Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

A GUI Frame work for Homomorphic Encryption Operations

Karlapalem Sujitha, Reddyvari Venkateswara Reddy, Vemuri Chinmai Sai Abhishek,
L.L.N.V.S.R.K. Sai Surya, Shruthi Jha
Assistant Professor, Department of CSE (Cybersecurity), CMR College of Engineering & Technology, Hyderabad, Telangana State, India.
Associate Professor, Department of CSE (Cybersecurity), CMR College of Engineering & Technology, Hyderabad, Telangana State, India.
B. Tech Students, Department of CSE (Cybersecurity), CMR College of Engineering & Technology, Hyderabad, Telangana State, India.

***Abstract*—This GUI framework builds upon homomorphic encryption operations underlying it to provide convenience for its users. The users are able to perform secure computations that are private and confidential without effort. A structure is developed with four homomorphic encryption algorithms, including Paillier, BGV, CKKS, and BFV, for specific data types and operations. Paillier provides partial homomorphic decryption on integer numbers, which can be performed, whereas BGV and BFV allow for complete homomorphic encryption in integers along with additional operations for instance multiplication and exponentiation. CKKS makes it possible to perform more complicated operations than integer calculations, such as floating-point instructions, which is essential for safety regarding decimal value computations. Customtkinter (Python) was used for the GUI development, which presents easy controls and display elements for the users to input two numbers and select encryption algorithms and operations via drop-down menus. The introduction of the Pyfhel and phe modules brings about the homomorphic encryption functionalities that guarantee data security during the computation process. Pillow is the image handling library for this GUI interface, which facilitates the usage visual aesthetics and a good user experience. As well, it checks transparency and accountability by writing the algorithm specifics, input numbers, operations, decrypted results, and cryptographic keys in the output folder.**

***Keywords*—Homomorphic encryption, Operations on encrypted information, Framework, Strategies, Algorithms, Applications, Additive homomorphism, Multiplicative homomorphism, Performance overhead, Cipher text growth, Real-life applications, Privacy-preserving computation.**

1 INTRODUCTION

In the perspective of data security, which is filled with dangers and vulnerabilities, homomorphic encryption offers an opportunity to learn how to overcome such risks. As our digital footprint grows at an astonishing rate, it is now more significant than ever to keep the safety of personal data a priority. Homomorphic encryption offers the possibility of secure encryption calculations on encrypted data without decryption, something that was regarded as impossible before. This cryptographic masterpiece is based on four different algorithms: Paillier, BGV, CKKS, and BFV. Investigate every algorithm's options for functions for different datasets and operations. It is evident that this model has the capability to adapt and is flexible. While Paillier, BFV, and BGV perform computations with integer numbers, CKKS is used for floating-point values.

1.1 A Conceptual Overview

Before taking a deep dive into the technical components of particular algorithms, the simple principles behind homomorphic encryption must first be understood. At its heart is the actuality that homomorphic encryption allows computation on ciphertext so as to produce an encrypted result, which is decrypted only by the authorized recipient. This progressive technique stands in mathematically sharp contrast with ordinary encryption methods where authentication is an input for computing, which exists with security hazards. Homomorphic encryption is an approach to hiding user privacy during both data computation and evaluation. The robbery of facts and homomorphic encryption are able to run the operations without destroying the integrity of the data; thus, they become the first method to solve the challenge of both privacy and performance, an accomplishment that was earlier thought to be unattainable. It is this general principle that we should keep in mind as we embark on this fascinating journey into the intricacies of homomorphic encryption, which, in a way, guides our minds right ahead as it illuminates the way forward for us.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

2 LITERATURE REVIEW

"Homomorphic Encryption: A Comprehensive Review" This evaluation outlines homomorphic encryption's important components, introduces consultant homomorphic encryption schemes, and indicates how homomorphic encryption enables computation on encrypted statistics. It touches on the homomorphic encryption technique evolution stage, along with the partially homomorphic, somewhat homomorphic, and fully encrypted schemes. The evaluation assesses the sturdy facets along with shortcomings in every scheme. It also stresses packages in stable computation, privacy-maintaining information analysis, and cloud computing. As nicely as this, the paper touches on the demanding situations and prospects inside the homomorphic encryption realm and provides examples of ways to relieve the burden on performance while tightening safety and adapting to scalability.

"Homomorphic Encryption for Privacy-Preserving Cloud Computing: A Survey" This review gives an introduction to the homomorphic encryption methods for privacy-preserving cloud computing, where sensitive data is kept from being sent to the cloud computers. It studies various homomorphic encryption schemes in different cloud computing scenarios in accordance with security, performance, and usability. The survey focuses on the available frameworks and protocols for privacy-preserving cloud computing using homomorphic encryption, specifying their strengths and limitations.

