"SHeMeD": An Application on Secure Computation of Medical Cloud Data Based on Homomorphic Encryption

Hara Salaga[1(✉)], Nikos Karanikolas[1], and Christos Kaklamanis[1,2]

[1] Department of Computer Engineering and Informatics, University of Patras, 26504 Rion, Greece
{salaga,nkaranik,kakl}@ceid.upatras.gr
[2] Computer Technology Institute and Press "Diophantus", 26504 Rion, Greece

Abstract. In today's Internet-centric world, two main challenges have arisen regarding data exploration: a) the privacy of sensitive data and b) the need for users to have an efficient way to manage their data. Cloud computing offers a cost-effective way towards direct data storage and access, which can function as a catalyst to modernize and update legacy systems. Although cloud computing provides efficient dynamic storage space, data security remains a concern. To this end, we propose a novel design and implementation of an application that enables secure storage of personal data in the cloud through a complete homomorphic encryption system. The application is named "SHeMed", which stands for Secure Homomorphic encryption on Medical cloud data. We have concentrated on the secure storage of medical data for breast cancer patients to ensure patients' data privacy and prevent unauthorized access. With the introduction of homomorphic encryption in the core of this application, we achieve four significant milestones regarding the secure "sealing" of data privacy: a) the user does not share the private key with the cloud provider, b) the user can proceed with implementing mathematical operations on encrypted data, c) cloud can be easily scaled and provide a convenient way to store data, d) after the implementation of mathematical operations, the new processed data may be shared with other users, mainly doctors, without providing the original ones. Our design implies that our users will need to retrieve their patient's data just once. In this paper, we present a case study of our application that utilizes homomorphic encryption to eliminate the burden of preserving data confidentiality in the cloud and enable a convenient analysis of medical data through computations on encrypted data.

Keywords: Homomorphic encryption · Cloud computing · Medical data · Data privacy · Data storage · Malicious acts · Confidentiality · Computations · Breast cancer

1 Introduction

Nowadays, the expansion of the Internet has made a tremendous impact on the amount of data exchanged and on the way those data are retrieved, interpreted, and stored. There are two key issues regarding data analysis, i.e., the privacy of sensitive data and

O. Gervasi et al. (Eds.): ICCSA 2023 Workshops, LNCS 14109, pp. 116–133, 2023.
https://doi.org/10.1007/978-3-031-37120-2_8

the data management in the new era of cloud computing. We focus, but not restrict, our attention on medical data analysis for the following reasons. First, medical data privacy and confidentiality are critical and any violation, meaning that integrity may be compromised from adversaries or could be stolen, may even lead to legal sanctions. Second there is a worldwide attempt and ongoing process of medical data digitization and storage. The goal is that health information and patients' records can be created and managed by authorized providers in a digital format having the ability of being shared with other providers across more than one health care organization. For example, healthcare industry proceeds with medical data digitization, referred to as Electronic Health Records (EHR), which is a digital record of a patient's medical history maintained by hospitals or healthcare providers for better handling of medical examinations.

This is likely to improve the quality and efficiency of healthcare as there is ease of access to records, where frequent examinations are required, for example for patients with breast cancer. Given the potential for adversaries to steal or tamper with sensitive data, ensuring its safety is crucial to prevent the dissemination of misleading information. The necessity for full protection of medical digital data is essential, the consequence of which is the rapid development of cryptographic systems as a countermeasure to security threats of third parties.

Homomorphic encryption was first proposed in 1978 [1] as a solution to eliminate the need for multiple decryptions that can potentially compromise the integrity of the information received by the end-user. In other words, homomorphic encryption is a cryptographical method designed to allow mathematical operations to be performed on the already encrypted information while hiding the appearance of plaintext in any third party. Homomorphic encryption enables processing of ciphertext such that the resulting encrypted output data matches the output produced during the first decryption, while preserving the confidentiality of the processed and encrypted data. The practical implementation of homomorphic encryption was introduced by Gentry in 2009 [1]. Homomorphic encryption techniques can be categorized into three groups based on the number and complexity of mathematical operations applied to the encrypted message, resulting in diverse types of homomorphic encryption.

- Partially Homomorphic Encryption (PHE) [13]: Only one mathematical operation is allowed to be performed on the encrypted message, either addition or multiplication, with no limit on the number of repetitions.
- Somewhat Homomorphic Encryption (SWHE) [14]: It becomes possible to use addition and multiplication with a limited number of iterations.
- Fully Homomorphic Encryption (FHE) [15]: It becomes possible to use the mathematical operations, multiplication, and addition, on the encrypted information without limitation in the number of iterations.

Currently, there are several implemented homomorphic encryption libraries. In our implementation, the SHeMed application, SEAL library [3] and specifically EVA compiler [5] have been deployed. SEAL (Simple Encrypted Arithmetic Library), proposed by Microsoft, is an open-source library that implements Fully Homomorphic Encryption (FHE) technology. It provides a set of encryption libraries that allow computation to be performed directly on encrypted data. This enables the creation of encrypted data storage and computation services, where the client never needs to share its key with the service

(e.g., cloud). In addition, this library supports two different homomorphic schemes with different properties: the Brakerski-Fan-Vercauteren (BFV) [17], which allows arithmetic operations on encrypted integers and Cheon-Kim-Kim-Song (CKKS) [8] scheme, which allows approximate computations applied on encrypted real or complex data. The latter method yields approximate results, due to Learning With Errors (LWE) [16] noise, which is the foundation of CKKS homomorphic encryption scheme. CKKS aims at applications of medical nature or the application of machine learning models to encrypted data, etc. Encrypted Vector Arithmetic (EVA) is a compiler for fully homomorphic encryption, implementing the functionality of the SEAL library. EVA has a Python frontend system, called PyEVA, which allows calculations to be expressed using basic arithmetic operations in Python.

