2019

PREVEIL SECURITY AND DESIGN

A DESCRIPTION OF THE PREVEIL SYSTEM ARCHITECTURE

FEBRUARY 2019
PREVEIL
V. 1.0

Table of Contents

Introduction

PreVeil is an end-to-end encrypted email and file sharing service for businesses and individuals. PreVeil was designed and built from the ground up with the understanding that current information assurance architectures and paradigms are no longer sufficient.

PreVeil's philosophy centers around the three core principles:

- End-to-end encryption - the server never has access to private keys
- No central point of attack - no single person or machine that, if compromised, will leak data for a large number of users
- Ease of use – high quality user experience, integration with existing applications and systems, no passwords

PreVeil protects user information with end-to-end encryption which ensures that no third party – not even PreVeil – can read any user-supplied information stored within the PreVeil system. Unlike most other service providers who say they use end-to-end encryption, PreVeil does not store the decryption keys in a centralized location such as a key server.

PreVeil was built with the assumption that centralized locations are attractive targets that can eventually be compromised by sufficiently motivated attackers. Whereas a successful attack on another provider's key server can compromise their entire encryption system, PreVeil's decentralized key management system is designed from the ground up to thwart this type of attack.

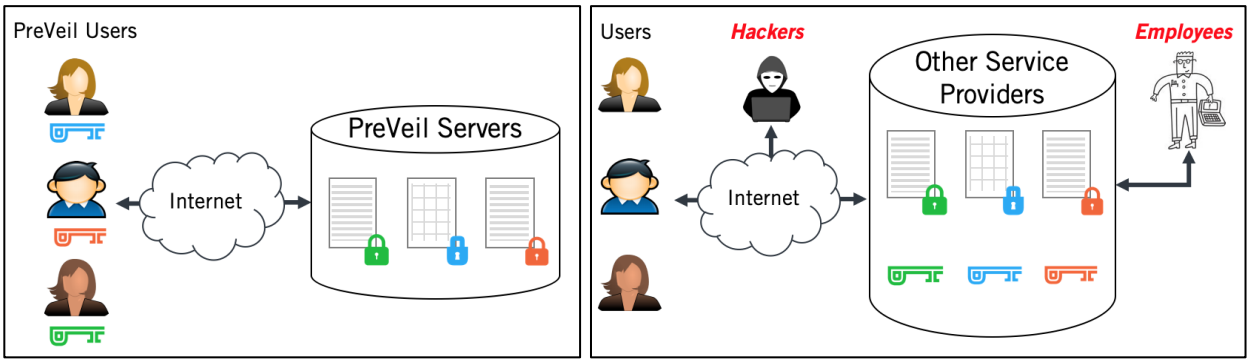


Figure 1 – In the PreVeil system (shown above left) the servers only store encrypted content and the decryption keys are only on user devices. Other service providers (shown above right) may encrypt documents with user-specific keys, but these keys are stored on and accessible by the service provider - meaning that employees or hackers who obtain access to the server can potentially access all user information.

Administrators are often targeted by attackers because they have elevated privileges. To mitigate risks from compromised administrators or other insiders, PreVeil introduces Approval Groups™ - a powerful cryptographic method for distributing trust among a pre-determined group of users. With Approval Groups compromising one administrator does not result in an entire organization being compromised.

PreVeil was designed based on the idea that security is effective only if it's easy to use. Since end-users have ingrained expectations and habits for both email and file-sharing, PreVeil has taken great care to replicate existing user experiences while minimizing friction of adoption. Most of the security features described in this paper occur automatically and without requiring the user to take any special actions.

High Level Design and Enterprise Compatibility

PreVeil is a closed network built on a client-server model in which users download PreVeil software in order to securely collaborate with other PreVeil users. PreVeil has been built upon an end-to-end encryption platform which will enable the service to extend beyond email and file sharing into other collaboration methods.

Downloading PreVeil Software
When users join PreVeil, they first install PreVeil software onto their computer or mobile device and then create an account, which creates a public/private key pair that is associated with their identity and email address. Since PreVeil's current design requires all cryptographic operations to be performed on user devices, PreVeil software must first be installed on each device with which a user is intending to use PreVeil (although PreVeil is currently prototyping a secure solution which does not first require software to be installed on the device).

Client-side Encryption and Decryption & Key Wrapping
Encryption and decryption operations are executed on users' devices and never on the server. The encryption keys that are generated on user devices are themselves encrypted using a key wrapping architecture that allows for great scalability and granular access control.

Dynamic Keys
Keys within the PreVeil system are version controlled and are designed to be dynamic. This design, along with a "rekey" functionality, allows users and IT administrators to reset keys within the PreVeil system should a key be compromised. Rekeying allows the proper distribution of new keys so that an attacker with an old key will no longer be able to access new information.

Authenticating Users and Devices
PreVeil clients communicate directly with the PreVeil server and authenticate with a series of keys which are only stored on user devices. Users must copy their private user key onto each device from which they wish to access their PreVeil account. Their key is securely copied over an authenticated, Diffie-Hellman encrypted tunnel which prevents anyone from snooping on the transfer and extracting a key.

Distributed Trust with Approval Groups
Approval Groups are a novel application of Shamir Secret Sharing that PreVeil implements to distribute trust among a group of people. Approval groups allow certain actions in the PreVeil

system to require X of Y signatures from a specific group before an action is authorized to proceed. This powerful construct means that no single compromised administrator can cause any damage to the PreVeil system.

Designed for Enterprise

Unlike some other end-to-end encrypted email service providers, PreVeil has been built to support enterprise capabilities without weakening its security. Among many, several of the enterprise capabilities are:

- Distributed Account Recovery. Through Approval Groups, PreVeil is able to securely back up private keys without centralizing them or duplicating them. This means that user accounts can be recovered without introducing a vulnerability in the security system.

- Data Export. While PreVeil's distributed keys and architecture make impossible to obtain an organization's data through one attack, organizations sometimes need to access all the emails and documents created by one, some or all users. Through Approval Group technology, PreVeil enables the decryption and export of data only after a number of authorized users have granted their approval for the action.

- User Administration through Distributed Trust. PreVeil has an admin console that pre-determined admins can use to manage PreVeil users at their organization, including adding users, promoting users, and deleting users. PreVeil's Approval Group technology is used to ensure that privileged actions only take place after a number of trusted individuals approve the action.

- Logs and Audit Trail. All user actions within the PreVeil system are logged and encrypted.

Threat Model and Security Guarantees

Any security system should be accompanied by an attacker model as well as a discussion of what security guarantees the system provides and what guarantees it does not provide.

PreVeil provides **end-to-end encryption** to protect the content of documents and email, a well-respected security notion. Additionally, PreVeil implements various traditional security mechanisms for safeguarding our clients and servers from attackers.

Server attacker model

For PreVeil's cryptographic protocol, PreVeil targets a *passive* server-side attacker. This is an attacker that can eavesdrop on the data stored at the server (including the information in memory or on disk) but does not actively modify the execution of any part of the protocol or the data. In particular, we assume that the attacker does not control the key distribution protocol, for example, during which clients look up the public keys of other clients. For an attacker to become active and perform such attacks, the attacker would need to radically compromise the PreVeil servers and gain control of them. To prevent such server-side attacks, PreVeil uses state-of-the-art systems security mechanisms for safeguarding the server (e.g., firewalls, access control, etc.).