"Homomorphic Encryption for Privacy-Preserving Machine Learning in Healthcare: A Review" This review deals with the chosen homomorphic encryption for the goal of preserving privacy in machine learning in healthcare applications since personal medical data is researched and analyzed without breaching patient privacy. It does a survey on the current homomorphic encryption procedures and their aptitude for healthcare data analysis tasks, e.g., disease prediction, treatment recommendation, and medical picture analysis. The study examines the efficacy and scalability of homomorphic encryption-based machine learning techniques in healthcare settings via examination of factors like computational functionality, data interoperability, and compliance with regulations.

3 METHODOLOGY

We have finished the mission. We entirely examined the available homomorphic encryption methods in Python, and we accordingly chose phe and Pyfhel as the basic components. These choices were justified by their high functionality, complexity, trustworthiness, and conformity with the system goals. Modules used are:

- phe won the spot in this category by virtue of its Pallier encryption, which permits partial homomorphic encryption suitable for integer computations. Its simplicity and efficiency of execution of operations like addition, subtraction, and scalar multiplication have made it the first choice for basic operations in mathematics. Furthermore, the integration was rather simple to do, and we had clear documentation which helped in the successful development progress, allowing us to recalibrate towards the front-end GUI interface design and user experience expedition.
- Pyfhel is the combined effort of all those above-mentioned BGV, BFV, and CKKS algorithms and popular choices for fully homomorphic encryption schemes. These features, which include multiplication, exponentiation, and floating point number calculations, accommodated the various needs of our project well. Pyfhel's features and flexibility were at the core of the subsequent execution of the homomorphic encryption functionalities within the GUI.
- During the advancement of GUIs, we tried different Python packages for the making of Graphical User Interfaces (GUIs). Finally, we decided on customtkinter because of its modern looks and the ease with which it works with tkinter, which is the standard GUI toolkit for Python. customtkinter offers more styling choices and makes it easier to build attractive GUIs. This alignment follows our aim to let users do the homomorphic encryption operations visually and intuitively.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

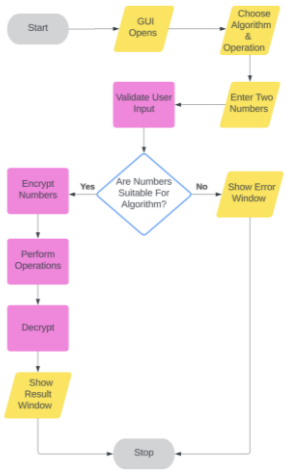

Fig. 1 – Flowchart

The application initiates the process on the GUI interface, where options to choose an algorithm and operation come up on the drop-down menus. When they are chosen, users enter two numbers into the designated fields. The application finally checks if the user input meets the defined rules of the algorithm selected. If the numbers are suitable for the algorithm, they are encrypted utilizing the related homomorphic scheme. Encryption happens first; then the mathematical operation is performed, and finally, decryption takes place to produce the desired result. The decrypted output appears in the result window for the user. When the user inputs numbers that are unnecessary for the specific algorithm, the error window pops up. It informs the user about the discrepancy. The flowchart depicts the flow of the process in the application GUI, making sure that the operations of the homomorphic encryption are done smoothly and with ease for the user.

4 ARCHITECTURE

The architecture and design of our project are purpose-built to form the foundation of a fairly resilient, user-friendly system of homomorphic encryption techniques. The crucial part of our project lies in an architectural modular and scalable solution that incorporates a series of encryption modules on a back-end side with a GUI interface, thus enabling intuitive interaction and transparent operations.

The architecture of our project is structured around two main components: the GUI interface and the server-side(backend). The GUI frontend acts as the main communication channel with which users communicate and get to select algorithms and operations, visualize the results, and input the designated data using understandable controls and feedback. the frontend is designed by use of customtkinter, a Python library that helps in the creation of GUI elements tailored to the specific needs of the application.

The cryptography back ends, working on Pyfhel, phe, and other libraries, form the computational core of our project, performing various homomorphic encryptions with different levels of complexity. The modules implement the Paillier, BGV, CKKS, and BFV cipher schemes, each designed for specific applicability and required mathematical operations. The modular design of the backend functions as a plug-and-play element, enabling effortless integration of cutting-edge encryption techniques and schemes to meet the demands of increased security and the ever-evolving landscape of homomorphic encryption technology in the future. The result window displays the result of the operation. The application then asks users to refer to the output folder, where they will come across other elaborated information stored in the result.json file. The file saves structural data about the encoding process, for example the algorithm performed, the input numbers, the operation performed, and the result that underwent decryption. Besides, the output folder has keys stored in files; due to this fact, users are allowed to review and ensure the cryptographic operations carried out by the application have been correct.

