Technical White Paper

Teams and Personal Accounts

Contents

Key Security Principles

Confidentiality

Only users themselves have ability to share their sensitive data. No data can be viewed or modified if the owner of this data did not give permission to do that.

Integrity

We ensure data is not tampered with or altered by unauthorized users. Each change is signed by the user, so we validate changes in both the client and the server.

Availability

We ensure that our system and data are available to authorized users whenever they need it. All securely encrypted data is stored in PassCamp cloud, so you have the ability to access your account at any time and any place, where internet connection is available.

End-to-End Encryption

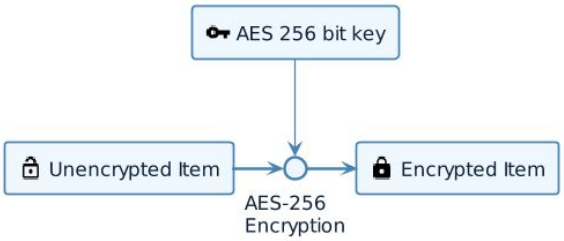
All cryptographic keys are generated and managed by the users on their devices. All encryption is completed locally. Our cloud server does not know anything about your secret data.

Encryption Technology

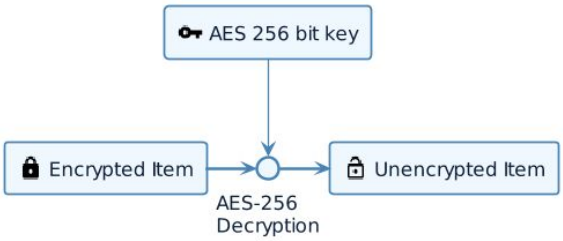
Symmetric Encryption

Symmetric encryption algorithm that PassCamp uses is Advanced Encryption Standard (AES). This method of encryption of any type of data is considered to be particularly secure and effective. All password items with sensitive data are encrypted with a new random AES 256-bit key generated at the client side.
Without knowing the key, encrypted data cannot be decrypted and is seen as completely scrambled and unintelligible.

Encryption



Decryption



Asymmetric encryption

We use RSA 4096-bit keys for asymmetric encryption. Each user has a pair of public and private RSA keys.

Public key

Public key is known to owner's contacts and is used to:

- encrypt data only a specific user can decrypt;
- verify that the author of changes on encrypted data is valid.

Private key

Only the owner of the private key knows it. It is used to:

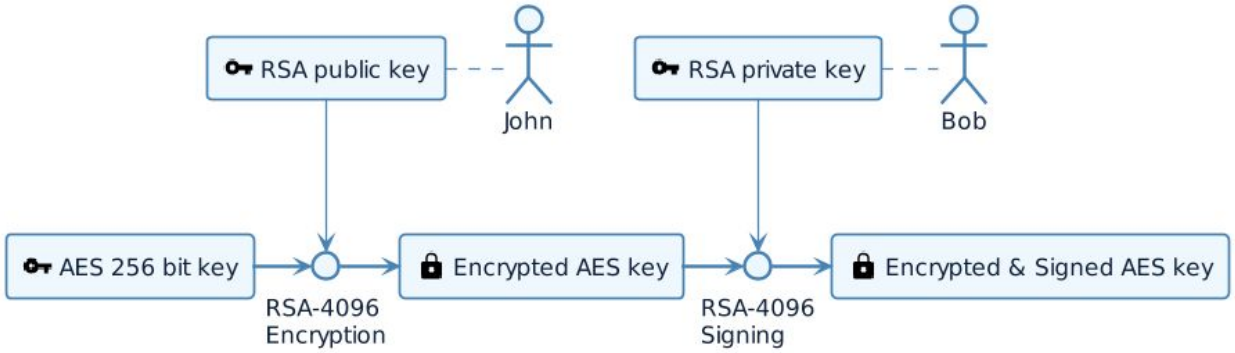
- decrypt data that was shared to user;
- sign changes on data to prove that it was made by a specific user.

