Storage and Maintenance of Sensitive Data using Homomorphic Encryption

Mazharul Islam
Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
mazharul.islam1@northsouth.edu

Mubasshir Ahmed
Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
mubasshir.ahmed@northsouth.edu

Rajesh Palit
Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
rajesh.palit@northsouth.edu

2023 16th International Conference on Security of Information and Networks (SIN) | 979-8-3503-4321-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/SIN60469.2023.10474775

***Abstract*—In today's data-driven world, the storage and maintenance of sensitive information present significant challenges related to privacy and security. Traditional encryption methods provide a robust solution for data protection at rest and in transit. While these encryption techniques are highly effective at ensuring the confidentiality of data at rest and in transit, they present challenges when performing operations on the encrypted data without decrypting it first. This limitation has significant implications for various applications, especially when privacy and security are paramount, raising concerns about data confidentiality during processing. Homomorphic encryption is a groundbreaking cryptographic technique that addresses this limitation by allowing computations on encrypted data without decryption. This paper examines the foundational principles of homomorphic encryption, its mathematical underpinnings, and its applicability in real-world scenarios. The paper also discusses the practical implementation of homomorphic encryption in sensitive data such as academic student records, including its impact on data storage infrastructure key management, emphasizing its role in protecting data privacy, enabling secure data outsourcing, highlighting how it empowers organizations to harness the value of their data without compromising security. By encrypting sensitive data using homomorphic encryption, we achieve a dual goal: data remains confidential even in a breach, and authorized users can perform computations directly on encrypted data, eliminating the need for decryption and minimizing privacy risks.**

***Index Terms*—ElGamal, homomorphic encryption, privacy, security, student records**

I. Introduction

In an age where data plays a pivotal role in shaping decision-making processes and driving innovation across various sectors, the security and confidentiality of sensitive information have become paramount. Organizations, from healthcare providers and financial institutions to academic institutions, must grapple with securely storing and maintaining sensitive data while still harnessing its value for analytical and operational purposes [1]. Sensitive data encompasses a wide array of information, including personal identifiable information (PII), financial records, medical histories, and academic transcripts, to name a few. Protecting such data from unauthorized access, data breaches, and insider threats is a non-negotiable imperative, often governed by stringent regulatory frameworks and ethical considerations [2]. Yet, these same organizations increasingly seek to leverage the insights latent within this sensitive data for research, analytics, and improving services.

Traditional encryption methods, while effective in securing sensitive data during transmission or when stored at rest, may fall short when it comes to the comprehensive storage and maintenance of sensitive data. These methods typically require data to be decrypted for meaningful use, creating a vulnerability during decryption [3]. Once decrypted, the data becomes susceptible to unauthorized access, compromising its confidentiality and integrity. Moreover, traditional encryption does not offer a practical solution for performing computations or analytics on the encrypted data without exposing it to security risks. This limitation inhibits organizations from harnessing the full potential of their sensitive data for valuable insights and operational purposes.

In contrast, homomorphic encryption [4], an innovative cryptographic approach, enables secure data processing while data remains encrypted, providing a transformative solution that addresses the shortcomings of traditional encryption. This makes it an increasingly attractive option for safeguarding sensitive data in storage and ensuring its security throughout its lifecycle.

This research paper delves into secure data storage and maintenance, specifically focusing on sensitive data. We explore the transformative potential of homomorphic encryption, an innovative cryptographic technique that allows computations on encrypted data without decryption. By enabling secure data processing while data remains encrypted, homomorphic encryption promises a breakthrough in data security without sacrificing data utility.

Our primary objective is to investigate the practical application of homomorphic encryption in storing and maintaining sensitive data. We aim to comprehensively understand how this technique can be employed across various domains to safeguard sensitive information while preserving its integrity and utility. To achieve this, we examine the ElGamal encryption [5] in homomorphic, focus on various literature reviews, choose student academic records as an example of sensitive data, discuss key management strategies, and assess encrypted calculation considerations within secure data storage. This system is designed to meet the following objectives:

- **Data Encryption:** All student academic data, such as transcripts, is encrypted using homomorphic encryption before being stored or transmitted.