4.1 Interaction Flow for Performing Homomorphic Encryption Operations

The process flow in the guide user interface (GUI) follows in an understandable way to users through the usage of a sequence of logical steps that guides them in the process of efficient implementation of homomorphic encryption operations. The process starts with the GUI interface opening, and the drop-down menus are shown for algorithm selection and operation. After you pick the options you want, you enter two numbers into the specific fields and then click the encryption button.

Then, the user input gets examined to ensure fit with regard to the selection algorithm. If these numbers are found to be suitable, they are encrypted with the chosen encryption scheme. The encrypted numbers are subjected to the operation specified and then decrypted to derive the output. After completion of the encryption and computation tasks, the decrypted final result appears in the output area for user review.

4.2 Integrating Algorithms

The homomorphic encryption framework provides support for four algorithms varying in functional scope and practical complexity, allowing them to satisfy the differing parameters of encryption. The Paillier algorithm, well-known for its partial homomorphic encryption faculties, ensures the carrying out of the principal operations of addition, subtraction, and scalar multiplication on integer numbers. It is perfect for scenarios where data honestly is of primary importance and where all the operations are based on integers. On the other hand, the BGV is a full homomorphic encryption

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

implemented for integers. Therefore, more complicated operations can also be executed by the GUI platform, such as multiplication and exponentiation. This furthers a functionality aimed at making the GUI interaction matrices expand beyond their immediate scope for a wider range of use cases, thereby allowing the users to make complex computations on their encrypted data.

The homomorphic encryption method designed by CKKS and applicable to the whole field of floating-point number operations has been created, ensuring the necessary requirements for using it in numerical computing. Incorporating CKKS into the architecture of the GUI framework presents its ability to manipulate floating-point data types, unlocking new options for secure computation in finance, natural sciences, and machine learning, to name just a few fields. As an additional layer of flexibility, the BFV algorithm acts as another method for fully homomorphic encryption of integers besides the BGV algorithm. BFV, similar to BGV in some aspects, may be preferable and more robust in other circumstances, conditioned by the particular needs of an individual. These four algorithms put into the GUI framework allow the GUI framework to become a tool that can grapple with a variety of homomorphic encryption ranges with dexterity and agility.

4.3 Handling Data Input and Output

The GUI system makes it possible for the client to enter data by means of intuitive controls where the client has to present two numbers, choose an encryption algorithm, and specify the operation to undertake. Input validation blocks the introduction of data that differs from system requirements and is formed based on algorithm parameters and operation usage; hence, it ensures a flawless computational process. Once the data entry portion is completed, the data is passed to the encryption module, where it is then encrypted using the particular algorithm chosen. The homomorphic encryption algorithm is supported well by four frameworks: Pailler, BGV, CKSK, and BFV, which pay attention to the differences in data types and operations, respectively. The encryption process attains the confidentiality objective for the entered data and also provides the important function of secure communication with encrypted values.

The impact of the input data encryption is further influenced by the GUI framework, which executes the operation on the encrypted values, utilizing the possibilities of the selected one from the arsenal of homomorphic encryption algorithms. An illustration is the case when adding is the picked operation; the GUI framework calculates the sum of the encrypted numbers, allowing for the homomorphic addition operation used in the selected algorithm. The modular GUI framework design, which allows for the convenient connection of different encryption programs and actions, is a feature of the system. It ensures flexibility and adaptability to the different computational requirements of the system. Whether it is basic arithmetic operations on integers or complex computations on floating-point numbers, the GUI framework offers the most secure solution for the handling of the data.

Once the computations are performed on the encrypted data, the GUI framework will be responsible for decrypting the result to obtain the final output. Decryption is the inverse process in which corresponding decryption keys, which are created and maintained internally by the encryption module, are used. The outcome of decryption is exhibited in the output area of the GUI interface, where the users get a vivid and understandable statement of the computation result. Moreover, the GUI interface will not only display the decrypted file but also prompt the client to explore the output folder for more details in the result.json file. This file embodies data about the encryption process, for instance the algorithm, input numbers, operation, and decrypted outcome. Besides, users are able to save the cryptographic keys in output folder files, so they can audit and validate the operations by the cryptographic application.

