Team FSociety

An End-to-End Encryption Chat Application Overview

Estefany Montoya
Rone Caballero
Prof. Mehrdad Aliasgari
September 23, 2016

Application Properties

Our web chat application consists of sending a message to another. Only the user themselves have the access to their own chat. The users can find other clients through the server. A database will be used to store a history of messages for the user even if they are not online.

The backend will consist of using Java with the platform Akka. The server that is going to be used is Amazon Web Server. The user can visit the website, request a user to chat with by using the server database, have a window open for sending each other messages. The use of Java with Akka will be used to create the RESTful web server. JSON will be used for maintenance of the message database. MySQL will be used for a database. The Transport Security will be secured by SSL encryption.

Assets/Stakeholders

The assets of this project are to: maintain the integrity and privacy of messages

- Assets:
  - Protect messages
  - Privacy of User

The stakeholder of this project is the user.

- Stakeholders:
  - User
  - Service Provider

Adversarial Model

- We need to take into consideration all type of adversaries. Including passive, active, insider, and outsider.
- Our assets include protecting the messages so we would need to focus on different attacks that can come from:
  1. Server administrator
  2. Transport Channels
- In all threat situations knowing the knowledge of your adversary is very important. Knowing what the adversary knows can be the best defense in finding what spots the adversary wants to attack. You will be one step ahead of the adversary

Possible vulnerabilities

- A high priority threat is a server with an admin that can read the message. This is considered an insider attack. If the server administrator can read the messages that the users send, then their privacy will be broken.
- A The highest priority threat to take into consideration is a man in the middle attack, as shown in Figure 1.1. In this situation the connection between the server and client is intercepted and the adversary, Eve, can see the conversation between the user and the recipient and change the messages sent between them if wanted.

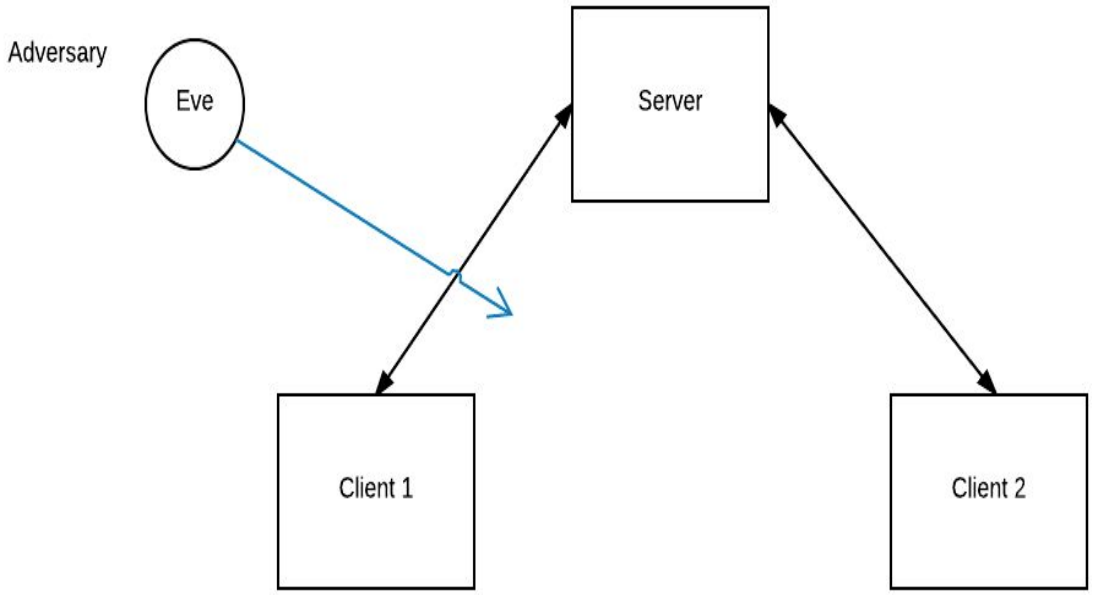


Figure 1.1 An example of a Man in the Middle Attack

- Infiltrate connections between server and client.
- One possible attack is depicted in Figure 1.2