Example: Consider the following example: Alice wants to share her private file in Dropbox with Bob, but not with Eve. She sets as an access control policy on Dropbox that only Bob can access her file. She relies on the security of the Dropbox software stack. However, Eve works for Dropbox and manages to steal a copy of Alice's document. Eve can now read Alice's document offline. With PreVeil, Eve only obtains an encrypted copy of the document content. Eve cannot decrypt this content even if Eve stole all information at the server: the decryption key is never sent to the server in PreVeil.

PreVeil encrypts as much information about documents or emails as possible and exposes little information to the PreVeil server. In particular, PreVeil encrypts the contents and title of the documents or emails as well as email attachments. PreVeil also encrypts some metadata about the documents such as the title of a document/email and the initial snippet of an email. Like other end-to-end encrypted systems, some metadata is not encrypted in PreVeil because it is needed as part of the basic operation or performance of the system. This includes, but is not limited to, which users access or have access to what collections or documents (e.g., who sends an email to whom), who does what operation at what time, and the count and size of encrypted data in operations or documents. For example, the PreVeil server needs to know that Alice is sending an email to Bob, so that it puts the encrypted email in Bob's inbox and in Alice's sent box, but the server does not see the content, title, attachments, or snippet of the email.

As part of allowing users to find other PreVeil members, PreVeil enables a user to tell if an email address is a member of PreVeil. This also means that an attacker can take advantage of this

feature via a dictionary attack (that is, enumerate many possible emails) to identify who are all the members of PreVeil. To minimize this type of attack, administrator of PreVeil services can restrict who sees PreVeil membership via a Trusted Community feature (e.g., only users within a company).

User attacker model

A user could be malicious, or the user's machine could get compromised (e.g., the user carelessly installs malware, an attacker steals a user's laptop). Even if this is the case, PreVeil prevents this user from accessing the document content of another user, as long as the second user did not previously share (directly or indirectly) the document with the malicious user. The approval group is meant to help a victimized user recover their account and change their key.

A user must choose a trustworthy approval group in order to protect his/her data. PreVeil also assumes that a user's email account was not compromised at the time of setting up an account with PreVeil.

Email or document search in PreVeil happens locally on the client-side, so there is no side channel leakage from a server-side search.

Description of Key Infrastructure

Overview

In the PreVeil system, information is encrypted and decrypted only on a user's client device, and the information stored on the cloud server is always encrypted. Each document or message – including its metadata such as file name and email subject - is encrypted with its own unique key.

Types of Keys in the PreVeil System

The PreVeil system generally uses asymmetric keys to manage identity and access and perform digital signatures and uses symmetric keys to encrypt information.

- Symmetric keys are used to encrypt and decrypt information. Symmetric keys are much faster at performing encryption / decryption operations than asymmetric, but by themselves aren't very scalable because as a system grows, it becomes increasingly difficult to distribute and update keys as more information and users are added.

- Asymmetric keys are used to manage identity and access to information. Asymmetric keys have a public component, which is shared to everyone in the PreVeil system, and a private component, which remains solely on user devices. Asymmetric cryptography was designed to be scalable, since each new user only needs to share a single public key in order to share their identity and have information be encrypted for them.

The following table lists the types of keys in the PreVeil system along with their function:

Key Name	Type	Purpose
User Key	Asymmetric	Each user has their own unique User Key. This is the cryptographic identity for each user in the PreVeil system. The User key is used to encrypt information for a user and to verify the authenticity of information sent from the user.
Device Key	Asymmetric	Each device on the PreVeil system has its own unique device key, which in conjunction with the User Key, is used to authenticate communication to the PreVeil servers.
Group Key	Asymmetric	Groups in the PreVeil system are used to facilitate distribution and provide another level of access control and differentiation. Membership is based on ownership of the private keys. The primary use for groups is currently to differentiate administrators from normal users in PreVeil organizations.

Data Block Key	Symmetric	In the PreVeil system, information is stored in blocks of identical size, with each block being encrypted with a different key. Each of these keys is then encrypted under another key which is shared with the appropriate PreVeil members. This key encryption happens client-side, so PreVeil servers never have access to decrypted keys.
Folder Permission Keys	Asymmetric	Shared folders have six different keys which are used to control access to the shared folders: • Read – read files stored in the folder • Write – write files or folders in the shared folder • View ACL – view the shared folder access list • Log access – access to logs for the shared folder • Share – share the folder to other PreVeil users • Owner – reserved for future use
Directory key	Symmetric	Each folder directory in the PreVeil system stores metadata, including the names of files/folders in the folder. This information is encrypted using the Read Folder Permission Key.
Organization Keys	Asymmetric	Organizations are special constructs in the PreVeil system which are used to group PreVeil users together. Organizations also have admin groups and have three special keys: • Log key – shared with Admin group to see user logs • Owner key – reserved for future use • Share key – reserved for future use

Identity & Authentication in the PreVeil System

Identity & Digital Signatures

Identity in PreVeil is defined by the association of three factors which are stored together in the PreVeil server user database.

- a public key which is created by the user when they install PreVeil and create an account
- an email address (ownership of which is verified) that is used as a PreVeil ID
- a first and last name, which are displayed in PreVeil along with the user's email address

All PreVeil users have a public key. As described in Key Management and Key Wrapping, the user's public key is used to encrypt other keys and documents uniquely for those users. As is further described in "Separating Signatures from Encryption" each request made to the server is signed by the user's private signing key to cryptographically prove that a user performed an action or guarantee the authenticity of a document or encrypted key.

User Account Setup & Verification

To join PreVeil, a new user first downloads PreVeil software onto their computer or mobile device and have access to the email address they plan on using as a PreVeil ID. The following describes the process to set up a new PreVeil account:

- PreVeil software is downloaded onto the user's computer or mobile device. In enterprise settings, PreVeil can be remotely installed using services such as Active Directory.
- The user enters their name and email address and submits this to the PreVeil server. In enterprise settings, this provisioning is performed by Administrators and end users simply need to click a button.
- The server generates a secret code and sends it to the email address as a clickable link
- The user has a set amount of time to click the link, upon which
    - The PreVeil software on the user's device creates a new user key
    - The PreVeil software on the user's device creates a new device key
    - Both public keys are sent to the PreVeil server and are associated with the user
- The server adds the user to its identity database and configures storage space for the user's email and files.
- At this point, existing PreVeil users can use the new user's public key to send them information.

Note that the flow is slightly different for enterprise users – see here: Adding Users and Creating User Accounts.

Multiple Devices Without a Login/Password

Since PreVeil does not use a login or password system, using PreVeil on multiple devices is slightly different because users must copy their private key onto each device from which they would like to access PreVeil. Their private key is sent through a user-authenticated, Diffie-Hellman protected encrypted tunnel which prevents any third party from intercepting the key.