- **Secure Data Storage:** Encrypted student records are stored securely in cloud storage, ensuring that even administrators or service providers cannot access the plaintext data without proper authorization.
- **Homomorphic Operations:** Authorized users, administrators, registrars, students, or exam controllers have access to homomorphic encryption keys that allow them to perform specific computations on the encrypted data without decrypting it.
- **Data Processing:** Common operations like calculating grade averages or generating reports can be performed on the encrypted data without revealing sensitive information.
- **Access Control:** Role-based access control mechanisms are implemented to ensure that only authorized personnel can perform specific computations and access certain portions of the encrypted data.

The rest of the paper is organized as follows. Section II focuses on the background of homomorphic encryption and how the calculation computation works on encrypted data, including the ElGamal Encryption procedure. Section III discusses literature reviews on related work of homomorphic encryption in sensitive data. The discussion on how the proposed model works is given in section IV. Section V discusses the implementation of the proposed model on student records data. Limitations of the proposed scheme and related future work are in Section IV with the conclusion.

II. Background

Understanding the background of homomorphic encryption and the ElGamal encryption algorithm is important, as both are critical components of the proposed solution for securing sensitive data. This part discusses how homomorphic encryption and Elgamal encryption can be used to store and maintain sensitive data securely.

A. Homomorphic Encryption

The concept of homomorphic encryption has its roots in the work of Whitfield Diffie and Martin Hellman, who introduced public-key cryptography in the late 1970s [6]. However, the first practical homomorphic encryption scheme, known as the Paillier cryptosystem, was developed by Pascal Paillier in 1999. Homomorphic encryption is a type of encryption that allows computations to be performed on encrypted data without decrypting it first. This makes it possible to perform secure data analysis and machine learning on sensitive data without exposing the data to unauthorized users.

Homomorphic encryption schemes are typically based on complex mathematical concepts, such as lattice cryptography or post-quantum cryptography. These schemes work by transforming the plaintext data into a ciphertext representation that can be operated on algebraically. The result of the operation is then encrypted again and can be decrypted to reveal the result of the operation on the plaintext data.

In this proposed model, we will use this FHE technique to enable the system to compute encrypted data. The registrar is responsible for generating public and private keys to encrypt student records data to be able to perform edit or write operations on the record without knowing the content of the record itself. On the other hand, the students can send requests to the server and get results of the encrypted data. Fig. 1 demonstrates how this system can be used in a secure computation.

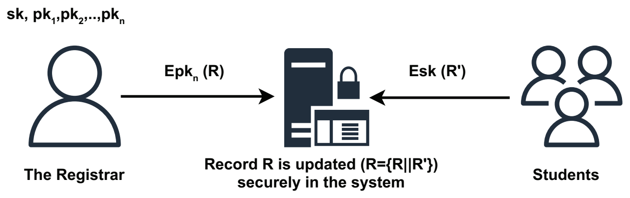

sk, pk_1,pk_2,..,pk_n
Epk_n (R)
Esk (R')
The Registrar
Record R is updated (R={R||R'}) securely in the system
Students

Fig. 1. Store and retrieve data within a secure computation by Homomorphic Encryption

B. ElGamal Encryption

ElGamal encryption was introduced by Taher ElGamal in 1985 as an alternative to RSA encryption [7]. It is based on the Diffie-Hellman key exchange and the discrete logarithm problem.

ElGamal encryption is a widely used public-key encryption algorithm named after its creator, Taher ElGamal. It is based on the mathematical properties of modular exponentiation and the computational difficulty of solving the discrete logarithm problem. ElGamal encryption provides confidentiality and data integrity, making it suitable for securing sensitive data.

III. Literature Review

Gentry [8]] introduced the concept of fully homomorphic encryption (FHE), a revolutionary advancement that allows computations to be performed directly on encrypted data without the need for decryption, thereby preserving data privacy and security. This breakthrough has far-reaching implications for secure data processing, enabling applications in cloud computing, secure outsourcing, and privacy-preserving machine learning. Gentry's FHE scheme, based on lattice-based cryptography, has spurred extensive research and development efforts, creating efficient and practical homomorphic encryption libraries and systems.

Suwandi *et al.* [9] discusses the need for an in-depth analysis of dropout cases in educational institutions to address sensitive issues and ensure proper handling of the cases. The paper highlights the risks of creating a single repository for sensitive information and suggests using homomorphic encryption. It emphasizes the need for data security to protect sensitive information from internal and external attacks and recommends using cryptography to guarantee security.

