Messenger Secret Conversations

Technical Whitepaper

July 8, 2016

Contents

Introduction

In this document we provide a brief technical overview of SECRET CONVERSATIONS — a specialized conversation mode within Facebook Messenger. SECRET CONVERSATIONS provides *end-to-end encryption for messages* using keys that are only available on users' devices.

SECRET CONVERSATIONS is a distinct conversation mode inside Facebook Messenger. Individual secret conversations are displayed as separate threads in Messenger and share many UX elements with regular Messenger conversations. However, SECRET CONVERSATIONS uses a different transport protocol, specialised on-device storage and separate back-end infrastructure.

The SECRET CONVERSATIONS threat model considers the compromise of server and networking infrastructure used by Messenger — Facebook's included. Attempts to obtain message plaintext or falsify messages by Facebook or network providers result in explicit warnings to the user. We assume however that clients are working as designed, e.g. that they are not infected with malware.

SECRET CONVERSATIONS relies upon the Signal Protocol. Messenger uses Signal Protocol's implementation as available in the open-source libsignal-protocol-java and libsignal-protocol-c libraries for Android and iOS respectively. SECRET CONVERSATIONS also incorporates new abuse-reporting features which are not present other platforms which use the Signal Protocol.

In this document we describe SECRET CONVERSATIONS, starting with the transport protocol. We will then cover abuse reporting and close with how we handle on-device storage.

Transport Protocol Overview

SECRET CONVERSATIONS is a *device-to-device* conversation mode. Messages are only accessible from the two devices which participate in a conversation. This differs from regular Messenger conversations where users are typically logged in with the same Facebook account in multiple devices and browser instances. Instead, to use SECRET CONVERSATIONS users designate a preferred device upon which their secret conversations will be available, e.g. their primary phone.

Keys

Each device manages various cryptographic keys. All keys are generated or derived on-device. Private keys are never sent to Facebook.

Public keys All public key operations use Curve25519. Each device uses the following public-secret keypairs:

- The *Identity Key* keypair ($\mathsf{IK}_{pk}, \mathsf{IK}_{sk}$). This is a long-term keypair which is generated the first time Messenger runs.
- The *Signed Pre-Key* keypair ($\mathsf{SPK}_{pk}, \mathsf{SPK}_{sk}$). This is a medium-term keypair which is rotated periodically. It is signed by IK_{sk}.
- The *One-Time Pre-Key* keypairs ($\mathsf{OTPK}_{pk}, \mathsf{OTPK}_{sk}$). These keypairs are generated in batches by clients. They facilitate asynchronous conversation initiation.
- The *Ephemeral Key* keypairs ($\mathsf{EK}_{pk}, \mathsf{EK}_{sk}$). A new ephemeral keypair is generated for each round of communication within a secret conversation and is subsequently discarded.

Session keys When starting a secret conversation the participating devices derive symmetric session keys. These are:

- The *Root Key (*RK*)* is a 256-bit key which is used to derive Chain Keys in the Signal Protocol ratchets.

- *Chain Keys* (CK) are each 256-bit values which are used to derive Message Keys.
- *Message Keys* (MK) are each 640-bit values which consist of 256 bits for an AES-256 key, 256 bits for an $\mathsf{HMAC{\cdot}SHA256}$ key, and 128 bits for an Initialization Vector (IV) for AES-CBC encryption.

When a Messenger initialises Secret Conversations it generates and then uploads to Facebook its permanent IK_{pk} and the current SPK_{pk}. It also generates a batch of one-time pre-key keypairs and uploads their public parts to Facebook.

