WHATSAPP ENCRYPTION OVERVIEW

End to end encryption system

This document provides a technical explanation of WhatsApp's end-to-end encryption system. For more information, please visit WhatsApp's website at www.whatsapp.com/security

CLIENT REGISTRATION

INITIATING SESSION STARTUP

RECEIVING SESSION STARTUP

EXCHANGING MESSAGES

GROUP MESSAGE

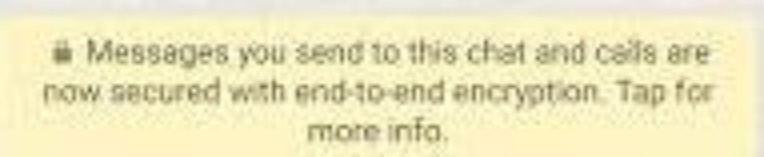
ABHISHEK GAUTAM
+91 875081005

Abhishek240196@outlook.com

22 Aug, 2016

INTRODUCTION

This document provides the overview of what is WhatsApp **end to end encryption** is which all are getting notified when we use WhatsApp. For more information on this you may visit www.whatsapp.com/security.



All of you might have seen this message which says that the WhatsApp chats and calls are end to end encrypted. WhatsApp Messenger gives people a platform to **exchange messages** (including chats, group chats, images, videos, voice messages and files) and make WhatsApp calls around the world. WhatsApp messages and calls between a sender and receiver that use **WhatsApp client software released after March 31, 2016** are end-to-end encrypted.

According to WhatsApp" *WhatsApp's end-to-end encryption ensures only you and the person you're communicating with can read what is sent,* ***and nobody in between, not even WhatsApp*** *can read your message. Your messages are secured with a lock, and only the recipient and you have the special key needed to unlock and read your message. For added protection,* ***every message you send has a unique lock and key****. All of this happens automatically. no need to turn on settings or set up special secret chats to secure your messages.* ***End-to-end encryption is always activated****, provided all parties are using the latest version of WhatsApp. There* ***is no way to turn off*** *end-to-end encryption. WhatsApp has always prioritized making your data and communication as secure as possible. And today, we're proud to announce that we've completed a technological development that makes WhatsApp a leader in protecting your private communication: full end-to-end encryption. From now on when you and your contacts use the latest version of the app, every call you make, and every message, photo, video, file, and voice message you send, is end-to-end encrypted by default, including group chats.*

The idea is simple: when you send a message, the only person who can read it is the person or group chat that you send that message to. ***No one can see inside that message. Not cybercriminals. Not hackers. Not oppressive regimes. Not even us.*** *End-to-end encryption helps make communication via WhatsApp private – sort of like a face-to-face conversation.*

We live in a world where more of our data is digitized than ever before. Every day we see stories about sensitive records being improperly accessed or stolen. And if nothing is done, more of people's digital information and communication will be vulnerable to attack in the years to come. Fortunately, end-to-end encryption protects us from these vulnerabilities.

Encryption is one of the most important tools governments, companies, and individuals have to promote safety and security in the new digital age. Recently there has been a lot of discussion about encrypted services and the work of law enforcement. While we recognize the important work of law enforcement in keeping people safe, efforts to weaken encryption risk exposing people's information to abuse from cybercriminals, hackers, and rogue states.

While WhatsApp is among the few communication platforms to build full end-to-end encryption that is on by default for everything you do, we expect that it will ultimately represent the future of personal communication.

The desire to protect people's private communication is one of the core beliefs we have at WhatsApp, and for me, it's personal. I grew up in the USSR during communist rule and the fact that people couldn't speak freely is one of the reasons my family moved to the United States.

Today more than a billion people are using WhatsApp to stay in touch with their friends and family all over the world. And now, every single one of those people can talk freely and securely on WhatsApp.”

The signal protocol used for WhatsApp end-to-end encryption is designed by **Open Whisper Systems**.
This end to end encryption has been done because even when anyone sniff chats or calls then also they **can’t get the plain text message**, they will get an encrypted message which according to WhatsApp even they can’t decrypt. What’s more, even if encryption keys from a user’s device are ever physically compromised, they cannot be used to go back in time to decrypt previously transmitted messages.

TERMS

public Key Types

- **Identity Key Pair** – A **long-term Curve25519 key pair**, generated at install time.
- **Signed Pre Key** – A **medium-term Curve25519 key pair**, generated at install time, signed by the Identity Key, and rotated on a periodic timed basis.
- **One-Time Pre Keys** – A **queue of Curve25519 key pairs for onetime use**, generated at install time, and replenished as needed.

