bitwarden | Help Center Article

ADMIN CONSOLE > LOGIN WITH SSO >

About Trusted Devices

View in the help center:
https://bitwarden.com/help/about-trusted-devices/

About Trusted Devices

SSO with trusted devices allows users to authenticate using SSO and decrypt their vault using a device-stored encryption key, eliminating the need to enter a master password. Trusted devices must either be registered in advance of the login attempt, or approved through a few different methods.

SSO with trusted devices gives business end users a passwordless experience that is also zero-knowledge and end-to-end encrypted. This prevents users from getting locked out due to forgotten master passwords and allows them to enjoy a streamlined login experience.

Start using trusted devices

To get started using SSO with trusted devices:

1. Setup SSO with trusted devices for your organization.
2. Provide administrators with information on how to approve device requests.
3. Provide end-users with information on how to add trusted devices.

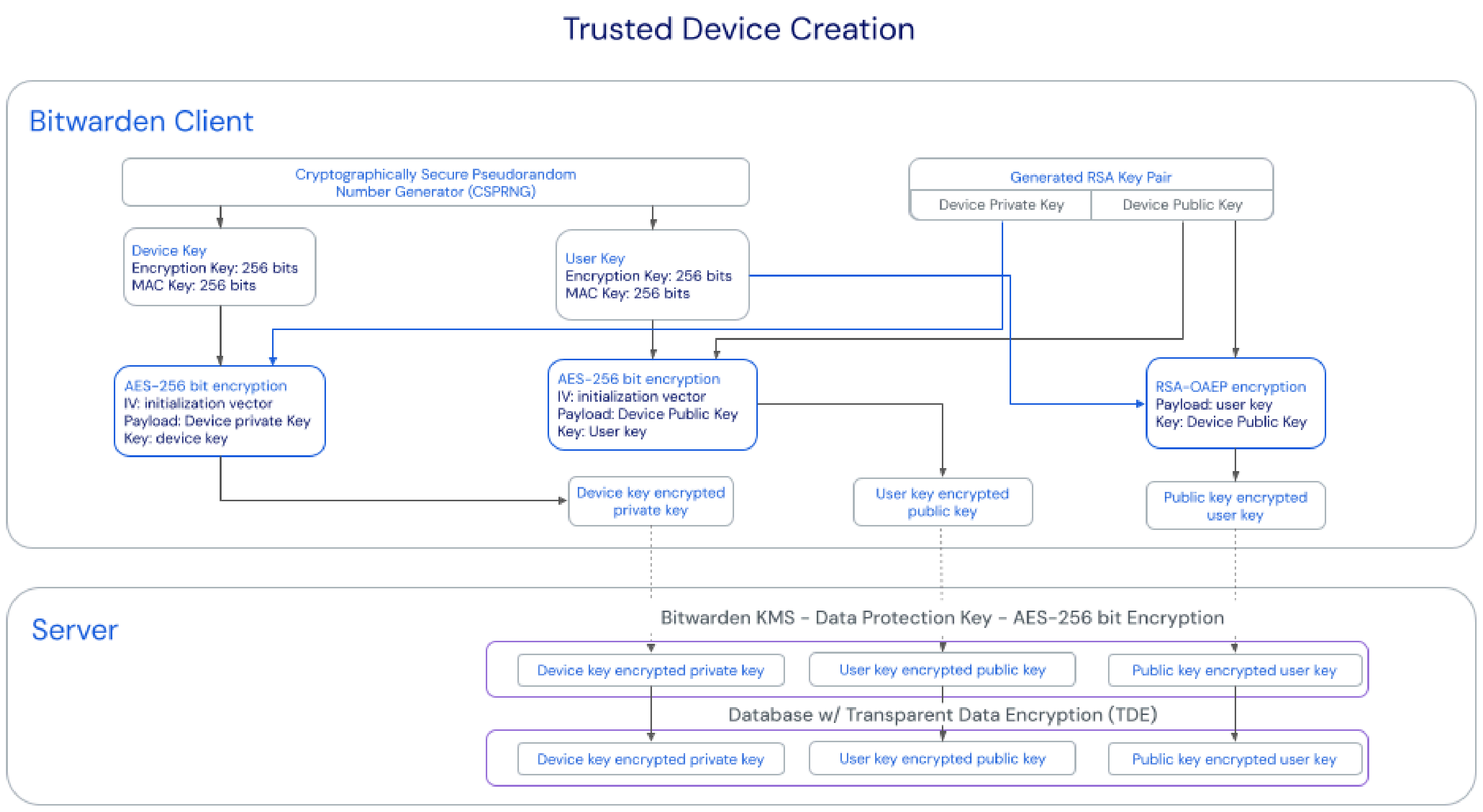
How it works

The following tabs describe encryption processes and key exchanges that occur during different trusted devices procedures:

⇒Onboarding

When a new user joins an organization, an **Account Recovery Key** (learn more) is created by encrypting their account encryption key with the **Organization Public Key**. Account recovery is required to enable SSO with trusted devices.

