Twake security specifications

June 2019

Table of contents

About this document

This document present how Twake manage data security. Because we constantly improve our security algorithms, some specifications can be currently deployed in production or currently planned. This document will be updated and to know which version of Twake is described here, please refer to the “Twake version” section. Planned specifications will be implemented and available in production when the corresponding version will be released. Do not hesitate to contact us for details about our releases dates at contact@twakeapp.com .
To download the latest version of this document, please go to twakeapp.com . For any other information, please contact us at contact@twakeapp.com .
This document will be updated frequently as we improve Twake security.

This document belongs to Twake Technologies SAS, SIRET 838962637 00016, company situated 193 Avenue Paul Muller, 54602 Villers-lès-Nancy in France.

Current Twake release

This document describe Twake Albatros **1.2.000** released on June 2019.

Tiers softwares and Twake versions

Twake is available in two main versions :
- **SaaS** : Twake as a service (SaaS available on twakeapp.com)
- **On-Premise** : Twake on-premise (on private servers or on dedicated servers managed by Twake)

The Twake SaaS version is hosted on AWS (Amazon Web Services) EC2 (Elastic Compute Cloud) and use other services by AWS like Amazon S3 for files storage. All servers are located in **Paris (UE)**.

The Twake On-Premise version can be deployed on any local server and can work fully disconnected from the Internet or with an unique and unidirectional connection to twakeapp.com for licence management, mailing and iOS and Android push notifications (that are not available by default in fully disconnected mode).

On SaaS version, Twake provide two main tiers softwares :
- ONLYOFFICE spreadsheet, ONLYOFFICE document and ONLYOFFICE presentation, https://onlyoffice.com
- Jitsi video-conferencing, https://jitsi.org/

On the On-Premise version those softwares are optionals. In this document we will describe how Twake interact with those softwares.

Twake security specifications in 7 questions

Where are your servers ?
Twake servers are all in France, in Paris. We use both OVH and AWS servers.

Are you GRPD compliant ?
Yes ! As you can see in this document, we do everything we can do to protect your personal and professional data. Concerning your password and personal data specifically, passwords are

hashed using PBKDF2 and your data is encrypted before to be sent to the database if it is compromised.

What protocol do you use ?
We use **HTTPS** for HTTP communication and **WSS** for realtime communication. Passwords are hashed with **PBKDF2.** Files are encrypted with **OpenSSL AES-256-CBC** with a key composed of three parts stored in database, in code and in the server configuration. Our WebSocket communication has an extra layer of security using **AES front-end encryption** with rotating random keys. And finally our database data is encrypted with **OpenSSL AES-256-CBC**.

How do you manage backups ?
We backup our encrypted database and your encrypted documents in Amazon S3 Glacier. We keep a backup of each month and a backup of the five three last days.

What if I want to export all my data ?
We do not provide an export feature on the app for now, but we have the ability to export your data on-demand. We can export your data in 2 business day depending on how you used Twake and how much data you need to export (files, calendars, metadata etc). We don't have direct access to your files, but you will be able to download them using the files identifiers generated in the export file and your Twake account.

How do you manage security with tiers applications and modules ?
We cannot guarantee the security of the data you send to tiers applications and modules. If you work with an external app like Zapier or Giphy, they will have access to some of your information. You can see the access scope of an application before to add it to your company.

How can I contact you ?
You can contact us on contact@twakeapp.com.

General architecture

Apps types

Twake is a collaborative platform which provide different tools and applications. We call Core Apps the apps developed and managed by Twake and External Apps the apps we integrated to our platform. External Apps can be managed by Twake and hosted on Twake servers or managed and hosted by tiers.
The core apps are : Drive (file storage), Messages (messaging system), Tasks (kanban and project management), Calendar (shared calendars), PDF viewer, Image viewer.
The external apps managed and hosted by Twake are : ONLYOFFICE office suite, Calls (video-conferencing), Giphy module (this one uses Giphy APIs).
The external apps managed and hosted by tiers are any apps integrated using an URL (for instance Google Analytics integration etc.) .

