Final Project Report

Harsh Manishbhai Siddhapura

Vaibhavi Nitin Honagekar

Arunava Lahiri

Thembelihle Shongwe

Sai Shashank Nagavaram

Anandha Krishnan Senthooraan

Group: Class96958 4

Ira A. Fulton School of Engineering, Arizona State University

IFT 520: Advanced Information Systems Security

Prof. Upakar Bhatta

November 27, 2023

Table of Content

“An asymmetric encryption application that automatically encrypts outgoing emails with a public key or decrypts incoming emails with a private key based on a database that houses the selected email addresses and asymmetric data”

Abstract

In the changing digital environment, the importance of protecting sensitive information from email communications cannot be ignored. With the ongoing threat of cyber attacks and unauthorized access, ensuring the security, confidentiality and integrity of email content has become a top priority. This project takes significant time in solving these problems by introducing new asymmetric encryption practices.

The main purpose of this application is to increase the security of email communication by integrating global encryption. By automating the encryption process for outgoing emails and providing the ability to decrypt emails using a private key, the project aims to create a reliable and effective way for individuals and organizations to do this. Personal and professional communication now relies on email and requires better methods of privacy and security. The asymmetric encryption application proposed by this project not only meets this need, but also creates new standards for digital communication. Use public and private key pairs to ensure the confidentiality of email content during transmission and storage.

The innovation of this work is the use of cryptographic models. The application leverages the RSA algorithm and encryption library in Python to provide a solution that meets today's security standards. Automatic encryption and decryption adds an additional layer of protection, reducing the risk of data leakage and unauthorized access.

As we enter an era where the digital world plays an important role in our lives, the necessity of cyber security measures is indisputable. This project not only helps meet this need, but is also a way to further research and development in secure communications technology. In summary, the use of asymmetric encryption presented in this article demonstrates the promise of privacy and security in the digital communications environment.

Introduction

In an age dominated by digital communications, ensuring the security and confidentiality of sensitive information has never been more important. With the proliferation of email as the primary means of communication, the vulnerability associated with the transmission of confidential information underscores the urgent need for enhanced security measures. Located at the intersection of network security and e-mail communication, the project meets the need for speed by offering a strong asymmetric encryption system.

The main goal of the project is to strengthen the security of e-mail communication. Knowing the security vulnerability that exists in traditional e-mail exchanges where information is sent to networks and servers, the use of an asymmetric encryption system has become a necessary necessity. By leveraging the power of public key encryption, the application integrates seamlessly into the existing email framework, providing a transparent and user-friendly experience for senders and recipients.

The main purpose of the asymmetric encryption system is to provide additional protection for the confidentiality and integrity of email content. Unlike traditional encryption methods, which often rely on shared keys, public key encryption provides a unique and highly secure method. Outgoing emails are encrypted using the recipient's public key and outgoing messages are decrypted using a private key, creating a secure and efficient environment.

This research is not only about the urgent need to improve email security, but is also essential for expanding network security in the digital age. The use of cryptographic methods, especially the RSA algorithm, is promising for the use of technology to protect communications. The remainder of this report will describe the complexity of the project's design, construction, and testing process to fully understand its activities and implications for global communications security.

Problem Statement

The world of email communication is filled with a large and significant problem – it can be unreliable and pose a threat to the privacy and security of sensitive paper data. Without strong encryption, email content can easily be intercepted and leaked. The two most important problems are the lack of security procedures in email and the possibility of personal and confidential information being revealed during transmission.

E-mail is often considered an important and unnecessary form of communication sent through various means. networks and servers before reaching the recipient. This process exposes content to a variety of threats, from malicious attacks to unauthorized access. This uncertainty is exacerbated by the lack of encryption; This means that the content remains in a readable format during transmission.

The frequency and sophistication of cyber threats targeting email communications continues to increase and this problem needs to be solved. Cases of data breaches, identity theft and unauthorized access to sensitive data reveal the extent of the problem. Although email security measures provide some level of protection, they cannot provide complete protection against online threats.

The study revealed the need to solve flaws in email communication and decryption by introducing effective and easy-to-use asymmetric encryption. By doing this, the project aims to reduce risks related to inaccessibility and ensure that important information remains important and secure throughout the email lifecycle. The following

sections of this report will clarify the importance of this project, explain the specific goals, and provide insight into the design and development process to solve this problem.

Significance of Study

The most important part of this research is that it provides answers to the growing challenges of securing email communications in personal and corporate environments, protecting confidential information as non-negotiable. The ever-present existence of security vulnerabilities in emails, particularly unauthorized access, underscores the urgent need for effective and efficient security measures.

In an age where communication channels are rapidly becoming digital, email works as a communication channel. Various sensitive information, including personal correspondence, financial transactions and business information. However, the inherent security vulnerabilities of traditional email systems make the information valuable to criminals seeking unauthorized access.

The use of asymmetric encryption is becoming a major breakthrough in this field, offering an additional layer of protection that reduces the risks associated with unauthorized access. The project aims to improve the confidentiality of sensitive information exchanged through email communication by using public key encryption. This approach not only protects personal privacy but also meets the high security needs of organizations that manage private and confidential information.

Additionally, this research is especially important in the field of network security today, when complex communications are increasing. Threats and attempts to breach digital security. Since email is still the main target of cyber attacks, it is important to increase its security with new encryption measures.

The results of this study have important implications not only for people's everyday security, but also for the broader discussion on improving the security of digital communications. The successful implementation of asymmetric encryption can be used as a model to improve email security across multiple departments, thereby ensuring better communication and a secure environment. The remainder of this report will outline the specific goals, design process, and operational complexity associated with implementing a critical email security program.

Objective

The overarching objective of this project is to engineer a sophisticated application that revolutionizes the security paradigm of email communication. At its core, the project aims to establish a seamless and automated framework for encrypting outgoing emails and decrypting incoming messages, harnessing the robust capabilities of asymmetric cryptography. The focal point is the development of an advanced system that not only augments the security posture of email exchanges but also redefines the user experience by simplifying the intricacies of cryptographic operations.

