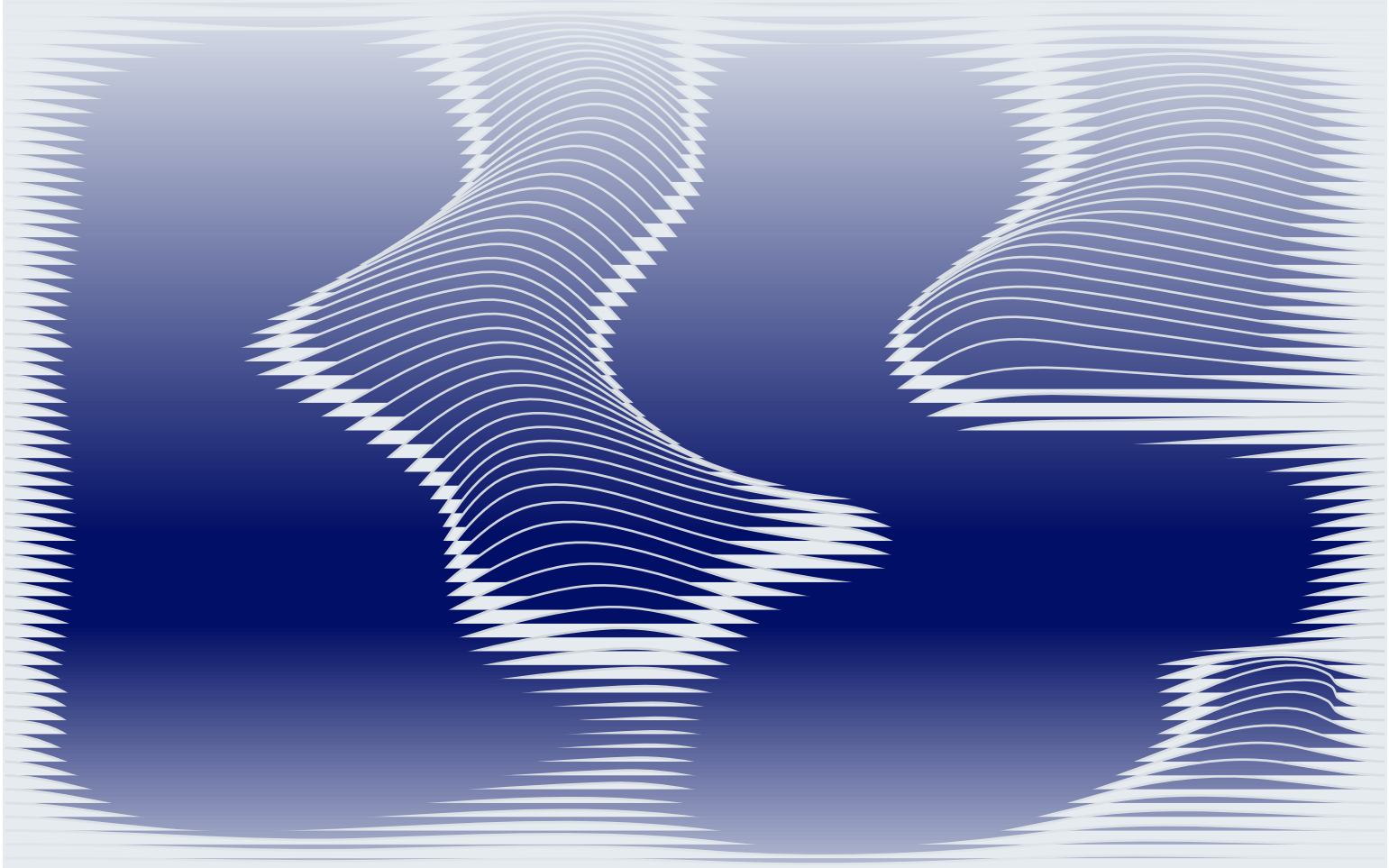
bitwarden | Help Center Article

SECURITY

Bitwarden Security Whitepaper



View in the help center:
https://bitwarden.com/help/bitwarden-security-white-paper/

Bitwarden Security Whitepaper

Overview

Everyone is more connected than ever

Internet-connected devices and services are more critical than ever in today's society. As more and more companies provide innovative software-as-a-service products that improve users' lives at home and at work, the number of credentials and machine secrets grow exponentially. As do the threats to their security.

Cybersecurity threats are high, but practices are low

Threats to user and customer data continue to rise. It's almost every week that a breach or ransomware attack makes the news, and those are only the incidents large enough to be published. In 2023, IBM reported that the average cost of a US data breach approaches $9.48 Million, taking into account investigation costs, legal fees, opportunity cost, and loss of customer trust.

Research from Verizon shows that compromised credentials account for 86% of data breaches. This includes the use of passwords that have been guessed, phished, or leaked in other breaches.

With such threats, one would expect that businesses armed their employees with as much training and tools as possible, but Bitwarden research shows that users aren't always following best practices, including 90% of respondents saying that they reused passwords.

Security experts recommend that users have a different, randomly generated password for every account. But how does one manage all those passwords? And how can good password habits be maintained across an organization?

Bitwarden helps secure individuals, businesses, and infrastructure secrets

Bitwarden offers a portfolio of security products to help secure everyone, prevent breaches, and ensure productivity.

Bitwarden Password Manager provides users the tools to create, store, and share passwords while maintaining the highest level of security. It is the easiest and safest way to store all of your logins, passwords, and other sensitive information while conveniently keeping them synced between all of your devices.

Bitwarden Secrets Manager empowers developers, DevOps, and IT teams to store, share, and automate machine secrets like authentication keys, database passwords, and API keys. The end-to-end encrypted secrets management solution supports the secure deployment of infrastructure and application code without the risk of exposing critical machine secrets.

Bitwarden Passwordless.dev provides APIs and tools needed for developers to implement FIDO2 WebAuthn based passkey authentication, the next generation of secure credential authentication, for websites and applications.

Maintaining security and compliance

Bitwarden solutions, software, infrastructure, and security processes have been designed from the ground up with a multi-layered, defense-in-depth approach. The Bitwarden Security and Compliance Program is based on the ISO27001 Information Security Management System (ISMS). Policies have been defined that govern security practices and processes and are continually updated to be consistent with applicable legal, industry, and regulatory requirements for services provided under the Terms of Service Agreement.

Bitwarden complies with industry-standard application security guidelines that include a dedicated security engineering team and include regular reviews of application source code and IT infrastructure to detect, validate, and remediate any security vulnerabilities.

This white paper provides an overview of Bitwarden security principles as well as links to additional documents that provide more detail in specific areas.

Bitwarden security principles

Protecting user data with Bitwarden products is a partnership between Bitwarden systems and employees, and users themselves. This section will cover, at a high-level, the key security measures Bitwarden utilizes and the tools Bitwarden makes available to users for protecting data stored in Bitwarden.

Key security measures

Bitwarden utilizes the following key security measures to protect data stored in Bitwarden:

End-to-end encryption: Lock your passwords and private information with end-to-end AES-CBC 256 bit encryption with HMAC authentication, salted hashing, and Key Derivation Functions such as PBKDF2 SHA-256 or Argon2id. All cryptographic keys are generated and managed by the client on your devices, and all encryption is done locally. See more details here.

Zero knowledge encryption: Bitwarden team members cannot see your passwords. Your data remains end-to-end encrypted with your individual email and master password. Bitwarden never stores and cannot access your master password or your cryptographic keys.

Secure password sharing: Bitwarden enables secure sharing and management of sensitive data with users across an entire organization. A combination of asymmetric and symmetric encryption protects sensitive information as it is shared.

Open source and source available code: The source code for all Bitwarden software products is hosted on GitHub and we welcome everyone to review, audit, and contribute to the Bitwarden codebase. Bitwarden source code is audited by reputable third-party security auditing firms as well as independent security researchers. Additionally, the Bitwarden Vulnerability Disclosure Program enlists the help of the hacker community at HackerOne to make Bitwarden more secure.

Privacy by design: Bitwarden stores all of your logins in an encrypted vault that syncs across all of your devices. Since it's fully encrypted before it ever leaves your device, only you have access to your data. Not even the team at Bitwarden can read your data (even if we wanted to).

Security Auditing: Third-party security reviews and assessments of applications and/or the platform are performed at a minimum of once per year.

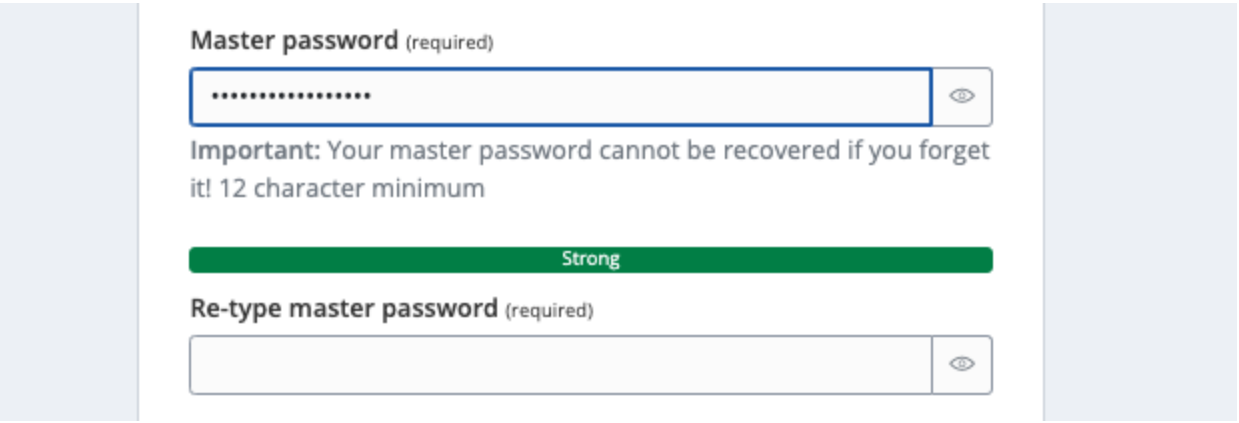
Compliance: Bitwarden complies with AICPA SOC2 Type 2 / Data Privacy Framework, GDPR, and CCPA regulations. Learn more.

