Enpass Security Whitepaper

Version 1.4

Updated: *Aug 22, 2025*

Topics

Security Principles

Vault

Sync

Enpass for Business

Key Features

Security Principles

Every design choice in Enpass follows core security principles that put the user's data protection and control first:

1. **Data Sovereignty ("Zero Possession")**

A fundamental design principle of Enpass is giving you full ownership of your data. Enpass never stores your credential vaults on its own infrastructure. Instead, your encrypted vault resides locally on your devices and, if you choose to sync, on your selected cloud storage or network drive. This also minimizes the attack surface as there is no central repository holding thousands of vaults, and thus no single point of failure.

2. **Use of Proven Cryptography**

Enpass exclusively relies on peer-reviewed, industry-standard cryptographic algorithms and implementations. We do not develop proprietary or untested cryptography.

3. **Zero Knowledge architecture**

Enpass is designed with a Zero-Knowledge architecture, meaning only you can access your vault data. All cryptographic operations occur locally on your device, and any data that leaves your device is already encrypted with a key that only you possess. Enpass (the company) has no possession of your vault and no access to the secrets that can unlock it, and therefore cannot decrypt your data.

Vault

At the heart of Enpass is the **Vault** – a set of files where all your credentials and attachments are stored in encrypted form. It consists of the following files:

1. **Metadata (vault.json)**

A small metadata file in plaintext, containing non-sensitive info required for synchronization. No passwords or sensitive fields are stored in this file.

2. **Credential Database (vault.enpassdb)**

This is the primary file in which all credentials are stored. It is a SQLite database encrypted using the open-source extension SQLCipher. SQLCipher ensures that 100% of its contents remain cryptographically protected at rest using AES-256 algorithm. The encryption key, also known as the Vault Key, is derived from your master password and keyfile.

3. **Attachments (*.enpassattach)**

If you attach files to your Enpass items, any attachment larger than a small threshold (1 KB) is stored in a separate encrypted file. Each such attachment file is itself a SQLite database encrypted using the SQLCipher extension. These files are encrypted by a randomly generated key that is stored within the main vault (encrypted under the vault key), so when you unlock your vault, Enpass can retrieve the attachment keys to access attachments. This design means attachments can be synced or handled individually and still remain secure, and large files don't bloat the main database. If an attacker doesn't have your master password, they also can't access the attachment decryption keys, so attachments are as safe as the rest of the vault.

Vault Key

The Vault Key is the cryptographic key used to perform AES encryption and decryption of your data. It is derived from your Master Password (and Keyfile, if used).

Master Password (Something You Know)

The master password is the primary secret that encrypts your data. It is chosen by you and known only to you. Enpass never stores or transmits your master password in plaintext or any reversible form. We strongly advise choosing a master password that is both memorable *and* strong.

Forgot your Master Password?

There is no “backdoor” or recovery method to retrieve your data if you forget your Master Password or lose your Keyfile. Enpass cannot reset it for you, as we never have any record of it. However, if you are using **Enpass for Business**, organizations can enable the Access Recovery feature. In this model, vault keys are encrypted with an organization-wide recovery public key and stored on the Enpass Hub, allowing an admin to assist in securely resetting a forgotten Master Password.

Keyfile (Something you have)

Enpass supports augmenting your Master Password with a Keyfile—essentially a secondary secret stored as a file. This functions as a form of two-factor protection: deriving the Vault Key requires both knowledge of the Master Password and possession of the Keyfile. The Keyfile consists of a 32-byte value generated using the system’s CSPRNG. This introduces high-entropy, machine-generated randomness in key derivation that is entirely independent of user-chosen master password. As a result, an attacker would not only need to guess or brute-force the Master Password, but would also need to obtain the Keyfile itself in order to decrypt the vault.

Vault Key derivation

The Vault Key is derived from your Master Password (+ Keyfile, if used). Vault Key derivation is performed using the PBKDF2-HMAC-SHA512 algorithm with 320,000 iterations, deliberately introducing computational cost to significantly slow down brute-force attacks.

How much slower is better?

PBKDF2-HMAC-SHA512 with 320,000 iterations represents a carefully chosen balance—sufficiently strong to withstand brute-force attacks while remaining performant across all supported devices. Rather than arbitrarily increasing the iteration count, the recommended approach is to use a Keyfile in conjunction with the Master Password, effectively rendering brute-force attempts computationally infeasible.

Vault encryption

Enpass vault is a SQLite database encrypted using SQLCipher. SQLCipher is an open source extension to SQLite that provides transparent 256-bit AES encryption of database files. One can find complete design details of SQLCipher here.

How SQLCipher is configured in Enpass

1. The encryption algorithm is 256-bit AES in CBC mode.
2. Enpass derives key data from your master password (and keyfile if used) using 320,000 rounds of PBKDF2-HMAC-SHA512 and use it as a raw key for SQLCipher. Each database is initialized with a unique random salt in the first 16 bytes of the file. This salt is used for key derivation and ensures that even if two databases are created using the same password, they will not have the same encryption key.
3. SQLCipher does not implement its own encryption. Instead, it uses the widely available encryption libraries. In case of Enpass, widely trusted OpenSSL libcrypto is being used as encryption provider.

Sync

Any modern password manager would be incomplete without the ability to synchronize data across devices. Synchronization typically involves storing data on a centralized server—usually the vendor's. A core design principle of Enpass is Data Sovereignty – to give you full ownership of your data. Hence, Enpass does not store your Vaults on its own servers. Enpass offers a unique approach: connect your existing cloud or network storage and use it as sync fabric. This delivers seamless cross-device sync without compromising control of your data.

Bring your own sync fabric

Enpass supports a wide variety of cloud and local storage options:

- Microsoft OneDrive / SharePoint
- Google Drive
- Dropbox
- iCloud
- NextCloud
- WebDAV
- Folder sync over a network drive
- Local Wi-Fi–only peer-to-peer sync

Security at rest

All cryptographic operations happen **strictly on your device**. The cloud (or network drive) is used only as a storage medium.

- Your data never leaves your device in unencrypted form.
- Your cloud storage always contains the same encrypted vault file that resides on your device.
- Even if your cloud provider is compromised or malicious, your vault is useless to them without your keys.

In short, your cloud acts as a **dumb drive**. This extends Enpass's **zero-knowledge** stance to third-party clouds as well i.e cloud storage providers have *zero knowledge* of your secrets.

Transport security

When Enpass communicates with cloud storage services to sync your data, it uses the **strongest security mechanism** supported by the provider, typically HTTPS/TLS connections. This means your data in transit is **double protected**:

- Protected by TLS against interception.
- Encrypted end-to-end by Enpass (AES-256), so even if TLS were compromised, your data remains secure.

Enpass always uses official APIs and OAuth authentication flows. We never see or handle your cloud password. Instead, you grant access through the provider's login screen, which issues a scoped access token. This token never reaches Enpass servers—it is stored securely only on your device.

Sync operation

Enpass uses a client-side merge sync algorithm:

1. When you make a change on Device A, Enpass updates the encrypted vault file and uploads it.
2. Device B downloads the updated encrypted vault, open it locally, merges changes and upload encrypted vault if there are local changes.
3. Conflict resolution and merging happen entirely on your device.

Enpass for Business

Subtopics : [Access Recovery] [Vault Sharing]

Enpass for Business operates on the same principles as consumer Enpass i.e. client-side encryption and data sovereignty. Vault data is always stored as encrypted vault files, either on users' devices or on business-controlled storage (e.g., Microsoft 365 OneDrive, SharePoint, or Google Drive). However, organizations often require additional capabilities—such as master password recovery and vault sharing—which are enabled by a supplementary server component called **Enpass Hub**.

The overall system of Enpass for Business consists of the following components:

- **Enpass Application** – Runs on each user's device (PC, phone, etc.) and continues to handle all encryption and decryption locally. The app communicates with Enpass servers for features such as vault sharing, recovery, and audit.
- **Enpass License Server** – This is an Enpass-operated server that manages subscriptions, and organization accounts. It authenticates users, brokers initial connections to Enpass Hub and store organization policies. It does not store any of the vault data. Conceptually, it acts as the registry of "who is a valid Enpass user in Org X" and "what business features are enabled."
- **Enpass Hub** – A server component hosted by Enpass (or self-hosted by the organization). It enables business features such as Master Password Recovery, Vault Sharing, and Security Audit. Enpass Hub stores only **encrypted vault keys and metadata**, never vault content.
- **Enpass Admin Console** – A web portal for organization admins to manage accounts. It runs as a static client-side app in the admin's browser and communicates with the License Server and Hub over secured REST APIs.
- **Business Cloud Storage** – Even when Hub is used, vault file data remains in the organization's chosen storage platform (e.g., OneDrive/SharePoint, Google Drive). For shared vaults, the encrypted database resides in business storage, while the corresponding **vault key—encrypted with recipient public keys—is stored on the Hub**. This separation ensures that possession of the vault file alone is insufficient without cryptographic authorization via the Hub.