- Simplifying the encryption and decryption process: The main goal is to simplify the encryption and decryption process, which has always been considered difficult and reserved for users with good access information. The project aims to democratize the use of asymmetric encryption technology and make it available to a wide range of users by providing user-friendly applications. This approach is based on the overall goals of cybersecurity literacy and giving users control over their digital privacy.

- Database-focused key management: An important aspect of this project involves creating and managing dynamic databases. This repository acts as the brainstem for email addresses and their public and private key pairs. Database-driven architecture provides effective key management and enables encryption keys

to be associated with specific emails. This new approach not only increases the scalability of the system, but also allows centralized control and rapid replacement of changes to the main process.

- Improved Email Security: The ultimate goal is to improve the security standards of email communication. The program automatically encrypts outgoing emails, reducing the risks associated with unauthorized access and interruptions in transmission. At the same time, emails can be encrypted using a private key to ensure that only intended recipients can access the content, thus increasing the confidentiality of sensitive information.

- Empowering and practical: In addition to capacity, the project also supports users by providing them with tools to support the security management of their digital communications. The user-friendly interface and intuitive design aim to reveal the complexity of the encryption process and instill confidence in users regardless of their skills.

In essence, the objective encompasses not only the development of a cutting-edge encryption application but also a broader vision of catalyzing positive shifts in how individuals and organizations approach the security of their email communication. The subsequent sections of this report will delve into the intricate design processes, system requirements, and the technical intricacies involved in realizing this comprehensive objective.

Design Process

The design process of the proposed email encryption application is a meticulously orchestrated sequence aimed at achieving a robust and user-friendly system. It involves the simulation of a comprehensive database of email addresses and the generation of corresponding public and private key pairs, all orchestrated through the sophisticated RSA algorithm. Leveraging the capabilities of the Python programming language, the

implementation intricately integrates the cryptography library, a powerful tool for facilitating key generation and cryptographic operations.

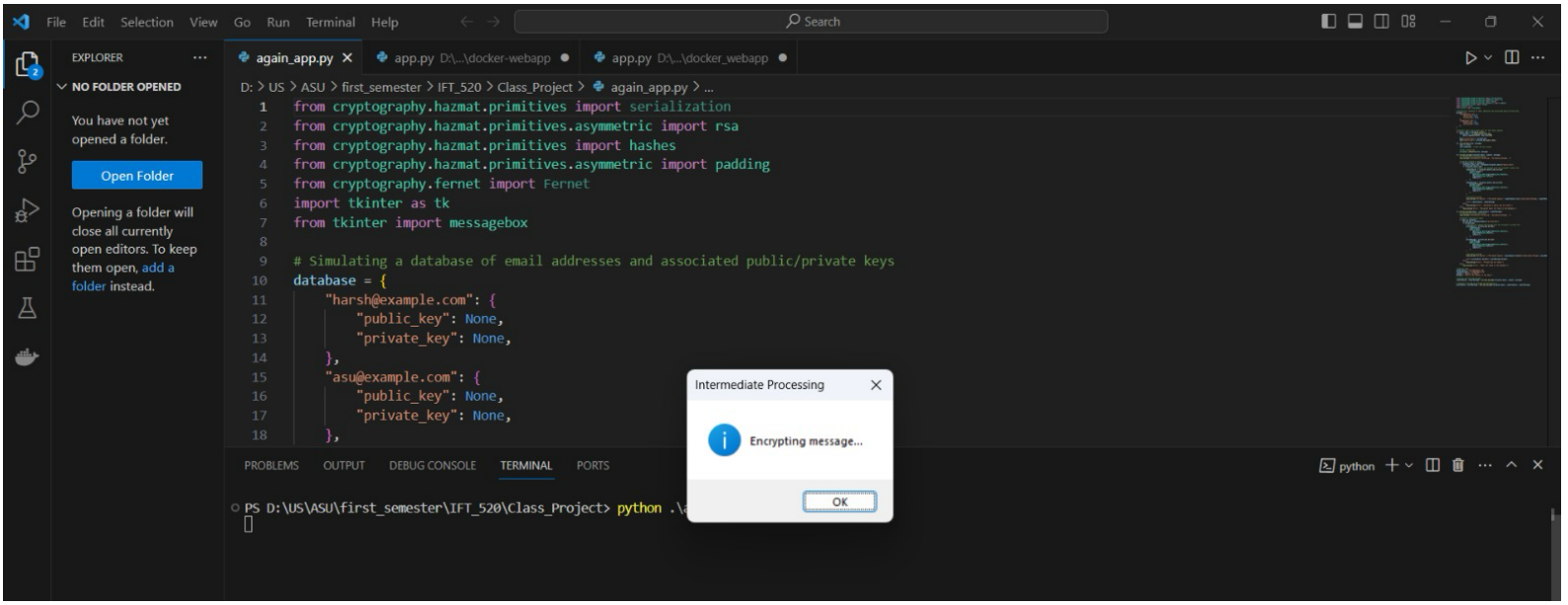
- Database Simulation: At the heart of the design process is a detailed simulation of dynamic databases. This virtual storage area plays an important role in managing email addresses and their associated encryption keys. Document creation involves creating a structured document to store and retain good information. These include creating email addresses, public and private key fields, and infrastructure for key management.