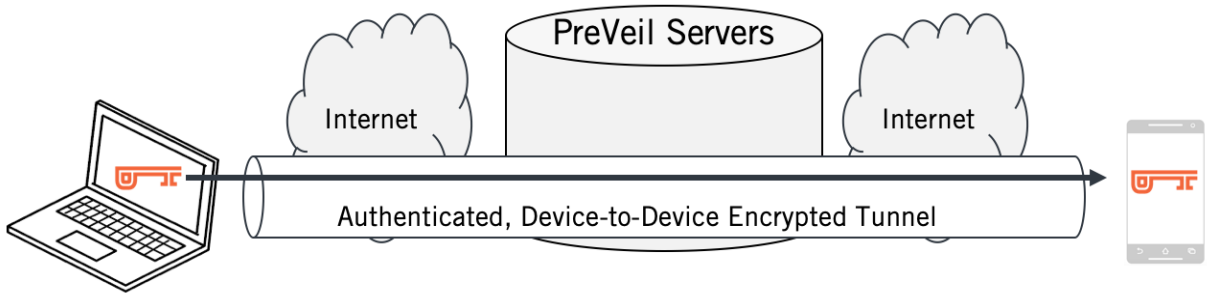


Figure 2 – Users must add their account to each device from which they want to use PreVeil. The process is simple, quick, and is encrypted so that no third party – not even PreVeil – can see the key.

The key copying flow is as follows:

- The user verifies that they have access to a device with their existing PreVeil account
- The user downloads PreVeil onto their new device and selects “add existing user”
- On the new device, the user sees a series of 8 characters
- The user enters these on the prompt on their existing device. These characters act as an authentication mechanism to ensure that one device is securely connected to another through a Diffie-Hellman exchange.
- The private key is copied to the new device, upon which a new device key is generated and associated with the user account (see below for more on device keys).

This architecture provides robust protection against remote attackers because logins cannot realistically be attempted. Since encryption keys are orders of magnitude harder to guess than passwords and are not re-used by users between services, PreVeil users are protected against attacks that may leverage passwords stolen from other services.

Device Keys and Management

When a PreVeil account is first created or copied onto a device, an asymmetric key is generated to uniquely identify the device. The public key is uploaded to the PreVeil server, which keeps track of which device keys are associated with each user. Device keys are rotated automatically on a 24-hour basis, a feature which provides additional protection in the case of a compromised device and that will be used in the future to further facilitate account setup and use on multiple devices.

From any active device, a user can remotely disable another device that is registered to their account and thereby block that device from accessing PreVeil information or performing actions in the system. Upon receiving such a request, the server checks to verify that the device making the request is still valid, after which the server then finds the corresponding device’s public key and temporarily revokes it. Subsequent requests or actions from that device will be ignored from the server. A user can then unlock the device from any active device by sending the server an unlock command.

Should an attacker obtain a user’s device and remotely lock out a user’s devices, a user that has set up an Approval Group will be able to recover their account by performing an account recovery onto a new or existing device (see process description: Recovering a user key.) After an account has been recovered, a user’s account will automatically be rekeyed, therefore locking the attacker out and preventing the attacker from accessing new material (although they will still be able to access any material they have already downloaded).

Authenticated Server Connections

PreVeil clients communicate to the server over encrypted transport layers, including HTTPS and encrypted Websockets. Each request made to the server is signed using the user key and the device key. The server only responds if both keys are active (not revoked) and correspond to the latest key versions listed on the server. Therefore, an attacker will not be able to extract information or perform actions in PreVeil by only using old keys or one of the two key types.

Key Management and Key Wrapping

Overview

For file storage and sharing, PreVeil uses a "wrapped key" architecture in which the keys required to decrypt information themselves are encrypted using encryption keys that only the appropriate users can access. For emails, the recipient's public user key is directly used to encrypt the keys protecting email subject, content, and attachment. For files, there is an additional layer of keys used to differentiate file permissions.

To ensure that PreVeil servers are never able to access decryption keys:

- All user information is only ever encrypted and decrypted on user devices
- The cryptographic action in which Folder Permission Keys are "wrapped" for the appropriate other users takes place on user devices, not the server
- The PreVeil server only stores symmetric or private keys that are encrypted - the PreVeil server never has access to any unencrypted keys which can then directly be used to access user information

Key wrapping allows for scalable distribution of encryption keys. Below is a simplification of the data stored on the PreVeil server which we'll use to describe how key wrapping works. While PreVeil can implement multiple tiered access controls, at a high level, there are three categories of encryption: document-level encryption, granular folder permissions, and user access controls – discussed in depth below.

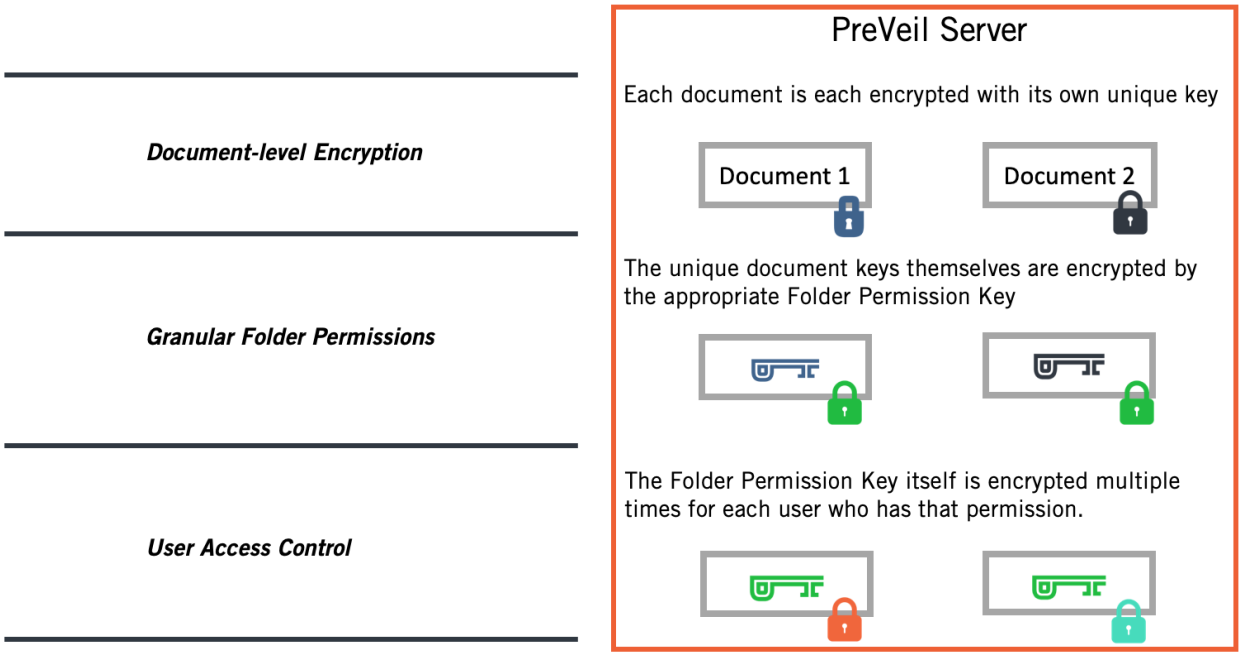


Figure 3 – Key wrapping allows for granular file protection and access control without generating a high number of encryption keys. Access is controlled by distributing access to a key which allows another key to be accessed which then allows document access. Access to keys is controlled on a per-user level in which user public keys are used to encrypt copies of the same access key per user with the appropriate permission. Note that the PreVeil server only contains encrypted information – even though keys are stored on the server, they are stored encrypted so that an attacker or employee is unable to access the key.