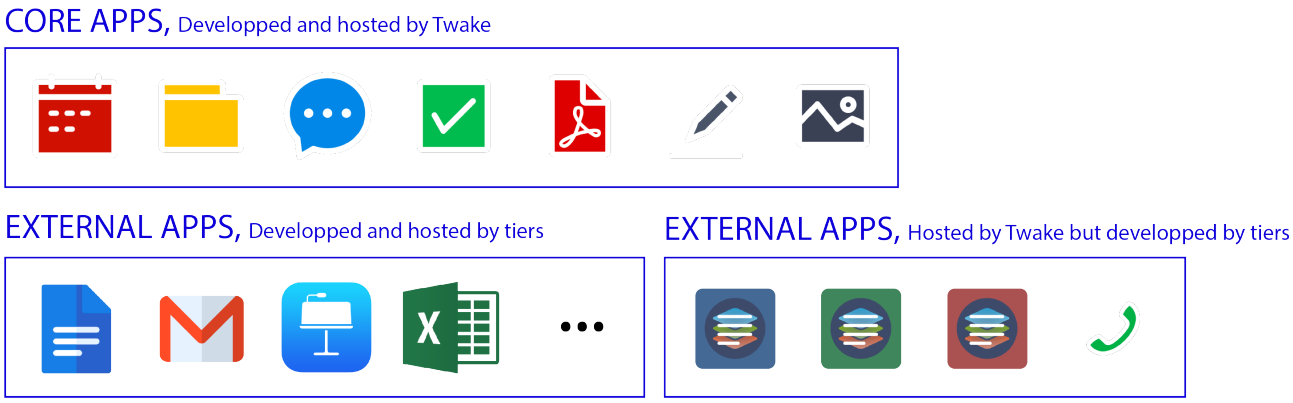

Figure 1. Types of applications available on Twake

General architecture

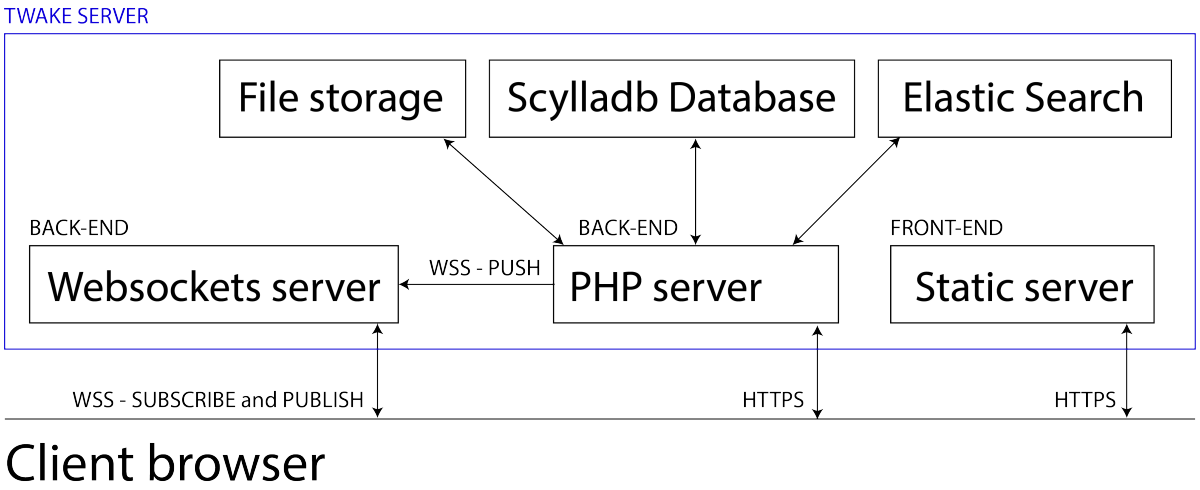

Figure 2. Twake general architecture

Twake contain multiple services, client to Twake connections use HTTPS protocol for uni-directional requests and WSS protocol for real-time bi-directional communication. We use SSL/TLS for data exchange in both protocols.

The “PHP server” is the only one allowed to communicate with internal services like File storage, Database, Elastic Search or external services like ONLYOFFICE.

On-premise fully disconnected and semi-disconnected modes

We offer two modes of on-premise installation for Twake on private servers managed client side.
- A fully disconnected mode that allows the client to use Twake in a local network disconnected from the Internet ;
- A semi-disconnected mode in which on-premise server will communicate with Twake servers to share usage data, send mails and send push notifications for smartphones and tablets.

