National College of Ireland

Implementing Homomorphic Encryption for Privacy-Preserving Cloud Communication in Healthcare Systems

MSc Research Project
M.Sc. in Cyber Security

Rohit Puligadda
Student ID: 23223715.

School of Computing
National College of Ireland

Supervisor: Michael Prior

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: **Rohit Puligadda**

Student ID: **23223715**

Programme: **M.Sc. in Cyber Security** **Year:** **2024-2025**

Module: **Practicum**

Supervisor: **Michael Prior**

Submission Due Date: **12-12-24**

Project Title: **Implementing Homomorphic Encryption for Privacy-Preserving Cloud Communication in Healthcare Systems**

Word Count: **Count…9686**…… **Page** **20**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: **Rohit Puligadda**

Date: **12-12-24**

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Implementing Homomorphic Encryption for Privacy-Preserving Cloud Communication in Healthcare Systems

Rohit Puligadda
23223715.

Abstract

This paper aims to design an encrypted secure system for processing sensitive healthcare information through AES encryption and homomorphic encryption with TenSEAL to authenticate health data on cloud storage platforms including Firebase. The necessity for this research stems from the rising concerns of healthcare information security and risks in addition to the rising obligatory compliance standards across the global such as HIPAA and GDPR. This study assesses effectiveness, security and acceptability of these encryption techniques for actual use in healthcare organizations. Principal conclusions are that AES encryption became rather safe and efficient algorithm for data protection but it has a certain loss in performance especially with the increase of block and key size. Homomorphic encryption though useful in data privacy during data processing adds massive computational cost. Integration with Firebase for the cloud storage was possible and viable due to security and faster data access. Overall, users expressed satisfaction on the system but there is a problem on how to effectively teach users on encryption. Taking into consideration the results of the research, the authors assume that encryption systems can become a useful instrument in the protection of healthcare data, however, to make them efficient and useful for all those healthcare organizations that need protection of sensitive data in real time, further development is still required. In future work, there will be improvements to the techniques of homomorphic encryption, improving the key management, and the implementation of the system on existing healthcare platforms.

1 Introduction

1.1 Background

The use of cloud computing service within the health care has also been realized due to its economical nature in managing data. Cloud technology improvement of healthcare data distribution and transfer improves the scope of the advancement in telehealth, patient-oriented monitoring and diagnosis, and treatment and other aspects of healthcare delivery. Nevertheless, the migration of highly sensitive health care data to the cloud increases concern of security and privacy of such information. Medical data which come under PII and PHI requires adherent privacy standards like the HIPAA in the United States and GDPR in the EU. Protecting such data is the law and the right thing to do as organizations continue to amass data.

Conventional measures for data security help to maintain the confidentiality of generated information during storage or transfer, but they do not work effectively for data protection during processing. Data that is encrypted loses its security the moment it is decrypted for computation and analysis, it can therefore be accessed or stolen by unauthorized parties. To fill these gaps, Homomorphic Encryption (HE) pose a better solution since data can be processed in an encrypted format. The goal of this study

is to propose a healthcare cloud communication system that is functional and effective when implemented in the real world, where patient's information is protected by using Homomorphic Encryption to meet privacy regulation requirement.

1.2 Problem Statement

By this, there is a significance of ensuring that the patient's information is protected due to increased adoption of cloud computing in the healthcare sector. Traditional safeguard techniques provide data security while at rest or in transit, but are impaired when computation must occur. This vulnerability opens up a Pandora box of insecure situations where patient privacy and trust in health care outcome are threatened. Since patient data even contains information about the medical history and diagnostic details of the patient, any unauthorized access to these data increases significant risks for the welfare of the patients and for the trust in and the compliance with the privacy norms of the healthcare practices.

Homomorphic Encryption allows performing operations on the data encrypted thus avoiding its plaintext exposure during computations. However, the integration of HE in healthcare cloud systems is challenging because it poses issues such as the complexity of computation and scalability. HE is computationally expensive and this often causes latency and resource wastage—both aspects that are hated in healthcare where speed is central. This work aims to assess how HE can be applied in cloud health care environment by achieving security as well as performance-oriented services.

1.3 Objectives

The research question of the study is as follows: What are the possibilities and challenges of the use of Homomorphic Encryption for healthcare cloud communication systems to meet the patient privacy and anonymity requirements, as well as avoid violating legal rules and regulations? The specific objectives include:

1. Comparing the existing Homomorphic Encryption schemes in order to understand how well they can be applied to the context of healthcare clouds, with respect to properties such as computational cost and security.
2. Developing and applying an effective cloud communication that incorporates HE for data security and at the same time catering for the data secrecy and fast processing.
3. Assessing how the proposed HE-based system performs and how well it scales, in terms of speed and area, and how it addresses the healthcare industry privacy issues.
4. Exploring methods to improve efficiency for Homomorphic Encryption and addressing the issues which occur when implementing it in healthcare applications, specifically the Cloud.

1.4 Research Question

How can Homomorphic Encryption be practically and efficiently implemented in healthcare cloud communication systems to preserve patient privacy and ensure compliance with healthcare regulations?

1.5 Significance of the Study

The creation of digital healthcare solutions has provided emphasis on the data privacy. Plants in health care motivate breaches and fraud because future funds and reputation can be lost together with the patient trust as well as the quality of the healthcare that is being handed out. An HE based solution for the secure data processing without decryption could significantly minimize privacy implications,

thereby helping the healthcare service providers achieve regulatory standards without unduly delaying their services.

This research makes valuable contributions to the themes of data privacy and mHealth informatics by showing how HE is realistic and effective as a privacy protection mean in cloud-based healthcare settings. By providing insights on the two main areas of performance optimization and regulation, the study provides practical relevance to the health care practitioners, cryptography practitioners and policy makers, who are interested in enhancing on health care secure communication systems.

1.6 Scope of the Study

They restrict the range of this research to the implementation of Homomorphic Encryption in the context of the encryption system in health care cloud. It entails designing, implementing and assessing a represented cloud-based communication system that would allow for secure processing of encrypted data. Another feature of the study involves a combination of performance and compliance checks that are used to confirm the practical application of the system in healthcare organizations. This research does not include other types of encryptions except Homomorphic Encryption and does not delve further into healthcare use cases that occur in cloud settings. Evaluation will include the use of random generated datasets and the MIMIC-III dataset, and not include real patients to incur a moral utilitarian perspective.

1.7 Structure of the Thesis

The thesis is structured as follows:

1. **Chapter One: Introduction** – Outlines the research background, problem statement, objectives, research question, significance, scope, and relevant definitions.
2. **Chapter Two: Literature Review** – Analyses existing research on cloud security, Homomorphic Encryption, and privacy preservation in healthcare, identifying gaps in current knowledge.
3. **Chapter Three: Methodology** – Details the research design, tools, and procedures, including Python Django for system implementation, and describes the development and testing steps for the HE-based communication system.
4. **Chapter Four: Design And Specification** – Presents Designs used to develop the HE system
5. **Chapter Five: Implementation** – Presents steps used to develop the HE system
6. **Chapter Six: Results and Discussion** – Presents findings from system implementation, including performance metrics, and discusses their implications for the research question.
7. **Chapter Seven: Conclusion and Recommendations** – Summarizes findings, highlights study limitations, and offers recommendations for future research and HE applications in healthcare.