By using RSA algorithm we can share AES keys with other users in a secure way. Also, we can verify if changes that a user made are valid in both the client and the server.

Example of sharing symmetric key using asymmetric encryption

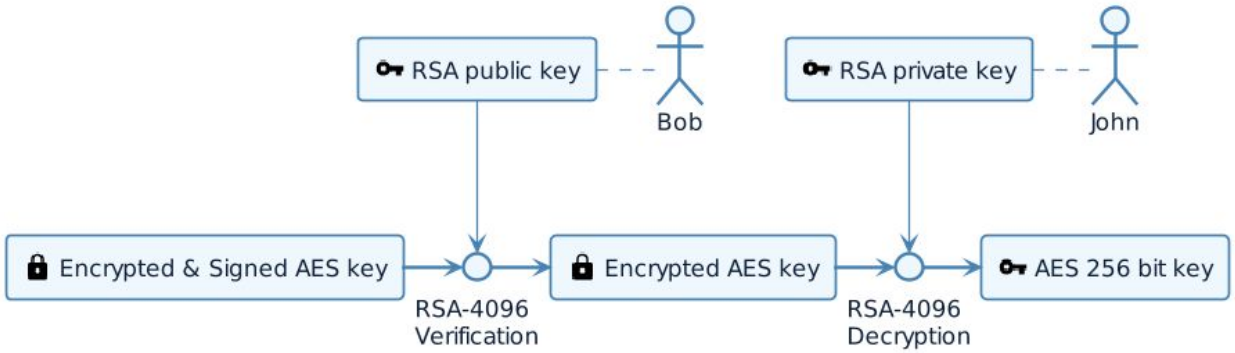
A. Bob shares a symmetric AES key with John. Steps:

1. Bob encrypts AES key using John's public key.
2. Bob signs encrypted AES key with Bob's private key.



B. John retrieves AES key. Steps:

1. John verifies signature by using Bob's public key
2. John decrypts AES key using his RSA private key.



User Authentication

We use Secure Remote Password (SRP) for user authentication. It conveys a zero-knowledge password proof from the user to the server. During SRP authentication, users demonstrate to the server that they know the master password, without sending the actual password itself, nor any other information from which the password can be extracted. The password never leaves user's device and is not known by the PassCamp server.

Master Password

When a user creates their PassCamp account, they also create a master password. The master password is used to authenticate access to the user's PassCamp account. The master password should never be used as a password for any other application or website. Users should also never share their master password with anyone. It is also used to lock and unlock user's RSA private key.

Registration

A user registers their credentials with PassCamp server by providing their user email, salt, a verifier, and RSA public, as well as encrypted private keys (See Figure 1). The verifier is a cryptographic value derived from the salt and master password using 100,000 iterations of PBKDF2 with HMAC as its pseudo-random function and SHA-256 hashing algorithm. Master password never leaves user's device and is used to lock user's RSA private key.

