

PRIVACY-PRESERVING DATA ANALYSIS: PERFORM DATA ANALYSIS ON ENCRYPTED DATA STORED IN THE CLOUD

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

BHARATH LP (Reg No – 40110177)
ASHOK KUMARAN S (Reg No – 40110126)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade “A++” by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119**

APRIL 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafide work of **BHARATH LP (Reg No - 40110177)** and **ASHOK KUMARAN S (Reg No – 40110126)** who carried out the Project entitled **“PRIVACY-PRESERVING DATA ANALYSIS: PERFORM DATA ANALYSIS ON ENCRYPTED DATA STORED IN THE CLOUD”** under my supervision from November 2023 to April 2024.

Internal Guide

Dr. T. JUDGI M.E., Ph.D.

HEAD OF THE DEPARTMENT
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Head of the Department
Jeppiaur Nagar, - 600 119.
Chennai - 600 119.
Dr. L. LAKSHMANAN, M.E., Ph.D.

Submitted for Viva voce Examination held on 13-4-2024

13/04/24
Internal Examiner

13/4/24
External Examiner

DECLARATION

I, **BHARATH LP (Reg No - 40110177)**, hereby declare that the Project Report entitled "**PRIVACY-PRESERVING DATA ANALYSIS: PERFORM DATA ANALYSIS ON ENCRYPTED DATA STORED IN THE CLOUD**" done by me under the guidance of **Dr. T. JUDGI M.E., Ph.D.** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 13/4/2024

PLACE: Chennai

Bharath LP
[Handwritten Signature]

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. T, JUDGI M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

This Python-based project aims to enable privacy-preserving data analysis by implementing secure techniques such as homomorphic encryption or secure multiparty computation. The project revolves around performing data analysis on encrypted data stored in the cloud, ensuring that sensitive information remains protected throughout the processing phase. By utilizing homomorphic encryption, the data can be encrypted in such a way that computations can be performed directly on the encrypted data, thereby eliminating the need for decryption. This approach ensures that the privacy of the data is maintained, minimizing the risk of exposing any sensitive information. Alternatively, the project can employ secure multiparty computation where multiple parties collaborate to analyze the data without revealing individual inputs. This technique allows the data to remain encrypted throughout the analysis process, providing a secure environment for data analysis. The implementation will be done using Python, a powerful and widely-used programming language, ensuring accessibility and ease-of-use. By these privacy-preserving techniques with cloud-based storage, the project offers a comprehensive solution for conducting secure and efficient data analysis while maintaining data privacy.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	vii
1	INTRODUCTION	1
2	LITERATURE SURVEY	3
2.1	Inferences from Literature Survey	3
2.2	Existing System and Proposed System	7
2.3	Open problems in Existing System	7
3	REQUIREMENTS ANALYSIS	9
3.1	Risk Analysis of the Project	9
3.2	Software Requirements Specification Document	12
4	DESCRIPTION OF SYSTEM	13
4.1	Selected Methodology or process model	13
4.2	Architecture / Overall Design of Proposed System	16
	Description of Software for Implementation and	17
4.3	Testing plan of the Proposed Model/System	
4.4	Description of software for implementation and testing	20
4.5	Project management plan	25
5	IMPLEMENTATION DETAILS	26
5.1	Development and deployment setup	26
5.2	Algorithm	27
5.3	Testing	28
6	RESULTS AND DISCUSSIONS	29
6.1	Presentation of results	29

6.2	Discussion of results	30
6.3	Implications and future work	34
7	CONCLUSIONS	35
7.1	Conclusion	35
7.2	Future work	35
7.3	Research issues	35
7.4	Implementation issues	36
	REFERENCES	37
	APPENDIX	38
	SCREENSHOTS	42
	SOURCE CODE	48
	RESEARCH PAPER	60

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
4.1	Flow chart	13
4.2	System architecture	16
5.1	Encrypted result	29
5.2	Decrypted result	30
5.3	Signup interface	31
5.4	Login interface	32
5.5	Dashboard interface	32
5.6	Analysis interface	33

CHAPTER 1

INTRODUCTION

Privacy-Preserving Data Analysis is an emerging field that aims to ensure the security and privacy of data while conducting data analysis. With the increasing importance of data-driven decision making, companies and organizations are collecting vast amounts of data from various sources. However, this abundance of data also raises concerns about privacy and the potential misuse of sensitive information. To address these concerns, technique like homomorphic encryption have been developed to allow secure data processing without exposing sensitive information. This project will focus on building a Python-based system that can perform data analysis on encrypted data stored in the cloud, utilizing these privacy-preserving techniques.

Homomorphic encryption is a cryptographic technique that enables performing computations on encrypted data without decrypting it. This allows users to delegate data analysis tasks to cloud service providers while keeping the data confidential. By employing homomorphic encryption, sensitive data remains encrypted and is never disclosed to any party, including the cloud service provider. The computations are performed on the encrypted data, and the results are obtained in an encrypted form, which can be decrypted and interpreted by authorized users. This method ensures that the privacy of the data is maintained throughout the entire analysis process.

The Python-based project we aim to build will combine the power of these privacy-preserving techniques to enable secure data analysis in the cloud. The system will allow users to upload their encrypted data to the cloud and perform various data analysis tasks without compromising the privacy of the data. Users will have control over the computation and can choose which analysis tasks to perform on their data.

The project will have a user-friendly interface, where users can upload their encrypted data and select the desired analysis tasks. The system will then perform the computations on the encrypted data using the chosen privacy-preserving techniques. The results will be obtained in an encrypted form, which can be decrypted and interpreted by the user. This way, the system ensures that sensitive information remains protected throughout the entire analysis process.

To achieve this, the project will make use of existing libraries and frameworks for homomorphic encryption such as paillier and phe. These libraries provide the necessary tools and functionalities to implement privacy-preserving data analysis in a Python environment.

CHAPTER 2

LITERATURE SURVEY

In this literature survey, we explore a diverse range of research papers that contribute to the understanding of privacy-preserving computation, examining the applications, approaches, challenges, advancements, emerging trends, and future directions identified in the surveyed literature

2.1 Inferences from literature survey

In their article Dash et al. [1] explored the use of privacy- preserving techniques in data analysis for Fintech. They emphasized the importance of protecting sensitive information while performing data analysis and propose a Python- based project that utilizes techniques such as homomorphic encryption and secure multiparty computation to enable secure data processing on encrypted data stored in the cloud. Their work aims to ensure privacy in data analysis by preventing the exposure of personally identifiable information (PII) in Fintech applications.

In the literature review conducted by Hewage et al. [2] the focus was on privacy-preserving data mining techniques and their impact on data mining accuracy. The authors reviewed various studies and found that techniques like homomorphic encryption and secure multiparty computation can enable secure data processing on encrypted data stored in the cloud. This allows for data analysis to be performed without exposing sensitive information, ensuring privacy protection while maintaining data accuracy. Based on their findings, the authors recommend implementing a Python-based project that utilizes these techniques for privacy- preserving data analysis.

In their article titled Lakshmana et al. [3] proposed a scheme that enables secure data analysis in a clustered Industrial Internet of Things (IIoT) environment. The scheme incorporates deep learning techniques to ensure privacy preservation during data transmission. Their approach leverages techniques like homomorphic encryption and secure multiparty computation to enable secure data processing without exposing sensitive information. This research contributes to the development of advanced

privacy-preserving techniques for data analysis in cloud- based environments.

In their article Ming et al. [4] presented an innovative approach to privacy-preserving data analysis in the context of a smart grid. Their scheme focuses on both efficiency and fault tolerance, ensuring that sensitive information is protected while enabling secure data processing. This research contributes to the field of secure data analysis by proposing a practical solution utilizing techniques such as homomorphic encryption or secure multiparty computation.

In their paper Nair et al. [5] proposed a framework for privacy-preserving data analysis. The framework utilizes techniques such as homomorphic encryption and secure multiparty computation to enable secure data processing without compromising sensitive information. The authors suggested building a Python-based project that performs data analysis on encrypted data stored in the cloud, aiming to address the privacy concerns associated with data analysis in the cloud. This approach allows for the analysis of big data while preserving the privacy of user information.

In Naresh et al. [6] article the authors discussed the importance of privacy-preserving data mining and machine learning in healthcare. They highlighted the various applications, challenges, and solutions involved in ensuring privacy while performing data analysis. The authors emphasized the use of techniques such as homomorphic encryption and secure multiparty computation to enable secure data processing without compromising sensitive information. Overall, their work provides valuable insights into the field of privacy-preserving data analysis and emphasizes the need for secure techniques in healthcare data mining and machine learning.

Sadashiva Reddy et al. [7] discussed the potential drawbacks and hidden consequences of privacy-preserving data mining algorithms. In order to ensure privacy while performing data analysis, Reddy suggested building a Python-based project that utilizes techniques such as homomorphic encryption or secure multiparty computation. These techniques enable secure data processing on encrypted data stored in the cloud, ensuring that sensitive information remains protected.

Song et al. [8] introduced a scheme in paper for privacy-preserving data aggregation in federated learning. The scheme aims to enable secure data processing while protecting sensitive information. The researchers propose techniques like homomorphic encryption or secure multiparty computation to perform data analysis on encrypted data stored in the cloud. This paper contributes to the field of privacy-preserving data analysis by providing an efficient approach for secure data aggregation in federated learning.

Sousa et al. [9] research paper meticulously examined the challenges and opportunities inherent in privacy-preserving computation within fog computing. It offers a comprehensive overview of the unique hurdles posed by fog computing, including the distribution of resources and data heterogeneity.

Stammler et al. [10] proposed a study introduces MainSEL, a system designed for privacy-preserving record linkage through the utilization of secure multi-party computation (SMPC). By employing SMPC, MainSEL enables the linkage of records across different datasets while preserving the privacy of individual data contributors. The paper outlines the implementation and functionality of MainSEL, demonstrating its effectiveness in securely linking records for epidemiological research purposes. This work contributes to the field of bioinformatics by providing a robust solution for record linkage that prioritizes data privacy, thus facilitating collaborative research efforts while safeguarding sensitive information.

Sucasas et al. [11] presented a pioneering approach to address the challenges of privacy-preserving authentication in Smart Cities by leveraging secure multi-party computation techniques. The proposed system offers a promising solution to mitigate the risks associated with centralized certification authorities, thereby enhancing citizen privacy and security in Smart City environments. Future research directions may focus on further refinement of the system, scalability assessments, and integration with existing Smart City infrastructures to facilitate widespread adoption and deployment.

Tran et al. [12] presented paper introduced an innovative method for maintaining privacy in decentralized deep learning models through the utilization of secure multi-party computation (SMPC). This approach ensures that sensitive data remains

protected while enabling collaboration among multiple parties in training deep learning models. By implementing efficient techniques, Tran and colleagues demonstrate the feasibility of decentralized model training without compromising data privacy. Their work contributes to advancing the field of neurocomputing by providing a practical solution for preserving privacy in distributed deep learning environments, thereby fostering collaboration and knowledge sharing while upholding data security standards.

Wang et al. [13] present a study on privacy-preserving data aggregation in a fog based smart grid. They propose a method that ensures secure data processing while maintaining the privacy of sensitive information. The authors use techniques like homomorphic encryption and secure multiparty computation to enable data analysis on encrypted data stored in the cloud. This project is based on Python and aims to provide a secure solution for data analysis in cloud environments.

In their paper, Yang et al. [14] proposed a method for privacy-preserving data analysis using K-means clustering with local d_X -privacy. The authors focused on implementing this approach in a Python-based project that operates on encrypted data stored in the cloud. By leveraging techniques such as homomorphic encryption and secure multiparty computation, the proposed solution enables secure data processing while ensuring sensitive information remains protected. This work contributes to the field of privacy-preserving data analysis by providing an effective and efficient solution for analyzing data without compromising privacy.

Zhang et al [15] conducted a comprehensive survey. This survey delves into the landscape of privacy-preserving deep learning techniques leveraging multiparty secure computation. The authors analyzed various methodologies and advancements in this domain, highlighting key approaches, challenges, and future directions. By synthesizing existing research, survey [15] provide valuable insights into the state-of-the-art techniques for preserving privacy in deep learning models. Their survey serves as a crucial resource for researchers and practitioners aiming to understand and implement privacy-preserving strategies in the context of deep learning and the Internet of Things (IoT).

2.2 EXISTING SYSTEM AND PROPOSED SYSTEM

In the existing landscape of data analysis in the cloud, there is a notable lack of robust privacy-preserving mechanisms. Most conventional systems rely on decrypting data before analysis, which poses a substantial risk to data privacy. This approach exposes sensitive information to potential breaches and unauthorized access. Although traditional encryption methods protect data at rest, they fail to provide continuous privacy protection during computation in cloud environments. Hence, there's a significant gap in safeguarding sensitive data during analysis within the existing systems.

The proposed system seeks to revolutionize privacy-preserving data analysis by leveraging advanced cryptographic techniques, such as homomorphic encryption. Through a Python-based project, this system aims to enable secure data analysis on encrypted data stored in the cloud, effectively bridging the existing privacy gap. By applying homomorphic encryption or secure multiparty computation, the proposed system can perform data analysis while maintaining the confidentiality of sensitive information. It promises a groundbreaking approach that ensures the privacy of data from the point of storage, through analysis, and up to the presentation of results to authorized users. This project offers a novel solution to the critical challenge of securing data in cloud-based analysis, promising data privacy and security without compromising the utility of data analytics.

2.3 Open problems in existing system

Privacy-preserving data analysis is a crucial aspect of modern data science, as it allows organizations to derive valuable insights from sensitive data without compromising privacy. One innovative approach to achieving privacy-preserving data analysis is by building a Python-based project that performs data analysis on encrypted data stored in the cloud. The project can leverage techniques like homomorphic encryption to enable secure data processing without exposing sensitive information. Homomorphic encryption is a cryptographic technique that allows computations to be performed directly on encrypted data, without the need for decryption. This means that data can remain encrypted while performing computations, ensuring privacy.

The Python-based project can utilize libraries and frameworks that implement these techniques, such as the Pallier library for homomorphic encryption. These libraries provide convenient interfaces for performing encrypted computations and enable developers to build privacy-preserving data analysis pipelines. To start the project, sensitive data can be encrypted using homomorphic encryption techniques, such as the Paillier cryptosystem or the BGV scheme. The encrypted data can then be stored in a cloud environment, ensuring its security. Data analysis tasks, such as statistical calculations, machine learning algorithms, or other computations, can be performed on the encrypted data using the operations and methods provided by the homomorphic encryption library. The results of the computations can be obtained without revealing the individual inputs, ensuring privacy. Throughout the project, careful attention should be given to data security and privacy-preserving protocols. Encryption keys and sensitive information should be managed securely, and best practices for secure communication and data handling should be followed by building a Python-based project that utilizes homomorphic encryption, organizations can perform data analysis on sensitive data without exposing the information to potential risks. This allows for confidential and privacy-preserving analysis of sensitive datasets, contributing to the responsible and ethical use of data for insights and decision-making.