Enpass Hub

As organizations began adopting Enpass, business-specific features were introduced to facilitate secure team vault sharing, access recovery and administrative oversight. To enable these, Enpass added an additional server component called **Enpass Hub**, which provides business functions that require a central server to store *some encrypted metadata* (but never the actual vault data). Key business features include:

- **Access Recovery** – This feature enables users to reset their master password. Vault keys are encrypted with an organization-wide master recovery public key (part of a PKI setup) and stored on the Hub. When recovery is requested, a recovery admin verifies and authorizes the request, then sends a recovery link to the user.
- **Vault Sharing** – This feature allows seamless sharing of vaults without manually passing vault keys to recipients. Vaults data files are shared via underlaying cloud storage APIs (e.g. Microsoft Graph API), and the vault key for the corresponding vault is shared via Enpass Hub server. This vault key is encrypted with the recipient’s public key and stored on the Enpass Hub. This ensures that only the intended recipients, holding the corresponding private keys, can decrypt and access the vault.
- **Security Audit** – Admins can view aggregated password health reports for the company via the Enpass Admin Console.
- **Event Logs** – Admins can monitor and audit events related to user activities across key areas, including accounts, vaults, password recovery, and sharing.

These features use time-tested public-key-cryptography primitives (RSA-3072 and OAEP padding scheme) for exchanging vault keys, executed entirely on the client side. The Enpass Hub acts as a PKI directory and a secure repository for encrypted blobs and related metadata, enforcing access permissions while remaining blind to keys and data. **No encryption, decryption, or key generation takes place on the Enpass Hub.** All encryption and decryption occur within the Enpass apps.

Using a self hosted Enpass Hub?

If you are opting for self-hosting Enpass Hub, you must first integrate it with your organization’s Enpass account to make its features available to users. It involves storing URL of your Hosted Enpass Hub and a Secret key that will act as a trust factor between Enpass server and Enpass Hub. Below are the security specific configurations:

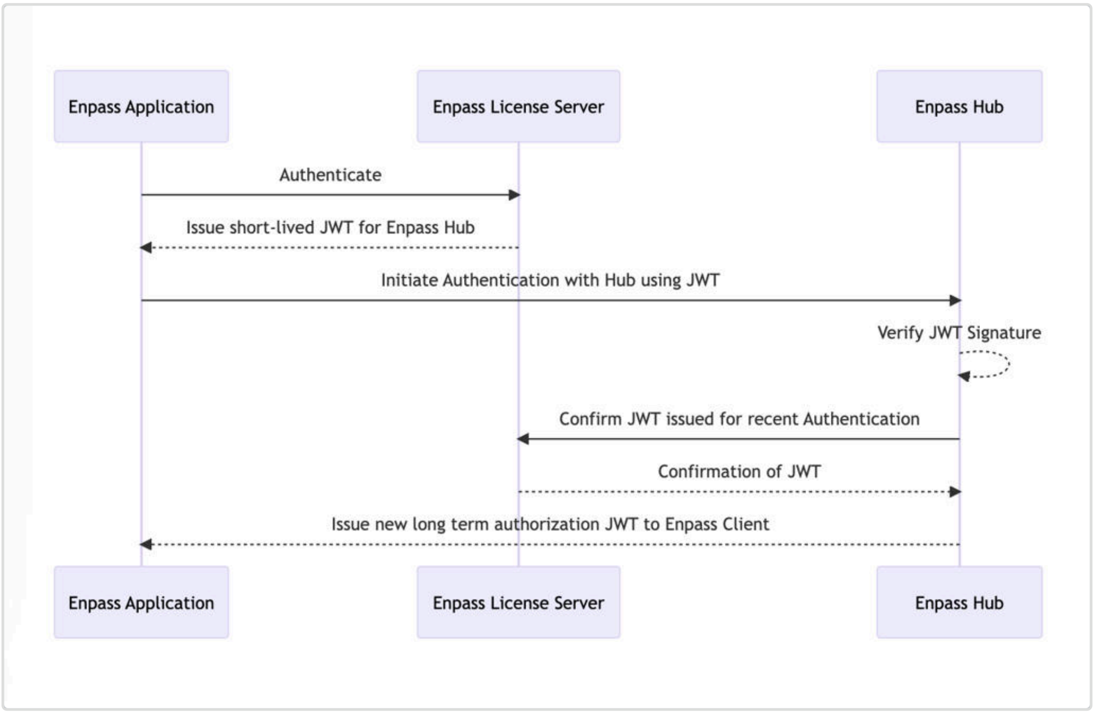
1. **Secret Key**: The integration requires with the admin generating a long, random secret key on the Enpass Hub specific to the organization. This key is configured in the Enpass

account of organization and it serves as a mutually agreed secret used to issue short-lived authentication tokens to app as part of the authentication process.

2. **SSL Certificate SHA-256 Fingerprint**: Optionally, the admin can provide the SHA-256 hash of the SSL certificate. The connected Enpass applications will utilize this for certificate pinning, ensuring there is no Man-in-the-Middle or intercepting proxy between app and Enpass Hub.

User Authorization

To use Enpass Hub features, a User's app must be authorized first, in order to maintain the security and privacy of the Enpass Hub. The authentication the process is as followed:


Enpass Application
Enpass License Server
Enpass Hub
Authenticate
Issue short-lived JWT for Enpass Hub
Initiate Authentication with Hub using JWT
Verify JWT Signature
Confirm JWT issued for recent Authentication
Confirmation of JWT
Issue new long term authorization JWT to Enpass Client
Enpass Application
Enpass License Server
Enpass Hub

The steps involved in user authorization to Enpass Hub are as follows:

1. **Authentication with Enpass License Server**: When a user's Enpass app authenticates with the Enpass license server, the server issues a short-lived JWT (JSON Web Token) for Enpass Hub with information such as Hub URL and SSL certificate pinning hash (if configured). The JWT is signed (HS-256 algorithm) using a pre-agreed Secret Key between the Enpass license server and Enpass Hub, configured during integration.
2. **Initiating Authentication with Enpass Hub**: The Enpass app then initiates authentication with the Enpass Hub using the signed JWT received from the Enpass license server.

3. **JWT Verification:** Enpass Hub verifies the JWT's signature locally with its copy of Secret Key and on successful verification, it contacts the Enpass license server to confirm if it was issued for a recent authentication i.e. within 1 minute.
4. **App Token Issuance**: After successful verification and confirmation, Enpass Hub issues an new JWT token to the Enpass app. This token allows the app to access the necessary features and resources on the Enpass Hub securely on the user's behalf.

Key Generation and Storage

The foundation of Enpass Hub's security architecture is the generation and storage of cryptographic keys. These keys play a crucial role in ensuring the security of sensitive data while enabling the desired features for organizations. The key generation and storage process involve the following steps:

1. **User Keypair Generation**: When a user's Enpass app connects to Enpass Hub first time, a unique RSA 3072-bit keypair (public and private key) is generated for the user by the app.
2. **Private Key Storage**: The user's private key is securely stored in the user's Enpass vault. **The private key never leaves the user's vault**, ensuring that it remains protected from unauthorized access. This private key will be only used for decrypting data locally.
3. **Public Key Storage**: The user's public key is sent to the Enpass Hub and stored there. This public key will be used for encrypting sensitive data that can only be decrypted by the user's private key.

