MIS 202 - NETWORK SECURITY

Midterm project - Secure Communication Practice

Name: Onat

Surname: Karabulut

Student No: 230303305

Departmant: Management Information Systems (MIS)

Date Delivered: 14.04.2024

Course Instructor: Ahmet Samet Yüzlü

Project Link:

https://github.com/onatkarabulut/MIS202_end_to_end_end_encryption_w_kafka

Techs used:

1- Programming Language: Python

2- Backend & API Development: FastAPI

3- Message Streaming: Apache Kafka & Zookeeper (Bootstrap Server)

Midterm Case & End-to-End Message Encryption Project

Midterm Case:

- **Secure Communication Practice**
    - **Objective:** Understand the principles of secure communication.
    - **Tasks:** Students use secure messaging apps (such as Signal or WhatsApp) to communicate and discuss the encryption protocols these apps use (e.g., end-to-end encryption). They present on how these apps ensure the privacy and security of communications.

[EN]

Encryption Algorithms:

Encryption algorithms are mathematical operations that enable data to be transmitted and stored securely. They ensure that the transmitted data is understood only by the recipient and prevent malicious people from accessing the data. Here are some of the most commonly used encryption algorithms:

1. **Symmetric Encryption Algorithms:** These algorithms are a system in which the same key is used in both encryption and decryption processes. The same key is used to encrypt and decrypt the data. For example, algorithms such as AES (Advanced Encryption Standard) and DES (Data Encryption Standard) are examples of symmetric encryption algorithms.
2. **Public Key (Asymmetric) Encryption Algorithms:** These algorithms are a system where two different keys are used: one to encrypt the data (public key) and one to decrypt it (private key). The data is encrypted using the recipient's public key before sending it, but can only be decrypted with the recipient's private key. RSA (Rivest-Shamir-Adleman) is one such algorithm.

What is RSA (Rivest-Shamir-Adleman):

- **Definition**: RSA is an asymmetric (public key) encryption algorithm. It was developed in 1977 by Ron Rivest, Adi Shamir and Leonard Adleman.
- **Principle of Operation**: RSA uses two different keys: a public key and a private key. The public key can be used by anyone and is used to encrypt data. The private key is kept secret and is used to decrypt the encrypted data.
- **Usages**: RSA is often used in areas such as secure data transmission, digital signatures, and certificate management.
- **Security**: RSA works with large prime numbers and therefore provides high security. However, it requires large key sizes, which can slow down processing speed.

What is AES (Advanced Encryption Standard):

- **Definition**: AES is a symmetric encryption algorithm. This means that it uses the same key for both encryption and decryption. AES was adopted and set as a standard by NIST

(National Institute of Standards and Technology) in 2001.
- **Principle of Operation**: AES works with 128-bit, 192-bit or 256-bit long keys. Data is processed in specific block sizes (usually 128-bit) and subjected to a series of complex operations (e.g. scrambling, substitution, rotation and XOR operations).
- **Use Cases**: AES provides fast and efficient encryption of data. Therefore, it is used in many areas such as file encryption, network traffic security and disc encryption.
- **Security**: AES is one of the most secure and robust encryption algorithms available today. It provides a high degree of security that is very difficult to break and offers good performance.

As a result, RSA and AES are two important algorithms used to meet different encryption needs. RSA is used in areas such as key distribution and digital signatures, while AES provides fast and secure data encryption. Therefore, many applications optimise data security and speed by using these two algorithms together.

RSA and AES Encryptions:

- **RSA Encryption:**
    - RSA is a public-key encryption algorithm.
    - It uses two keys: a public key and a private key.
    - The public key is used to encrypt the data and is known to everyone.
    - The private key is used to decrypt the data and is known only to the recipient.
    - The data is encrypted with the recipient's public key and can only be decrypted with the recipient's private key.
- **AES Encryption:**
    - AES is a symmetric encryption algorithm.
    - The same key is used for both encryption and decryption.
    - Data is encrypted with a specific key and decrypted with the same key.
    - The security of the key is critical to the security of the data; therefore, the key must be secured.

