Security in real-time chat application

Loïc Branstett – Algebra University College

Abstract

Since the 1990s, two technologies have reshaped how we see and experience the world around us. These technologies are the Internet and mobile communication, especially smartphones. The Internet provides a cheap and convenient way to explore and communicate with distant people. A multitude of services have converged on the smartphone platform, and potentially the most notable is social networking. With increased interconnectivity and use of online services, concerns about consumers' security and privacy are growing. In this paper, we evaluate the security- and privacy-preserving features provided by current and past online chat services and their transmission protocols.

Keywords: Security; Real-time tchat; TLS; Mobile tchat; chat

1. Introduction

Real-time chat application are a real-time system. A real-time system means that the system is subjected to real-time, i.e., the response should be guaranteed within a specified timing constraint or the system should meet the specified deadline.

There are two types of real-time systems based on timing constraints:

Hard real-time system: This type of system can never miss its deadline. Missing the deadline may have disastrous consequences. The usefulness of results produced by a hard real-time system decreases abruptly and may become negative if tardiness increases. Tardiness means how late a real-time system completes its task with respect to its deadline.

Soft real-time system: This type of system can miss its deadline occasionally with some acceptably low probability. Missing the deadline has no disastrous consequences. The usefulness of results produced by a soft real-time system decreases gradually with an increase in tardiness.

Generally real-time chat services are "Soft real-time system", they can miss receiving a message without disruption of service, but some of them are "Hard real-time system" meaning that after not receiving the information's it would not be possible to recover it or even if it was possible it wouldn't be useful anymore as the deadline for receiving the information may have been crossed.

Because of these constraints real-time system use very specific protection against malicious actors. Those protections are generaly directly integrated into their protocols.

2. Encryption

To protect communications modern protocols use some for of encryption between the sender and receiver in order to prevent interception of the content of the message.

Encryption is a way of scrambling data so that only authorized parties can understand the information. In technical terms, it is the process of converting human-readable plaintext to incomprehensible text, also known as ciphertext. Encryption requires the use of a cryptographic key: a set of mathematical values that both the sender and the recipient of an encrypted message agree on. See Illustration 1: Example of encryption of an example of encryption.

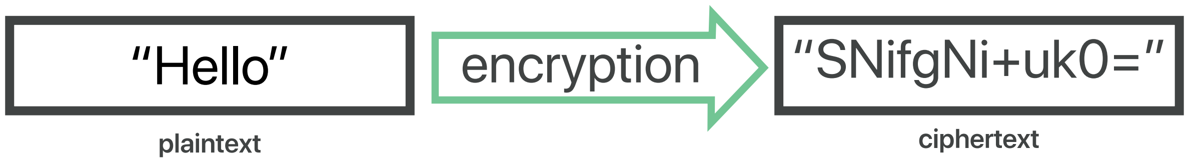

Illustration 1: Example of encryption

Although encrypted data appears random, encryption proceeds in a logical, predictable way, allowing a party that receives the encrypted data and possesses the right key to decrypt the data, turning it back into plaintext. Truly secure encryption algorthim will use keys complex enough that a third party is highly unlikely to decrypt or break the ciphertext by brute forcing it.

2.1. Encryption/Decryption Process

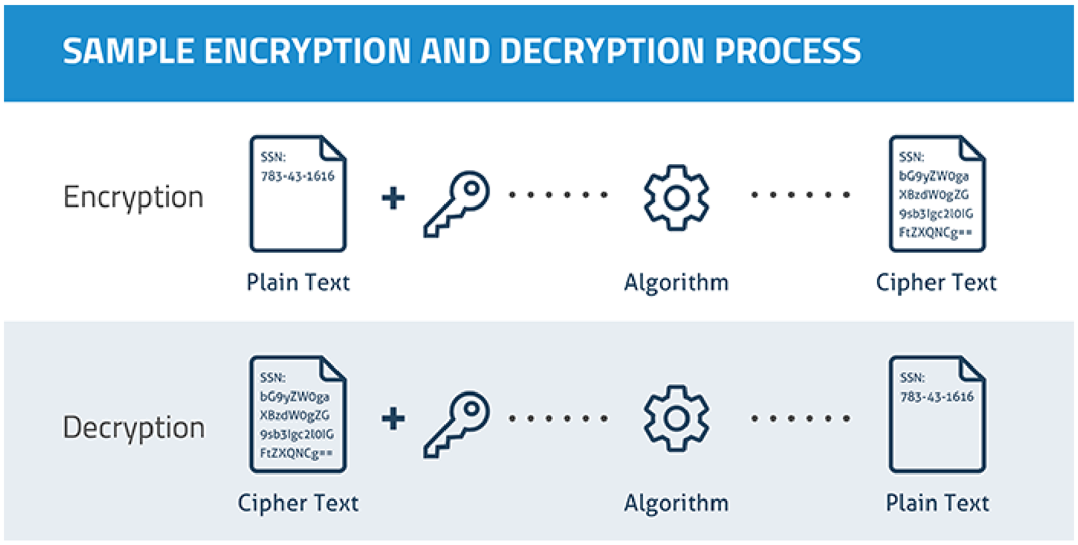
SAMPLE ENCRYPTION AND DECRYPTION PROCESS
Encryption
SSN: 783-43-1616
Plain Text
Algorithm
SSN: bG9yZW0ga XBzdW0gZG 9sb3Igc2l0IG FtZXQNCg==
Cipher Text
Decryption
SSN: bG9yZW0ga XBzdW0gZG 9sb3Igc2l0IG FtZXQNCg==
Cipher Text
Algorithm
SSN: 783-43-1616
Plain Text