Conversation Initiation

Each secret conversation consists of two devices: one *Initiator* device and one *Responder* device (I and R respectively). Let HKDF be a secure hash-based key derivation function, and ECDH indicate the elliptic curve Diffie-Hellman function applied to a secret and public key. To create a new conversation:

1. The Initiator obtains from Facebook $\mathsf{IK}^R_{pk}, \mathsf{SPK}^R_{pk}$ and OTPK^R_{pk} for one one-time pre-key keypair generated by the Responder device.
2. Facebook deletes OTPK^R_{pk} from the Responder's list of available one-time pre-keys.
3. The Initiator generates a fresh ephemeral keypair $(\mathsf{EK}^I_{pk}, \mathsf{EK}^I_{sk})$.
4. The Initiator now computes the first *root key* RK as follows:

$$a = \mathsf{ECDH}(\mathsf{IK}^I_{sk}, \mathsf{SPK}^R_{pk}), \quad b = \mathsf{ECDH}(\mathsf{EK}^I_{sk}, \mathsf{IK}^R_{pk})$$

$$c = \mathsf{ECDH}(\mathsf{EK}^I_{sk}, \mathsf{SPK}^R_{pk}), \quad d = \mathsf{ECDH}(\mathsf{EK}^I_{sk}, \mathsf{OTPK}^R_{pk})$$

$$\mathsf{RK} = \mathsf{HKDF}(a||b||c||d)$$

   Using the RK the Initiator can calculate the first CK and MK (as described next) and use those to start sending messages.

5. The Initiator sends the first encrypted message to the Responder, including the fresh EK^I_{pk} it generated previously.

Upon receiving the first encrypted message the Responder:

1. Recomputes RK as above by performing the four ECDH operations using the other part of the same keypairs available locally.
2. Recomputes the first CK and MK, and decrypts the message.
3. Deletes $(\mathsf{OTPK}^R_{pk}, \mathsf{OTPK}^R_{sk})$ from its local storage.

Message exchange

Each message in SECRET CONVERSATIONS is encrypted with $\mathsf{AES{\cdot}CBC}$ and authenticated using $\mathsf{HMAC{\cdot}SHA256}$. The unique MK is derived from the current CK and RK. Their first value is:

$$\mathsf{CK} = \mathsf{RK}$$

$$\mathsf{MK} = \mathsf{HKDF}(\mathsf{CK})$$

In each message exchange, the sender generates a fresh ephemeral keypair $(\mathsf{EK}_{pk}^{sender}, \mathsf{EK}_{sk}^{sender})$ and includes the public part in the outgoing message. The recipient calculates the current MK value using $\mathsf{EK}_{pk}^{sender}$ and can decrypt the message. It too then generates a fresh ephemeral keypair $(\mathsf{EK}_{pk}^{receiver}, \mathsf{EK}_{sk}^{receiver})$ and derives new keys RK', CK' and MK' for use with the next response by updating the previous symmetric key values as follows:

$$\mathsf{RK}', \mathsf{CK}' = \mathsf{HKDF}(\mathsf{ECDH}(\mathsf{EK}_{sk}^{receiver}, \mathsf{EK}_{pk}^{sender}))$$

$$\mathsf{MK}' = \mathsf{HKDF}(\mathsf{CK}')$$

If a second message is sent before the opposing party responds, the sender uses a new chain key $\mathsf{CK}'' = \mathsf{HKDF}(\mathsf{CK})$ and a corresponding $\mathsf{MK}'' = \mathsf{HKDF}(\mathsf{CK}'')$.

The Signal Protocol's implementation is open-source and available at `https://github.com/whispersystems/libsignal-protocol-java/`.

Image Attachments

If a message includes images they are encrypted and uploaded to Facebook. For each image, the sender:

1. Generates a pseudorandom 256-bit AES key K and a 96-bit pseudorandom Initialization Vector IV.

2. Encrypts the image using AES-GCM[1] encryption using K and IV. The IV is placed into the header (Associated Data) of the GCM encryption. The IV and the GCM authentication tag are attached to the resulting ciphertext to form the encrypted file.

3. Computes a SHA256 hash H of the resulting encrypted file and uploads the file to Facebook. The sender obtains a unique identifier for future retrieval.

4. Encodes the unique identifier, K, H, image metadata, and an optional thumbnail into a message. The message is encrypted and sent to the recipient who downloads the content, verifies the hash H, and decrypts the file using K.