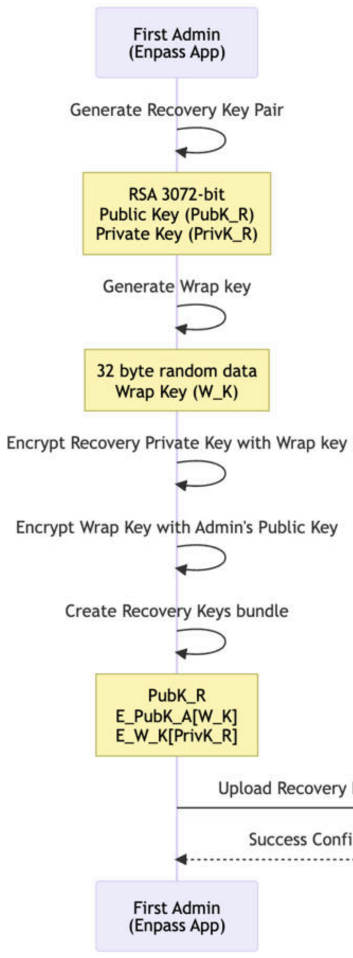
Access Recovery

In individual use, if you forget your master password, access to your vault is lost permanently. In an business setting, however, users forgetting passwords is common, and losing all data is unacceptable. Enpass addresses this with a secure recovery system that enables an admin to help a user reset their master password **without ever knowing it or decrypting the vault data**. This is achieved by leveraging the encrypted vault keys stored on the Hub and the organization's Recovery Key pair.

First Recovery Admin Setup

The initial admin setup is a the first step in enabling the Access Recovery feature in Enpass Hub. During this process, the first recovery admin sets up the necessary keys and shares them with the Enpass Hub. The steps involved in the initial admin setup are as follows:

1. **Recovery Key Pair Generation**: The first recovery admin generates a RSA 3072-bit master recovery keypair. This key pair consists of a public key (PubK_R) and a private key (PrivK_R).
2. **Encryption of the Recovery Private Key**: The recovery admin encrypts the private key (PrivK_R) with a random AES-256 symmetric wrap key (W_K): E_W_K[PrivK_R]
3. **Encryption of the Wrap Key**: The recovery admin encrypts the wrap key (W_K) with their own public key (PubK_A): E_PubK_A[W_K]
4. **Sharing Encrypted Keys with Enpass Hub**: The encrypted private key (E_W_K[PrivK_R]), encrypted wrap key (E_PubK_A[W_K]), and public key (PubK_R) are sent to the Enpass Hub. The keys are securely stored on the Enpass Hub, enabling the recovery process when necessary.



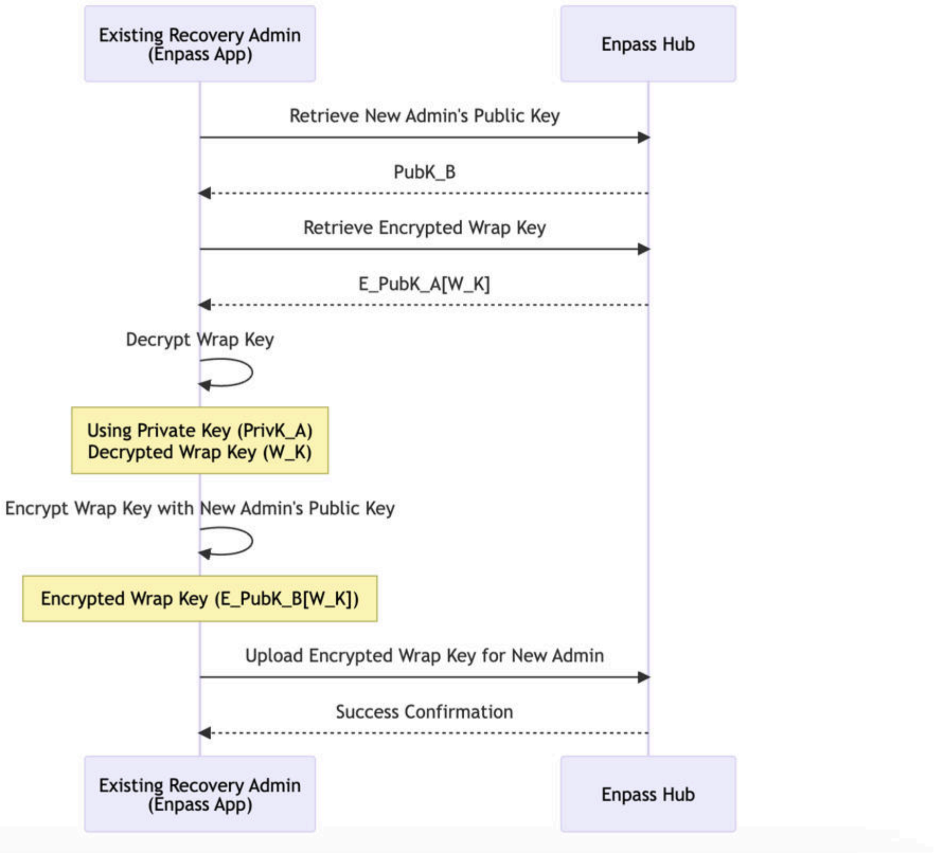
By completing the initial admin setup, the first recovery admin establishes the foundation for the Recovery feature by creating master recovery keypair. This process also ensures that only authorized admin has access to the master private key.

Adding a New Recovery Admin

Adding a new recovery admin is an important process to ensure redundancy and prevent lockouts in case of data loss or if an existing admin forgets their master password. An existing recovery admin with sufficient permissions can add another user as a recovery admin. The steps involved in adding a new recovery admin are as follows:

1. **Retrieve Encrypted Wrap Key**: The existing recovery admin retrieves the encrypted wrap key of the encrypted recovery private key (E_W_K[PrivK_R]) from Enpass Hub.
2. **Decrypt Wrap Key**: The existing recovery admin decrypts the encrypted wrap key (E_PubK_A[W_K]) using their private key (PrivK_A): D_PrivK_A[E_PubK_A[W_K]] = W_K
3. **Encrypt Wrap Key with New Recovery Admin's Public Key**: The existing recovery admin encrypts the decrypted wrap key (W_K) with the new recovery admin's public key (PubK_B): E_PubK_B[W_K]

4. **Share Encrypted Wrap Key with Enpass Hub**: The encrypted wrap key (E_PubK_B[W_K]) is sent to Enpass Hub and associated with the new recovery admin.


Existing Recovery Admin (Enpass App)
Enpass Hub
Retrieve New Admin's Public Key
PubK_B
Retrieve Encrypted Wrap Key
E_PubK_A[W_K]
Decrypt Wrap Key
Using Private Key (PrivK_A) Decrypted Wrap Key (W_K)
Encrypt Wrap Key with New Admin's Public Key
Encrypted Wrap Key (E_PubK_B[W_K])
Upload Encrypted Wrap Key for New Admin
Success Confirmation
Existing Recovery Admin (Enpass App)
Enpass Hub

By following these steps, the new recovery admin can participate in the Recovery process, ensuring that the organization maintains redundancy and security in the password recovery process.

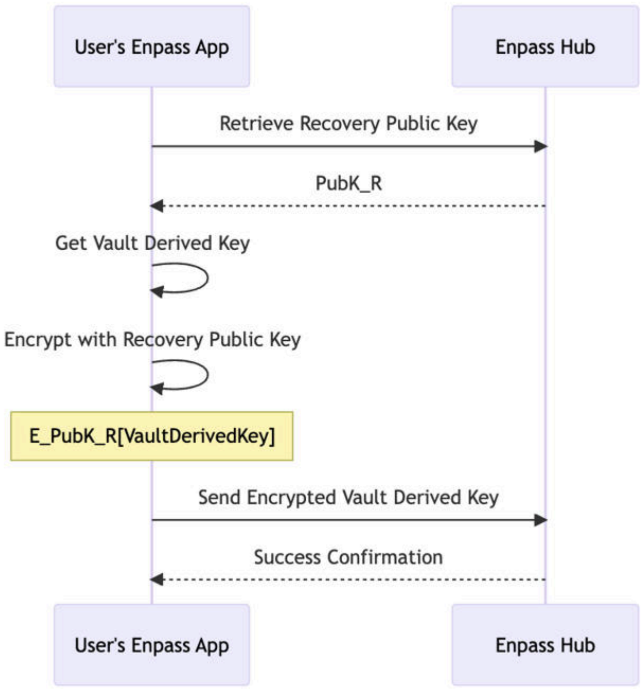
Enpass recommends having at least two active recovery admins at a time to avoid any lockout in case of data loss or if an admin forgets his master password, which won't be recoverable.