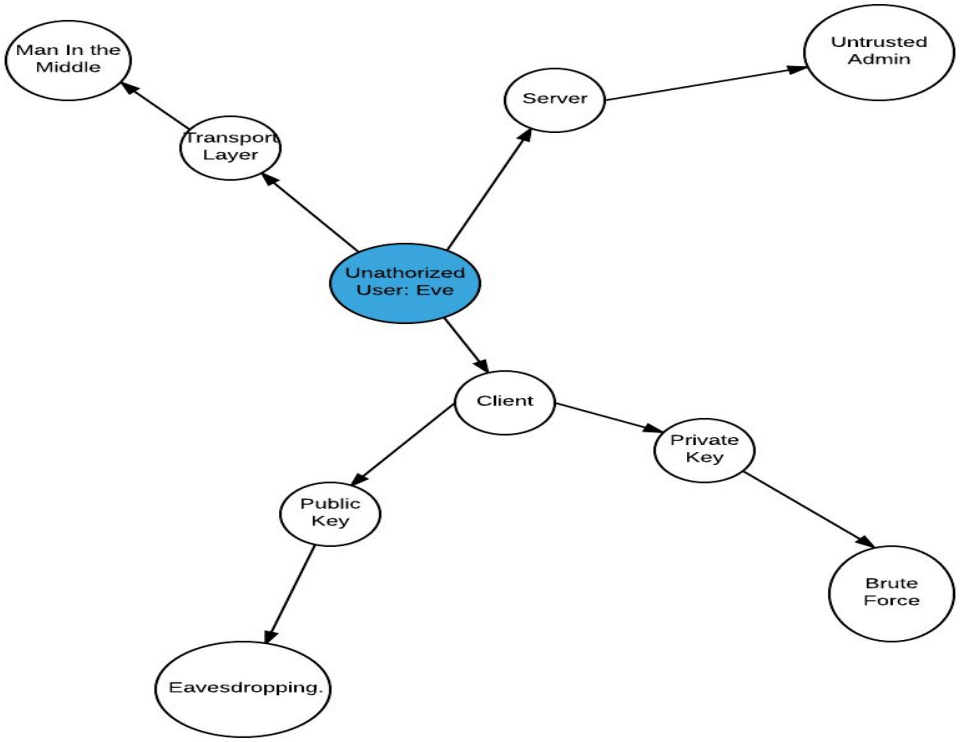


Figure 1.3 Attack Tree: An example of possible attacks

Related Work

WhatsApp is a popular chat application where millions of people communicate with one another. A main priority for WhatsApp is providing their users the privacy they need to communicate freely without being worried that their messages are being monitored by someone else. Unlike other applications that only encrypt messages between the user and whomever they are communicating with WhatsApp uses end-to-end encryption to make sure that it's only the user and the person that they are communicating with the only people in that conversation. WhatsApp end-to-end encryption secures messages with a lock and only the user and the receiver of the message have a special key that will unlock the message and thus be able to read them. Every message that is sent between user and receiver in WhatsApp has its own lock and key. The use of keys is very important in WhatsApp.

WhatsApp has two terms to declare the type of keys that they use. The two types of keys that WhatsApp uses are referred to as Public Key Types and Session Key Types.