[1] Messenger uses the AES-GCM implementation provided by OpenSSL. On Android, Messenger employs Conceal. Conceal is open-source and available at `https://facebook.github.io/conceal/`

Stickers

Stickers are images chosen from a set of collections hosted by Facebook. Each sticker is referenced by a unique identifier. The sender chooses a sticker to attach in a conversation from a set of sticker previews available in the message composer. The sender then sends the corresponding sticker identifier as an encrypted message to the recipient. Unless the sticker file is available locally, both the sender and the receiver submit the sticker identifier to Facebook and download the corresponding sticker file. Facebook may infer the use of individual stickers when they are first used on a device but devices cache sticker files and avoid repeat fetches if a sticker is later reused.

Conversation Metadata

During a secret conversation, devices generate metadata such as delivery and read receipts to indicate successful message transmission and display, respectively. Conversation metadata such as delivery and read receipts do not contain message plaintext and are not end-to-end encrypted.

Key Verification

For every secret conversation Messenger exposes in its interface both participants' identity keys (i.e. IK_{pk}). Users may optionally verify these keys in order to ensure no man-in-the-middle attack is compromising their secret conversations. Messenger displays the 256-bit IK_{pk} values in hexadecimal format.

Device Switchover

At any point a user may designate a new preferred device to use with Secret Conversations. Existing messages or cryptographic keys are not transferred to the new preferred device. Facebook responds with a *message-bounced* error to any future messages sent to the old preferred device by users with pre-established secret conversations. Senders that receive such bounces initiate a new secret conversation with the new preferred device and display *identity-key-changed* warnings to the user[2]. Messenger does not automatically resend bounced messages to the new preferred device — an explicit resend action from the user is required.

[2] In the sender device Messenger automatically merges the two secret conversations into a single thread.

Abuse Reporting

A participant in a secret conversation may voluntarily notify Facebook of abusive content[3]. Facebook uses such reports to identify users who violate Facebook's terms of service.

[3] The user interface for abuse reporting is currently under development.

The ability to report abuse does not represent a relaxation of the end-to-end encryption guarantees of Secret Conversations. Facebook will never have access to plaintext messages *unless one participant in a secret conversation voluntarily reports the conversation.*

Secret Conversations includes a mechanism to perform "franking" of messages sent through Facebook. This mechanism is analogous to placing a cryptographic stamp on the message, without learning the message content.

Franking

The franking mechanism must satisfy three main guarantees: *authenticity*, *confidentiality* and *third-party deniability*. The authenticity property ensures that if a user submits an report then the message must have legitimately originated from the sender's device. The confidentiality property ensures that no outside party — including Facebook — should learn the content of a message unless a participant in a secret conversation voluntarily shares that information. Finally, the third-party deniability property ensures that no party outside of Facebook can cryptographically determine the validity of a report.

Franking tag Authenticity for messages in a secret conversation is provided by the *Franking* tag $\mathsf{T_F}$. Senders must send the Franking tag along with each encrypted message. To compute $\mathsf{T_F}$, the sender first generates a 256-bit random nonce $\mathsf{N_F}$. $\mathsf{N_F}$ is added to the unencrypted message being transmitted. Next, the entire data structure is serialized into a string M, and $\mathsf{T_F}$ is computed as:

$$\mathsf{T_F} = \mathsf{HMAC{\cdot}SHA256}(\mathsf{N_F}, M)$$

$\mathsf{N_F}$ remains within the serialised, encrypted data sent to the recipient. The sender destroys any other copies of $\mathsf{N_F}$ after transmission. $\mathsf{T_F}$ is transmitted to Facebook along with each encrypted message.

Franking messages When Facebook receives $\mathsf{T_F}$, it uses a Facebook key $\mathsf{K_F}$ to compute the *Reporting* tag $\mathsf{R_F}$ over $\mathsf{T_F}$ and conversation context (*e.g.,* sender and recipient identifiers, timestamp) as:

$$\mathsf{R_F} = \mathsf{HMAC{\cdot}SHA256}(\mathsf{K_F}, \mathsf{T_F} \| \mathsf{context})$$

Both $\mathsf{T_F}$ and $\mathsf{R_F}$ are sent to the recipient along with the encrypted message. The recipient decrypts the ciphertext, parses the resulting plaintext to obtain $\mathsf{N_F}$, and verifies the structure of $\mathsf{T_F}$ prior to displaying the message. If $\mathsf{T_F}$ is not verified then the recipient discards the message without displaying it. The recipient stores the message M, $\mathsf{N_F}, \mathsf{T_F}, \mathsf{R_F}$ and $\mathsf{context}$ in its local storage.