Why We Chose RSA and AES?

- **RSA:**
    - Used for key exchange: RSA provides secure key exchange because it is a public-key encryption algorithm. This feature allows keys to be shared securely between the parties in the communication.
    - Reliability: RSA is a reliable and widely used encryption algorithm. It is based on a strong mathematical foundation and is widely accepted.
- **AES:**
    - Fast and efficient: AES is fast and efficient among symmetric encryption algorithms. It encrypts and decrypts data quickly, which ensures that there are no delays in communication.
    - Security: AES offers a strong level of security. When used with proper key management, it is a reliable option for securing data.

For these reasons, we have chosen encryption algorithms such as RSA and AES. RSA offers a secure option for key exchange, while AES provides fast and secure data encryption. When these two algorithms are used together, they form a powerful combination to ensure the security and confidentiality of data in communication.

Uses in Code

- **AES Usage:**
    - In the `encrypt_message` function, a random symmetric key (symmetric_key) and initialisation vector (IV) are generated using the `os.urandom()` function.
    - Then, an encryption object (`cipher`) is created with the AES algorithm using the `Cipher` class. This object is created in AES-CFB mode using symmetric key and initialisation vector.
    - The message is encrypted with the `encryptor` object (`cipher.encryptor()`).
    - The encrypted message and symmetric key are encrypted using RSA with `recipient_public_key`.
- **RSA Usage:**
    - The symmetric key (`symmetric_key`) and the encrypted message are encrypted using RSA with the recipient's public key (`recipient_public_key`).
    - The encrypted key and the encrypted message are bundled together and packaged in JSON format.

So, in this code, the message itself is encrypted with the AES algorithm. However, in order to transmit this encrypted message securely, the symmetric key is encrypted with the RSA algorithm. In this way, both the high speed and efficiency of AES and the secure key sharing features of RSA are utilised.

Steps:

In the following code I will explain in detail the process used to securely transmit messages. The code uses a combination of both RSA and AES encryption algorithms. The purpose of this combination is to ensure that data is transmitted and encrypted securely.

1. **RSA and AES Usage**:
    - The code uses a combination of RSA and AES algorithms. AES is a symmetric encryption algorithm and is used to encrypt the message itself. RSA is an asymmetric algorithm and is used to securely transmit the symmetric key.
2. **RSA Key Generation**:
    - In `get_public_key` function, private and public key pair is generated using RSA algorithm. The function `rsa.generate_private_key` generates a 2048 bit private key (`private_key`). The public key of the private key is obtained with `private_key.public_key()`.

- The variables `private_pem` and `public_pem` contain the private and public keys serialised in PEM format respectively.
- The global variable `server_private_key` is used to store the generated private key.

3. **Message Encryption**:
    - The `encrypt_message` function uses the AES algorithm to encrypt the message.
    - First, a 32 byte random symmetric key (`symmetric_key`) and a 16 byte initialisation vector (`iv`) are generated by `os.urandom()` functions.
    - Using the `Cipher` class, an encryption object (`cipher`) is created in AES-CFB mode.
    - With the `encryptor` object, the message (`message`) is encrypted using the AES algorithm.
4. **Encryption of the Symmetric Key with RSA**:
    - To securely transmit the encrypted message and the symmetric key, the symmetric key is encrypted with the recipient's public key (`recipient_public_key`) using RSA.
    - The function `recipient_public_key.encrypt` is used to encrypt the `symmetric_key`.
5. **Packaging**:
    - The encrypted key, the IV and the encrypted message are combined to form a `package`.
    - The package is made ready for transmission in JSON format and using base64 encoding.
6. **Transmitting the Message as an Encrypted Packet**:
    - The `send_message` function creates an `encrypted_package` to be sent to the recipient.
    - The packet is transmitted through Kafka on a specific topic (`topic`).
7. **Receipt of the message by the recipient**:
    - The `receive_message` function receives the encrypted packet from the Kafka topic (`topic`).
    - The received packet is decrypted by passing it to the `decrypt_message` function.
8. **Decryption of Message**:
    - In the `decrypt_message` function, the received encrypted packet is decrypted.
    - Firstly, the `base64` encoding is decoded to obtain the `encrypted_key`, `iv` and `encrypted_message` data.
    - The encrypted symmetric key is decrypted using RSA with the function `server_private_key.decrypt`.
    - The decrypted symmetric key and IV are used to decrypt the encrypted message by generating `cipher` in AES-CFB mode.