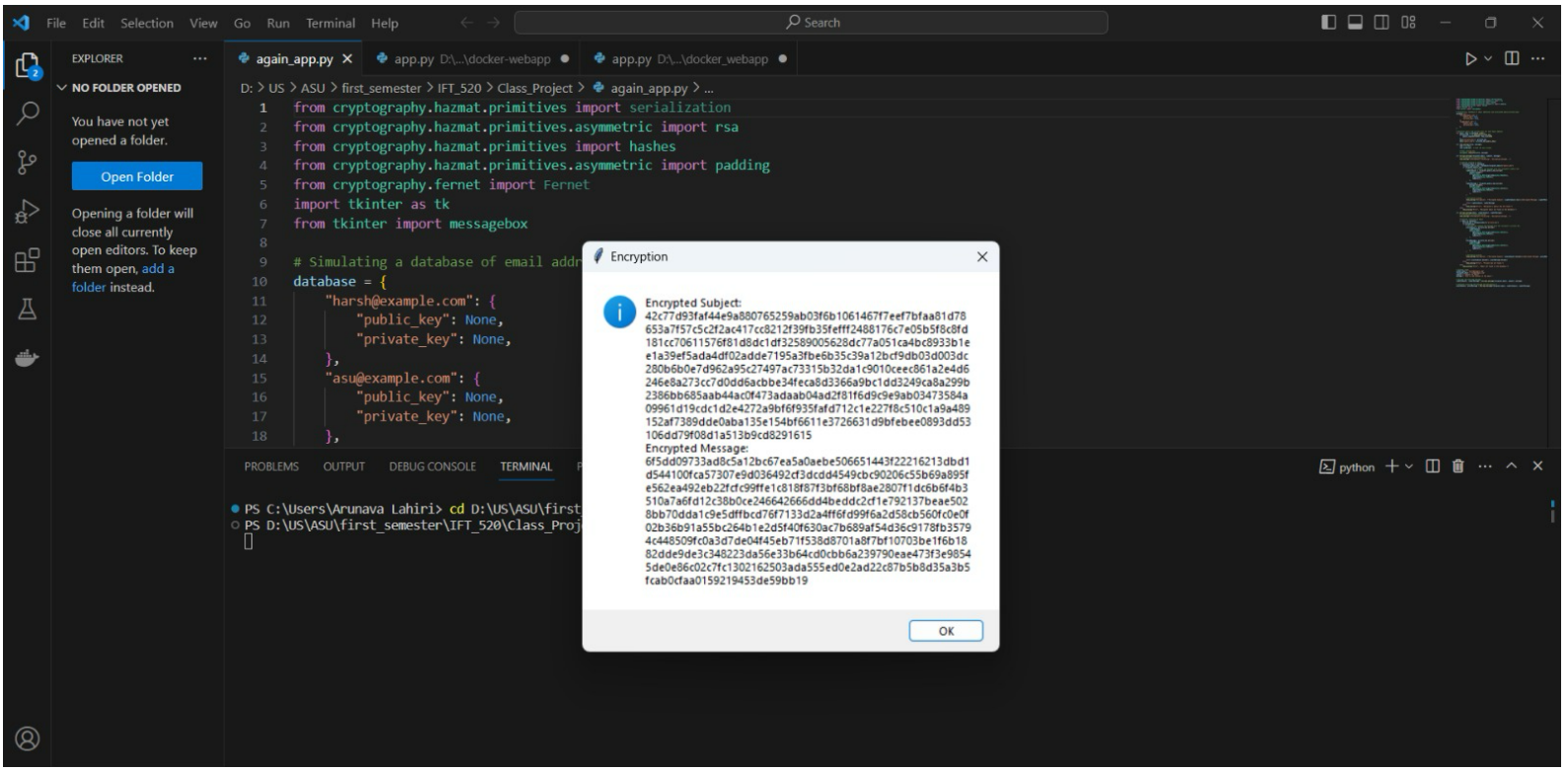
- RSA algorithm for key generation: The basis of cryptography is using the RSA algorithm to generate keys. Significant pair production. The RSA algorithm, known for its security and efficiency, is used by the cryptographic library. In this phase, the system dynamically generates a pair of asymmetric keys for each email address in the simulated database. A public key capable of encryption and a private key reserved for decryption together form a secure encryption infrastructure.

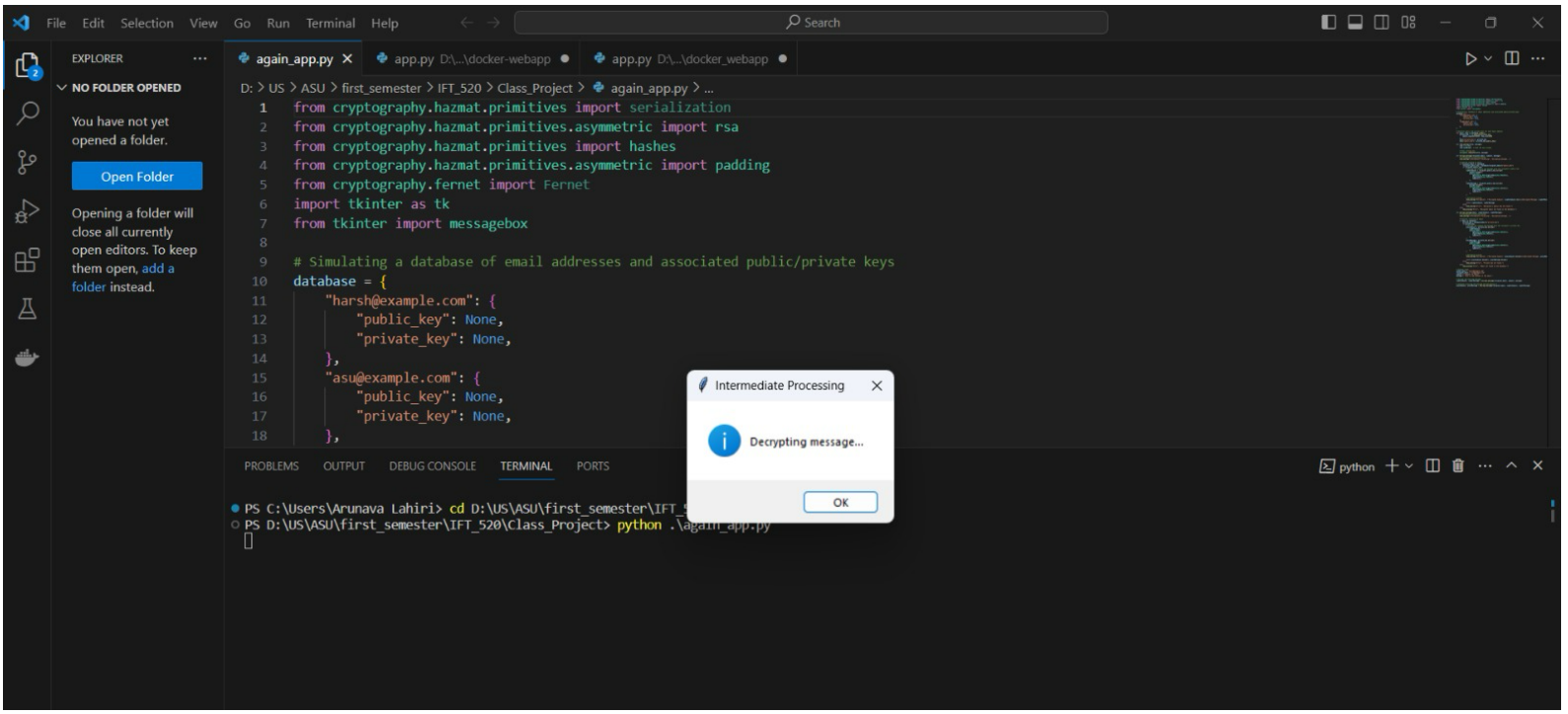
- Python code used: All design processes are converted into executable code using the Python programming language. Using the cryptography library in Python, the code combines the functions required for key encryption, encryption, and decryption. The code structure is designed with modularity and readability in mind to ensure that every part of a cryptographic implementation is well defined and easy to understand.