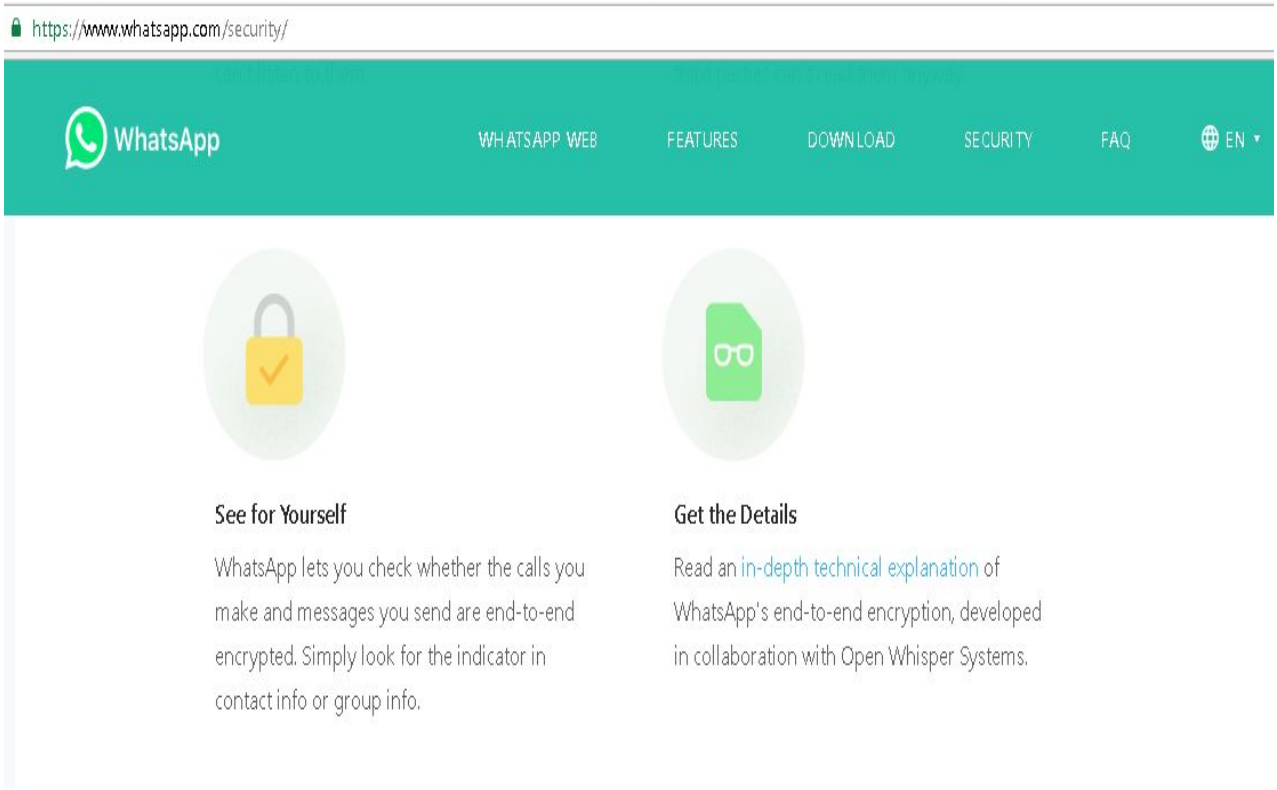
Public Key Types	Session Key Types

Identity Key Pair - A long-term Curve25519 key pair, generated at install time.	**Root Key**- A 32-byte value that is used to create *Chain Keys*.
Signed Pre Key-A medium-term Curve25519 key pair generated at install time, signed by the *Identity Key*, and rotated at a periodic time basis.	**Chain Key**- A 32-byte value that is used to create *Chain Keys*.
One-Time Pre Keys- A queue of Curve25519 key pairs for one time use, generated at install time, and replenished as needed	**Message Key**- An 80-byte value that is used to encrypt message contents. 32 bytes are used for an AES-256 key, 32 bytes for a HMAC-SHA256 key, and 16 bytes for an IV.

*Info taken from pg. 3 from WhatsApp Encryption Overview.

The basis of WhatsApp end-to-end encryption is the Signal Protocol who is Open Source and available at http://github.com/whispersystems/libsignal-protocol-java/ This protocol prevents WhatsApp and other parties from accessing the plaintext of either the messages or call that the user makes. The Signal Protocol prevents encryption keys from any user, even if they were compromised by other parties, to be used again to decrypt old messages. Thus blocking completely the messages from being read.

WhatsApp uses end-to-end encryption to secure the privacy of their users, their messages and their use of keys is very extensive. For more information on how WhatsApp ensures end-to-end encryption download their Encryption Overview located at: http://whatsapp.com/security/ and scroll down to where it says Get Details and click on the link *in-depth technical explanation*.



Solution Description

Clients will both have a private key and public key. The public key, is shared to anyone upon request for those who wishes to communicate with another client. The connection between two clients is encrypted with TLS 1.2 (http) to keep any adversaries away, as shown in figure 1.3 where the tube surrounding the connection indicates the TLS encryption.