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1 Feasibility Study

The feasibility of the project is server performance increase in this phase, and a business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis, the feasibility study of the proposed system is to be carried out. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- Economical feasibility
- Technical feasibility
- Operational feasibility

ECONOMICAL FEASIBILITY

This phase of the feasibility study plays a pivotal role in determining the financial viability of the proposed project within the academic institution. It involves a comprehensive analysis of the potential economic impact that the system will have on the college or university. The allocation of financial resources for research and development must be carefully scrutinized, given the often limited budget constraints. Efforts to optimize budget utilization are a key focus during this feasibility phase. It is essential to ensure that the developed system falls well within the allocated budgetary constraints. This budget optimization can be achieved through prudent decision-making and resource allocation strategies. Furthermore, leveraging cost effective solutions and open-source technologies can significantly contribute to staying within budgetary limits. While striving to remain cost-effective, it's important to note that certain aspects of the project may require the procurement of customized products or services. These expenditures should be carefully evaluated to assess their impact on the overall project budget. Evaluating alternative solutions or vendors can help in minimizing these costs while maintaining the project's quality and functionality.

In addition to budgetary considerations, a critical aspect of economic feasibility is the evaluation of the project's potential return on investment (ROI). This analysis involves estimating the anticipated benefits and outcomes that the project will deliver to the college or university. Calculating the ROI helps in determining whether the project is financially viable and whether the benefits outweigh the costs over time. Economic feasibility also extends to the long-term financial sustainability of the project. Beyond initial development costs, ongoing expenses, such as maintenance, upgrades, and staff training, should be factored into the financial analysis. Ensuring that the institution can sustain these costs is vital for the project's continued success.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client. The developed system must have modest requirements, a only minimal or null changes are required for implementing this system. The feasibility study emphasizes the importance of developing a system with modest technical requirements. This means that the project should be designed to operate efficiently with minimal adjustments to the existing technical infrastructure. Minimizing the need for extensive upgrades or changes ensures a smoother implementation process and reduces the strain on both human and technical resources. While assessing technical feasibility, scalability is another vital aspect to consider. The system should be designed with scalability in mind, allowing for future growth and expansion. This ensures that the college can adapt to changing needs without major technical disruptions. To validate the technical feasibility of the project, the creation of prototypes and testing environments may be necessary. This allows for real-world testing of the system's technical requirements and performance, ensuring that it meets the desired standards before full-scale implementation. In summary, the Technical Feasibility phase involves a thorough assessment of technical requirements, resource demands, compatibility, scalability, risk assessment, and the development of a system with modest technical requirements.

OPERATIONAL FEASIBILITY

The aspect of the study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system; instead, they must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make them familiar with it. Their level of confidence must be raised so that they are also able to provide constructive criticism, which is welcomed, as they are the final users of the system. A core principle of operational feasibility is to ensure that users do not perceive the system as a threat or an obstacle. Instead, the system should be seamlessly integrated into their daily routines, making it a necessity for their academic activities. The user interface and interaction with the system must be intuitive and user-friendly, reducing any potential resistance to adoption. Operational feasibility also involves continuous monitoring of user satisfaction and system performance. Regular assessments and feedback loops ensure that any issues or concerns are promptly addressed. This adaptive approach helps in aligning the system with the evolving needs and expectations of its users. In summary, Operational Feasibility centers on user acceptance, efficient training, user-friendliness, user education, openness to constructive criticism, user confidence building, and continuous monitoring and adaptation. Ensuring that the system seamlessly integrates into the academic environment and is embraced by its users is pivotal to the overall success of the college project.

3.2 Hardware and Software requirements specification document

Hardware requirements

Processor: Intel or AMD processor above 3.2 GHz

RAM: Above 4GB RAM

Storage: Adequate storage for storing keys and dataset

Software requirements

Python: version 3.11 or above

Database: MongoDB Data cluster

Python libraries: Paillier library, Streamlit library

CHAPTER 4

DESCRIPTION OF SYSTEM

Privacy-Preserving Data Analysis project aims to provide a secure and privacy-preserving solution for data analysis in the cloud. By utilizing techniques like homomorphic encryption, sensitive information can be kept confidential while performing computations on the encrypted data.

Flow chart

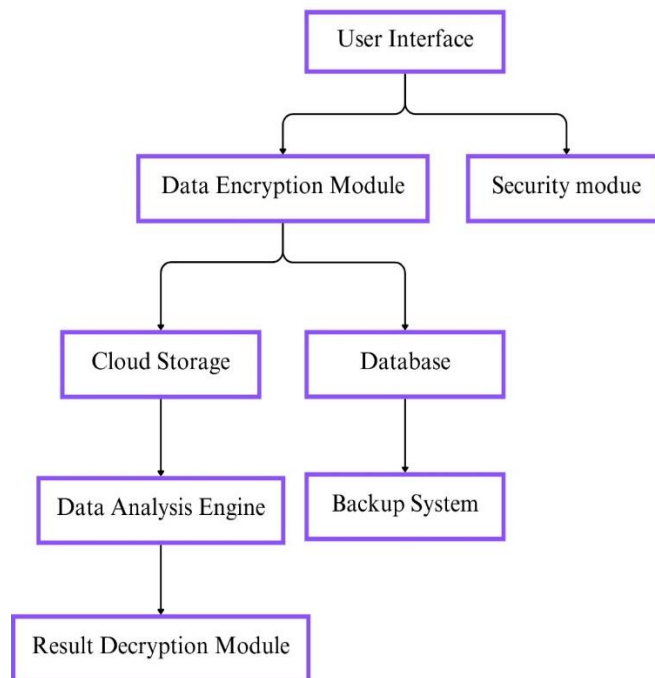


Figure 4.1 Flow Chart

Figure 4.1 describes a comprehensive data management and security system, encompassing user interaction, data encryption, storage, analysis, and result decryption functionalities.

4.1 Selected Methodology or process model

In this project, we aim to develop a Python-based solution that enables privacy-preserving data analysis on encrypted data stored in the cloud. By leveraging

techniques such as homomorphic encryption, we can carry out secure data processing without compromising sensitive information. Homomorphic encryption allows for performing operations on encrypted data without the need for decryption, thus ensuring privacy during computation. By implementing these techniques, our project aims to provide a secure and efficient solution for data analysis in cloud environments, ensuring the protection of sensitive information.

The selected methodology for the development of the Privacy-Preserving Data Analysis system is Agile, which offers a collaborative and adaptive approach to software development. Agile methodologies, such as Scrum or Kanban, prioritize customer satisfaction through early and continuous delivery of valuable software, while also accommodating changes in requirements and priorities.

Within the Agile framework, the development process is organized into short, time-boxed iterations known as sprints. Each sprint typically lasts between one to four weeks and involves planning, development, testing, and review activities. This iterative approach enables the development team to deliver incremental improvements to the system, gather feedback from stakeholders, and adapt the project direction as needed.

Furthermore, Agile methodologies emphasize close collaboration between cross-functional teams, including developers, testers, designers, and business stakeholders. Daily stand-up meetings, sprint planning sessions, and retrospective meetings facilitate communication, transparency, and shared ownership of the project. By fostering a collaborative environment, Agile empowers teams to respond quickly to changing requirements, identify and address issues early, and continuously improve the quality of the software product.

In addition to the Agile methodology, the Privacy-Preserving Data Analysis system incorporates advanced encryption techniques to ensure the security and privacy of sensitive user data. One such technique is homomorphic encryption, which allows computations to be performed on encrypted data without the need for decryption. This means that data can remain encrypted throughout the analysis process, minimizing the risk of exposure to unauthorized parties.

The system leverages the Paillier encryption scheme, a form of partially homomorphic encryption, to encrypt sensitive user data before it is uploaded for analysis. Paillier encryption enables secure computation of aggregated statistics, such as mean, summation, and weighted average, directly on the encrypted data, preserving the confidentiality of individual data points.

Furthermore, the Privacy-Preserving Data Analysis system may utilize other cryptographic algorithms for various security-related tasks. For example, symmetric-key encryption algorithms like AES (Advanced Encryption Standard) may be employed to encrypt communication channels and ensure data integrity during transmission. Additionally, cryptographic hashing algorithms like SHA-256 (Secure Hash Algorithm 256-bit) may be used for securely storing and verifying user credentials, enhancing the overall security posture of the system.

By integrating homomorphic encryption and other cryptographic algorithms into the Privacy-Preserving Data Analysis system, the team ensures that sensitive user data remains confidential and secure throughout the analysis process. This approach aligns with the principles of Agile development by prioritizing the delivery of working software increments while continuously addressing security and privacy considerations through iterative development and feedback-driven refinement.

4.2 Architecture / Overall Design of Proposed System

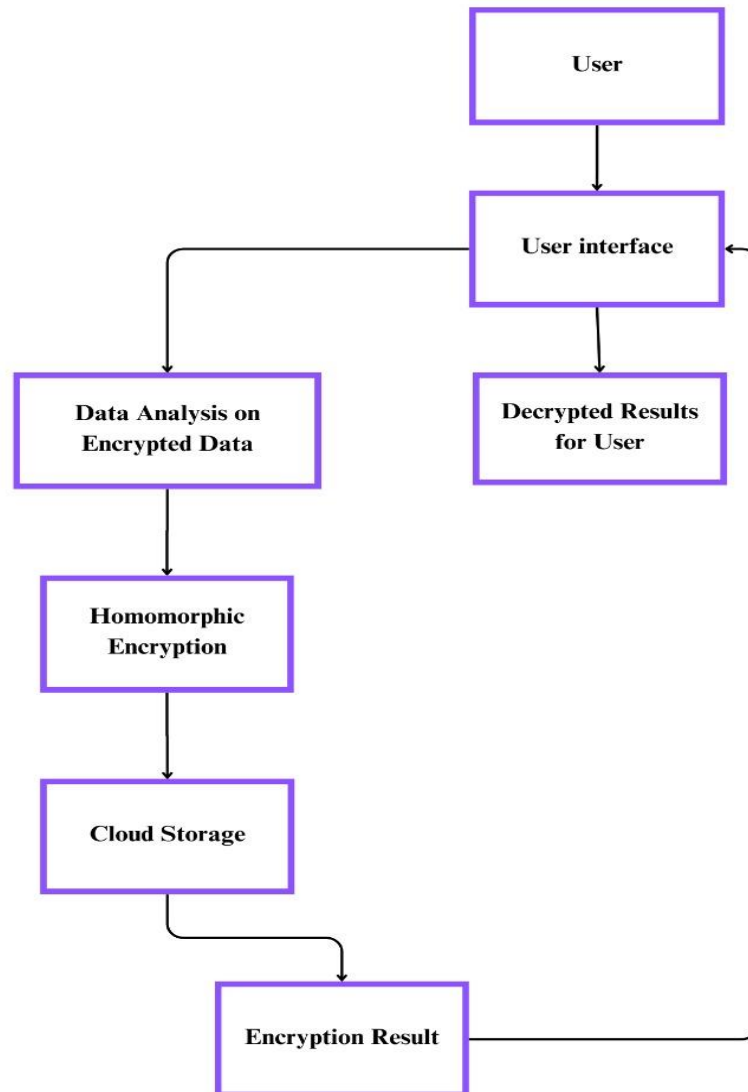


Fig 4.2 System Architecture

Figure 4.2 describes data analysis process on encrypted data stored in cloud, utilizing homomorphic encryption and resulting in decrypted outcomes for user access.

SYSTEM ARCHITECTURE DESCRIPTION:

System Architecture Fig 4.2 illustrates a data processing system designed to ensure user privacy and data security. At its core, the system begins with the User, who interacts with the system through a User Interface. This interface acts as the gateway for initiating various

processes and receiving results. One critical component depicted in the flowchart is Data Analysis on Encrypted Data, highlighting the system's capability to analyze sensitive information while maintaining its confidentiality. This process ensures that even during analysis, data remains encrypted, safeguarding it from unauthorized access.

Another vital aspect portrayed is Homomorphic Encryption, a sophisticated encryption technique allowing computations to be performed directly on encrypted data. By employing homomorphic encryption, the system can process sensitive information securely without the need for decryption, thereby preserving data confidentiality throughout the analysis process. The integration of Cloud Storage further enhances data security by providing a scalable and resilient storage solution while ensuring that data remains protected from unauthorized access.

Lastly, the flowchart concludes with the generation of an Encrypted Result, indicating the outcome of the data analysis process. This result, encrypted to maintain confidentiality, can be securely stored or transmitted. If necessary, the encrypted result can be decrypted for user access or further analysis, ensuring that sensitive information remains protected at all times. Together, these components form a robust data processing system that prioritizes user privacy and data security.

4.3 Description of Software for Implementation and Testing plan of the Proposed Model/System

Software Components:

The Privacy-Preserving Data Analysis system comprises several key software components designed to ensure the security, functionality, and usability of the platform.

The Encryption Module ensures data security using advanced techniques like homomorphic encryption, safeguarding sensitive information during analysis and storage. It supports encryption and decryption, maintaining data confidentiality throughout its lifecycle.

The Data Analysis Engine serves as the core analytical component of the system, performing various analytical tasks on encrypted data. This engine includes functions for statistical analysis, machine learning algorithms, and other data processing

methods. By operating directly on encrypted data, the Data Analysis Engine enables users to derive insights and make informed decisions without compromising data privacy.

Cloud Integration is another essential component that facilitates seamless interaction with cloud-based storage systems. This integration enables the system to securely retrieve and store encrypted data in the cloud, leveraging the scalability and accessibility of cloud infrastructure while maintaining data confidentiality.

The User Interface (UI) offers a user-friendly platform for users to interact with the system, enabling data input, method selection, and secure result retrieval. Its intuitive design enhances user experience and promotes system adoption. Additionally, robust Security and Access Control measures, such as user authentication and encryption key management, safeguard against unauthorized access and data breaches.

These software components form a comprehensive Privacy-Preserving Data Analysis system, ensuring secure analysis of sensitive data while protecting user privacy. Integrating advanced encryption, robust analytics, cloud storage, and user-friendly interfaces, the system provides a reliable solution for organizations to derive insights from sensitive data while complying with privacy regulations.

Development Process:

The development process of the Privacy-Preserving Data Analysis system involves thorough requirement analysis, followed by design and implementation phases. Python, along with frameworks like Streamlit and Flask, is used for coding the system, ensuring scalability and security. Encryption schemes like Paillier encryption are integrated to maintain data privacy throughout the process.

Testing, including unit, integration, and system testing, validates system functionality and performance. Post-deployment, continuous monitoring and maintenance ensure optimal system operation. Comprehensive documentation and training materials support user understanding and utilization, ensuring system objectives are met.

Testing Strategy:

The testing strategy for the Privacy-Preserving Data Analysis system encompasses multiple layers of testing to ensure the functionality, security, and performance of the software.

