DEVELOPMENT OF AN ASYMMETRIC EMAIL ENCRYPTION APPLICATION FOR ENHANCED EMAIL SECURITY AND PRIVACY

Table Of Contents

Problem Statement

- Presently, email communication faces security gaps in transmission which has the absence of end-to-end encryption, complexities in asymmetric encryption, and dependency on external services.
- To tackle these issues, the project intends to create an email encryption application to improve security, streamline user interactions, and ensure confidentiality by employing advanced asymmetric encryption and a strong database infrastructure.
- The objective is to set a new standard for email security that is both resilient and user-friendly.

Significance of Study

- ➔ This study presents an innovative application employing asymmetric encryption to boost email security.
- ➔ It automatically encrypts outgoing emails, allowing only the designated recipient to decrypt using their private key, ensuring data privacy and security.
- ➔ Prioritizing user-friendliness, it streamlines the encryption process, facilitating user acceptance.
- ➔ It supports legal adherence by aligning with data protection regulations such as GDPR and plays a role in bolstering overall cybersecurity by mitigating potential threats.

Objective

The objective of the project is to create a secure and encrypted email system using the steps as follows:

- ➔ Generate a Secret Key
- ➔ Scramble the Email Message
- ➔ Secure Delivery
- ➔ Utilize the Cryptography Library
- ➔ Integration with Email Platforms

Design Process

Thorough Design Process	User - Friendly Interface	Precedence of Security	Rigorous Testing
The design process for the Python-based email encryption system is a meticulous and thorough undertaking that balances technical intricacies with user-centric considerations.	The user interface is simple and accessible, with easy-to-use features allowing users security in emails without technical knowledge.	Security measures take precedence, addressing potential vulnerabilities such as man-in-the-middle attacks.	Thorough testing, including unit, integration, and security tests, ensures the reliability and effectiveness of the email encryption system.

Requirements

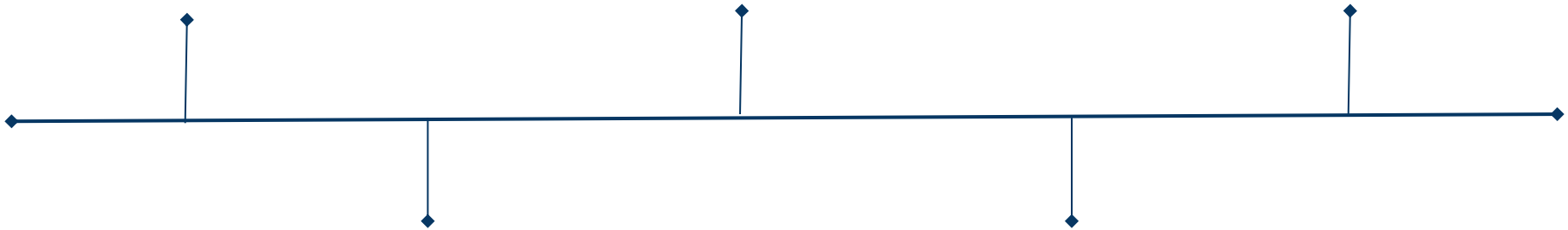
- ➔ User authentication mechanisms play a central role, involving the integration of secure user registration and login processes to safeguard user accounts.
- ➔ Robust mechanisms for secure key exchange must be implemented.
- ➔ Conforming to pertinent data protection and privacy regulations, such as GDPR and HIPAA.
- ➔ The provision of comprehensive documentation tailored for users, administrators, and developers is crucial to facilitate a clear understanding of the system, its installation procedures, and troubleshooting processes.

Scope Of Project

Key Elements
A strong key management system guarantees the secure creation, storage, and retrieval of both public and private keys.

Security Measures
Implementing strong mechanisms for secure key exchange, preventing unauthorized access by countering potential man-in-the-middle attacks.

Testing & Documentation
Rigorous testing is conducted, covering both unit and security testing.



Seamless & Simple Interface
The program prioritizes user-friendliness with a simplified one-click encryption & decryption interface, making email security accessible without requiring technical expertise.

Regulatory Compliance
A critical aspect of the program's design is its compliance with data protection regulations, including GDPR and HIPAA.

Developmental Process

- The primary focus is on safeguarding data integrity and confidentiality, employing the SHA algorithm for robust hashing.
- Utilizing dedicated libraries such as 'cryptography' and 'pycryptodome,' the project seamlessly incorporates asymmetric encryption algorithms, ensuring secure key.
- Streamlining integration with popular email platforms, the initiative utilizes Python's SMTP libraries to automate encryption and decryption processes.
- Thorough testing, unit tests, guarantees robustness, with additional security testing tools addressing potential vulnerabilities.