Document-level protection

Each document is encrypted with a unique key, so that an attacker who compromises one document cannot then obtain access to other documents. These symmetric keys are generated on the user devices that first upload the documents. These document keys themselves are then encrypted and stored on the server.

Granular Folder Permissions

All files under the same folder share the same access control and permissions list, so all documents being shared to the same folder use the same permission key to encrypt the document encryption keys. In the picture above, the green lock represents a permission key that allows an arbitrary number of unique document encryption keys (such as the blue and black keys) to be encrypted without requiring subsequent modifications to the key infrastructure.

Each folder has 6 asymmetric permission keys which enable users to perform actions if they have access to the corresponding private key. For example, the keys that encrypt documents in a certain folder are encrypted under that folder's Read Permission Key (the blue Document 1 encryption key itself is encrypted with the green Folder Read Permission Key in the above figure). Similarly, other permissions are authorized or verified by the server by looking to see if the request has been signed with the appropriate permission key (e.g. a user can only write something to a shared folder if the request has been signed with the Write Permission Key).

User Access Control
Folder permissions are granted by giving a user access to the private key component of the corresponding Folder Permission key – by encrypting the folder permission private key for each user with the user's public key. This 'wrapping' is performed on the device of the user granting permissions to others and happens automatically when a user shares a folder.

In the above example, if Alice is sharing Document 1 and Document 2, the only action Alice needs to perform is to encrypt the appropriate private Folder Permission key (the green colored key) using the recipient's public key (in this case the teal colored lock). Since each user has their own copy of the private key, the PreVeil system can granularly track access and creates an audit log based on each user.

Customizable and Scalable
The key wrapping architecture described can be used to extend to further levels of wrapped keys. Although not graphically depicted, the notion of Groups in PreVeil is simply another set of "Granular Folder Permissions" whose keys wrap other permission keys. Since keys themselves take up negligible amounts of disk space, wrapping keys for multiple users is feasible even at scale.

Key Wrapping Example – Reading Shared Information

Below is an example to describe the cryptographic components and their interactions. In this scenario, Alice is trying to access Document 1, which has been shared with her. All of this key management and distribution is completely transparent to the user and happens automatically.

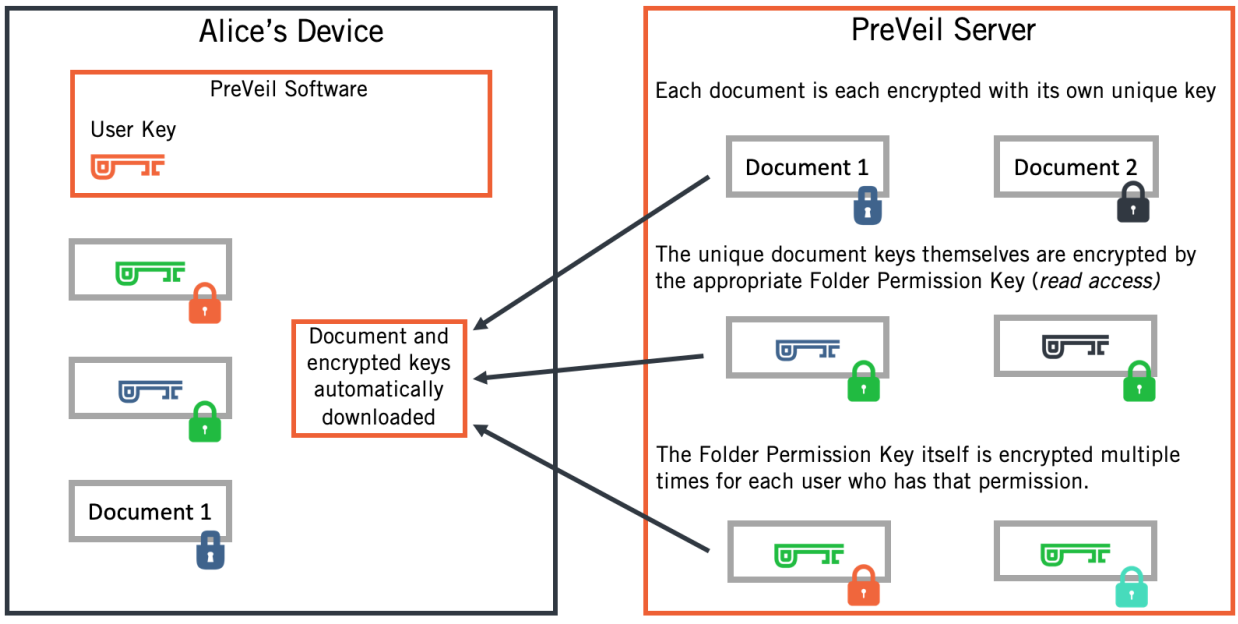


Figure 4 – The PreVeil software on Alice's device sees that Document 1 has been added to the server and so it downloads the encrypted Document 1, the Document 1 decryption key which has been encrypted under the Folder Read Key, and the Folder Read Key which has been encrypted under her own user key.

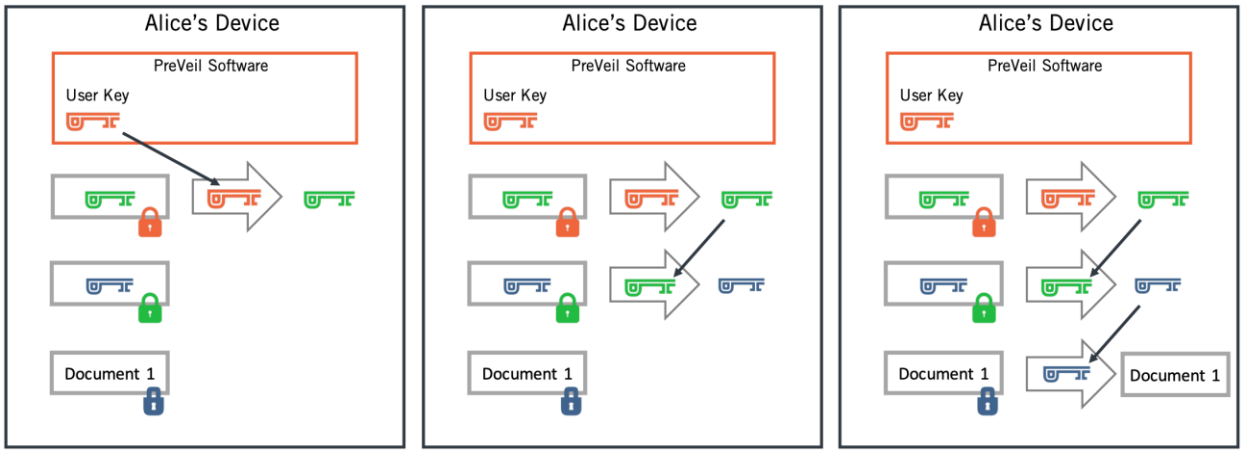


Figure 5 – After downloading all the files, the PreVeil software on Alice's device first uses Alice's orange private key (which is only kept on her devices) to decrypt the green Folder Read Key (left picture). The Folder Read Key is then used to decrypt the blue Document 1 decryption key (middle picture). The Document 1 decryption key is then used to decrypt Document 1 (right picture). This entire process happens automatically in the PreVeil software.