The user is then asked if they want to remember, or trust, the device. When they opt to do so:


Trusted Device Creation
Bitwarden Client
Cryptographically Secure Pseudorandom Number Generator (CSPRNG)
Generated RSA Key Pair
Device Private Key
Device Public Key
Device Key
Encryption Key: 256 bits
MAC Key: 256 bits
User Key
Encryption Key: 256 bits
MAC Key: 256 bits
AES-256 bit encryption
IV: initialization vector
Payload: Device private Key
Key: device key
AES-256 bit encryption
IV: initialization vector
Payload: Device Public Key
Key: User key
RSA-OAEP encryption
Payload: user key
Key: Device Public Key
Device key encrypted private key
User key encrypted public key
Public key encrypted user key
Server
Bitwarden KMS - Data Protection Key - AES-256 bit Encryption
Device key encrypted private key
User key encrypted public key
Public key encrypted user key
Database w/ Transparent Data Encryption (TDE)
Device key encrypted private key
User key encrypted public key
Public key encrypted user key

Create a trusted device

1. A new **Device Key** is generated by the client. This key never leaves the client.

2. A new RSA key pair, called the **Device Private Key** and **Device Public Key**, is generated by the client.

3. The user's account encryption key is encrypted with the unencrypted **Device Public Key** and the resultant value is sent to the server as the **Public Key-Encrypted User Key**.

4. The **Device Public Key** is encrypted with the user's account encryption key and the resultant value is sent to the server as the **User Key-Encrypted Public Key**.

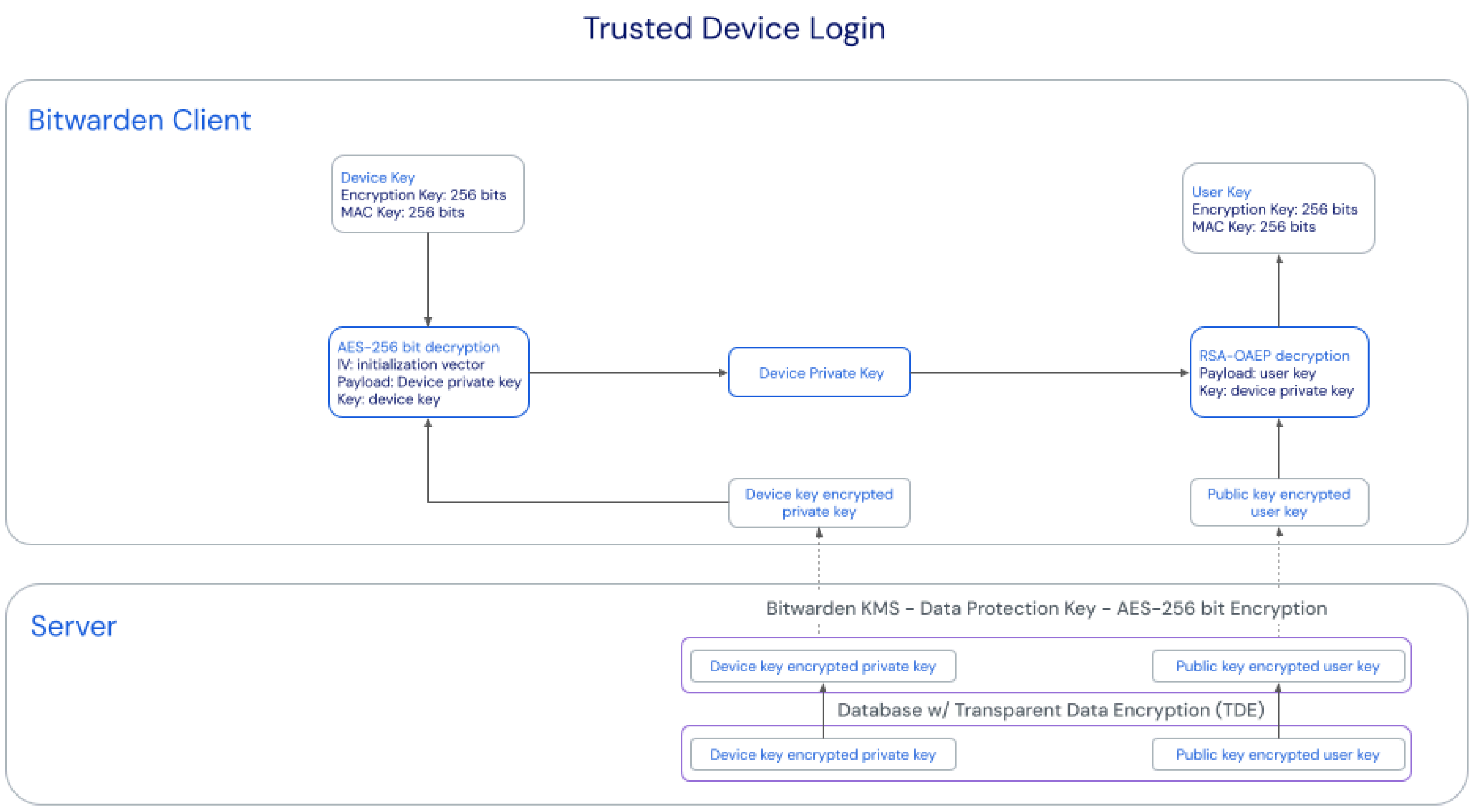
5. The **Device Private Key** is encrypted with the first **Device Key** and the resultant value is sent to the server as the **Device Key-Encrypted Private Key**.