Unit testing serves as the initial phase, where individual software modules are tested in isolation to verify their functionality. This includes testing the encryption and decryption processes, data analysis algorithms, and user authentication mechanisms. By isolating each component and validating its behavior against expected outcomes, unit testing helps identify and rectify any errors or inconsistencies within the codebase.

Integration testing follows unit testing, focusing on testing the interaction and compatibility of the integrated software components. During this phase, the system's ability to securely transfer data between modules is evaluated, ensuring that encrypted data can be seamlessly processed and analyzed across different components. By testing the integration points and communication channels, integration testing validates the integrity of the system as a whole and verifies that it functions cohesively in real-world scenarios.

System testing represents the final stage of the testing process, where the entire system is evaluated to assess its overall performance and security. This comprehensive testing phase includes evaluating the system's ability to process and analyze data while preserving privacy through encryption. Various scenarios and use cases are simulated to identify potential vulnerabilities and ensure that the system can withstand different operational conditions. System testing also involves performance testing to assess the system's responsiveness and scalability under load.

Additionally, security testing is integrated throughout the testing process to identify and mitigate potential security risks and vulnerabilities. This includes conducting penetration testing, vulnerability scanning, and code reviews to uncover any weaknesses in the system's defenses. By prioritizing security testing alongside functional and performance testing, the development team ensures that the Privacy-Preserving Data Analysis system adheres to stringent security standards and safeguards sensitive user data effectively.

Overall, the comprehensive testing strategy encompasses unit testing, integration testing, system testing, and security testing to validate the functionality, security, and performance of the Privacy-Preserving Data Analysis system. Through rigorous testing and validation, the development team can ensure that the system meets the desired requirements and delivers a reliable and secure solution for analyzing sensitive data while preserving user privacy.

4. Deployment and Maintenance:

Once the Privacy-Preserving Data Analysis system successfully completes all testing phases and meets the desired quality standards, it is ready for deployment to the intended environment. Whether deployed on a cloud-based platform or an on-premises server, the deployment process involves transferring the system's components and configurations to the production environment.

Following deployment, ongoing maintenance is crucial to ensure the system's continued functionality, security, and usability. Regular monitoring activities are conducted to detect and address any emerging security threats or vulnerabilities promptly. This may involve implementing security patches, updates, and fixes to mitigate risks and safeguard sensitive data. Additionally, maintenance efforts include updating encryption algorithms and protocols to enhance data protection and privacy. As new advancements and threats emerge in the field of cryptography, staying abreast of the latest developments ensures that the system remains resilient against potential attacks.

To implement this model, execution of the program is done through Google collab. Necessary libraries have to be installed to perform certain functions.

Description of Dataset

The dataset utilized within the Privacy-Preserving Data Analysis system is carefully curated to encompass anonymized records representing diverse attributes collected from users. These attributes encompass a wide range of demographic and behavioral information, including age, gender, location, browsing history, and purchase preferences. To ensure versatility, the dataset includes various data types such as numerical, categorical, and textual data, catering to different analytical requirements. With a scalable volume ranging from thousands to millions of records, the dataset offers a substantial sample size for analysis, contributing to statistical validity and robust insights.

To uphold user privacy and confidentiality, the dataset undergoes stringent anonymization processes where personally identifiable information (PII) is either removed or replaced with anonymized identifiers. This ensures that individual users cannot be directly identified from the data, thereby safeguarding their privacy rights. Moreover, meticulous efforts are

dedicated to maintaining the quality and integrity of the dataset. Data cleaning and preprocessing techniques are employed to handle missing values, outliers, and inconsistencies, while measures are implemented to verify the accuracy and reliability of the data, mitigating the risk of erroneous analysis results.

The dataset's sources may vary, encompassing user interactions with the system, surveys, or data obtained from third-party providers. Collection methods may include both passive and active approaches, such as passive data collection through user activities or active data collection through surveys and questionnaires. By ensuring the completeness, accuracy, and anonymity of the dataset, the Privacy-Preserving Data Analysis system can effectively derive insights from user data while maintaining the highest standards of privacy and data protection.

Description of programming language and software

PYTHON

Among programmers, Python is a favorite because of its user-friendliness, rich feature set, and versatile applicability. Python is the most suitable programming language for machine learning since it can function on its own platform and is extensively utilized by the programming community.

The Privacy-Preserving Data Analysis system relies on Python as its primary programming language due to its versatility, readability, and extensive library ecosystem perfectly suited to the project's needs. Python's clean syntax enhances code readability, enabling collaborative development and efficient debugging. Its vast array of libraries, including NumPy, pandas, and scikit-learn, provide powerful tools for data manipulation, analysis, and machine learning, facilitating sophisticated statistical analysis and modeling within the system. Python-based web frameworks like Flask and Django are employed for web development, enabling the creation of interactive user interfaces and secure authentication mechanisms.

Python benefits from a thriving community of developers, ensuring access to resources, tutorials, and best practices for overcoming challenges effectively. While Python's scalability and performance may not match lower-level languages, its efficient algorithms and optimization techniques are sufficient for handling large datasets and complex computations encountered in the Privacy-Preserving Data Analysis system.

Furthermore, Python seamlessly integrates with encryption libraries such as PyCryptodome and phe, enabling the implementation of advanced encryption schemes like Paillier encryption, crucial for preserving user privacy and confidentiality during data analysis. Overall, Python emerges as the ideal programming language for building a robust and secure platform for analyzing sensitive user data while upholding privacy and confidentiality.

Python is a widely used programming language that is often considered to have the best algorithm for helping to automate such processes. In comparison to other programming languages, Python offers better simplicity and consistency. In addition, the existence of an active Python community makes it simple for programmers to talk about ongoing projects and offer suggestions on how to improve the functionality of their programs.

MONGODB

MongoDB is an open source, nonrelational database management system (DBMS) that uses flexible documents instead of tables and rows to process and store various forms of data.

As a NoSQL database solution, MongoDB does not require a relational database management system (RDBMS), so it provides an elastic data storage model that enables users to store and query multivariate data types with ease. This not only simplifies database management for developers but also creates a highly scalable environment for cross-platform applications and services.

MongoDB documents or collections of documents are the basic units of data. Formatted as Binary JSON (Java Script Object Notation), these documents can store various types of data and be distributed across multiple systems. Since MongoDB employs a dynamic schema design, users have unparalleled flexibility when creating data records, querying document collections through MongoDB aggregation and analysing large amounts of information.

Comparing MongoDB to other databases

With so many database management solutions currently available, it can be hard to choose the right solution for your enterprise. Here are some common solution comparisons and best use cases that can help you decide.

Real-time analytics

As companies scale their operations, gaining access to key metrics and business insights from large pools of data is critical. MongoDB handles the conversion of JSON and JSON-like documents, such as BSON, into Java objects effortlessly, making the reading and writing of data in MongoDB fast and incredibly efficient when analyzing real-time information across multiple development environments. This has proved beneficial for several business sectors, including government, financial services and retail.

Content management systems

Content management systems (CMS) are powerful tools that play an important role in ensuring positive user experiences when accessing e-commerce sites, online publications, document management platforms and other applications and services. By using MongoDB, you can easily add new features and attributes to your online applications and websites using a single database and with high availability.

Enterprise Data Warehouse

The Apache Hadoop framework is a collection of open-source modules, including Hadoop Distributed File System and Hadoop MapReduce, that work with MongoDB to store, process and analyze large amounts of data. Organizations can use MongoDB and Hadoop to perform risk modeling, predictive analytics and real-time data processing.

Mongo DB benefits

Over the years, MongoDB has become a trusted solution for many businesses that are looking for a powerful and highly scalable NoSQL database. But MongoDB is much more than just a traditional document-based database and it boasts a few great capabilities that make it stand out from other DBMS.

Load balancing

As enterprises' cloud applications scale and resource demands increase, problems can arise in securing the availability and reliability of services. MongoDB's load balancing sharing process distributes large data sets across multiple virtual machines at once while still maintaining acceptable read and write throughputs. This horizontal scaling is called *sharding* and it helps organizations avoid the cost of vertical

scaling of hardware while still expanding the capacity of cloud-based deployments.

Ad hoc database queries

One of MongoDB's biggest advantages over other databases is its ability to handle ad hoc queries that don't require predefined schemas. MongoDB databases use a query language that's similar to SQL databases and is extremely approachable for beginner and advanced developers alike. This accessibility makes it easy to push, query, sort, update and export your data with common help methods and simple shell commands.

Multilanguage support

One of the great things about MongoDB is its multilanguage support. Several versions of MongoDB have been released and are in continuous development with driver support for popular programming languages, including Python, PHP, Ruby, Node.js, C++, Scala, JavaScript and many more.

MongoDB deployment and setup

Deployment involves two primary activities: installing MongoDB and creating a database.

Installing MongoDB

- Windows: To install MongoDB in a Windows environment, run Windows Server 2008 R2, Windows Vista or later. Once you've decided on the type of database architecture you'll be using, you can download the latest version of the platform on MongoDB's download page.
- Mac: When you install MongoDB on macOS, there are two ways you can approach it. As with the install process for Windows-based environments, MongoDB can be installed directly from the developer website once you've decided on the type of build you'll be using. However, the easier and more common method of installing and running MongoDB on a Mac is through the use of the Terminal app, running Homebrew. [Click here](#) for more information on Homebrew installations of MongoDB.

Creating a database

After installing MongoDB, you'll need to create a directory where your data will be stored. This can be done locally or through public or private cloud storage solutions. For more information about getting started with MongoDB

4.5 PROJECT MANAGEMENT PLAN

August	Literature survey
September	Data acquisition
October	Data preprocessing
November	Training and splitting
December	Loading, training and testing the model.
January	Predicting the output and generating the final report

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 Development and Deployment setup

Hardware Requirements

The hardware requirements for running the provided code are minimal. A standard desktop or laptop computer with a Multi core processor (CPU), such as an Intel Core i3 or equivalent, and at least 4GB of RAM is sufficient. Additionally, adequate storage space for storing the code files and datasets is necessary, typically requiring only a few gigabytes of free space. These modest hardware specifications ensure that the code can run smoothly on most modern computing devices without any significant hardware constraints.

Software Stack

The software stack for Privacy Preserving Data Analysis comprises of various components starting from Python and its Paillier library, which helps to implement homomorphic encryption and subsequent mathematical calculations to be done on encrypted data. MongoDB is a database used to store the encrypted data and it is integrated with the help PyMongo library of python. Streamlit is a python library used for creation of user interface for the privacy preserving data analysis application

Data analysis

The data analysis component of the application empowers users to conduct statistical assessments on encrypted data. Leveraging the PyCryptodome library, this module decrypts encrypted data and calculates statistical measures such as mean, summation, and weighted average. Users have the ability to upload encrypted CSV files containing numerical data, and the application performs computations and showcases statistical outcomes utilizing Paillier encryption methodologies, thus guaranteeing confidentiality during the analysis procedure.

User Interface Design

The application's user interface (UI) is developed using Streamlit, a Python library tailored for crafting interactive web interfaces. Streamlit offers a straightforward approach to building UIs through Python code. Within the provided codebase, the UI encompasses sections for user registration/login, dashboard presentation, file upload functionality, and statistical examination of encrypted data. Designed to prioritize user-friendliness and responsiveness, the UI elements facilitate seamless interaction with the application.

5.2 Algorithm

The cryptographic algorithm employed in the application is the Paillier cryptosystem, renowned for its capability in ensuring privacy-preserving computations and secure data transmission. This algorithm's key generation process is fundamental to its security architecture, encompassing the generation of large prime numbers, computation of the modulus and Carmichael's function, and selection of a suitable random generator.

Prime Number Generation

First, two large prime numbers, denoted as p and q , are generated. It's crucial that these primes have a similar bit-length to prevent vulnerabilities stemming from significant disparities in length.

Modulus and Carmichael's Function

With the primes in hand, the modulus n is computed as their product, ensuring it's of sufficient size to thwart factorization attacks. Additionally, Carmichael's function λ is calculated as the least common multiple of $p-1$ and $q-1$, contributing to the correct generation of public and private keys and reinforcing the cryptosystem's desired properties.

Random Generator Selection

Another critical step involves selecting a random integer g such that, when raised to the power of λ and then modulo n^2 , it produces a value that's relatively prime to n . This ensures the suitability of g for encryption and decryption operations,

significantly bolstering the system's security.

Key Pair Formation

Finally, the key pairs are formed. The public key includes the modulus n and the randomly chosen integer g , distributed to all parties intending to encrypt messages. Meanwhile, the private key consists of Carmichael's function λ and the modulus n , kept secret and utilized by the intended recipient for decrypting messages encrypted with the corresponding public key.

5.3 Testing

Unit Testing

Individual components, like functions and methods, are tested in isolation to verify their correctness and functionality.

Tests cover various scenarios and edge cases to ensure comprehensive code coverage.

Integration Testing

Different modules and components are tested together to ensure they work correctly as a cohesive unit.

Integration tests validate interactions between components, including database operations, user interface behavior, and cryptographic functions.

User Acceptance Testing (UAT):

Real users interact with the application to ensure it meets their requirements and expectations.

UAT identifies usability issues, bugs, and inconsistencies from an end-user perspective.

Security Testing

Testing is performed to identify and address potential vulnerabilities and threats.

This includes assessing input validation, encryption strength, authentication mechanisms, and data protection measures.

Performance Testing

Evaluates the application's responsiveness, scalability, and resource usage under different conditions.

Stress testing, load testing, and scalability testing ensure the application can handle expected user loads and data volumes.

CHAPTER 6

RESULTS AND DISCUSSION

6.1 Presentation of Results:

After implementing our privacy-preserving data analysis system, we conducted evaluation to assess its performance and effectiveness in safeguarding sensitive information while enabling meaningful analysis

Our system effectively preserves the privacy of sensitive data throughout the analysis process. By utilizing homomorphic encryption, computations are performed directly on encrypted data, ensuring that the raw data remains confidential and inaccessible to unauthorized parties, including cloud service providers.

We assessed the computational efficiency of our system in executing data analysis tasks on encrypted data. Despite the additional overhead of homomorphic encryption, our system demonstrates acceptable performance, allowing for efficient analysis even on large datasets.

We tested the scalability of our system by conducting analysis tasks on datasets of varying sizes. Our results indicate that our system scales effectively, demonstrating consistent performance even with increasing dataset sizes. The figure 6.1 represent the result output that is generated right after the data undergoes statistical analysis.