Hariss *et al.* [10] discusses the implementation of privacy-preserving cloud storage using Homomorphic Encryption. It focuses on a Proof of Concept (POC) scenario of a university using the cloud for storing and operating on sensitive data. The main contribution of the application is the utilization of HE to encrypt different types of data, such as student information, grades, and indexes, enabling computation on the encrypted data on the cloud side. It presents the modified Domingo Ferrer (MDF) encryption scheme as the HE solution for this application. It describes various queries that can be sent from the university to the cloud and highlights the security measures implemented, such as AES encryption, hash functions, and fake student data to prevent statistical attacks.

Yufeng *et al.* [11] proposed a secure and trusted student information management system based on blockchain technology. The system aims to overcome challenges such as information silos and difficulties in data sharing. It incorporates ring signature technology for fine-grained access control and homomorphic encryption to protect sensitive information while allowing operations on encrypted data. Overall, the system offers a solution to the challenges faced in student information management and sharing, providing enhanced security, privacy protection, and efficiency.

IV. The proposed Model

In this section, we describe our proposed model of a comprehensive system proposal that leverages homomorphic encryption to provide end-to-end security for storing and processing sensitive data, such as student records of educational systems. The proposed system addresses this critical gap by integrating homomorphic encryption into the data lifecycle.

A. System Model

We choose student records of an educational system for storage and maintenance using homomorphic encryption for calculation. We consider the academic transcript an example of student records where homomorphic and ElGamal encryption is used for encryption in real-life scenarios. The system design of the Proposed model is shown in Fig. 2. This system comprises the following entities: the registrar, Access control lists, key manager, student records server, cloud storage, students, and exam controller, which are discussed below.

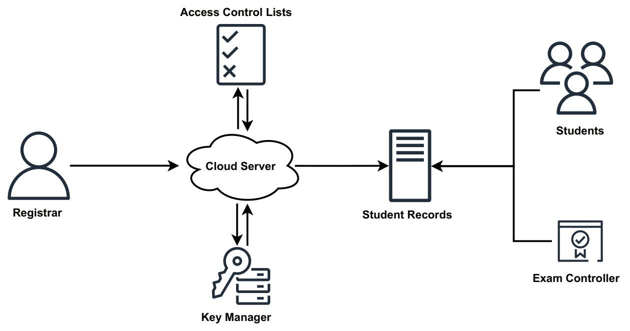


Fig. 2. System model of the proposed scheme.

- **The Registrar:** The registrar can be considered an administrative staff collecting sensitive data from students or other sources, such as personal identification information and academic records. The registrar encrypts the collected data using homomorphic encryption before sending it to storage.
- **Access Control Lists:** It defines and manages access control policies for who can access the encrypted data stored in the student's record through cloud storage. This includes specifying roles, permissions, and authentication mechanisms. It also logs and monitors access to sensitive data to detect any unauthorized access attempts.
- **Key Manager:** It generates, stores, and manages encryption keys required for homomorphic encryption. This includes key generation, rotation, and revocation as needed. It controls the access to encryption keys to ensure that only authorized entities can perform operations on the encrypted data.
- **Student Records Server:** It is a core system of record (SOR) for higher education institutions and supports routine administrative and academic activities. It plays a crucial role in maintaining accurate and secure student records while facilitating access to authorized users.
- **Cloud Storage:** Cloud storage for student records refers to storing student-related data in cloud-based storage solutions, such as academic records, personal information, assignments, and more. It securely stores encrypted sensitive data of student records, ensuring data integrity and availability.
- **Students:** Students are the primary data subjects in the student records. They often must interact with the system to provide or update their personal information. This includes details like contact information, emergency contacts, and changes to their program or major. Students rely on the student records system to access their academic records, including grades, transcripts, class schedules, and degree progress. This information helps them monitor their academic performance. They can request access to their own data or specific operations by their need, which authorized entities process using homomorphic encryption techniques.
- **Exam Controller:** Exam Controller in a Student Record System is pivotal to ensuring the accurate and efficient management of all examination-related information and processes within an educational institution. Their responsibilities encompass various aspects of examination planning, administration, and record-keeping. The Exam Controller ensures that grades and exam scores are accurately recorded and maintained in the student records system. They verify the correctness of grade transcripts.

The proposed system aims to protect the confidentiality and privacy of student records data using homomorphic encryption while enabling authorized parties to perform certain computations on the encrypted data for academic and administrative purposes.