During this process, both the speed and efficiency advantages of AES and the secure key sharing features of RSA are utilised to ensure secure transmission of messages.

---

only_rsa_private_key/main.py

Advantages:

1-Simpler Code Structure: The code structure has been made simpler and easier to understand for encryption and decryption operations using an RSA key pair.

2-Direct Key Retrieval: The get_public_key endpoint makes it easy for users to get a public key directly from the server.

3-Communication Security: RSA encryption secures communication between users and enables private messaging.

4-Easy Startup and Shutdown: startup_event and shutdown_event functions are used to startup and shutdown the Kafka consumer efficiently.

5-Fast Prototyping: FastAPI's quick installation and configuration makes it possible to prototype quickly.

Disadvantages:

1-High Processing Cost: RSA encryption can be costly, especially when processing large data.

2-Key Upload Failure: Malicious users can compromise the security of communication by sending incorrect or corrupted public keys.

3-Heavy Key Management: Continuous generation of new key pairs can complicate the key management process.

4-Poor Error Handling: The code has a simple HTTPException throwing process for errors, which may return generic errors instead of more comprehensive error handling.

5-Limited Scalability: There may be limitations on scalability due to the complexity of the code and potentially long RSA key duration.

---

w_aes_private_key/main.py

Advantages:

1-Hybrid Encryption: The combination of both RSA and AES encryption provides advantages in terms of security and performance.

2-Advanced Security: Combining symmetric and asymmetric encryption adds more layers of security and makes communication more secure.

3-Diverse Encryption Modes: The code provides higher flexibility by using different modes for AES encryption.

4-Automatic Key Management: The code includes operations for key generation and management, which automates and simplifies management.

5-Extensibility: The fact that the code supports more complex key and encryption methods provides an advantage for future extensibility.

Disadvantages:

1-High Resource Usage: Complex encryption and key management operations can increase server resources.

2-More Complex Code Structure: The use of hybrid encryption and key management of code can make the overall code structure and debugging more complex.

3-Security Vulnerabilities: By accepting public keys from the user, the code may allow malicious users to create vulnerabilities by sending fake keys.

4-Loss of Efficiency: When AES and RSA ciphers are used together, they can have negative effects on performance.

5-Complex Error Handling: The complexity of the code can make error handling more difficult and potentially less reliable.

Bibliography:

OpenAI - ChatGPT

https://github.com/burke-software/simple-asymmetric-python/blob/master/simple_asym/asymmetric_encryption.py

https://medium.com/synologyc2/what-is-end-to-end-encryption-7b10bef2ffff

https://medium.com/@hicranozkan/simetrik-ve-asimetrik-anahtarl%C4%B1-%C5%9Fifreleme-algoritmalar%C4%B1-a60a4e0eb079

https://medium.com/@TechTalkWithAlex/cryptography-in-python-a-practical-example-to-code-2899b9bd176c

https://github.com/cheah/adyen-cse-python/tree/master

[TR]

Şifreleme Algoritmaları:

Şifreleme algoritmaları, verilerin güvenli bir şekilde iletilmesini ve saklanmasını sağlayan matematiksel işlemlerdir. İletilen verilerin sadece alıcı tarafından anlaşılmasını sağlarlar ve kötü niyetli kişilerin verilere erişmesini engellerler. İşte en yaygın kullanılan şifreleme algoritmalarından bazıları:

1. **Simetrik Şifreleme Algoritmaları:** Bu algoritmalar, aynı anahtarın hem şifreleme hem de deşifreleme işlemlerinde kullanıldığı bir sistemdir. Veriyi şifrelemek ve deşifrelemek için aynı anahtar kullanılır. Örneğin, AES (Advanced Encryption Standard) ve DES (Data Encryption Standard) gibi algoritmalar simetrik şifreleme algoritmalarına örnektir.
2. **Açık Anahtarlı (Asimetrik) Şifreleme Algoritmaları:** Bu algoritmalar, iki farklı anahtarın kullanıldığı bir sistemdir: biri veriyi şifrelemek için (açık anahtar) ve diğeri deşifrelemek için (özel anahtar). Veriyi göndermeden önce alıcının açık anahtarını kullanarak şifrelenir, ancak sadece alıcının özel anahtarıyla çözülebilir. RSA (Rivest-Shamir-Adleman) bu tür bir algoritmadır.

RSA Nedir? (Rivest-Shamir-Adleman):

- **Tanım**: RSA, bir asimetrik (genel anahtar) şifreleme algoritmasıdır. 1977'de Ron Rivest, Adi Shamir ve Leonard Adleman tarafından geliştirilmiştir.
- **Çalışma Prensibi**: RSA, iki farklı anahtar kullanır: bir genel anahtar ve bir özel anahtar. Genel anahtar, herkes tarafından kullanılabilir ve veri şifrelemek için kullanılır. Özel anahtar ise gizli tutulur ve şifrelenmiş verinin çözülmesi için kullanılır.
- **Kullanım Alanları**: RSA, genellikle güvenli veri iletimi, dijital imzalar ve sertifika yönetimi gibi alanlarda kullanılır.
- **Güvenlik**: RSA, büyük asal sayılarla çalışır ve bu nedenle yüksek güvenlik sağlar. Ancak, büyük anahtar boyutları gerektirir ve bu da işlem hızını yavaşlatabilir.

AES Nedir? (Advanced Encryption Standard):

- **Tanım**: AES, bir simetrik şifreleme algoritmasıdır. Bu, hem şifreleme hem de şifre çözme işlemleri için aynı anahtarı kullandığı anlamına gelir. AES, 2001 yılında NIST (National Institute of Standards and Technology) tarafından kabul edilmiş ve standart olarak belirlenmiştir.
- **Çalışma Prensibi**: AES, 128-bit, 192-bit veya 256-bit uzunluğunda anahtarlarla çalışır. Veri, belirli blok boyutlarında (genellikle 128-bit) işlenir ve bir dizi karmaşık işlemlerden geçirilir (örn. karıştırma, değiştirme, dönme ve XOR işlemleri).
- **Kullanım Alanları**: AES, verilerin hızlı ve verimli bir şekilde şifrelenmesini sağlar. Bu nedenle, dosya şifreleme, ağ trafiği güvenliği ve disk şifreleme gibi birçok alanda kullanılır.
- **Güvenlik**: AES, günümüzün en güvenli ve sağlam şifreleme algoritmalarından biridir. Kırılması çok zor olan yüksek derecede güvenlik sağlar ve iyi bir performans sunar.

Sonuç olarak, RSA ve AES farklı şifreleme ihtiyaçlarını karşılamak için kullanılan iki önemli algoritmadır. RSA, anahtar dağıtımı ve dijital imzalar gibi alanlarda kullanılırken, AES hızlı ve güvenli veri şifrelemesi sağlar. Bu nedenle, birçok uygulama bu iki algoritmayı bir arada kullanarak veri güvenliğini ve hızını optimize eder.

RSA ve AES Şifrelemeleri:

- **RSA Şifrelemesi:**
    - RSA, açık anahtarlı bir şifreleme algoritmasıdır.
    - İki anahtar kullanır: açık anahtar ve özel anahtar.
    - Açık anahtar, veriyi şifrelemek için kullanılır ve herkes tarafından bilinir.
    - Özel anahtar, veriyi deşifrelemek için kullanılır ve sadece alıcının bilir.
    - Veri, alıcının açık anahtarıyla şifrelenir ve sadece alıcının özel anahtarıyla çözülebilir.
- **AES Şifrelemesi:**
    - AES, simetrik bir şifreleme algoritmasıdır.
    - Aynı anahtar hem şifreleme hem de deşifreleme işlemlerinde kullanılır.
    - Veri, belirli bir anahtarla şifrelenir ve aynı anahtarla deşifrelenir.
    - Anahtarın güvenliği, verinin güvenliği için kritiktir; bu nedenle anahtarın güvenliği sağlanmalıdır.