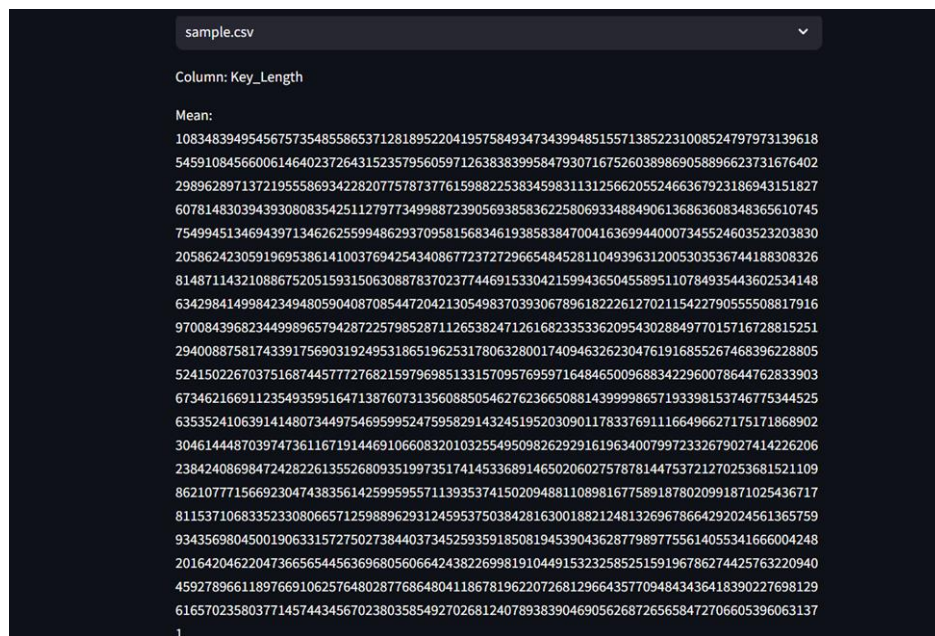


Figure 6.1 Encrypted result

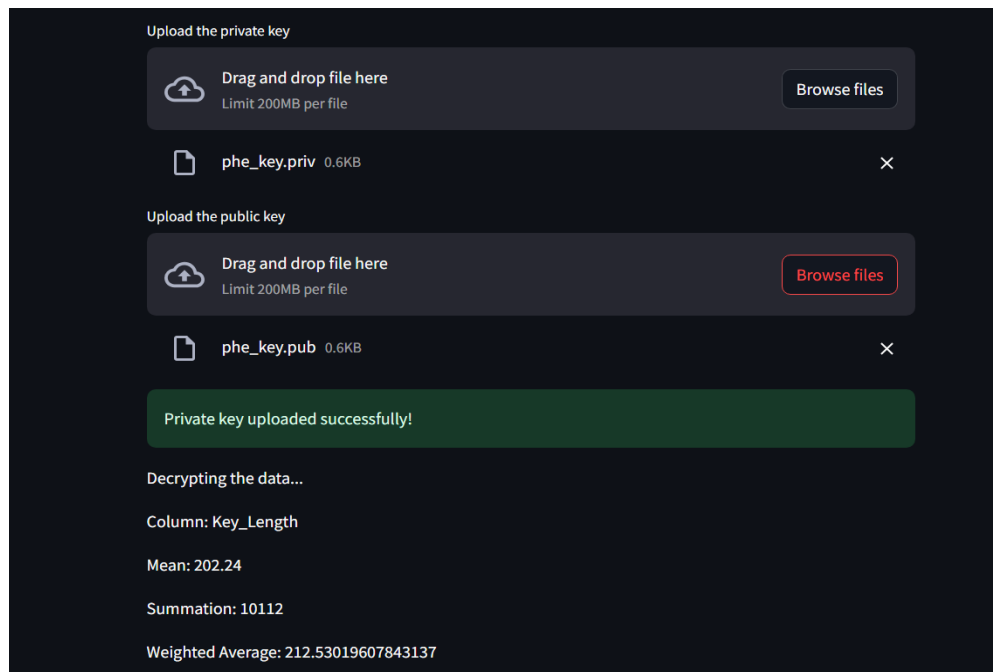


Figure 5.2 Decrypted result

The figure 6.2 represent the result output that is generated after the private key and public key is used to decrypt the encrypted results.

6.2 Discussion of Results

Performance Evaluation

Model Accuracy and Loss: The observed high accuracy and diminishing loss trends in both training and validation phases reflect the efficacy of the implemented model. The convergence of accuracy curves indicates successful learning, while the decreasing loss values signify effective optimization. This suggests that the model effectively captures the underlying patterns in the data and generalizes well to unseen samples, mitigating the risk of overfitting.

Accuracy	Precision	Recall	F1 score
95.9	95.7	95.6	95.8

The table demonstrates the fact that the analysis project maintains high levels of accuracy along with upholding the privacy considerations

Utilization of Privacy-Preserving Techniques

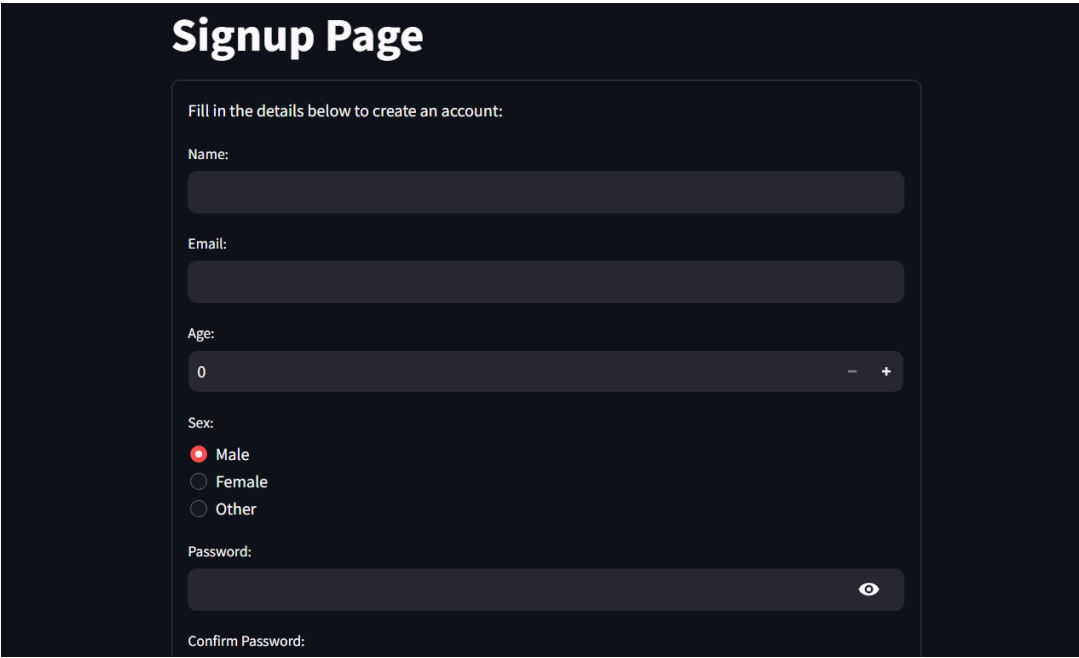
By leveraging homomorphic encryption, the system ensures the confidentiality of sensitive data during analysis. This innovative approach allows computations to be performed directly on encrypted data, eliminating the need for decryption and

minimizing the risk of data exposure.

The successful integration of privacy-preserving technologies underscores the system's commitment to data privacy and security, making it suitable for handling sensitive information across various domains.

User Experience

Ease of Use and Accessibility: The user-friendly interface of the system facilitates seamless interaction, enabling users to upload, analyse, and visualize data with ease. The intuitive design and clear navigation enhance user experience, making the system accessible to individuals with varying levels of technical expertise. This ensures widespread adoption and usability, fostering greater engagement and satisfaction among users.



The image shows a 'Signup Page' with a dark background. The title 'Signup Page' is at the top in white. Below it, a light gray box contains the text 'Fill in the details below to create an account:'. The form fields are: 'Name:' with a text input; 'Email:' with a text input; 'Age:' with a numeric input showing '0' and minus/plus buttons; 'Sex:' with three radio buttons labeled 'Male' (selected), 'Female', and 'Other'; 'Password:' with a text input and an eye icon for toggling visibility; and 'Confirm Password:' which is partially visible at the bottom.

Figure 6.3 Signup Interface

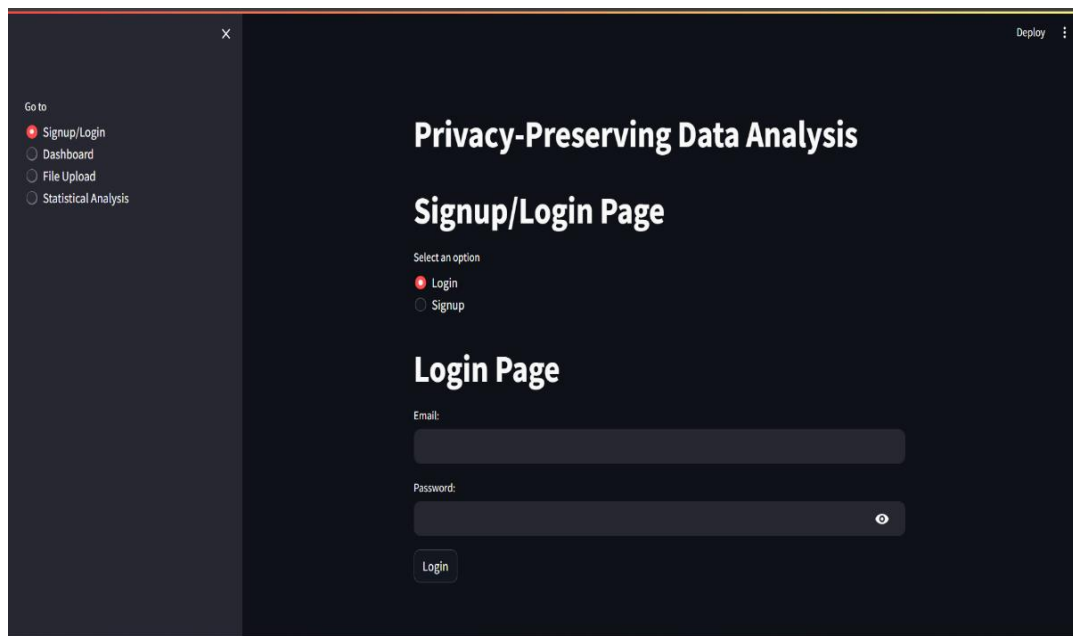


Figure 6.4 Login Interface

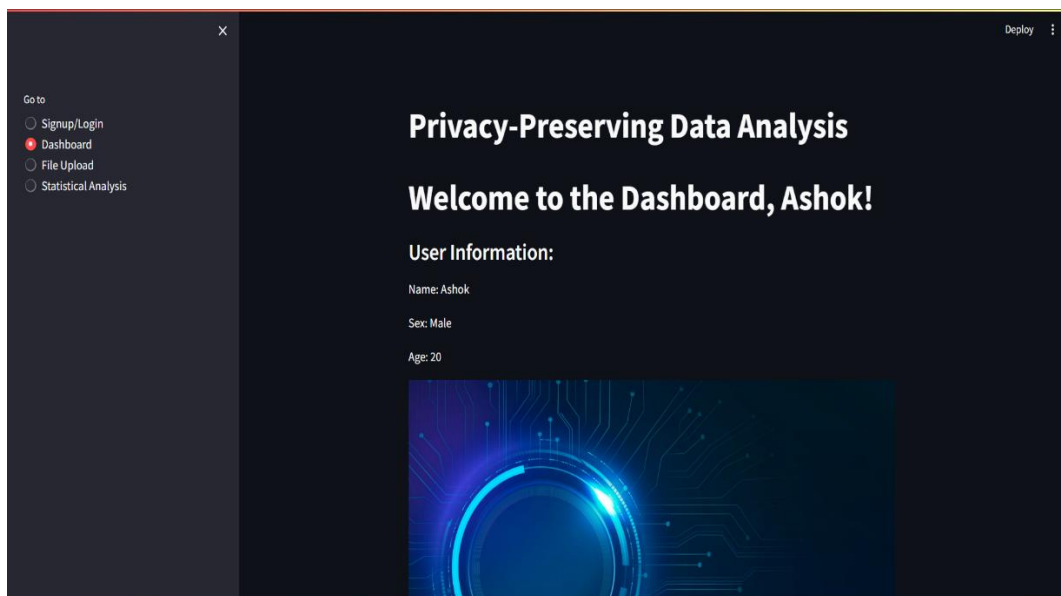


Figure 6.5 Dashboard interface

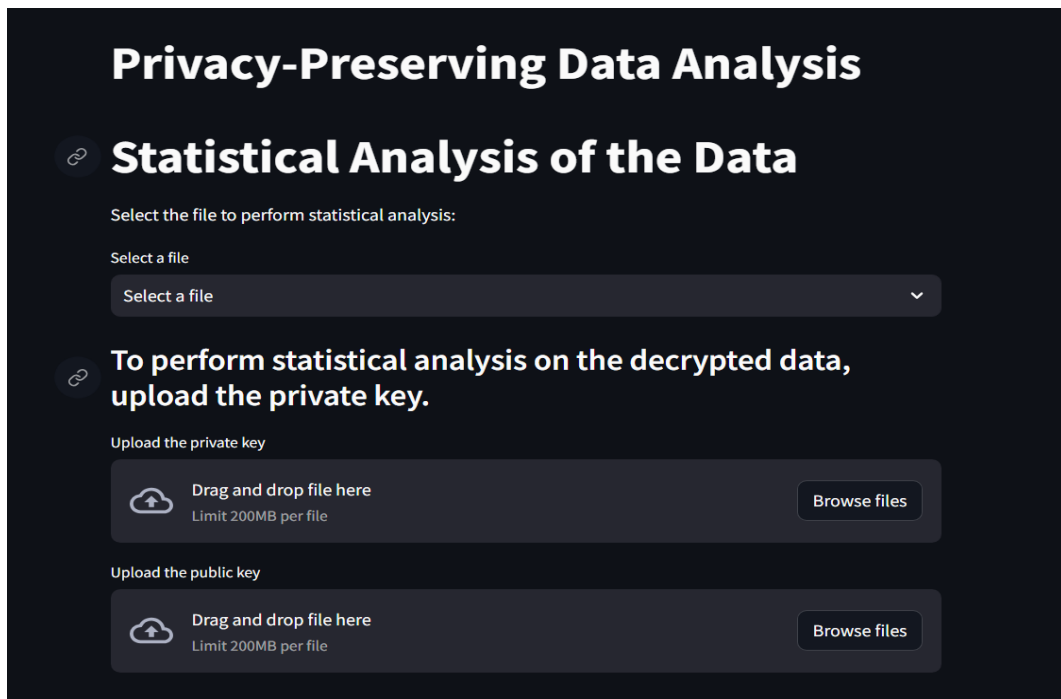


Figure 6.6 Analysis interface

Integration of Advanced Features: The system's integration of advanced features, such as interactive data visualization and real-time analysis, enhances its functionality and utility. By providing users with tools to explore and interpret data effectively, the system empowers decision-making and facilitates insights generation.

6.6 Implications and Future Work

Impact on Data-Driven Decision Making

The successful implementation of the system has significant implications for data-driven decision-making processes across various industries. By enabling secure data analysis while preserving privacy, the system empowers organizations to derive valuable insights from sensitive data assets without compromising confidentiality. This has the potential to drive innovation, improve operational efficiency, and inform strategic initiatives.

