Implementation of Public-Key Cryptography for Securing API Data Transmission over HTTP and Its Analysis Using Wireshark

Brandon Theodore Ferrinov - 18223020
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: brandonferrinov@gmail.com , 18223020@std.stei.itb.ac.id

***Abstract*—Application Programming Interface (API) has become a commonly used component of modern software development, because it can enable communication between different applications and services. However, API communication using the Hypertext Transfer Protocol (HTTP) is vulnerable to cyber attack such as eavesdropping or man-in-the-middle attack because the data is being transmitted in plaintext. This research investigates the implementation of Public-Key Cryptography (PKC) to improve confidentiality of API data that is being transmitted via HTTP. The proposed system uses the RSA algorithm to encrypt the API's payload. An experimental approach is used by comparing both scenarios. Wireshark is used to capture and analyze network traffic in both scenarios. The results show that plaintext payloads are fully visible whereas encrypted payloads are unreadable without access to the private key. Although the implementation of PKC increases the payload size from 134 to 425 bytes, the positive values still outweigh the negative ones because it still greatly improves confidentiality.**

Keywords—API Security, Public-Key Cryptography, RSA, HTTP, Wireshark, Network Security, Data Confidentiality.

I. Introduction

Application Programming Interface (API) has been commonly used in modern software development, API has become a fundamental when developing a system. API enables communication between different applications, services, and devices by exchanging data using commonly used protocols such as HTTP/HTTPS. The increasing use of web services, cloud computing, mobile applications, and distributed systems has significantly increased the usage of secure API communication.

One of the most common protocols to be used for API communication is HTTP or Hypertext Transfer Protocol. HTTP provides a resource-efficient mechanism for transmitting data between client and server although it has its downsides for not being able to provide confidentiality and protection against some cyber attacks such as eavesdropping or man-in-the-middle attacks. Data that is sent through HTTP is being transmitted in plaintext, this scenario allows attackers who can intercept network traffic to see the content of the message that's being transmitted. Sensitive information such as usernames, email addresses, password, and personal data may be exposed to unauthorized parties.

To solve this problem, cryptography implementations are commonly used to protect data that's being transmitted through the network. Public-Key Cryptography (PKC) is one of cryptography approaches that use a pair of keys (public and private key). Public-Key Cryptography works by encrypting data using a public key and decrypting it using the private key. Using this method, confidential information can be transmitted securely even though insecure communication channels are being used.

In real life, secure communication is usually achieved by using HTTPS or Hypertext Transfer Protocol Secure which use the Transport Layer Security (TLS) protocol. However, using TLS may obscure the uses and benefits of using cryptography techniques for securing data confidentiality. Therefore, this research is evaluating the use of Public-Key Cryptography without using the Transport Layer Security (TLS) mechanism to provide a clearer understanding of the effectiveness of using Cryptography, especially Public-Key Cryptography in protecting transmitted data.

This research implements the Public-Key Cryptography at the application layer to secure API data transmission while using the HTTP protocol. By comparing the data that's transmitted while being encrypted and not using wireshark, this research aims to demonstrate the importance of cryptographic protection on data confidentiality and analyze how encrypted data appears within network traffic that is captured using wireshark.

The objective of this research is to design and implement a Public-Key Cryptography mechanism for securing API data transmission over HTTP, compare plaintext and encrypted API communication using wireshark, and analyze the visibility of encrypted and unencrypted data within the network packet captures. With that, this research is expected to increase the awareness regarding the risks associated with transmitting sensitive data through unencrypted HTTP channels.

II. Theoretical Framework

A. Application Programming Interface (API)

Application Programming Interface (API) is a set of rules and protocols that allows one application to communicate with another application. API acts as an intermediary that connects an application with another application so it can communicate and transmit data safely and securely. API also enables interoperability between different systems such as web application, mobile application, cloud service, and distributed system environment.