The **Public Key-Encrypted User Key** and **Device Key-Encrypted Private Key** will, crucially, be sent from server to client when a login is initiated.

The **User Key-Encrypted Public Key** will be used should the user need to rotate their account encryption key.

⇒Logging in

When a user authenticates with SSO on an already-trusted device:


Trusted Device Login
Bitwarden Client
Device Key
Encryption Key: 256 bits
MAC Key: 256 bits
User Key
Encryption Key: 256 bits
MAC Key: 256 bits
AES-256 bit decryption
IV: initialization vector
Payload: Device private key
Key: device key
Device Private Key
RSA-OAEP decryption
Payload: user key
Key: device private key
Device key encrypted private key
Public key encrypted user key
Server
Bitwarden KMS - Data Protection Key - AES-256 bit Encryption
Device key encrypted private key
Public key encrypted user key
Database w/ Transparent Data Encryption (TDE)
Device key encrypted private key
Public key encrypted user key

Use a trusted device

1. The user's **Public Key-Encrypted User Key**, which is an encrypted version of the account encryption key used to decrypt vault data, is sent from the server to the client.

2. The user's **Device Key-Encrypted Private Key**, the unencrypted version of which is required to decrypt the **Public Key-Encrypted User Key**, is sent from the server to the client.

3. The client decrypts the **Device Key-Encrypted Private Key** using the **Device Key**, which never leaves the client.

4. The now-unencrypted **Device Private Key** is used to decrypt the **Public Key-Encrypted User Key**, resulting in the user's account encryption key.

5. The user's account encryption key decrypts vault data.

⇒Approving

When a user authenticates with SSO and opts to decrypt their vault with an un-trusted device (i.e. a **Device Symmetric Key** does not exist on that device), they are required to choose a method of approving the device and optionally trusting it for future use without further approval. What happens next depends on the selected option:

- **Approve from another device**:

    1. The process documented here is triggered, resulting in the client having obtained and decrypted the account encryption key.

    2. The user can now decrypt their vault data with the decrypted account encryption key. If they have chosen to trust the device, trust is established with the client as described in the **Onboarding** tab.

- **Request admin approval**:

    1. The initiating client POSTs a request, which includes the account email address, a unique **auth-request public key**[a], and an access code, to an Authentication Request table in the Bitwarden database.

    2. Administrators can approve or deny the request on the Device approvals page.

    3. When the request is approved by an administrator, the approving client encrypts the user's account encryption key using the **auth-request public key** enclosed in the request.

    4. The approving client then PUTs the encrypted account encryption key to the Authentication Request record and marks the request fulfilled.

    5. The initiating client GETs the encrypted account encryption key and **locally** decrypts it using the **auth-request private key**.

    6. Using the decrypted account encryption key, trust is established with the client as described in the **Onboarding** tab.