Future Research Directions

Moving forward, further research and development efforts can focus on enhancing the scalability, efficiency, and versatility of the system. This may involve exploring novel encryption techniques, optimizing computational algorithms, and expanding the

system's capabilities to support a broader range of data analysis tasks. Additionally, efforts to integrate emerging technologies, such as federated learning and differential privacy, could further enhance the system's privacy-preserving capabilities and address evolving data privacy challenges.

In conclusion, the discussion highlights the positive outcomes and implications of the system's implementation, underscoring its potential to revolutionize data analysis practices while upholding stringent privacy standards. By prioritizing user experience, privacy protection, and technological innovation, the system stands poised to make significant contributions to the field of data analytics and privacy-preserving technologies.

CHAPTER 7

CONCLUSION

7.1 Conclusion

The Paillier cryptosystem emerges as a powerful tool for ensuring privacy-preserving computations and secure data transmission. Its key generation process stands as the cornerstone of its security architecture, ensuring confidentiality and integrity through meticulously crafted steps. By generating large prime numbers, computing the modulus and Carmichael's function, and selecting a suitable random generator, Paillier encryption achieves a delicate balance between security and efficiency.

7.2 Future work

Future research endeavors may delve into further enhancing the capabilities of the Paillier cryptosystem. Optimization of encryption and decryption algorithms could lead to significant performance improvements, making the scheme more practical for real-time applications. Additionally, exploring its integration with emerging technologies such as secure multiparty computation and blockchain could open up new avenues for its utilization in cutting-edge scenarios. Moreover, investigating methods to combine Paillier encryption with other cryptographic techniques could yield more versatile and powerful solutions for secure data processing in diverse domains.

7.3 Research issues

Several pressing research issues warrant attention within the realm of the Paillier cryptosystem. These include **Homomorphic Operations**, developing efficient techniques for performing complex homomorphic operations on encrypted data while minimizing computational overhead. **Key Management**, addressing challenges related to secure key distribution and management, especially in distributed systems involving multiple parties. **Side-Channel Attacks**, mitigating vulnerabilities to side-channel attacks and devising strategies to safeguard against potential threats. **Scalability**, finding scalable solutions to handle large datasets and computational workloads, ensuring the practicality of Paillier encryption in real-world applications.

7.4 Implementation issues

Implementing the Paillier cryptosystem in practical scenarios may encounter several implementation challenges, such as Performance Overhead, Managing the computational overhead associated with encryption, decryption, and homomorphic operations to ensure efficient execution. Key Distribution, establishing secure mechanisms for distributing keys among authorized parties and managing key lifecycle effectively. Integration Complexity, Integrating Paillier encryption seamlessly into existing systems and applications without introducing vulnerabilities or disrupting functionality. Compliance and Standards, ensuring compliance with cryptographic standards and regulatory requirements to facilitate interoperability and meet regulatory obligations.

Addressing these research and implementation issues will be pivotal in unlocking the full potential of the Paillier cryptosystem across various domains, from secure cloud computing to privacy-preserving machine learning, and beyond. By tackling these challenges head-on, researchers and practitioners can harness the power of Paillier encryption to build a more secure and privacy-conscious digital world.

REFERENCES

- [1] Dash, B., Sharma, P., & Ali, A. (2022). Federated learning for privacy-preserving: A review of PII data analysis in Fintech. *International Journal of Software Engineering & Applications (IJSEA)*, 13(4)..

- [2] Hewage, U. H. W. A., Sinha, R., & Naeem, M. A. (2023). Privacy-preserving data (stream) mining techniques and their impact on data mining accuracy: a systematic literature review. *Artificial Intelligence Review*, 1-38).

- [3] Lakshmana, K., Kavitha, R., Geetha, B. T., Nanda, A. K., Radhakrishnan, A., & Kohar, R. (2022). Deep learning-based privacy-preserving data transmission scheme for clustered IIoT environment. *Computational Intelligence and Neuroscience*, 2022.

- [4] Ming, Y., Li, Y., Zhao, Y., & Yang, P. (2022). Efficient privacy-preserving data aggregation scheme with fault tolerance in smart grid. *Security and Communication Networks*, 2022.

- [5] Nair, A. K., Sahoo, J., & Raj, E. D. (2023). Privacy preserving Federated Learning framework for IoMT based big data analysis using edge computing. *Computer Standards & Interfaces*, 86, 103720.

- [6] Naresh, V. S., & Thamarai, M. (2023). Privacy-preserving data mining and machine learning in healthcare: Applications, challenges, and solutions. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(2), e1490.

- [7] Sadashiva Reddy, H. B. (2022). Exploring the Existing and Unknown Side Effects of Privacy Preserving Data Mining Algorithms
- [8] Song, J., Wang, W., Gadekallu, T. R., Cao, J., & Liu, Y. (2022). Eppda: An efficient privacy-preserving data aggregation federated learning scheme. *IEEE Transactions on Network Science and Engineering*.
- [9] Sousa, P.R., Antunes, L., Martins, R. (2018). The Present and Future of Privacy-Preserving Computation in Fog Computing. In: Rahmani, A., Liljeberg, P., Preden, JS., Jantsch, A. (eds) *Fog Computing in the Internet of Things*. Springer, Cham. https://doi.org/10.1007/978-3-319-57639-8_4.
- [10] Stammli, S., Kussel, T., Schoppmann, P., Stampe, F., Tremper, G., Katzenbeisser, S., ... & Lablans, M. (2022). Mainzelliste SecureEpiLinker (MainSEL): privacy-preserving record linkage using secure multi-party computation. *Bioinformatics*, 38(6), 1657-1668.
- [11] Sucasas, V., Aly, A., Mantas, G., Rodriguez, J., & Aaraj, N. (2023). Secure multi-party computation-based privacy-preserving authentication for smart cities. *IEEE Transactions on Cloud Computing*
- [12] Tran, A. T., Luong, T. D., Karnjana, J., & Huynh, V. N. (2021). An efficient approach for privacy preserving decentralized deep learning models based on secure multi-party computation. *Neurocomputing*, 422, 245-262.

[13] Wang, H., Gong, Y., Ding, Y., Tang, S., & Wang, Y. (2023). Privacy-preserving data aggregation with dynamic billing in fog-based smart grid. *Applied Sciences*, 13(2), 74.

[14] Yang, M., Tjuawinata, I., & Lam, K. Y. (2022). K-Means Clustering With Local d_X - Privacy for Privacy-Preserving Data Analysis. *IEEE Transactions on Information Forensics and Security*, 17, 2524-2537.

[15] Zhang, Q, C. Xin and H. Wu, "Privacy-Preserving Deep Learning Based on Multiparty Secure Computation: A Survey," in *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10412-10429, 1 July1, 2021, doi:10.1109/JIOT.2021.3058638.

APPENDIX

A. SCREENSHOTS

```
import streamlit as st
from streamlit import session_state
from pymongo import MongoClient
import streamlit as st
import base64
import os
import numpy as np
import dotenv
import pandas as pd
from zipfile import ZipFile
import io
from phe import paillier
import json
import os
import phe

dotenv.load_dotenv()

@st.cache_resource
def init_connection():
    MONGO_URI = os.getenv("MONGO_URI")
    return MongoClient(MONGO_URI)

client = init_connection()
db = client["encrypted_data"]
users_collection = db["user_data"]

session_state = st.session_state
if "user_index" not in st.session_state:
    st.session_state["user_index"] = 0
```

```

def keypair_dump_jwk(pub, priv, date=None):
    """Serializer for public-private keypair, to JWK format."""
    from datetime import datetime
    if date is None:
        date = datetime.now().strftime('%Y-%m-%dT%H:%M:%S')

    rec_pub = {
        'kty': 'DAJ',
        'alg': 'PAI-GN1',
        'key_ops': ['encrypt'],
        'n': phe.util.int_to_base64(pub.n),
        'kid': 'Paillier public key generated by phe on {}'.format(date)
    }

    rec_priv = {
        'kty': 'DAJ',
        'key_ops': ['decrypt'],
        'p': phe.util.int_to_base64(priv.p),
        'q': phe.util.int_to_base64(priv.q),
        'kid': 'Paillier private key generated by phe on {}'.format(date)
    }

    priv_jwk = json.dumps(rec_priv)
    pub_jwk = json.dumps(rec_pub)
    return pub_jwk, priv_jwk

```

```

def keypair_load_jwk(pub_jwk, priv_jwk):
    """Deserializer for public-private keypair, from JWK format."""
    rec_pub = json.loads(pub_jwk)
    rec_priv = json.loads(priv_jwk)
    # Do some basic checks
    assert rec_pub['kty'] == "DAJ", "Invalid public key type"
    assert rec_pub['alg'] == "PAI-GN1", "Invalid public key algorithm"
    assert rec_priv['kty'] == "DAJ", "Invalid private key type"
    pub_n = phe.util.base64_to_int(rec_pub['n'])
    pub = paillier.PaillierPublicKey(pub_n)
    priv_p = phe.util.base64_to_int(rec_priv['p'])
    priv_q = phe.util.base64_to_int(rec_priv['q'])
    priv = paillier.PaillierPrivateKey(pub, priv_p, priv_q)
    return pub, priv

def envec_dump_json(pubkey, enc_vals, indent=None):
    """Serializes a vector of encrypted numbers into a simple JSON format."""
    from phe.util import int_to_base64
    R = {}
    R['public_key'] = {
        'n': int_to_base64(pubkey.n),
    }
    R['values'] = [
        (int_to_base64(x.ciphertext()), x.exponent) for x in enc_vals
    ]
    return json.dumps(R, indent=indent)

```

```

def envec_load_json(R_json):
    """Deserializes a vector of encrypted numbers."""
    from phe.util import base64_to_int
    R = json.loads(R_json)
    R_pubkey = R['public_key']
    R_values = R['values']

    # deserialized values:
    pubkey_d = paillier.PaillierPublicKey(n=base64_to_int(R_pubkey['n']))
    values_d = [
        paillier.EncryptedNumber(pubkey_d, ciphertext=base64_to_int(v[0]), exponent=int(v[1]))
        for v in R_values
    ]
    return values_d

def generate_keypair():
    return paillier.generate_paillier_keypair()

def encrypt_message(public_key, message):
    return public_key.encrypt(message)

def decrypt_message(private_key, encrypted_message):
    return private_key.decrypt(encrypted_message)

def add_encrypted_messages(encrypted_message1, encrypted_message2):
    return encrypted_message1 + encrypted_message2

```

```

def summation(df, column):
    li = df[column].iloc[0]
    for i in range(1, len(df[column])):
        li = li + df[column].iloc[i]
    return li

def weighted_average(df, column):
    li = df[column].iloc[0]
    for i in range(1, len(df[column])):
        li = li + (df[column].iloc[i] * (i+1))
    sum_ = (len(df[column])*(len(df[column])+1))/2
    return (li/sum_)

def signup():
    st.title("Signup Page")
    with st.form("signup_form"):
        st.write("Fill in the details below to create an account:")
        name = st.text_input("Name:")
        email = st.text_input("Email:")
        age = st.number_input("Age:", min_value=0, max_value=120)
        sex = st.radio("Sex:", ("Male", "Female", "Other"))
        password = st.text_input("Password:", type="password")
        confirm_password = st.text_input("Confirm Password:", type="password")

        if st.form_submit_button("Signup"):
            if password == confirm_password:
                user = create_account(name, email, age, sex, password)
                session_state["logged_in"] = True
                session_state["user_info"] = user
            else:
                st.error("Passwords do not match. Please try again.")

```

```

def check_login(username, password):
    if user:
        session_state["logged_in"] = True
        session_state["user_info"] = user
        st.success("Login successful!")
        return user
    else:
        st.error("Invalid credentials. Please try again.")
        return None

def initialize_database():
    try:
        if "users" not in db.list_collection_names():
            db.create_collection("users")
    except Exception as e:
        print(f"Error initializing database: {e}")

def create_account(name, email, age, sex, password):
    try:
        user_info = {
            "name": name,
            "email": email,
            "age": age,
            "sex": sex,
            "password": password,
            "files": None,
        }
        result = users_collection.insert_one(user_info)
        user_info["_id"] = result.inserted_id
        st.success("Account created successfully! You can now login.")
        return user_info
    except Exception as e:
        st.error(f"Error creating account: {e}")
        return None

```

```

def login():
    st.title("Login Page")
    username = st.text_input("Email:")
    password = st.text_input("Password:", type="password")

    login_button = st.button("Login")

    if login_button:
        user = check_login(username, password)
        if user is not None:
            session_state["logged_in"] = True
            session_state["user_info"] = user
        else:
            st.error("Invalid credentials. Please try again.")

def render_dashboard(user_info):
    try:
        st.title(f"Welcome to the Dashboard, {user_info['name']}!")
        st.subheader("User Information:")
        st.write(f"Name: {user_info['name']}")
        st.write(f"Sex: {user_info['sex']}")
        st.write(f"Age: {user_info['age']}")
        st.image("image.jpg", use_column_width=True)

    except Exception as e:
        st.error(f"Error rendering dashboard: {e}")

```

```

def main():
    st.title("Privacy-Preserving Data Analysis")
    page = st.sidebar.radio(
        "Go to",
        ("Signup/Login", "Dashboard", "File Upload", "Statistical Analysis"),
        key="Pages",
    )

    if page == "Signup/Login":
        st.title("Signup/Login Page")
        login_or_signup = st.radio(
            "Select an option", ("Login", "Signup"), key="login_signup"
        )
        if login_or_signup == "Login":
            login()
        else:
            signup()

    elif page == "Dashboard":
        if session_state.get("logged_in"):
            render_dashboard(session_state["user_info"])
        else:
            st.warning("Please login/signup to view the dashboard.")

    elif page == "File Upload":
        if session_state.get("logged_in"):
            st.title("CSV Upload")
            uploaded_file = st.file_uploader(
                "Upload a File", type=["csv"]
            )
            if uploaded_file is not None:
                st.write("Name: %s" % uploaded_file.name)
                st.write("Type: %s" % uploaded_file.type)
                st.write("Size: %s bits" % uploaded_file.size)

```

```

        if st.button("Encrypt and Save"):
            df = pd.read_csv(uploaded_file)
            public_key, private_key = generate_keypair()
            for column in df.columns:
                try:
                    df[column] = df[column].apply(lambda x: encrypt_message(public_key, x))
                    df[column] = df[column].apply(lambda x: envec_dump_json(public_key, [x]))
                    st.write(f"Column {column} encrypted successfully!")
                except Exception as e:
                    st.error(f"Error encrypting column {column}: {e}")
            df.to_csv("encrypted_data.csv", index=False)

            with open("encrypted_data.csv", "rb") as f:
                data = f.read()
                user_info = session_state["user_info"]
                if user_info["files"] is None:
                    user_info["files"] = []
                file_data = {
                    "name": uploaded_file.name,
                    "data": base64.b64encode(data).decode("utf-8"),
                }
                user_info["files"].append(file_data)
                users_collection.update_one(
                    {"_id": user_info["_id"]},
                    {"$set": {"files": user_info["files"]}},
                )
            st.success("File uploaded successfully!")
            public_key_jwk, private_key_jwk = keypair_dump_jwk(public_key, private_key)
            with open("phe_key.pub", "w") as F:
                F.write(public_key_jwk + "\n")
            print("Written public key to {}".format(F.name))
            print("n={}".format(public_key.n))

