WhatsApp

Security of End-To-End Encrypted Backups

WhatsApp Security Whitepaper

Version 1 Originally published September 10, 2021

Contents

Introduction

This whitepaper provides a technical explanation of WhatsApp's planned implementation of end-to-end encrypted backups, to be released later in 2021. For additional context and information on WhatsApp's end-to-end message encryption, please visit WhatsApp's website at www.whatsapp.com/security.

Since 2016, all personal messages, calls, video chats and media sent on WhatsApp have been end-to-end encrypted. This means that no one except the user, not even WhatsApp, can access them.

The content of message chats is valuable to WhatsApp users and WhatsApp offers an in-app backup feature to protect the content in the event a user's device is lost or stolen; and to enable the transfer of their chat history to a new device.

WhatsApp's backup management relies on mobile device cloud partners, such as Apple and Google, to store backups of the WhatsApp data (chat messages, photos, etc.) in Apple iCloud or Google Drive. Prior to the introduction of end-to-end encrypted backups, backups stored on Apple iCloud and Google Drive were not protected by WhatsApp's end-to-end encryption. Now we are offering the ability to secure your backups with end-to-end encryption before they are uploaded to these cloud services.

With the introduction of end-to-end encrypted backups, WhatsApp has created an HSM (Hardware Security Module) based Backup Key Vault to securely store per-user encryption keys for user backups in tamper-resistant storage, thus ensuring stronger security of users' message history.

With end-to-end encrypted backups enabled, before storing backups in the cloud, the client encrypts the chat messages and all the messaging data (i.e. text, photos, videos, etc) that is being backed up using a random key that's generated on the user's device.

The key to encrypt the backup is secured with a user-provided password. The password is unknown to WhatsApp, the user's mobile device cloud partners, or any third party. The key is stored in the HSM Backup Key Vault to allow the user to recover the key in the event the device is lost or stolen. The HSM Backup Key Vault is responsible for enforcing password verification attempts and rendering the key permanently inaccessible after a certain number of unsuccessful attempts to access it. These security measures provide protection against brute force attempts to retrieve the key.

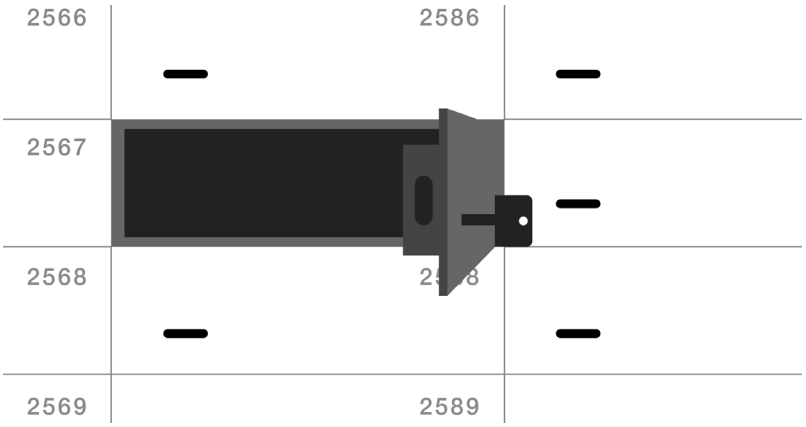
Additionally, the users have a choice to use a 64-digit encryption key instead of a password, which would require them to remember the encryption key themselves or store it manually as in this case the key is not sent to the HSM Backup Key Vault.

Currently, end-to-end encrypted backups are only supported on a user's primary device. In addition, we recommend that users who opt in to end-to-end encrypted backups also deselect WhatsApp from the apps that are

included in their device-level backups. We will inform users of the need to do this when they set up their end-to-end encrypted backup in WhatsApp.

HSM-based Backup Key Vault as a safe deposit box

WhatsApp's HSM-based Backup Key Vault can be compared to safe deposit boxes that are often offered by conventional banks.



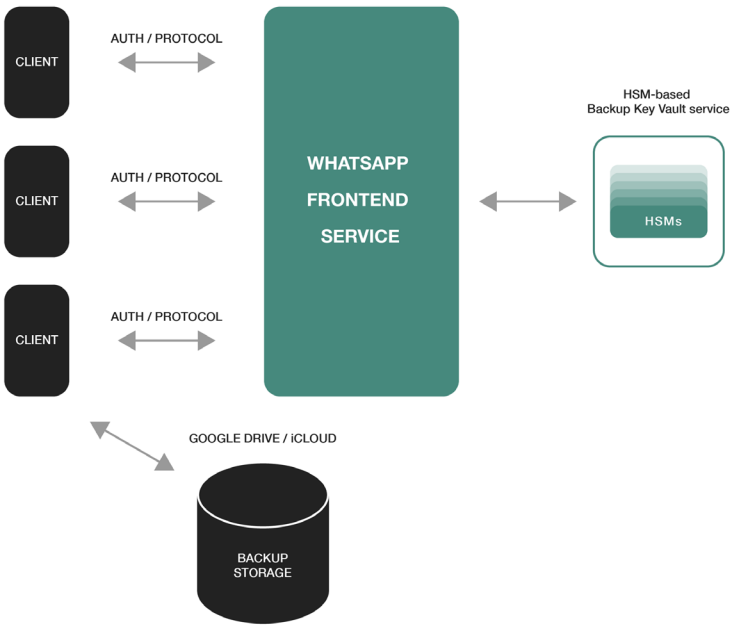
In a conventional bank, a user is given a key to their box with an assurance that even bank employees are not able to access the contents of the box.