User Recovery Data

Once the recovery public key (PubK_R) is available, Enpass app of each user, connected to the Enpass Hub, will send their business vault's derived keys encrypted with the master recovery public key to Hub. The steps involved in generating and sending the recovery data are as follows:

1. **Vault Derived Key Encryption**: Each user's Enpass application encrypts the derived key of their business vault and the identity info of the user, using the recovery public key (PubK_R): E_PubK_R[VaultDerivedKey]

2. **Sending Encrypted Vault Derived Key to Enpass Hub**: The encrypted vault derived key (E_PubK_R[VaultDerivedKey]) is sent to the Enpass Hub and securely stored. This ensures that only authorized recovery admins with the corresponding recovery private key can access and decrypt the vault keys in case of a recovery request.


User's Enpass App
Enpass Hub
Retrieve Recovery Public Key
PubK_R
Get Vault Derived Key
Encrypt with Recovery Public Key
E_PubK_R[VaultDerivedKey]
Send Encrypted Vault Derived Key
Success Confirmation
User's Enpass App
Enpass Hub

The recovery data storage process ensures that the Enpass Hub can securely facilitate the Master Password Recovery feature, allowing recovery admins to access the necessary data for recovering a user's master password.

Recovery Process

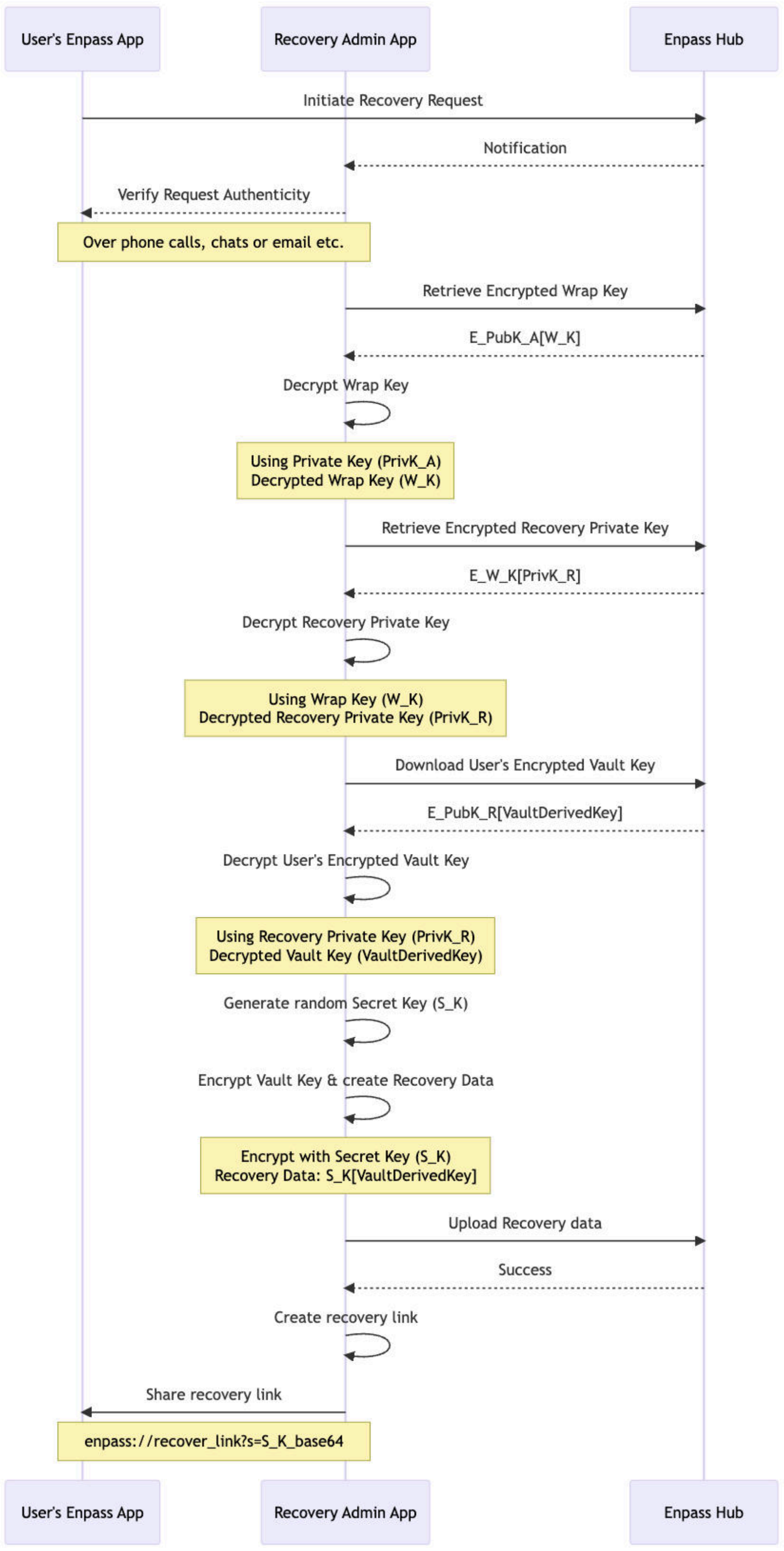
The recovery process is initiated when a user forgets their master password and requests assistance from a recovery admin. This process is designed to securely restore the user's access to their Enpass vault while maintaining the confidentiality of the data. The steps involved in the recovery process are as follows:

1. **Recovery Request Initiation**: The user initiates the recovery process from their Enpass application, authenticates with their email, and sends a recovery request to the recovery admin.
2. **Request Verification**: The recovery admin receives the request and verifies its authenticity using external means such as phone calls, chats, or the company's support system.
3. **Recovery Admin Decrypts Wrap Key and Recovery Private Key**: The recovery admins' app retrieves the encrypted wrap key (E_PubK_A[W_K]) and encrypted recovery private key

(E_W_K[PrivK_R]) from Enpass Hub. The wrap key (W_K) is decrypted using the admin's private key (PrivK_A), and the recovery private key (PrivK_R) is decrypted using the wrap key (W_K).

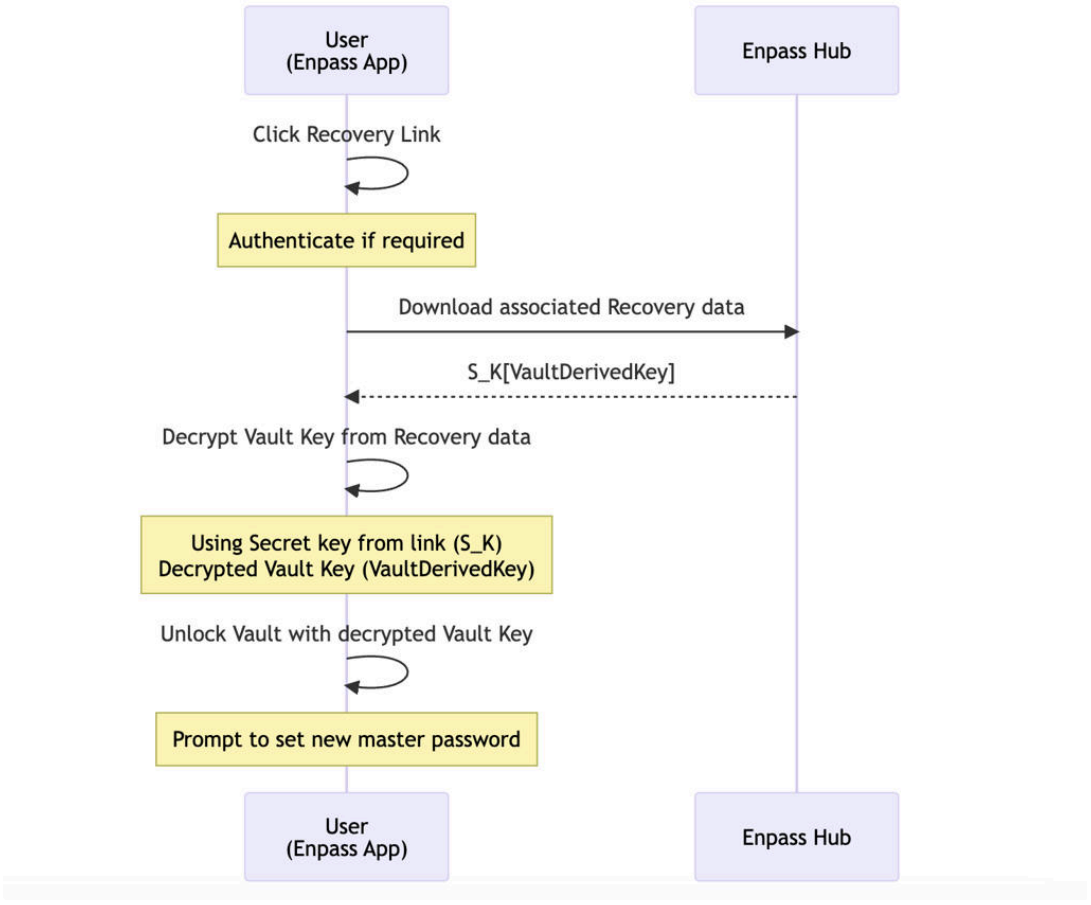
4. **Download and Decrypt User's Encrypted Vault Key**: The user's encrypted vault key (E_PubK_R[VaultDerivedKey]) is downloaded from the Enpass Hub. The recovery admin decrypts the encrypted vault key using the recovery private key (PrivK_R).
5. **Generate Temporary Secret and Encrypt Vault Key**: The recovery admin's app generates a random secret (s) and encrypts the vault key with it. The encrypted vault key is stored as a temporary accessible item on Enpass Hub, and a link is generated containing the random secret:
   enpass://recover_link?s=<secret-base64-encoded>.
6. **Share Recovery Link with User**: The recovery admin shares the link with the user through an external means. The link is valid for 2 hours by default. Super admin of self-hosted Enpass Hub can adjust this links validity according to the their organization security policy.