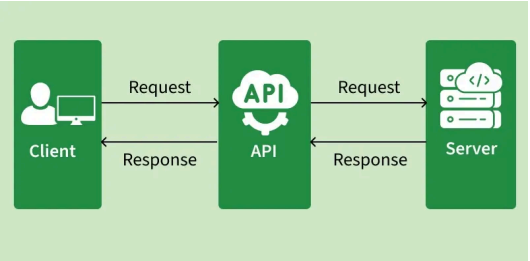

Figure 1. Example of how API work in software with client-server architecture

(Source: GeeksForGeeks)

In client-server architecture, API commonly uses HTTP requests and responses to transmit information. Data is frequently presented using JavaScript Object Notation (JSON) due to its lightweight structure and ease of use.

B. Hypertext Transfer Protocol (HTTP)

Hypertext Transfer Protocol (HTTP) is an application-layer protocol that clients can use to communicate with servers on the internet. HTTP use a request-response model in which a client sends a request and the server will return a response.

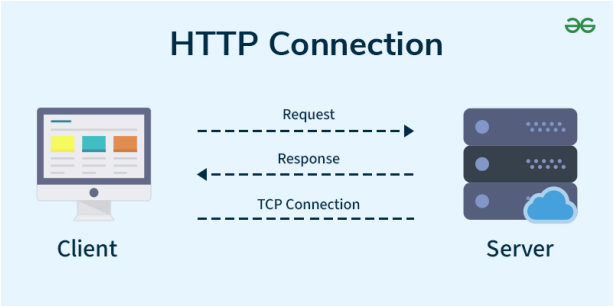

Figure 2. How HTTP works

(Source: GeeksForGeeks)

Although HTTP is simple and efficient, it does not provide encryption and confidentiality. Information transmitted via HTTP can be seen and inspected by some parties capable of monitoring network traffic. This can lead to sensitive data transmitted via HTTP to be exposed to unauthorized parties.

HTTP has a few methods including GET, POST, PUT, and DELETE. In this research the POST method is used to transmit API data from client to server.

C. Information Security Principles

Information Security is a field of expertise that works to protect data and systems against unauthorized access, modification, or deletion. There are three fundamental principles of Information Security commonly known as the CIA Triad:

- **Confidentiality**: This principle ensures that information is only accessible to authorized parties.
- **Integrity**: This principle ensures that data remains accurate and unaltered during transmission or storage.
- **Availability**: This principle ensures that information and services remain accessible when required.

This research primarily focuses on confidentiality through the application of cryptography protection.

D. Cryptography

Cryptography is a field of expertise that use mathematical calculations to secure information by transforming readable data (plaintext) into an unreadable form (ciphertext), this process is also known as encryption whereas the reverse process is known as decryption.

Cryptography systems generally consist of:

- Plaintext
- Ciphertext
- Encryption Algorithm
- Decryption Algorithm
- Cryptographic Key

the primary objective of cryptography is to protect information from unauthorized disclosure.

E. Public-Key Cryptography

Public-Key cryptography also known as asymmetric cryptography uses two separate keys; public key and private key. The public key can be distributed openly and is used for encryption while the private key is being kept as a secret and is used for decryption.

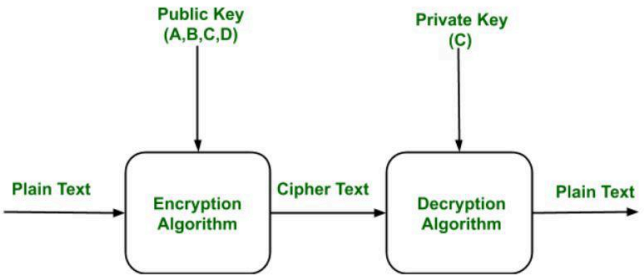

Figure 3. How Public-Key cryptography works

(Source: GeeksForGeeks)

Public key cryptography communication process usually consists of a few steps, such as:

- Sender obtains the recipient's public key
- Sender encrypts the plaintext using the public key
- Encrypted message is transmitted through the network
- The recipient decrypts the cipher text using private key.

This method enables secure communication without requiring a pre-shared secret key.

F. RSA Algorithm

RSA is one of the most widely used Public-Key cryptography algorithms. RSA security is based on the computational difficulty of factoring large integers. The RSA process consists of key generation, encryption, and decryption.

RSA provides confidentiality by ensuring that only the holder of the private key is able to decrypt and recover the original plaintext. In this research, RSA is used to encrypt API payload data before transmission through HTTP.