Neden RSA ve AES'i Seçtik?

- **RSA:**
    - Anahtar değişimi için kullanılır: RSA, açık anahtarlı bir şifreleme algoritması olduğu için güvenli anahtar değişimi sağlar. Bu özellik, iletişimdeki taraflar arasında güvenli bir şekilde anahtarların paylaşılmasını sağlar.
    - Güvenilirlik: RSA, güvenilir ve yaygın olarak kullanılan bir şifreleme algoritmasıdır. Güçlü bir matematiksel temele dayanır ve geniş çapta kabul görmüştür.
- **AES:**
    - Hızlı ve etkili: AES, simetrik şifreleme algoritmaları arasında hızlı ve etkilidir. Verileri hızlı bir şekilde şifreler ve deşifreler, bu da iletişimde gecikme olmamasını sağlar.
    - Güvenlik: AES, güçlü bir güvenlik seviyesi sunar. Doğru anahtar yönetimi ile kullanıldığında, verilerin güvenliğini sağlamak için güvenilir bir seçenektir.

Bu nedenlerden dolayı, RSA ve AES gibi şifreleme algoritmalarını tercih ettik. RSA, anahtar değişimi için güvenli bir seçenek sunarken, AES hızlı ve güvenli veri şifreleme sağlar. Bu iki algoritma birlikte kullanıldığında, iletişimdeki verilerin güvenliğini ve gizliliğini sağlamak için güçlü bir kombinasyon oluştururlar.

Kod İerisindeki Kullanımlar

- **AES Kullanımı:**
    - `encrypt_message` fonksiyonunda, `os.urandom()` fonksiyonu kullanılarak rastgele bir simetrik anahtar (symmetric_key) ve başlangıç vektörü (IV) oluşturuluyor.
    - Ardından, `Cipher` sınıfı kullanılarak AES algoritmasıyla bir şifreleme nesnesi (`cipher`) oluşturuluyor. Bu nesne, simetrik anahtar ve başlangıç vektörü kullanılarak AES-CFB modunda oluşturuluyor.
    - `encryptor` nesnesi (`cipher.encryptor()`) ile mesaj şifreleniyor.
    - Şifrelenmiş mesaj ve simetrik anahtar `recipient_public_key` ile RSA kullanılarak şifreleniyor.
- **RSA Kullanımı:**
    - Simetrik anahtar (`symmetric_key`) ve şifreli mesaj, alıcının açık anahtarı (`recipient_public_key`) ile RSA kullanılarak şifreleniyor.
    - Şifrelenmiş anahtar ve şifrelenmiş mesaj bir araya getirilip JSON formatında paketleniyor.

Yani, kodda mesajın kendisi AES algoritması ile şifreleniyor. Ancak bu şifreli mesajın güvenli bir şekilde iletilebilmesi için simetrik anahtar RSA algoritması ile şifreleniyor. Bu sayede, hem AES'in yüksek hız ve verimliliği hem de RSA'nın güvenli anahtar paylaşım özelliklerinden yararlanılmış oluyor.

Adımlar:

Aşağıdaki kodda mesajları güvenli bir şekilde iletmek için kullanılan süreci detaylı olarak açıklayacağım. Kodda hem RSA hem de AES şifreleme algoritmaları bir arada kullanılmıştır. Bu kombinasyonun amacı, verilerin güvenli bir şekilde iletilmesini ve şifrelenmesini sağlamaktır.