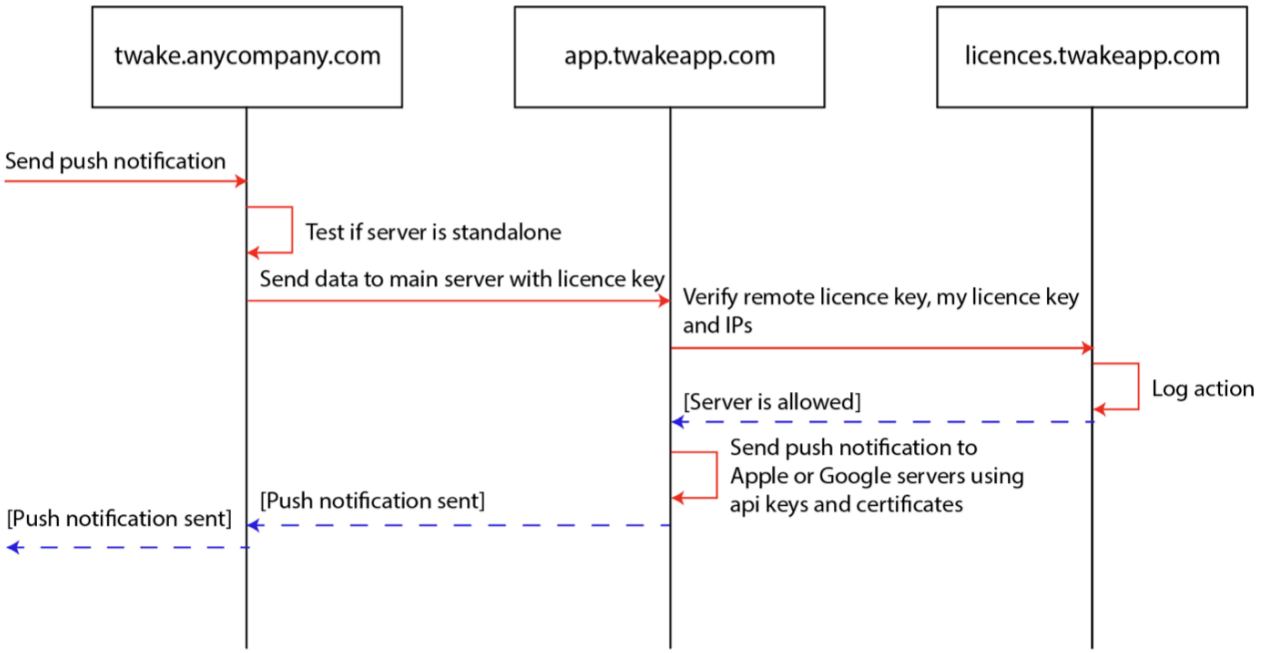

Figure 3. On-premise semi-disconnected mode for push notifications

The semi-disconnected mode avoid clients to configure mails servers and push notifications. The client can also use the mobile app provided by Twake on Google Play Store or App Store with push notifications implemented like shown in figure 3.

For more secure contexts, Twake can be installed without any connections to the outside. In this case, in order to receive Twake emails the client must configure a mail server to use.