B. Details Explanation

To provide a detailed explanation of the proposed model for the storage and maintenance of sensitive student records data using homomorphic encryption, we'll break it down into key components and steps:

1) Key Generation: Key generation is critical in storing and maintaining the student records system. These keys play a pivotal role in the security and functionality of the encryption process. ElGamal encryption is a public-key cryptosystem that uses a pair of keys: a public key for encrypting each student's transcript and a private key for decryption. With ElGamal encryption, the registrar generates the private and public keys x and y accordingly, as shown in Algorithm 1.

Algorithm 1 Key Generation by the Registrar

```
1: Choose a large prime number, $p$.
2: Select a primitive root modulo $p$ denoted as $g$.
3: Generate a private key, $x$ between 1 and $p-1$.
4: Calculate the public key, $y = g\hat{}x \bmod p$.
```

- **Selection of Parameters**
  (a) The Registrar chooses a large prime number p and a primitive root modulo p, g. These parameters will be used to create both the public and private keys.
  (b) These parameters can be considered system-wide constants and are used by all parties involved.
- **Private Key Generation**
  (a) The Registrar generates a secret key, x, which is a random integer such that $1 <= x < p - 1$. This secret key is kept confidential and should not be shared with anyone.
- **Public Key Derivation**
  (a) The Registrar calculates the corresponding public key, y as $y = g\hat{}x \bmod p$. This public key is part of the system's parameters and can be shared openly with students and exam controllers.

2) Encryption of Student transcript by the Registrar: To encrypt a student transcript, the registrar uses the public key of the student accessing the record. This encrypts the record so that the student and exam controller can only decrypt it with the corresponding private key. The encryption of student transcripts by the Registrar is shown in Algorithm 2.

Algorithm 2 Encryption

```
1: For each plaintext message, $M$, select a random integer $k$ between 1 and $p-2$.
2: Compute the ciphertext as follows:
    a: Compute $c1 = g\hat{}k \bmod p$.
    b: Compute $s = y\hat{}k \bmod p$.
    c: Compute $c2 = (M * s) \bmod p$.
3: The ciphertext consists of $c1$ and $c2$.
```

- **Message Encoding**
  (a) The Registrar wishes to encrypt a student's transcript, represented as a plaintext message M.
- **Key Exchange for This Specific Transcript**
  (a) For each record encryption, the registrar selects a random integer k such that $1 <= k < p - 1$. This random value k is used only for this particular encryption, ensuring that the same record encrypted multiple times produces different cipher texts.
- **Calculation of Temporary Values by the Registrar**
  (a) The Registrar computes two temporary values:
    - $c1 = g\hat{}k \bmod p$.
    - $s = y\hat{}k \bmod p$.
    - $c2 = (M * s) \bmod p$.
- **Ciphertext Generation by the Registrar**
  (a) The ciphertext for the student's transcript consists of the pair ($c1$, $c2$). This ciphertext is securely stored on the cloud storage.

3) Decryption by Authorized Party such as Students and Exam Controllers: The decryption of sensitive student transcript data is performed by authorized users (students or exam controllers) who possess the private key x necessary for decryption. This ensures that data security and privacy are maintained while allowing authorized parties to access and utilize the encrypted transcript as needed. The decryption of student transcript by the Registrar is shown in Algorithm 3.

Algorithm 3 Decryption

```
1: To decrypt the ciphertext ($c1$, $c2$), the recipient uses their private key $x$:
2: Calculate $s = c1\hat{}x \bmod p$.
3: Compute the modular multiplicative inverse of $s$, denoted as $s_{inv}$, such that $s * s_{inv} = 1 \pmod p$.
4: Recover the plaintext message as $M = (c2 * s_{inv}) \bmod p$.
```