G. Base64 Encoding

Base64 is a binary-to-text encoding scheme that converts binary data into ASCII characters. RSA encryption produces binary output, so the ciphertext must first be encoded into Base64 before being included within JSON payloads transmitted through HTTP requests. It is also important to note that Base64 does not provide security and servers only as a data representation mechanism.

H. Wireshark

Wireshark is an open-source network protocol analyzer that can be used to capture, inspect, and analyze network traffic. It allows people to observe communication that is occurring between devices and identify security issues within the network protocols.

Wireshark provides capabilities such as packet capture, protocol analysis, traffic filtering, payload inspection, and network troubleshooting. In this research, Wireshark is used to compare contents of a packet before and after cryptographic protection is applied.

I. Research Framework

In this research, the framework consists of a few stages from the development of the API communication protocol using HTTP-based protocol until the packet capture and analysis using Wireshark. The hypothesis of this research is that Public-Key Cryptography significantly improves confidentiality by preventing malicious parties from inspecting API payload contents, even when the communication uses HTTP protocol.

III. Research Methodology

A. Research Method

This study uses an experimental research method to evaluate the effectiveness of Public-Key Cryptography in securing API data transmission when using HTTP protocol. The experiment compares two scenarios, the first one being plaintext HTTP communication and the second is encrypted HTTP communication using RSA Public-Key Cryptography.

The objective of this experiment is to determine whether the implementation of Public-Key Cryptography can protect API payload confidentiality while still using the HTTP protocol and the same network environment. Network traffic that is generated during both scenarios is being captured and analyzed using Wireshark.

B. Research Design

In this research, there are two experimental scenarios that are being used:

Scenario A: Plaintext Communication

In this scenario, API data is being transmitted via HTTP without any cryptographic protection. The data that is being transmitted is using the JSON format and can be directly inspected using Wireshark.

Scenario B: Encrypted Communication

In this scenario, the API payload is being encrypted using RSA Public-Key Cryptography before the transmission. The encrypted text is then encoded using Base64 and was sent via the same HTTP communication channel that is being used on scenario A. The server will decrypt the received message using the correct private key.

The comparison between scenario A and scenario B allows the impact of cryptography protection on data confidentiality to be analyzed and evaluated.

C. System Architecture

The system used in this research can be further divided into three main components that are Client Application, API Server, and Network Monitoring Environment. The communication architecture for plaintext communication is illustrated as follows:

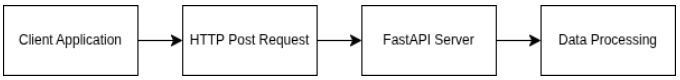

Figure 4. Plaintext communication architecture

Simultaneously, Wireshark captures network traffic to analyze the transmitted packet.

For the encrypted communication scenario, the architecture is illustrated as follows:

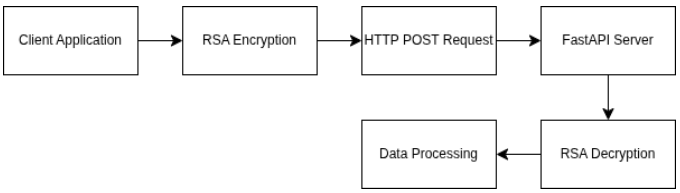

Figure 5. Encrypted communication architecture

D. Public-Key Cryptography Implementation

The implementation of Public-Key Cryptography follows the general way, first an RSA key pair is generated prior to communication, the generated key pair consist of public key and private key. The public key is distributed to the client application and user for encryption whereas the private key remains stored securely on the server and is used for decryption. The encryption and decryption process will be explained as follows:

Encryption Process

(1) Create API payload data in JSON format
(2) Convert JSON data into plaintext

(3) Encrypt plaintext using server's public key
(4) Encode the cipher text using Base64
(5) Transmit the cipher text via HTTP POST request

Decryption Process

(1) Extract cipher text from the HTTP request
(2) Decode the Base64
(3) Decrypt cipher text using the RSA private key
(4) Recover the original plaintext JSON
(5) Process the received information

E. Experimental Procedures

This research is conducted by follow a few steps as follows:

Environment Preparation

(1) Install required software and libraries
(2) Configure the code for FastAPI server
(3) Configure the code for client application
(4) Install and configure Wireshark