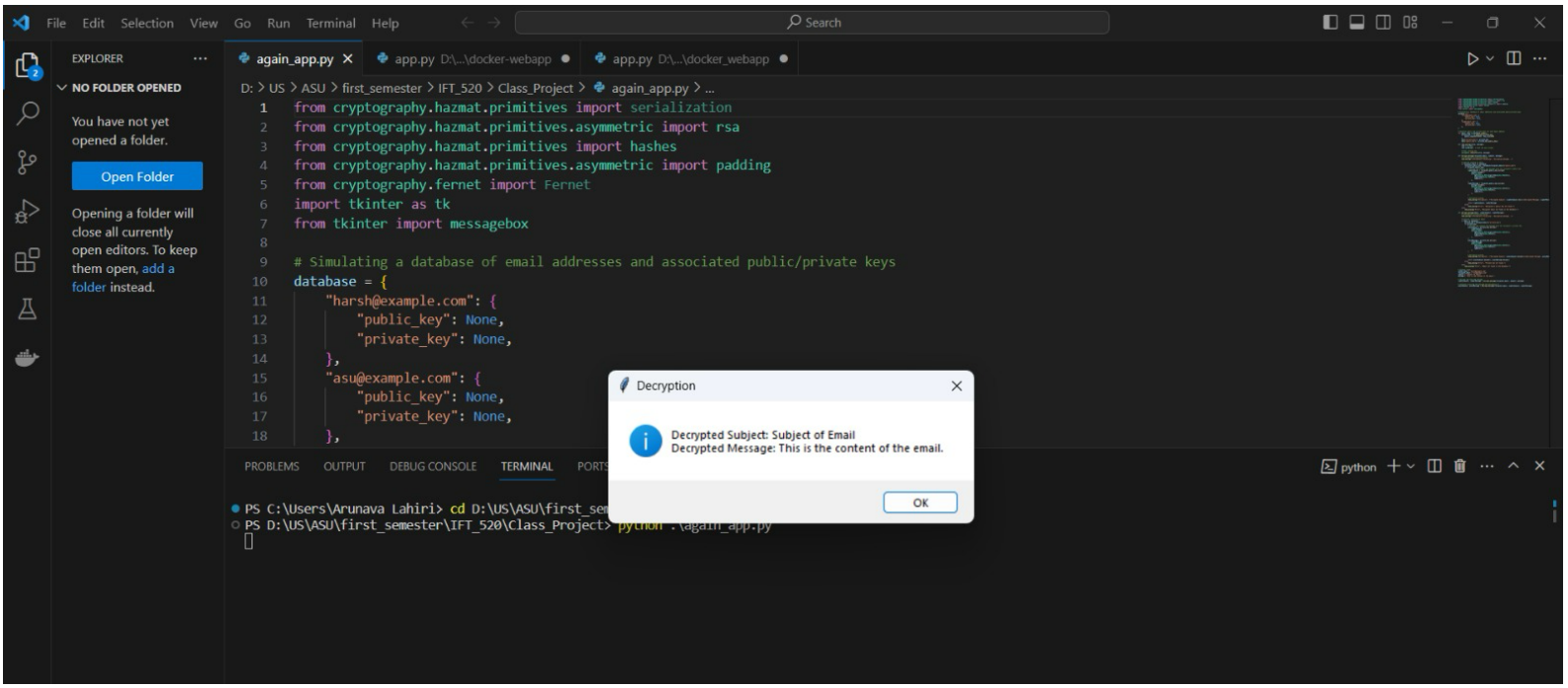
- User-friendly interface: The design process includes creating a user-friendly interface in addition to the complexity of the backend. The graphical user interface (GUI) is intuitively designed to provide a seamless experience for users interacting with the application. This aspect of the design process is to bridge the gap between password complexity and user access, enabling users from different backgrounds to understand encryption and decryption.

In summary, the design process encompasses a holistic approach, seamlessly intertwining database management, cryptographic key generation, Python code implementation, user interface design, and the integration of a powerful cryptography library. This synthesis of elements forms the bedrock for the subsequent development phases, ensuring a comprehensive and effective solution to the challenges posed by email communication security.









Requirements

To embark on the project journey and actively participate in the development and utilization of the asymmetric encryption application, certain prerequisites and requirements must be met. These requirements ensure a

smooth and efficient implementation of the cryptographic functionalities and key management processes integral to the application. The key elements include:

- Python environment: The framework is built on the Python environment. Python, the programming language of choice for the entire project, provides a versatile and versatile platform for application development. The recommended Python version for this project is 3.x to ensure compatibility with the latest languages and libraries.