Security tools for users

The following tools are provided by Bitwarden, and must be acted on by individual users and businesses, to optimize account protection and lockout avoidance:

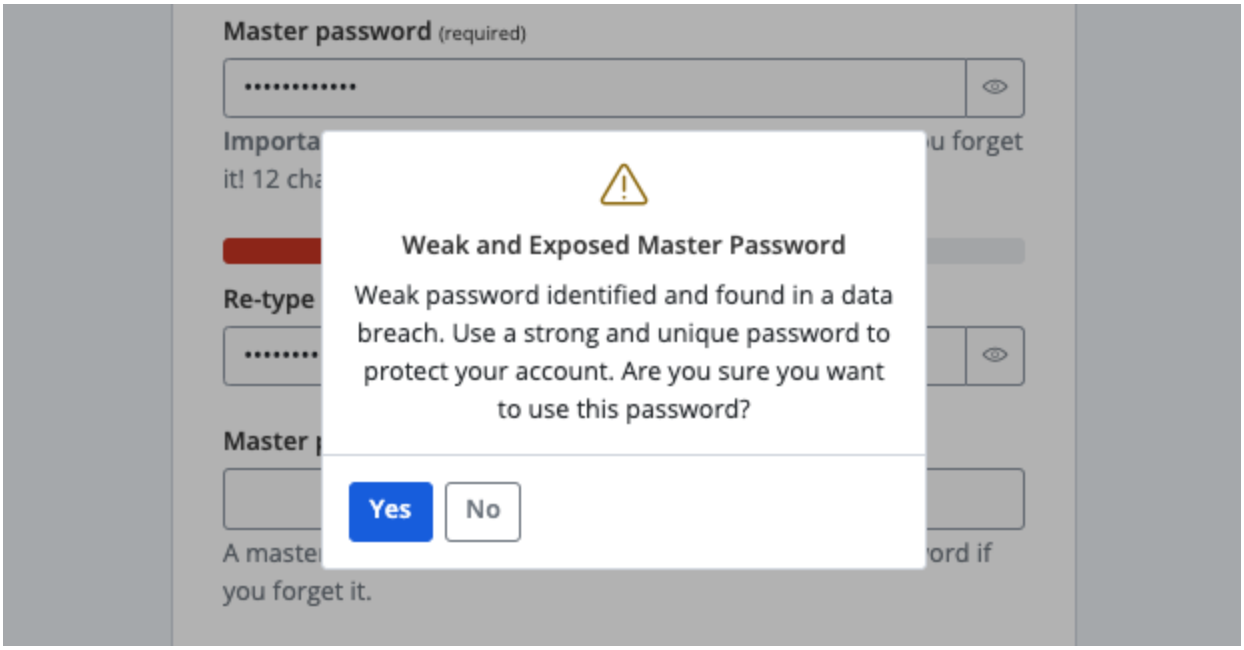
Master passwords

User data protection in Bitwarden begins the moment a user creates an account and a master password. A master password is the token a user uses to access their vault, where sensitive data is stored. Users should create their accounts with a strong master password and Bitwarden includes a password strength meter as a guide to help users do so:



Password strength meter

Users are warned when they try to sign up with a weak master password, and are also given the option to check known data breaches for the master password using an integration with Have I Been Pwned (HIBP):



Weak or exposed master password

It is very important that users never forget their master passwords. Master passwords are:

- Cleared or marked for removal from memory after usage.
- Never transmitted over the internet to Bitwarden servers.
- Unable to be seen, read, or reverse engineered by anyone at Bitwarden.

Because of this, and the fact that your data is fully encrypted and/or hashed before ever leaving your local device, forgetting a master password **will** result in a user being locked out of their account unless they have emergency access or account recovery active, both of which will be covered later in this paper.

> **Tip**
> Users can change their master password from the Bitwarden web app. Learn how.

Alternative log in methods

Bitwarden clients offer the following alternative methods of authentication. Some of these methods may also be used for decryption on log in:

- **Log in with device**: Initiate an authentication request from a Bitwarden client and complete authentication by approving the request on a device you're already logged in to. Learn how it works.
- **Log in with passkeys**: Use a passkey to log in to a Bitwarden client and, if the passkey is PRF-capable, use it to decrypt your vault data. Learn how it works.
- **SSO with 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. Learn how it works.

Two-step login

Two-step login (also called "two-factor authentication" or "2FA") is an extra layer of security for online accounts, designed to protect access to Bitwarden even if someone has the master password. When two-step login is active, users are required to complete a secondary step while logging into Bitwarden, like using a FIDO2 security key or an authenticator app to confirm the login attempt. As a best practice, **Bitwarden recommends all users activate and use two-step login**.

Bitwarden provides users a recovery code that they can use to turn off two-step login in the event a secondary device is lost, for example if a YubiKey goes missing. **Users should retrieve and save their recovery code immediately after activating the feature**, as Bitwarden employees and systems have no way of deactivating two-step login on users' behalf.

Learn more about the available two-step login methods, using multiple methods, and what to do in the event of a lost secondary device.

Emergency access

Premium users, including members of paid organizations (Families, Teams, or Enterprise) can designate trusted emergency contacts who may request access to their vault in cases of emergency. Trusted emergency contacts can be assigned either view-only or takeover access to users' accounts.

Emergency access uses asymmetric encryption to allow users to give a trusted emergency contact permission to access vault data in a zero knowledge environment:

> **Note**
> The following information references encryption key names and processes that are covered in the Hashing, key derivation, and encryption section. Consider reading that section first.

1. A Bitwarden user (the grantor) invites another Bitwarden user to become a trusted emergency contact (the grantee). The invitation (valid for only five days) specifies a user access level and includes a request for the grantee's **RSA Public Key**.
2. Grantee is notified of the invitation via email and accepts the invitation to become a trusted emergency contact. On acceptance, the grantee's **RSA Public Key** is stored with the user record.
3. Grantor is notified of the invitation's acceptance via email and confirms the grantee as their trusted emergency contact. On confirmation, the grantor's **User Symmetric Key** is encrypted using the grantee's **RSA Public Key** and stored with the invitation. Grantee is notified of confirmation.

4. An emergency occurs, resulting in grantee requiring access to grantor's vault. Grantee submits a request for emergency access.
5. Grantor is notified of the request via email. The grantor may manually approve the request at any time, otherwise the request is bound by a grantor-specified wait time. When the request is approved or the wait time lapses, the **Public Key-encrypted User Symmetric Key** is delivered to the grantee for decryption with the grantee's **RSA Private Key**.
6. Depending on the specified user access level, the grantee will either:
   - Obtain view/read access to items in the grantor's vault.
   - Be asked to create a new master password for the grantor's vault.

Account recovery

Account recovery allows designated administrators of Enterprise organizations to recover member accounts and restore access in the event that an employee forgets their master password. Businesses may also wish to use account recovery to reclaim ownership of a member's account when an employment relationship is ended.

> **Note**
> The following information references encryption key names and processes that are covered in the Hashing, key derivation, and encryption section. Consider reading that section first.

When an organization member enrolls in account recovery, that user's account encryption key (a.k.a. **User Symmetric Key**) is encrypted with the organization's RSA Public Key. The result is stored as the **Account Recovery Key**.

When a recovery action is taken:

1. The organization **RSA Private Key** is decrypted with the **Organization Symmetric Key**.
2. The user's **Account Recovery Key** is decrypted with the decrypted **RSA Private Key**, resulting in the **User Symmetric Key** (referred to as "account encryption key" in-product).
3. The **User Symmetric Key** is encrypted with a new **Master Key** and a new **Master Password Hash** is seeded from the new master password, both the **Master Key-encrypted User Symmetric Key** and the **Master Password Hash** replace pre-existing server-side values.
4. The user's **User Symmetric Key** is encrypted with the organization's **RSA Public Key**, replacing the previous **Account Recovery Key** with a new one.

