Applied Cryptography Workshop

Final Project Report

"*This document should contain a detailed explanation of the project goals and how you accomplished them. Make sure to specify what your security assumptions are (what you assume the adversary can/cannot do) and what happens if the assumptions are violated. This document should be in PDF format; put it in a file named "`about-project.pdf`" in the root of your main rhodecode repository.*" - Instructions

Goals:

Our goals for this project were creating a software library for secure group conversation over an unsafe media. The library should provide the following guarantees / functionality:

Basic Functionality:

- Communication in sessions
    - We support establishing secure sessions with multiple peers using the *SecureParty* class which wraps session management functionality.
- Multiple messages exchanged within each session
    - Once a secure session is established the *SecureParty* class instance maintains a SessionCipher instance for that session and it can be used for encrypting and decrypting unbounded amounts of messages within the session.
- All parties know who the session participants are
    - Identity is based on some unique name for each peer (an email would be a reasonable choice). Once two users establish trust using public key fingerprint, this identity can also be validated.
- Long term secrets used for authentication
    - Every user generates an ECDH key pair at first login which is used for identification for all future sessions. This key pair is saved securely on the disk and can be easily transported between machines.
- Short term secrets used for privacy
    - For starting a secure conversation the peers run a simple Diffie-Hellman based protocol that enables creating a shared secret between the two parties over an unsafe media.
- Forward Secrecy: Long-term secret leak doesn't reveal past sessions
    - The key is derived using a cryptographic hash function, for every message, immediately after use. Therefore, an adversary that managed to leak the keys at some point in time will not be able to decrypt messages recorded in the past.
- Future Secrecy:Short-term secret leak doesn't compromise future sessions
    - Every message is sent with an "ephemeral" key in its header which is a public key generated by the sender's ratchet. When a peer replies to a message within a conversation, it generates an ECDH key pair and derives a new shared key

        with the ephemeral key in the received message header. This provides the randomization required for future secrecy.

- <u>Deniability:Leaked secrets can't be used to prove authorship</u>
    - The message signature key is derived by the axolotl ratchet with every message sent. After verification of a received message, the signature key may be published for deniability.

The Axolotl Ratcheting Algorithm:
Our library uses a ratcheting algorithm called axolotl which was written for TextSecure and examined by community and industry professionals and is considered safe.
Originally, this algorithm and its implementation were designed according to the TextSecure architecture which is based on a central server through which all parties communicate.

In this project, we modified this architecture and implemented a mechanism that enables two or more peers to have a secure conversations based on the axolotl ratcheting algorithm **without the need for a central server**. We did that by defining a simple protocol that emulates the server response towards each of the parties, that is, each party is the “server” for the other party.

Design Considerations:
Our library was designed work on top of any type of media and can be easily integrated into any existing chat / IM application. Key exchange messages, metadata and user payload are all serialized and encoded in Base64 and can be transported as simple character strings.

The user does not need to worry about securing any of the messages as the library secures anything of value. It is the user's responsibility to make sure the library generated key exchange message is exchanged between every pair of peers before the actual conversation begins.

Advanced Features:
- <u>Usable Out-Of-Band authentication mechanism</u>
    - Our library generates an identity key at first use. The fingerprint for this key is converted to an english sentence that can be transported to the peer during a face-to-face, phone call or email. Once the fingerprint was consumed, the peer is automatically authenticated for future sessions.

- <u>Group sessions</u>
    - We support group conversations using multiple peer-to-peer sessions. Every message sent in a secure group conversation is sent and encrypted separately to every member of the group.

- <u>Inconsistency detection</u>

    - Every message is sent with metadata that enables the receiving user to detect any conversation inconsistencies. The library also provides a flexible API displaying the inconsistency to the user.

- Repairing inconsistencies
    - Once some inconsistency was detected, a peer can ask another to transmit some message according to its index. The user can ask the library for a retransmission and the history on the receiving side will automatically be repaired.

Security Assumptions:

In this part we specify the assumptions we make about our adversary.

- Attack models
    - We assume the adversary to be able to perform, and our system to be resilient to the following:
        - Ciphertext-only attacks (COA)
        - Known-plaintext attacks (KPA)
        - Chosen-plaintext attacks (CPA)
        - Adaptive Chosen-plaintext attacks (CPA2)
        - Chosen-ciphertext attacks (CCA)
        - Adaptive Chosen-ciphertext attacks (CCA2)

- Denial of Service
    - We do not protect the users against denial of service attacks, as an adversary can always drop the communication between the peers. It is the user's responsibility to worry about message drop, reordering and other line interruptions.

- Non-Authenticated conversations
    - Two peers can have a secure channel before establishing trust (that is, before they used an authenticated, Out-Of-Band channel for exchanging fingerprints). At this stage, the peers are vulnerable to MITM (Man-In-The-Middle) attacks by an active adversary. It is important to understand that even in this vulnerable stage, it is safer for the peers to have the un-authenticated conversation than to send the plain data since MITM attacks are considered to require more resources than passive eavesdropping / plain data forgery.

- Mostly honest parties
    - While performing group chat, our library provides the ability for a peer to detect conversation inconsistencies and know if a party tries to display a different conversation to different peers. This mechanism relies on the assumption that only some of the peers try to "cheat", and not all of them.