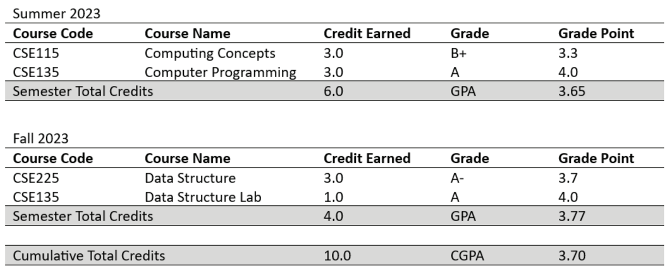
- **Retrieve the Ciphertext**
  (a) An authorized party, such as students or exam controllers, receives the ciphertext ($c1$, $c2$).
- **Shared Secret Calculation**
  (a) The authorized party, having appropriate access to the necessary public parameters, computes a shared secret s as $s = c1\hat{}x \bmod p$. This shared secret is critical for decryption.
- **Decryption by Authorized Party**
  (a) The authorized party decrypts the ciphertext to obtain the original plaintext academic record, M, using the formula: $M = (c2 * s_{inv}) \bmod p$., where $s_{inv} - 1$ represents the modular multiplicative inverse of s modulo p.

The Registrar, who possesses the private key x, can also perform decryption when necessary. However, the Registrar's primary role is in encryption and managing the public parameters, while authorized parties with appropriate permissions can perform decryption to access the student's academic record. Here, ElGamal encryption, with its public-key infrastructure, ensures that multiple parties can securely exchange sensitive student transcripts without sharing their private keys, enhancing data security and privacy in the student record system.

V. Implementation

Implementing the storage and maintenance of student records data using homomorphic encryption involves several steps and considerations. We use ElGamal encryption for homomorphic encryption, adding, multiplying, and dividing the ciphered values. With this, it encrypts two integers, multiplies the ciphered values, adds the calculated values, divides for the actual value in encrypted form, and then decrypts the result. The step-by-step process of how the system works with a proper example is discussed below.

1) Input Data: Using a student transcript to encrypt and store data on cloud storage is an example of data protection and security. A sample transcript is shown in Fig. 3, which explains the implementation details of encryption using ElGamal in Homomorphic encryption.



Summer 2023

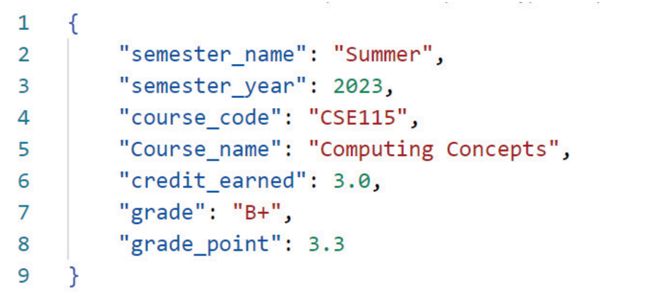
Course Code	Course Name	Credit Earned	Grade	Grade Point
CSE115	Computing Concepts	3.0	B+	3.3
CSE135	Computer Programming	3.0	A	4.0
Semester Total Credits		6.0	GPA	3.65

Fall 2023

Course Code	Course Name	Credit Earned	Grade	Grade Point
CSE225	Data Structure	3.0	A-	3.7
CSE135	Data Structure Lab	1.0	A	4.0
Semester Total Credits		4.0	GPA	3.77
Cumulative Total Credits		10.0	CGPA	3.70

Fig. 3. A sample transcript of a student

The registrar inputs a student's transcript in plaintext form and converts it into JSON format for encryption, as shown in Fig. 4.



```
{
    "semester_name": "Summer",
    "semester_year": 2023,
    "course_code": "CSE115",
    "Course_name": "Computing Concepts",
    "credit_earned": 3.0,
    "grade": "B+",
    "grade_point": 3.3
}
```

Fig. 4. Academic Transcript in JSON Format

2) ASCII Value: The registrar converts the plaintext of the transcript into ASCII form. ASCII encoding is used before encryption to ensure that arbitrary binary data can be processed and encrypted as text. Encoding data as text makes it more interoperable across different systems, protocols, and programming languages. It ensures that data can be transmitted and stored in a standardized way. When data is encoded into a text format, it can be checked for validity more easily. This can help detect and prevent issues caused by invalid or unexpected binary data. To convert the plaintext form of the CSE115 course is considered. The ASCII value of the course Code 'CSE115' and the grade 'B+' is shown in Table I.

TABLE I
Conversion of plaintext into ASCII value

Course Details	Plain text	ASCII Value
Course Code	CSE115	[67, 83, 69, 49, 49, 53]
Grade	B+	[65, 43]

3) Encryption: The register encrypts the ASCII value of the plaintext form by using the student's public key, Pk.

This array has 6 elements. The 1st element represents 'C', 2nd 'S', and so on.