Hashing, key derivation, and encryption

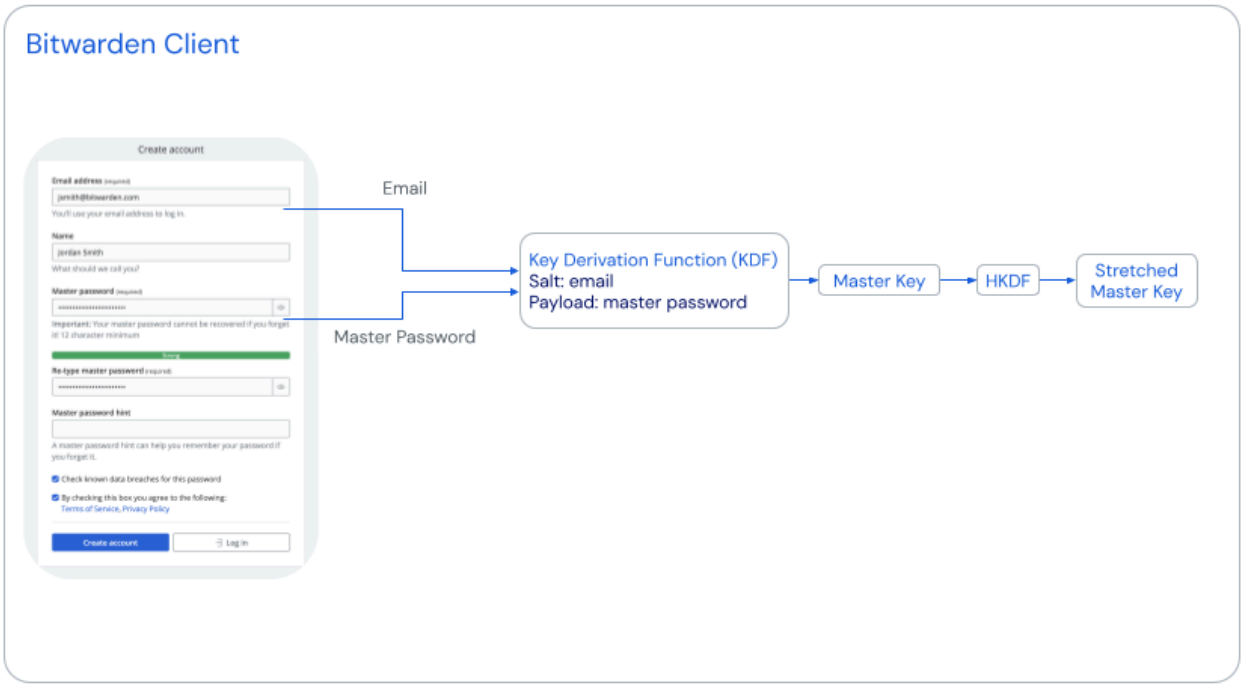
This section will cover the cryptographic processes that are implemented when a user creates a Bitwarden account and subsequently logs in to access their data:

Account creation

When an account is created, Bitwarden uses Password-Based Key Derivation Function 2 (PBKDF2) with 600,000 iteration rounds to stretch the user's master password with a salt of the user's email address.

> **Note**
> Though user accounts are initiated with PBKDF2, users may elect to change their key derivation function to Argon2id after the account has been created. Learn how to change the KDF algorithm.

The resulting salted value is the 256-bit **Master Key**. The **Master Key** is then again stretched to 512-bits using HMAC-based Extract-and-Expand Key Derivation Function (HKDF), resulting in the **Stretched Master Key**. The **Master Key** and **Stretched Master Key** are never stored on or transmitted to Bitwarden servers:

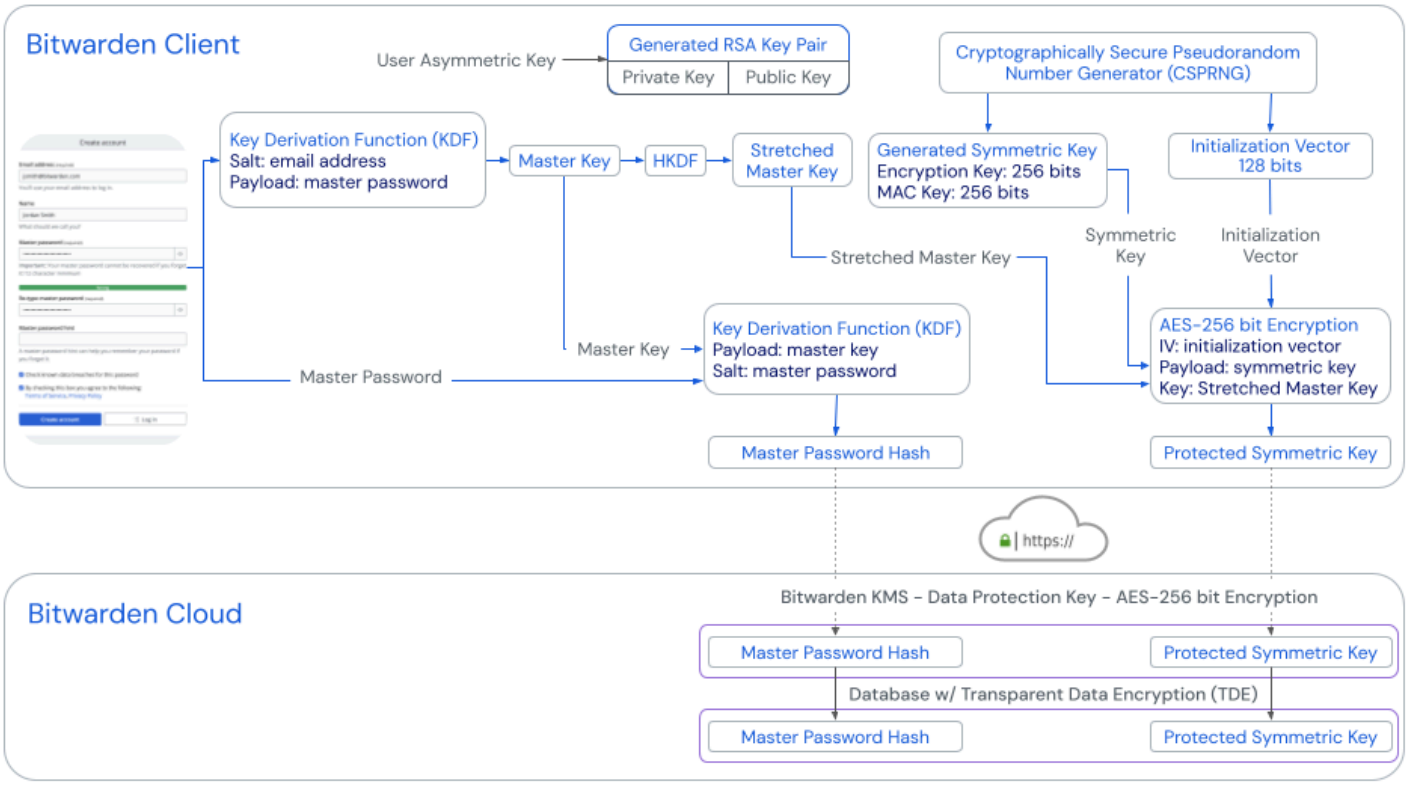


Password-based key derivation

Next, a 512-bit **Generated Symmetric Key** and 128-bit **Initialization Vector** are created using a Cryptographically Secure Pseudorandom Number Generator (CSPRNG). The **Generated Symmetric Key** is encrypted with AES-256 bit encryption using the **Stretched Master Key** and **Initialization Vector**. The result is called the **Protected Symmetric Key**, and is the main key associated with the user. The **Protected Symmetric Key** is sent to the Bitwarden server upon account creation and sent back to Bitwarden client applications upon syncing.

An asymmetric key pair is also created when the user registers their account. This **Generated RSA Key Pair** is used when the user creates an organization and in processes like emergency access that can be used to share data between users.

Finally, a **Master Password Hash** is generated using PBKDF-SHA256 with a payload of the **Master Key** and with a salt of the master password. The **Master Password Hash** is sent to the Bitwarden server upon account creation and login, and used to authenticate the user account. Once reaching the server, the **Master Password Hash** is hashed again using PBKDF2-SHA256 with a random salt and 600,000 iterations:



Bitwarden password hashing, key derivation, and encryption

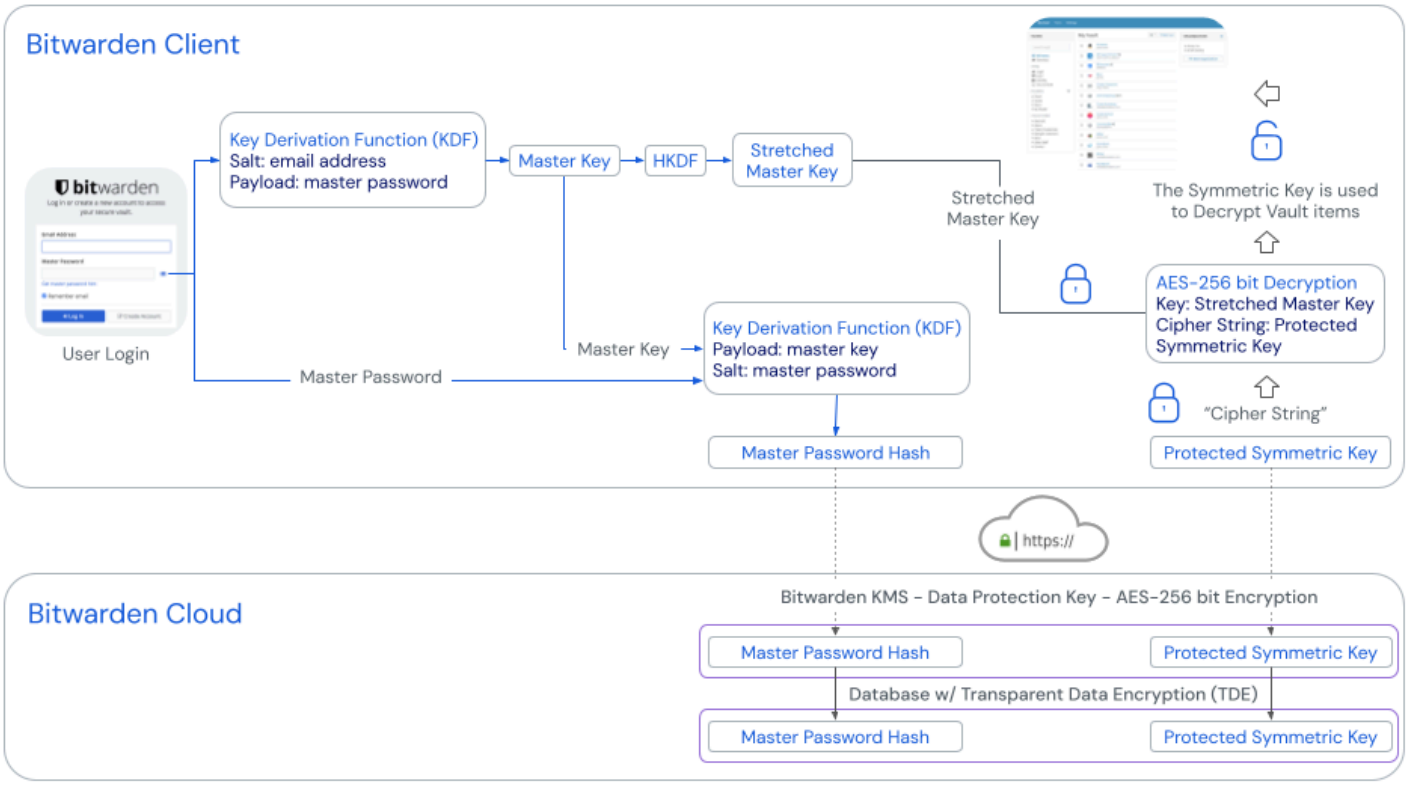
Authentication and decryption

Users are required to enter an email address and, typically, a master password in order to log in to a Bitwarden account.

When they do so, Bitwarden uses Password-Based Key Derivation Function 2 (PBKDF2) with a default of 600,000 iteration rounds to stretch the master password with a salt of the account email address. The resulting salted value is the 256-bit **Master Key**. A **Master Password Hash**, generated using PBKDF-SHA256 with a payload of the **Master Key** and with a salt of the master password, is sent to the server for authentication by comparing the hash to that which is stored server-side.

> **Note**
>
> Though user accounts are initiated with PBKDF2, users may elect to change their key derivation function to Argon2id after the account has been created. Learn how to change the KDF algorithm.

Concurrently, the **Master Key** is stretched to 512-bits in length using HMAC-based Extract-and-Expand Key Derivation Function (HKDF), resulting in the **Stretched Master Key**. The **Protected Symmetric Key**, which is stored server-side and retrieved by the client, is decrypted using this **Stretched Master Key**. The resultant **Symmetric Key** is used by the client to decrypt vault data. This decryption is done entirely on the Bitwarden client. Master passwords and **Stretched Master Keys** are never stored on or transmitted to Bitwarden servers:



An overview of user login

Bitwarden does not keep the master password itself stored locally or in-memory on the Bitwarden client. Your account encryption key (**User Symmetric Key**) is kept in memory while the app is unlocked in order to decrypt vault data.

When the Bitwarden client is locked, your encryption keys and vault data are purged as aggressively as possible from memory. After a certain period of inactivity on the lock screen, we reload application processes or perform other cleanup operations to ensure that any leftover managed memory is also purged or expect underlying systems to purge that memory when it is outside our control. We do our best to ensure that any data that may be in memory for the application to function is only held in memory for as long as you need it and that memory is cleaned up appropriately whenever the application is locked. We regularly review the Bitwarden application's memory state and adjust processes wherever possible to clear sensitive contents while locked.

Rotating the account encryption key

During a password change operation, users have the option to rotate (i.e. change) their **User Symmetric Key** (referred to as "account encryption key" in-product). Rotating this key is a good idea if users believe that their previous master password was compromised or that the data they've stored in Bitwarden was stolen from one of their devices.

> **⚠ Warning**
>
> Rotating the account's encryption key is a sensitive operation, which is why it is not a default option when changing a master password. A key rotation involves generating a new, random encryption key for the account and **re-encrypting all vault data** using this new key. See additional details in this article.

Variations

This section will cover variations to encryption processes when users are using Log in with device, Log in with passkeys, or SSO with trusted devices.

Log in with device

When logging in with a device is initiated:

1. The initiating client sends a request to the Bitwarden server, which includes the account email address, a unique **Auth-request Public Key**[a], and an access code.
2. Registered devices, meaning mobile or desktop apps that are logged in and have a device-specific GUID stored with the Bitwarden server, are provided the request.
3. When the request is approved, the approving client encrypts the account's **Master Key** and **Master Password Hash** using the **Auth-request Public Key** enclosed in the request.
4. The approving client then sends the **Encrypted Master Key** and **Encrypted Master Password Hash** to the Bitwarden server and the request is marked as fulfilled.
5. The initiating client requests the **Encrypted Master Key** and **Encrypted Master Password Hash** from the Bitwarden server.
6. The initiating client then **locally** decrypts the **Master Key** and **Master Password Hash** using the **Auth-request Private Key**.
7. The initiating client then uses the access code and fulfilled authentication request to authenticate the user with the Bitwarden Identity service.

[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. Requests expire and are purged from the Bitwarden server every 15 minutes if they aren't approved or denied.

Log in with passkeys

The following describes the mechanics of logging in with passkeys when users' passkeys are set up for encryption. Users may opt to not use their passkeys for encryption instead.

When a passkey is registered for log in to Bitwarden:

1. A **Passkey Public and Private Key Pair** is generated by the authenticator via the WebAuthn API. This key pair, by definition, is what constitutes your passkey. Attestation options, such as what encryption algorithm to use, and provided by Bitwarden to the authenticator.
2. A **PRF Symmetric Key** is generated by the authenticator via the WebAuthn API's PRF extension. This key is derived from an **internal secret** unique to your passkey and a **salt** provided by Bitwarden.
3. A **PRF Public and Private Key Pair** is generated by the Bitwarden client. The PRF public key encrypts your **User Symmetric Key** (referred to as "account encryption key" in-product), which your client will have access to by virtue of being logged in and unlocked, and the resulting **PRF-Encrypted User Symmetric Key** is sent to the server.
4. The **PRF Private Key** is encrypted with the **PRF Symmetric Key** (see Step 2) and the resulting **PRF-Encrypted Private Key** is sent to the server.
5. Your client sends data to Bitwarden servers to create a new passkey credential record for your account. If your passkey is registered with support for vault encryption and decryption, this record includes:
   - The passkey name
   - The Passkey Public Key
   - The PRF Public Key
   - The PRF-Encrypted User Symmetric Key
   - The PRF-Encrypted Private Key

Your **Passkey Private Key**, which is required to accomplish authentication, only ever leaves the client in an encrypted format.

When a passkey is used to log in and, specifically, to decrypt your vault data:

1. Using WebAuthn API public key cryptography, your authentication request is asserted and affirmed.
2. Your **PRF-Encrypted User Symmetric Key** (referred to as "account encryption key" in-product) and **PRF-Encrypted Private Key** are sent from the server to your client.
3. Using the same **salt** provided by Bitwarden and the **internal secret** unique to your passkey, the **PRF Symmetric Key** is re-created locally.
4. The **PRF Symmetric Key** is used to decrypt your **PRF-Encrypted Private Key**, resulting in your **PRF Private Key**.
5. The **PRF Private Key** is used to decrypt your **PRF-Encrypted User Symmetric Key**, resulting in your **User Symmetric Key**. This is used to decrypt your vault data.