[a] - **Auth-request public** and **private keys** are uniquely generated for each passwordless login request and only exist for as long as the request does. Unapproved requests will expire after 1 week.

- **Approve with master password**:

    1. The users's account encryption key is retrieved and decrypted as documented in the Authentication and decryption section of the security whitepaper.

    2. Using the decrypted account encryption key, trust is established with the client as described in the **Onboarding** tab.

⇒Key rotation

ⓘ **Note**

Only users who have a master password can rotate their account encryption key. Learn more.

When a user rotates their account encryption key, during the normal rotation process:

1. The **User-Key Encrypted Public Key** is sent from the server to the client, and subsequently decrypted with the old account encryption key (a.k.a. **User Key**), resulting in the **Device Public Key**.

2. The user's new account encryption key is encrypted with the unencrypted **Device Public Key** and the resultant value is sent to the server as the new **Public Key-Encrypted User Key**.

3. The **Device Public Key** is encrypted with the user's new account encryption key and the resultant value is sent to the server as the new **User Key-Encrypted Public Key**.

4. Trusted device encryption keys for all other devices that are persisted to server storage are cleared for the user. This leaves only the three required keys (**Public Key-Encrypted User Key**, **User Key-Encrypted Public Key**, and **Device Key-Encrypted Private Key** which was not changed by this process) for that single device persisted to the server.

Any now-untrusted client will be required to re-established trust through one of the methods described in the **Approving** tab.

Keys used for trusted devices

This table provides more information about each key used in the procedures described above:

Key	Details
Device Key	AES-256 CBC HMAC SHA-256, 512 bits in length (256 bits for key, 256 bits for HMAC)
Device Private Key & Device Public Key	RSA-2048 OAEP SHA1, 2048 bits in length
Public Key-Encrypted User Key	RSA-2048 OAEP SHA1
User Key-Encrypted Public Key	AES-256 CBC HMAC SHA-256
Device Key-Encrypted Private Key	AES-256 CBC HMAC SHA-256

Impact on master passwords

While SSO with trusted devices eliminates the need for a master password, it doesn't in all cases eliminate the master password itself:

- If a user is onboarded **before** SSO with trusted devices is activated, their account will retain its master password.

- If a user is onboarded **after** SSO with trusted devices is activated and they select **Log in → Enterprise SSO** from the organization invite for JIT provisioning, their account will not have a master password. Should you change to the master password member decryption option, these users will be prompted to create a master password when they log in as long as they are still a member of the organization (learn more).

> ⚠ **Warning**
>
> For those accounts that do not have master passwords as a result of SSO with trusted devices, removing them from your organization will cut off all access to their Bitwarden account unless:
>
> 1. You assign them a master password using account recovery beforehand.
> 2. The user logs in at least once post-account recovery in order to fully complete the account recovery workflow.
>
> Additionally, users will not be able to re-join your organization unless the above steps are taken before they are removed from the organization. In this scenario, the user will be required to delete their account and be issued a new invitation to create an account and join your organization.
>
> Revoking access to the organization, but not removing them from the organization, will still allow them to log in to Bitwarden and access **only** their individual vault.

- If a user account is recovered using account recovery, their account will necessarily be assigned a master password. A master password cannot currently be removed from an account once it has one, so to avoid this outcome we recommend that you (i) instruct the user to export their data to a backup, (ii) completely delete the lost account, (iii) ask the user to re-onboard to your organization using trusted devices and (iv) once they've done so instruct them to import their backup.

Impact on other features

Depending on whether a master password hash is available in memory for your client, which is dictated by how your client application is initially accessed, it may exhibit the following behavior changes:

Feature	Impact
Verification	There are a number of features in Bitwarden client applications that ordinarily require entry of a master password in order to be used, including exporting vault data, changing two-step login settings, retrieving API keys, and more. If the user doesn't use a master password to access the client, **all these features** will replace master password confirmation with email-based TOTP verification.
Vault lock/unlock	Under ordinary circumstances, a locked vault can be unlocked using a master password. If the user doesn't use a master password to access the client, locked client applications can only be unlocked with a PIN or with biometrics. If neither PIN nor biometrics are enabled for a client application, the vault will always log out instead of lock. Unlocking and logging in will **always** require an internet connection.
Master password re-prompt	If the user does not unlock their vault with a master password, master password re-prompt will be disabled.

Feature	Impact
Changing email address	Users who do not have master passwords **will not** be able to change their email address.
CLI	Users who do not have master passwords **will not** be able to access Password Manager CLI.