Furthermore, healthcare industries require to store multiple EHRs for their patients and the conventional hardware storages are not considered flexible while their cost is significant. Thus, modern organizations adopt Cloud computing, which is defined by NIST [12] as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. One of the most reliable and widely known cloud providers is Google Cloud Provider (GCP).

It is worth emphasizing the presence of ad hoc algorithms in the medical domain, which have been demonstrated by several recent studies, including those referenced as [6] and [7]. These algorithms are developed on an as-needed basis and are designed to address specific problems, making them highly tailored and efficient. Through these studies, it is shown that machine learning and data mining methods can effectively tackle specific medical issues, including but not limited to tracking COVID-19 cases and analyzing genomic data. The application of these techniques can provide valuable insights and aid medical professionals in making more informed decisions, ultimately improving patient outcomes.

In the same manner, the motivation behind our research is driven by the objective to offer medical professionals valuable insights and support, empowering them to make well-informed decisions and improve patient outcomes. To this end, our proposed application seal the data securely using homomorphic encryption without providing the encryption secret key in the cloud provider. The data in our case study are related to patients with breast cancer and we have demonstrated the way that we managed these big data set. In the following sections, we will analyze the design as well as the implementation of an application that allows the secure storage of medical data using a fully homomorphic encryption scheme in cloud storage. Moreover, we will present relevant screenshots of SHeMed application.

1.1 Related Work

The proposed application "SHeMed", which stands for SEAL Homomorphic encryption on medical data, has a major role in applying arithmetic operations in encrypted data using the homomorphic theory while exploits the capabilities of cloud computing. By implementing homomorphic encryption operations in the cloud provider, we exploit the capabilities of cloud computation while we take advantage of the storage.

However, there has been significant research towards securing cloud data with the aim of homomorphic encryption. The research of Poteya M. M., Dhoteb C. A. and Deepak H. S. in 2016 [11] presents an application in which they demonstrate the functionality of a FHE scheme on the cloud. They use the AWS as a provider and add a simple logic to present the accuracy of data from IDE environment as well as show the storage of the encrypted data in DynamoDB of the AWS cloud provider. Another paper that expanded our knowledge towards homomorphic encryption, was introduced by Chen B., Zhao N. [4], where they analyze two fully homomorphic encryption FHE systems, DHCV and CAFED, along with their capabilities and vulnerabilities towards the cloud. In both systems, they propose alternative improved ways for algorithms having as objective to send the key to the cloud server to retrieve ciphertext. Another research worth to be mentioned by Chauhan K. K., Sanger K. S. A. and Verma A. [10], examines the processing state, which is considered as a critical stage for data security especially in cloud computing. They proceeded with the analyzation of three partial homomorphic encryption HE methods and their applications in cloud computing, expanding traditional encryption techniques which are not efficient for processing state of data. Although, homomorphic encryption methods successfully manage the problem of data security in cloud computing, currently, both fully as well as partial homomorphic methods are not so easy to implement for cloud computing. In addition, similar research has been conducted by Vankudoth, Biksham & Vasumathi [18], where they present an overview of homomorphic encryption in cloud computing area urged from the need of security at infrastructure: network level, host level, application level and data. Kocabaş, Övünç and Soyata, Tolga [9] have contributed to the value of fully homomorphic encryption systems by introducing a health monitoring system that detects patient health issues by continuously monitoring the ECG data. This system consists of ECG acquisition devices, a cloud-based medical application, and back-end devices that display the monitoring results. Thus, in order to override security risks associated with personal health information in cloud providers, such as Amazon, a fully homomorphic encryption scheme has been formulated. With the aim of this encryption schema, the operation on encrypted data can be applied without collisions in the cloud and with no concern about personal health data privacy. The prominent work of Ahmad, I., and Khandekar, A. [2], which enhances the knowledge around homomorphic encryption, suggests RSA and Paillier algorithm for homomorphic encryption by using proxy re-encryption algorithm that prevents cipher data from Chosen Cipher text Attack (CCA).

Overall, the abovementioned papers state that homomorphic encryption schemes introduce a new era for security of cloud computing, since calculations on encrypted data without knowing the plain data can be provided while respecting their confidentiality.

1.2 Our Work

The related work as presented above provides a valuable foundation for the research on homomorphic encryption and its adaptation for cloud computing security, which primarily focuses on theoretical aspects. Unlike other studies in the field, our paper adopts a practical approach and expands on the theoretical concept by introducing a user-friendly graphical user interface (GUI) on a cloud-based implementation. Hence, in comparison with the existing literature our work differs as we bring an applicative dimension to

the concept of cloud computing security on medical data. To this end, we introduce a practical implementation of a fully homomorphic encryption scheme to securely store medical records for patients with breast cancer in cloud storage, by introducing the "SHeMed" application. The application offers a convenient way to remove any barriers in terms of confidentiality, availability, integrity of personal medical data in the cloud and enhance doctors to freely use the capabilities of the cloud. For this reason, Google Cloud Provider (GCP) is considered as an appropriate cloud provider in which data can be stored in a single cloud bucket, separated by folders which are used by different users to store their patients' health records, which use the application. Since medical records are decimal results, the implementation of a FHE system, named CKKS, which indicates the capabilities of SEAL library is the most appropriate to be used. To manage quite conveniently the complexity of SEAL library, it has been introduced in the core of SHeMed application the usage of EVA compiler of SEAL library. The user experience has been implemented using the Auth0 page for a flexible and convenient way to identify the application's users. The SHeMed scheme results in improving the communication gap between the cloud provider and the possible users, in our case doctors, having as impact the development of patient's confidence and medical staff in the use and exploitation of personal data in cloud storage.