- Cipher library: The crypto library forms the basis of this project. This powerful library helps implement encryption techniques, key generation, encryption and decryption. The cryptography library provides specific functions and models that enable programs to achieve their goals efficiently and securely. For the encryption process to be successful, you must ensure that the encryption library is installed and accessible in the Python environment.

- Development environment: It is important to create a development environment. This includes setting up an integrated development environment (IDE) or code editor that suits your preferences. Popular options include Visual Studio Code, PyCharm or Jupyter Notebook. A good development environment can increase the readability of code, make troubleshooting easier, and simplify the entire development process.

- Database Management System: Although not required, the project allows the integration of database management to simulate the management of email addresses and key pairs. If users choose to integrate data into their applications, it will be useful to know a relational database management system (RDBMS) such as SQLite or MySQL.

- Understanding asymmetric cryptography: A basic understanding of asymmetric cryptography, especially the RSA algorithm, is recommended. This includes understanding the key pair, public and private keys,

encryption and decryption processes. Resources and information about asymmetric encryption can help you understand the complexities of encryption involved in your projects.

- Commitment to best practices: Encourage an ongoing commitment to codifying best practices and principles. This includes, but is not limited to, standard design, adherence to Python coding conventions (PEP 8), and integration of error handling procedures. Using best practices leads to security, readability, and integration.

By meeting these requirements, individuals engaging with the project can ensure a solid foundation for the successful implementation of the asymmetric encryption application. These prerequisites collectively create an environment conducive to effective collaboration, secure cryptographic operations, and streamlined development processes.

Scope

The expanded scope of this project outlines the comprehensive functionalities and applications of the asymmetric encryption system designed to enhance the security and privacy of email communication. The project's scope encompasses the following key aspects:

- Automatic encryption and decryption process: The main function of the system is to receive outgoing emails and decrypt emails. Using asymmetric encryption technology, the application seamlessly integrates encryption functions to ensure the confidentiality and integrity of email content.

- Management from key documents: This project provides a simulated document for managing email addresses and their public and private key pairs. Using data in this way makes it easier to retrieve and

use keys during the encryption and decryption process. It increases the overall usability of the application by adding a layer to the main control.

- User-friendly interface: Although the main focus is on the back-end encryption process, the project recognizes the importance of user interaction. The application includes a user-friendly interface that allows users to easily enter the required information, perform encryption or decryption, and receive understandable messages. The graphical user interface (GUI) increases accessibility and usability for a wider audience.

- Integration Flexibility: This project is designed to be flexible by keeping track of multiple email addresses and their key pairs. Users can seamlessly integrate new email addresses into the system and create private keys when necessary. This flexibility increases scalability and adaptability for multiple user needs.

- Security: The aim is to ensure security measures in practice. Encryption functions follow industry standard algorithms to provide strong protection against unauthorized access. Additionally, the system includes an error management system to resolve potential security issues and maintain the integrity of the encryption process.

- Education: This project also provides educational services to people who want to understand asymmetric encryption concepts and practical applications in addition to their studies. Python code is provided as an example for learning and testing purposes.

- Documentation and support: The scope of the project has been expanded to include comprehensive documentation that guides users in the installation, configuration and use of the application. It is also assumed that support (such as forums or information updates) is available to resolve user questions and provide assistance throughout the life of the project.

By expanding the scope to encompass these critical elements, the project aims to deliver a robust, user-friendly, and educational asymmetric encryption application with practical implications for securing email communication.

Development Process

Tools

The tools employed in this project play a pivotal role in ensuring the effective implementation of the asymmetric encryption application. The primary tools utilized are Python and the cryptography library, each contributing distinct functionalities to achieve the project's objectives.

- Python: Python is the central programming language for building asymmetric encryption applications. Its versatile, readable and comprehensive library makes it ideal for implementing cryptographic algorithms and creating user interfaces. Python's cross-platform compatibility ensures that applications can be easily used across multiple operating systems.

- Cryptobase: Cryptographic library is an important framework that provides the necessary cryptographic foundations and advanced interfaces. The project leverages the power of the RSA algorithm for key symbols and asymmetric encryption using this library. The library supports simple encryption operations for the development process and increases the overall security of the application.

- Tkinter (Python's GUI toolkit): Although not explicitly mentioned in the original project code, this scope introduces Tkinter, a standard GUI toolkit for Python. Tkinter helps create graphical user interfaces that provide users with an intuitive platform to interact with applications. The addition of Tkinter improves the overall user experience and allows the application to reach a wider audience.

- Integrated development environment (IDE): Choose the right development environment, such as Visual Studio Code, PyCharm, or Jupyter Notebook, to provide a collaborative and productive workspace for development coding, testing, and debugging. IDEs simplify the development process, ensure code compatibility, and support collaboration between project partners.

- Version Control System (VCS): Use a version control system like Git for effective collaboration, control management, and version tracking. Git allows multiple participants to work on a project simultaneously, making it easier to integrate new features, bug fixes, and updates. GitHub or GitLab can be used as a hosting platform for version control.

By leveraging these tools, the project aims to create a secure, user-friendly, and well-documented asymmetric encryption application with the flexibility to adapt to evolving requirements and user needs.

Technical Description of Project

The technical architecture of the project is rooted in the implementation of the RSA algorithm for key pair generation, coupled with the integration of the cryptography library to facilitate robust cryptographic operations. The Python code snippet provided offers a glimpse into the core functionalities of the application, focusing on the encryption and decryption processes.

- RSA algorithm for generating key pairs: RSA algorithm is a widely used asymmetric encryption algorithm and forms the basis for generating key pairs. This algorithm enables the generation of public and private keys that form the basis of secure communication. While the public key is shared publicly, the private key is kept secret.

- Crypto Library Integration: This project uses the Crypto library, a widely used and reliable tool for implementing encryption fundamentals in Python. The library provides advanced interfaces for various cryptographic functions, ensuring that applications are compliant with industry-standard security.