Illustration 2: Sample encryption and decryption process

2.2. Key principle of encryption

Privacy: Encryption ensures that no one can read communications or data at rest except the intended recipient or the rightful data owner. This prevents attackers, Internet service providers, and in some cases governments from intercepting and reading sensitive data.

Security: Encryption helps prevent data breaches, whether the data is in transit or at rest. If a corporate device is lost or stolen and its hard drive is properly encrypted, the data on that device will still be secure. Similarly, encrypted communications enable the communicating parties to exchange sensitive data without leaking the data.

Data integrity: Encryption also helps prevent malicious behavior such as on-path attacks. When data is transmitted across the Internet, encryption (along with other integrity protections) ensures that what the recipient receives has not been tampered with on the way.

Authentication: Public key encryption, among other things, can be used to establish that a receiver owns the private key listed in their certificate. This allows the sender to be sure that they are connected and exchanging with the right sender.

Regulations: For all these reasons, many industry and government regulations require companies that handle user data to keep that data encrypted. HIPAA, PCI-DSS, GDPR are example of such governmental or industrial regulations.

3. End-to-End Encryption

Based on these Encryption key principles (above) a new trend of security encryption called End-to-End encryption has started to appear in the early 2010s to further improve the security and robustness of communications.

End-to-end encryption (E2EE) is a type of messaging that keeps messages private from everyone, including the messaging service. When E2EE is used, a message only appears in decrypted form for the person sending the message and the person receiving the message. The sender is one "end" of the conversation and the recipient is the other "end"; hence the name "end-to-end."

A parallel can be established with mail. The person sending the letter is able to read it, and the person who receives it can open it and read it, however Postal service employees cannot read the letter because it remains sealed in the envelope.

3.1. Uniqueness of end-to-end encryption

Many messaging services offer encrypted communications without true end-to-end encryption. A message is encrypted as it travels from the sender to the service's server, and from the server to the recipient, but when it reaches the server, it is decrypted briefly before being re-encrypted. This is the case with the common encryption protocol, like TLS.

A service can promise that it will not read the message in its decrypted form — just as a postal service who transfer the letter to a new envelope before delivering it to the recipient might promise that its employees will never read letters while they are transferred to their new envelope. But someone sending a message still has to trust that the messaging service will keep its promise.

E2EE is "end-to-end" because it is impossible for anyone in the middle to decrypt the message. Users do not have to trust that the service they are using will not read their messages: it is not possible for the service to do so. This is like instead of sending a letter in an envelope, someone sent it in a locked box to which only they had the key. It would be physically impossible for anyone to read the letter aside from its intended recipient.

This makes E2EE really interesting and particularly important for any communication regardless of the importance of the message.

3.2. Encryption algorithms of E2EE

End-to-end encryption uses a specialized form of encryption called public key encryption also called asymmetric encryption. Public key encryption enables two parties to communicate without having to send the secret key over an insecure channel.

Public key encryption relies on using two keys instead of one: a public key and a private key. While anyone, including the messaging service, can view the public key, only one person knows the private key. Data encrypted with the public key can only be decrypted with the private key (not the public key), and vice versa. This contrasts with symmetric encryption, where only one key is used to both encrypt and decrypt.