Documents storage

General

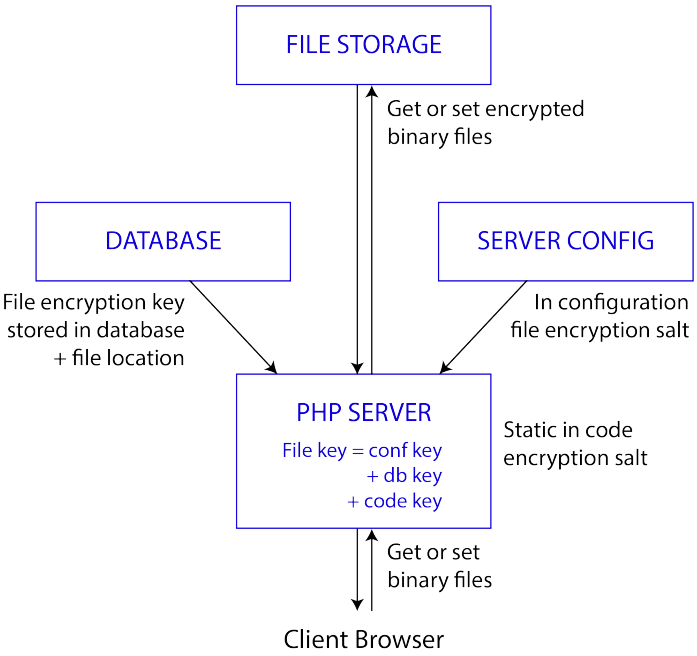

Figure 4. Encryption by Twake before storage

Twake store encrypted binary files in a document repository, in the SaaS version we use Amazon S3 file storage to store our encrypted files. In the On-Premise version, files are stored in a directory not accessible from the Internet (behind the web root directory).

Documents uploaded to Twake Drive are encrypted using AES256 standard with a key built using three strings (see Figure 3) :
- A static in code string salt ;
- A configured salt defined during Twake installation ;
- A random key generated for each document and stored in database.

The three strings are needed to decrypt binary documents stored in Twake Drive or in Messages attachments.
PHP Server verify client rights before each unencryption of documents.

Twake don't use this encryption system for the following files : users avatars, groups logos, workspaces logos, and workspaces wallpapers. Those files are stored unencrypted and are available over simple GET request without access rights verifications.

SaaS version details

With Amazon S3, we store the file encryption key in our database and we generate this key ourselves but the encryption job is executed on the Amazon side.

On-Premise version details

For now we only provide classic storage of documents on the Twake server disk. In the future it will be allowed to use WebDav / SSH or FTP storage, but in each case we will provide encryption on Twake PHP server side.

Web Sockets security

Web Sockets are used for realtime collaboration. It is used in Messages for realtime messaging, but also in Drive, Calendars and Tasks. We use PubSub architecture, it means that a client can *Subscribe* to a room and then will receive any data *Published* by other clients in this same room. The PHP Server can itself send data to this room, we call this action a *Push*.
This architecture don't provide any security by default and if a non-allowed client want to see what's happening in a room (for instance, a messaging stream), it only need the room name.
Connection to the Web Sockets Server is done over SSL/TLS Web Sockets protocol.

Extra layer of security with front-end encryption

To improve our performances, we use an extra layer of security to keep shared data secure.