   Note: The secret is never stored or relayed though Hub. Admin should use an appropriate channel to send this link to user.


User's Enpass App
Recovery Admin App
Enpass Hub
Initiate Recovery Request
Notification
Verify Request Authenticity
Over phone calls, chats or email etc.
Retrieve Encrypted Wrap Key
E_PubK_A[W_K]
Decrypt Wrap Key
Using Private Key (PrivK_A)
Decrypted Wrap Key (W_K)
Retrieve Encrypted Recovery Private Key
E_W_K[PrivK_R]
Decrypt Recovery Private Key
Using Wrap Key (W_K)
Decrypted Recovery Private Key (PrivK_R)
Download User's Encrypted Vault Key
E_PubK_R[VaultDerivedKey]
Decrypt User's Encrypted Vault Key
Using Recovery Private Key (PrivK_R)
Decrypted Vault Key (VaultDerivedKey)
Generate random Secret Key (S_K)
Encrypt Vault Key & create Recovery Data
Encrypt with Secret Key (S_K)
Recovery Data: S_K[VaultDerivedKey]
Upload Recovery data
Success
Create recovery link
Share recovery link
enpass://recover_link?s=S_K_base64
User's Enpass App
Recovery Admin App
Enpass Hub

7. **User's Access Recovery with Recovery Link**: The user processes the link in their Enpass app (after authenticating with email if not already), app downloads the encrypted vault key from

the server, decrypts it with the secret from the link, and unlocks their vault. Enpass prompts the user to set a new master password.


User (Enpass App)
Enpass Hub
Click Recovery Link
Authenticate if required
Download associated Recovery data
S_K[VaultDerivedKey]
Decrypt Vault Key from Recovery data
Using Secret key from link (S_K) Decrypted Vault Key (VaultDerivedKey)
Unlock Vault with decrypted Vault Key
Prompt to set new master password
User (Enpass App)
Enpass Hub

By following these steps, the recovery process ensures a secure method for restoring a user's access to their Enpass vault without compromising the confidentiality of their data.

Handling Key-Pair Loss

In an unlikely event that a user loses their access to vault and he will need to start everything from scratch. This is because all the access keys shared to the user were encrypted using his registered public key on the Enpass Hub and now he has no longer access to the corresponding private key. To start everything from scratch, the following steps must be taken to ensure the user can regain access to shared vaults and, if applicable, their recovery admin status:

1. **Reset Public Key Registration**: The user must ask the administrator to reset their registered public key on the Enpass Hub server. This will remove the old public key associated with the user.

2. **Register new Public Key**: The user will now start the Enpass app and create a new vault if not already. The new public key will be generated and registered with Enpass Hub as described in “Key Generation and Storage” section.
3. **Loss of Shared Vault Access**: The user's access to all previously shared vaults will be lost due to the absence of their original private key. Vault owners will need to re-share access to the affected vaults with the user.
4. **Re-adding a Recovery Admin**: If the user was a recovery admin, they will need to be re-added by another recovery admin.

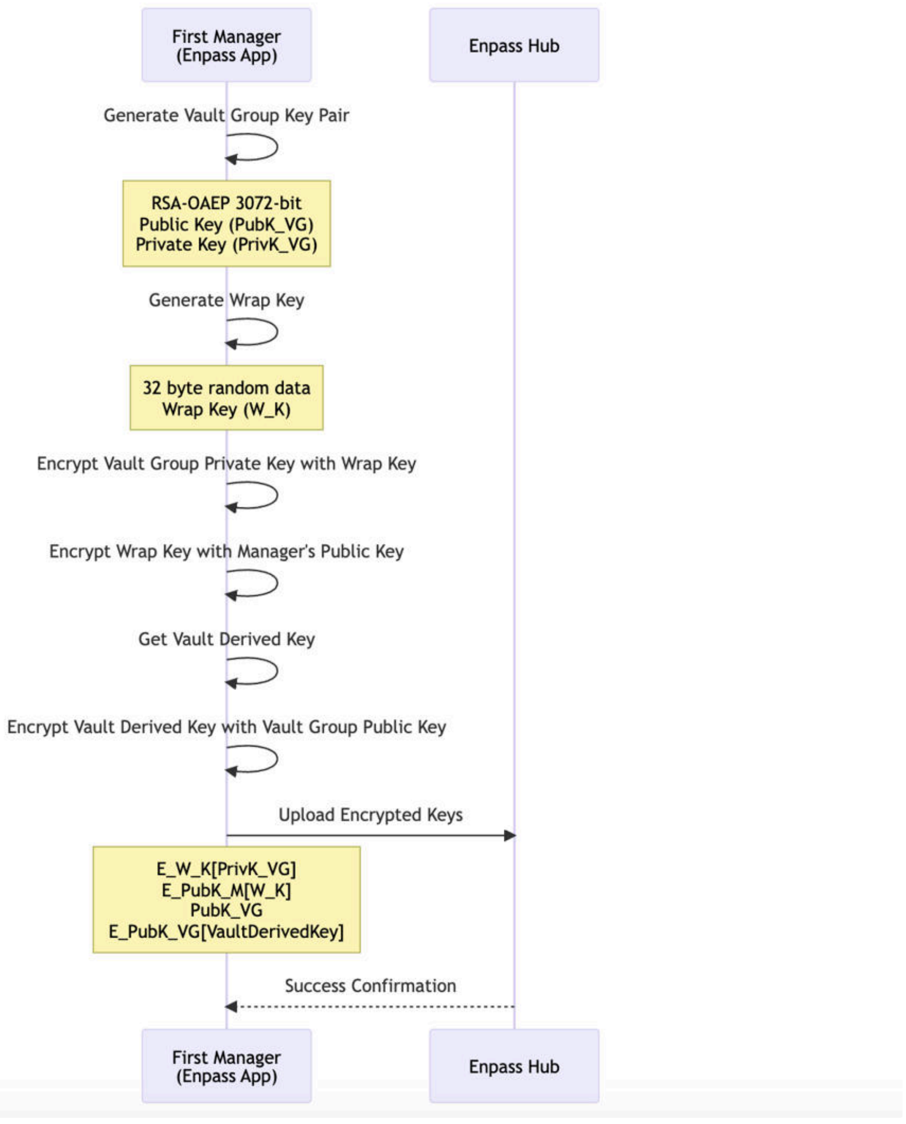
Vault Sharing

Vault sharing is an essential feature for organizations that allows users to securely share access to their vaults with other users. The process ensures that only authorized users can access the shared vault and is based on public-key cryptography. It involved following processes:

Enabling Sharing of a Vault

When the first manager or creator of a vault initiates the sharing process for the first time on their Enpass application, vault is registered on the hub for sharing. The process involves following steps:

1. **Vault sharing group Key Pair Generation**: The first manager generates a vault group key pair using the RSA 3072-bit algorithm. This key pair consists of a public key (PubK_VG) and a private key (PrivK_VG).
2. **Encrypting Vault Group Private Key**: The vault group private key is encrypted with a random AES-256 symmetric wrap key (W_K): E_W_K[PrivK_VG]
3. **Encrypting Wrap Key**: The wrap key (W_K) is encrypted with the manager's public key (PubK_M): E_PubK_M[W_K]
4. **Encrypting Vault Derived Key**: The vault-derived key is encrypted with the group's public key (PubK_VG): E_PubK_VG[VaultDerivedKey]
5. **Sharing Encrypted Keys with Enpass Hub**: The encrypted private key (E_W_K[PrivK_VG]), encrypted wrap key (E_PubK_M[W_K]), group public key (PubK_VG), and encrypted vault key (E_PubK_VG[VaultDerivedKey]) are sent to Enpass Hub.