WhatsApp's HSM-based Backup Key Vault provides a similar assurance, which is that only the user and nobody else, including Apple, Google, Facebook, or WhatsApp, is able to access a user's backed-up messages.

Such assurance enhances the security and privacy of the user's data. Such assurance is also similar to end-to-end encryption used in WhatsApp messaging: only the users that are engaged in a message exchange are able to access the contents of the messages, while WhatsApp infrastructure does not have access.

System Overview

Clients connect to WhatsApp via the WhatsApp front-end service. WhatsApp front-end service handles client connections, client-server authentication, as well as implements the protocol for uploading and downloading of encrypted backup keys.



Behind the front end is the HSM-based Backup Key Vault service. The purpose of that service is to provide highly-available secure storage of encrypted keys for end-to-end encrypted backups. The HSM-based Backup Key Vault service is geographically-distributed across multiple datacenters, allowing it to continue operating in the event of a datacenter outage.

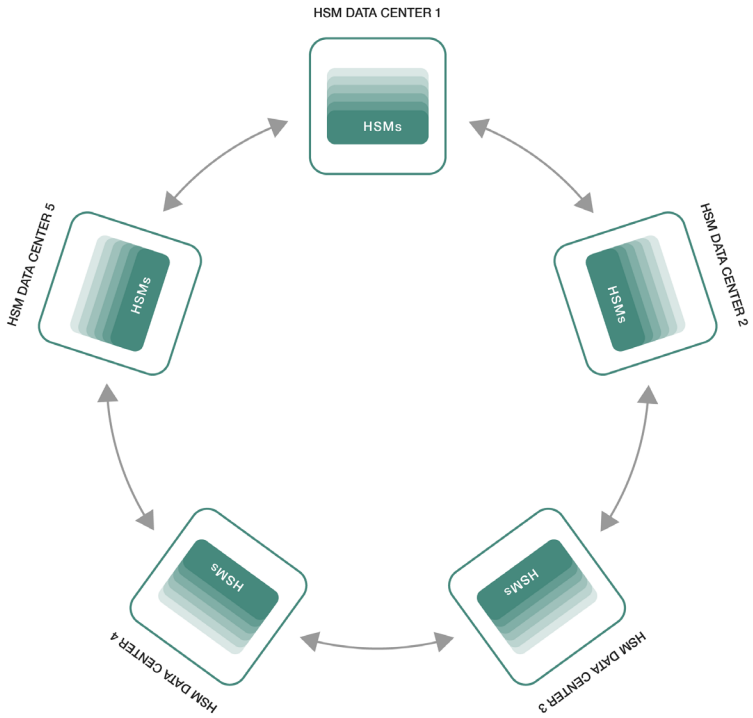
The backups themselves are generated on the client as data files which are encrypted using symmetric encryption with the locally generated key. After a backup is encrypted, it is stored in the third party storage (for example iCloud or Google Drive). Because the backups are encrypted with a key not known to Google or Apple, the cloud provider is incapable of reading them.

HSM-based Backup Key Vault resilience

HSM-based Backup Key Vault is designed to be a majority-consensus based system. That means that its data is replicated among a group of replicas and a simple majority of the replicas is sufficient to be online and to continue system operation. In other words, some replicas can fail, yet the failure is tolerable.

In order to tolerate two kinds of failures (for example when one member replica completely fails and one data center is temporarily disconnected due to a disaster) the HSM-based Backup Key Vault is deployed in five datacenter sites.

HSM-based Backup Key Vault is organized as a fleet or a collection of machines. The fleet is composed of islands or a set of identical replicas. Each island holds a portion of the data that the fleet is responsible for.



Security of client connection to ChatD

Client communication to Chatd for all general needs (including end-to-end encrypted backups) is layered within an encrypted channel. Clients use Noise Pipes from the Noise Protocol Framework for long-running interactive connections. For more information on transport security, see the WhatsApp Encryption Overview whitepaper.

The encrypted backups feature takes advantage of this secure channel to establish trusted connectivity to the server and given that secure channel, in the rest of this whitepaper, we focus only on attacks mounted from within WhatsApp's infrastructure.

Backup generation and backup key registration

Once the end-to-end encrypted backups feature is enabled in the WhatsApp application, the client uses a built-in cryptographically secure pseudorandom number generator to generate a unique encryption key K. The cryptographic strength of K in terms of length is 256 bits, i.e. 32 bytes. That key K is only known to the client and is never transmitted outside of the client unencrypted.