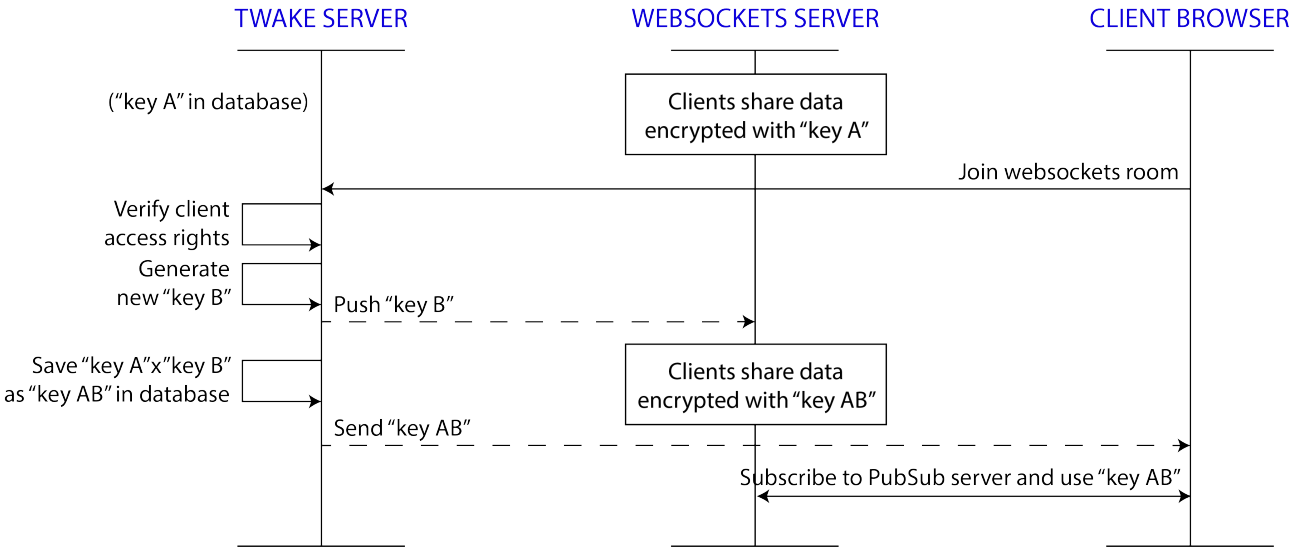

Figure 5. Future Web Sockets security layer

In order to keep Web Sockets data exchanges secure keeping the Web Sockets server the most simple possible, we exchange encrypted data only with an evolutive encryption key. Only the PHP Server verify the client access rights and generate a new key for everyone. Because the complete key is never sent over the Web Sockets Server, the only way to get a correct encryption key is to ask the PHP Server.

The figure 4 shows how we secure communication between clients in a Web Sockets room. Here one or more clients are already sharing data in realtime encrypted end to end with the "key A". At this time only those clients and the PHP Server know the "key A". When a new client want to join the Web Sockets room, he need to be authenticated beside the PHP Server. The PHP Server generate a random new "key B" and send this key to all users already connected to the room. Clients will not use this "key B" but a mix between "key A" and "key B" called "key AB". Each clients make this operation and start sharing data using this new "key AB". The PHP Server do the same operation and send the result to the new client waiting to join the Web Sockets room.
If a malicious client is listening to the protected room, he will never be able to decrypt data.

Tiers softwares security

JITSI

We use Jitsi for video-conferencing. Jitsi is a very simple software that take the url components to generate the video-conference room unique name. Each conference generated in Twake is basically a long token used as room unique name and is stored nowhere else than in our database.

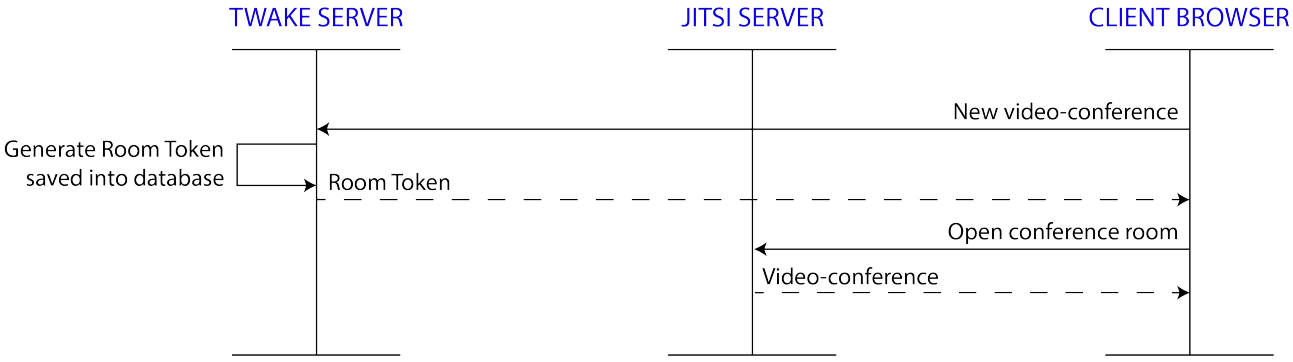

Figure 6. New Jitsi video-conference

The figure 5 describe how a conference room is generated. When a new conference call is created, the PHP Server generate a random token used as room unique name and save this token into database. Each user who want to access the conference room need the room unique name given by this same PHP Server after access rights verifications.

In SaaS version, the Jitsi servers are managed by Twake only. In On-Premise version, Jitsi can be disabled if not used, or it can be installed locally, or using Jitsi servers directly.

ONLYOFFICE

The figure 6 describe how a document is opened in ONLYOFFICE for realtime edition. We use ONLYOFFICE office suite for realtime collaboration and edition of Microsoft Office and Open Office documents. ONLYOFFICE provide an API used to communicate with our document storage server via our PHP Server.

