International Journal of Trend in Research and Development, Volume 8(4), ISSN: 2394-9333
www.ijtrd.com

A Secure Instant Messager

Bing Lv
College of Computer Science and Technology, Shan Dong University of Technology, Zibo, Shandong, China

Abstract: This system is based on E2EE (End-to-End Encryption) technology and implements a secure instant messager. Including instant messager, file sharing and video call functions. E2EE means that in a communication system, only the end user can decrypt messages, preventing Internet service providers, including but not limited to, even the communication system providers from obtaining the key used to decrypt the content of the communication. At present, even if the typical communication system does not include end-to-end encryption, this type of system can only ensure that the communication between the client and the server is protected, and the service provider itself can see and store all the user's communication content. End-to-End encryption is considered more secure because it attempts to fundamentally avoid this problem.

Keywords: *E2EE; Instant Messager; Key; Secure*

I. INTRODUCTION

This system hopes to have a deep understanding of the technology of an instant messaging system based on end-to-end encryption, explore better authentication mechanisms, promote the increase of the amount of data on the Internet that cannot be decrypted by service operators, support and promote the progress of end-to-end encryption technology and Standardization work.

This system is divided into three main parts. API server, message server and client. The API server is used to process the business requirements of the system. The message server responsible for user session management and message forwarding is designed with high performance and can be deployed in multiple data centers to cope with the connection scenarios of a large number of users.

II. SYSTEM GOAL

1. Instant messaging

This system needs to complete the common functions of the instant messaging system, including instant messaging, file sharing, and video call functions.

2. Performance and flexibility

This system provides a high-performance messaging server that can be distributed to multiple data centers to carry a large number of users.

3. Identity verification

This system is designed to provide only verified communications. During the conversation, both parties can determine the identity of the other party. To avoid man-in-the-middle attacks.

4. End-to-End encryption

This system establishes an end-to-end encryption link. The shared key is deterministically derived using a variant of the Diffie-Hellman key exchange algorithm, and the session key is never directly transmitted on the network. This will prevent any third party (including the service provider itself) from viewing or tampering with the user's communication content, because it is difficult for a third party without a key to decrypt the data transmitted or stored in the system.

5. Privacy

When this system establishes a link, it will avoid leaking the keys of any parties involved to a third party.

6. Cross-platform and audit

This system provides cross-platform end-user components and allows users to audit the security of the system.

Anonymous communication is not within the scope of this system.

III. FEASIBILITY ANALYSIS

1. Symmetric key encryption

The client already has some kind of symmetric key that can be used to encrypt data before sending it and decrypt it when it is received. This system uses AEAD (Authenticated Encryption with Associated Data) symmetric encryption. The AEAD algorithm has the authentication and encryption of the associated data, which is an encryption mode that can guarantee the confidentiality, integrity and authenticity of the data at the same time. This algorithm has been used by industrial-grade encrypted transmission protocols such as TLS and SSH, and has been widely supported. Compared with algorithms such as AES-GCM, it requires lower equipment performance and higher speed.

2. Key Exchange

This system uses the ECDH (Elliptic Curves Diffie-Hellman) algorithm, which is a variant of Diffie-Hellman key exchange, and uses elliptic curve cryptography to enhance performance and security. Compared with RSA, this algorithm is more efficient, has a shorter key, and is less susceptible to side-channel attacks (including cache timing attacks, hyper-threading attacks, etc.), and is widely supported by the industry.

3. Identity verification/identity key management

This is the core part of the entire system. This system uses PKI (Public Key Infrastructure) and submits the identity verification to CA (Certificate Authority) to complete. The PKI system is currently a widely used and highly secure identity authentication method.

IV. SYSTEM DESIGN

The system as a whole is divided into three parts: API server, message server, and client App.

The API server is used to process user requests and implement business logic. Use MongoDB replica set to store data. Use Nginx reverse proxy for load balancing.