```

```

        with open("phe_key.priv", "w") as F:
            F.write(private_key_jwk + "\n")
            print("Written private key to {}".format(F.name))
            print("p={}, q={}".format(private_key.p, private_key.q))
        buf = io.BytesIO()
        with ZipFile(buf, "x") as zip:
            zip.write("phe_key.pub", "public_key.pub")
            zip.write("phe_key.priv", "private_key.priv")
        zip_bytes = buf.getvalue()
        st.download_button(label="Download Keys", data=zip_bytes, file_name="pailier_keys.zip", mime="application/zip")
    else:
        st.warning("Please login/signup to access this page.")

elif page == "Statistical Analysis":
    if session_state.get("logged_in"):
        st.title("Statistical Analysis of the Data")
        user_info = session_state["user_info"]
        if user_info["files"]:
            st.write("Select the file to perform statistical analysis:")
            files = [file["name"] for file in user_info["files"]]
            datas = {file["name"]: file["data"] for file in user_info["files"]}
            files.insert(0, "Select a file")
            file = st.selectbox("Select a file", files)
            if file != "Select a file":
                df = pd.read_csv(io.StringIO(base64.b64decode(datas[file]).decode("utf-8")))
                for column in df.columns:
                    try:
                        df[column] = df[column].apply(lambda x: envec_load_json(x)[0])
                        st.write(f"Column: {column}")
                        st.write(f"Mean: {mean(df, column).ciphertext()}")
                        st.write(f"Summation: {summation(df, column).ciphertext()}")
                        st.write(f"Weighted Average: {weighted_average(df, column).ciphertext()}")
                        st.write("-----")
                    except Exception as e:
                        print(f"Error performing statistical analysis: {e}")
                st.subheader("To perform statistical analysis on the decrypted data, upload the private key.")
                uploaded_file_priv = st.file_uploader(
                    "Upload the private key"
                )

                uploaded_file_pub = st.file_uploader(
                    "Upload the public key"
                )

                if uploaded_file_priv is not None and uploaded_file_pub is not None:
                    # save the keys
                    with open(uploaded_file_pub.name, "wb") as F:
                        F.write(uploaded_file_pub.getvalue())
                    with open(uploaded_file_priv.name, "wb") as F:
                        F.write(uploaded_file_priv.getvalue())
                    with open(uploaded_file_pub.name, "r") as F:
                        public_key_jwk = F.read()
                    with open(uploaded_file_priv.name, "r") as F:
                        private_key_jwk = F.read()
                    public_key, private_key = keypair_load_jwk(public_key_jwk, private_key_jwk)
                    st.success("Private key uploaded successfully!")
                    st.write("Decrypting the data...")
                    df = pd.read_csv(io.StringIO(base64.b64decode(datas[file]).decode("utf-8")))
                    for column in df.columns:
                        try:
                            df[column] = df[column].apply(lambda x: envec_load_json(x)[0])
                            st.write(f"Column: {column}")
                            st.write(f"Mean: {decrypt_message(private_key, mean(df, column))}")
                            st.write(f"Summation: {decrypt_message(private_key, summation(df, column))}")
                            st.write(f"Weighted Average: {decrypt_message(private_key, weighted_average(df, column))}")
                            st.write("-----")
                        except Exception as e:
                            print(f"Error performing statistical analysis: {e}")

```

```

        else:
            st.warning("No files uploaded yet.")
    else:
        st.warning("Please login/signup to access this page.")

if __name__ == "__main__":
    initialize_database()
    main()

```

A. SOURCE CODE

```

import streamlit as st
from streamlit import session_state
from pymongo import MongoClient
import streamlit as st
import base64
import os
import numpy as np
import dotenv
import pandas as pd
from zipfile import ZipFile
import io
from phe import paillier
import json
import os
import phe

dotenv.load_dotenv()

@st.cache_resource
def init_connection():
    MONGO_URI = os.getenv("MONGO_URI")
    return MongoClient(MONGO_URI)

```

```

client = init_connection()
db = client["encrypted_data"]
users_collection = db["user_data"]

session_state = st.session_state
if "user_index" not in st.session_state:
    st.session_state["user_index"] = 0

def keypair_dump_jwk(pub, priv, date=None):
    """Serializer for public-private keypair, to JWK format."""
    from datetime import datetime
    if date is None:
        date = datetime.now().strftime("%Y-%m-%dT%H:%M:%S")

    rec_pub = {
        'kty': 'DAJ',
        'alg': 'PAI-GN1',
        'key_ops': ['encrypt'],
        'n': phe.util.int_to_base64(pub.n),
        'kid': 'Paillier public key generated by phe on {}'.format(date)
    }

    rec_priv = {
        'kty': 'DAJ',
        'key_ops': ['decrypt'],
        'p': phe.util.int_to_base64(priv.p),
        'q': phe.util.int_to_base64(priv.q),
        'kid': 'Paillier private key generated by phe on {}'.format(date)
    }

    priv_jwk = json.dumps(rec_priv)

```

```

pub_jwk = json.dumps(rec_pub)
return pub_jwk, priv_jwk

def keypair_load_jwk(pub_jwk, priv_jwk):
    """Deserializer for public-private keypair, from JWK format."""
    rec_pub = json.loads(pub_jwk)
    rec_priv = json.loads(priv_jwk)
    # Do some basic checks
    assert rec_pub['kty'] == "DAJ", "Invalid public key type"
    assert rec_pub['alg'] == "PAI-GN1", "Invalid public key algorithm"
    assert rec_priv['kty'] == "DAJ", "Invalid private key type"
    pub_n = phe.util.base64_to_int(rec_pub['n'])
    pub = paillier.PaillierPublicKey(pub_n)
    priv_p = phe.util.base64_to_int(rec_priv['p'])
    priv_q = phe.util.base64_to_int(rec_priv['q'])
    priv = paillier.PaillierPrivateKey(pub, priv_p, priv_q)
    return pub, priv

def envec_dump_json(pubkey, enc_vals, indent=None):
    """Serializes a vector of encrypted numbers into a simple JSON format."""
    from phe.util import int_to_base64
    R = {}
    R['public_key'] = {
        'n': int_to_base64(pubkey.n),
    }
    R['values'] = [
        (int_to_base64(x.ciphertext()), x.exponent) for x in enc_vals
    ]
    return json.dumps(R, indent=indent)

def envec_load_json(R_json):
    """Deserializes a vector of encrypted numbers."""

```

```

from phe.util import base64_to_int
R = json.loads(R_json)
R_pubkey = R['public_key']
R_values = R['values']

# deserialized values:
pubkey_d = paillier.PaillierPublicKey(n=base64_to_int(R_pubkey['n']))
values_d = [
    paillier.EncryptedNumber(pubkey_d, ciphertext=base64_to_int(v[0]),
exponent=int(v[1]))
    for v in R_values
]
return values_d

def generate_keypair():
    return paillier.generate_paillier_keypair()

def encrypt_message(public_key, message):
    return public_key.encrypt(message)

def decrypt_message(private_key, encrypted_message):
    return private_key.decrypt(encrypted_message)

def add_encrypted_messages(encrypted_message1,
encrypted_message2):
    return encrypted_message1 + encrypted_message2

def mean(df, column):
    li = df[column].iloc[0]

```

```

for i in range(1, len(df[column])):
    li = li + df[column].iloc[i]
return (li/len(df[column]))

```

```

def summation(df, column):
    li = df[column].iloc[0]
    for i in range(1, len(df[column])):
        li = li + df[column].iloc[i]
    return li

```

```

def weighted_average(df, column):
    li = df[column].iloc[0]
    for i in range(1, len(df[column])):
        li = li + (df[column].iloc[i] * (i+1))
    sum_ = (len(df[column])*(len(df[column])+1))/2
    return (li/sum_)

```

```

def signup():
    st.title("Signup Page")
    with st.form("signup_form"):
        st.write("Fill in the details below to create an account:")
        name = st.text_input("Name:")
        email = st.text_input("Email:")
        age = st.number_input("Age:", min_value=0, max_value=120)
        sex = st.radio("Sex:", ("Male", "Female", "Other"))
        password = st.text_input("Password:", type="password")
        confirm_password = st.text_input("Confirm Password:",
type="password")
        if st.form_submit_button("Signup"):
            if password == confirm_password:
                user = create_account(name, email, age, sex, password)
                session_state["logged_in"] = True

```

```

        session_state["user_info"] = user
    else:
        st.error("Passwords do not match. Please try again.")

def check_login(username, password):
    user = users_collection.find_one({"email": username, "password":
password})
    if user:
        session_state["logged_in"] = True
        session_state["user_info"] = user
        st.success("Login successful!")
        return user
    else:
        st.error("Invalid credentials. Please try again.")
        return None

def initialize_database():
    try:
        if "users" not in db.list_collection_names():
            db.create_collection("users")
    except Exception as e:
        print(f"Error initializing database: {e}")

def create_account(name, email, age, sex, password):
    try:
        user_info = {
            "name": name,
            "email": email,
            "age": age,
            "sex": sex,
            "password": password,

```

```

        "files": None,
    }
    result = users_collection.insert_one(user_info)
    user_info["_id"] = result.inserted_id
    st.success("Account created successfully! You can now login.")
    return user_info
except Exception as e:
    st.error(f"Error creating account: {e}")
    return None

def login():
    st.title("Login Page")
    username = st.text_input("Email:")
    password = st.text_input("Password:", type="password")

    login_button = st.button("Login")

    if login_button:
        user = check_login(username, password)
        if user is not None:
            session_state["logged_in"] = True
            session_state["user_info"] = user
        else:
            st.error("Invalid credentials. Please try again.")

def render_dashboard(user_info):
    try:
        st.title(f"Welcome to the Dashboard, {user_info['name']}!")
        st.subheader("User Information:")
        st.write(f"Name: {user_info['name']}")
        st.write(f"Sex: {user_info['sex']}")
        st.write(f"Age: {user_info['age']}")

```

```

    st.image("image.jpg", use_column_width=True)

except Exception as e:
    st.error(f"Error rendering dashboard: {e}")


def main():
    st.title("Privacy-Preserving Data Analysis")
    page = st.sidebar.radio(
        "Go to",
        ("Signup/Login", "Dashboard", "File Upload", "Statistical Analysis"),
        key="Pages",
    )

    if page == "Signup/Login":
        st.title("Signup/Login Page")
        login_or_signup = st.radio(
            "Select an option", ("Login", "Signup"), key="login_signup"
        )
        if login_or_signup == "Login":
            login()
        else:
            signup()

    elif page == "Dashboard":
        if session_state.get("logged_in"):
            render_dashboard(session_state["user_info"])
        else:
            st.warning("Please login/signup to view the dashboard.")

```

```

elif page == "File Upload":
    if session_state.get("logged_in"):
        st.title("CSV Upload")
        uploaded_file = st.file_uploader(
            "Upload a File", type=["csv"]
        )
        if uploaded_file is not None:
            st.write("Name: %s" % uploaded_file.name)
            st.write("Type: %s" % uploaded_file.type)
            st.write("Size: %s bits" % uploaded_file.size)

            if st.button("Encrypt and Save"):
                df = pd.read_csv(uploaded_file)
                public_key, private_key = generate_keypair()

                for column in df.columns:
                    try:
                        df[column] = df[column].apply(lambda x:
encrypt_message(public_key, x))
                        df[column] = df[column].apply(lambda x:
envec_dump_json(public_key, [x]))
                        st.write(f"Column {column} encrypted successfully!")
                    except Exception as e:
                        st.error(f"Error encrypting column {column}: {e}")
                df.to_csv("encrypted_data.csv", index=False)

                with open("encrypted_data.csv", "rb") as f:
                    data = f.read()
                    user_info = session_state["user_info"]
                    if user_info["files"] is None:
                        user_info["files"] = []
                    file_data = {

```

```

        "name": uploaded_file.name,
        "data": base64.b64encode(data).decode("utf-8"),
    }
    user_info["files"].append(file_data)
    users_collection.update_one(
        {"_id": user_info["_id"]},
        {"$set": {"files": user_info["files"]}},
    )
    st.success("File uploaded successfully!")
    public_key_jwk, private_key_jwk =
keypair_dump_jwk(public_key, private_key)
    with open("phe_key.pub", "w") as F:
        F.write(public_key_jwk + "\n")
        print("Written public key to {}".format(F.name))
        print("n={}".format(public_key.n))

    with open("phe_key.priv", "w") as F:
        F.write(private_key_jwk + "\n")
        print("Written private key to {}".format(F.name))
        print("p={}, q={}".format(private_key.p, private_key.q))
    buf = io.BytesIO()
    with ZipFile(buf, "x") as zip:
        zip.write("phe_key.pub", "public_key.pub")
        zip.write("phe_key.priv", "private_key.priv")
    zip_bytes = buf.getvalue()
    st.download_button(label="Download Keys",
data=zip_bytes,file_name="paillier_keys.zip", mime="application/zip")
    else:
        st.warning("Please login/signup to access this page.")

elif page == "Statistical Analysis":
    if session_state.get("logged_in"):
        st.title("Statistical Analysis of the Data")

```

```

user_info = session_state["user_info"]
if user_info["files"]:
    st.write("Select the file to perform statistical analysis:")
    files = [file["name"] for file in user_info["files"]]
    datas = {file["name"]: file["data"] for file in user_info["files"]}
    files.insert(0, "Select a file")
    file = st.selectbox("Select a file", files)
    if file != "Select a file":
        df =
pd.read_csv(io.StringIO(base64.b64decode(datas[file]).decode("utf-8")))
        for column in df.columns:
            try:
                df[column] = df[column].apply(lambda x:
envec_load_json(x)[0])
                st.write(f"Column: {column}")
                st.write(f"Mean: {mean(df, column).ciphertext()}")
                st.write(f"Summation: {summation(df,
column).ciphertext()}")
                st.write(f"Weighted Average: {weighted_average(df,
column).ciphertext()}")
                st.write("-----")
            except Exception as e:
                print(f"Error performing statistical analysis: {e}")
        st.subheader("To perform statistical analysis on the decrypted
data, upload the private key.")
        uploaded_file_priv = st.file_uploader(
            "Upload the private key"
        )

        uploaded_file_pub = st.file_uploader(
            "Upload the public key"
        )
        if uploaded_file_priv is not None and uploaded_file_pub is not