2 Related Work

2.1 Introduction

Today's enhanced use of cloud computing services in the health care sector has made data privacy and protection a critical issue particularly in handling patient's identification data. Homomorphic Encryption (HE) has been considered in research regarding cloud computing as a solution that is able to preserve data security while performing computation on data in its encrypted state. This chapter overviews the emergence and integration of HE into clouds through a healthcare lens and presents prior research on the constraints, innovations, and real-world implementation factors of HE. Some of

the topics include: conventional cryptographic techniques, roles of HE in healthcare, the issues of performance and scalability in healthcare data encryption and possible legislations governing data privacy and encryption in healthcare organizations.

2.2 Cloud Security and Privacy in Healthcare

Cloud computing is an effective technology in sharing, storing and processing of large volumes of information especially in the realm of health care with various disadvantages in its ability to protect health information. Several papers stress the importance of the privacy and security measures that do not let patients' data be stolen. Symmetric and asymmetric encryption methods are used to secure data at rest, as well as in motion; however, they are insufficient at the processing stage. Rindell et al. (2020) have established that decryption for computation makes data sensitive and opens up space for a breach. Privacy infringement can cost a company quite a lot and can be of a legal and ethical nature given the rules governing data privacy across the world and at the regional level including HIPAA in the U.S. and GDPR in Europe. In order to overcome these issues various authors have investigated encryption in use approaches that guard information during processing. Traditional approaches of data security like encryption and data security at rest and during its transmission are not adequate since they do not assure end user data protection. Homomorphic Encryption is useful under this aspect because it enables the data to be in an encrypted form during the processing, which has an added advantage of enhancing security. This section will discuss how HE can reduce the privacy risk in cloud-based healthcare while being compliant with the rules.

2.3 Homomorphic Encryption: Theory and Evolution

Homomorphic Encryption (HE) is a cryptographic technique allowing the computation results that have been obtained directly on encrypted data. Developed by Gentry (2009), FHE expanded the horizon of future cryptographic exploration by allowing such computations on ciphertexts without entering the clear notation of plaintexts. Such foundational work paved way for other HE schemes like Partial Homomorphic Encryption (PHE) scheme which allows some operations on ciphertexts, Somewhat Homomorphic Encryption (SHE) which endorse more functionalities on the encrypted data. Many progresses have been made since Gentry's work and the efficiency and applicability have been enhanced in HE. Some of these developments are CKKS scheme (Cheon-Kim-Kim-Song) that can perform approximate operations on encrypted data good for machine learning and data analysis. Realized the CKKS scheme that has been discussed by Behera and Prathuri (2024) and pointed out that though the researchers have found this the CKKS scheme shows additional advancement than the previous FHE schemes, it is still time-consuming for the large amount of data used in healthcare. The complexity of HE schemes has increased with the development of several schemes, each having more or less advantages and disadvantages when applied in practice mostly within the healthcare ecosystem that requires data privacy while at the same time being feasible for computation.

2.4 Homomorphic Encryption in Healthcare: Applications and Challenges

Healthcare industry demands safe execution of data particularly to applications such as Electronic Health Records, diagnostics and analysis of risks, and predictive modeling. HE has been found to enable probabilistic computations that enable the healthcare providers to process the patient data securely as it remains encrypted. Jeyaraman et al. (2023) described the results of a study on the applicability of HE for healthcare implementation and concluded that although the results seemed encouraging, heist, especially due to its high computational requirements which form the biggest hurdle in its implementation. Since healthcare data involves different records, images and diagnostic

results encrypted, it requires more processing power to analyse, which decreases the system response and scalability.

Some of the previous research has explored ways of improving HE for use in the health sector through either approximate calculation techniques or combining different encryption types. For example, Vizitiu et al. (2020) was focused on usage of the CKKS scheme in the healthcare domain and concluded that it is suitable for machine learning tasks which do not strictly need accurate values. They added that CKKS is less accurate and may not work well for applications that require certain precision in computation such as billing or accurate diagnosis. This renders a major drawback working with HE in the healthcare setting with the system designers focusing on the performance. Another is how HE is integrated with cloud platforms. It is evidenced by the integration of EHRs with cloud storage to ensure that healthcare obtain promptly updated data which cannot experience extended delays. The computational needs of HE, however, may introduce latency that limits its usefulness in many clinical situations where speed is of essence.

2.5 Performance and Scalability of Homomorphic Encryption

HE in healthcare applications has one major disadvantage and that is computational complexity which may limit the practical applicability of algorithm, scalability. Conventional HE schemes, especially FHE, demand a great amount of computation because homomorphic operations are intricate. Märtin et al. (n.d.) posited that to solve these problems, there are performance enhancements, including parallelism and dual encryption models. They showed that applying multi-core processors could pave the way for cutting the duration for the execution of HE, however, they pointed out that such configurations are not necessarily always available in healthcare environments since computational assets may be scarce.

There are several HE optimization strategies that lie in the use of the additional methods together with homomorphic encryption, including symmetric encryption, that improve HE efficiency. In Zhang and Wang (2024), the authors proposed a hybrid approach that employs HE for the data privacy aspect, and symmetric encryption for the remaining computations.

That study pointed out that this approach was helpful in getting over what they call the computational overhead which makes HE more reasonable for real time uses. Nevertheless, the security characteristics of hybrid methods are often less than these of pure HE since they include the vulnerability of symmetric encryption systems. Thus, the question of whether security needs must be sacrificed to achieve high levels of efficiency in HE remains a hot issue in healthcare.

Another thing that defines the scalability of HE in healthcare is the ability to fine-tune algorithms to work with big data. HE algorithms require capacity to input large volumes of medical data that are most of the time intricate. Some of the researchers recommend a selective use of HE wherein data that is most sensitive and confidential is encrypted using HE while other data are encrypted using less computation-intensive encryption methods.

2.6 Regulation Compliance and Homomorphic Encryption

In healthcare data, an important issue of compliance with data privacy regulations was observed. Some legal requirements like the HIPAA and GDPR require use of security measures like encryption of information, control of access, and the creation of audit trails. For example, healthcare data: must meet the provision of the healthcare insurance portability and accountability act; must only be accessed by legitimate users; and must be protected from invasions of privacy. In ToS, Basil et al. (2022) evaluated applicability of HE in healthcare and provided that although HE guarantees strong privacy, its computational cost may leave a footprint on the system, which is essential to support real-time healthcare services for compliance. Similarly, Chauhan and Shiaeles (2023) discussed how seven

out of ten GDPR principles can be met with HE solutions for cloud-based systems, including data minimization and pseudonymization. But they pointed out that HE is computationally intensive, which is a problem with regard to accessibility and ease of use. The dilemma of addressing HE implementation in a manner where regulatory issues are met without comprising the systems' performance is the next concern in addressing secure healthcare cloud environment.