First Manager (Enpass App)
Enpass Hub
Generate Vault Group Key Pair
RSA-OAEP 3072-bit
Public Key (PubK_VG)
Private Key (PrivK_VG)
Generate Wrap Key
32 byte random data
Wrap Key (W_K)
Encrypt Vault Group Private Key with Wrap Key
Encrypt Wrap Key with Manager's Public Key
Get Vault Derived Key
Encrypt Vault Derived Key with Vault Group Public Key
Upload Encrypted Keys
E_W_K[PrivK_VG]
E_PubK_M[W_K]
PubK_VG
E_PubK_VG[VaultDerivedKey]
Success Confirmation
First Manager (Enpass App)
Enpass Hub

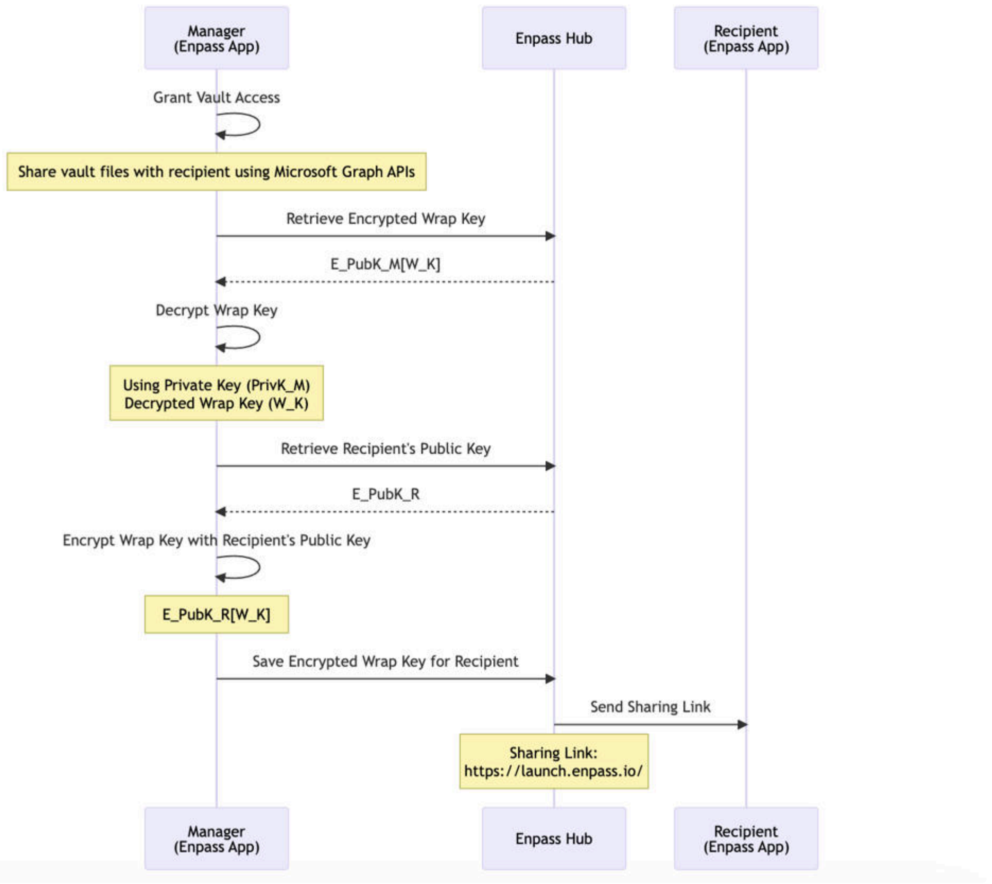
The vault is now ready to be shared securely with other users.

Sharing with Other Users

The manager can now share the vault by adding more users to the vault sharing group. The process involves following steps:

1. **Granting Vault Access**: The manager's app shares the vault files with the intended recipient using Microsoft Graph APIs, ensuring that the recipient can access the vault files on Microsoft 365 OneDrive or SharePoint.
2. **Retrieving and Decrypting Wrap Key**: The manager's app retrieves the his copy of encrypted wrap key of the vault sharing group (E_PubK_M[W_K]) from Enpass Hub and decrypts it with his private key (PrivK_M): D_PrivK_M[E_PubK_M[W_K]] = W_K
3. **Encrypting Wrap Key with Recipient's Public Key**: The manager encrypts the decrypted wrap key (W_K) with the new receiving party's public key (PubK_R): E_PubK_R[W_K]

4. **Saving Encrypted Wrap Key**: The encrypted wrap key (E_PubK_R[W_K]) is now sent to Enpass Hub, associated with the new user and appropriate permissions.