4.4 Error Messaging

Errors that might arise in the input data validation process and during the encryption process will have the GUI framework provide clear and useful error messages for the user's explanation and to further assist troubleshooting. The error messages are created to be terse and particular, providing the users with the kind of information that they need to come up with a solution to the problem.

As an example, the GUI framework reveals error messages that pinpoint the particular problems and invite users to enter valid inputs if users attempt to introduce wrong values. The error messages are clearly displayed within the GUI interface in a very noticeable form, so that the client can clearly locate them and solve any input-related issues.

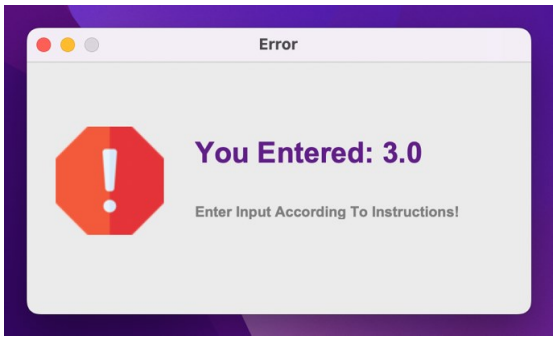

Fig. 2 – Error Handling Window

Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

5 SYSTEM REQUIREMENTS

- Processor: A modern multicore processor (e.g., Intel Core i5 or AMD Ryzen) is suggested to tackle the computational challenges of homomorphic encryption operations effectively.
- Memory (RAM): The RAM requirement should be at least 4GB to have the GUI framework run smoothly, particularly in certain situations.
- Storage: Adequate space for the installation of Python and the related libraries, input data, output files, and keys generated after encryption operations must be provided.
- Operating System: GUI Framework supports major operating systems such as Windows, macOS, and various Linux distributions, which include Ubuntu, Fedora, and CentOS.
- Python: It requires the Python 3 version to exist on the system. Users can download and install Python from the official Python website (https://www.python.org/). Otherwise, you can use the package managers provided by your operating system.
- Python Libraries: The homomorphic encryption-based GUI framework is grounded on several Python libraries, namely, for homomorphic encryption (Pyfhel, phe), GUI development (customtkinter) and image handling (Pillow). Users must check that these libraries are installed and correctly configured in the Python environment they are using.

6 RESULT

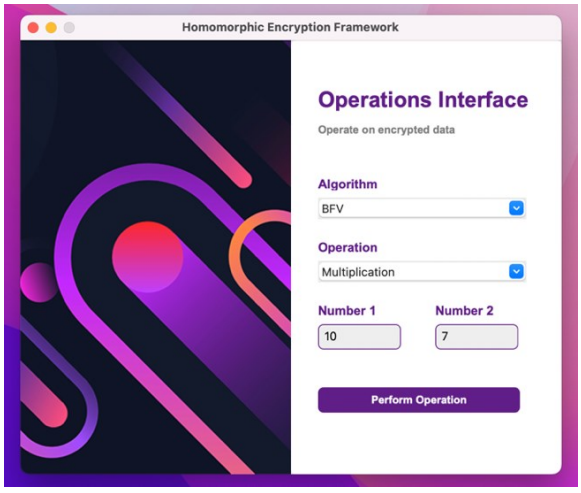

Fig. 3 – Input Using BFV Homomorphic Encryption

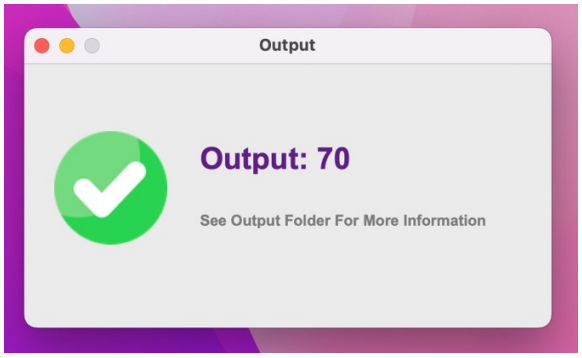

Fig. 4 – Computation Output

Fig. 5 – Output Folder and result.json

7 CONCLUSION

In short, the introduced GUI framework represents great progress in the area of homomorphic encryption, where secure computations are performed while maintaining data confidentiality and privacy. GUI architecture adopting multiple homomorphic encryption algorithms, among which we can list Paillier, BGV, CKKS, and BFV, is capable of supporting a wide range of calculations that vary from integer-based ones to floating-point problems. The GUI interface of the framework has made the encryption process less complex for the users by providing them with a list of algorithms and operations that can be selected from the drop-down menu. Data input and result visualization are made easy by such an interface. The GUI gains cross-platform usability by employing the potential provided by Python, and customtkinter empowers integration with already existing Python environments.