2.7 Homomorphic Encryption Optimization for Healthcare

Current studies have guided efforts to improve the efficiency of the HE process in healthcare specifically through efforts at minimizing the amount of computation required and increasing the ease-of-use of the model. For example, Zhang and Wang (2024) presented a method that incorporated HE with symmetric encryption making the processing speed far better than the previous methodologies used in healthcare domain. But they agreed that this may decrease some security advantages of pure HE as hybrid models could be more sensitive to attacks on the symmetric encryption part. The second possible optimization technique is related to parallel processing to avoid minimizing operations on different processors in parallel. n Märtin et al, (n.d.) found that parallel processing could enhance HE performance to a level that is suitable for application on large health care databases. Yet, this method calls for specific hardware arrangements that might not be available in many health facilities thus its applicability is restricted.

2.8 Limitations of Existing Literature and Research and Implication of this Study

Although a substantial number of studies have examined, HE, there is still a lack of knowledge regarding how such theory can be effectively implemented in actual health care environments. First, the majority of publications are devoted to presenting theoretical estimations of its performance, and only a limited number of works investigates the operation of HE in the context of healthcare activities. Second, most of the current research mainly focuses on achieving high efficiency for general HE in data processing, ignoring the characteristics of the healthcare data, which are confidential and restrictive in the application process. Furthermore, there is insufficiency of information concerning optimization with relation to performance, security and regulatory compliance in the development of HE solutions. Healthcare needs accurate real time data processing platform which should also be compliant to HIPAA and GDPR. This paper seeks to fill these gaps by proposing a feasible cloud communication system design using HE that meets both the performance and privacy needs of the cloud application without violating legal standards.

3 Research Methodology

This section of the chapter discusses the approach followed in developing, deploying and assessing the encryption-based system for protecting patient's sensitive data within a health care application. The methodology adheres to an assembled procedure that involves phases including data acquisition, encryption of gathered data, putting in place the system, testing and finally the evaluation phase. A set of current cryptographic method and the advanced cloud technology is adopted to make sure that the system is efficient and secure.

3.1 Research Design

In the current study, an action research design approach that includes theoretical and applied components will be used. The most time-conscious part belongs to the theoretical section, concerning the identification of literature concerned with encryption methods, cloud storage, and health care

systems security. The practical part involves creating and implementing encryption and making the actual design of the developed system; incorporating AES encryption, Firebase, and TenSEAL homomorphic encryption.

The approach is structured into the following key phases:

- Literature Review and Analysis: An extensive review of related work to identify the current state-of-the-art and available approaches to encryption of the sensitive health data.
- System Design and Architecture: Proposal for the development of a large-scale encryption using AES & Firebase Storage and TenSEAL.
- System Development: The use of the designed encryption system utilizing AES encryption, Firebase and homomorphic encryption system in a Python and Django context.
- Testing and Evaluation: The simulation of the system and its running involving assessment in respect of parameters and criteria for success.
- Data Security Assessment: Evaluation of the system in terms of the likelihood of patient information being stolen and the patient's data privacy regulation compliance.

3.2 Data Collecting and Data Cleaning

Data collection involves obtaining secret patients' information which ought to be encoded when stored or transmitted. The key data types include:

- Patient Personal Information: For example, name, age, medical record, and so on.
- Medical Records and Diagnoses: Personally identifiable information like the patient's diagnosis, test, and treatment.
- Uploaded Files: Patient records including medical records, records and images scans and records.

In order to better protect the data privacy, all the information related with the patient's identity and his/her medical history are considered sensitive and are encrypted accordingly.

3.2.1 Preprocessing steps

- ✓ Data Validation: Data input is checked for accuracy and any data that is eventually input into the system is tested to check that there is no missing or irrelevant information.
- ✓ File Encryption: Files that are uploaded are first encrypted using Advanced Encryption Standard before being uploaded in Firebase Storage.
- ✓ Sensitive Field Encryption: The diagnoses of the non-file patient data are homomorphically encrypted with TenSEAL.

The TenSEAL library is used to compute related operations on encrypted data since it is a necessary approach to meet modern privacy-conscious goals.

3.3 System Development

3.3.1 Choice of Encryption Algorithms:

- **AES (Advanced Encryption Standard)**: AES is used for file encryption because it has industry recognition for being a reliable and fast algorithm. AES using key of 256 bits work in CBC mode making the encryption policy and the data secured and protected from tampering. The encryption keys are not the same for every file, they are issued randomly for every file that needs to be encrypted.
- **TenSEAL**: Ten SEAL is deployed to implement homomorphic encryption techniques on identifiable patient data, including diagnosis. Homomorphic encryption enables computing of

values on encrypted data with no wishful disclosure of the original plain text. This guarantees that the data that would have been worked on will not be exposed even as they are worked on.

3.3.2 Integration with Firebase

- ❖ Firebase Storage: Firebase is chosen for managing patients' information in the cloud in a secure manner. That is why, most of the Firebase services like Firebase
- ❖ Firebase SDK Integration: firebase_admin SDK is employed to work with Firebase Storage to upload encrypted files and to get the URLs of the files.
- ❖ File Upload Process: When encrypted, the file is then transferred to the Firebase Storage. Each file is assigned a special number, while a link is provided for the access to the file.
- ❖ Django Application for Data Management: As for the interface of the given system, the Django web application is going to be used as the tool for further interaction with users.
- ❖ Patient Record CRUD Operations: In addition, users can interact with this system through creation, reading, updating, and deleting records of a patient. Such fields are encrypted before they are stored in the database and can therefore include sensitive records.
- ❖ File Uploads and Decryption: The application is designed with an interface that allows the user to input files for upload, in which case once the input is submitted, the files are then encrypted and then stored in Firebase. People with permission can also unencrypt files for viewing only.
- ❖ TenSEAL Integration: Ciphertext of patient data such as diagnosis is processed using TenSEAL for storage in the system.

3.4 Security Considerations:

- ✓ Key Management: AES encryption key and IV are kept secured and because of this proper key management principle are followed so that keys are not revealed.
- ✓ Authentication and Authorization: Firebase Authentication is used for regulating the user access to the system and is made to allow only an authorized user to upload or view/decrypt patient data.
- ✓ Audit Trails: Patient records are accessed through the system to monitor and review compliance with privacy legislation and to record who accessed the patient data.