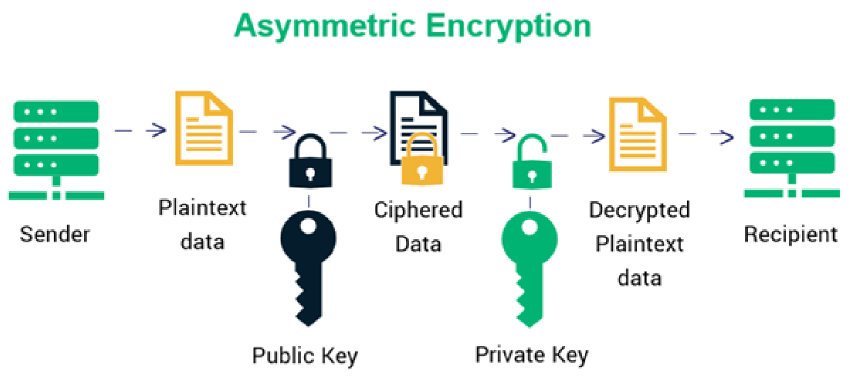

Illustration 3: Asymmetric Encryption [0]

4. Protocols

4.1. Plain-text protocols

Plain-text is generally defined as an encryption algorithm that does nothing to its input. The output is the input without any modification whatsoever. So, in a plain-text protocol like HTTP (Hypertext Transfer Protocol) all the data will be transferred in plain-text meaning that there is no protection whatsoever.

These type of protocols are discouraged for any real world application. As of 2022 no online chat service is using a plain-text protocol.

4.2. WebSocket

WebSocket is a Web standard for an application layer network protocol and a World Wide Web programming interface for creating full-duplex communication channels over a TCP connection for Web browsers.

The WebSocket protocol enables interaction between a web browser (or other client application) and a web server with lower overhead than half-duplex alternatives such as HTTP polling, facilitating real-time data transfer from and to the server. This is made possible by providing a standardized way for the server to send content to the client without being first requested by the client, and allowing messages to be passed back and forth while keeping the connection open. In this way, a two-way ongoing conversation can take place between the client and the server.

4.3. Tox Protocol

Tox is a peer-to-peer instant-messaging and video-calling protocol that offers end-to-end encryption.

Users are assigned a public and private key, and they connect to each other directly in a fully distributed, peer-to-peer network. Users have the ability to message friends, join chat rooms with friends or strangers, voice/video chat, and send each other files. All traffic over Tox is end-to-end encrypted using the NaCl library, which provides authenticated encryption and perfect forward secrecy.

4.4. Matrix Protocol

Matrix is an open standard and communication protocol for real-time communication. It aims at making real-time communication work seamlessly between different service providers, in the way that standard Simple Mail Transfer Protocol email currently does for store-and-forward email service, by allowing users with accounts at one communications service provider to communicate with users of a different service provider via online chat, voice over IP, and video telephony. It therefore, serves a similar purpose as protocols like XMPP, but is not based on any existing communication protocol.

From a technical perspective, it is an application layer communication protocol for federated real-time communication. It provides HTTP APIs and open source reference implementations for securely distributing and persisting messages in JSON format over an open federation of servers. It can integrate with standard web services via WebRTC, facilitating browser-to-browser applications.

4.4.1. Double Ratchet Algorithm

In cryptography, the Double Ratchet Algorithm is a key management algorithm that was developed by Trevor Perrin and Moxie Marlinspike in 2013. It can be used as part of a cryptographic protocol to provide end-to-end encryption for instant messaging. After an initial key exchange it manages the ongoing renewal and maintenance of short-lived session keys. It combines a cryptographic so-called "ratchet" based on the Diffie–Hellman key exchange (DH) and a ratchet based on a key derivation function (KDF), such as a hash function, and is therefore called a double ratchet. See Illustration 4: Double Ratchet Algorithm.

The algorithm is considered self-healing because under certain conditions it prevents an attacker from accessing the clear-text of future messages after having compromised one of the user's keys. New session keys are exchanged after a few rounds of communication. This effectively forces the attacker to intercept all communication between the honest parties, since they lose access as soon as a key exchange occurs that is not intercepted. This property was later named Future Secrecy, or Post-Compromise Security.