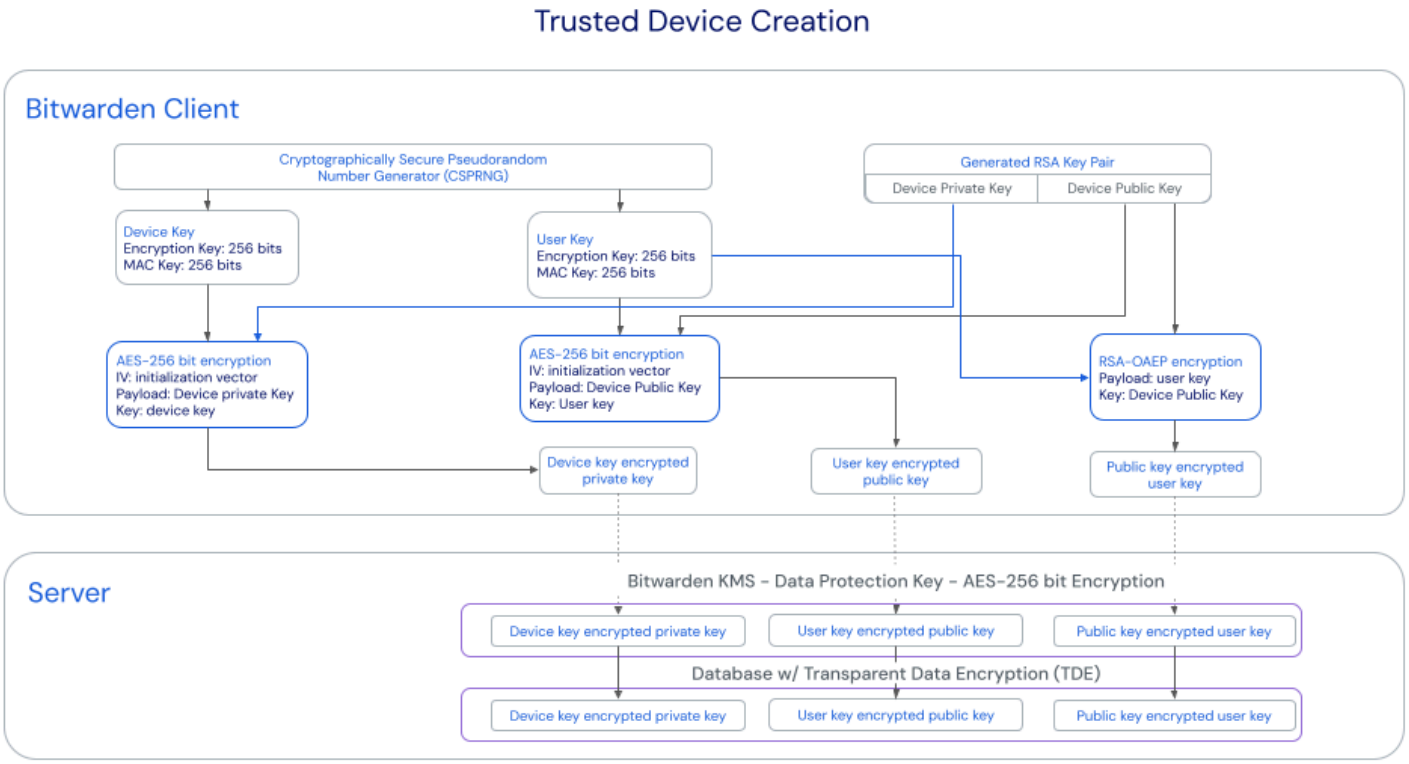
SSO with trusted devices

The following sections 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:



Create a trusted device

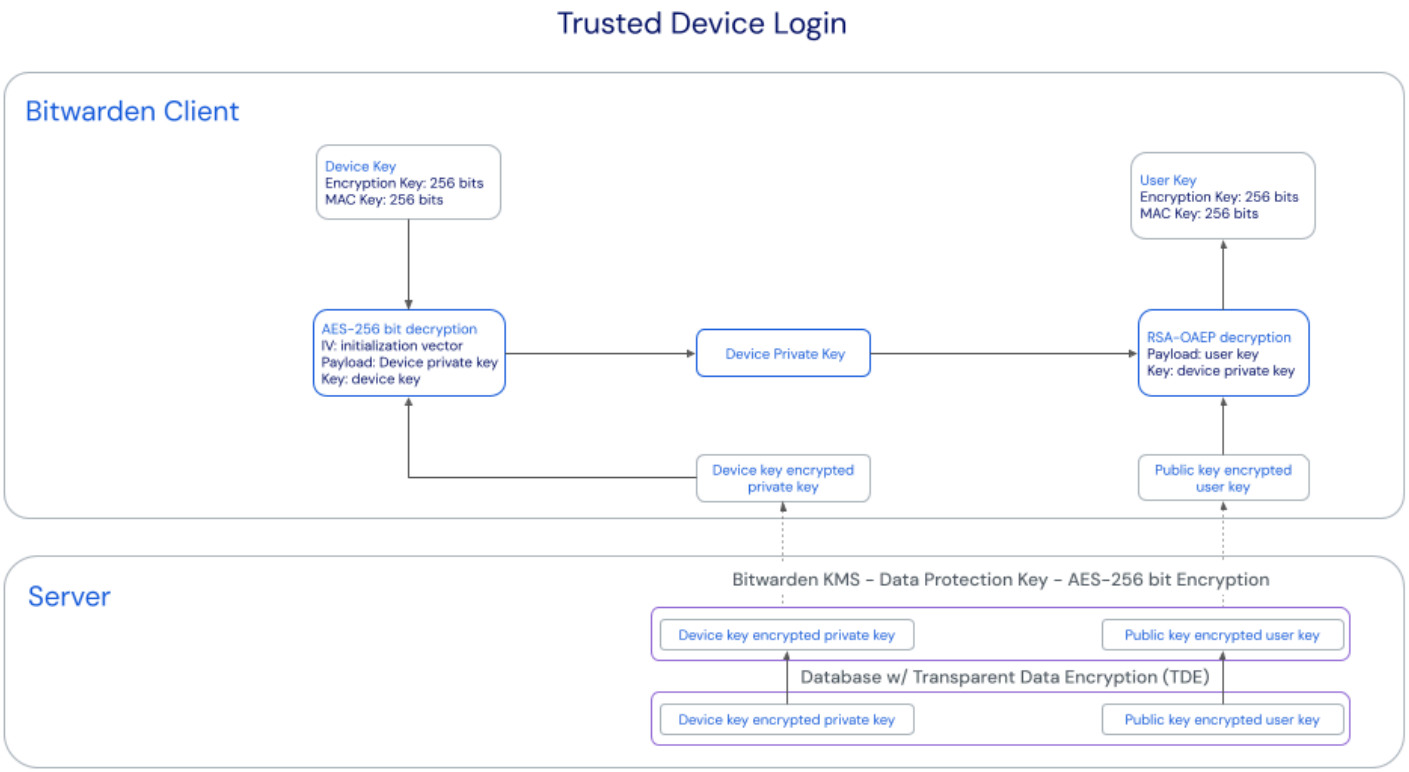
1. A new **Device Key** is generated by the client. This key never leaves the client.
2. A new RSA key pair, called the **Device Private Key** and **Device Public Key**, is generated by the client.
3. The user's account encryption key is encrypted with the unencrypted **Device Public Key** and the resultant 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:



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**[α], 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.

[α] – **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**.

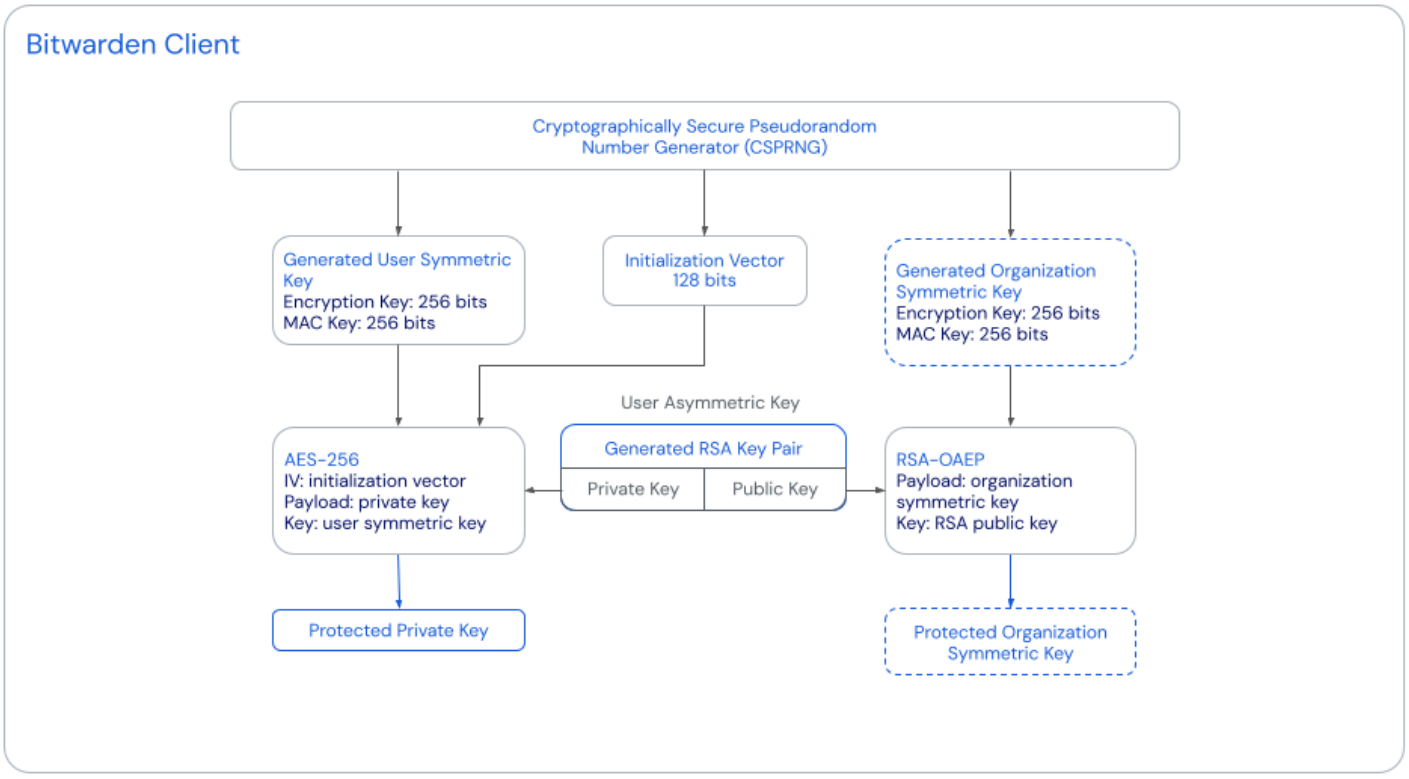
2. The user's new account encryption key is encrypted with the unencrypted **Device Public Key** and the resultant value is sent to the server as the new **Public Key-Encrypted User Key**.
3. The **Device Public Key** is encrypted with the user's new account encryption key and the resultant value is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

Sharing data between users

Collaboration is one of the leading benefits of using a password manager. In order to enable sharing, you need to first create an organization. A Bitwarden organization is an entity that relates users together that want to share items. An organization could be a family, team, company, or any other type of group that desires to share data.