3.5 Testing and Evaluation

The assessment is critical in the methodology to guarantee reliability, security features and functionality of the system. The testing process involves several stages:

- — Unit Testing: Testing of every element of the system is done separately: encryption algorithms and file uploading functionality; CRUD operations database. This Is to help make sure that each module runs the way it is want to.
- — Integration Testing: After sub-components of a given system are evaluated, an integrated approach is conducted with an aim of evaluating the systems. This involves confirming that encrypted files are correctly uploaded to Firebase, or that the decryption process goes smoothly, or the various data management capacities of the Django application do.
- — Security Testing: Security tests are conducted on the system to determine the extent of security on the data. Penetration testing is done in order to assess possible risks. Some of the tests are to detect vulnerabilities in encryption, authentication, and key management.
- — Performance Testing: Stress testing focuses on identifying the system behavior under different loads, like having many files uploaded at once or a number of large files.

Performance measures are the uploaded and downloaded encrypted files while the encryption and decryption time spent is also considered.

3.6 Evaluation Criteria

The evaluation of the system's effectiveness is based on several criteria:

- Confidentiality: How the system has used AES and TenSEAL to ensure that patient's data is safe and secure.
- Integrity: Security of the information from the time it is entered, while in-transit or when stored at the database.
- Availability: Ability to make the patient information available to the relevant users at the right time.
- Scalability: The efficiency of inter alia the amount of data that the system can process as well as the number of users that can the system accommodate without posing efficiency challenges to the system efficiency.
- Usability: The convenience offered by the system, that is, flexibility in the ability of health care workers to upload as well as manage files and the ability to decrypt data.

3.7 Ethical Considerations

- Informed Consent: Prescribers and patients must know how their data is going to be utilised and processed.
- Privacy: The system provides an assurance that only the right persons can get access to the patient's details. Security considerations apply across the whole data lifecycle in relation to the use of encryption to protect patient information.
- Compliance with Regulations: The system is developed being mindful of data protection laws (HIPAA, GDPR) to guarantee a proper deal with patient info.

Thus, the presented above methodology enables the effective conjunction of classic cryptographic methods, safe storage of patients' data in the cloud, and contemporary approaches to data management to provide protection of sensitive patient information. Confidentiality integrity availability and usability criteria are used to assess performance of the system whereby patient data is protected at the same time the information is available to the right users.

4 Design Specification

4.1 System Architecture

The architecture of the solution follows a modular approach, utilizing cloud-based storage (Firebase) for secure file uploads and decryption. The design of the system can be divided into four major components:

Encryption Module: The primary function of this component is to properly encrypt the patient's information before it can be uploaded in Firebase. The encryption ensures that sensitive information like the diagnosis of a particular patient is not accessed by other individuals in Firebase storage in the event that the database is accessed by an intruder. The file data is encrypted using AES (Advanced Encryption Standard) and the required AES implementation in CBC mode is provided by the cryptographic library PyCryptodome.

Firebase Storage Integration: Firebase Storage is preferred due to the simplicity it offers, its flexibility and its security. This component will serve to write the encrypted data onto Firebase where

it will be retrievable via a public URL. The storage integration process is done by the firebase_admin SDK, which is engaged to the Firebase storage bucket. Firebase offers strong security features like, user authentication and file permission which are used for allowing access to uploaded files.

Patient Record Management: This component is involved in record keeping of the patient in a web application using Django technology. It enables creation, viewing, updating, and deleting patient information and encrypts fields like diagnosis using TenSEAL. TenSEAL is an open-source library based on homomorphic encryption that enables users to work on obscured data without the actual decoding. The patient data is stored in a relational database and the important fields of the database are encrypted hence intruders initially gain access to the patient data but won't be able to access the sensitive information.

Web Interface (Django Application): For uploading files, patients record and for viewing encrypted data, there is a web interface developed with the help of Django. It offers the user-friendly environment where there is a form for uploading files and areas where the patient information should be filled in.

4.2 Data Flow and Workflow

Patient Data Collection: Patient details are entered by a user through the web interface, perhaps a healthcare professional. Additionally, one may provide name, age, diagnosis, any kind of file data, for example medical records or images. The patient data is transferred in JSON messages by means of HTTP request/response.

Encryption of Patient Data: After the patient data is submitted loading completes, the system first check for file upload. If a file exists the data is written to a temporary folder on the server. The file is then encrypted using AES encryption with a randomly generated key and IV (Initialization Vector). The encrypted file should be ready for upload to Firebase storage environment. The system then gets a public URL for the uploaded file and along with the encryption key and IV to use in the future to decrypt the file. In the case of the diagnosis field, it is encrypted using the TenSEAL library to preserve the possibility of performing computations on encrypted data without decrypting it. This is beneficial in ensuring that while some forms of information are kept secret, they are still affecting some operations.

Uploading to Firebase Storage: Subsequently the encrypted file is then uploaded to firebase storage using the firebase_admin SDK. This involves comes to the reference of the storage bucket of firebase storage then uploading of the file and at times public accessibility of the file where necessary. After the file has been uploaded, its URL is passed on to the user for their use.

Database Storage of Patient Data: In addition to file data encryption, any other patient information including the encrypted diagnosis is stored in the Django database. The encryption of these fields makes it impossible even if the database is leaked to access the data due to encryption keys.

*Access and Decryption***:** When a healthcare worker or authorized personnel wishes to access the patient record, they can retrieve the encrypted file and the associated keys (key and IV). Using the keys, the encrypted file is decrypted to restore the original content. Similarly, the encrypted diagnosis field is decrypted using TenSEAL to retrieve the original data.

4.3 Module Breakdown

To ensure the system is modular, each major function is divided into separate components, following object-oriented principles for scalability and maintenance.

Encryption Module: In file encryption, the encryption module employs AES while TenSEAL is used for the encryption of the diagnosis field. AES guarantees that the file data is encrypted using a 256 key, making it difficult for the key to be cracked by any unauthorized third party. For encryption, the system uses randomly produced a 256-bit key and a 128 bit IV for each encryption creating a distinct encrypted file form of the same data.

Firebase Integration: Firebase Storage module handles all processes related to Firebase Storage system, such as uploading files, setting metadata, and setting access level. This information is the name of the uploaded file and is passed to the Firebase storage bucket. After the file has been uploaded, the user receives a URL to the said file for better convenience. Functionality used to limit access rights to the file embedded in Firebase environment, such as Firebase Authentication.

Django Application and API Endpoints: This social site application built using Django has an api through which it is possible to communicate with the system. These endpoints are described leveraging Django REST Framework paradigm. Patient data (diagnosis and file) is captured from the patient by making a POST request to API and the system sends back a confirmation message together with the URL of the uploaded file. The system also delivers endpoints to query patient record data, modify database records, and to delete records too.

PatientRecordViewSet is used for creating, viewing, updating and deleting the patient records. upload_file_view is responsible for accepting a file upload, encrypt and upload to Firebase Storage.