Plaintext Communication Test

(1) Start packet capture using Wireshark
(2) Send plaintext API requests from client to server
(3) Capture the network traffic
(4) Inspect the packet that is being captured to be analyzed
(5) Observe payload visibility with Wireshark

Encrypted Communication Test

(1) Generate RSA public and private keys
(2) Use the code for encrypted payload on client
(3) Start packet capture using Wireshark
(4) Send encrypted payload API requests via HTTP
(5) Capture the network traffic
(6) Inspect the packet that is being captured to be analyzed
(7) Observe the ciphertext visibility with Wireshark

Comparative Analysis

(1) Compare packet contents from both scenario
(2) Evaluate payload readability
(3) Analyze the difference in the packet size
(4) Assess Confidentiality improvements

F. Data Collection Method

In this research, the data is collected through direct observation of network traffic in Wireshark, the collected data include:

(1) Packet captures from Wireshark
(2) API request payload
(3) Payload size

G. Evaluation Metrics

The effectiveness of this research is evaluated using two metrics. First one is the payload readability, this metrics measures whether the transmitted information can be directly interpreted from the payload. There are 3 criteria that will be used:

- Readable
- Partially Readable
- Unreadable

The second metric is the payload size, this metric will measure the increase in payload size when and when not using cryptographic encryption.

$$\text{Payload Increase (\%)} = \left(\frac{\textit{Encrypted Payload Size} - \textit{Original Payload Size}}{\textit{Original Payload Size}}\right) \times 100\% \quad (1)$$

IV. Implementation, Results and Discussion

A. System Implementation

This research implements client-server architecture to evaluate the effectiveness of Public-Key Cryptography in protecting API payload during transmission via HTTP communication. The system consists of two main components that are Client Application and FastAPI Server. The client application generates API requests and transmits the data to the server via HTTP POST requests. The server then receives the requests and processes the transmitted data.

There are two communication scenarios that are being used that are plaintext API communication and Encrypted API communication. The first scenario transmits the data directly via HTTP without any encryption whereas the second scenario encrypts the payload using RSA public key cryptography before being transmitted.

In the first scenario, the client transmits JSON data directly to the API endpoint without any encryption. The payload that will be transmitted is shown as follows:

```
data = {
    "username": "TheoKaitou",
    "password": "secret123",
    "role": "admin"
}
```

In the encrypted scenario, the client performs a few operations as follows:

(1) Converts JSON data into plaintext
(2) Encrypt the plaintext using server's RSA public key
(3) Encode the ciphertext using Base64
(4) Sends the ciphertext as an HTTP POST request to the server

after receiving the request, the server will do as follows:

(1) Decode the Base64 ciphertext
(2) Decrypt the ciphertext using the RSA private key
(3) Recover the original JSON payload

B. Experimental Setup

The experiment was conducted using the following environment

TABLE 1.

Component Specification

Component	Specification
Operating System	Fedora Linux 42
Programming Language	Python 3.13.13
API Framework	FastAPI
Web Server	Uvicorn
Cryptography Library	Python Cryptography
Network Analyzer	Wireshark
Communication Protocol	HTTP

C. Plaintext Communication Result

The first experiment that will be evaluated is the plaintext communication. In this experiment, the client transmitted the JSON payload directly via HTTP without any encryption. Below is the image of the captured HTTP packet:

```
POST /api/plain HTTP/1.1
Host: localhost:8000
User-Agent: python-requests/2.34.2
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Length: 68
Content-Type: application/json

{"username": "TheoKaitou", "password": "secret123", "role": "admin"}
```

Figure 6. Captured plaintext's HTTP packet

The packet inspection revealed that the entire JSON payload is visible in the captured traffic. These results demonstrate that any party capable of monitoring the network traffic can directly access sensitive data/information that is being transmitted via the HTTP channel.