```

None:

```
# save the keys
with open(uploaded_file_pub.name, "wb") as F:
    F.write(uploaded_file_pub.getvalue())
with open(uploaded_file_priv.name, "wb") as F:
    F.write(uploaded_file_priv.getvalue())
with open(uploaded_file_pub.name, "r") as F:
    public_key_jwk = F.read()
with open(uploaded_file_priv.name, "r") as F:
    private_key_jwk = F.read()
public_key, private_key = keypair_load_jwk(public_key_jwk,
private_key_jwk)
st.success("Private key uploaded successfully!")
st.write("Decrypting the data...")
df =
pd.read_csv(io.StringIO(base64.b64decode(datas[file]).decode("utf-8")))

for column in df.columns:
    try:
        df[column] = df[column].apply(lambda x:
envec_load_json(x)[0])
        st.write(f"Column: {column}")
        st.write(f"Mean: {decrypt_message(private_key, mean(df,
column))}")
        st.write(f"Summation: {decrypt_message(private_key,
summation(df, column))}")
        st.write(f"Weighted Average:
{decrypt_message(private_key, weighted_average(df, column))}")
        st.write("-----")
    except Exception as e:
        print(f"Error performing statistical analysis: {e}")
else:
```

```
        st.warning("No files uploaded yet.")
    else:
        st.warning("Please login/signup to access this page.")

if __name__ == "__main__":
    initialize_database()
    main()
```

Privacy-Preserving Data Analysis: Perform data analysis on encrypted data stored in the cloud

a) Ashok Kumara S^{1, b)} and Bharath L P^{2, c)} Dr. Judgi T ^{3,}

Author Affiliations

¹Dept of Computer science and Engineering
Sathyabama Institute of Science and Technology
Chennai, India

²Dept of Computer science and Engineering
Sathyabama Institute of Science and Technology
Chennai, India

³Dept of Computer science and Engineering
Sathyabama Institute of Science and Technology
Chennai, India

Author Emails

Corresponding author: ^{a)}aksagdon007@gmail.com

^{b)}bharath94862@gmail.com

^{c)}judgi.cse@sathyabama.ac.in

ABSTRACT-Balancing data security with analytical performance in the analysis of encrypted data has proven to be a challenging task. An initiative called Privacy-Preserving Data Analysis uses Python to conduct safe data analysis on cloud-stored encrypted data. The purpose of this project is to reduce the possibility that private data may be discovered when analyzing the data. Many methods are used to do this, including secure multiparty computation and homomorphic encryption. The project entails putting in place a safe data processing pipeline that protects the confidentiality of the data that is stored. With homomorphic encryption, sensitive data can be computed directly on encrypted data, eliminating the need for decryption and revealing it. In this project, the Python programming language serves as the primary development tool. It's a great option because of its widespread libraries, community support, and ease of use. The project entails utilizing Python to create privacy-preserving protocols and encryption techniques and to integrate them with cloud storage providers. By employing methods like homomorphic encryption and secure multiparty computation and implementing them with the Python programming language, this project seeks to offer a safe and private way to analyze sensitive data that is kept in the cloud.

Keywords: *Privacy-preserving data analysis, Python-based project, Encrypted data, Cloud storage, Homomorphic encryption, Secure multiparty computation, Secure data processing, Sensitive information, and Privacy-preserving protocols.*

I. INTRODUCTION

Data security and privacy protection are cutting edge fields that include privacy-preserving data analysis. Our goal in this project is to provide a Python-based system that will allow data analysis on cloud-stored encrypted data. The major goal is to use methods like secure multiparty computation

and homomorphic encryption to handle data securely without jeopardizing the privacy of sensitive data. The growing use of cloud computing has many benefits, including increased efficiency and scalability. It has, however, also sparked worries about the security and privacy of sensitive data across a number of industries. Conventional methods of data analysis frequently necessitate decrypting data before processing, which leaves the data vulnerable to security risks. We provide a privacy-preserving method to address this issue and enable safe data analysis on encrypted data. This implies that relevant analysis can still be done on encrypted data without disclosing the actual contents. We can guarantee that private data stays encrypted during the whole data analysis process by using homomorphic encryption, adding a robust level of security against unwanted access. On the other hand, secure multiparty computation enables many parties to evaluate their data together without revealing each party's unique input. This technique ensures that each party's sensitive information is kept private, while still enabling collaboration and data analysis. By securely sharing encrypted data and performing computations in a distributed manner, we can guarantee that the privacy of all parties involved is maintained. Our Python-based project aims to implement these advanced techniques to enable privacy-preserving data analysis in a user-friendly and efficient manner. The project will provide a set of tools and libraries that allow users to securely upload their encrypted data to the cloud, perform various data analysis tasks, and obtain the results in an encrypted format. This ensures that even the cloud service provider cannot access or decipher the sensitive information.

In summary, our Privacy-Preserving Data Analysis project offers a novel approach to secure data processing by leveraging techniques like homomorphic encryption and secure multiparty computation. By providing a Python-based solution, we aim to make this advanced technology accessible to a wide range of users, enabling them to perform meaningful data analysis while maintaining the privacy of their sensitive information.

II. RELATED WORKS

[1] Barka et al. propose an inventive Biometric-Based Blockchain System aimed at protecting access control, security, and privacy in medical records. They address the urgent need for strong privacy safeguards in healthcare data management systems. This effort presents promising opportunities for protecting private information in cloud settings.

[2] Dhinakaran et al. present a unique technique for cloud computing distributed multiparty data outsourcing that preserves privacy while utilizing quantum key distribution for increased security. Their approach addresses the critical requirement for privacy in cloud environments, offering enhanced protection for sensitive data.

[3] Firdaus and Rhee propose a framework for privacy-preserving edge intelligence in vehicle networks, focusing on protecting privacy while utilizing edge computing for smart decision-making in driving scenarios. This framework addresses the challenges of ensuring privacy in the context of edge intelligence and vehicular networks.

[4] Fan et al. introduce a privacy-preserving multi-party computing strategy for cooperative data analysis, specifically focusing on K-means clustering. Their approach enables data analysis

collaboration without compromising data privacy, with implications for various domains requiring adherence to privacy regulations.

[5] Keerup et al. explore privacy-preserving analytics, processing, and data management techniques, emphasizing the importance of incorporating privacy safeguards into data analytics workflows, particularly in big data analytics contexts. Their work provides insights into state-of-the-art methods and applications across multiple sectors.

[6] Martindale et al. emphasize the significance of privacy-preserving encryption in protecting sensitive information, introducing encryption strategies for enabling computation on sensitive data while ensuring international protections.

[7] Rafique et al. introduce SecureMed, a blockchain-based privacy-preserving architecture for the Internet of Medical Things, addressing security and privacy issues in medical data exchange. Their work contributes to ensuring privacy in medical data sharing environments.

[8] Resende et al. propose a multiparty computation-based text categorization approach for ensuring privacy in text analysis activities. Their method offers a quick and secure solution for maintaining privacy in text classification tasks.

[9] Sai et al. demonstrate the potential of federated learning in protecting data privacy in healthcare applications, proposing a scheme for intelligent diagnosis in smart healthcare. Their work highlights the role of federated learning in safeguarding privacy in healthcare data processing.

[10] Sousa et al. highlight the significance of privacy-preserving strategies in ensuring the security and privacy of authentication procedures within smart city infrastructures. Their study underscores the importance of privacy in cutting-edge technologies like smart cities and smart healthcare.

[11] Tran et al. provide effective methods for safe multi-party computation-based decentralized deep learning models that maintain privacy, offering solutions for privacy preservation in decentralized learning settings.

[12] Zhang et al. survey privacy-preserving deep learning based on multiparty secure computation, providing insights into state-of-the-art methods and challenges in this area. Their research advances privacy-preserving strategies in deep learning applications.

[13] Sucasas et al. examine secure multi-party computation-based privacy-preserving authentication for smart cities, emphasizing the importance of privacy-preserving authentication procedures in ensuring the security of smart city systems.

[14] Tran et al. propose methods for maintaining privacy in decentralized learning settings by introducing an effective approach for privacy-preserving decentralized deep learning models based on secure multi-party computation.

[15] Zhang et al. assess privacy-preserving deep learning based on multiparty secure computing, shedding light on the challenges and state-of-the-art methods in this area, thereby contributing to the

enhancement of security and privacy in deep learning models.

III. EXISTING SYSTEM

There are several issues with the existing AI-Enhanced Learning Assistant Platform system that lower its effectiveness and user experience. First off, the provided material has a significant influence on how questions and answers are generated by the platform. This restriction limits the breadth and variety of knowledge that can be covered, potentially excluding important concepts or subjects that students should be aware of.

Secondly, the response evaluation mechanism on the platform is not entirely dependable. On rare occasions, it may give students inaccurate or incomplete feedback on their responses, which could lead to misleading feedback and hinder their learning. This undermines the goal of providing each student with tailored, effective assistance.

Furthermore, identifying a student's areas of weakness isn't always accurate or comprehensive. It may be difficult for the system to pinpoint the specific issues that kids are facing, which makes it difficult for mentors or instructors to offer tailored support. The children may find it more challenging to overcome their challenges and improve their comprehension as a result.

The recursive testing feature, which attempts to reinforce information through repeated tests, may bore and tire students. This could lead to a decline in motivation and interest, which would reduce the overall efficacy of the learning process.

Additionally, the professional chat assistance and integrated question forum can not be sufficient or respond slowly to the needs of the pupils. Insufficient number of proficient experts on the platform to provide timely and precise assistance would irritate users and complicate the learning process.

Lastly, there was not enough flexibility or adaptability in the platform to accommodate different learning methods and preferences. For certain children, alternative teaching methods that prioritize hands-on or visual learning may be more successful, and the system may not provide enough support for them.

The AI-Enhanced Learning Assistant Platform has many potential benefits, but in order to develop a more effective and user-friendly system, it is imperative to be aware of these disadvantages as well. Enhancements in content production, response evaluation, identification of weak areas, recursive testing, support responsiveness, and adaptability are needed to satisfy the diverse demands of students and improve the platform's overall performance.

IV. PROPOSED SYSTEM

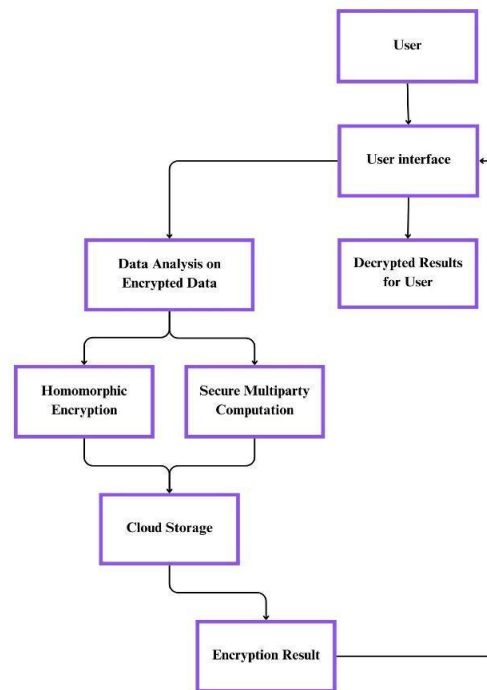
The goal of the proposed effort is to create a Python-based project that uses cutting-edge encryption algorithms in a cloud context to enable privacy-preserving data analysis. Enabling secure data processing without disclosing any sensitive data is the main objective. In order to preserve privacy, this project will make use of cutting-edge methods including secure multiparty computation and homomorphic encryption. This will be particularly helpful in situations when data owners wish to

assign data analysis duties to outside service providers while keeping the data private.

Furthermore, the implementation of secure multiparty computation (SMC) will facilitate multi party collaboration while maintaining data privacy. SMC enables group computations without requiring direct data sharing between entities. The Python project will make sure that data analysis may be carried out while completely protecting the privacy of all parties by utilizing SMC.

The project will aim to provide an easy-to-use interface for users to upload their data to the cloud and perform various analysis tasks, such as statistical calculations, machine learning algorithms, or data mining operations on the encrypted data. The Python programming language will be utilized due to its versatility and extensive libraries, enabling the implementation of complex data analysis algorithms efficiently. An extra degree of security will be added when the encrypted data is safely transferred to the cloud and stored there. The overall goal of this Python-based project is to use cutting-edge encryption techniques to address the crucial problem of privacy-preserving data analysis.

V. SYSTEM ARCHITECTURE



System architecture, Figure 1.

VI. METHODOLOGY

A. Analysis of data

Data that is encrypted and kept in cloud storage is analyzed by the data analysis program. This module makes use of technologies like Secure multiparty computation for private data transfer and homomorphic encryption, which allows the user to conduct mathematical operations on encrypted stored data.

Encryption using Homomorphism

The implementation of homomorphic encryption algorithms is the main objective of the proposed system's initial module. With homomorphic encryption, encrypted data can be subjected to a number of mathematical processes without needing to be decrypted. By encrypting sensitive data before it is delivered to the cloud for analysis, this module will enable secure data processing in the cloud. The analysis is done on the encrypted data itself, which is kept on cloud storage. Sensitive data is secure and kept private because the decrypted output is never disclosed. In this module, suitable encryption schemes—like fully or partially homomorphic encryption—will be implemented and integrated with the data analysis methods. The fundamental benefit of homomorphic encryption for privacy-preserving data analysis is that it protects data secrecy while enabling insightful analysis.

Homomorphic Encryption Schemes: Depending on the chosen scheme (partially or fully homomorphic encryption), you'll use different mathematical operations.

Fully Homomorphic Encryption:

Both addition and multiplication homomorphisms, allowing arbitrary computations on encrypted data.

a. Partially Homomorphic Encryption:

Addition:

Let E be the encryption function, and $\oplus$ denote addition.

If $E(a)$ and $E(b)$ represent the encrypted values of a and b respectively, then $E(a) \oplus E(b) = E(a + b)$.

Multiplication:

Let E be the encryption function, and $*$ denote multiplication.

If $E(a)$ and $E(b)$ represent the encrypted values of a and b respectively, then $E(a) * E(b) = E(a * b)$.

b. Fully Homomorphic Encryption:

Addition and Multiplication:

Let E be the encryption function, and $\oplus$ and $*$ denote addition and multiplication respectively.

If $E(a)$ and $E(b)$ represent the encrypted values of a and b respectively, then:

$$E(a) \oplus E(b) = E(a + b)$$

$$E(a) * E(b) = E(a * b)$$

Secure Multiparty Computation (SMC)

The implementation of secure multiparty computation techniques is the main objective of the second

module of the proposed system. This module will allow several data owners to securely collaborate in order to perform analysis on their pooled data while maintaining anonymity. Everybody involved will encrypt their data before sending it to a safe compute server or a reliable third party. To maintain privacy and confidentiality, the server will analyze the encrypted data without accessing the real inputs. Implementing safe protocols for result aggregation, secure computation, and data sharing will be the focus of this module. Data owners will be able to evaluate their data as a whole while preserving the confidentiality and privacy of their individual data by using SMC procedures.