Decryption: Encryption is done when only there is a need for an exclusive group of people to view the data or information encrypted is only decrypted when needed. The AES key and IV are used to decrypt the file content and used also the TenSEAL context to decrypt a diagnosis. The decryption process is handled by separate methods within the PatientRecord model, ensuring that only authorized users can decrypt sensitive patient information.

4.4 Security and Performance Considerations

4.4.1 Security Measures:

- **Encryption:** The use of AES-256 and homomorphic encryption (TenSEAL) ensures strong confidentiality for sensitive data.
- **Firebase Security Rules:** Firebase provides powerful tools for controlling access to the storage bucket. Firebase security rules are configured to restrict access to only authorized users, such as medical professionals with appropriate permissions.
- **Key Management:** The AES encryption keys and IVs are stored securely and are necessary for decrypting the file. Proper key management practices are followed to ensure these keys are not exposed unnecessarily.

- **Data Validation and Integrity:** Before uploading the data to Firebase, validation checks ensure that the data is well-formed and encrypted. The integrity of the data is checked by comparing the hash of the encrypted data.

4.4.2 Performance Considerations:

- **Scalability:** Firebase Storage provides scalable storage for potentially large files. The system is designed to scale with the increasing number of patient records and file uploads.

- **Efficiency:** Encryption is performed using optimized libraries (e.g., PyCryptodome for AES and TenSEAL for homomorphic encryption), ensuring that the performance of the system is not compromised, even when processing large datasets.

4.5 User Interface Design

The user interface of the application developed through Django framework is friendly, easy to access. There are sections for submitting patient data and files and brief guidelines and error checks to indicate that this is a form used to submit tabular data and files of various forms. It gives information on the state of the file uploads and if the encryption or the upload failed, it displays an error message. The front-end is optimized to work on mobile devices and any other devices possible to help create enhanced user experience.

4.6 Future Enhancements

- ✓ *Automated Decryption*: Deploying an automatic system of key distribution to those who are allowed to decode the files with patient data.

- ✓ *Multi-Factor Authentication (MFA*): Under the security enhancements, MFA can be combined with the Firebase Authentication system.

- ✓ *Data Analytics*: Introducing data analytics for getting some insights of patient and then collect those data in an encrypted form so that patient privacy could be protected.

Finally, the system exhibits an excellent architecture for achieving efficient encryption, storage, and management of data as individual modules of design that can expand or shrink with the system's rate of growth. That way, patient data which maybe of a sensitive nature to the healthcare provider, patient or third party, is kept secure and there is always a functional, easy to navigate interface available for use by medical practitioners.

5 Implementation

In this section, we will demonstrate how the final solution will look like by giving the details of how the different design specifications will be put into practice. This section will state the major components, instrumentation, and resources used, how they were used and what was achieved with this system. The implementation is concerned with a development of an encrypted file uploading and storage in Firebase along with secure patient data handling and encryption.

5.1 Tools and Technologies Used

In order to satisfy the specified criteria for storing patient records as well as their related files, the proposed solution had to be developed using a set of technologies and libraries. The primary tools and languages used include:

- ✓ Python: Considered as the primary language for development at the backend of the website The key responsibility of this language is to perform the encryption process, integrate with Firebase, and manage the file operation system.
- ✓ Django: A level Python web framework that has been employed in the development of the backend system for patient record management and file handling.
- ✓ Firebase: It refers to the online platform through which files can be stored and accessed. Firebase Storage was used to upload files in a secure way and encrypt them.
- ✓ Cryptography Library: A Python library that can be used to implement AES encryption with the view of providing maximum security to data before it is uploaded to Firebase Storage.
- ✓ TenSEAL: An open source and portable library for machine learning used in encryption of patient information so that computation can be safely done on encrypted data.
- ✓ HTML, and CSS: For the front-end design to interface with the backend it as used for uploading files and displaying the results.

5.2 General System Operations

The last step of the system implementation is focused on the safe use of patient information and patient's records or any related files. The system has several steps, which are the major steps to ensure that data is safe and has not been tampered with, through encryption, and is kept in the cloud. The workflow can be broken down into the following major steps:

Patient Data Creation: The all-patient details are given by the user – name, age, diagnosis, and any files that should be upload (for example, medical reports). This data is initially handled by Django forms and is then validated.

File Encryption: If the files to be given as input are from the user (for example, medical records), then they must be encrypted with AES. AES encryption is used in the CBC mode, in successively using a randomly generated 256-bit encryption key and 128-bit initialization vector for each file. This makes sure that each file has its own encryption code as well as protect files from being accessed by other people.

Uploading Files to Firebase: Instead, the content of the file is encrypted before being uploaded to Firebase Storage, in which they are stored in an encrypted format. Its accessibility is also made public (where necessary) to allow the authorized persons to easily access the file. Upload process also provides the URL link for file access and the encryption key IV for decryption of the file.

Patient Record Encryption: Besides, the TenSEAL library is used to encrypt the patient's records (e.g. diagnosis). The diagnosis text is transformed into a CKKS encrypted vector for computations on encrypted data. Such data can be stored in the database securely in encrypted form, and later can be decrypted for use in later stages.

Decryption of Files and Data: When authorized users need access to the patient's data, the system decrypts the encrypted files and records using the stored encryption keys and IVs. The patient's medical data can be decrypted from the CKKS vector format, allowing it to be used in medical decision-making. The decrypted files are retrieved from Firebase using their public URLs.

5.3 Backend Implementation

5.3.1 Data Models

The main data models in the system revolve around the PatientRecord data model that contains basic identifying information of the patient and also name and age, diagnosis as well as the file data of the patient. These records are kept in a relational database-management system like PostgreSQL or SQLite based on the deployment option.

PatientRecord Model: It is a model created in Django with fields that correspond to patient identification number (patient_id), patient's name, age and diagnosis as well as the file itself. The model is also capable of handling the encryption of the fields before storing the information like the diagnosis.

```
class PatientRecord(models.Model):
    patient_id = models.CharField(max_length=20, unique=True)
    name = models.CharField(max_length=100)
    age = models.IntegerField()
    diagnosis = models.TextField()
    file_data = models.FileField(upload_to ='media/')

    def encrypt_data(self):
        # Generate encryption context
        context = ts.context(
            ts.SCHEME_TYPE.CKKS, poly_modulus_degree=8192, coeff_mod_bit_sizes=[60, 40, 40, 60]
        )
        context.global_scale = 2**40
        context.generate_galois_keys()
        context.generate_relin_keys()

        # Encrypt diagnosis field
        encrypted_diagnosis = ts.ckks_vector(context, [ord(char) for char in self.diagnosis])
        return encrypted_diagnosis.serialize()

    def decrypt_data(self, encrypted_diagnosis):
        context = ts.context(
            ts.SCHEME_TYPE.CKKS, poly_modulus_degree=8192, coeff_mod_bit_sizes=[60, 40, 40, 60]
        )
        decrypted_vector = ts.ckks_vector_from(context, encrypted_diagnosis)
        return ''.join(chr(int(value)) for value in decrypted_vector.decrypt())
```

Encryption Methods: Encrypting of data (like the diagnosis) is done by the encrypt_data() method of the PatientRecord model. This encryption makes use of the TenSEAL library which is used in implementing secure computation based on homomorphic encryption. The method also encrypts the diagnosis giving it a CKKS vector for safe storage.