Our goal is that medical results after the SHeMed usage can be shared freely with other users from around the world for better patients' treatments and for exchanging knowledge without barriers. In short terms, the innovation that homomorphic encryption provides is a "swiss knife" against the struggles that someone may face using cloud providers. We expect our application to trigger the usage of homomorphic encryption in other applications, too. In short, we propose an innovative and convenient solution, where the users can easily navigate through application's UI and the creation of the cryptographic keys can be done through a dedicated terminal locally. The implementation of SEAL library through EVA compiler as we have adjusted it in our application, provides a novel procedure which enables the practical use of homomorphic encryption in medical data towards the aim of breast cancer treatment, expanding most of the research held in this area, which focus on the theoretical issues of homomorphic encryption.

2 Problem Description

The advantages of cloud computing are considered as paramount regarding the usefulness and the easy scalability. Overall cloud computing provides a convenient, easy-to-use way to store personal data with the aim of cloud storage (depending on the provider, buckets etc.). On the contrary, the cloud lacks security which leaves an open gap in healthcare industries and individual doctors, according to their security. The security of medical related data is considered as critical, especially when it comes to patient's data privacy. The health records are required to be confidential because of the protection law between patients and doctors and can be only accessible from them. A patient with breast cancer is required to perform a lot of medical examinations, as well as a doctor needs an efficient way to store her data whiles extracts an output about the progress of medical treatment.

In data's life cycle, as shown in Fig. 1, in every state data need to be accessible by retaining confidentiality for authorized users, integrity for data correctness and availability in any time. One state, which raises concerns about data integrity is the storage.

Despite the storage capabilities of the cloud, which are instantly upgradable based on users demands, the data can be compromised, since to keep the personal data encrypted for safety reasons, public and private keys need to be given in most of cloud providers. The role of homomorphic encryption is a catalyst for this case, since the data stored in the cloud can be changed while are encrypted, using mathematical operations. For enhancing data security, in our application we use Google Cloud Provider (GCP), where private key does not have to be provided. Thus, the state of storage can be locked in the context of security. Hence, in this case of the homomorphic encryption problem studied in this paper, our input is data which contains the frequent results, in a decimal form, of medical examinations of several patients with breast cancer (which are monitored by a doctor) and the goal is to manage and store the patients' data in an encrypted form securely in the cloud while we implement homomorphic operations, which result to a different encrypted form.

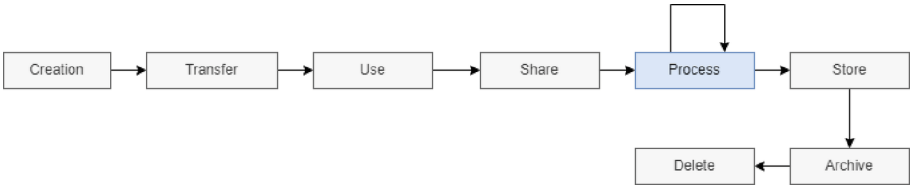

Fig. 1. Data life cycle

3 The Design of the Cloud-Based Application "SHeMeD"

To tackle the problem mentioned above, we propose a cloud-based application SHeMed which is based on storing securely medical examinations for patients with breast cancer on cloud storage using fully homomorphic encryption. The outcome of this application is that users can use the advantages of cloud storage for their patients' medical data, without keeping them in their local systems, something that could compromise the data integrity. The proposed architecture can be shown in Fig. 2 and its components will be analyzed further below.

"Python CLI" Subsystem. The CLI is an external program, part of "SHeMed scheme, that users need to install locally on their computer, so they can use the " SHeMed" application. The possible functions of CLI, use PyEVA, are the following and perform the following actions. First, function "KeyGenerator" is responsible for generating public and private cryptographic keys. Second, "Encryptor", is able to encrypt the initial exam and last, "Decryptor" decrypts the data on its private computer.

"Client" Subsystem. The role of the client subsystem is to provide to the user client, all the necessary services for encryption and decryption of her medical data. To be more precise, through the user's local machine, the following occur:

The user has access to her patient's original medical results for a specific day in excel format, which will be encrypted using the Python CLI. The SEAL library will be implemented for the encryption, key generation, and decryption processes via its compiler,