1. **RSA ve AES Kullanımı**:
    - Kod, RSA ve AES algoritmalarının kombinasyonunu kullanmaktadır. AES, simetrik şifreleme algoritmasıdır ve mesajın kendisini şifrelemek için kullanılır. RSA ise asimetrik bir algoritmadır ve simetrik anahtarın güvenli bir şekilde iletilmesi için kullanılır.
2. **RSA Anahtar Üretimi**:
    - `get_public_key` fonksiyonunda, RSA algoritması kullanılarak özel ve genel anahtar çifti oluşturuluyor. `rsa.generate_private_key` fonksiyonu ile 2048 bitlik bir özel anahtar (`private_key`) oluşturuluyor. Özel anahtarın genel anahtarı `private_key.public_key()` ile elde ediliyor.
    - `private_pem` ve `public_pem` değişkenleri, sırasıyla özel ve genel anahtarların PEM formatında seri hale getirilmiş hallerini içeriyor.

- `server_private_key` global değişkeni, oluşturulan özel anahtarı saklamak için kullanılıyor.

3. **Mesaj Şifreleme**:
    - `encrypt_message` fonksiyonunda, mesajı şifrelemek için AES algoritması kullanılıyor.
    - İlk olarak, `os.urandom()` fonksiyonları ile 32 byte'lık rastgele bir simetrik anahtar (`symmetric_key`) ve 16 byte'lık bir başlangıç vektörü (`iv`) oluşturuluyor.
    - `Cipher` sınıfı kullanılarak AES-CFB modunda bir şifreleme nesnesi (`cipher`) oluşturuluyor.
    - `encryptor` nesnesi ile mesaj (`message`) AES algoritması kullanılarak şifreleniyor.
4. **Simetrik Anahtarın RSA ile Şifrelenmesi**:
    - Şifrelenmiş mesajı ve simetrik anahtarı güvenli bir şekilde iletmek için, simetrik anahtar alıcının genel anahtarı (`recipient_public_key`) ile RSA kullanılarak şifreleniyor.
    - `recipient_public_key.encrypt` fonksiyonu, `symmetric_key`'yi şifrelemek için kullanılıyor.
5. **Paketleme**:
    - Şifrelenmiş anahtar, IV ve şifrelenmiş mesaj bir araya getirilerek bir paket (`package`) oluşturuluyor.
    - Paket, JSON formatında ve base64 kodlaması kullanılarak iletilmeye hazır hale getiriliyor.
6. **Mesajın Şifreli Paket Olarak İletilmesi**:
    - `send_message` fonksiyonunda, alıcıya gönderilmek üzere şifreli paket (`encrypted_package`) oluşturuluyor.
    - Paket, Kafka aracılığıyla belirli bir konu (`topic`) üzerinde iletiliyor.
7. **Mesajın Alıcı Tarafından Alınması**:
    - `receive_message` fonksiyonunda, Kafka konusundan (`topic`) şifreli paket alınıyor.
    - Alınan paket, `decrypt_message` fonksiyonuna iletilerek çözümleniyor.
8. **Mesajın Şifresinin Çözülmesi**:
    - `decrypt_message` fonksiyonunda, alınan şifreli paket çözümleniyor.
    - İlk olarak, `base64` kodlaması çözülerek `encrypted_key`, `iv` ve `encrypted_message` verileri elde ediliyor.
    - Şifrelenmiş simetrik anahtar, `server_private_key.decrypt` fonksiyonu ile RSA kullanılarak çözülüyor.
    - Çözülen simetrik anahtar ve IV, AES-CFB modunda `cipher` oluşturularak şifrelenmiş mesajın şifresi çözülüyor.

Bu süreç boyunca, hem AES'in hız ve verimlilik avantajlarından hem de RSA'nın güvenli anahtar paylaşım özelliklerinden yararlanılarak mesajların güvenli bir şekilde iletilmesi sağlanıyor.

---

only_rsa_private_key/main.py

Avantajlar:

1-Daha Basit Kod Yapısı: Kod yapısı, bir RSA anahtar çifti kullanarak şifreleme ve deşifreleme işlemleri için daha basit ve kolay anlaşılır hale getirilmiş.

2-Doğrudan Anahtar Alımı: get_public_key endpoint'i, kullanıcıların doğrudan sunucudan açık anahtar almasını kolaylaştırır.