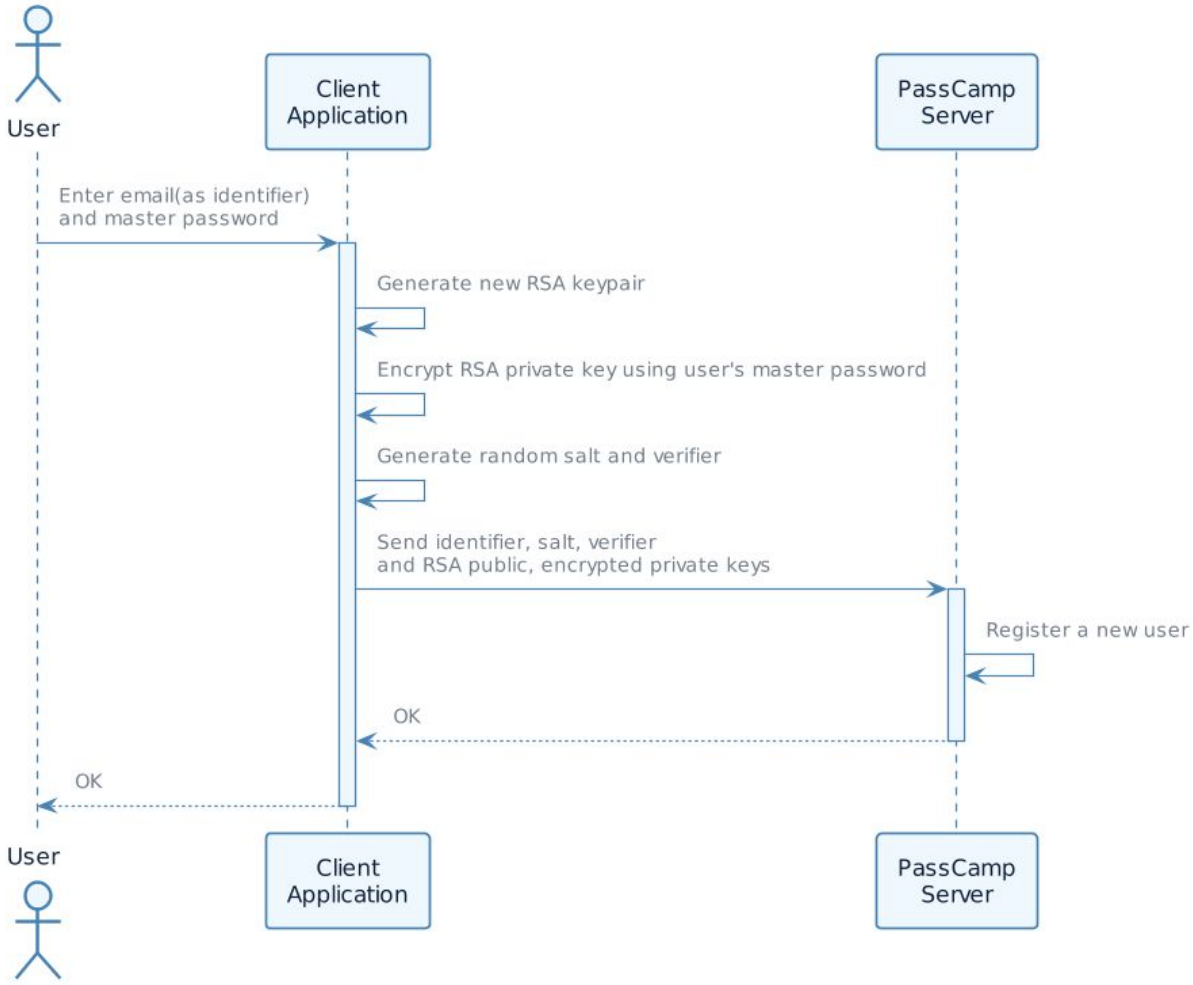


Figure 1: User Registration

Authentication

The actual SRP authentication is a multi-step process (See Figure 2).

1. Client generates private and public keys and sends email with public key to server.
2. Server checks the user:
    - If user exists - server responds with a valid SRP challenge.
    - If user does not exist - server responds with a fake SRP challenge. This is so third party couldn't check if a user with registered email is using our services.
3. Client tries to solve the SRP challenge and sends proof to the server.
4. Server validates proof and if it is valid, the server responds with access token that will be used to retrieve encrypted data from server.