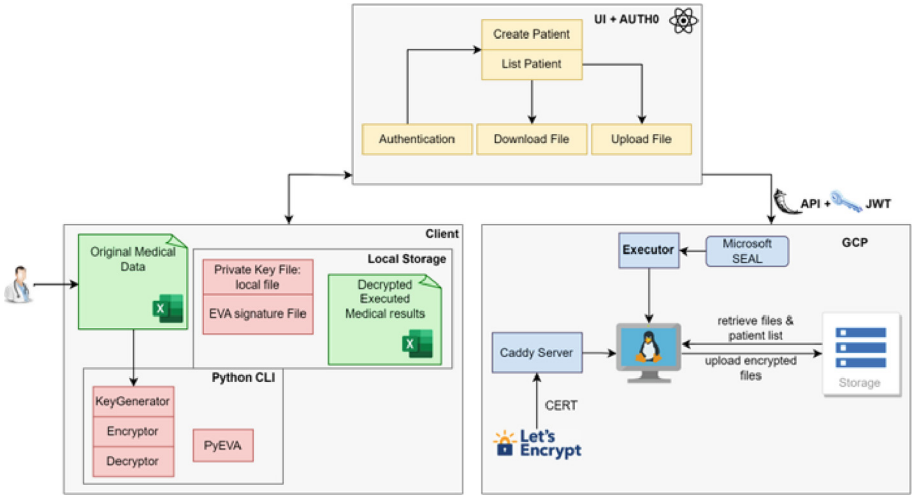

Fig. 2. Architecture of SHeMed

PyEVA. If the required keys are generated via "KeyGenerator", then the "Encryptor" is called to encrypt the original data entered by the user. Finally, the "Decryptor" will decrypt the processed data, which will be received from the cloud provider. Thus, to avoid any compromise of the sensitive information that the data possesses, the private key file is stored locally and the use of the signature of the EVA compiler is also required for the final decryption of the files in their original excel format.

"UI + AUTH0" Subsystem. The function of the "UI + AUTH0" Subsystem is to provide the end user, i.e., the client, all the components needed to interact with the "SHeMed" application. There is an appropriate interface from which the following tasks are performed. In order to ensure authentication, the user should be registered or logged in, otherwise she cannot use it. Then, she can create new patients "Create Patients" to add in bucket, have her total list of patients "List Patients", upload files to the cloud "Upload files" and receive them locally from the cloud "Download files".

"GCP" Subsystem. The "GCP" Subsystem is the server side of the "SHeMed" application and performs the following tasks. First, the interface communicates with the Google Cloud Provider (GCP) using the Flask framework to build REST APIs and establish authentication via JWT generation. Then, Caddy server is responsible for using the SSL protocol through "Let's Encrypt", which provides the appropriate certificate. Next, the virtual machine on which our application code resides is responsible for communicating with the storage in the cloud. Last, the "Executor" call is responsible for performing on the virtual machine the operations on the encrypted data which occur with the aim of SEAL.

4 Implementation of "SHeMeD"

Implementation of Python CLI. The patients' medical examinations need to be encrypted before they are sent to the cloud provider GCP. Hence, an encryption procedure needs to be followed from a user's CLI terminal. Primarily, an input polynomial program is created using PyEVA. Due to some limitations in the CKKS model, the parameters of the vector entered by the user must be of power of two and in this case 16 test results were selected. The aforementioned polynomial program needs to be compiled based on CKKS scheme requirements, which requires two more coefficients: the fixed-point scale for the inputs, i.e., set_input_scales, and the maximum coefficient ranges on the outputs, i.e., set_output_ranges, corresponding to several bits. After the compilation of the program into a fully functional EVA program that can be run on encrypted data, the compile() method returns: the compiled program, the encryption parameters for Microsoft SEAL, and a signature object which specifies how the inputs/outputs should be encoded and decoded, respectively.

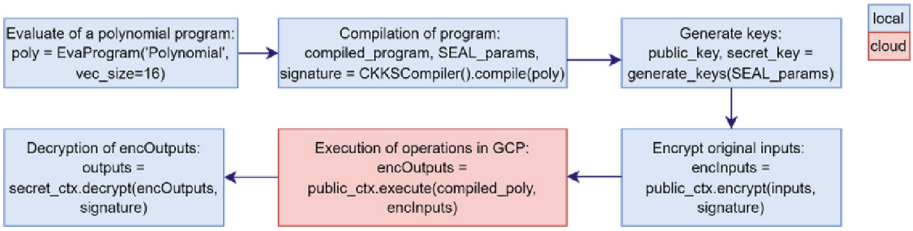

Fig. 3. Process of homomorphic encryption and decryption

Also, EVA through generate_keys() uses SEAL parameters, as an input, to generate a pair of keys: one private and one public, where the public key will be used to encrypt the original input data and the private key will be used to decrypt them after applying homomorphic encryption operations in Google Cloud Provider (GCP). Figure 3 illustrates the prior described process.

Implementation of Backend GCP. The application of the polynomial operation requires the loading of the program (poly), the public key and the encrypted data, entered by the user, from the subfolder (located in medicaleva) of each patient. The call to the EVA compiler's built-in function execute() is responsible for implementing the homomorphic operation, where eventually the result of this function results in an encOutputs vector. Then, again using the save() function, the results end up in a suitably customized file, where it will end up again in the patient's folder.

The application as a backend system is located and "running" on a virtual machine of Ubuntu 22.04 LTS operating system. Therefore, the user from the graphical user interface (UI), selects certain network addresses (endpoints), which through the REST API respond with the corresponding response to the user's request. Below, in Fig. 4, there is a complete scheme of the application's endpoints, that the users can access once they are identified.