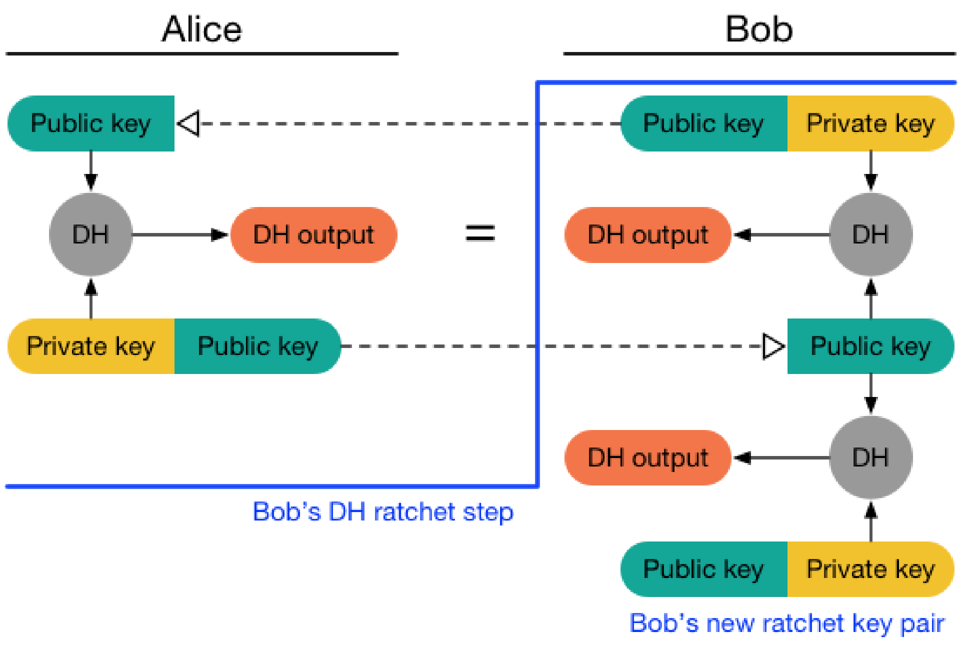

Illustration 4: Double Ratchet Algorithm

4.5. Signal Protocol

Used by the Signal application software.

The protocol combines the Double Ratchet algorithm, prekeys, and a triple Elliptic-curve Diffie–Hellman (3-DH) handshake, and uses Curve25519, AES-256, and HMAC-SHA256 as primitives.

The protocol provides confidentiality, integrity, authentication, participant consistency, destination validation, forward secrecy, post-compromise security (aka future secrecy), causality preservation, message unlinkability, message repudiation, participation repudiation, and asynchronicity. It does not provide anonymity preservation and requires servers for the relaying of messages and storing of public key material.

The Signal Protocol also supports end-to-end encrypted group chats. The group chat protocol is a combination of a pairwise double ratchet and multicast encryption.[18] In addition to the properties provided by the one-to-one protocol, the group chat protocol provides speaker consistency, out-of-order resilience, dropped message resilience, computational equality, trust equality, subgroup messaging, as well as contractible and expandable membership.

5. Messaging apps comparison

According to [1] and [2] here is a summary of the major messaging apps based on features related to the all overall security:

- Transparency Reports: Do they provide one ?
- Collects User Data: Do they collect user data ?
- Encryption by default: Are they using encryption by default ?

- Open Source (client and server): Is the client and/or server open source ?
- Encryption of metadata: Does they encrypt metadata ?
- Store metadata: Does they store metadata on their data center ?

Features	Messenger [3]	iMessage [4]	Telegram [5]	WhatsApp [6]	Wire [7]	Signal [8]	Session
Transparency Reports	✓	✗	✗	✓	✓	✓	✓
Collects User Data	✓	✓	✓	✓	✗	✗	✗
Encryption by default	✗	✓	✗	✓	✓	✓	✓
Open Source (client and server)	✗	✗	✗	✗	✓	✓	✓
Encryption of metadata	✗	✗	✗	✗	✗	✓	✓
Store metadata	✓	✓	✓	✓	✓	✗	✗

6. Conclusion

We saw in this paper different approaches taken by real-time security protocols about security in real-time chat applications. The most advanced protocol and secure one are the Matrix and Signal protocols, they use the Double Ratchet Algorithm for the key exchange and management. We also saw that being secure nowadays require some form of end-to-end encryption. It would be interesting to explore the new kind of encryption algorithm like homomorphic encryption for real-time chat applications.

Acknowledgments

We wish to thank Algebra University College for having provided the resources and inspiration for this paper as part of their System Engineering course.

References

0: , Asymmetric Encryption, , https://sectigostore.com/blog/what-is-asymmetric-encryption-how-does-it-work/
1: Jonny Zerah, Elements of Private, Secure Messaging Apps, , https://our.status.im/elements-of-private-secure-messaging-apps/
2: , Secure Messaging Apps Comparison, , https://www.securemessagingapps.com/
3: , Messenger, , https://www.messenger.com/
4: , iMessage, , https://support.apple.com/fr-fr/messages
5: , Telegram, , https://telegram.org/
6: , WhatsApp, , https://www.whatsapp.com/
7: , Wire, , https://wire.com/en/
8: , Signal, , https://www.signal.org/fr/