Key Wrapping Example – Writing Shared Information & Sharing with a New User

When a user, Alice, uploads and shares a new document, the process is reversed:

1. The PreVeil software identifies a newly added file and on Alice's device, generates a unique file key which encrypts this file (this is a simplification - as described later, a series of unique block encryption keys, not a single file encryption key, are generated).

2. The document is encrypted with the file key

3. The encrypted document is sent to the PreVeil server

4. The file key is then encrypted using the public key of the appropriate "Read Folder Permission Key"

5. Any user who has access to the Read Folder Permission Key is then able to decrypt the file key and then subsequently decrypt the document

6. To share the document with a new user, Alice would wrap the private Read Folder Permission Key using the recipient's public key

Note that in this flow, neither the document nor the document key are encrypted directly with Alice's recipients' user keys. PreVeil's key wrapping architecture separates the need to map a document directly to a user. As long as a user is granted the correct permission, they will be able to access the correct documents.

Approval Groups

Decentralize Trust and Remove Single Points of Failure

An Approval Group is a set of users who together authorize an operation. Traditional systems allow each admin to have very broad privileges to reset user passwords, access user data, etc. PreVeil tries to avoid such central points of attack. Approval Groups cryptographically distribute trust amongst a set of people, making the system much more secure than systems that centralize trust in a set of admins.

Implementation

Approval Groups can be constructed to allow any x out of y approvers to authorize an operation. They can also be cascaded together to enable more complex combinations, for example: any 3 of 10 users of group A (IT administrators) plus 1 out of 2 users of group B (executives).

Approval Groups for User Recovery

Approval Groups recover secret keys by using the Shamir Secret Sharing algorithm. To set up an Approval Group (an operation that's executed on the client), a secret key is broken into fragments or shards, with each shard assigned to an approver. The shard is encrypted with authenticated encryption using the approver's public key and the user's private signing key (thereby ensuring that the approver can verify that the correct user is the one who actually sent the wrapped shard to the server). These encrypted shards are stored on the PreVeil servers.

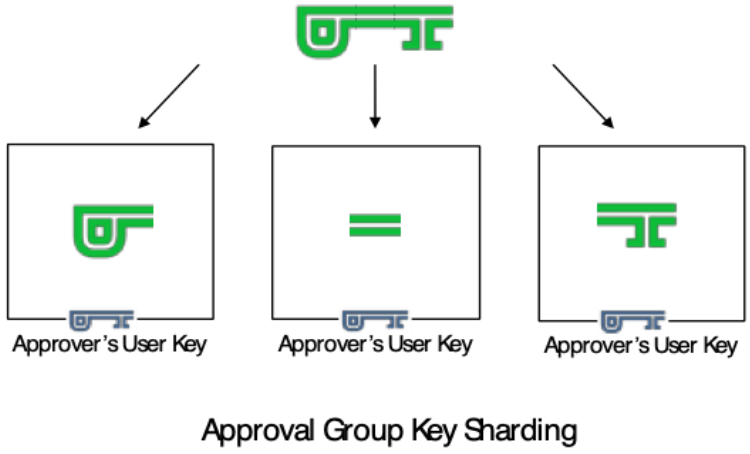


Figure 6 – A conceptual illustration of a user's private key being "sharded" so that each shard is encrypted in a manner that only one approval group member can access. When recovering a key, the shards are securely copied and reconstituted. Note that Shamir Secret Sharing ensures that the 'shard' is not a part of a key, just a piece of information that allows the key to be reconstituted.

When a user needs to recover the secret key, each of the required approvers (x out of y) retrieves their encrypted shard from the server, decrypts it using the approver's private User key, and sends the decrypted shard securely to the user that's recovering the key. Using the Shamir Secret Sharing algorithm, the shards can be mathematically combined to recover the original secret key. Note that the secure transmission of the shards is implemented via the Diffie Hellman Key Exchange protocol. After a key is recovered, it must be re-keyed to prevent a repeat of the key recovery without the approvers' consent.

Approval Group Enterprise Uses

Enterprise use of Approval Groups extends beyond account recovery. A description of these uses can be found here:

Approval Group Functions.

PreVeil Keys in Depth, Key Versioning & Rekeying

PreVeil Key Details and Metadata

All asymmetric keys in the PreVeil system actually consist of three components:

1. **Key Version** - used to track the version of the key so that the PreVeil server only authenticates and accepts requests from the latest version of the key
2. **Signing Key** – used to cryptographically sign requests and actions resulting in cryptographic proof that certain users or groups performed actions
3. **Decryption Key** – used to perform encryption/decryption operations for specific users or groups

Separating Signatures from Encryption

Previously, asymmetric keys had been discussed as a key pair – a public key component and a private key component. While this is correct, the cryptographic libraries and implementation used by PreVeil actually use two key pairs – one public/private key pair for encryption and decryption, and another public/private key pair for signing.

The separation of these functions provides greater mitigation against any attacks in which a component of a key is compromised – if an attacker is able to obtain a decryption key, they still will not be able to masquerade as a user and sign as them, and vice versa.

Separating the encryption function from the signature function also allows the enterprise Data Export & E-Discovery feature to decrypt information without compromising the identities of users – in this process, only the encryption/decryption keys are copied to perform the decryption, but the signing keys are not copied and thus a user's identity cannot be compromised.

Key Versioning & Old Key Access

Keys in the PreVeil system are intended to be dynamic to mitigate the scope and effect of attackers who are able to access old versions of keys. As such, PreVeil's internal protocols allow each asymmetric key to be versioned so that only the latest key version is acknowledged by the server and authorized to perform actions.

To ensure access to material encrypted using older keys, when a key version is increased (see Rekeying below), old keys are encrypted using the new encryption key. This allows a user with access to the latest key to access all other previous keys.

Rekeying Guarantee

Rekeying is an integral part of the PreVeil security architecture because it provides a method to ensure that any malicious user with access to an older key is not able to access any new material.

Rekeying is a multi-step process that originates on a user's device (to guarantee that the server does not access the newly generated key). Some iterations of rekeying require Approval Group™ authorization, such as Recovering a user key.

Rekey Types and Triggers

Below is a list of events which trigger rekeying events across the three different classes of keys explained in the overview section of Identity & Authentication in the PreVeil System

Identity & Digital Signatures

Identity in PreVeil is defined by the association of three factors which are stored together in the PreVeil server user database.

