PassHub

PassHub Security Explained

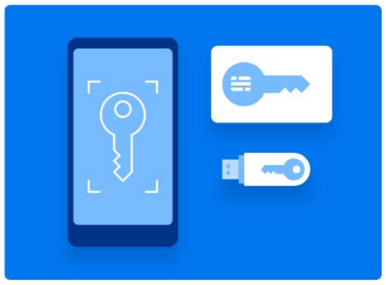
TECHNICAL WHITE PAPER

July 18, 2018

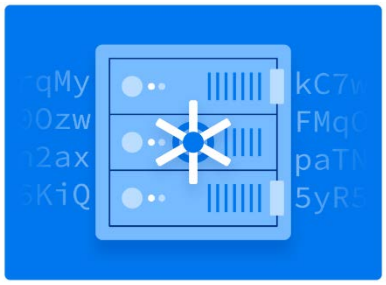
Table of Contents

Introduction

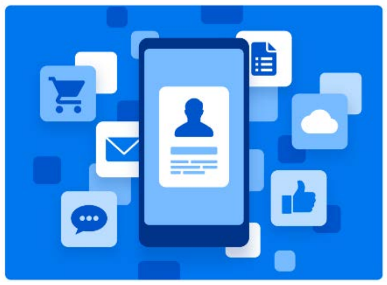
PassHub is a highly available web-based password manager powered by WWPass encryption technology, which allows users and ad-hoc groups to share credentials, notes and files in a secure, highly-controlled way.



PassHub features Client-side encryption (based on WWPass service and hardware devices or smartphone apps), which provides higher security than password managers based on user login/password pairs.



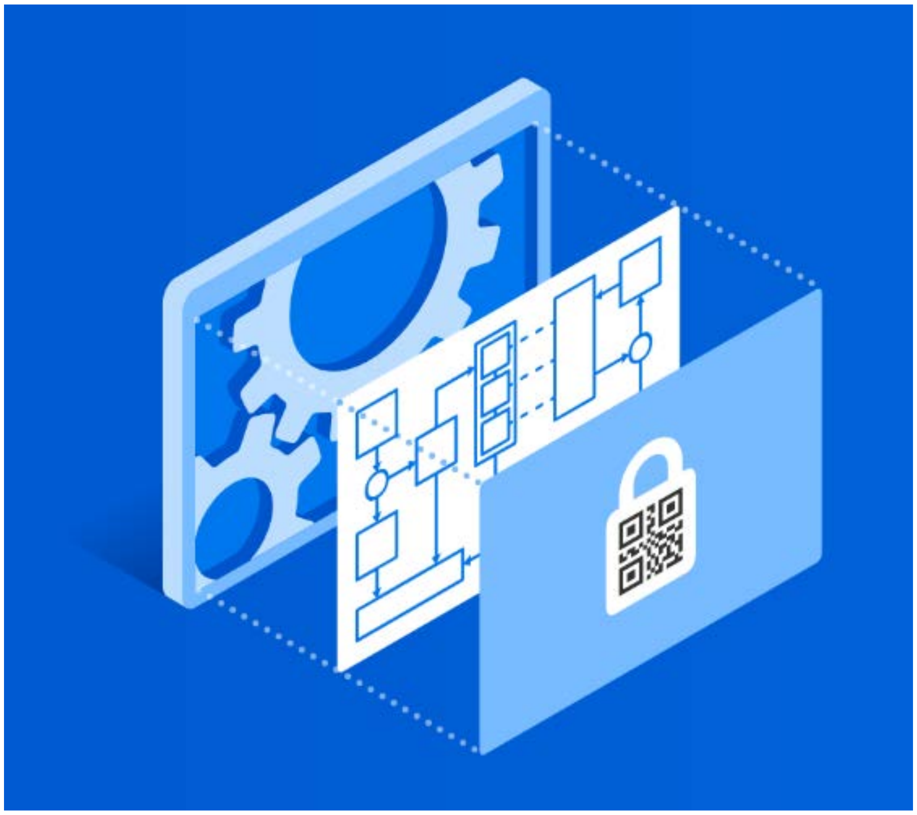
Encryption keys are bound to cryptographic user authenticators (hardware or software) and hence do not depend on particular browsers or devices. PassHub can be accessed on any device, desktop or mobile.



Sensitive user data is encrypted directly in user devices and is not known to the PassHub.net server, so WWPass and third parties cannot read sensitive data, and all information is only accessible by the intended user or recipient.

WWPass Service

To achieve its exceptional security features, PassHub employs WWPass service to authenticate users and to provide personal encryption keys.



Users get their own “authenticator” in a form of hardware token or smartphone app – PassKey. In fact, the PassKey is more than just authenticator – it is also used for local cryptographic operations, crucial for client-side encryption.