The message server is responsible for user session management and message forwarding. The message server is a high-performance WebSocke server developed using C++ and Oat++ frameworks, and is allowed to be deployed to multiple data centers to cope with the connection scenarios of a large number of users.

The client, as the software provided to the end user, calls the API provided by the API service to complete various services, and uses two sets of WebSocket connections (in this system they are called Replica and Contact) to send and receive instant messages with the message server. Provide users with end-to-end encrypted instant messaging and file sharing, as well as video call functions.

In addition, a streaming media server needs to be deployed in the data center where the message server is located to meet the needs of video calls. It is also necessary to deploy a set of object storage services based on the S3 (S3 Simple Storage Service) protocol to meet users' image and file sharing needs.

1. Message Server

The message server consists of the following 3 parts:

Controller layer,used to provide WebSocket connection establishment and implementation of REST API.

Session service,used to implement session management, keep-alive, online and offline, and identity verification. Message forwarding, including forwarding instant messages, returning messages, offline notifications, and time call signaling.

Statistics and log service,used to record statistical information and system logs for administrators to check and monitor the running status of the system in real time

2. Client App

The core part of the client is the key exchange engine, which is divided into two parts, crypto_lowlevel and HLC. The former implements basic cryptographic operation primitives, and the latter implements an abstraction layer that encapsulates the process of message encryption and decryption by key exchange.

Crypto_lowlevel contains the abstraction of the Curve25519 key, the realization of the X25519 key exchange and the encapsulation of Chacha20Poly1305. Each session will have its own HLC. HLC internally completes key exchange, encryption and decryption, packaging of messages and metadata, etc., and provides abstraction for other parts.

3. Asynchronous IO Framework

The client encapsulates an asynchronous task framework. This is because the compatibility of PyGObject and Python's native asynchronous framework asyncio is poor, so the asynchronous IO framework is implemented by itself. The bottom layer is based on the GLib message loop to provide support for other parts of the client. .

4.Session Management

Manage two WebSocket connections between replica and contact and the message server. The former receives messages sent by the other party, and the latter is used to send messages, receive online and offline notifications, receive rejection notifications, and so on.

5.Video call session management:

The internal is a state machine, which manages the status of the idle, initiating, receiving notification, and official start of the video call.

6.Push Stream Management

Responsible for managing the encoding, decoding, pushing and receiving of streaming media

7.Data persistence

First, it is responsible for storing user configuration information locally, such as tokens, private keys, etc. The second is responsible for backing up the message to the database through the ORM layer for later viewing, as well as storing the public key of the contact and the blocking list, etc.

CONCLUSION

In response to the increasing demand for the security and privacy of instant messaging systems in recent years, this system proposes an implementation of an end-to-end encrypted instant messaging tool. End-to-end encryption means that in a communication system, only the end user can decrypt messages, preventing Internet service providers, including but not limited to, even the communication system providers from obtaining the key used to decrypt the content of the communication. This system uses X25519 key exchange, the first use of trust (TOFU) such as `authentication, and symmetric encryption with Poly1305 Chacha20 verify implementation of end to end encryption.

References

[1] Yoav Nir, Adam Langley, RFC8439, ISSN: 2070-1721. ChaCha20 and Poly1305 for IETF Protocols, Internet Research Task Force (IRTF), 2018.

[2] David A. McGrew, Kevin M. Igoe, Margaret Salter, RFC6090, ISSN: 2070-1721, Fundamental Elliptic Curve Cryptography Algorithms, Internet Engineering Task Force (IETF), 2011.

[3] Abu-Salma R, Redmiles E M, Ur B, et al. Exploring user mental models of end-to-end encrypted communication tools. 8th USENIX Workshop on Free and Open Communications on the Internet (FOCI 18). 2018.

[4] Lee J, Choi R, Kim S, et al. Security analysis of end-to-end encryption in Telegram. Simposio en Criptografía Seguridad Informática, Naha, Japón. Disponible en https://bit. ly/36aX3TK. 2017.