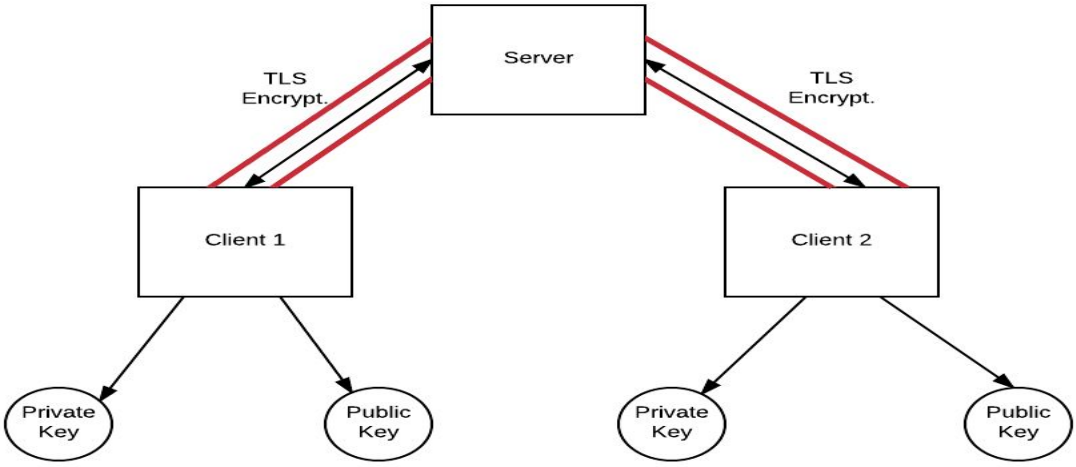


Figure 1.3 channel between server and clients is encrypted by TLS 1.2

The communication between users is done using PGP, Pretty Good Privacy. PGP uses the key system. This implies that each user has a public key, generated by RSA, and a private key. A private key is only known by the user whom the key belongs to because it is what is used to decrypt messages. A public key is known as the encryption key, this key will encrypt the messages the user is sending. Figure 1.4 depicts how messages between users is being sent using PGP.