Tools used

01

Programming Tool: Python

Python readability, versatility, and rich library support make it ideal for developing email encryption, excelling in algorithms, key management, authentication, and interface design.

02

SHA and OAEP encryption

SHA, a secure hashing algorithm, enhances data and certificate hashing. OAEP, Optimal Asymmetric Encryption Padding, is the primary RSA encryption padding, formatting messages for heightened security.

03

SMTP Libraries

These libraries facilitate the automatic encryption of outgoing emails and decryption of incoming emails within the user's email client.

04

Testing Tools

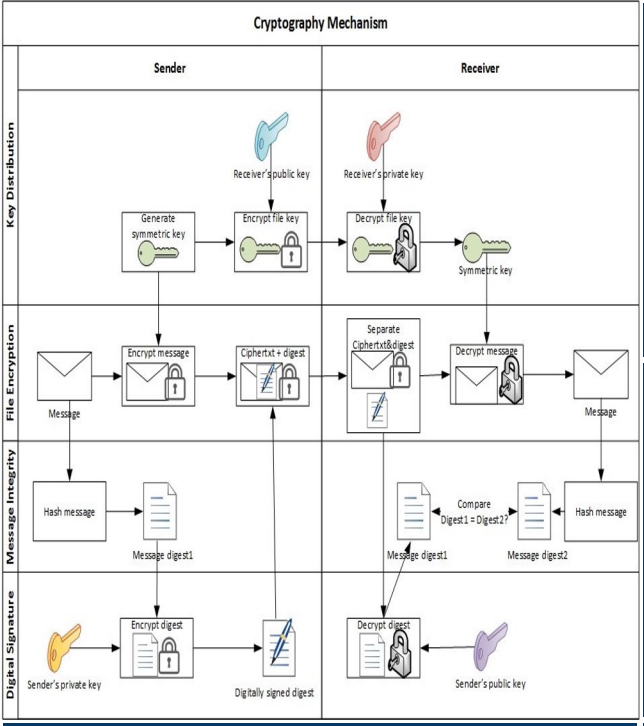
Ensuring the effectiveness of the email encryption system demands rigorous testing. Python offers powerful testing tools like unit tests for comprehensive testing.

Technical Description

This Python script establishes a secure email system through encryption and decryption.' The 'generate_or_load_keys' function dynamically creates or retrieves key pairs for emails.

'encryption_of_file' secures files using public keys and OAEP encryption, while 'decryption_of_file' decrypts with the user's private key. 'send_encrypted_email_with_attachment' connects to Gmail SMTP, sending encrypted emails.

'receive_and_decrypt_emails_with_attachment' decrypts Gmail IMAP attachments. While foundational, real-world implementation would necessitate heightened security measures



Workflow of Project

01

Key Generation: Utilizes `generate_or_load_keys` to check and generate RSA keys for user emails, storing them in the database.

02

File Encryption: Employs `encryption_of_file` to encrypt specific text files using the recipient's RSA public key with OAEP.

03

Sending Encrypted Email: Establishes an SMTP connection to Gmail, sending emails with encrypted attachments using `send_encrypted_email_with_attachment`.

04

Email Reception and Decryption on Recipient's Side: Connects to Gmail IMAP, retrieves and decrypts received emails with encrypted attachments using `receive_and_decrypt_emails_with_attachment`

05

File Decryption on Recipient's Side: Saves decrypted files on the recipient's side after successful decryption.

06

Execution and Result: Initializes parameters and demonstrates secure email communication with encryption, including successful sending, receiving, and file decryption.

Summary and conclusion

- The Python-based email encryption system prioritizes security through advanced cryptography, utilizing robust algorithms via Python's cryptography library.
- With a user-centric approach, it emphasizes friendly design, robust key management, and regulatory compliance, offering a comprehensive solution for evolving email security challenges.
- Rigorous testing and detailed documentation ensure the reliability of the system, focusing on secure key generation, encryption, and delivery, making it a dependable safeguard for email privacy in the digital landscape.

References

- Byrne, D. (2021). *Full stack python security: Cryptography, TLS, and attack resistance*. Manning Publications Co.
- Desai, A. (2000). The security of all-or-nothing encryption: Protecting against exhaustive key search. *Advances in Cryptology — CRYPTO 2000*, 359–375. https://doi.org/10.1007/3-540-44598-6_23
- Khan, A. (2020, April 24). *Send secure emails using python in few lines of code*.
- Medium.https://medium.com/@mamir.khan/send-secure-emails-using-python-in-few-lines-of-code
- Simple python encryption: How to encrypt A message | codementor. (n.d.).
- https://www.codementor.io/python/tutorial/python-encryption-message-in-python-via-reverse-cipher

Thank You!