5.3.2 File Encryption and Upload Procedure

Managing the file encryption and uploading, there is a function named upload_file_to_firebase() which is placed in the firebase_utils.py. The content of the file is encrypted using AES algorithm in CBC mode before uploading the file to Firebase.

The steps involved are as follows:

Generate Key and IV: Each physical file is encrypted using a new key and initialization vector to meet the unique encryption of each file.

Read File Content: The content of the file is read in the binary format and that would include PDFs and images, other than true text files.

AES Encryption: The AES cipher is then loaded with the Key and IV that has been generated. The content of the file is Adjusted to an even multiple of the block size that is characteristic of AES. The padded content is then encrypted cipher text through the Cipher Pad Algorithm in mobile devices.
Upload to Firebase: The encrypted file is uploaded to Firebase Storage using the Firebase SDK, and a public URL is generated for accessing the file. The key and IV used for encryption are returned along with the URL.

5.3.3 API and Views

Creation of the API used to handle the patient record creation and file uploads was done using Django rest framework. Particularly, PatientRecordViewSet is used, enabling user to perform POST request to create patient records, and GET/DELETE requests to view and remove records respectively. The key aspects of the API implementation include:
Create Patient Records: For creating the patient records, the create() method of the PatientRecordViewSet class deals with POST request. It expects a Json payload that has data of the patient such as medical records and file data. The input file is stored, encrypted, and uploaded to Firebase storage briefly before the URL and encryption information are given.
File Upload: The upload_file_view() function is used for the management of file upload through a Django form. After the form is validated the script checks that the file is valid and then proceeds to encrypt and upload it by using the functions upload_file_to_firebase().

```
def upload_file_to_firebase(local_file_path, firebase_file_name):
    # Encrypt the file content
    encrypted_data, key, iv = encrypt_file_content(local_file_path)

    # Reference to Firebase Storage
    bucket = storage.bucket()
    blob = bucket.blob(firebase_file_name)

    # Upload encrypted file data to Firebase
    blob.upload_from_string(encrypted_data)

    # Return the file's public URL and encryption key/IV for decryption
    blob.make_public()  # Optional: make public if necessary
    return blob.public_url, key, iv
```

5.4 Frontend Implementation

The frontend implementation gives the user a graphical user interface in the form of a web page for file uploading and record creation for patients. Unlike most of this project's interfaces, this is implemented using HTML forms, with integration into Django templates. The key components of the frontend are:

- ✓ File Upload Form: An interface with fields that allow a user to choose a file and then upload it after it has been encrypted. The form also enables the entry of extra data, for instance, the name of the patient, and some other details regarding the patient.

Upload File to Firebase Storage

Select a file to upload: Choose File No file chosen

Upload

- ✓ Upload Success Page: On successful upload of a file, the users are redirected to another page where they see the public URL of the file as well as the encryption key and IV as hexadecimal value. This lets those who need access to the file itself be able to recall and decrypt the file if needed.

File Uploaded Successfully!

View your file: https://storage.googleapis.com/rohit-4fc41.appspot.com/uploads/abstract.docx

Encryption Key: ab73202dfddfca05fc27db853bb5198c815012c2a41dfee87c1c646f0585b649

IV: 2d738ecc74b2f696d8eee5574bb32bd9

5.5 Security Considerations

Security is a major concern in the implementation, and various measures have been taken to ensure that both patient data and files are protected:

- ➢ AES Encryption: Use of AES encryption in CBC mode with randomly generated keys and IVs made certain that files are encrypted prior to upload. The fact that different files have their key and IV assures that there will be minimal leakage of data.
- ➢ TenSEAL Encryption: By employing the TenSEAL solution for Patient Record encryption, there is never any medical information in any manner transmitted or stored in plain text. This form of encryption allows computations to be made directly effectively from encrypted data thus improving security.
- ➢ Firebase Storage: Firebase storage for file storage comes with strong security properties such as access to control and authentication.

5.6 Challenges and Solutions

During the implementation, several challenges were encountered and successfully addressed:
Handling Large Files: One of the main difficulties was to work with the big patient files for upload and encrypting. To counter this, files were processed in batches and stored in a separate system unique temporary directory for some time before being uploaded to Firebase.
Encryption Overhead: AES encryption and decryption for large files especially is a task that consumes a lot of computational power. To control performance problems, file sizes were watched and optimizations are done to handle smaller files first in order to increase scalability of the whole system.

6 Evaluation

In this section, the outcomes yielded from the experimentation and case cases studies performed in the course of the study are comprehensively discussed. The analysis of these aspects is performed in order to gather an idea on the efficiency of the encryption-based system for securing sensitive healthcare data and in an effort to provide insights from an academic and practitioner standpoint. This study only provides both the qualitative and quantitative greatest findings relevant to the research question and objectives only. Instruments with statistical properties are employed to analyze the results rigorously in order to provide evidence that is comprehensible, meaningful, and practical.

6.1 AES Encryption Performance

The first experiment looks at the efficiency of the AES encryption system at protecting patient information. The actual experiment in this case was done by implementing AES algorithm and then comparing the performance of the algorithm when used with different key sizes, namely 128-bit, 192-bit and 256-bit. Moreover, the experiment introduced files into the firebase storage, and it assessed how well the system performs with large files. The time taken to encrypt a file of 10MB was also timed for various key sizes. The above experiments revealed that encryption with the 128-bit AES key consumed the shortest time of about fifteen (15) seconds when compared to the two hundred and fifty-six (256) AES key that took nearly thirty (30) seconds for a single file. The decryption times also resembled each other with the factor of time being slightly more with 256-bit key decryption of files due to the extra level of the key.

Impact on Storage: The encrypted files were uploaded to Firebase Storage where the average file size of five encrypted files increased by 1.5 times the average file size of the five unencrypted original files as expected for AES encryption. It was observed that it took 12989ms for 128-bit and 13166ms for 192-bit, so the above difference is insignificant The encryption time difference for 256-bit key size was remarkable and took 397339ms.TenSEAL in homomorphic encryption demonstrated that computation time was higher because data has to be enciphered for computation. , together with ONT APIS,and the system was 1.5 slower for performing operations on the encrypted data than on the plain ones. The results of these operations that were obtained on the encrypted data were similar to those that were obtained on the unencrypted data proving that TenSEAL worked just as accurately without changing any plaintext data.