- a public key which is created by the user when they install PreVeil and create an account
- an email address (ownership of which is verified) that is used as a PreVeil ID
- a first and last name, which are displayed in PreVeil along with the user's email address

All PreVeil users have a public key. As described in Key Management and Key Wrapping, the user's public key is used to encrypt other keys and documents uniquely for those users. As is further described in "Separating Signatures from Encryption" each request made to the server is signed by the user's private signing key to cryptographically prove that a user performed an action or guarantee the authenticity of a document or encrypted key.

User Account Setup & Verification

To join PreVeil, a new user first downloads PreVeil software onto their computer or mobile device and have access to the email address they plan on using as a PreVeil ID. The following describes the process to set up a new PreVeil account:

- PreVeil software is downloaded onto the user's computer or mobile device. In enterprise settings, PreVeil can be remotely installed using services such as Active Directory.
- The user enters their name and email address and submits this to the PreVeil server. In enterprise settings, this provisioning is performed by Administrators and end users simply need to click a button.
- The server generates a secret code and sends it to the email address as a clickable link
- The user has a set amount of time to click the link, upon which
    - The PreVeil software on the user's device creates a new user key
    - The PreVeil software on the user's device creates a new device key
    - Both public keys are sent to the PreVeil server and are associated with the user
- The server adds the user to its identity database and configures storage space for the user's email and files.
- At this point, existing PreVeil users can use the new user's public key to send them information.

Note that the flow is slightly different for enterprise users – see here: Adding Users and Creating User Accounts.

Multiple Devices Without a Login/Password

Since PreVeil does not use a login or password system, using PreVeil on multiple devices is slightly different because users must copy their private key onto each device from which they would like to access PreVeil. Their private key is sent through a user-authenticated, Diffie-Hellman protected encrypted tunnel which prevents any third party from intercepting the key.

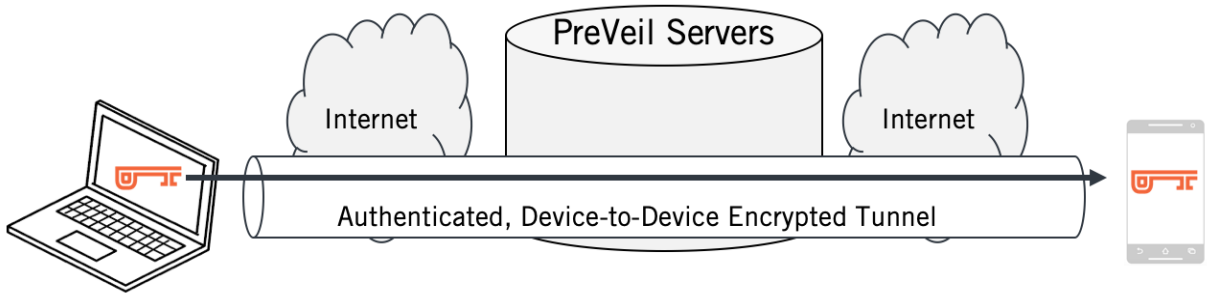


Figure 2 – Users must add their account to each device from which they want to use PreVeil. The process is simple, quick, and is encrypted so that no third party – not even PreVeil – can see the key.

The key copying flow is as follows:

- The user verifies that they have access to a device with their existing PreVeil account
- The user downloads PreVeil onto their new device and selects “add existing user”
- On the new device, the user sees a series of 8 characters
- The user enters these on the prompt on their existing device. These characters act as an authentication mechanism to ensure that one device is securely connected to another through a Diffie-Hellman exchange.
- The private key is copied to the new device, upon which a new device key is generated and associated with the user account (see below for more on device keys).

This architecture provides robust protection against remote attackers because logins cannot realistically be attempted. Since encryption keys are orders of magnitude harder to guess than passwords and are not re-used by users between services, PreVeil users are protected against attacks that may leverage passwords stolen from other services.

Device Keys and Management

When a PreVeil account is first created or copied onto a device, an asymmetric key is generated to uniquely identify the device. The public key is uploaded to the PreVeil server, which keeps track of which device keys are associated with each user. Device keys are rotated automatically on a 24-hour basis, a feature which provides additional protection in the case of a compromised device and that will be used in the future to further facilitate account setup and use on multiple devices.

From any active device, a user can remotely disable another device that is registered to their account and thereby block that device from accessing PreVeil information or performing actions in the system. Upon receiving such a request, the server checks to verify that the device making the request is still valid, after which the server then finds the corresponding device’s public key and temporarily revokes it. Subsequent requests or actions from that device will be ignored from the server. A user can then unlock the device from any active device by sending the server an unlock command.

Should an attacker obtain a user’s device and remotely lock out a user’s devices, a user that has set up an Approval Group will be able to recover their account by performing an account recovery onto a new or existing device (see process description: Recovering a user key.) After an account has been recovered, a user’s account will automatically be rekeyed, therefore locking the attacker out and preventing the attacker from accessing new material (although they will still be able to access any material they have already downloaded).

Authenticated Server Connections

PreVeil clients communicate to the server over encrypted transport layers, including HTTPS and encrypted Websockets. Each request made to the server is signed using the user key and the device key. The server only responds if both keys are active (not revoked) and correspond to the latest key versions listed on the server. Therefore, an attacker will not be able to extract information or perform actions in PreVeil by only using old keys or one of the two key types.

Key Management and Key Wrapping.

User / User Access Control Keys

- When an Approval Group is changed, a user key is rekeyed to prevent the old Approval Group from later colluding to get the user's key.
- After recovering an account using an Approval Group, a user key is rekeyed to prevent the Approval Group from later colluding to get the user's key (in production 2Q 2019).

Folder Permission Keys

- When someone unshares or changes user permissions on a shared folder, the user performing the action creates a new version of all Folder Permission Keys and re-encrypts these for accordingly for the remaining users. This ensures that those who have been unshared are not able to access the Folder Permission Keys and so cannot access document any new encryption keys.

Encrypted information

- Updating keys does not require data to be re-encrypted. Doing so would require a user's entire email and document base to be downloaded onto one device and re-encrypted using new unique information keys (for each email and document). By rewrapping the keys themselves under new Folder Permission Keys, as described above, PreVeil still minimizes the amount of information attackers can obtain with old access keys.

Rekeying Details

The following are changes that take place when a user rekeys. For the sake of clarity, the example below describes what happens when a user "Alice" rekeys.

User Device

- A new user key is created and the public key is uploaded to the PreVeil server.
- The old user key is encrypted with the new key - so Alice uses her new private key to access her old private key to decrypt old messages. The encrypted old user key is stored

on the server to ensure that it as well as access to old data can be accessed from any of Alice's devices.
- Alice's home Drive folder and existing folder permission keys are re-encrypted under her new user key (in production 2Q 2019).
- Alice's new user private key is re-sharded and distributed to her Approval Group.

Server Directory
- The public component of Alice's user key replaces the public component of her old key.

Emails
- Moving forward, new emails sent to Alice will be encrypted using Alice's new user key.
- Alice's old emails will remain encrypted using Alice's old key.