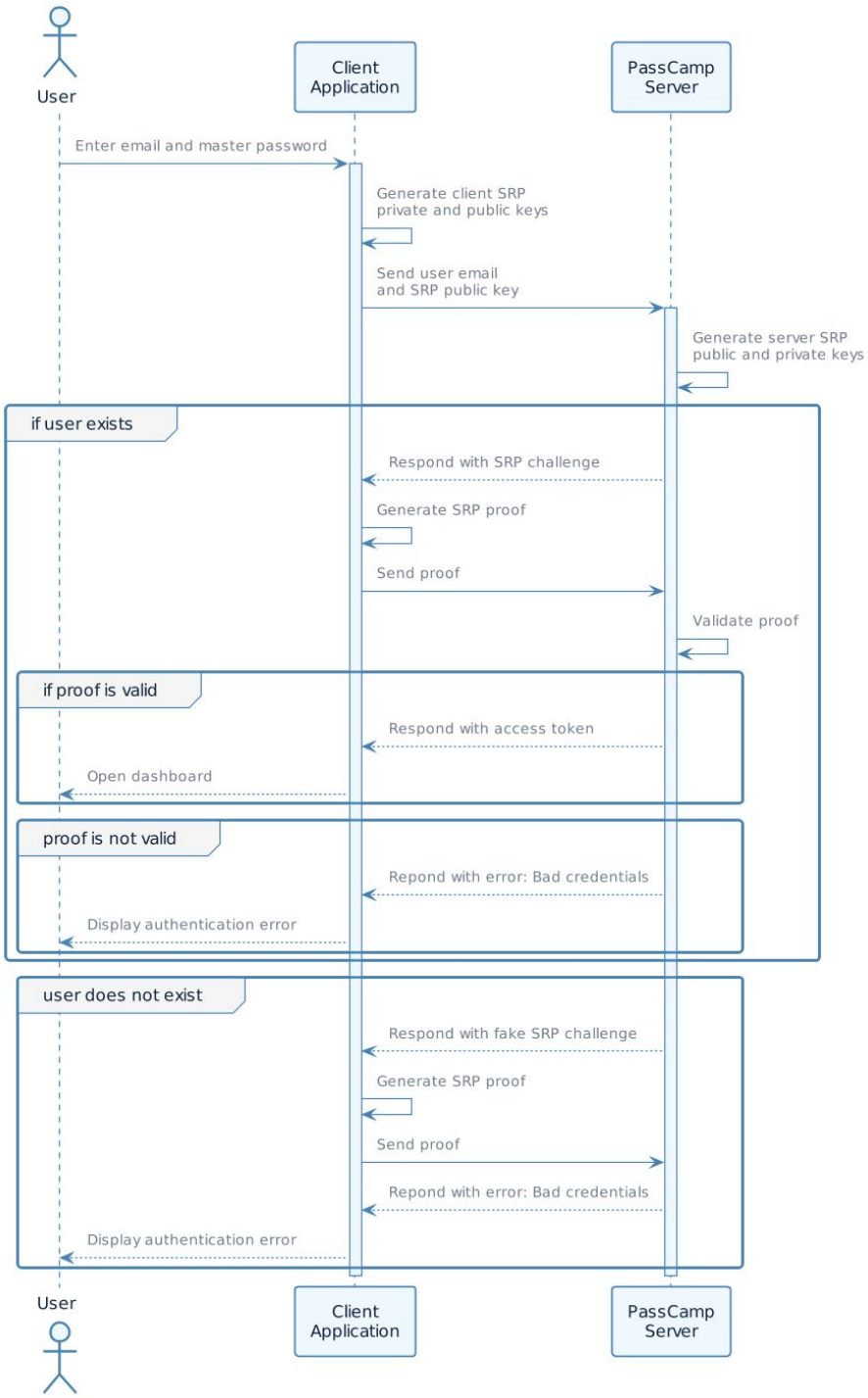
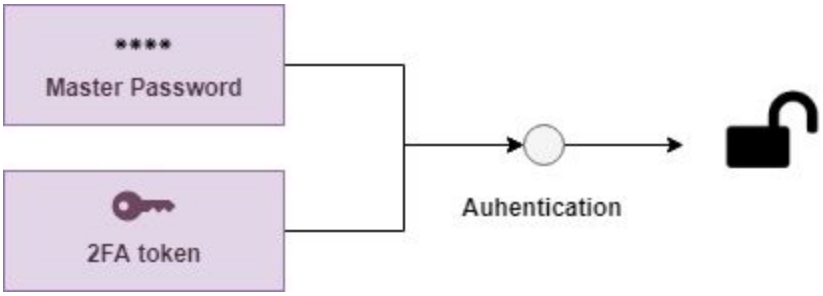


Figure 2: User authentication

Two-Factor Authentication (2FA)

PassCamp allows users to use two-factor authentication (2FA) to add an additional layer of protection. By requiring not only the master password, but also an additional login step (randomly-generated 6-digit TOTP code), a user adds another layer of protection against unauthorized access to their account. If an attacker were to discover a user's Master Password, it's unlikely that they would also have access to a valid 2FA token. This minimizes ability to gain access to the user's data.