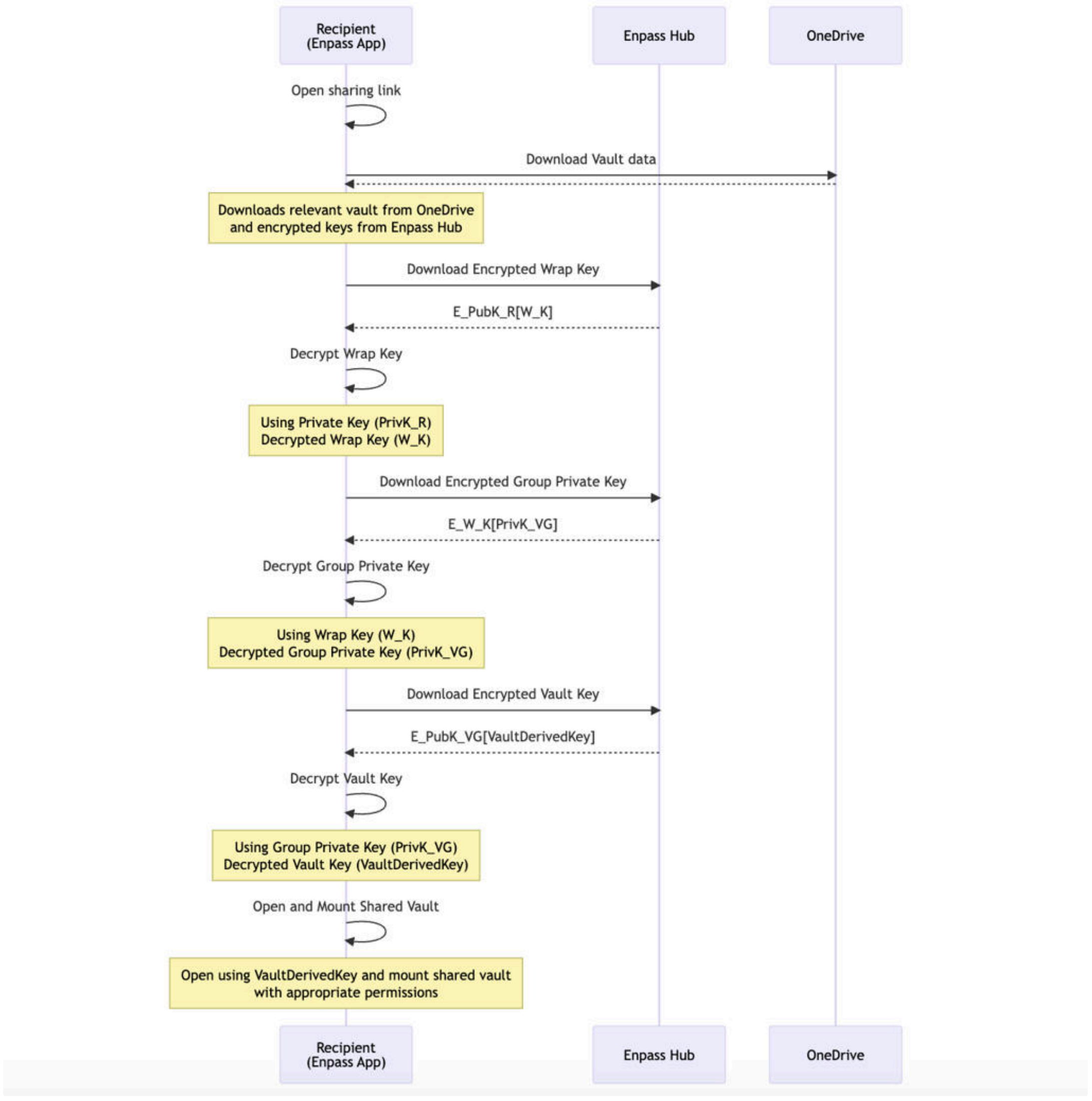

Manager (Enpass App)
Enpass Hub
Recipient (Enpass App)
Grant Vault Access
Share vault files with recipient using Microsoft Graph APIs
Retrieve Encrypted Wrap Key
E_PubK_M[W_K]
Decrypt Wrap Key
Using Private Key (PrivK_M) Decrypted Wrap Key (W_K)
Retrieve Recipient's Public Key
E_PubK_R
Encrypt Wrap Key with Recipient's Public Key
E_PubK_R[W_K]
Save Encrypted Wrap Key for Recipient
Send Sharing Link
Sharing Link: https://launch.enpass.io/
Manager (Enpass App)
Enpass Hub
Recipient (Enpass App)

5. **Generating Sharing Link**: A sharing link is generated and sent to the recipient via email, containing the identification of the shared vault: Enpass <share-id>

Adding a Shared Vault

1. **Opening Enpass App**: The recipient clicks the sharing link, opening their Enpass app.
2. **Downloading Vault and Encrypted Keys**: The app downloads the relevant vault from OneDrive, the encrypted vault group private key (E_W_K[PrivK_VG]), the encrypted vault key (E_PubK_VG[VaultDerivedKey]), and his copy of encrypted wrap key (E_PubK_R[W_K]) from Enpass Hub.
3. **Decrypting Wrap Key**: The recipient decrypts the wrap key (E_PubK_R[W_K]) with his private key (PrivK_R): D_PrivK_R[E_PubK_R[W_K]] = W_K
4. **Decrypting Group Private Key**: The decrypted wrap key (W_K) is used to decrypt the encrypted vault key group private key (E_W_K[PrivK_VG]): D_W_K[E_W_K[PrivK_VG]] = PrivK_VG

5. **Decrypting Vault Key**: The decrypted PrivK_VG is used to decrypt the encrypted vault key (E_PubK_VG[VaultDerivedKey]): D_ PrivK_VG [E_PubK_VG[VaultDerivedKey]] = VaultDerivedKey
6. Opening and Mounting Shared Vault: The shared vault is opened and mounted with appropriate permissions using the decrypted vault derived key.



Recipient (Enpass App)
Enpass Hub
OneDrive
Open sharing link
Download Vault data
Downloads relevant vault from OneDrive and encrypted keys from Enpass Hub
Download Encrypted Wrap Key
E_PubK_R[W_K]
Decrypt Wrap Key
Using Private Key (PrivK_R) Decrypted Wrap Key (W_K)
Download Encrypted Group Private Key
E_W_K[PrivK_VG]
Decrypt Group Private Key
Using Wrap Key (W_K) Decrypted Group Private Key (PrivK_VG)
Download Encrypted Vault Key
E_PubK_VG[VaultDerivedKey]
Decrypt Vault Key
Using Group Private Key (PrivK_VG) Decrypted Vault Key (VaultDerivedKey)
Open and Mount Shared Vault
Open using VaultDerivedKey and mount shared vault with appropriate permissions
Recipient (Enpass App)
Enpass Hub
OneDrive

By following these steps, the vault sharing process ensures that only authorized users can access the shared vault while maintaining the confidentiality and integrity of the data.

Key Features

Subtopics : [Browser Extensions] [Quick Unlock] [Wearables] [Wi-Fi Sync Server]

Password Generator

Using unique passwords for each website is the first and most important step toward security. Enpass includes a built-in password generator that provides this capability. It uses **OpenSSL RAND_bytes** to produce cryptographically secure randomness, ensuring that all generated passwords are unique and unpredictable.

In some situations, you may need passwords that are easier to remember or pronounce. Enpass can also generate pronounceable passwords, which are created using the Diceware methodology with a 14996-word dictionary. More details about Diceware are described here.

Password Strength Estimation

Password strength is a measure of how resistant a password is to being guessed or cracked. Password **entropy** is the primary metric for assessing password strength. It measures the unpredictability of a password and higher entropy corresponds to stronger passwords.

Entropy Calculation

Enpass uses the **Zxcvbn,** a popular open-source password strength estimation algorithm, to calculate the entropy of random passwords. Zxcvbn evaluates password strength by analysing: Length, variety of character types (letters, numbers, symbols), common patterns (e.g., "123456" or "qwerty"), substitutions, predictable sequences, common passwords and dictionary word usage.

This comprehensive analysis provides a realistic estimate of password strength, considering factors beyond simple randomness. For more details please visit: zxcvbn: realistic password strength estimation

For Pronounceable passwords

While Zxcvbn is excellent at evaluating the complexity of random passwords and identifying weak patterns, the pronounceable passwords generated using Diceware, have their own entropy calculation model. Diceware entropy focuses on the randomness of the word selection process in passphrases and it is derived from the number of words in the passphrase and the size of the wordlist. The formula for Diceware entropy is:

Entropy = log2(wordlist_size ^ number_of_words)

Enpass uses a 14996-word dictionary for diceware based pronounceable passwords generation. For a 14996 word dictionary, a 8 word passphrase would have an entropy of approximately 110 bits. This calculation assumes that each word is selected independently and uniformly at random from the dictionary.

Enpass calculates both Zxcvbn and Diceware entropy for pronounceable passwords and uses the lower value to determine the password's strength. This approach ensures a conservative estimation of security.

For Master password

Master password of Enpass is a special kind of password where extra measure has been taken to reduce attack surface of brute-force attacks such use of PBKDF2 with adequate number of iterations. It significantly increases the computational cost of each password guess, making brute-force attacks far more time-consuming. This means that the effective strength of a master password is much higher compared to the same password string used without PBKDF2. The password strength estimator always takes this extra protection into consideration. Enpass implements **PBKDF2** with at least 100K iterations (old versions) before it is actually used for unlocking the vault. Hence, the attacker's effective guess rate is reduced by a factor of 100000.

You can observe that a password will be labelled with a better strength, when used for a master password of vault due to the effective guess time increased.

Password Strength Meter

Enpass shows a password with corresponding strength meter or a human readable strength label. It helps users easily understand the security level of their passwords and encourages the use of stronger credentials. The Enpass password strength meter is calibrated as follows:

Entropy range	Strength
<= 35	Very Poor
35-50	Weak
50-70	Average
70-100	Good
>= 100	Excellent

Checking for Compromised Passwords

Enpass allows you to check your passwords against the public database of breached credentials maintained by Troy Hunt's *Have I Been Pwned* service. This process is designed with strict privacy guarantees—your actual passwords are never transmitted over the internet.

The check leverages the **k-Anonymity model**, which ensures that partial information is shared without revealing the full password hash. Specifically, only the first five characters of your password's SHA-1 hash (a 40-character cryptographic digest) are sent to https://api.pwnedpasswords.com. The service responds with a list of all leaked password hashes that share the same prefix. Enpass then performs a **local comparison** between your full hash and the returned list.

This approach guarantees that:

- The plaintext password is never exposed.
- The full hash is never shared outside your device.
- The external service cannot determine which specific password you are checking.

If a match is found, Enpass issues a warning that the password has appeared in known breaches and must not be reused.

Pre-Shared Key–Based Item Sharing

Enpass provides a feature that allows users to securely share individual items with others. While your data is always protected within Enpass, security risks may arise when an item is shared outside the vault. To mitigate this, Enpass supports sharing with a Pre-Shared Key (PSK).

In this method, you generate a PSK for the intended recipient from the advanced settings in Enpass. When you share an item, it is encrypted using this PSK, ensuring that only you and the recipient—who both know the key—can decrypt it. For maximum security, it is strongly recommended to transmit the PSK to the recipient through a **separate communication channel** from the one used to deliver the encrypted item.

> For Enpass Business users, this one-way sharing mechanism is not recommended. It lacks revocable access control, and shared items must be re-shared if they are updated. Instead, organizations are encouraged to use **Vault Sharing**, which provides centralized, revocable, and seamless access management.