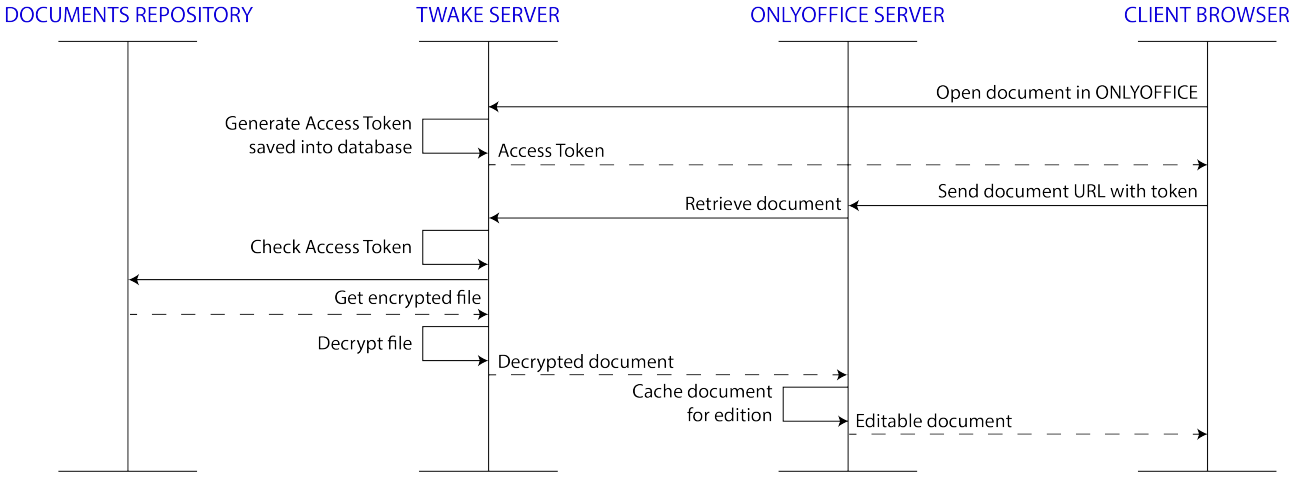

Figure 7. Opening a file in ONLYOFFICE

When an user want to open an Office document, a request to the PHP Server is made. This server verify access rights and generate a new random token. This token is sent to the user who can build a new url used by ONLYOFFICE to retrieve the document. This url make a request to the PHP Server which verify the token given in this same url and returns the requested document to ONLYOFFICE. ONLYOFFICE put this file in cache for realtime edition and destroy it once edition done. The same process happen when the document is saved.

In SaaS version, the ONLYOFFICE servers are managed by Twake only. In On-Premise version, ONLYOFFICE can be disabled if not used, or it can be installed locally, or using ONLYOFFICE SaaS offers.

Database encryption

Data are stored in **ScyllaDB encrypted databases** and searchable entities will be stored in **Elastic Search but only identifiers** will be available over ElasticSearch.
In addition of the security specifications discussed in this section, please refer to the General Architecture section to see how servers are or are not available from the public Internet network.

ScyllaDB

Any data (except non-sensible data like identifiers, dates and counters) stored in ScyllaDB is encrypted using OpenSSL AES-256-CBC.

Indexed search storage

We provide advanced search features to our product Twake using Elastic Search to index our entities. Entities are not stored in Elastic Search and only indexes will be available if Twake Elastic Search servers are compromised.

Miscellaneous

Random token generation

For random generation we use the PHP built-in *random_bytes* function with *bin2hex* if needed.

Passwords storage and authentication

Passwords are not stored in clear, only a hash of it is stored in database. We use the *hash_pbkdf2* method with a randomly generated salt for each user.
We use a custom authentication built over FOS User Bundle for authentication and session management.

SQL Injection

Each database access use Doctrine ORM sanitisation before write or reads, we never use custom access to the database to reduce risks of injection. We developed a middleware to use ScyllaDB databases but this middleware works after Doctrine ORM sanitisation.

XSS

We use React as front-end framework and we never execute tiers Javascript. External apps are loaded in webviews or iframes and use a bridge designed like a router for data exchange between iframes.

Contact

If you have any question or request, please contact us on **contact@twakeapp.com**.