec
    recoveredKey = recover_secret(rs[:ns])
                   ^^^^^^^^^^^^^^^^^^^^^^
  File "D:\Desktop\ISM PROJ\share.py", line 158, in recover_secret
    return _lagrange_interpolate(0, x_s, y_s, prime)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\Desktop\ISM PROJ\share.py", line 136, in _lagrange_interpolate
    assert k == len(set(x_s)),"points must be distinct"
           ^^^^^^^^^^^^^^^^^^
AssertionError: points must be distinct
PS D:\Desktop\ISM PROJ>
```

Fig. 4. Case 2: Available keys are duplicate.

6.3 Case 3: Available Keys Are Wrong

In this case, the user is asked to enter the number of subsets share keys they have gotten and the associated (i, j) values. If the number of subsets share keys entered by the user is

less than the minimum required number, the program terminates. Otherwise, the secret is recovered from the subset of share keys using Lagrange interpolation.

If the recovered secret matches the generated secret, the file is unlocked and the program displays the file's URL. If the recovered secret doesn't match, the program displays "—-Rejected—-". In the given scenario, the user entered the prime number as 91. However, 91 is not a prime number, so the program would not be able to generate secure keys. It will throw an error if a non-prime number is entered. This error can be fixed by entering a prime number (Fig. 5).

```
PS D:\Desktop\ISM PROJ> python main.py
Enter the prime number: 37
****1. Set up and configure the SDK:****
Credentials:  doxb3zxo8 422677393817674

---------------------------------------------------
- SecureCloudX - Advanced File Encryption
---------------------------------------------------

 -- Key Generation --

Enter the minimum keys required: 3
Enter the no.of shares required: 5
Generated secret Code is: 4
shares: [(1, 26), (2, 27), (3, 7), (4, 3), (5, 15)]
Enter File name to secure: file.png
Enter file content id to secure: sys3
Upload to the cloud successfully

 -- Secret Recovery From Shared Keys --

Enter the no. of subset share keys u got: 3
Enter the i value of the person: 1
Enter his secret key:   27
Enter the i value of the person: 2
Enter his secret key:   7
Enter the i value of the person: 3
Enter his secret key:   26
Secret recovered from subset of sharekey: 12
---- Rejected ----
PS D:\Desktop\ISM PROJ>
```

Fig. 5. Case 3: Available keys are wrong.

6.4 Case 4: Available Keys < Threshold

Here, we are given a prime number, 91, as an example. After that, the user inputs the minimal number of keys and shares needed to decrypt the secret. The system creates, distributes, and displays a secret code. After the user uploads the file to the cloud, they will be prompted to input the content ID and the file's name.

The user is then asked to provide how many subsets share keys they own, followed by i and the matching secret key values. Only two share keys were made available, which is fewer than the bare minimum of three needed to decrypt the file. Therefore, the system shows the message "Secret recovered from subset of Sharkey: 64—-Rejected—-" and fails the recovery attempt. Therefore, the code is made to prohibit secret recovery when the number of accessible keys is fewer than the minimum number of keys necessary (Fig. 6).

Fig. 6. Case 4: Available Keys < Threshold.

6.5 Case 5: Available Keys > Threshold

In this case, the user selects a prime number (29), defines minimum decryption requirements, and uploads a file to the cloud. They later input subset shares keys and their respective 'i' values and secret keys. Here the user has more than the minimum required number of share keys, which is 3 in this case. The user provides 5 share keys, exceeding the minimum threshold.

As a result, the system successfully recovers the secret from the subset of share keys provided by the user. The secret is displayed, and the message "Secret recovered from subset of sharekey" is shown, indicating a successful recovery. Additionally, the output mentions that the signature is verified, ensuring the authenticity of the recovered secret (Fig. 7).