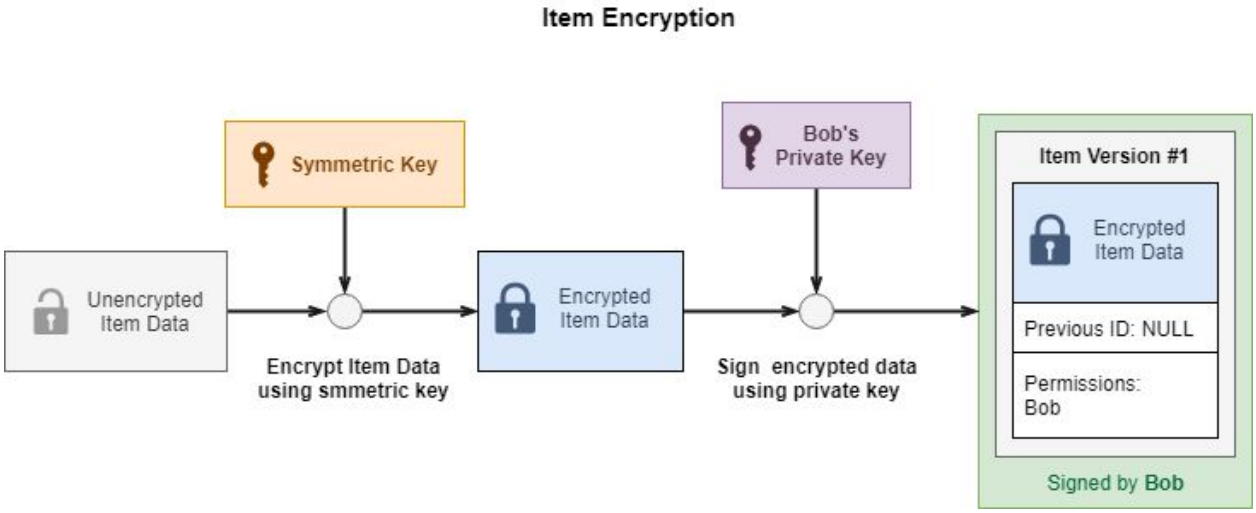


Password Items

What is different about PassCamp from other password managers is that each item is encrypted with a different symmetric key giving higher protection and observability. Each change on the item made by the user creates a new signed version allowing better item history tracking and item synchronization between users.