4) Addition and Multiplication in Encrypted Data: Here, it is focused on how the CGPA calculation works in plaintext format, which is:

CGPA Calculate = Sum(Credit Earned * Grade Point) / Total Credits

From Fig. 3, we can see that the student gets a B+ and A grade in the CSE115 and CSE135 courses, respectively. The grade points of B+ and A grade are 3.3 and 4.0. Therefore, the total GPA for the summer semester is (3.0*3.3)+(3.0*4.0)/(3+3) = 3.65. According to the formula of CGPA calculation, firstly, we need to multiply completed (course grade and course credit). In homomorphic multiplication, course code and course credit must be converted into homomorphic form and multiplied. However, there is an issue with the ElGamal Homomorphic Multiplication. It does not give accurate results when multiplying two decimal values in encrypted form. To solve this problem, multiply the grade point by 1000, and then it will generate a round value. It takes two round numbers, then convert that number to homomorphic form and multiply it. The ElGamal homomorphic addition and multiplication of credit earned and grade point use are shown in Fig. 5.

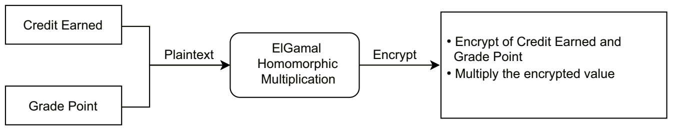

Credit Earned
Grade Point
Plaintext
ElGamal Homomorphic Multiplication
Encrypt
• Encrypt of Credit Earned and Grade Point
• Multiply the encrypted value

Fig. 5. Multiplication of Encrypted data

To get an accurate GPA for the summer semester, it is necessary to divide the addition of (Credit Earned * Grade Point) by total credits in homomorphic encryption. The Elgamal homomorphic division is needed for this calculation. It performs well when it returns a round value result. For example, 20 / 5 = 4, Elgamal homomorphic encryption will return 4; however, when it is 20 / 3, which is 6.66. It does not return accurate results after the point. To get the accurate result of division in encrypted form, it needs to keep the rounds in an array of [x,y] so that it's easier to calculate the division in encrypted form, which fulfills the partially homomorphic encryption requirements.

5) Retrieve Data: The register encrypts the student transcript by an individual public key of students and the exam controller. Note that there might be more than these two entities in the education system that can get access from the registrar. The registrar outsources the encrypted data to the cloud storage, including key manager and access control lists. The students and exam controllers can get the transcript by logging into student records. Here, we separate the student records from cloud storage due to the privacy and security of the encrypted data. The data outsourcing and retrieval process is shown in Fig. 6.

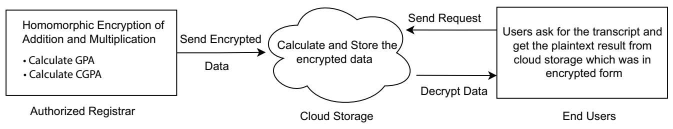

Homomorphic Encryption of Addition and Multiplication
• Calculate GPA
• Calculate CGPA
Authorized Registrar
Send Encrypted Data
Calculate and Store the encrypted data
Cloud Storage
Send Request
Decrypt Data
Users ask for the transcript and get the plaintext result from cloud storage which was in encrypted form
End Users

Fig. 6. Storage and Retrieve Data

6) Overview of the system: The overview of the implementation scheme is shown in Fig. 7 for better understanding. The process of the from the inputs to retrieve data is discussed below.

(a) The registrar takes input by following the transcript of a student.
(b) The registrar converts data into ASCII value.
(c) The register encrypts data by public keys.
(d) The system performs the calculation on encrypted data.
(e) The register stores encrypted data on cloud storage.
(f) The authorized users send requests for encrypted data.
(g) The authorized users decrypted the transcript and got results.

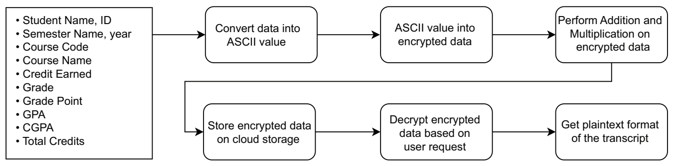

• Student Name, ID
• Semester Name, year
• Course Code
• Course Name
• Credit Earned
• Grade
• Grade Point
• GPA
• CGPA
• Total Credits
Convert data into ASCII value
ASCII value into encrypted data
Perform Addition and Multiplication on encrypted data
Store encrypted data on cloud storage
Decrypt encrypted data based on user request
Get plaintext format of the transcript

Fig. 7. System Design

7) Comparison: The comparison between the existing models, discussed in the literature review section, and our proposed model is shown in Table II.