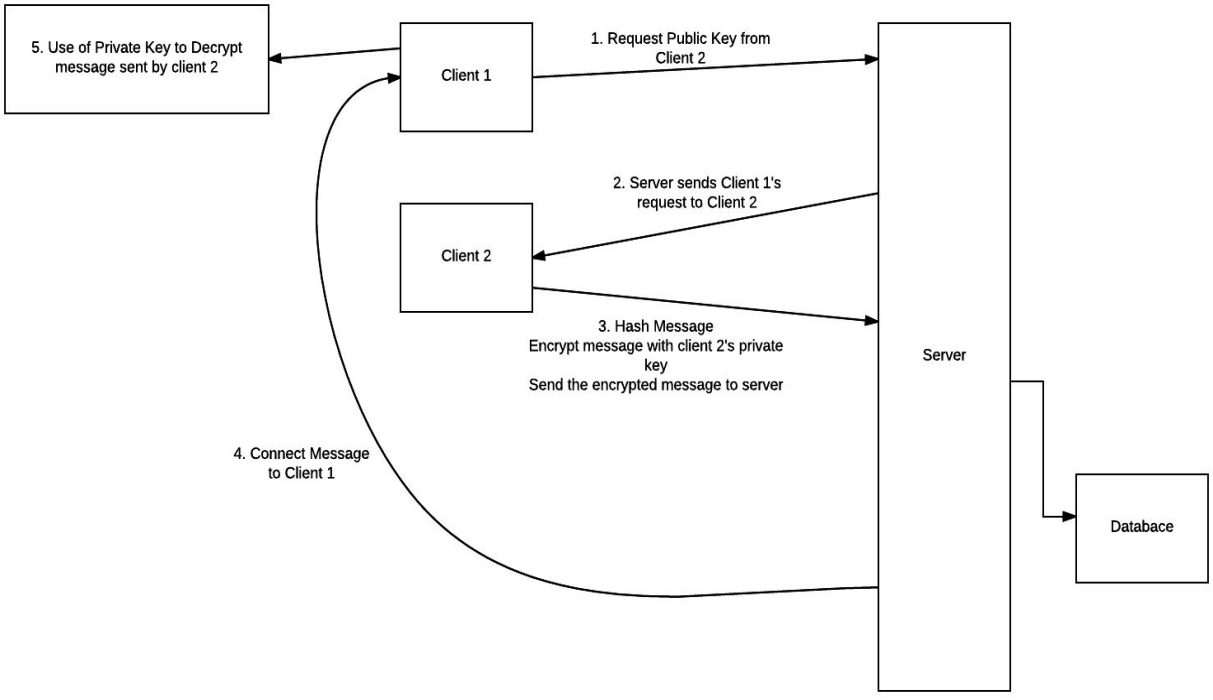


Figure 1.4 Depiction of how messages are transmitted between two clients.

There is a question of how to find another user's public key. This question is answered by encryption and database. An example would be making a copy of the key, hashing the key value, and encrypting the key. The encryption will have the copy of the public key multiplied by a prime factor (random generated). Then, for the database, the user will see a list of other clients to send their copied, encrypted key. The recipient will then be able to decrypt the key with their private key to do the vice versa. Figure 1.5 gives a general example of a user requesting another user's public key.

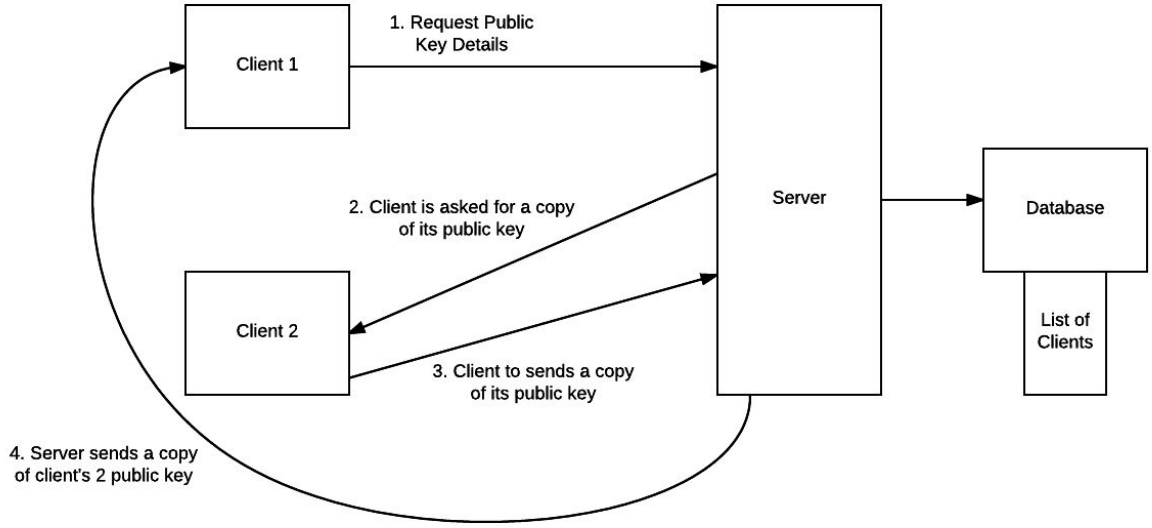


Figure 1.5 Requesting Public Key from Another User

Overall, The solution is involves three types. There is a transport layer security to encrypt the connection between the clients, preventing Man in the Middle Attacks. There is Pretty Good Privacy to encrypt message content from untrusted servers. Lastly, there is Requesting for Public Key Copy to share public keys without other clients seeing the activity.

Analysis

The solutions mentioned above will help establish end-to-end encryption for a chat application. T.L.S will allow the client and server to have secure communication because the channel that links them together will be private and no adversaries will be able to be a threat. Next, PGP will allow for a secure exchange of message content with integrity because PGP will encrypt messages that will not be tampered or violated by third parties. With the help of RSA, the encryption will be concatenated by an alphanumeric key with the size of 2048 bits, making it hard to brute force or any other method of cryptography. Lastly, we took into consideration a user request for another user's public key. We understand that user activity is meant to be

discrete. Using a prime number to exchange public keys will be used since it’s not factorable by any other set of keys except by 1 and itself. With the help of the database, the exchange of public keys will be possible and kept protected from eavesdroppers.