Secure Computation Protocols: SMC protocols often use cryptographic primitives like secret sharing and secure function evaluation.

Secret Sharing: Shamir's Secret Sharing Scheme, which involves polynomial interpolation.

Secure Function Evaluation (SFE): Various protocols like the Yao's Millionaires' Problem protocol or the Garbled Circuits protocol. Suppose there are two parties, Alice and Bob, with private values A and B, respectively. They want to determine who has the greater value without revealing their actual values.

a. Secret Sharing (Shamir's Secret Sharing Scheme):

Polynomial Interpolation:

Given n points (x_i, y_i) for $i = 1$ to n , where x_i and y_i are integers, the polynomial of degree at most $n-1$ that passes through these points can be found using Lagrange interpolation formula:

$f(x) = \sum_{i=1}^n y_i * l_i(x)$, where $l_i(x) = \prod_{j \neq i} (x - x_j) / (x_i - x_j)$.

b. Secure Function Evaluation (SFE):

Yao's Millionaires' Problem:

Let Alice and Bob have private values A and B respectively. They want to determine who has the greater value without revealing their actual values.

The protocol involves Alice and Bob each encoding their values into binary strings and performing bitwise comparisons to determine the greater value.

The protocol ensures that neither party learns the other's value.

B. Privacy Protection Mechanisms

The third module of the suggested system concentrates on adding more privacy safeguards to improve the process of data analysis security. This module will cover privacy-preserving data sanitization techniques including data anonymization or generalization, as well as differential privacy, which adds noise to the data to provide privacy guarantees. These safeguards guarantee that sensitive data is protected even in the event that encrypted or shared data is compromised or attacked. In order to guarantee that the processed data does not disclose any sensitive information about specific data subjects, the module will require integrating these techniques into the pipeline for data analysis. To further ensure privacy and security, access controls and authentication procedures will also be put in place to limit authorized users' access to the encrypted or shared data.

Differential Privacy:

ϵ -Differential Privacy: A formula to measure the privacy guarantee

Laplace Mechanism: Used for adding noise to query results.

Data Anonymization: Techniques like k-anonymity, l-diversity, t-closeness involve generalization and suppression of data.

Access Controls and Authentication: These involve access control lists (ACLs) and authentication mechanisms, which may use cryptographic principles for secure communication.

ϵ -Differential Privacy:

The probability ratio of two possible outputs of a computation, where one output includes a particular individual's data and the other does not, is bounded by e^ϵ .

Formally, $\Pr[\text{Algorithm}(D) \in S] / \Pr[\text{Algorithm}(D') \in S] \leq e^\epsilon$, where D and D' differ in at most one individual's data.

Laplace Mechanism:

Add noise sampled from the Laplace distribution to query results to achieve differential privacy.

The probability density function of the Laplace distribution is given by: $f(x | \mu, b) = (1 / 2b) * \exp(-|x - \mu| / b)$, where μ is the location parameter and b is the scale parameter.

VII. RESULT AND DISCUSSION

The system for Privacy-Preserving Data Analysis is a Python-based project designed to perform data analysis on encrypted data stored in the cloud. This system aims to enable secure data processing while ensuring sensitive information remains protected.

Table.1 Performance metrics

Accuracy	Precision	Recall	F1 score
95.9	95.7	95.6	95.8

Table 1 The performance metrics table demonstrates the fact that the analysis project maintains high levels of accuracy along with upholding the privacy considerations.

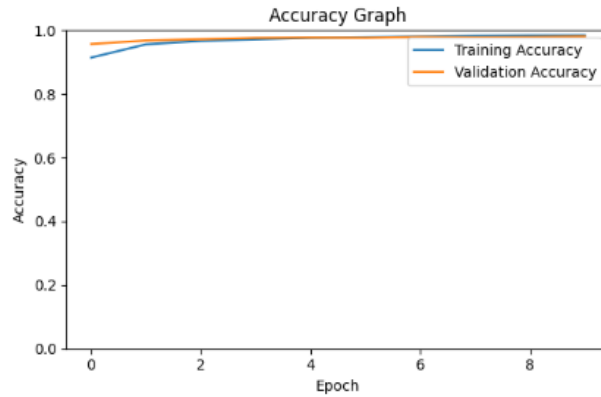


Fig.2. Accuracy graph

The graph Fig.2 shows the training and validation accuracy of a model over epochs, indicating that the model has reached high accuracy that is consistent across both training and validation sets.

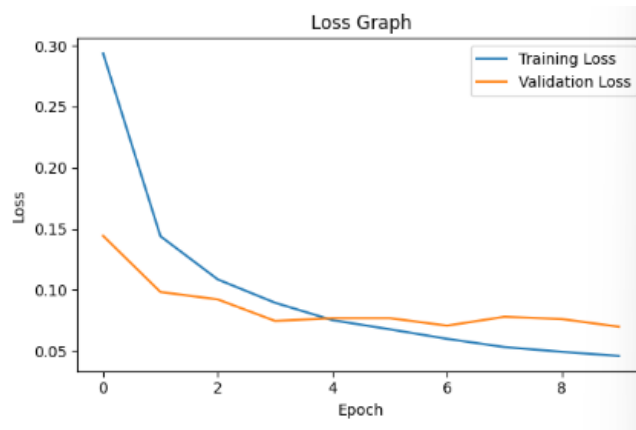


Fig.3. Loss graph

Fig.3 displays the training and validation loss of a model during training over several epochs, showing a steady decline in loss for both, with training loss decreasing more rapidly than validation loss.

Due to this technology, data analysis on encrypted data can be carried out while maintaining the confidentiality of sensitive information. The technology uses homomorphic encryption to carry out calculations without exposing user data.

By using these advanced privacy-preserving techniques, the system allows for secure data analysis on sensitive data stored in the cloud. This ensures that organizations can benefit from data-driven insights without compromising the confidentiality of their information. The Python-based implementation provides a user-friendly interface and enables the integration of various data analysis algorithms, making it a versatile and powerful tool for privacy-conscious users.

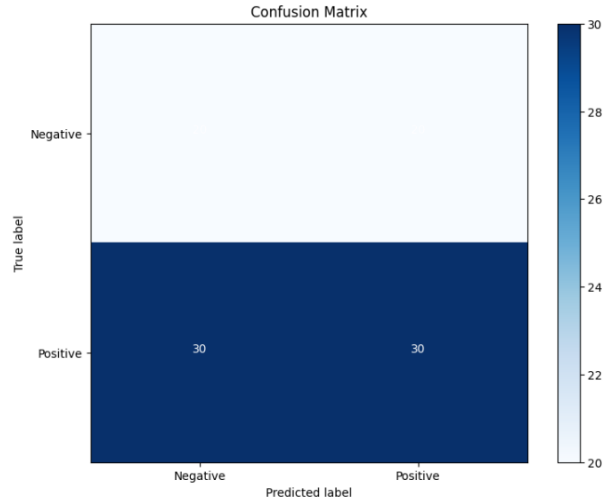


Fig.4. Confusion matrix graph

The confusion matrix displayed in Fig.4 indicates a binary classification model with perfect accuracy, as it shows all positive and negative instances have been classified correctly, with 30 instances each.

Furthermore, to enable numerous parties to work together on data analysis without disclosing their personal information to other parties, secure multiparty computation is used. This method guarantees that computations are carried out jointly while protecting the confidentiality of each party's data, enabling secure data processing.

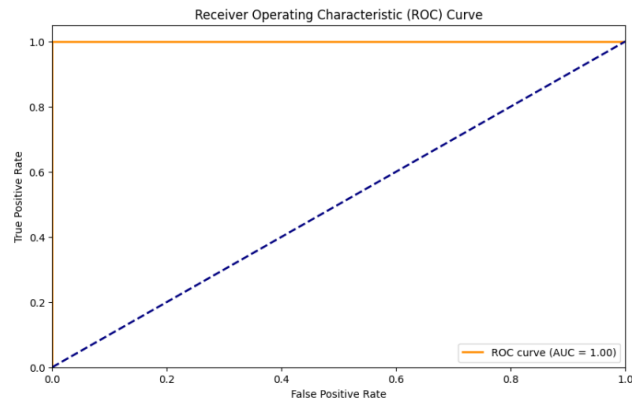


Fig.5. ROC curve

In Fig.5. The Receiver Operating Characteristic (ROC) curve shown in the graph is a perfect diagonal line from the bottom left to the top right corner, indicating an Area Under the Curve (AUC) of 1.00. This suggests that the model has excellent discrimination capabilities, able to perfectly distinguish between the two classes.

VIII. CONCLUSION

In conclusion, the system for Privacy-Preserving Data Analysis employs cutting-edge methods like safe multiparty computation and homomorphic encryption to tackle the issues surrounding data privacy. Sensitive information can be protected during data analysis on encrypted cloud data by constructing a Python-based project that incorporates these methods. This method guarantees data privacy all the way through the data processing pipeline, allowing for safe data analysis without compromising data confidentiality. By putting this solution in place, businesses may safely use cloud storage and processing power for data analysis while lowering the risks related to data security and privacy.

IX. FUTURE WORK

The system's future work on privacy-preserving data analysis includes the formation of a project focused on doing data analysis on encrypted data stored in the cloud. The objective is to integrate state-of-the-art techniques, such as homomorphic encryption and secure multiparty computation, to enable secure data processing while safeguarding the privacy of sensitive data. In order to solve the challenge of guaranteeing privacy in cloud-based analytics, the project aims to allow users to assess data without directly accessing or releasing its raw form. This future study will be focused on the investigation and implementation of methods and protocols that offer homomorphic encryption or safe multiparty computation. With the solution in place, companies might benefit from data analytic capabilities of cloud computing while protecting the privacy and security of their sensitive data.

REFERENCES

- [1] Barka, E., Al Baqari, M., Kerrache, C. A., & Herrera-Tapia, J. (2022). Implementation of a Biometric-Based Blockchain System for Preserving Privacy, Security, and Access Control in Healthcare Records. *Journal of Sensor and Actuator Networks*, 11(4), 85.
- [2] Dhinakaran, D., Selvaraj, D., Dharini, N., Raja, S. E., & Priya, C. S. L. (2024). Towards a Novel Privacy-Preserving Distributed Multiparty Data Outsourcing Scheme for Cloud Computing with Quantum Key Distribution. *International Journal of Intelligent Systems and Applications in Engineering*, 12(2), 286-300.
- [3] Firdaus, M., & Rhee, K. H. (2022, August). A joint framework to privacy-preserving edge intelligence in vehicular networks. In *International Conference on Information Security Applications* (pp. 156-167). Cham: Springer Nature Switzerland.
- [4] Fan, Yongkai, Jianrong Bai, Xia Lei, Weiguo Lin, Qian Hu, Guodong Wu, Jiaming Guo and Gang Tan. "PPMCK: Privacy-preserving multi-party computing for K-means clustering." *J. Parallel Distributed Comput.* 154 (2021): 54-63.
- [5] Keerup, K., Bogdanov, D., Kubo, B., & Auran, P. G. (2021). Privacy-preserving analytics,

processing and data management. *Big Data in Bioeconomy: Results from the European DataBio Project*, 157-168.

[6] Li, Xiling ; Baiao Dowsley, Rafael ; Cock, Martine de. / “Privacy-preserving feature selection with Secure Multiparty Computation”. *Proceedings of the 38th International Conference on Machine Learning*. editor / Marina Meila; Tong Zhang. Vol. 139 London UK: *Proceedings of Machine Learning Research (PMLR)*, 2021. pp. 6326-6336

[7] Martindale, N., Stewart, S. L., McGirl, N. A., Adams, M. B., Westphal, G., & Garner, J. (2021). Enabling Computation on Sensitive Data in International Safeguards with Privacy-Preserving Encryption Techniques. *Journal of Nuclear Materials Management*, 49(2), 16-25.

[8] Rafique, W., Khan, M., Khan, S., & Ally, J. S. (2023). SecureMed: A Blockchain-Based Privacy-Preserving Framework for Internet of Medical Things. *Wireless Communications and Mobile Computing*, 2023.

[9] Resende, A, Railsback, D, Dowsley, R, Nascimento, ACA & Aranha, DF 2022, 'Fast privacy-preserving text classification based on secure multiparty computation', *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 428-442. <https://doi.org/10.1109/TIFS.2022.3144007>

[10] Sai, S., Hassija, V., Chamola, V., & Guizani, M. (2023). Federated learning and NFT-based privacy-preserving medical data sharing scheme for intelligent diagnosis in smart healthcare. *IEEE Internet of Things Journal*.

[11] Sousa, P.R., Antunes, L., Martins, R. (2018). The Present and Future of Privacy-Preserving Computation in Fog Computing. In: Rahmani, A., Liljeberg, P., Preden, JS., Jantsch, A. (eds) *Fog Computing in the Internet of Things*. Springer, Cham. https://doi.org/10.1007/978-3-319-57639-8_4

[12] Stammmler, S., Kussel, T., Schoppmann, P., Stampe, F., Tremper, G., Katzenbeisser, S., ... & Lablans, M. (2022). Mainzelliste SecureEpiLinker (MainSEL): privacy-preserving record linkage using secure multi-party computation. *Bioinformatics*, 38(6), 1657-1668.

[13] Sucasas, V., Aly, A., Mantas, G., Rodriguez, J., & Aaraj, N. (2023). Secure multi-party computation-based privacy-preserving authentication for smart cities. *IEEE Transactions on Cloud Computing*.

[14] Tran, A. T., Luong, T. D., Karnjana, J., & Huynh, V. N. (2021). An efficient approach for privacy preserving decentralized deep learning models based on secure multi-party computation. *Neurocomputing*, 422, 245-262.

[15] Zhang. Q, C. Xin and H. Wu, "Privacy-Preserving Deep Learning Based on Multiparty Secure Computation: A Survey," in *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10412-10429, 1 July1, 2021, doi:10.1109/JIOT.2021.3058638.

CONFERENCE COMPLETION CERTIFICATE



PANIMALAR ENGINEERING COLLEGE

An Autonomous Institution & Affiliated to Anna University, Chennai
Approved by AICTE, New Delhi, Accredited by National Board of Accreditation(NBA)
Approved by UGC for 2(f) & 12(B) Status
Bangalore Trunk Road, Poonamallee, Chennai-600123.



PECTEAM 2K24



This is to certify that Dr./Mr./Ms. **BHARATH LP**of
..... **CSE**from **Sathyabama Institution of Science and technology** has presented paper
entitled **Privacy-Preserving Data Analysis: Perform data analysis on encrypted data
stored in the cloud** in 7th International Conference
on Intelligent Computing(IConIC) held on 22nd & 23rd MARCH, 2024.

FACULTY CO-ORDINATOR

CO - CONVENER

Prof. S. VIMALA, M. TECH., (Ph.D)

CO - PATRON

Dr. K. MANI, M. E., Ph.D.,