VI. Conclusion

Using homomorphic encryption, we have proposed a system model for storing and maintaining sensitive data, such as student records. It effectively protects sensitive academic data without compromising the functionality of data storage and retrieval systems. This breakthrough allows educational institutions to store and maintain academic records in the cloud while preserving data privacy and confidentiality. Additionally, it addresses the challenges associated with key management, access control, and calculation of encrypted data in homomorphic encryption. However, it does not perform division operations directly on encrypted data as it returns some garbage value in the round. It is planned to make this system more efficient in handling many requests and implementing homomorphic search, performing all create, update, and delete operations using homomorphic encryption. It is demonstrated that homomorphic encryption holds immense promise for the secure storage and maintenance of academic data. By combining the power of encryption with the convenience of cloud storage, educational institutions can balance data security and accessibility, ultimately enhancing the trust and privacy of academic records in the digital era.

TABLE II
The Comparison of different models on educational systems

Paper Lists	**Data Domain**	**Encryption Type**	**Algorithm**	**Operations performed**
Suwandi et al.[9]	Dropout Data Collection of School	Somewhat	Paillier	Additive
Hariss et al.[10]	Students' indexes and grades	Partially	Domingo Ferrer	Additive and Multiplicative
Yufeng et al.[11]	Student Information Management System	Partially	Paillier	Multiplicative
Proposed Model	Storage and maintenance academic records	Fully	ElGamal	Additive and Multiplicative

References

[1] Ramachandra, M.N., Srinivasa Rao, M., Lai, W.C., Parameshachari, B.D., Ananda Babu, J. and Hemalatha, K.L., 2022. An efficient and secure big data storage in cloud environment by using triple data encryption standard. Big Data and Cognitive Computing, 6(4), p.101.

[2] Chen, N., Chen, B. and Shi, W., 2022, October. A Cross-layer Plausibly Deniable Encryption System for Mobile Devices. In International Conference on Security and Privacy in Communication Systems (pp. 150-169). Cham: Springer Nature Switzerland.

[3] Munjal, K. and Bhatia, R., 2023. A systematic review of homomorphic encryption and its contributions in healthcare industry. Complex and Intelligent Systems, 9(4), pp.3759-3786.

[4] Oladunni, T. and Sharma, S., 2019, September. Homomorphic encryption and data security in the cloud. In Proceedings of 28th International Conference (Vol. 64, pp. 129-138).

[5] Kara, M., Laouid, A., Yagoub, M.A., Euler, R., Medileh, S., Hammoudeh, M., Eleyan, A. and Bounceur, A., 2022. A fully homomorphic encryption based on magic number fragmentation and El-Gamal encryption: Smart healthcare use case. Expert Systems, 39(5), p.e12767.

[6] Kocher, P., 2022. Public Key Cryptography in Computer and Network Security. In Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman (pp. 57-76).

[7] Jintcharadze, E. and Iavich, M., 2020, September. Hybrid implementation of Twofish, AES, ElGamal and RSA cryptosystems. In 2020 IEEE East-West Design and Test Symposium (EWDTS) (pp. 1-5). IEEE.

[8] Gentry, C., 2009. A fully homomorphic encryption scheme. Stanford university.

[9] Suwandi, R. and Wuryandari, A.I., 2022, July. A Safe Approach to Sensitive Dropout Data Collection Systems by Utilizing Homomorphic Encryption. In 2022 International Symposium on Information Technology and Digital Innovation (ISITDI) (pp. 168-171). IEEE.

[10] Hariss, K., Chamoun, M. and Samhat, A.E., 2020, October. Cloud assisted privacy preserving using homomorphic encryption. In 2020 4th Cyber Security in Networking Conference (CSNet) (pp. 1-8). IEEE.

[11] Yufeng, L. and Fei, L., 2023, August. Design of a Student Information Management System Based on Homomorphic Encryption and Blockchain. In 2023 IEEE International Conference on Sensors, Electronics and Computer Engineering (ICSECE) (pp. 1295-1299). IEEE.