This section will cover the cryptographic processes that are implemented to ensure a secure, end-to-end, zero knowledge encryption method for sharing data as well as the additional security measures implemented to ensure control of your data:



Organization key protection and exchange

When you create an organization

When you create an organization, a Cryptographically Secure Pseudorandom Number Generator (CSPRNG) is used to generate the **Organization Symmetric Key**. This key is what's used to encrypt vault data owned by the organization, therefore sharing data with

organization members requires securely providing access to the **Organization Symmetric Key**. The unprotected **Organization Symmetric Key** is never stored on Bitwarden servers.

As soon as the **Organization Symmetric Key** is created, RSA-OAEP is used to encrypt it with the organization creator's **RSA Public Key**.

> **Note**
>
> A **RSA Key Pair** is generated for every user upon account creation, regardless of whether they are an organization member or not, so this key will already exist prior to organization creation. The **RSA Private Key**, the use for which is described below, is stored encrypted with the user's **User Symmetric Key**, so users must be fully logged in to gain access to it.

The resultant value of this operation is referred to as the **Protected Organization Symmetric Key** and is sent to Bitwarden servers.

When the organization creator, or any organization member, logs in to their account, the client application uses the decrypted **RSA Private Key** to decrypt the **Protected Organization Symmetric Key**, resulting in the **Organization Symmetric Key**. Using this, organization-owned vault data is decrypted locally.

When users join an organization

The process for subsequent users joining an organization is quite similar, however some differences are worth noting.

First, an established member of the organization, specifically someone with permission to onboard other users, confirms the user to the organization. This established member, by virtue of having already logged in to their account and gone through the organization data decryption process described in the previous section, has access to the decrypted **Organization Symmetric Key**.

So, when the new user is confirmed, the established member's client reaches out to Bitwarden servers, retrieves the new user's **RSA Public Key**, which is stored on Bitwarden servers at the time of account creation, and encrypts the decrypted **Organization Symmetric Key** with it. This results in a new **Protected Organization Symmetric Key** that is sent to Bitwarden servers and stored for the new member.

> **Note**
>
> Each **Protected Organization Symmetric Key** is unique to its user, but each will decrypt to the same required **Organization Symmetric Key** when decrypted with its specific user's **RSA Private Key**.

When the new user logs in to their account, the client application uses the decrypted **RSA Private Key** to decrypt the new **Protected Organization Symmetric Key**, resulting in the raw **Organization Symmetric Key**. Using this, organization-owned vault data is decrypted locally.

Additional security measures

Access controls, permissions, and roles

Bitwarden organizations use collections, projects, and groups to logically group together vault data and users:

- **Collections & projects**: Logically organize your vault data into discrete units to help ensure members are getting access to all and only the resources they need.
- **Groups**: Logically organize your members into discrete units to help ensure that everyone is getting access to everything and only what they need.
- **Member roles**: Assign roles to members to provide them access to the appropriate level of tools within the context of your organization.

- **Permissions**: Designate what actions your members are allowed to take on the vault data they've been granted access to.

Event logs

Event logs contain time-stamped, detailed information about what actions or changes have occurred within an organization. These logs are helpful with researching changes in credentials or configuration and are very useful for audit trail investigation and troubleshooting purposes. Event logs are available for Teams and Enterprise organizations for both Password Manager and Secrets Manager. Learn more about event logs.

Teams and Enterprise organizations may also use the Bitwarden public API to gather more data for their event logs.

SIEM integrations

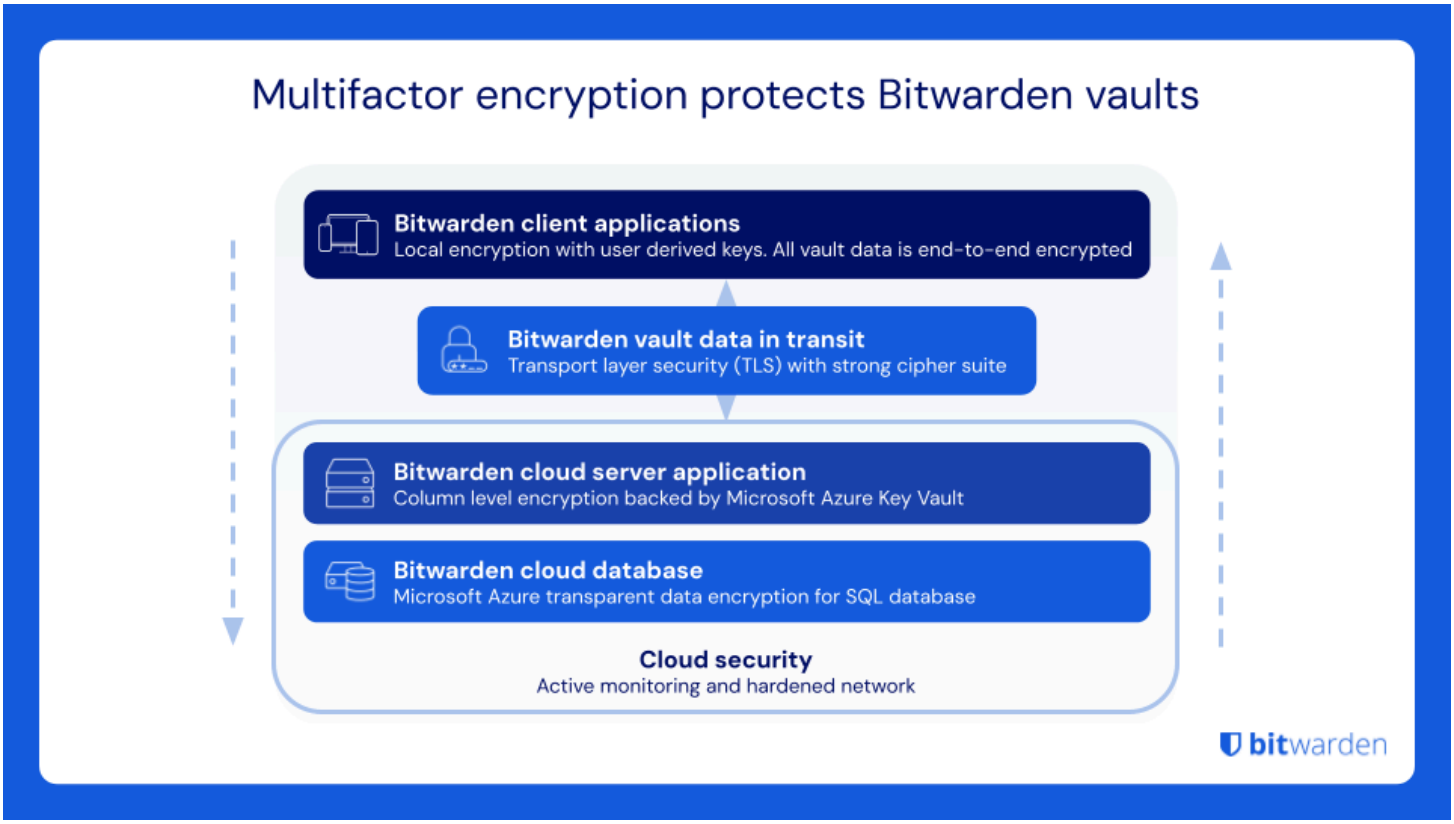
Several Security Information and Event Management (SIEM) integrations are available for Bitwarden:

- Splunk
- Panther
- Elastic

For other SIEM systems, a combination of data from the API and CLI may be used to gather data. This process is outlined here.

Data protection

This section will cover the measures taken to ensure that data remains secure:



Multifactor encryption

How vault data is encrypted

All vault data (logins, cards, identities, notes, and secrets) are protected with end-to-end-encryption. Data that you choose to store in Bitwarden is first stored as an object called a Cipher. Ciphers are encrypted locally when a vault item is created, edited, or imported, using a unique, random, 64-byte **Cipher Key**. Each **Cipher Key** is encrypted with either the **User Symmetric Key** or the **Organization Symmetric Key**, depending on whether the item is individually- or organizationally-owned, before being sent to Bitwarden servers. These encryption operations are performed entirely on the Bitwarden client application.