Session Key Types

- **Root Key** – A **32-byte value** that is used to **create Chain Keys**.
- **Chain Key** – A **32-byte value** that is used to **create Message Keys.**

• **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.

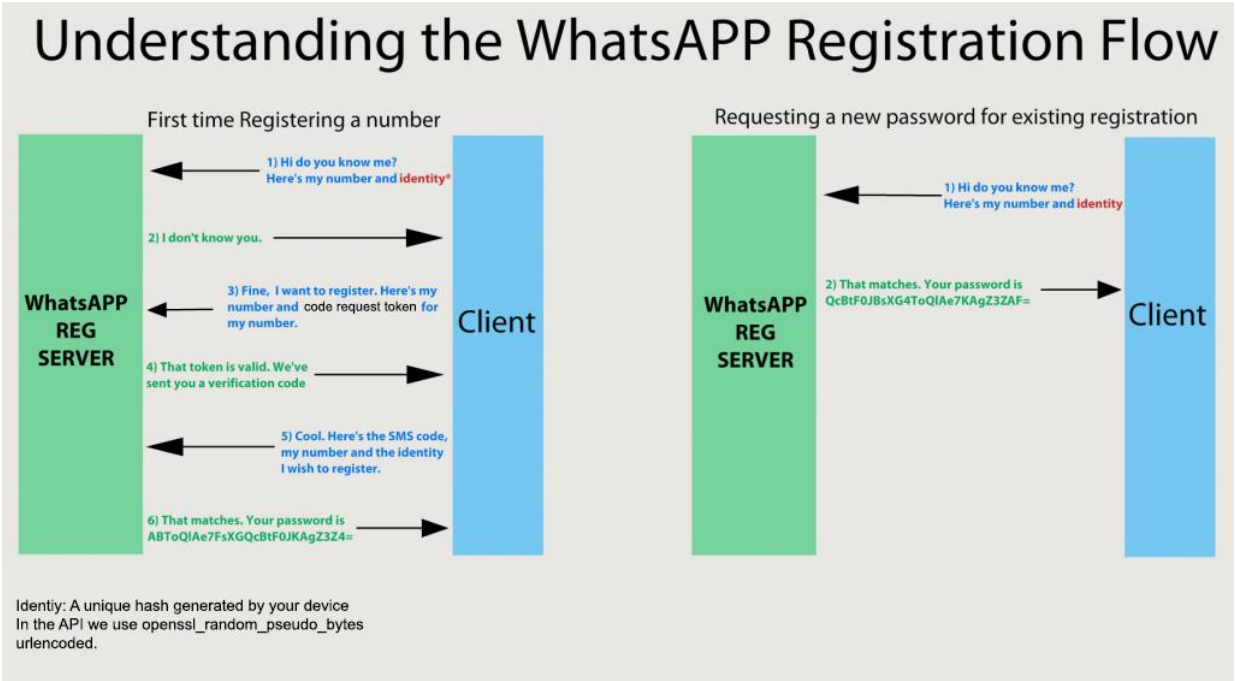
*__*NOTE:*__ In cryptography, **Curve25519** is an **elliptic curve offering 128 bits of security** and designed for use with the **elliptic curve Diffie–Hellman (ECDH) key agreement scheme.** It is one of the **fastest ECC curves**; it is not covered by any known patents, and it is less susceptible to weak random numbergenerators (Source: Wikipedia) *

CLIENT REGISTRATION

At registration time, a WhatsApp **client transmits its public Identity Key, public Signed Pre Key (with its signature), and a batch of public One-Time Pre Keys to the server.** The WhatsApp server stores these public keys associated with the user's identifier. **At no time does the WhatsApp server have access to any of the client's private keys!**

INITIATING SESSION SETUP

To communicate with another WhatsApp user, a WhatsApp client first needs to **establish an encrypted session.** Once the session is established, clients do not need to rebuild a new session with each other until the existing session state is lost through an **external event such as an app reinstall or device change.**



To establish a session:

1. The initiating client (“**initiator**”) requests the **public Identity Key, public Signed Pre Key, and a single public One-Time Pre Key for the “recipient”**.
2. The server returns the **requested public key values**. A One-Time Pre Key is only used once, so it is removed from server storage after being requested. If the recipient’s latest batch of One-Time Pre Keys has been consumed and the recipient has not replenished them, no One-Time Pre Key will be returned.
3. The initiator saves the recipient’s Identity Key as **Irecipient**, the Signed Pre Key as **Srecipient**, and the One-Time Pre Key as **Orecipient**.
4. The **initiator** generates an **ephemeral Curve25519 key pair, Einitiator**.
5. The **initiator loads** its own Identity Key as **Iinitiator**.
6. The initiator calculates a master secret as
7. **master_secret = ECDH(Iinitiator, Srecipient) || ECDH(Einitiator, Irecipient) || ECDH(Einitiator, Srecipient) || ECDH(Einitiator, Orecipient)**

***Note** ECDH defines (to some extent) how keys should be generated and exchanged between parties. How to actually encrypt data using such keys is up to us. *

If there is no One Time Pre Key, the final ECDH is omitted.

8. The initiator uses **HKDF (HMAC based Extract and Expand Key Derivation Function)** to **create a Root Key and Chain Keys from the master_secret**.

RECEIVING SESSION SETUP

After building a **long-running encryption session**, the initiator can immediately start sending messages to the recipient, even if the recipient is offline. Until the recipient responds, the initiator includes the information (in the header of all messages sent) that the recipient requires to build a corresponding session. This includes the initiator’s Einitiator and Iinitiator.

When the recipient receives a message that includes session setup information:

1. The **recipient calculates the corresponding master_secret using its own private keys and the public keys advertised in the header of the incoming message**.
2. The recipient **deletes the One-Time Pre Key used by the initiator**.
3. The initiator uses HKDF to **derive a corresponding Root Key and Chain Keys from the master_secret**.

EXCHANGING MESSAGES

Once a session has been established, clients exchange messages that are **protected with a Message Key using AES256 in CBC mode for encryption and HMAC-SHA256 for authentication.**

The Message Key changes for each message transmitted, and is ephemeral, such that the Message Key used to encrypt a message cannot be reconstructed from the session state after a message has been transmitted or received.

The Message Key is derived from a sender's Chain Key that "ratchets" forward with every message sent. Additionally, a new ECDH agreement is performed with each message roundtrip to create a new Chain Key. This provides forward secrecy through the combination of both an immediate "hash ratchet" and a round trip "DH ratchet."

Calculating a Message Key from a Chain Key

Each time a new Message Key is needed by a message sender, it is calculated as:
1. **Message Key = HMAC-SHA256(Chain Key, 0x01).**

2.The Chain Key is then updated as **Chain Key = HMAC-SHA256(Chain Key, 0x02).**

This causes the Chain Key to "ratchet" forward, and also means that a stored Message Key can't be used to derive current or past values of the Chain Key.

Calculating a Chain Key from a Root Key

Each time a message is transmitted, an ephemeral Curve25519 public key is advertised along with it. Once a response is received, a new Chain Key and Root Key are calculated as:

1. **ephemeral_secret = ECDH(Ephemeralsender, Ephemeral_recipient).**

2. **Chain Key, Root Key = HKDF(Root Key, ephemeral_secret).**

A chain is only ever used to send messages from one user, so message keys are not reused. Because of the way Message Keys and Chain Keys are calculated, messages can arrive delayed, out of order, or can be lost entirely without any problems.

Transmitting Media and Other Attachments

large attachments of any type (video, audio, images, or files) are also end-to-end encrypted:

1. The WhatsApp user sending a message (“sender”) **generates an ephemeral 32 byte AES256 key**, and an **ephemeral 32 byte HMAC- SHA256 key.**

2. The sender **encrypts the attachment with the AES256 key in CBC mode with a random IV**, then appends a MAC of the cipher text using HMAC-SHA256.

3. The sender **uploads the encrypted attachment to a blob store**.

4. The sender transmits a normal encrypted message to the recipient that contains the encryption key, the HMAC key, a SHA256 hash of the encrypted blob, and a pointer to the blob in the blob store.

5. The recipient decrypts the message, retrieves the encrypted blob from the blob store, verifies the SHA256 hash of it, verifies the MAC, and decrypts the plaintext.

GROUP MESSAGES

Traditional unencrypted messenger **apps typically employ “server-side fan-out”** for group messages. A client wishing to send a message to a group of users transmits a single message, which is then distributed N times to the N different group members by the server.