Then again, the GUI framework sets additional high level for data safety and transparency while storing encrypted information on input data and performing operations that result in a specified output folder. This affords the user not only an improved understanding of the encryption system but also easy auditing and verification of the results. The GUI framework, together with the user input validation feature and error handling mechanism, guarantees the precision and resistance of the encryption framework, with almost no possibility of false computations and a better user experience.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

8 FUTURE SCOPE

- Improvement in Performance: Future refinements on the GUI framework could be oriented towards accomplishing this goal through parallelization methods and exploiting hardware accelerators such as GPUs. By taking advantage of the large number of processors (i.e., hardware) found in the current generation, the GUI framework can easily have shorter execution times and perform more sophisticated and advanced operations.
- Integration with Cloud Computing Platforms: As cloud computing technology gains more and more attention, it has become an amazing chance for the cloud platform and GUI framework to go together and thus provide unlimited expansion. Through the utilization of cloud infrastructure, users can transfer computationally heavy encryption functions distant from local machines, which could eventually result in less load on the local computers and easier roughing because of the computation upon demand while maintaining security. Along with this, integration with cloud storage services can lead to the accessibility of encrypted data and outcomes from any place that is smooth and retains these same values.
- Support for Additional Operations and Data Types: At present, the architecture provides diverse basic integer and floating-point number operations. New enhancements are expected to improve arithmetic with a wider variety of operations and data types. This extension may include support for division, logarithm, and trigonometric operations, apart from columns and rows of data structures like matrices and vectors. Diversification of the supported operations and data types can make the framework the best solution for various applications and for solving many tasks.
- Integration with Machine Learning and Data Analytics: Encryption with homomorphic characteristics can be a powerful solution for ensuring user privacy in machine learning and data analysis pipelines because it allows sensitive data to be processed without breaking secrecy. At the next stage, effort should be put into making the machine learning libraries and data analytics platforms integrated into the GUI framework to attain the high security and privacy requirements. This integration could be tapped into to achieve tasks like encrypted machine learning, encrypted data aggregation, and encrypted data mining, which are vital for the security of private data and open new areas of operation with regard to privacy-preserving data analysis.

REFERENCES

[1] "Homomorphic Encryption: A New Paradigm for Secure Remote Outsourcing of Biometric Identification Services" by Jose M. Delgado et al. - This research paper is keen to the usage of homomorphic encryption in the area of remote biometric identification services. It shows the possibility to enhance the privacy and security of this outsourcing scenario.

[2] "Homomorphic Encryption for Network Security: A Review" by Muhammad Taha and Marwan Krunz – This paper focuses on homomorphic encryption as a security enhancing tool, with specific instances such as outsource secure computation and privacy-preserving data analysis as applications.

[3] "PyCrypto: A Python Library for Cryptanalysis Operations" by Dwayne C. Litzenberger – this marks the introduction of PyCrypto, a Python library which provides cryptographic functions, enabling future development of PyEncrypto, a GU framework for homomorphic encryption.

[4] "Homomorphic Encryption for Privacy-Preserving Machine Learning: A Tutorial" by Raluca Ada Popa et al. - This tutorial presents in a practical manner a step-by-step guide to implementing homomorphic encryption for privacy-preserving machine learning, encouraging the readers to use those insights for their research and practice.

[5] "Homomorphic Encryption: A Brief History of Homomorphic Encryption: Past, Present, and Future Directions" by Kristin Lauter- This review paper offers a historical account of homomorphic encryption, ranging from the initial theoretical breakthroughs to practical implementations and finally into the future research directions.

[6] "Secure Computation with Homomorphic Encryption: A Review" by Xiao Wang et al. - This article provides a wholesome perspective of the secure computation methods using homomorphic encryption, including the applications, limitations, and future prospect.

[7] "Homomorphic Encryption for Privacy-Preserving Data Analysis: Challenges and Opportunities: Homomorphic Encryption in Privacy-Preserving Data Analysis" by Yulia R. Gel et al. The paper discusses the challenges and opportunities of using homomorphic encryption to solve the privacy-preserving data analysis problem, but at the same stage notes that here is a trade-off between efficiency, security, and usability.

[8] "Homomorphic Encryption for Secure Outsourcing of Genomic Data Analysis: A Review" by Mohammad Sadegh Rasooli et al. - The present review paper focuses on the homomorphic encryption method for secure distribution of genome data analysis; the method can permit the hidden information of data to be kept in the purview of researchers's collaboration without loss of privacy of data.