D. Encrypted Communication Results

The second experiment that will be evaluated is the encrypted communication using RSA Public Key. The same JSON payload is being used in this experiment. Below is the image of the captured HTTP packet:

Figure 7. Captured encrypted HTTP packet

As can be seen in the image above, the original JSON content was no longer visible in the Wireshark. This shows, although the HTTP protocol remained unchanged, the transmitted information became unreadable without possession of the corresponding RSA private key. However the server successfully decrypted the received ciphertext and recovered the original payload as can be seen below:

Figure 8. Decrypted ciphertext

E. Comparative Analysis

To compare the results between both communication scenarios, see table below:

TABLE 2.

Comparison Between Plaintext and Encrypted Communication

Metric	Plaintext	Encrypted
Protocol	HTTP	HTTP
Readability	Readable	Not Readable
Confidentiality	Low	High
Decryption Required	No	Yes
Size	134 Bytes	425 Bytes

after being compared in the table above, now the payload increase can also be calculated using the formula given before.

$$\text{Payload Increase (\%)} = \left(\frac{425 - 134}{134}\right) \times 100\% = 217,16\%$$

The results indicate that Public-Key Cryptography significantly improves confidentiality while using the same communication protocol. Although the payload increase is very big at 217% however this can happen because the payload that's being used is relatively small, if the payload is bigger, the payload increase will be lower.

F. Discussion

The result of this experiment demonstrates that Public-Key Cryptography can effectively protect API payloads that are being transmitted via HTTP communication.

During the plaintext communication scenario, sensitive information was directly observable within the captured

network traffic in Wireshark. In contrast, the encrypted communication scenario prevents malicious parties that can observe the network from interpreting the transmitted data despite using the same HTTP protocol.

This finding supports the hypothesis before that confidentiality can be improved through the use of application-layer cryptographic protection even when the transport-layer default encryption such as HTTPS are not being used.

However, the implementation of Public-Key Cryptography needs more resources to use, this resource consuming process is related to the encryption and decryption process. Additionally, ciphertext payloads use more space that plaintext messages because of RSA encryption and Base64 encoding.

Despite these limitations, the proposed approach of using Public-Key Cryptography successfully protects sensitive API data from network monitoring and demonstrates the effectiveness of Public-Key Cryptography in securing HTTP-based communications.

V. Conclusion

This research successfully designed and implemented a Public-Key Cryptography mechanism to secure API data transmission via Hypertext Transfer Protocol (HTTP). The implementation uses the RSA algorithm to encrypt API payloads before being transmitted and decrypted by the server. The encryption process was integrated at the application layer, allowing data confidentiality to be achieved without modifying the communication protocol that's being used.

The experimental results demonstrate a clear difference between plaintext and encrypted communication. In the plaintext scenario, API payload contents were directly visible in Wireshark packet captures, indicating that sensitive information that is being transmitted via HTTP can be easily seen by unauthorized parties. In contrast, the encrypted communication scenario uses RSA Public-Key Cryptography that produced ciphertext that was unreadable when captured by Wireshark. Although the packet still can be accessed because HTTP was being used, the original information could not be interpreted without the possession of the RSA private key.

The comparative analysis showed that the implementation of Public Key Cryptography significantly improved confidentiality while still maintaining the same communication protocol, HTTP. The encrypted payload increased in size from 135 bytes to 425 bytes, resulting in approximately 217% overhead, although the size is increasing, the confidentiality benefits provided by the encryption outweigh the additional costs for sensitive data.

In the end, it can be concluded that Public-Key Cryptography is an effective solution for protecting API payloads that is being transmitted through HTTP protocols. The results confirm that application-layer encryption can reduce the risk of information leakage caused by network traffic interception by malicious parties. Future work may investigate the use of hybrid scenario such as RSA-AES combinations, evaluate the performance by using a larger datasets, or compare this approach with HTTPS/TLS-based communication protocols.

References

[1] R. T. Fielding and J. Reschke, *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*, RFC 7230, Internet Engineering Task Force (IETF), Jun. 2014.

[2] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 8th ed. Hoboken, NJ, USA: Pearson, 2023.

[3] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, 1996.

[4] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[5] S. Josefsson, *The Base16, Base32, and Base64 Data Encodings*, RFC 4648, Internet Engineering Task Force (IETF), Oct. 2006.

[6] G. Combs, *Wireshark User's Guide*, Wireshark Foundation. [Online]. Available: https://www.wireshark.org/docs/

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026

Brandon Theodore Ferrinov - 18223020