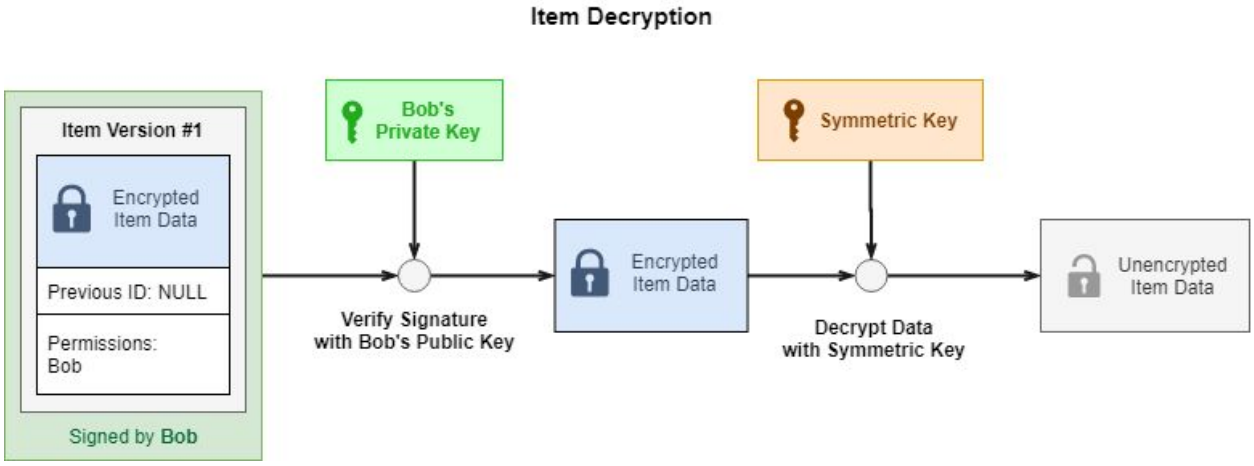
Encryption

For an encryption PassCamp uses both symmetric and asymmetric algorithms. Unencrypted data is encrypted with a randomly generated symmetric key for this item and signed with the user's private key.



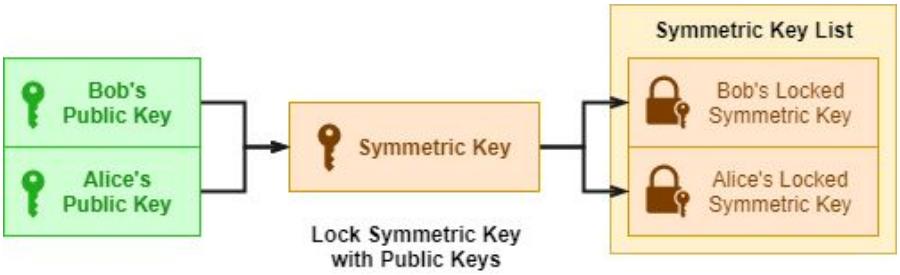
Decryption

Before the decryption process begins a signature must be validated to ensure item data was not modified by unauthorized parties. Item data is decrypted using the same symmetric key that was previously used for encryption.

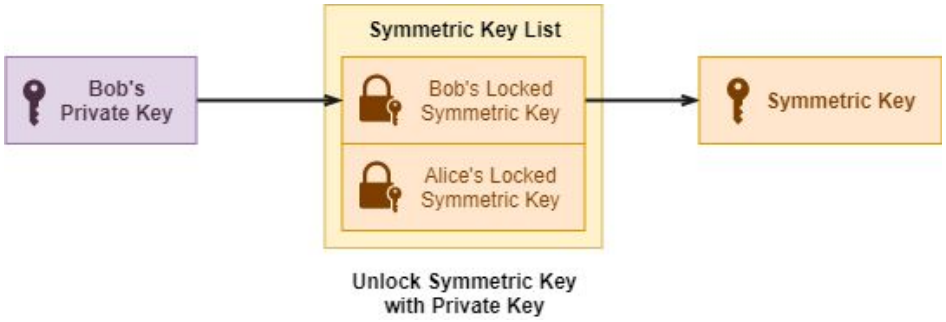


Sharing access

PassCamp allows users to share items with other users in such a way that unencrypted user's data never leaves his client's device when sharing. Sharing access relies heavily on asymmetric encryption. Sharing access requires sharing a symmetric key that is used to encrypt the item data. This key is locked with public keys of multiple users.