Smartphone

Smartcard

Smartstick

You can read more on WWPass technology on WWPass site:
https://www.wwpass.com/pdf/docs/HowWWPassAuthenticationWorks.pdf

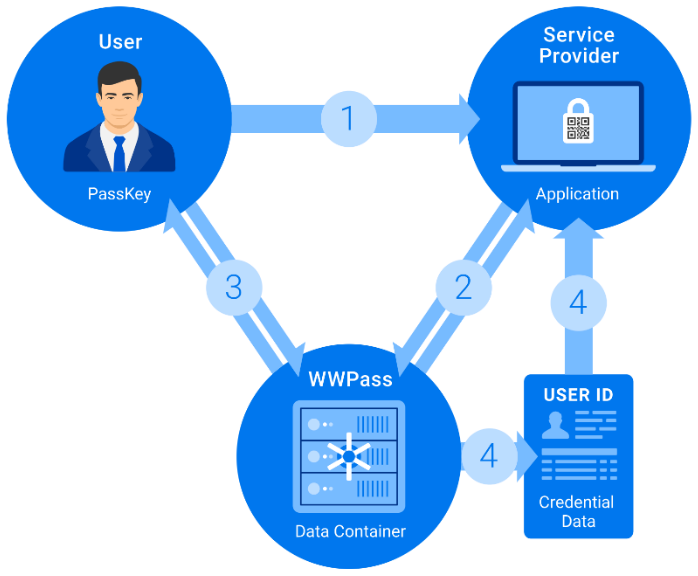
Strong User Authentication to PassHub

Unlike many other solutions for user data encryption, PassHub does not use usernames and passwords to allow users to access their PassHub accounts.

A traditional approach is to attach a user to their account by a username or e-mail address (usually known to the outside world) and then use a password to not only authenticate the user, but to also derive a user encryption key. It is a well-known fact that passwords are often reused in many accounts (one of the weakest sides of login/password technology), thus making it possible to guess user credentials.

In PassHub, the WWPass PassKey app authenticates the user via a trusted device. This is far more secure than the traditional approach, but can be configured to be even more secure with an additional factor – like a PIN or biometric scan. The same trusted device is used to handle user encryption keys, but this time the encryption keys are totally separated from authentication parameters of the PassKey.

How it works



1. The user presents a PassKey to log in to your app or website.
2. WWPass mutually authenticates both your app or website and the user's PassKey, and obtains their unique identifiers, which are used to create a pointer to a set of application-specific data containers. These data containers are used to store the user's identity and sensitive application-specific information.
3. WWPass sends the information stored in the data containers to your app or website via a secure channel.
4. Your app or website verifies the information received and logs in the user. The user's identity is unintelligible to attackers, and is never disclosed to WWPass itself.

If a PassKey is lost or stolen, it can be disabled and restored through our recovery process to keep all linked accounts intact.

The PassKey is a one-to-many authenticator. The same PassKey may be used to authenticate the user on many sites. Unlike the OAuth protocol, WWPass fully isolates user data of different Service Providers -a capability known as "directed identity."

Client-Side Encryption

PassHub features client-side encryption architecture: user data, such as credentials, files, notes, etc., are encrypted in the browser on the user desktop or mobile device. The PassHub server does not have access to any secret information. Here is an example of a record stored in a PassHub database on the server:

```
{
    "_id" : ObjectId("5a9a1b20dab32d06ec49081c"),
    "SafeID" : "5a9a1b20dab32d06ec490816",
    "iv" : "XAaS59w8IgbsZOvXK+Vk8Q==",
    "data" : "0nfky8UsDOsa4w==",
    "tag" : "SFvqeiWESWrjmT2LErFi/Q==",
    "folder" : "5a9a1b20dab32d06ec490818",
    "lastModified" : "2018-03-03T03:48:48+00:00",
    "version" : 3
}
```

When a user opens the PassHub page, the data is decrypted in the user's browser and then presented directly to the user. The PassHub server never sees this data in a decrypted form and, by design, no one but the user can view this decrypted data.

Client-side encryption implies the existence of cryptographic key in a user device (in a browser or in an application). This key is constructed by PassKey and delivered to the browser. Since a PassKey is a one-to-many authenticator (the same PassKey may be used on different web sites and services, with full isolation between them) the client-side encryption key is unique for every user-to-service provider pair and is not known to WWPass itself.

More on Client Encryption Key, K_c

This client-side encryption feature leverages a unique WWPass service. During personalization, each PassKey generates and stores a secret symmetric key, which never leaves the PassKey itself. WWPass authentication scenarios may include a "client-side key" request. When a PassKey authenticates to the WWPass network, the latter sends a unique service provider ID (SPID) to the PassKey. With its internal secret key, the PassKey generates a specific "client-side" encryption key using the SPID as a key material. This way every user-to-service provider pair gets its independent encryption key K_c on the client side. Another challenge is to send the key to the browser (the QR code is a one-way transport). To send the encryption key in an opposite direction, we use a two-hop channel between a PassKey and WWPass and then, from WWPass to the browser via web socket. The client-side encryption key is ciphered so WWPass does not have it in clear text.