When a user logs in to Bitwarden, the client gains access to their **User Symmetric Key** by decrypting their **Protected Symmetric Key** using the **Stretched Master Key**. If they're a member of an organization, the client gains access to the **Organization Symmetric Key** through their **RSA Private Key**. With one of these keys, **Cipher Keys** are locally decrypted and the resultant value is used to decrypt individual or organization vault data.

When a user rotates their account encryption Key, here referred to as their **User Symmetric Key**, each existing **Cipher Key** is re-encrypted with the new **User Symmetric Key**.

> **Note**
>
> In the case of attachments, the **Cipher Key** is used to encrypt the attachment's metadata, specifically the file name and size. The **Cipher Key** is also used to encrypt the **Attachment Key**, which in turn is used to encrypt the attachment data itself.

Vault health reports

Vault health reports can be used to evaluate the security of the data stored in Bitwarden Password Manager. Reports, for example the Reused Passwords and Weak Passwords reports, are run locally on the Bitwarden client application. This allows offending items to be identified without Bitwarden ever having access to unencrypted versions of this data. Learn more about the available vault health reports.

Data protection in transit

Bitwarden uses TLS/SSL to secure communications between Bitwarden clients and user devices to the Bitwarden cloud. Bitwarden's TLS implementation uses X.509 certificates for server authentication and key exchange and a strong cipher suite for bulk encryption. Bitwarden servers are configured to reject weak ciphers and protocols.

Bitwarden also implements HTTP Security headers such as HTTP Strict Transport Security (HSTS), which will force all connections to use TLS. This additional layer of protection with HSTS mitigates the risks of downgrade attacks and misconfiguration.

Data protection at rest

Bitwarden always encrypts and/or hashes your data on your local device before it is sent to the cloud servers for syncing. Bitwarden servers are only used for storing and synchronizing encrypted vault data. It is not possible to get your unencrypted data from the Bitwarden cloud servers. Specifically, Bitwarden uses AES 256-bit encryption as well as PBKDF2 SHA-256 or Argon2id to secure your data. Passkeys, when stored in the vault, are generated using the ES256 algorithm.

AES is a standard in cryptography and used by the U.S. government and other government agencies around the world for protecting top-secret data. With proper implementation and a strong encryption key (i.e your master password), AES is considered unbreakable.

PBKDF2 SHA-256 or Argon2id are used to derive the encryption key from your master password. Then this key is salted and hashed for authenticating with the Bitwarden servers. The default iteration count used with PBKDF2 is 600,000 iterations on the client (this client-side iteration count is configurable from your account settings).

> **Note**
>
> Though user accounts are initiated with PBKDF2, users may elect to change their key derivation function to Argon2id after the account has been created. Learn how to change the KDF algorithm.

The Bitwarden cloud database stores your encrypted vault and is hosted within the secure Microsoft Azure cloud infrastructure. It is configured with an encryption-at-rest technology provided by Azure called Transparent Data Encryption (TDE). TDE performs real-time encryption and decryption of the entire Bitwarden cloud database, associated backup data, and transaction log files when they're not in-use. Azure handles the encryption keys for TDE, which only authorized Bitwarden server components are able to access. Read more about Azure's Transparent Data Encryption here.

Additionally, Bitwarden server applications perform their own encryption of sensitive database columns related to your user account. Master password hashes and protected user keys are encrypted on the fly as they move in and out of the Bitwarden cloud database. These column-level encryption operations are performed with keys that Bitwarden manages in a strictly controlled key management service (KMS).

Learn more: How end-to-end encryption paves the way for zero knowledge and What encryption is being used

Data types and data retention

Bitwarden processes two kinds of user data to deliver the Bitwarden Service: (i) Vault Data and (ii) Administrative Data.

(i) Vault Data

Vault Data includes all information stored within accounts to the Bitwarden Service and may include Personal Information. If we host the Bitwarden Service for you, we will host Vault Data. Vault Data is encrypted using secure cryptographic keys under your control. Bitwarden cannot access Vault Data.

Data Retention of Vault Data: You may add, modify, and delete Vault Data at any time.

(ii) Administrative Data

Bitwarden obtains Personal Information in connection with your account creation, usage of the Bitwarden Service and support, and payments for the Bitwarden Service such as names, emails address, phone and other contact information for users of the Bitwarden Service and the number of items in your Bitwarden Service account ("Administrative Data"). Bitwarden uses Administrative Data to provide the Bitwarden Service to you. We retain Administrative Data for as long as you are a customer of Bitwarden and as required by law. If you terminate your relationship with Bitwarden, we will delete your Personal Information in accordance with our data retention policies.

When you use the website or communicate with us (e.g., via email) you will provide, and Bitwarden will collect certain Personal Information such as:

- Name
- Business name and address
- Business telephone number
- Email address
- IP-address and other online identifiers
- Any customer testimonial you have given us consent to share.
- Information you provide to the Site's Interactive Areas, such as fillable forms or text boxes, training, webinars or event registration.

- Information about the device you are using, comprising the hardware model, operating system and version, unique device identifiers, network information, IP address, and/or Bitwarden Service information when interacting with the Site.
- If you interact with the Bitwarden Community or training, or registered for an exam or event, we may collect biographical information and the content that you share.
- Information gathered via cookies, pixel tags, logs, or other similar technologies.

Please refer to the Bitwarden Privacy Policy for additional information.

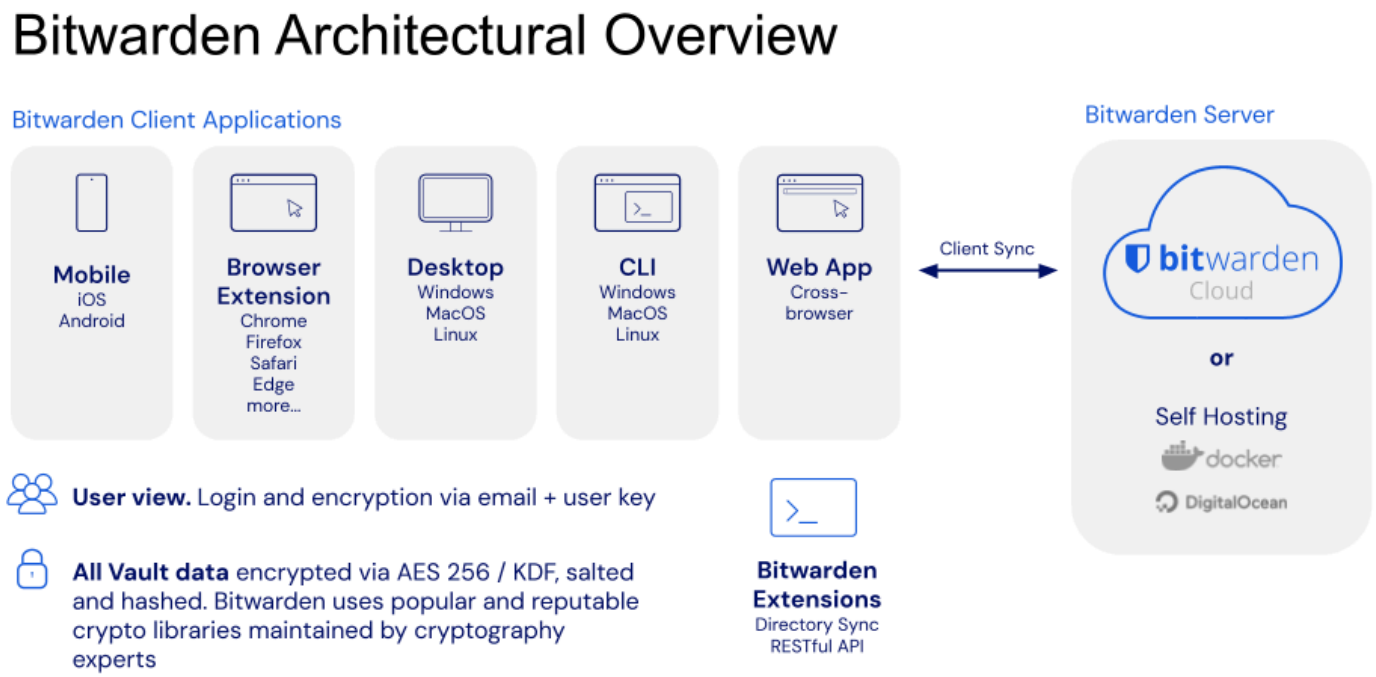
Cloud platform and web application security

Architecture overview

Bitwarden processes and stores all data securely in the Microsoft Azure cloud using services that are managed by the team at Microsoft, including Azure Kubernetes Services (AKS).

Azure Kubernetes Services is a managed Kubernetes service provided by Microsoft that reduces the complexity of deploying and managing Kubernetes clusters. Microsoft fully manages the control plane. The control plane contains all of the components and services that are used to operate and maintain the Bitwarden Kubernetes clusters. Microsoft and the AKS team deploy, operate, and are responsible for the AKS service availability and functionality. The team at Bitwarden manages:

- The access management of the AKS service
- The patching and updating to apply the Node OS security patches, Node image version upgrades, and the Kubernetes version (cluster upgrades)
- The container security for the docker images and running containers in AKS
- The network security of the nodes



Bitwarden architectural overview

Security updates and patching

Azure Kubernetes Services (AKS)

Microsoft provides patches, new node images, and new Kubernetes versions for their AKS service. The team at Bitwarden manages and monitors the AKS environment and follows the upgrade recommendations from Microsoft and vulnerability reports to ensure that Node OS security patches, Node image version upgrades, and the Kubernetes version (cluster upgrades) are applied. In addition, the Bitwarden team applies updates and patches to maintain the container security for the docker imagers and running containers in AKS.