Browser extensions

Enpass browser extensions are additional software that need to be install in the browser. They communicate through the local Web Sockets with Enpass app. Each extension needs to be paired with user intervention diminishing the chance of MITM attack. All information is exchange through an encrypted channel with a common key derived using SRP-6a handshake for each session.

The following precautions have been taken to keep the data secure from any unauthorized access.

- **Extension Validity Check-** When a browser extension tries to connect to main Enpass App, we verify the origin of the connection, it must be the unique identifier of our browser extension (Browsers does not allow installation of any two extensions with the same ID).
- **Browser Validity Check-** When a browser extension tries to connect to the main Enpass app, we also verify its authenticity by checking its code signature i.e. browser must be code signed with relevant company's certificate (i.e Chrome is signed by Google). If a browser is code signed and not in our allowlist, an additional confirmation is required whether it should be allowed to connect to Enpass or not. This browser Validity Check is not performed on Linux as code signing is not available on the platform.
- **Pairing-** Browser extension need to initiate a SRP handshake with a pairing code displayed. User have to manually enter that pairing code into Enpass. If the pairing code matches, both Enpass app and browser extension will be having a common secret at the end of handshake. Further communication will be encrypted with that common secret. Browser extension will also store a provided pairing key for further sessions. It is stored in extension sandbox. User can opt not to store next session pairing key on the disk and in that case browser need to pair for every session.
- **Same-origin policy-** To prevent the attacks like sweep-in, Enpass auto-fills only on the web pages matching with the saved domain/host names. Furthermore, Enpass never automatically executes any scripts on page-loading for auto-filling, rather, first it presents a list of items matching with the same domain/hostname and later executes the script to autofill, once the user has selected the item to autofill with.

Quick Unlock

Mobiles

Enpass facilitate easy unlocking of data through biometrics authorization using fingerprint or face recognition. However, Enpass still requires the master password to decrypt the database where various OS provided hardware protection APIs help us to protect your master password that can only be revealed after biometric authentication.

Biometric and PIN on iOS

iOS Keychain provides a way to store app-specific sensitive data that can only be accessed by that app. When you enable Biometric or PIN on your device, Enpass stores an obfuscated version of your master password in iOS Keychain that can only be accessed by Enpass. In any case, your master password does not leave the device; neither during the backup of iTunes nor of iCloud Keychain.

Unlocking through Biometrics uses an addition protection of Secure Enclave. Secure Enclave lock the saved master password in such a way that it can be only accessible after a successful biometric authentication. This is so secure that even if someone gets access to your device passcode and legally adds his fingerprints in the phone, iOS immediately invalidates all the cryptographic keys and makes them unusable.

We have restricted the quick unlocking of Enpass only on the devices having a device passcode. Setting up a device passcode ensures that all the data (including the saved master password) in iOS keychain is protected by iOS itself. When you disable device passcode, quick unlock also gets disabled, removing saved master password.

Also, if five consecutive attempts to quick unlock are unsuccessful, Enpass erases the master password from the iOS keychain and prompts to enter master password to unlock Enpass. On next successful attempt, it is saved again in the iOS keychain. When you disable Quick Unlock from Enpass settings, Enpass erases the master password from the iOS keychain.

Biometric in Android

Android 6.0 and later, provides a new Fingerprint/Biometric API along with enhancement in Android Keystore which is reliable enough to protect our master password. As soon as you enable Biometric from Enpass settings on your device, we create a Biometric authenticated, Enpass specific random encryption/decryption key in Android Keystore (accessible to Enpass only) and encrypt your master password using this key and store in private area in the device storage. This solution is highly secure because your master password is accessible only after it gets decrypted using the same key stored in the Android Keystore. And that key (in a usable format) can only be retrieved when you authenticate Enpass with your Biometric. This is so secure that even if someone gets access to your Android device passcode and legally adds his fingerprints in the phone, Android immediately invalidates all the encryption/decryption keys and makes them unusable. Even when in use, Enpass will detect this and will automatically disable Biometric in Enpass settings and removes your master password.

PIN in Android

When locking condition met i.e. inactivity time or when app goes to background, Enpass restricts access to all its features and user have to provide a PIN to access data. This does not actually close the Enpass database but adds just an extra screen for authorization. On wrong PIN entry, database will be immediately closed and a master password will be required. Also, if app is killed by OS in background (some Android devices does it frequently if the Battery Optimization is enabled), Master Password will be required at next start.

Desktops

Enpass facilitates easy unlocking of data similar to mobile platforms through biometrics authorization using platform-specific APIs and Hardware Enclaves. When no Hardware Enclave is detected, it is advised to always use a master password for unlocking. However, in some cases, convenience wins over and a way to quickly access the Enpass data is required. Enpass does support the quick unlock through Fallback Biometric and PIN but a master password is required every time to unlock Enpass when the application is completely closed locking down the Enpass database.

Windows Hello (Microsoft Store version only)

Modern Windows devices have both biometric and TPM hardware. Thanks to Windows Hello as both can be utilized to solve the purpose reliably enough. As soon as you enable Windows Hello on a supported device, we create Windows Hello authenticated Enpass specific cryptographic keys in TPM and encrypt your master password using those keys and store in App's private area on-device storage. This solution is highly secure because your master password is accessible for use, only after it gets decrypted using the same keys stored in the TPM of your device. And those keys can only be retrieved when you authenticate Enpass with Windows Hello.

Touch ID on macOS (Apple Store version only)

Unlocking through TouchId in macOS uses the protection of Secure Enclave similar to iOS. Secure Enclave locks the saved master password in such a way that it can be only accessible after successful biometric authentication.

Fallback Biometrics

Quick unlocking through biometrics is available in both Windows and Mac through Windows Hello and Touch ID respectively. When locking conditions are met i.e. inactivity time or when the app goes to background, Enpass restricts access to data without locking down the database and users need to authorize using available Biometric method. If the user fails to authorize using biometric, the database will be closed and a master password will be required next time.

PIN

Enpass on all the desktop systems support the unlocking through PIN and it works the same way as fallback biometric for authorization.

Wearables

Limited processing power, memory, input capabilities and lack of APIs restrict Enpass to perform any true cryptographic operations on the supported wearables. Security of your data on wearables relies on default security provided by OS.

Apple Watch and Android Wearables

Enpass for Apple Watch and Android wearable works only with the main Enpass app, installed on paired iPhone or Android device using Apple Watchkit and Android Wearable Data Layer API respectively. Only those items are accessible on Watch which you manually mark as a Watch item on the main Enpass app. These shared items are stored on Watch and instead of your master password, they're protected by watch OS. However, these stored items do not leave the watch in any case; neither during the backup of iTunes and nor of iCloud Keychain. For extra safety, along with the Watch PIN, you can enable the PIN for Enpass on Watch also. Otherwise, if anyone gets access to your watch, will be able to see the items stored on it without any authorization required. So, one should use Apple watch with great attention and care.

Using Enpass on Watch is like carrying your physical wallet. For better security of your data, while using wearables, it is advised to

- Use a good device passcode.
- Use a good Enpass Watch PIN Code.
- Avoid storing unnecessary items on Watch. Only the frequently used short items like locker code etc. must be shared for Watch access.

Wi-Fi Sync Server

Enpass provide Wi-Fi Sync Server as an alternative to cloud offerings to sync between devices. Enpass Wi-Fi Sync Server is a small WebDAV file server under the hood with a self-signed certificate, restricted to a folder of your shared vaults. Since this server is on your local machine, its address is made discoverable by the mDNS protocol. Enpass client on other devices can discover running Wi-Fi Sync Server on the network and pair with it.

At any point of the time in transit, your data is always protected with your master password. The following precautions are taken to restrict unauthorized access to your encrypted data.

- **Access Password-** Access Password a unique password generated randomly for each vault added to the Enpass Wi-Fi Sync Server. It acts as a WebDAV server password for that vault.
- **Verification code-** The Wi-Fi Sync Server uses a self-signed certificate for https protocol. Since there is no certificate authority involved, it’s the users’ responsibility to check if the certificate is the right one. A time-based verification code is displayed on both the devices, the desktop that’s running the server and the device which is being paired. If this code does not match, there might be a man in middle if everything else is right. It is advised not to continue the pairing.

Mobile phone can be paired directly via a QR code without manual entry of these access code and verification code.