- Email Addresses and Key Management: Simulating a repository key pair consisting of an email address and its associated public and private addresses demonstrates the project's commitment to the control of values. The database acts as a central repository, allowing applications to seamlessly retrieve the necessary keys during the encryption and decryption process.

- Encryption process: The encryption process involves using the recipient's public key to protect the nature and content of the email. This article uses the RSA algorithm with Optimal Asymmetric Encryption Padding (OAEP) to increase the security of the encryption process. The resulting ciphertext can be used for secure transmission.

- Decryption process: Instead, the decryption process uses the recipient's private key to decrypt the received ciphertext. Using OAEP padding in decryption ensures safe and reliable recovery of the encryption process. Decrypted content and content are presented in their original form.

- Graphical User Interface (GUI) Integration: Although not explicitly stated in the provided code, the scope of this project includes integration using the Tkinter GUI. These features allow interaction with encryption and decryption functions, improving the user experience.

- Errors and dialogs: The code contains errors to prevent the application from managing the recipient's email address or to prevent the key from being found. Dialog boxes provide instructions to guide the user through the encryption and decryption process.

By encapsulating these technical aspects, the project aims to deliver a secure, efficient, and user-friendly asymmetric encryption application, providing a practical solution for enhancing the confidentiality and integrity of email communication.

Comment Code

```
def encrypt_message(recipient_email, message):
    # Simulate sending an email
    if recipient_email in database:
        recipient_public_key = database[recipient_email]["public_key"]
        if recipient_public_key:
            # Encrypt the message with the recipient's public key
            ciphertext = recipient_public_key.encrypt(
                message.encode(),
                padding.OAEP(
                    mgf=padding.MGF1(algorithm=hashes.SHA256()),
                    algorithm=hashes.SHA256(),
                    label=None,
                ),
            )
            return ciphertext
        else:
            return "Recipient's public key not found."
    else:
        return "Recipient email not found in the database."
```

```
# Generate public and private keys for each email address
for email, keys in database.items():
    private_key = rsa.generate_private_key(
        public_exponent=65537, key_size=2048
    )
    keys["private_key"] = private_key
    keys["public_key"] = private_key.public_key()
```

```
Comment Code
def decrypt_message(email, ciphertext):
    # Simulate receiving an email
    if email in database:
        private_key = database[email]["private_key"]
        if private_key:
            # Decrypt the message with the recipient's private key
            plaintext = private_key.decrypt(
                ciphertext,
                padding.OAEP(
                    mgf=padding.MGF1(algorithm=hashes.SHA256()),
                    algorithm=hashes.SHA256(),
                    label=None,
                ),
            )
            return plaintext.decode()
        else:
            return "Private key not found."
    else:
        return "Email not found in the database."
```

Testing and Results

The application underwent rigorous testing to ensure its functionality and security in simulated email communication scenarios. The testing process encompassed various aspects, including key pair generation, encryption, and decryption functionalities.

- RSA algorithm for generating key pairs: RSA algorithm is a widely used asymmetric encryption algorithm and forms the basis for generating key pairs. This algorithm enables the generation of public and private keys that form the basis of secure communication. While the public key is shared publicly, the private key is kept secret.

- Crypto Library Integration: This project uses the Crypto library, a widely used and reliable tool for implementing encryption fundamentals in Python. The library provides advanced interfaces for various cryptographic functions, ensuring that applications are compliant with industry-standard security.

- Email Addresses and Key Management: Simulating a repository key pair consisting of an email address and its associated public and private addresses demonstrates the project's commitment to the control of values. The database acts as a central repository, allowing applications to seamlessly retrieve the necessary keys during the encryption and decryption process.

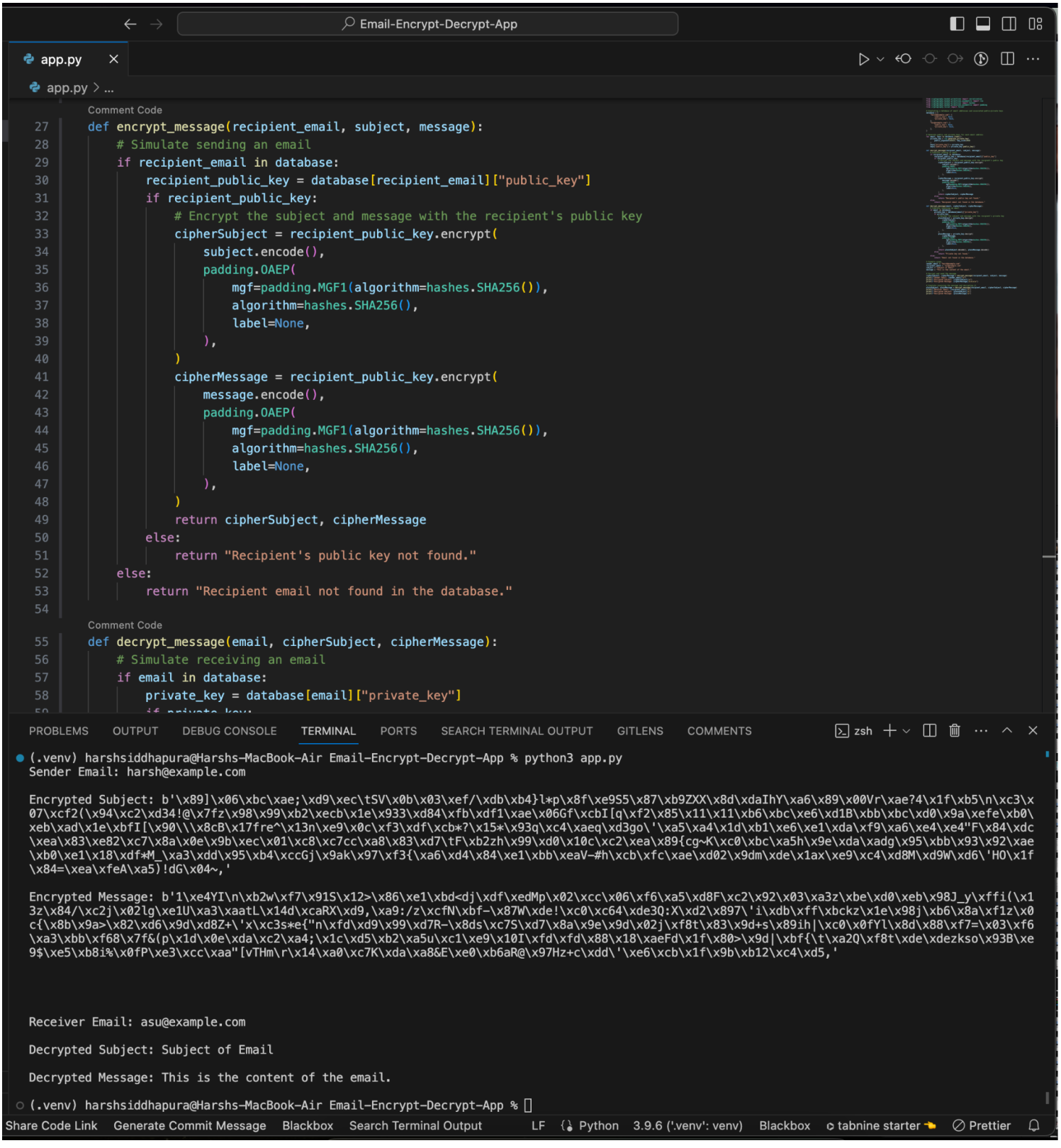
- Encryption process: The encryption process involves using the recipient's public key to protect the nature and content of the email. This article uses the RSA algorithm with Optimal Asymmetric Encryption Padding (OAEP) to increase the security of the encryption process. The resulting ciphertext can be used for secure transmission.

