An end-to-end encrypted application

1 Introduction

Many people may have noticed that as long as you chat about any product on whassup, Instagram, or other social software, soon Amazon, eBay, etc. all web portals will push you relevant information as if negotiated. We have reason to believe that these instant messages are spying on our every word. A better approach is to use end-to-end encryption to prevent middlemen from intercepting our chats. Most products implement an encrypted channel from the service to the client. Even if you're browsing the web, most websites already support HTTPS, and you'll see a lock-shaped security sign near the address bar. Unfortunately, in this case, it can only be guaranteed that there is no middleman between the client and the service provider, but the service provider is essentially a "middleman" for both clients, and we still want to be sensitive to some Information Confidentiality. This requires end-to-end encryption, denying the service provider the true content of the information. Therefore, we intend to design an end-to-end conversational application and improve it to implement encrypted conversations and other functions.

2 Main components

(1) Asymmetric encryption

(2) RSA algorithm

(3) Firewall system

(4) Data sniffing system

3 System Overview

(1) Encryption

There are exactly two categories of popular encryption schemes: symmetric encryption and asymmetric encryption. Since symmetric encryption needs to agree on a password with the other party in advance, this is difficult in many cases, because how to transmit the password is another problem. The encrypted transmission method is as follows.

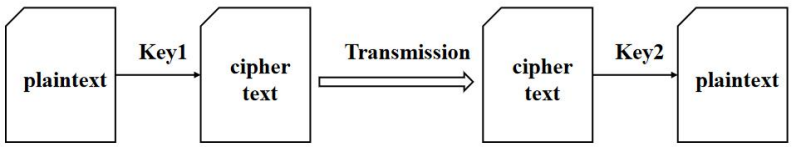

Figure 1: encrypted transmission method

We can publicly transmit it to the left side Key1, and then the left side calculates the ciphertext and transmits it to the right side, and the right side decrypts it with Key2.

Due to the characteristics of asymmetric encryption, all the information encrypted by Key1 can only be decrypted by Key2. We only need to ensure that Key2 is not leaked. No matter if Key1 or ciphertext is leaked, it will not cause secret leakage. Although Key1 can calculate ciphertext, it cannot be decrypted and can only be decrypted by the corresponding Key2. This is a very interesting feature of asymmetric cryptosystems.

(2) Firewall

Firewalls isolate risk areas from safe areas but do not block people's access to risk areas. In general, the firewall we designed has the following functions:

- Restrict unauthorized users from accessing the server, and filter out unsafe services and illegal users.
- Prevent intruders from approaching the defense facilities of the internal network, and detect and alarm network attacks.

- Restrict internal users from accessing special sites.
- Record the content and activities of information passing through the firewall.

(3) Data sniffing system

We designed a data sniffer that monitors data status, flow, and transmission. If the target content is detected, the feedback output is performed.

(4) System implementation

We design such a system, which has the following functions:

- Generate a pair of asymmetric keys;
- The ciphertext can be obtained by public-key encryption;
- The plaintext can be obtained by decrypting the private key.

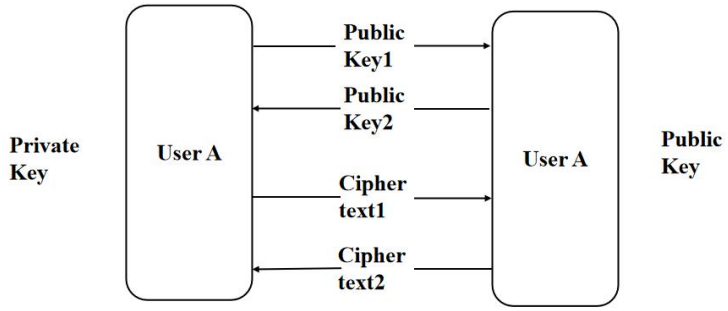

Figure 2: system structure

Before the communication starts, A and B exchange each other's public keys and each retains its private key. Then use the other party's public key to encrypt the information and send it. The received ciphertext is decrypted with its private key.

4 Methods

(1) Define the TargetServer class to allow concurrent connections of TCP. In this class, the client's request is received and verified, if the user meets the firewall policy, then return a 404 and alarm. If it passes, the socket communication connection is made.

(2) Design the client, instantiate the socket object and connect with the server.

(3) Generate key pair. First, create a key pair with a key length of 512. If you print them out, you will find that they are a set of numbers.

(4) Encrypt and decrypt. After completing the key creation, send the public key in pem format to the other party and request the other party's public key. The public key transmitted in the chat tool is in pem format, a type of text data. We already know that the key is a series of numbers, so the public key parameters need to be recovered from the public key in pem format.

(5) Data sniffing. The server-side sets the sniffing keyword, and if the data sent by the client contains the keyword, the data will be displayed.

(6) UI design. Added UI for chat encryption tool for easy use. In this event, two functions are provided: encryption and decryption.

5 Reference

[1] Zhao, Ran, Q., Yuan, L., Chi, Y., & Ma, J. (2015). Key Distribution and Changing Key Cryptosystem Based on Phase Retrieval Algorithm and RSA Public-Key Algorithm. Mathematical Problems in Engineering, 2015, 1 – 12. https://doi.org/10.1155/2015/732609

[2] Reddy. (2022). RM- RSA algorithm. Journal of Discrete Mathematical Sciences & Cryptography, 25(1), 1 – 13. https://doi.org/10.1080/09720529.2020.1734292