Reporting abuse To report abuse, the recipient of a message submits to Facebook the full serialized message plaintext, $\mathsf{R_F}$, $\mathsf{N_F}$, and $\mathsf{context}$. Upon receiving this message Facebook first recomputes $\mathsf{T_F}$ and then validates $\mathsf{R_F}$ using the provided information as well as its internal key $\mathsf{K_F}$.

Security and Privacy The authenticity properties of the franking mechanism are based on reasonable assumptions about the collision-resistance of the SHA256 hash function and the unforgeability of $\mathsf{HMAC{\cdot}SHA256}$.

Authenticity In order to "forge" invalid content M', a user must either (a) produce a forged HMAC tag under Facebook's key $\mathsf{K_F}$, or (b), identify a collision $\mathsf{N_F}', M', \mathsf{context}'$ such that the HMAC of these values is equal to the HMAC of a different valid message M sent through Facebook.

Confidentiality Similarly, under reasonable assumptions about $\mathsf{HMAC{\cdot}SHA256}$, the resulting tag reveals no information about the message to Facebook or to eavesdroppers.

Third-party deniability The guarantee holds under the assumption that $\mathsf{HMAC{\cdot}SHA256}$ is a pseudorandom function and that $\mathsf{K_F}$ is never publicly revealed. Facebook rotates $\mathsf{K_F}$ periodically.

Message Storage

Secret Conversations plaintext messages are stored permanently only on the devices that participate in each conversation. Plaintext messages are protected using on-device symmetric-key encryption and optional Disappearing Messages functionality.

On-device encryption ensures that messages stored permanently on a particular device are only accessible while a user is authenticated to Facebook. Messenger allows users to switch accounts. While a second user is logged in to a particular device messages of the first user should not be accessible. However, when the first user returns to the same device they should find their messages intact.

To fulfill these requirements, clients employ two encryption keys: K_{local} and K_{remote}. Both these keys are used for AES-GCM encryption. K_{local} is generated on-device and never leaves the device it was generated on. It is used to encrypt plaintext messages before these are stored permanently on a device. K_{remote} is a long-term, user-specific key held on Facebook and delivered to the device when a user authenticates. It is used to encrypt K_{local} in local storage[4]. When Messenger switches accounts the device persists an encrypted version of K_{local} and erases K_{remote}. Upon successful reauthentication the device obtains K_{remote} from Facebook, uses it to decrypt K_{local} and gains access to messages.

[4] K_{remote} is currently only available and used to encrypt K_{local} on Android. On iOS the feature is under development.

On iOS, database files are protected using the filesystem protection level `FileProtectionComplete`. This ensures that database files can no longer be accessed shortly after the user locks their device. The key used for local storage, i.e. K_{local}, is stored in keychain using the `kSecAttrAccessibleWhenUnlockedThisDeviceOnly` attribute.

Disappearing Messages ensures that messages are no longer visible within a selected time after they are sent or received. In Disappearing Messages a *timeout* value is added to the message data structure before serialisation and encryption. Both devices automatically hide messages that specify a timeout once the message timeout has elapsed. The actual deletion of message plaintext from local storage occurs shortly after each message has expired[5] in order to enable abuse reporting in the interim.

[5] Messages are currently deleted immediately after expiration. An additional timeout will be introduced once abuse reporting becomes available.

Conclusion

Secret Conversations is a new conversation mode in Messenger. Messages in Secret Conversations are encrypted end-to-end between the sender and the recipient using the Signal Protocol and its open-source implementations. Third parties — Facebook included — do not have access to message plaintext and messages can only be decrypted by their intended recipient. Users may inspect the identity keys used for end-to-end encryption and verify the confidentiality and authenticity of their communications. Decrypted messages do not leave the devices that participate in the conversation. Users retain the ability to report abusive content to Facebook.