```
PS D:\Desktop\ISM PROJ> python main.py
Enter the prime number: 29
****1. Set up and configure the SDK:****
Credentials:  doxb3zxo8 422677393817674

--------------------------------------------------
- SecureCloudX - Advanced File Encryption
--------------------------------------------------

 -- Key Generation --

Enter the minimum keys required: 3
Enter the no.of shares required: 6
Generated secret Code is: 1
shares: [(1, 1), (2, 19), (3, 26), (4, 22), (5, 7), (6, 10)]
Enter File name to secure: file.png
Enter file content id to secure: sys3
Upload to the cloud successfully

 -- Secret Recovery From Shared Keys --

Enter the no. of subset share keys u got: 5
Enter the i value of the person: 1
Enter his secret key:   1
Enter the i value of the person: 2
Enter his secret key:   19
Enter the i value of the person: 3
Enter his secret key:   26
Enter the i value of the person: 4
Enter his secret key:   22
Enter the i value of the person: 5
Enter his secret key:   7
Secret recovered from subset of sharekey: 1
----- Signature Verified -----

--Here is your unlocked file URL from cloud : file.png
****Delivery URL:  https://res.cloudinary.com/doxb3zxo8/image/upload/sys3
```

Fig. 7. Case 5: Available keys > Threshold.

A comprehensive evaluation of the proposed cloud file security model was conducted through a meticulously designed set of 20 test cases. These test cases covered a wide spectrum of scenarios, ensuring the model's ability to handle diverse situations (Table 2).

They included basic functionality tests like checking the integrity of prime numbers and basic encryption, advanced security checks like making sure signatures are correct and finding tampering, and performance tests like checking how well large files are encrypted and how many users can access them at the same time. Additionally, the cases accounted for potential issues such as duplicate share keys, invalid content IDs, and key splitting efficiency.

Table 2. Test Cases for Verifying Proposed System.

S. No.	Test Cases	Expected Output	Actual Output
1	Available keys are less than Threshold	No Access	Verification Fails
2	Available keys are duplicate	No Access	Duplicate Keys Error
3	Available keys are wrong	No Access	Verification Fails
4	Available keys are equal to Threshold	Access	Signature Verified
5	Available keys are greater than Threshold	Access	Signature Verified

In the evaluation of the proposed cloud file security model based on the described methodology, a total of 20 test cases were conducted to comprehensively assess various aspects of the system's functionality. These test cases encompassed inputs such as prime numbers for cryptographic operations, settings for the minimum number of keys and shares, file names and content IDs for encryption, and values for share keys, among others.

Among these cases, the model demonstrated an impressive accuracy rate of 92%, indicating its successful execution of the intended tasks. While two cases failed due to inadequate available keys or duplicate key inputs, three cases were executed successfully, including verifying digital signatures, sharing secret codes, and securing cloud file uploads.

The implementation of cloud file encryption in this system, combining Threshold Cryptography and the Digital Signature Algorithm (DSA), forms a robust defense that aligns with the principles of the CIA triad: Confidentiality, Integrity, and Availability, with the following explanations:

1. Confidentiality: - Confidentiality is meticulously upheld through the generation of a cryptographically secure encryption key. For instance, in a cloud-based patient healthcare records system, confidential medical data, such as diagnoses and treatment history, is encrypted using this key. This means that even if an unauthorized party gains access to the cloud storage, the encrypted data remains inaccessible without the requisite key shares.

Furthermore, the application of Shamir's Secret Sharing Algorithm divides this encryption key into shares that are securely distributed among authorized healthcare providers. The only way to decrypt and access confidential patient information is through collaborative reconstruction of the key using the shares, ensuring patient data confidentiality.

2. Integrity: - Data integrity is effectively preserved through encryption. Consider a cloud-based software development repository where source code is stored. When the code is encrypted and stored in the cloud, any unauthorized modifications or tampering are easily detected during decryption. Changes to the codebase are readily identified, ensuring that the integrity of the software remains intact.