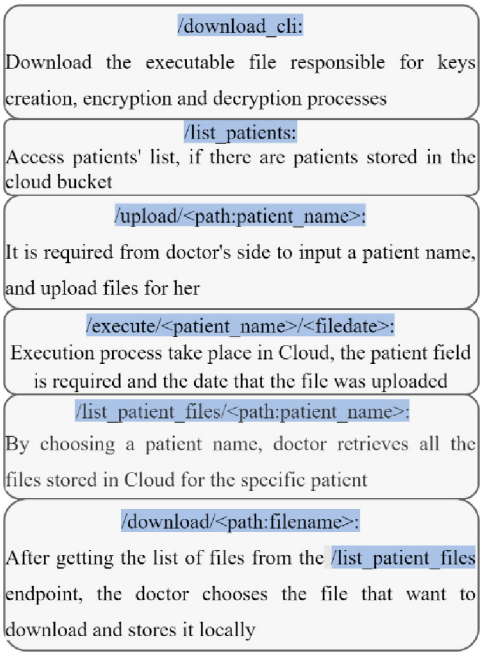

Fig. 4. Endpoints of application

4.1 Environment Analysis

For the implementation of the system, the usability provided by Docker is more convenient as with the implementation of three (3) different Dockerfiles, it is possible to specify all the requirements of the system, without any dependencies installed on a local machine. We have developed a specialized Docker environment that satisfies the functional requirements of the SEAL library and EVA compiler. The second one has been created to cover the appropriate requirements for our backend system, that the application will run on. The last Dockerfile, is responsible for creating appropriate executable files, one for backend application, which can be uploaded in the cloud's virtual machine, and a second one to create an executable SEAL CLI, which end users will store and use in their local machines. The abovementioned Dockerfiles that make up the requirements of the final system, are combined in a bash file to create the final images.

4.2 Theoretical Approach

Our case-study includes a demonstration of program operation and the practical implementation of the CKKS fully homomorphic encryption scheme. This case study uses statistical data for women with breast cancer obtained from [20].

- Initial medical data: The data include measurements from categories of tests for women with benign or malignant disease. A total of 16 categories were sampled from the total. Data obtained from [20]. For these medical records, it is assumed that

each test category shares and follows a normal (Gaussian) distribution, $N(\mu, \sigma^2)$. The values for the mean, μ, and standard deviation, σ, are obtained from Eq. (1–2):

$$\mu = \sqrt{\frac{\sum_{i=1}^{N} x_i}{N}} \tag{1}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}} \tag{2}$$

and after examining medical datasets, x_i, from a wide sample of patients, N, the final values are as following: $\mu = 0.1137$ and $\sigma = 0.05689$ (Fig. 5).

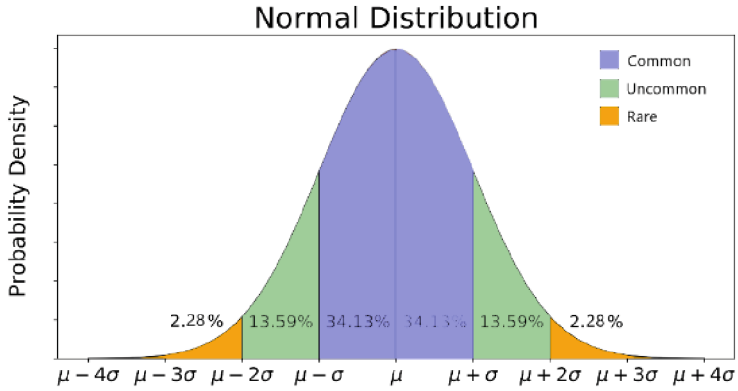

Fig. 5. Normal Distribution indicating three statistical categories [19]

As mentioned, the input data are showcased as part of a normal distribution. A polynomial is, then, calculated as a means of approximating – to some extent – the probability function of the normal distribution. This was achieved by implementing Taylor series of second order, centered around the mean value μ, and based on the maximum range ([0, 1]) of a Probability Density Function (PDF). Hence, the following polynomial is obtained:

$$f(x_i) = \frac{1}{2} - \frac{1}{\sqrt{\pi}} * \left(\frac{\mu - x_i}{\sigma\sqrt{2}}\right) + \frac{1}{5\sqrt{\pi}} * \left(\frac{\mu - x_i}{\sigma\sqrt{2}}\right)^3, x_i : inputdata \tag{3}$$

In the proposed function, after the replacement of μandσ the function that results is the following one:

$$f(x_i) = -216.67x_i^3 + 73.91x_i^2 - 1.4x_i + 0.021 \tag{4}$$

The x_i variable describes the input data of each patient.

- Data output f(x): The values of f(x) belong to the range from 0 to 1. The conceptual meaning of the output indicates a result, ranging from 0 to 1, corresponding to a

percentage (%), which categorizes the result of an examination into three different statistical categories (see Fig. 4), based on the rarity of each medical recording. The categorization of the medical records according to the range of the function values is the following.

– functionvaluesof$[0.4, 0.6]$ are considered common,
– functionvaluesof$\{[0.3, 0.4) \cup (0.6, 0.7]\}$ are considered uncommon,
– functionvaluesof$\{(0, 0.3) \cup (0.7, 1)\}$ are considered rare.

This categorization along with the benignity or malignancy of cancer, could prompt the doctor to conduct the necessary treatment.

It should be mentioned that the aforementioned mathematical polynomial function is not a scientific medical criterion, but an assumption based on the nature of the medical data on breast cancer patients and acts merely as an example to showcase the program's capabilities.

5 Graphical User Interface

So far, we have described the functionality of this application, which can be seen through the layout of the application interface. It is an easy walkthrough for any user to be able to use it without confusion. This application is addressed to particular users, mainly doctors, who want to keep track of their patients' health records, which are stored in the cloud. The application's home page is the following one, shown in Fig. 6.