This is in contrast to **“client-side fan-out,” where a client would transmit a single message N times to the N different group members itself.**

Messages to WhatsApp groups build on the pairwise encrypted sessions outlined above to achieve efficient server-side fan-out for most messages sent to groups. This is accomplished using the “Sender Keys” component of the Signal Messaging protocol. The first time a WhatsApp group member sends a message to a group:

1. The **sender generates a random 32-byte Chain Key**.

2. The **sender generates a random Curve25519 Signature Key pair**.

3. The sender combines the 32-byte Chain Key and the public key from the Signature Key into a Sender Key message .

4. The **sender individually encrypts the Sender Key to each member of the group, using the pairwise messaging protocol explained previously.**

For all subsequent messages to the group:

1. The sender derives a Message Key from the Chain Key, and updates the Chain Key.

2. The sender encrypts the message using AES256 in CBC mode .

3. The sender signs the ciphertext using the Signature Key .

4. The sender transmits the single ciphertext message to the server, which does server-side fan-out to all group participants .

The “hash ratchet” of the message sender’s Chain Key provides forward secrecy . Whenever a group member leaves, all group participants clear their Sender Key and start over.

CALL SETUP

WhatsApp calls are also end-to-end encrypted. When a WhatsApp user initiates a call:

1. The initiator builds an encrypted session with the recipient (as outlined in Section Initiating Session Setup), if one does not already exist.

2. The initiator generates a **random 32-byte SRTp master secret**.

3. The initiator transmits an encrypted message to the recipient that signals an incoming call, and contains the SRTp master secret.

4. If the responder answers the call, a SRTp encrypted call ensues.

VERIFYING KEYS

WhatsApp users **additionally have the option to verify the keys of the other users with whom they are communicating** so that they are able to confirm that an unauthorized third party (or WhatsApp) has not initiated a man-in-the-middle attack. This can be done by scanning a QR code, or by comparing a 60-digit number.

The QR code contains:

1. **A version**.

2. **The user identifier for both parties**.

3. **The full 32-byte public Identity Key for both parties**.

When either user scans the other’s QR code, the keys are compared to ensure that what is in the QR code matches the Identity Key as retrieved from the server.

The 60-digit number is computed by concatenating the two 30-digit numeric fingerprints for each user’s Identity Key. To calculate a 30-digit numeric fingerprint:

1. Iteratively SHA-512 hash the public Identity Key and user identi- fier 5200 times.

2. Take the first 30 bytes of the final hash output.

3. Split the 30-byte result into six 5-byte chunks.

4. Convert each 5-byte chunk into 5 digits by interpreting each 5-byte chunk as a big-endian unsigned integer and reducing it modulo 100000.

5. Concatenate the six groups of five digits into thirty digits.

TRANSPORT SECURITY

All communication between WhatsApp clients and WhatsApp servers is layered within a separate encrypted channel. On Windows phone, iphone, and Android, those end-to-end encryption capable clients use Noise pipes with Curve25519, AES-GCM, and SHA256 from the Noise Protocol Framework for long running interactive connections.

This provides clients with a few nice properties:

1 Extremely fast lightweight connection setup and resume.

2 Encrypts metadata to hide it from unauthorized network observers. No information about the connecting user's identity is revealed.

3. No client authentication secrets are stored on the server. Clients authenticate themselves using a Curve25519 key pair, so the server only stores a client's public authentication key. If the server's user database is ever compromised, no private authentication credentials will be revealed.

CONCLUSION

Messages between WhatsApp users are protected with an end- to-end encryption protocol so that third parties and WhatsApp cannot read them and so that the messages can only be decrypted by the recipient. Even if someone sniff the message he will get the encrypted message which he/she will not be able to decrypt. All types of WhatsApp messages (including chats, group chats, images, videos, voice messages and files) and WhatsApp calls are protected by end-to-end encryption.

WhatsApp servers do not have access to the private keys of WhatsApp users, and WhatsApp users have the option to verify keys in order to ensure the integrity of their communication.

The Signal protocol library used by WhatsApp is Open Source, available here: https://github.com/whispersystems/libsignal-protocol-java/

RESOURCES

https://github.com/mgp25/Chat-API/wiki/WhatsApp-Registration-Flow

https://github.com/whispersystems/libsignal-protocol-java/

https://whatsapp.com/security/

https://blog.whatsapp.com/

https://blog.whatsapp.com/10000618/end-to-end-encryption