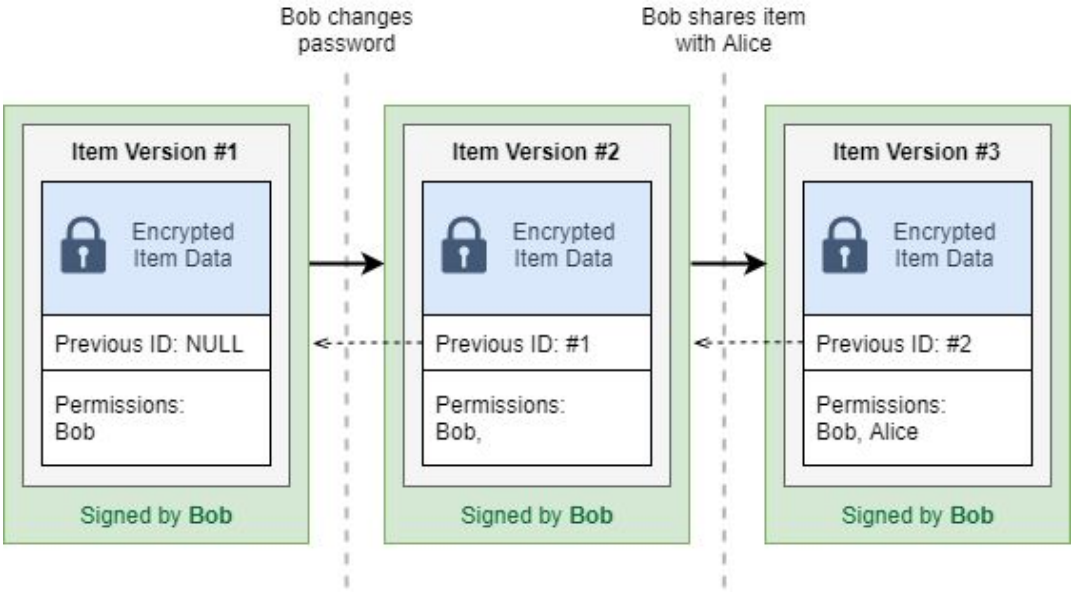


To decrypt an item and view item data, the user needs to retrieve a symmetric key. This can be done by using his private key.

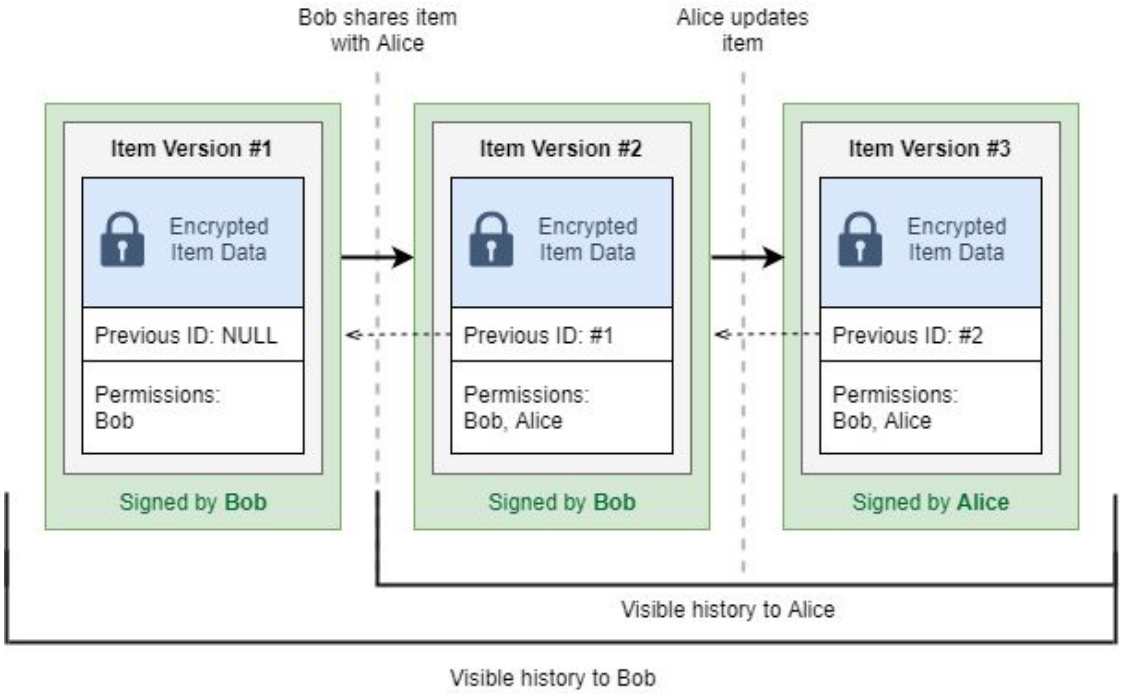


Versioning

PassCamp is versioning all items. Every change on the item produces a new version that is also signed by the user who made changes. Signatures provide a way for validating item modification authenticity. All of the versions have links to the previous versions and keeping them all strongly chained together.