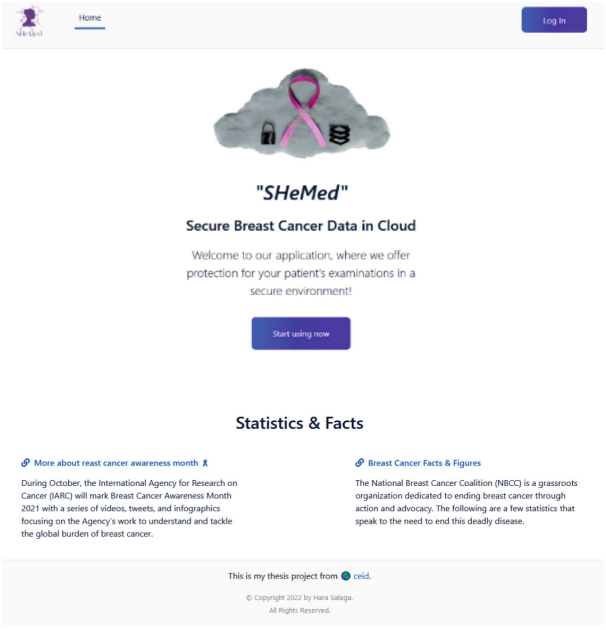

Fig. 6. Home page

Registered users select the "Log In" button, where the following page appears (Fig. 7):

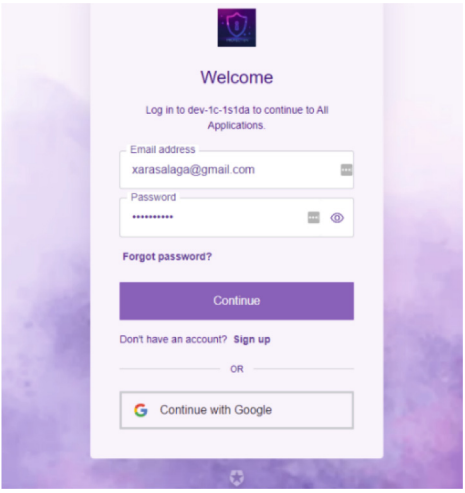

Fig. 7. Sign In

After logging in, the user may navigate in tabs of SHeMed application. In the next Fig. 8, the user is required to download the executable file to her computer via "Download Seal-CLI" to be able to generate the required keys and encrypt the original data before sending it to the GCP.

The user needs to use the terminal of her operating system to access the executable file and 'run' the command "./seal_cli", which is mentioned in the instructions. The relevant commands are:

- ./seal_cli --action encrypt --mode create --path < PATH > --file < PLAIN_FILE >.xls: user creates the required keys (public, private), the crypto-program, and the medical data entered are encrypted.
- ./seal_cli --action encrypt --mode add_file --path < PATH > --key < PUBLIC_KEY > --file < PLAIN_FILE >.xls: user already has necessary keys at her disposal and simply encrypts new medical data of her patients with the keys to store them in the cloud.
- ./seal_cli --action decrypt --mode decrypt_file --key < PRIVATE_FILE > --path < PATH > --file < ENC_OUTPUTS >: the encrypted data retrieved by the user from
- the cloud is decrypted with the private key, which is located exclusively on the local computer of the treating physician. This data is saved in excel file format and ends up in a new folder "medical_results", which is created when the command is executed.

Then, the user can upload file to corresponding endings (.eva,.sealvals,.sealpublickey) in GCP storage by entering the name of a new patient or by choosing an existing one. Select one or multiple files to upload to the cloud, for which it is ensured through the system that they correspond to an allowed file type. Finally, user

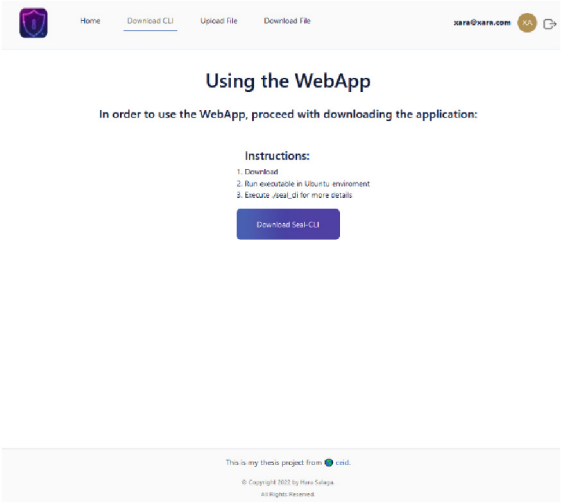

Fig. 8. SEAL CLI

selects the "Submit" button, where she uploads the encrypted medical files to the cloud for the selected patient. The user is presented with a new window with the option to proceed to run the polynomial at the cloud server level (Fig. 9).

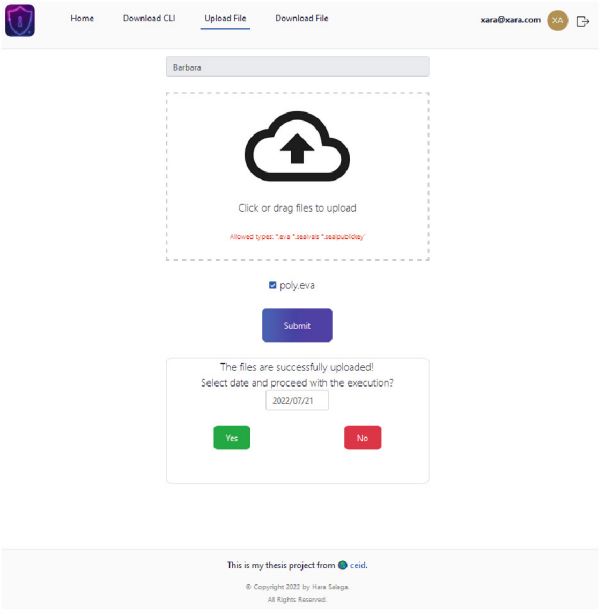

Fig. 9. Upload and execution in the cloud

The user is finally able to download the encrypted files for a chosen patient from the corresponding folder in the unified cloud bucket (Fig. 10).