Alice's Approval Group
- Each member of Alice's Approval Group receives a shard of Alice's new private key, with each shard uniquely encrypted for a single approval group member.

Cryptographic Algorithms & FIPS 140-2

PreVeil was built upon the open-source cryptographic libraries NaCl and LibSodium, found at https://nacl.cr.yp.to/ and https://download.libsodium.org/doc/ respectively. The legacy[1] encryption settings that PreVeil uses as a first layer of encryption are as follows:

- Symmetric keys are generated using XSalsa20
- Asymmetric keys are generated using elliptic curve algorithm Curve25519

Moving forward, PreVeil supports use cases that require NIST FIPS 140-2 certified encryption algorithms. To support this standard, a new symmetric encryption algorithm is used, and the existing asymmetric algorithm supplemented with a FIPS approved algorithm – meaning that the original protection provided by Curve25519 remains. All PreVeil users will use the same encryption algorithms. The encryption parameters used for this are listed below:

- Symmetric keys are generated using AES-256 in GCM mode
- Asymmetric keys are generated using elliptic curve P-256

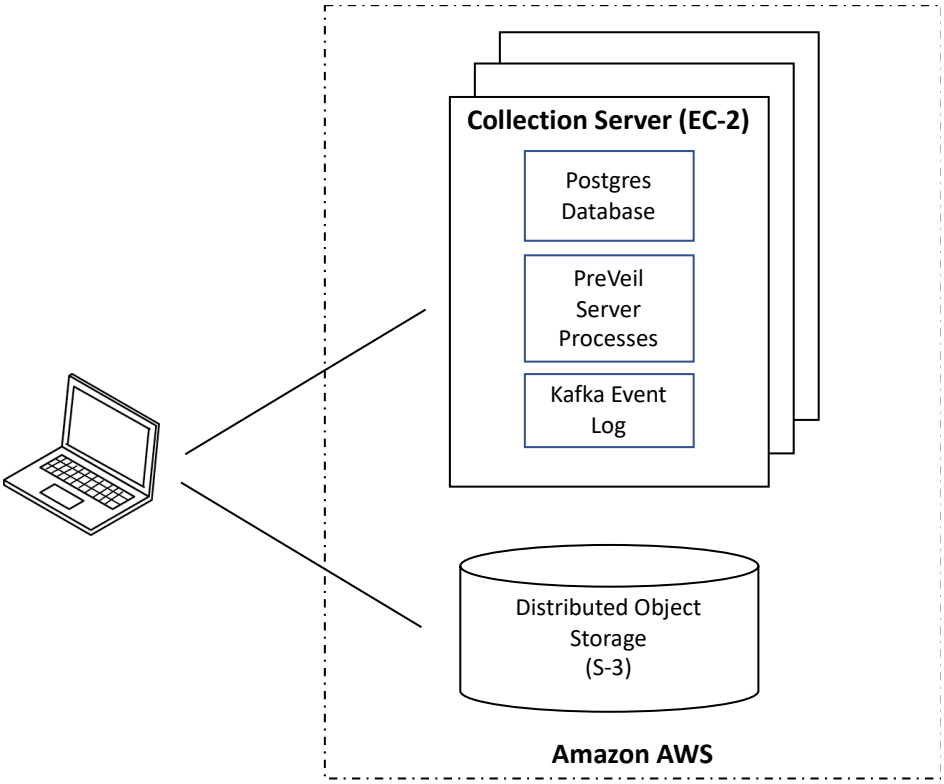
The asymmetric keys are integrated in a manner such that both Curve25519 and P-256 keys are required for a user to decrypt information. This architecture allows PreVeil to be used by those requiring FIPS 140-2 algorithms as well as those who are looking for the protection offered by the community-approved NaCl/Libsodium project. Both sets of user key pairs – those generated

[1] PreVeil will be using all FIPS-140-2 compliant algorithms starting in April 2019. Documents and email in PreVeil created before this time will be protected using legacy encryption algorithms.

by NaCl/LibSodium as well as those generated by FIPS 140-2 algorithms are in fact pairs of key pairs – one used for signing and another used for encryption.

Data Infrastructure — Email and Files

Architecture for Storage and Processing



Server Architecture

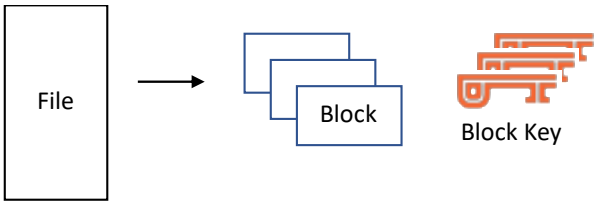
Storage. Both file and email data are stored on the server in a key-value storage system using Amazon's S-3 service. Alternatively, data can be stored on-premise on a server owned by a customer or a 3rd party storage provider that provides an S-3 style key-value interface.

Processing of client requests, storage of metadata and keys are handled by a set of Collection Servers deployed on Amazon's EC-2 service. Collections are described in more detail below, but for now consider a Collection to be the home of a user's email mailbox or a directory of files. Each Collection resides on a single Collection Server, but a Collection Server can support many Collections. Each Collection Server is also associated with a Postgres database that contains the keys associated with the Collections residing on that server. Private keys are always encrypted on the client before being stored on a server; the server can never access private keys directly. Each Collection Server also contains public User keys for all users in the PreVeil system.

Client requests and events are posted to a Kafka event log that's used to reconstruct state should a Collection Server fail. Collections are replicated on two servers, so if a server suddenly crashes, other servers can take over with minimal Kafka replay. When a server fails, its

responsibilities are distributed across the remaining servers. Additional servers may be added at any time, and the total load will be redistributed across the available server infrastructure.

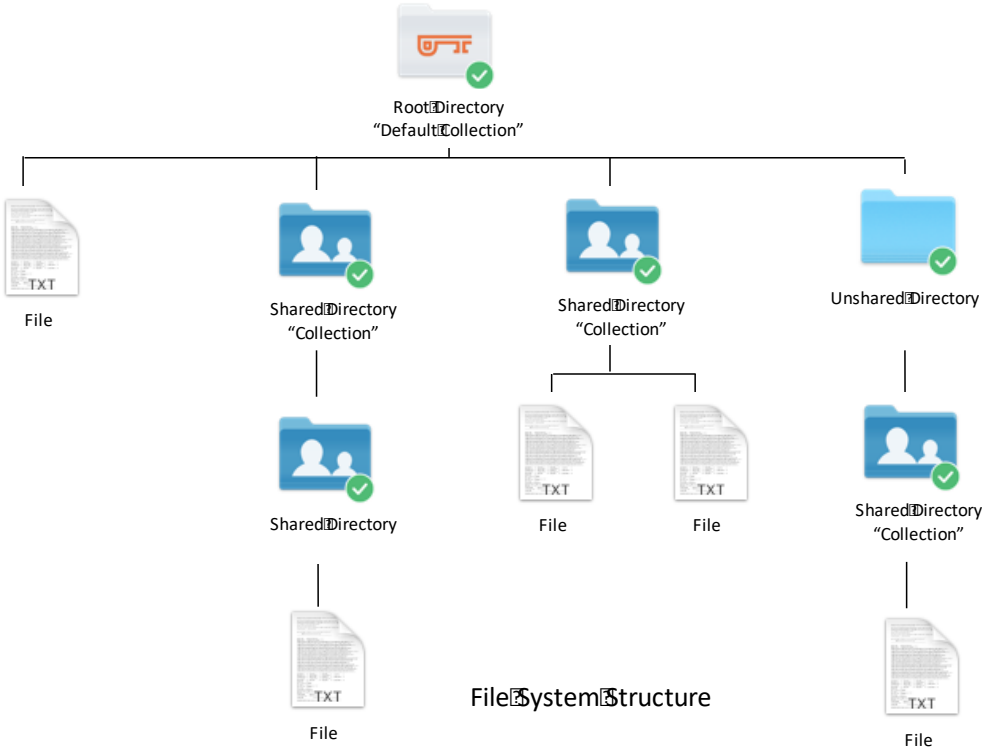
PreVeil Data Structure and Nomenclature - Files, Directories, and Collections