6.1.1 Integration with Firebase

The third experiment aimed at testing the ability of combining the encrypted data with firebase for their further storage and use. The goal was to test how the encrypted information could be handled in Firebase without zero or small hindrances to performance. The time taken to upload a 10 MB encrypted file to Firebase was 20 seconds, which are reasonable time compared to other cloud storage systems. Average time to retrieve the encrypted files was 15 seconds the same as the unencrypted file this eliminated any fear that was instilled by the encryption process in the retrieval process. There was no instance of unauthorized access during testing, while the encryption made sure data was safe. The upload and retrieval time were also monitored to detect whether there were some considerable variations. Another security check was done to determine areas of risks successions with forest products. The result of the test showed that there were no main and secondary open or other significant vulnerabilities in this system.

6.2 User Acceptance and Compliance Testing.

The last activity required is to survey healthcare practitioners on the practicability and conformity of the encryption system. The data has been collected through the questionnaires and interviews with the users in order to ask them about their level of satisfaction regarding the system as well as to consider compliance with the data protection regulation. About 85% of the users said that the system was easy to use with a score of 4 with the use of the 1 to 5 scale. Although the participants had a better understanding of the homomorphic encryption process, a small percentage (15%) of the participants was found to have had problems comprehending this concept.

6.3 Discussion

The experiments reveal the advantages and disadvantages of the encryption-based system in addressing the problem. Key findings include: AES requires key length for high level security but the speed of its operations is relatively slow compared to other algorithms when key length is extended. Whereas, 128 key encryption is faster 256 'bit key' is stronger security at the expense of speed. Specifically, TenSEAL infringes privacy during the process of data processing but adds a remarkable overhead cost. This is good for small sample size but may need to be further optimized for large data works. The overall encryption of data, and its storage and retrieval through firebase is efficiently manageable without contributing much to the increased delay. Though the system was able to protect the specimen of exigent healthcare information, the following could be enhanced. Because the current homomorphic encryption operation is still slow and requires much improvement, such as increasing the scale of use in real-world healthcare systems. Security of encryption keys in production should undergo stronger key management policies with other means like the HSM. Healthcare workers could benefit from a more diverse user training program that would explain the types of encryptions and the need to protect patient information.

6.3.1 Suggestions for Improvement:

- ✓ Optimization of TenSEAL: More work could be done toward making TenSEAL more efficient for faster computations to make it applicable for a real-time application.
- ✓ Cloud Storage Optimization: More enhancements of the performance of the cloud storage, including parallel uploading and downloading of files could enhance the handling of large datasets.
- ✓ Enhanced User Interface: Making the file encryption and decryption functionality less complex can go a long way in enhancing experience that many people have regarding encryption systems.

These findings, when compared with previous research in the literature, suggest that encryption is an essential part of securing sensitive healthcare data, and while there are performance trade-offs, the benefits of privacy and compliance far outweigh these challenges.

7 Conclusion and Future Work

The primary aim of this research was to investigate and develop a secure system for handling sensitive healthcare data using encryption techniques, particularly focusing on **AES encryption** and **homomorphic encryption** (via TenSEAL), along with their integration with cloud storage solutions such as **Firebase**. The research sought to answer the question: *How can encryption-based systems effectively secure sensitive healthcare data while balancing performance, compliance, and usability?*

7.1 Objectives and Work Done:

The objectives of the study were as follows:

1. To assess the performance of AES encryption for securing healthcare data and its impact on storage and retrieval efficiency.
2. To explore the potential of homomorphic encryption in preserving privacy while allowing for computations on encrypted data.
3. To evaluate the integration of encrypted data storage and retrieval using Firebase.
4. To assess user acceptance of the encryption system and its compliance with healthcare data protection regulations, including HIPAA and GDPR.

The work completed included multiple experiments involving:

- Testing AES encryption with varying key sizes (128-bit, 192-bit, and 256-bit).
- Evaluating homomorphic encryption using TenSEAL for secure data processing.
- Analyzing the integration of encrypted data with Firebase for cloud storage.
- Conducting user surveys to assess the system's usability and compliance.

7.2 Key Findings:

- ✓ AES Encryption: AES encryption appeared as the most suitable and effective method of protecting vital health care data. Despite increasing the performance overhead especially with longer sizes of keys, it enhanced the degree of data security. The system illustrated the trade-offs between and key sizes where there was a trade of with 128-bit keys achieving higher file processing rates than the 256-bit keys at the cost of security processing time.
- ✓ Homomorphic Encryption: The consideration of employing homomorphic encryption in TenSEAL applied a lot of hopes fitting for the privacy of data during processing. However, the computational overhead was high enough and hence, may not be suitable for large-scale real-time operation. Nevertheless, TenSEAL enabled computations on encrypted data with desirable privacy-preserving and security guarantees where no plaintext information is disclosed.
- ✓ Firebase Integration: By incorporating encrypted data into firebase, it was seen that cloud storage can deal with encrypted files with ease and there is no much delay to upload or download the files. This integration did not bring down the performance of Firebase and the system was also able to uphold high level of security.
- ✓ User Acceptance and Compliance: A vast majority of the users considered the system interface to be friendly with high levels of satisfaction despite a few complaints concerning likely difficulties in comprehending the encryption procedure. In terms of the standards, it was reported that the system corresponded to HIPAA and GDPR that made it suitable for the healthcare industry.

7.3 Implications and Efficacy:

The study effectively put in place and tested an efficient means of securing confidential health information. The outcomes depicted in the study reveal that applications of encryption-based system offer fairly high degrees of data security while still passing the legal and regulatory legal demands. The conclusion also underlines that research pay great attention to optimization of ciphered systems, where safety and productivity should be achieved in realistic conditions, without having a negative impact on healthcare. However, the study also has some drawbacks that are associated with the limitations of the undertaken homomorphic encryption tests because these tests were conducted on a relatively small scale, and, therefore, they cannot be absolutely reliable for the definition of the prospects of this technology on a large scale. The encryption systems created efficiency or latency

that, although not communications/security threats per se, negatively impacted AI by slowing real-time applications below ‘real-time’ speed.

7.4 Future Work:

Although the feasibility of homomorphic encryption was shown in this research much improvement in performance, it is necessary to make homomorphic encryption more realistic in its application. The future of this work can be toward optimizing TenSEAL’s performance or analyzing other HE methods that provide increased computational speed without compromising privacy. Subsequent research may explore the effectiveness of these techniques when implemented to massive datasets that are characteristic to the large accredited health care organizations with records in terms of millions. There are many directions for further improvement of performance, including parallel processing or hardware acceleration of computations to minimize the delay of data encryption and computation.

Enhanced Key Management Solutions: Key management is still an issue when it comes to security and efficiency of certification-based encryption. It is, therefore, possible that future work could consider other key management systems that are more secure than the current type of KMV; for instance, the HSMs to provide better protection from key loss in cloud environments.