Control of production systems

Bitwarden maintains documented runbooks for all production systems that cover deployment, update, and troubleshooting processes. Extensive alerts are set up to notify and escalate in case of issues.

Baseline configurations

Bitwarden processes and stores all data securely in the Microsoft Azure cloud using services that are managed by the team at Microsoft, including Azure Kubernetes Service (AKS).

Azure Kubernetes Services (AKS)

Security baseline configurations are established and monitored using Cloud Security Posture Management and Vulnerability Management services.

HTTP security headers

Bitwarden leverages HTTP Security headers as an additional level of protection for the Bitwarden web application and communications. For example, HTTP Strict Transport Security (HSTS) will force all connections to use TLS, which mitigates the risks of downgrade attacks and misconfiguration. Content Security Policy headers provide further protection from injection attacks, such as cross-site scripting (XSS). In addition, Bitwarden implements X-Frame-Options: SAMEORIGIN to defend against clickjacking.

Key management procedures

Keys and other secrets utilized by the Bitwarden platform itself, including credentials for Bitwarden cloud provider accounts, are generated, securely stored, and rotated as needed in accordance with industry-standard practices. Bitwarden uses internal Bitwarden vaults for secure storage and backup of sensitive keys or other secrets utilized by the Bitwarden platform. Access to these vaults is carefully managed with access controls, permissions, and roles.

Logging, monitoring, and alert notification

Bitwarden maintains documented runbooks for all production systems that cover deployment, update, and troubleshooting processes. Extensive alerts are set up to notify and escalate in case of issues. A combination of manual and automated monitoring of Bitwarden cloud infrastructure provides a comprehensive and detailed view of system health as well as proactive alerts on areas of concern. Issues are surfaced quickly so that the Bitwarden infrastructure team can effectively respond and mitigate problems with minimal disruption.

Threat prevention and response

Bitwarden performs continuous security monitoring of our networks, assets, data, and services leveraging services and tools including but not limited to Security Information Event Management (SIEM), established Security Operations Center (SOC), Vulnerability Management, Data Loss Prevention (DLP), and Endpoint Detection and Response (EDR). Bitwarden maintains a Security Incident Response policy and plan which is designed to minimize the overall impact of cyber incidents and includes the following as part of the Incident Response Lifecycle:

- Preparation and Planning
- Detection and Analysis
- Containment

- Eradication
- Recovery
- Post-Incident Activities

Bitwarden uses Content Delivery Network (CDN) services in order to provide Web Application Firewalls (WAF) at the edge, better DDoS protection, distributed availability, and caching. Bitwarden also uses proxies within the CDN provider for better network security and performance of its services and sites.

Code assessments

Bitwarden is open source software. All of our source code is hosted on GitHub and is free for anyone to review. Bitwarden source code is audited by reputable third-party security auditing firms as well as independent security researchers. In addition, the Bitwarden Vulnerability Disclosure Program enlists the help of the hacker community at HackerOne to make Bitwarden more secure.

Business continuity and disaster recovery

Bitwarden employs a full range of disaster recovery and business continuity practices from Microsoft Azure that are built into the Bitwarden cloud. This includes high availability and backup services for our application and database tiers.

Software lifecycle and change management

Bitwarden evaluates changes to platform, applications, and production infrastructure to minimize risk and such changes are implemented following the standard operating procedures at Bitwarden.

Change request items are planned based on the roadmap and submitted to engineering. Engineering will review and evaluate their capacity and assess the level of effort for each change request item. After review and evaluation, the product and engineering teams will formulate what they are going to work on for a specific release. The CTO provides details of the release through communication channels and management meetings when the development life cycle begins for that release.

At a high-level, the development, release, testing, and approval process includes:

- Develop, build, and iterate using pull requests in GitHub.
- Get features to a point where they are testable.
- Engineering performs functional testing of the feature and/ or product as they are developing and building.
- Unit testing build and static application security testing (SAST) are automated as part of Bitwarden Continuous Integration (CI) pipelines.
- Some testing is also performed by the Customer Success team.
- Engineering management assists with review and helps to formalize the process, including documentation updates.
- CTO Provides Final Go / No-Go Approval

Meeting Attendance: To ensure successful review, approval implementation and closure of change requests, each core Operation and IT service staff should be represented during the meeting to review and discuss the change request.

Emergency deployment and hotfixes get escalated priority, and review and approval of the change is received from a manager or director prior to the change being made and is subsequently reviewed, communicated and closed during the next scheduled change meeting. This is normally in a service outage, system down or in an urgent outage prevention situation.

Auditability and compliance

The Bitwarden Security and Compliance Program is based on the ISO-27001 Information Security Management System (ISMS). Bitwarden staff have defined policies that govern security and processes, and continually update the security program to be

consistent with applicable legal, industry, and regulatory requirements for services that are provided to you under our Terms of Service Agreement.

Bitwarden complies with industry-standard application security guidelines that include a dedicated security engineering team and include regular reviews of application source code and IT infrastructure to detect, validate, and remediate any security vulnerabilities.

External security reviews

Third-party security reviews and assessments of applications and/or the platform are performed at a minimum of once per year.

Certifications

Bitwarden certifications include:

- SOC2 Type II (renewed annually)
- SOC3 (renewed annually)

According to the AICPA, the use of the Systems and Organization Controls (SOC), SOC 2 Type II report is restricted. For SOC 2 report inquiries, please contact us.

Read More: Bitwarden achieves SOC2 certification

The SOC 3 report provides a summary of the SOC 2 report and is distributed publicly. According to the AICPA, SOC 3 is the SOC for service organizations to report on trust services criteria for general use. Bitwarden makes a copy of the SOC 3 report available here and the summary demonstrates our commitment to security and privacy standards.

These SOC certifications represent one facet of Bitwarden's commitment to safeguarding the security and privacy of customers, and compliance with rigorous standards. Bitwarden also performs a regular cadence of audits on our network security and code integrity.

Read more: Bitwarden 2020 security audit is complete and Bitwarden completes third-party security audit

Employee access controls

Bitwarden employees have significant training and expertise for the type of data, systems, and information assets that they design, architect, implement, manage, support, and interact with.

Bitwarden follows an established on-boarding process to ensure that the appropriate level of access is assigned and maintained. Bitwarden has established levels of access that are appropriate for each role. All requests, including any access change requests, need to be reviewed and approved by the manager. Bitwarden follows a least-privilege policy that grants employees the minimum level of access required to complete their duties. Bitwarden follows an established off-boarding process through Bitwarden Human Resources that revokes all access rights upon an employee's termination.

Threat model and attack surface analysis overview

Bitwarden follows a risk-based approach to designing secure services and systems which include threat modeling and attack surface analysis to identify threats and develop mitigation for them. The risk and threat modeling analysis extends to all areas of the Bitwarden platform including the core Bitwarden cloud server application and the Bitwarden clients such as mobile, desktop, web application, browser and/or command line interfaces.

Bitwarden clients

Users primarily interact with Bitwarden through client applications such as mobile, desktop, web application, browser and/or command line interfaces. The security of these devices, workstations, and web browsers is critical because if one or more of these devices are compromised an attacker may be able to install malware such as a keylogger which would capture all

information entered on these devices including any of your passwords and secrets. You, as the end-user and/or device owner, are responsible for ensuring that your devices are secured and protected from non-authorized access.

HTTPS TLS and web browser crypto end-to-end encryption

The Bitwarden web client runs in your web browser. The authenticity and integrity of the Bitwarden web client depend on the integrity of the HTTPS TLS connection by which it is delivered. An attacker capable of tampering with the traffic that delivers the web client could deliver a malicious client to the user.

Web browser attacks are one of the most popular ways for attackers and cybercriminals to inject malware or inflict damage. Attack vectors on the web browser might include:

- An element of **social engineering, such as phishing,** to trick and persuade the victim to take any action that compromises the security of their user secrets and account.
- **Web browser attacks and browser extension / add-on exploits:** A malicious extension designed to be able to capture user secrets as they are typed on the keyboard.
- **Attacks on web applications through the browser:** Clickjacking, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF).

Bitwarden leverages HTTP Security headers as an additional level of protection for the Bitwarden web application and communications.

Conclusion

This overview of the Bitwarden Security and Compliance program is offered for your review. Bitwarden's solution, software, infrastructure, and security processes have been designed from the ground up with a multi-layered, defense-in-depth approach.

The Bitwarden Security and Compliance Program is based on the ISO-27001 Information Security Management System (ISMS). Bitwarden staff have defined policies that govern security and processes, and continually update the security program to be consistent with applicable legal, industry, and regulatory requirements for services that are provided to you under our Terms of Service Agreement.

If you have any questions, please contact us.

Document changelog

Date published	Summary of changes
December 11, 2024	Adjusted language around memory management.
August 2, 2024	Restructured the document for easier navigation, improved information architecture, and more consistent style
July 25, 2024	Added information related to Cipher Keys for vault item encryption
March 23, 2024	Added new descriptions and diagrams to **Sharing data between users** section
Jan 12, 2024	Added information related to Log in with Passkeys