To decrypt the backup, the key K is needed. Thus, to safeguard K in the HSM-based Backup Key Vault, the client performs a registration of K with WhatsApp.

The registration process creates a record within the HSM-based Backup Key Vault associated with the particular client for the purpose of storing the backup encryption key K.

At the heart of the key K registration and an ability to later securely retrieve K is the OPAQUE protocol. The OPAQUE protocol allows the client and the HSM-based Backup Key Vault to establish a registration record inside the HSM-based Backup Key Vault. OPAQUE protocol allows the record to be secured with a password, without disclosing the actual password to the HSM-based Backup Key Vault.

This record can also later be used to retrieve K for the client, provided the client is able to produce evidence that it knows the password. This scheme ensures that only the user who knows the password is able to access the key K.

HSM-based Backup Key Vault validation

The first step the client performs when interacting with the HSM-based Backup Key Vault is Vault validation. The client relies on a public key that is associated with a specific HSM-based Backup Key Vault fleet, which is hardcoded into the client. This prevents man in the middle type of attacks.

Registration

The input to the registration process is a user password, that the user has to remember.

The registration process steps are:

1. WhatsApp application asks the user to enter the password.

2. The client utilizes the `OpaqueClientRegistrationStart()` function of the `OPAQUE` library to obtain byte buffer `RegistrationRequest` and sends it to the server. The response to `RegistrationRequest`

will be a message `RegistrationResponse`, a signature over `RegistrationResponse` by the HSM `RegistrationResponse_sig`, and a challenge `OPAQUE_C`. The challenge is a random number generated by the server to prevent replay attacks.

3. The client calls the `OpaqueClientRegistrationFinish()` OPAQUE api, the result of which is buffer `RegistrationUpload` and a `OPAQUE_K`, a symmetric key which is used to secure backup key K. The client encrypts key K using `AES-GCM` with `OPAQUE_K` and sends it along with the buffer `RegistrationUpload` to the server.

4. The registration is complete.

Key retrieval

For key retrieval and decryption, the client performs an OPAQUE login operation to derive the same symmetric key on both the client and the server.

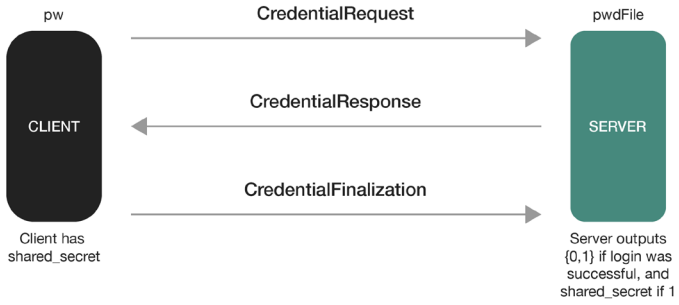
Then the client authenticates to the HSM-based Backup Key Vault via OPAQUE login process:

1. The client calls `OpaqueClientLoginStart()`, provides the password (pw in the diagram), and obtains a buffer `CredentialRequest`. The client sends `CredentialRequest` to the server, which in turn decrements the attempt count and sends back buffer `CredentialResponse` and HSM signature `CredentialResponse_sig`.

2. The client calls `OpaqueClientLoginFinish()` OPAQUE function and supplies `CredentialResponse`. The call returns a shared symmetric key `SHARED_K`, symmetric key `OPAQUE_K`, and a buffer `CredentialFinalization`.

3. The client sends `CredentialFinalization` to the server, which returns `EK` and `K_NONCE`. Both are encrypted with `SHARED_K` and `RK_NONCE`, which is a nonce for unwrapping them. `EK` is the encrypted backup key.

4. The client then unwraps `EK || K_NONCE` using `AES-GCM`.
5. Finally, the client unwraps EK with `OPAQUE_K` and to get `K: K = AES-GCM(data = EK, key = OPAQUE_K, nonce = K_NONCE)`.
6. Now the client has K and can decrypt the backup with it



Conclusion

In this document we discussed the mechanisms that are employed in WhatsApp's end-to-end encrypted backup solution and help keep users' backups secure.

Our goal with this project is to create a system to provide additional security for users' message history from WhatsApp, as well as the users' own cloud providers and any other third parties.

WhatsApp is committed to building technologies that safeguard the content of users' messages which includes launching end-to-end encryption at scale in 2016 and now advancing it further to protect messages at rest in the cloud with end-to-end encrypted backups.

Resources

1. OPAQUE publication: https://eprint.iacr.org/2018/163.pdf
2. OPAQUE protocol at IETF: https://tools.ietf.org/id/draft-krawczyk-cfrg-opaque-03.html
3. The Opaque key exchange protocol implementation in Rust : https://github.com/novifinancial/opaque-ke/
4. OPAQUE documentation : https://docs.rs/opaque-ke/