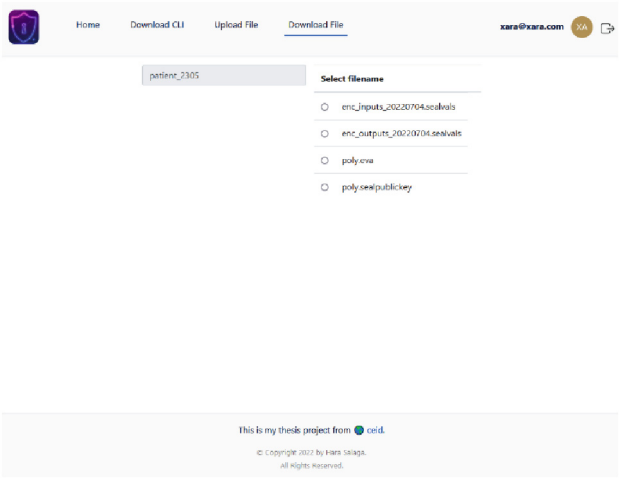

Fig. 10. Download files from the cloud

6 Results and Inferences on Real Data Examples

In this section, we provide our case-study, which was specifically designed for demonstrating the usefulness of our homomorphic scheme. To accomplish this, for the needs of our application, we have used medical breast cancer data from online platform [20], related to smoothness, compactness, concavity, concave points, symmetry, etc. in a decimal form. Mathematical operations combined in a polynomial, which form is encrypted, are implemented in the encrypted form of the original medical data, likewise. After performing mathematical operations, the results were stored in the cloud storage bucket. The EVA compiler of SEAL library was used to perform the mathematical operations inside a cloud VM.

The user can proceed with downloading locally the encrypted modified results from the cloud, by first selecting the patient, and retrieve the relevant ".sealvals" file. The user is now able to decrypt the retrieved cloud data locally. The aforementioned data indicate that the accuracy we achieved is pretty significant. The MSE (Mean Squared Error) is quite low compared to the case where the original data would have been modified after the polynomial was applied: the error approximates the value $1.9878780595116836e-18$.

In Table 1, there is a demonstration regarding a sample of the medical examinations for a specific day, which indicates for each category the condition of a patient regarding the results. Specifically, a category corresponds to some initial test values from which the results are derived, after the application of the mathematical polynomial. The results indicate the condition of the patient according to the fields of values corresponding to them, as shown in Sect. 4.2.

Table 1. Sample of results for patient's health records

Examination category	Original data	Results	Condition
Smoothness	0.08983	0.334590	Uncommon
Compactness	0.03766	0.061527	Rare
Concavity	0.1063	0.447085	Common
Concave points	0.1322	0.627030	Uncommon
Symmetry	0.1467	0.722176	Rare

7 The Novelty of SHeMed: Features and Outcomes

In this paper, we presented a user-friendly application based on a homomorphic scheme, along with a suitable user interface to store medical data in organized folders on the cloud using the same bucket while applying homomorphic operations on encrypted data with the aim of SEAL library. In a more detailed analysis, the application offers the following outcomes:

- Enables an oncology physician to use the cloud as a method of storing medical examinations of breast cancer patients.
- At the same time, it applies mathematical operations, in particular the application of a probability function to the encrypted results of breast cancer's examinations regarding symmetry, coherence, regularity, etc. This possibility presents three important advantages:

  – Our application utilizes the computational power of cloud computing to securely perform complex mathematical operations on encrypted data, without requiring a powerful computing system on the user's end. This allows users to perform computationally intensive tasks within the application that would otherwise be impractical or impossible.
  – The doctor can draw conclusions about the health history of the patient in question. Based on our approach, the doctor may revisit thoroughly application's results about patient's condition for medical categories, where the states uncommon or rare are present. Thus, the doctor can proceed with changing the patient's current treatment or run additional tests to crossover results.
  – In order to protect patient privacy and maintain data security, oncologists can share modified results, i.e., data output $f(x)$ of medical data with other medical professionals, rather than the original patient data. This approach, known as de-identification, allows doctors to collaborate and share knowledge without compromising the integrity of the medical data or the privacy of their patients. By doing so, medical research and patient care can be improved while still maintaining high standards of data security and privacy.

- The separation of the patients' medical data is delegated through the "SHeMed" application to the cloud provider, GCP in our case, by creating the corresponding folders in the unified bucket. For instance, a user, i.e., a doctor, is advised to categorize her patients with a unique ID or nickname and can upload medical examinations for *"patient_2305"* and a corresponding folder is being created in the bucket.

Therefore, the main conclusion derived from the introduction of this application is the complete security in data lifecycle in the cloud, since:

- A separate CLI is created with the ability for the user to generate the necessary keys and provide the original medical data encrypted, without sharing anything on the web.
- Within the application, appropriate authentication is established using Auth0, where the communication with the provider is done during the respective user sessions.
- The type of files that can be stored on the internet is controlled through the application. The application prevents users from uploading file types *".sealprivatekey"*, which holds the private key that is generated from CLI.
- Only the public key and the encrypted operation defined for the imported data are available to the provider, without the risk of decrypting the medical results, since the private key is in the possession of the user.
- From a list of patients and based on a date, selected by the attending physician, the attending physician selects and retrieves the encrypted tests, which will eventually be decrypted via the CLI with the usage of the private key, which is reserved locally.