3-İletişim Güvenliği: RSA şifrelemesi, kullanıcılar arasındaki iletişimi güvenli hale getirir ve özel mesajlaşmayı sağlar.

4-Kolay Başlatma ve Kapatma: startup_event ve shutdown_event fonksiyonları ile Kafka tüketicisinin başlatılması ve kapatılması etkin bir şekilde yapılır.

5-Hızlı Prototipleme: FastAPI'nin hızlı kurulum ve yapılandırması, hızlı bir şekilde prototip oluşturmayı mümkün kılar.

Dezavantajlar:

1-Yüksek İşlem Maliyeti: RSA şifrelemesi, özellikle büyük veriler üzerinde işlem yaparken maliyetli olabilir.

2-Anahtar Yükleme Hatası: Kötü niyetli kullanıcılar, hatalı veya bozuk açık anahtar göndererek iletişimin güvenliğini tehlikeye atabilir.

3-Ağır Anahtar Yönetimi: Sürekli yeni anahtar çiftleri oluşturulması, anahtar yönetimi sürecini karmaşık hale getirebilir.

4-Zayıf Hata İşleme: Kodda hatalar için basit bir HTTPException atma işlemi var, bu daha kapsamlı hata işleme yerine genel hataları döndürebilir.

5-Sınırlı Ölçeklenebilirlik: Kodun karmaşıklığı ve potansiyel olarak uzun RSA anahtar süresi nedeniyle ölçeklenebilirlik konusunda sınırlamalar olabilir.

---

w_aes_private_key/main.py

Avantajlar:

1-Hibrid Şifreleme: Hem RSA hem de AES şifrelemesinin birleşimi, güvenlik ve performans açısından avantaj sağlar.

2-Gelişmiş Güvenlik: Symmetric ve asymmetric şifrelemeyi birleştirmek, daha fazla güvenlik katmanı ekler ve iletişimi daha güvenli hale getirir.

3-Çeşitli Şifreleme Modları: Kod, AES şifrelemesi için farklı modlar kullanarak daha yüksek esneklik sağlar.

4-Otomatik Anahtar Yönetimi: Kodda anahtar oluşturma ve yönetimi için işlemler yer almakta, bu da yönetimi otomatikleştirerek basitleştirir.

5-Genişletilebilirlik: Kodun daha karmaşık yapıdaki anahtar ve şifreleme yöntemlerini desteklemesi, gelecekteki genişletilebilirlik için avantaj sağlar.

Dezavantajlar:

1-Yüksek Kaynak Kullanımı: Karmaşık şifreleme ve anahtar yönetimi işlemleri, sunucu kaynaklarını artırabilir.

2-Daha Karmaşık Kod Yapısı: Kodun hibrid şifreleme kullanımı ve anahtar yönetimi, genel kod yapısını ve hata ayıklamayı daha karmaşık hale getirebilir.

3-Güvenlik Açıkları: Kod, kullanıcıdan gelen açık anahtarları kabul ederek, kötü niyetli kullanıcıların sahte anahtarlar göndererek güvenlik açığı yaratmasına izin verebilir.

4-Verimlilik Kaybı: AES ve RSA şifrelemeleri birlikte kullanıldığında, performans üzerinde olumsuz etkileri olabilir.

5-Karmaşık Hata İşleme: Kodun karmaşıklığı, hata işleme sürecini daha zor ve potansiyel olarak daha az güvenilir hale getirebilir.

Kaynakça:

OpenAI - ChatGPT

https://github.com/burke-software/simple-asymmetric-python/blob/master/simple_asym/asymmetric_encryption.py

https://medium.com/synologyc2/what-is-end-to-end-encryption-7b10bef2ffff

https://medium.com/@hicranozkan/simetrik-ve-asimetrik-anahtarl%C4%B1-%C5%9Fifreleme-algoritmalar%C4%B1-a60a4e0eb079

https://medium.com/@TechTalkWithAlex/cryptography-in-python-a-practical-example-to-code-2899b9bd176c

https://github.com/cheah/adyen-cse-python/tree/master

FastAPI docs --> http://0.0.0.0:8000/docs