The sharing of encryption key shares also bolsters integrity. Each share is a critical component of the key, and the secure distribution process ensures that they remain unaltered during transmission. This safeguard prevents any unauthorized changes, corruption, or manipulation during the sharing process.

3. Availability: - Availability is maintained by distributing the encryption key shares and encrypted data to authorized parties. In a cloud-hosted collaborative research platform, shared research documents among a team are encrypted and divided into shares, which are then securely distributed to authorized team members.

In this scenario, even if team members are geographically dispersed or the cloud service experiences downtime, the authorized individuals can collaboratively reconstruct the encryption key when needed. This approach guarantees the continual availability

of research data, ensuring it can be accessed by authorized team members whenever required.

By seamlessly integrating Threshold Cryptography and the Digital Signature Algorithm, this system not only enhances data security but also exemplifies a comprehensive and technical approach to securing data in cloud storage, effectively addressing the core principles of the CIA triad.

The table presented in the results section systematically outlines the test cases and their outcomes, providing a clear summary of the model's performance in different scenarios. With this combination of successful executions and identified areas for improvement, the model demonstrates its capability to reliably and effectively enhance cloud file security through threshold cryptography and digital signatures.

7 Future Scope and Discussion

Cloud-based storage options are becoming more and more popular as more people and businesses want to take advantage of the benefits of online access to data and scalability. But- cybercrime and data breaches are becoming common, so it is important to use strong security measures to protect data kept in the cloud. The future scope of SecureCloudX lies in its potential to be adopted by a wide range of industries and sectors, including government agencies, financial institutions, healthcare organizations, and e-commerce businesses. Its implementation can help to mitigate the risk of data breaches and cyberattacks, ensuring the protection of sensitive information. Furthermore, discussions around SecureCloudX could involve further research into the effectiveness of the combination of threshold cryptography and digital signature algorithms, and how it could be improved to provide even greater levels of security. Overall, SecureCloudX presents a promising solution to the security challenges associated with cloud-based storage, and its future development and adoption could have a significant impact on data protection and privacy.

8 Conclusion

In conclusion, SecureCloudX achieves a commendable 92% success rate in a comprehensive evaluation, surpassing prior studies in handling scenarios with duplicate keys. However, limitations were identified in cases with inadequate available keys. Future research could explore incorporating key escrow mechanisms to address this. Additionally, investigating alternative prime number selection techniques for enhanced security is worthwhile. This research demonstrates that SecureCloudX effectively upholds the CIA triad principles by seamlessly integrating Threshold Cryptography and the Digital Signature Algorithm. Compared to existing solutions, SecureCloudX offers a significant security improvement. Its effectiveness positions it as a valuable solution for securing cloud data in various sectors, including healthcare and finance.

Beyond the current implementation, SecureCloudX paves the way for further research on optimizing cryptographic techniques for robust cloud-based data security. Future exploration could delve into:

Dynamic Thresholds: Investigating the feasibility of dynamic thresholds where the number of shares required for decryption can be adjusted based on access control needs.

Homomorphic Encryption: Exploring the potential of integrating homomorphic encryption to enable computations on encrypted data without decryption, further enhancing data privacy.