In this paper, the usability of SHeMed, as an implementation of a FHE scheme, has been demonstrated through the field of medical data. However, SHeMed application can be easily extended and adapted to a broad range of applicability, including various data and scenarios.

8 Conclusion and Future Work

In this paper, we have proposed an efficient, easy-to-use application for implementing fully homomorphic encryption by using SEAL library for data stored in a cloud provider. The usage of this application indicates that important confidential data can be stored in the cloud securely. Consequently, it can provide the possible elimination of any countermeasure and risk of interception or destruction of personal medical data. Although homomorphic encryption is still an emerging field, research has indicated that it has the potential to protect against attacks from malicious actors, even in the presence of quantum computing. Also, the computational power and complexity that quantum computing offers can provide the resources so that operations take place in less computation time. Thus, this application will provide a practical foundation for working with big data sets of medical data and developing standards for homomorphic encryption in healthcare organizations and other industries. The current implementation has a limited set that can be managed efficiently. However, the problem is expected to be solved with the introduction of quantum theory. Overall, instead of classic encryption techniques, homomorphic encryption will be a milestone in the elimination of the unencrypted security gap, which will provide innovative options for cyber security. The unlimited medical data storage on cloud will introduce a worldwide access for users to gather information

without collisions or the fear of compromising the patients' privacy. From homomorphic encryption's perspective, another future research direction would be the enhancement of existing homomorphic encryption techniques, like Fully Homomorphic Encryption (FHE) schemes, which are currently time consuming and cannot support management of unlimited data. Also, another potential avenue for future enhancements is to adapt the secure multi-party computation (MPC) technique to facilitate multiple parties computing a function on their private inputs without compromising their inputs to each other. This advancement holds great promise for the healthcare sector, as it would enable doctors from around the world to collaborate and achieve superior medical outcomes. Further research can concentrate on developing more efficient homomorphic encryption methods that are optimized for secure MPC, improving the privacy, security, and efficiency of healthcare data analytics.

References

1. Acar, A., Aksu, H., Uluagac, S., Conti, M.: A survey on homomorphic encryption schemes: theory and implementation. ACM Comput. Surv. **51**, 1–35 (2017)
2. Ahmad, I., Khandekar, A.: Homomorphic Encryption Method Applied to Cloud Computing. (2014)
3. Chen, H., Laine, K., Player, R.: Simple encrypted arithmetic library - SEAL v2.1. In: Brenner, M., et al. (eds.) Financial Cryptography and Data Security. LNCS, vol. 10323, pp. 3–18. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70278-0_1
4. Chen, B., Zhao, N.: Fully homomorphic encryption application in cloud computing. In: Proceedings of the 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing (ICCWAMTIP) (2014)
5. Chowdhary, S., Dai, W., Laine, K., Saarikivi, O.: EVA improved: compiler and extension library for CKKS. In: Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pp 43–55 (2021)
6. D'Angelo, G., Palmieri, F.: Discovering genomic patterns in SARS-CoV-2 variants. Int. J. Intell. Syst. **35**(11), 1680–1698 (2020). https://doi.org/10.1002/int.22268
7. D'Angelo, G., Palmieri, F.: Enhancing COVID-19 tracking apps with human activity recognition using a deep convolutional neural network and HAR-images. Neural Comput. Appl. **35**(19), 13861–13877 (2021). https://doi.org/10.1007/s00521-021-05913-y
8. Huynh, D.: CKKS explained, Part 3: Encryption and Decryption. https://blog.openmined.org/ckks-explained-part-3-encryption-and-decryption/ (2020)
9. Kocabas, Ö., Soyata, T.: Medical data analytics in the cloud using homomorphic encryption. In: Raj, P., Deka, G.C. (eds.) Handbook of Research on Cloud Infrastructures for Big Data Analytics, pp. 471–488. IGI Global (2014). https://doi.org/10.4018/978-1-4666-5864-6.ch019
10. Chauhan, K.K., Sanger, A.K., Verma, A.: Homomorphic encryption for data security in cloud computing. In: 2015 International Conference on Information Technology (ICIT), pp. 206–209 (2015)
11. Manish, M.P., Dhote, C.A., Sharma, D.H.: Homomorphic encryption for security of cloud data. Proc. Comput. Sci. **79**, 175–181 (2016)
12. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg (2011)
13. Liu, J., Mesnager, S., Chen, L.: Partially homomorphic encryption schemes over finite fields. In: Carlet, C., Hasan, M.A., Saraswat, V. (eds.) Security, Privacy, and Applied Cryptography

Engineering. LNCS, vol. 10076, pp. 109–123. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49445-6_6
14. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. IACR Cryptology ePrint Archive , 144 (2012)
15. Gentry C.: A Fully Homomorphic Encryption scheme. Stanford University. (2009)
16. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM (JACM). **56**, 84–93 (2009)
17. Wibawa, F., Catak, F.O., Sarp, S., Kuzlu, M.: BFV-based homomorphic encryption for privacy preserving CNN models. Cryptography. **6**(3), 34 (2022)
18. Vankudoth, B., Vasumathi, D.: Homomorphic encryption techniques for securing data in cloud computing: a survey. Int. J. Comput. Appl. **160**, 1–5 (2017)
19. Normal distribution. https://money.stackexchange.com/questions/144051/calculate-stock-price-range-using-standard-deviation
20. Kaggle datasets homepage. https://www.kaggle.com/datasets