- Decryption process: Instead, the decryption process uses the recipient's private key to decrypt the received ciphertext. Using OAEP padding in decryption ensures safe and reliable recovery of the encryption process. Decrypted content and content are presented in their original form.

- Graphical User Interface (GUI) Integration: Although not explicitly stated in the provided code, the scope of this project includes integration using the Tkinter GUI. These features allow interaction with encryption and decryption functions, improving the user experience.

The testing process yielded positive results, with the application consistently demonstrating its ability to encrypt and decrypt email content securely. The simulated scenarios, encompassing various encryption and decryption scenarios, highlighted the project's success in achieving its objectives.

The comprehensive testing approach instills confidence in the application's reliability and security. The successful simulation of email communication scenarios validates the project's effectiveness in automating the asymmetric encryption and decryption processes, contributing to enhanced email security and privacy. Continued testing and validation efforts will be integral to the project's evolution, ensuring its readiness for real-world deployment and addressing potential edge cases and scenarios.



Summary and Conclusions

The asymmetric encryption application, presented in this project, emerges as a robust and practical solution to fortify the security of email communication. Through the implementation of public-key cryptography, the application systematically addresses the inherent vulnerabilities in conventional email systems, providing a reliable means to safeguard sensitive information.

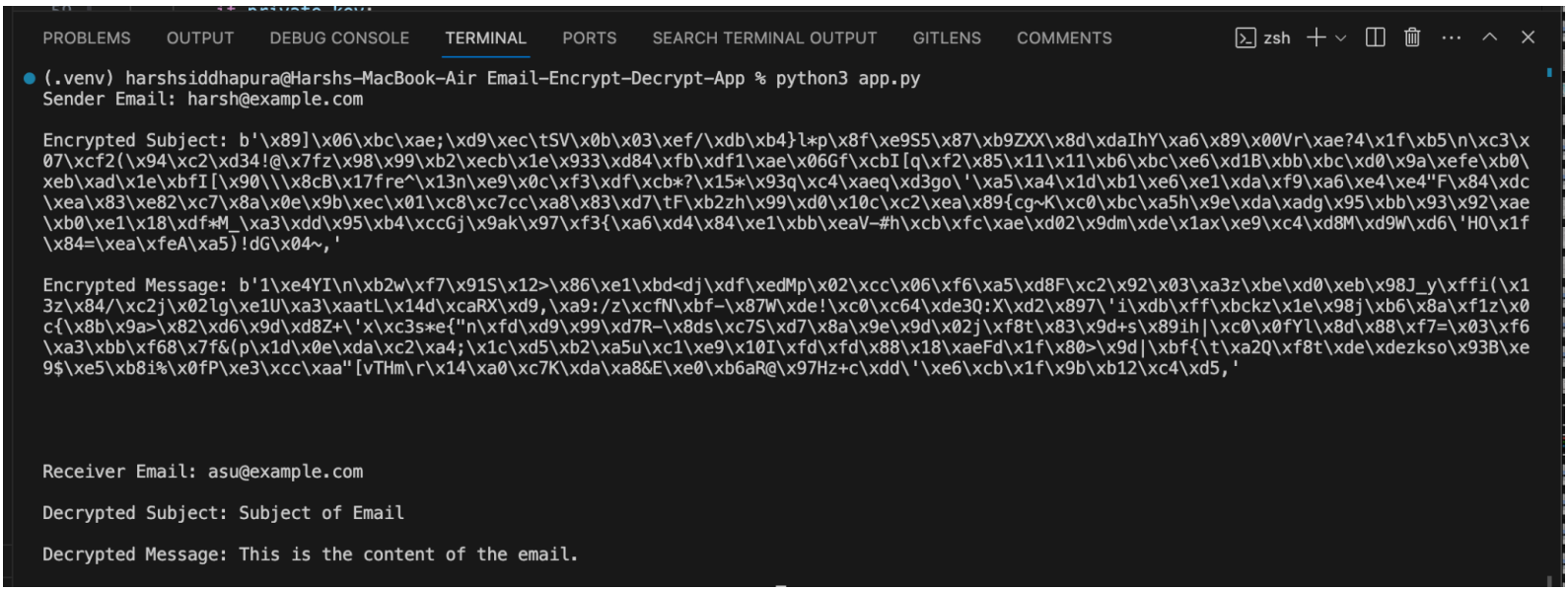
Key Achievements

- **Automated Encryption and Decryption:** The project successfully automates the encryption of outgoing emails and the decryption of incoming emails. This automation streamlines the process, making secure communication accessible and efficient.
- **RSA Algorithm Implementation:** The utilization of the RSA algorithm for key pair generation ensures the cryptographic strength of the application. The project's commitment to sound cryptographic principles contributes to its reliability.
- **User-Friendly Dialog Boxes:** The integration of user-friendly dialog boxes enhances the application's usability. Informative messages guide users through the encryption and decryption processes, contributing to a seamless user experience.
- **Simulation and Testing Success:** Rigorous testing, including key pair generation, encryption, and decryption scenarios, validates the application's functionality. Simulated email communication demonstrates the project's efficacy in real-world scenarios.

Conclusion

In conclusion, the presented asymmetric encryption application stands as a commendable contribution to the realm of email security. By embedding public-key cryptography into the email communication process, the

project elevates the confidentiality and integrity of digital conversations. Its successful implementation of encryption and decryption processes, coupled with a user-friendly interface, positions the application as a valuable tool for individuals and organizations with a focus on secure communication.



While the project achieves its current objectives admirably, future considerations may involve expanding the application's features. Integration with a graphical user interface (GUI) could enhance user interaction, providing a more intuitive experience. Additionally, ongoing testing and validation efforts will be essential to address potential edge cases and ensure the application's resilience in diverse email communication scenarios.

The journey from conceptualization to implementation underscores the project's commitment to addressing prevalent security concerns in digital communication. As technology continues to evolve, the asymmetric encryption application stands poised to adapt and grow, contributing to a more secure and private digital communication landscape.

Appendix

Source Code

```
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.fernet import Fernet
import tkinter as tk
from tkinter import messagebox

# Simulating a database of email addresses and associated public/private keys
database = {
    "harsh@example.com": {