End-to-End Encryption for Data Sharing

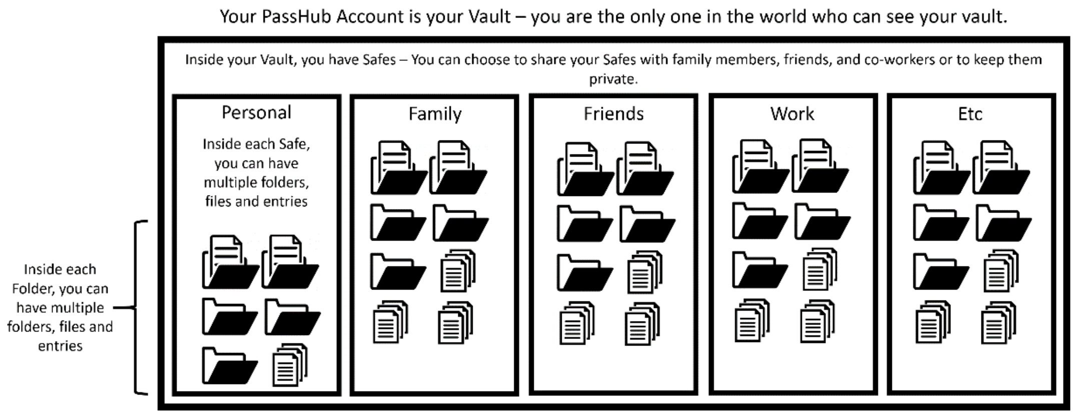
For a data sharing, PassHub uses multi-tier cryptographic key architecture.

User Asymmetric Key Pair { K_{pub}, K_{pr} }

When a user account is created in PassHub, an asymmetric RSA key pair is generated in the user browser and is sent to the server, with its private key encrypted with the client-side symmetric key: { K_{pub}, Enc(K_c ,K_{pr})) } Thus, the user private key is not known to the server.

Encryption Key of a Safe, K_s

User data is stored in groups called "safes." Each safe may contain credential records, notes and files, possibly placed in folders. Safes define sharing granularity: you can only share a whole safe, not a folder or individual record.


Your PassHub Account is your Vault – you are the only one in the world who can see your vault.
Inside your Vault, you have Safes – You can choose to share your Safes with family members, friends, and co-workers or to keep them private.
Personal
Inside each Safe, you can have multiple folders, files and entries
Family
Friends
Work
Etc
Inside each Folder, you can have multiple folders, files and entries

Each safe has its own symmetric encryption key. Every record in a safe is ciphered with the encryption key of this safe. File encryption includes another level of indirection: they are encrypted with individual keys, which, in turn, are encrypted with the key of the safe. This allows file moving from one safe to another without re-encrypting the file body. This safe encryption key is generated in user browser when the user creates a new safe. The key is encrypted with a user public key before sending it to the server: Enc(K_{pub} ,K_s). Thus, the safe encryption key Ks is not known to the PassHub server.

When user reads the content of a safe, all items of the safe are downloaded to the browser in encrypted form. Along with this data, the browser also receives an encrypted key of this safe Enc(K_{pub} ,K_s) and encrypted user private key Enc(K_c, K_{pr}). Now, the browser first decrypts the user private key with the client-side key Dec(K_c ,Enc(K_c, K_{pr})) and the safe key: Dec(K_{pr}, Enc(K_{pub} ,K_s)).

Sharing Access to a Safe

When an owner of a safe provides access to another user (recipient) the owner's browser downloads the owner's encrypted private key, the key of a safe and the recipient's public key. After decrypting the key of the safe, this key is re-encrypted with recipient's public key and sent to the PassHub server. The latter creates a new access record binding the safe ID, the recipient's ID, and the key of the safe, encrypted with the recipient public key. Now, the recipient is able to decrypt all the records and files of the safe.

High Service Availability

For a password manager as a public service, 24x7 availability is crucial. PassHub.net is deployed on 3 geographically separated Ubuntu servers. Two of them are identical and run Nginx HTTP servers, a PHP 7+ engine and MongoDB database in replica configuration. The third server has no public access and serves as a MongoDB arbiter, required by replica architecture.

Apart from its own availability, the PassHub relies on the availability of WWPass service. The WWPass network was specifically designed to be highly redundant and available. Since it is distributed over the globe with Reed-Solomon redundancy six-out-of-twelve data nodes, the WWPass network is a unique service with all components deployed in multiple instances, thus providing no single point of failure.

For a dedicated/corporate on-premises implementation, the most basic PassHub configuration requires a single Ubuntu server with Nginx HTTP server, PHP 7+ and MongoDB database.

Conclusion

PassHub is a highly available, web-based, zero knowledge password manager powered by WWPass encryption technology and featuring client-side encryption providing higher security than other password managers. Encryption keys are bound to cryptographic user authenticators (hardware or software) and therefore do not depend on particular browsers or devices. Therefore, PassHub can be accessed on any device, desktop or mobile. Because of the zero-knowledge architecture, the only person who can access data stored in PassHub is the person who holds the correct user authenticator.