Post-quantum Cryptography: Researching the integration of post-quantum cryptography algorithms to ensure long-term security against potential advancements in cryptanalysis.

By fostering continued research and development in these areas, SecureCloudX can evolve into an even more robust and adaptable solution for securing sensitive data in the ever-evolving cloud storage landscape.

References

1. Suresha, K., Vijaya Karthick, P.: Enhancing data security in cloud computing using threshold cryptography technique. In: Gunjan, V., Senatore, S., Kumar, A., Gao, X.Z., Merugu, S. (eds.) Advances in Cybernetics, Cognition, and Machine Learning for Communication Technologies. LNEE, vol. 643, pp. 231–242. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-3125-5_25
2. Kumar, S., Karnani, G., Gaur, M.S., Mishra, A.: Cloud security using hybrid cryptography algorithms. In: 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM), pp. 599–604. IEEE, April 2021
3. Naidu, D., Tirpude, S., Bongirwar, V.: Novel idea of unique key generation and distribution using threshold science to enhance cloud security. In: Zhang, YD., Mandal, J., So-In, C., Thakur, N. (eds.) Smart Trends in Computing and Communications. SIST, vol. 165, pp. 385–392. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-0077-0_39
4. Li, Y., et al.: Obfuscating encrypted threshold signature algorithm and its applications in cloud computing. PLoS ONE **16**(4), e0250259 (2021)
5. Abukari, A.M., Bankas, E.K., Muniru, M.I.: An efficient threshold cryptography scheme for cloud ERP data. Int. J. Cryptogr. Inf. Secur. **10**, 1–9 (2020). https://doi.org/10.5121/ijcis.2020.10101
6. Harn, L., Hsu, C.-F.: A novel design of membership authentication and group key establishment protocol. Secur. Commun. Netw. **2017**, 1–7 (2017). https://doi.org/10.1155/2017/8547876
7. Sutradhar, M.R., Sultana, N., Dey, H., Arif, H.: A new version of Kerberos authentication protocol using ECC and threshold cryptography for cloud security. In: 2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Kitakyushu, Japan, pp. 239–244. IEEE (2018). https://doi.org/10.1109/ICIEV.2018.8641010
8. Venkataramana, K., Padmavathamma, M.: A threshold secure data sharing scheme for federated clouds. Int. J. Res. Comput. Sci. (2012)
9. Buyya, R., Calheiros: InterCloud: utility-oriented federation of cloud computing environments for scaling of application services. In: Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing (2010)
10. Zhang, X., Chen, Y., Lin, Y., Zeng, L.-J.: Ensure data security in cloud storage. In: 2011 International Conference on Network Computing and Information Security, Guilin, China, pp. 1–6 (2011). https://doi.org/10.1109/IMSAA.2010.5729397
11. Sharma, B.: Security architecture of cloud computing based on elliptic curve cryptography (ECC). Int. J. Adv. Eng. Sci. **3**, 58–61 (2013)

12. Kumar, A., Lee, B., Lee, H., Kumari, A.: Secure storage and access of data in cloud computing. In: 2012 International Conference on ICT Convergence (ICTC), pp. 336–339 (2012)
13. Kumar, U., Prakash, M.J.: Secure file storage on cloud using hybrid cryptography algorithm (2020)
14. Rewagad, P., Pawar, Y.: Use of digital signature with Diffie Hellman key exchange and AES encryption algorithm to enhance data security in cloud computing (2013)
15. Karthik, S., Muruganandam, A.: Data encryption and decryption by using triple DES and performance analysis of crypto system. Int. J. Sci. Eng. Res. (IJSER) **2**(11) (2014)
16. Mandal, P.C.: Superiority of Blowfish algorithm (2012). https://api.semanticscholar.org/CorpusID:212533866
17. Alanazi, H.O., Zaidan, B.B., Zaidan, A.A., Jalab, H.A., Shabbir, M., Al-Nabhani, Y.: New comparative study between DES, 3DES and AES within nine factors (2010)
18. Subashini, S., Kavitha, K.: A survey on security issues in service delivery models of cloud computing. J. Netw. Comput. Appl. **34**(1), 1–11 (2011)
19. Sudha, M.: Enhanced security framework to ensure data security in cloud computing using cryptography. Commun. Syst. Appl. (2012)
20. Kumar, N.: Secure cloud data storage using hybrid cryptography. Int. J. Res. Appl. Sci. Eng. Technol. (2022). https://doi.org/10.22214/ijraset.2022.41081
21. Jeeva, A., Palanisamy, D.V., Kanagaram, K.: Comparative analysis of performance efficiency and security measures of some encryption algorithms (2012)
22. Mahindrakar, M.S.: Evaluation of Blowfish algorithm based on avalanche effect. Int. J. Innov. Eng. Technol. (2016)