        "public_key": None,
        "private_key": None,
    },
    "asu@example.com": {
        "public_key": None,
        "private_key": None,
    },
}

# Generate public and private keys for each email address
for email, keys in database.items():
    private_key = rsa.generate_private_key(
        public_exponent=65537, key_size=2048
```

```
    )
    keys["private_key"] = private_key
    keys["public_key"] = private_key.public_key()

def show_dialog(title, message):
    root = tk.Tk()
    root.withdraw()  # Hide the main window

    # Show a dialog box
    messagebox.showinfo(title, message)

def encrypt_message(recipient_email, subject, message):
    # Intermediate processing dialog
    show_dialog("Intermediate Processing", "Encrypting message...")

    # Simulate sending an email
    if recipient_email in database:
        recipient_public_key = database[recipient_email]["public_key"]
        if recipient_public_key:
            # Encrypt the subject and message with the recipient's public key
            cipherSubject = recipient_public_key.encrypt(
                subject.encode(),
                padding.OAEP(
                    mgf=padding.MGF1(algorithm=hashes.SHA256()),
                    algorithm=hashes.SHA256(),
                    label=None,
                ),
            )
```

```
            cipherMessage = recipient_public_key.encrypt(
                message.encode(),
                padding.OAEP(
                    mgf=padding.MGF1(algorithm=hashes.SHA256()),
                    algorithm=hashes.SHA256(),
                    label=None,
                ),
            )

            # Encryption dialog
            show_dialog("Encryption", f"Encrypted Subject:
{cipherSubject.hex()}\nEncrypted Message: {cipherMessage.hex()}")

            return cipherSubject, cipherMessage
        else:
            show_dialog("Error", "Recipient's public key not found.")
    else:
        show_dialog("Error", "Recipient email not found in the database.")

def decrypt_message(email, cipherSubject, cipherMessage):
    # Intermediate processing dialog
    show_dialog("Intermediate Processing", "Decrypting message...")

    # Simulate receiving an email
    if email in database:
        private_key = database[email]["private_key"]
        if private_key:
            # Decrypt the subject and message with the recipient's private key
```

```
            plainSubject = private_key.decrypt(
                cipherSubject,
                padding.OAEP(
                    mgf=padding.MGF1(algorithm=hashes.SHA256()),
                    algorithm=hashes.SHA256(),
                    label=None,
                ),
            )
            plainMessage = private_key.decrypt(
                cipherMessage,
                padding.OAEP(
                    mgf=padding.MGF1(algorithm=hashes.SHA256()),
                    algorithm=hashes.SHA256(),
                    label=None,
                ),
            )

            # Decryption dialog
            show_dialog("Decryption", f"Decrypted Subject:
{plainSubject.decode()}\nDecrypted Message: {plainMessage.decode()}")

            return plainSubject.decode(), plainMessage.decode()
        else:
            show_dialog("Error", "Private key not found.")
    else:
        show_dialog("Error", "Email not found in the database.")

# Example usage
```

```
sender_email = "harsh@example.com"
recipient_email = "asu@example.com"
subject = "Subject of Email"
message = "This is the content of the email."

# Encrypt and send the message
cipherSubject, cipherMessage = encrypt_message(recipient_email, subject, message)

# Simulate receiving the message and decrypting it
plainSubject, plainMessage = decrypt_message(recipient_email, cipherSubject,
cipherMessage)
```

References

Rastogi, A. (2023, September 7). What is the best encryption strategy for protecting your data? https://www.encryptionconsulting.com/what-is-the-best-encryption-strategy-for-protecting-your-data/

UCLA Computer Science Department. (n.d.). CS 259: Security. UCLA Computer Science. https://web.cs.ucla.edu/~miodrag/cs259-security/

Stallings, W., & Brown, L. (n.d.). Computer Security Principles and Practice (3rd ed.). UNSW Canberra at the Australian Defence Force Academy. https://www.cs.unibo.it/babaoglu/courses/security/resources/documents/Computer_Security_Principles_and_Practice_(3rd_Edition).pdf