Having separated versions of the item allows the user to track all modifications. Users can view history of item modifications and see how the item changed at the specific time. But users can only view versions that they previously had permission to view. For example if Bob creates a new item and shares it with Alice. Alice will be only allowed to view a new version and future versions where she has permission for access. She will not be able to view a previous version which has different data or a new version where she was removed from permissions.



Account Reset

Because PassCamp does not store the master password, it does not offer the same password reset option like other web services. Master Password is the key element of data protection and without it there is no possible way to decrypt user's private data. If a user forgets their master password, the user can only make a full reset of his account creating a new master password and an empty new vault.

Client Applications

Browser

PassCamp supports most of the modern browsers. Browser clients use webcrypto for most browser based cryptography. Master Password or Unlocked Private Key of the user are never stored in browser storage. So closing the window or tab will result in logout.

Mobile

PassCamp has mobile applications on iOS and Android. To ensure good protection all random keys and initialization vectors are generated using a secure random function. Cryptography algorithm implementations are provided by PointyCastle library.

Cloud Infrastructure

Hosting

PassCamp currently operates in Google Cloud Platform data center located in Belgium, Europe. System is located in the hosting facilities that constantly monitor environmental conditions and provide 24/7 physical security. In addition to that, we backup all databases daily to ensure that they can be easily restored in case an adverse situation takes place.

Service Architecture

PassCamp services run in separate Linux containers on the Kubernetes cluster.

Database

Each backend service stores data in a separate database giving even higher protection on user’s data.

Transport Layer Encryption

PassCamp uses TLS for secure data transfer even though most of the user data is already encrypted with AES. This protocol protects the data from other party listening in to the network traffic and ensures that the user is connecting directly to our servers to protect against man-in-the-middle attacks.

Types of data stored

User information

- Email address
- First name
- Last name

Team information

- Title
- Address name
- Team owner ID

Encrypted data

Most of your data is securely encrypted on your device even before it reaches our servers. This data includes:

- User’s secret items (passwords, secure notes) encrypted using symmetric AES keys.
- AES symmetric keys which have been encrypted with user’s RSA public keys.
- RSA public keys and encrypted private keys.

Billing information

All billing information including credit card numbers and zip codes are stored in Stripe. Stripe has been audited by an independent PCI Qualified Security Assessor (QSA) and is certified as a PCI Level 1 Service Provider. This is the most stringent level of certification available in the payments industry.

Glossary

AES

AES (acronym of Advanced Encryption Standard) is a symmetric encryption algorithm. AES was designed to be efficient in both hardware and software, and supports a block length of 128 bits and key lengths of 128, 192, and 256 bits.

HMAC

HMAC (sometimes expanded as either Keyed-Hash Message Authentication Code) is a specific type of message authentication code (MAC) involving a cryptographic hash function and a secret cryptographic key.

PBKDF2

PBKDF2 (acronym of Password-Based Key Derivation Function 2) is a key derivation function with a sliding computational cost, aimed to reduce the vulnerability of encrypted keys to brute force attacks.

RSA

The RSA algorithm is the basis of a cryptosystem - a suite of cryptographic algorithms that are used for specific security services or purposes - which enables public key encryption and is widely used to secure sensitive data, particularly when it is being sent over an insecure network such as the internet.

SRP

SRP (acronym of Secure Remote Password) is an interactive protocol which allows a server to confirm that some client knows a password without revealing what the password is to an eavesdropper. In addition, the server does not hold the actual password: instead it stores a "verifier" created by the client. If the server's private data is revealed (by a server compromise), the verifier cannot be used directly to impersonate the client.

Salt

Salt is random data that is used as an additional input to a one-way function that "hashes" data. Salts are usually used to safeguard passwords in storage.

SHA-256

SHA-256 is one of the successor hash functions to SHA-1 (collectively referred to as SHA-2), and is one of the strongest hash functions available.