Future studies can examine the feasibility of implementing the encryption system in operational healthcare IT structure like EHR systems to determine the efficiency and output and clinical usage scenario. Since, there are still users who did not fully comprehend the process of encryption, the future work may aim to improve the design of the user interface as well as educating and training the users. It would also provide an opportunity to say that encryption systems that are adopted by the healthcare profession can in fact be used effectively. The availability of safe and efficient encryption techniques for health care data holds great business opportunities because of the continued regulatory pressures with special focus on health systems to secure patients’ details. Sustaining the current work, a subsequent study could focus on the development and adjustments of the system depending on certain sectors of the healthcare domain, thereby marketing the system as a sales item to hospitals, clinics, and other medical establishments. This would involve questions of scaling and compatibility with multiple healthcare systems as well as mere installability into existing structures.

In conclusion, the research proved the impact of effectively securing important healthcare data can be achieved using encryption-based systems; however, there are so many areas of improvement and enhancement needed. In order to minimize the influence of such characteristics and address these challenges in the subsequent research, it will be possible to enhance better protective and effective means within the sphere of healthcare and medical data security and according to the GDPR requirements.

References

[1] Bharadwaja Reddy Chirra, “Enhancing Healthcare Data Security with Homomorphic Encryption: A Case Study on Electronic Health Records (EHR) Systems,” *Revista de Inteligencia Artificial en Medicina*, vol. 14, no. 1, pp. 549–59, 2023, Available: http://redcrevistas.com/index.php/Revista/article/view/249.

[2] P Sathishkumar, K Pugalarasan, C Ponnparamaguru, and M Vasanthkumar, “Improving Healthcare Data Security Using Cheon-Kim-Kim-Song (CKKS) Homomorphic Encryption,” pp. 1–6, Apr. 2024, doi: https://doi.org/10.1109/ickecs61492.2024.10616691.

[3] M. Corrales Compagnucci, J. Meszaros, T. Minssen, A. Arasilango, T. Ous, and M. Rajarajan, “Homomorphic Encryption: The ‘Holy Grail’ for Big Data Analytics and Legal Compliance in the

Pharmaceutical and Healthcare Sector?," *European Pharmaceutical Law Review*, vol. 3, no. 4, pp. 144–155, 2019, doi: https://doi.org/10.21552/eplr/2019/4/5.

[4] C. Marcolla, V. Sucasas, M. Manzano, R. Bassoli, F. H. P. Fitzek, and N. Aaraj, "Survey on Fully Homomorphic Encryption, Theory, and Applications," *Proceedings of the IEEE*, vol. 110, no. 10, pp. 1572–1609, Oct. 2022, doi: https://doi.org/10.1109/jproc.2022.3205665.

[5] J. Scheibner, M. Ienca, and E. Vayena, "Health data privacy through homomorphic encryption and distributed ledger computing: an ethical-legal qualitative expert assessment study," *BMC Medical Ethics*, vol. 23, no. 1, Dec. 2022, doi: https://doi.org/10.1186/s12910-022-00852-2.

[6] K. Munjal and R. Bhatia, "A systematic review of homomorphic encryption and its contributions in healthcare industry," *Complex & Intelligent Systems*, vol. 9, May 2022, doi: https://doi.org/10.1007/s40747-022-00756-z.

[7] Y. Al-Issa, M. A. Ottom, and A. Tamrawi, "eHealth Cloud Security Challenges: A Survey," *Journal of Healthcare Engineering*, vol. 2019, no. 7516035, pp. 1–15, Sep. 2019, doi: https://doi.org/10.1155/2019/7516035.

[8] S. Reddi *et al.*, "Privacy-Preserving Electronic Medical Record Sharing for IoT-Enabled Healthcare System Using Fully Homomorphic Encryption, IOTA, and Masked Authenticated Messaging," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 9, pp. 10802–10813, May 2024, doi: https://doi.org/10.1109/tii.2024.3397343.

[9] Bala Annapurna *et al.*, "Secured and cloud-based electronic health records by homomorphic encryption algorithm," *International Journal of Power Electronics and Drive Systems/International Journal of Electrical and Computer Engineering*, vol. 15, no. 1, pp. 1152–1152, Nov. 2024, doi: https://doi.org/10.11591/ijece.v15i1.pp1152-1161.

[10] W. Stallings, "CRYPTOGRAPHY AND NETWORK SECURITY PRINCIPLES AND PRACTICE SEVENTH EDITION GLOBAL EDITION," 2017. Available: https://www.cs.vsb.cz/ochodkova/courses/kpb/cryptography-and-network-security_-principles-and-practice-7th-global-edition.pdf

[11] Rajkumar Banoth and R. Regar, "An Introduction to Classical and Modern Cryptography," pp. 1–46, Jan. 2023, doi: https://doi.org/10.1007/978-3-031-32959-3_1.

[12] A. Verma, G. Agarwal, A. K. Gupta, V. Kumar, and S. Singh, "An adaptive secure internet of things and cloud-based disease classification strategy for smart healthcare industry," *Wireless Networks*, Jun. 2024, doi: https://doi.org/10.1007/s11276-024-03783-5.

[13] M. Li, W. Lou, and K. Ren, "Data security and privacy in wireless body area networks," *IEEE Wireless Communications*, vol. 17, no. 1, pp. 51–58, Feb. 2010, doi: https://doi.org/10.1109/mwc.2010.5416350.

[14] Ambika N, "An Augmented Edge Architecture for AI-IoT Services Deployment in the Modern Era," *Advances in information security, privacy, and ethics book series*, pp. 286–302, Jun. 2022, doi: https://doi.org/10.4018/978-1-6684-5250-9.ch015.

[15] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Manual for Using Homomorphic Encryption for Bioinformatics," *Proceedings of the IEEE*, pp. 1–16;, 2017, doi: https://doi.org/10.1109/jproc.2016.2622218.

[16] A. Alexandru, C. A. Alexandru, D. Coardos, and E. Tudora, "Healthcare, Big Data and Cloud Computing," *WSEAS TRANSACTIONS ON COMPUTER RESEARCH*, vol. 4, pp. 123–131, Oct.

2016, Available: https://www.researchgate.net/publication/310416741_Healthcare_Big_Data_and_Cloud_Computing

[17] K. Potter, D. Stilinki, and Selorm Adablanu, “Homomorphic Encryption for Secure Cloud Computing,” *ResearchGate*, Jul. 2024, doi: https://doi.org/10.13140/RG.2.2.19574.41285.

[18] M. J. Khan, B. Fang, and D. Zhao, “Toward Lossless Homomorphic Encryption for Scientific Computation,” *arXiv.org*, 2023. https://arxiv.org/abs/2309.07284 (accessed Dec. 11, 2024).

[19] N. A. Robinson *et al.*, “Applying genetic technologies to combat infectious diseases in aquaculture,” *Reviews in Aquaculture*, Sep. 2022, doi: https://doi.org/10.1111/raq.12733.

[20] Johnson, R., & Patel, S. (2022). "Homomorphic Encryption in Financial and Healthcare Sectors." *Journal of Applied Cryptography*, 12(2), 113–125.