Files partitioned into blocks then encrypted with unique Keys

Files are stored as a set of blocks, each encrypted under a unique Block Key, which is a symmetric key. Each block has a unique Block ID that identifies it for storage and retrieval in S-3 storage. Blocks are updated in "append" mode on S-3, which means that any block changes are added to storage. This allows retrieval of older versions of files. Each Collection has a data retention policy that specifies the length of time or number of versions to be retained. Blocks that are no longer required are flagged for deletion via a garbage collection process that occurs periodically.

Files are organized into directories, and the file name metadata in a directory is encrypted under a symmetric Directory Key. Directories can contain sub-directories.



File System Structure

Sharing Rules

A Collection is a special kind of directory in that it can be shared with other users. A Collection can contain files as well as sub-directories. All of the contents of a Collection have the same sharing permissions. In other words, if the sharing permissions for a directory include "Edit & Share" for user A & B and "read-only" for user C, then all directories and files in the Collection will be editable and sharable for users A&B and read-only for user C.

Collections may not contain other collections as a user interface design rule; users often get confused when sharing permissions differ across files or sub-directories. There is one exception to this rule: each user has a "root" level directory in which all other directories and Collections are kept. This root directory is called the Default Collection.

Logs & Audit Trail

Collections have **logs** associated with them to track activity (i.e. user x changed file y). Log entries are encrypted on the server with an asymmetric Log Key and stored with the Collection. The server creates a log entry for each operation and encrypts the log using the public Log Key. The private Log Key is encrypted under the User Key so that users can see their own log entries. In organizations (described below), Log Keys are also encrypted under an Admin Key so authorized admins can also view logs. More on administrative access below. Log entries are hash-chained and signed to ensure that an attacker cannot modify logs.

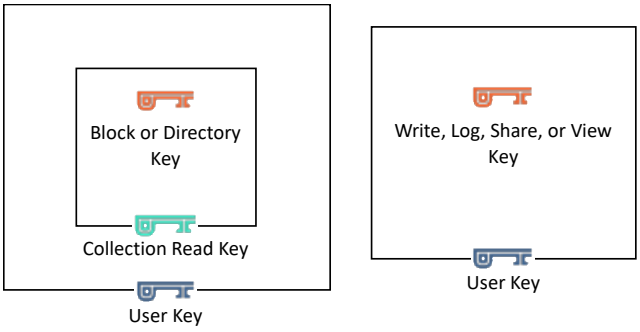
Advanced Permissions

Various **permissions** can be granted to users who have access to a Collection, including the following:

- Read access — the ability to read files and directories
- Write access — the ability to create and update files and directories
- View access — view the Access Control List, i.e. see the list of users who have access to the Collection and what their privileges are
- Share — the ability to share a collection with others.
- Log — the ability to view logs

These permissions are combined together to create sharing options:

- Read only = Read
- Edit & Share = Read, Write, View, Share, Log
- Edit = Read, Write, View, Log
- Blind Drop = Write



Key Wrapping

A Collection Read Key is an asymmetric key that's used to provide security for read operations on files and directories. Block keys and Directory Keys are encrypted under Collection Read Keys. Also, the metadata for file and directory names in the root directory (the Default Collection) are encrypted under the Collection Read Key. Each user who has read access to a Collection will have the corresponding Collection Read Key encrypted under their User Key stored on the server.

A Collection Write Key is an asymmetric key that's used to provide security for write operations. Whenever a client creates, uploads, updates, or deletes file blocks to the server, the request is signed by the client with the Collection Write Key. The signature is checked by the server before the data is stored. Users who have write access to a collection have the corresponding Collection Write Key encrypted under their User keys stored on the server.

The server also enforces signed requests for the following operations:

- Sharing collections with other users — requires a request to be signed by a Collection Sharing Key (the server also ensures that the sharer possesses the privileges being shared).
- Viewing other members of a shared collection — requires a request to be signed by an ACL Viewer Key (access control list).
- Re-keying a collection — requires a request to be signed by a Collection Owner Key.

File Operations

Create a file

When a client creates a file to be stored on the server, it divides the file into blocks, creates keys for each block, and encrypts the blocks using the block keys. All files are contained in a Collection, which could be the root (i.e. "default Collection" that's private to the user) or another collection that's shared with others. The block keys are encrypted under the relevant Collection Read Key. The client creates a request for the server to create the file and attaches to the request the encrypted block keys, filename and other metadata encrypted under the appropriate Directory Key, and a list of block IDs to upload. The request is signed with the

Collection Write Key and sent to the server. The server validates the signature of the request before processing. Client and server then communicate to upload the encrypted blocks to the server.

Read a file

Since a file is always contained in a collection, the client first retrieves the appropriate Collection Read Key. The client uses its private User Key to decrypt the Collection Read Key, which is then used to decrypt file metadata, directory keys, and block keys. The client sends a request to the server (S-3 storage) to retrieve the appropriate blocks (as specified by block IDs). As each block is retrieved, it's decrypted by the client using the corresponding block key. The decrypted blocks are then assembled into a file in the user's file system.

Delete a file

To delete a file from a Collection, the client packages a request to the server to delete the set of blocks (specified by block ID) associated with the file. It then signs the request with the Collection Write Key. The server checks the signature, and if valid, marks the specified blocks for later deletion. Physical deletion occurs during periodic garbage collection and is dependent on a data retention policy specified for the Collection.

Share a Collection

In PreVeil, sharing occurs at a directory (Collection) level; individual files aren't shared. To share a Collection with other users, the sharing client retrieves from the server the public User Key for each shared user. It then encrypts the appropriate permission keys under each user's public User Key and creates a request for the server. For example, if the recipient is to have "Edit and Share" access, the sharing user encrypts the following keys under the recipient's User Key in the creation of the server request: Collection Read, Collection Write, Collection Share, ACL Viewer, Log. The request is then signed with the Collection Sharing Key (to prove to the server that the sharer has permission) and sends the request to the server. The server checks the signature and, if valid, adds the relevant keys to the Postgres database associated with the target Collection Server.

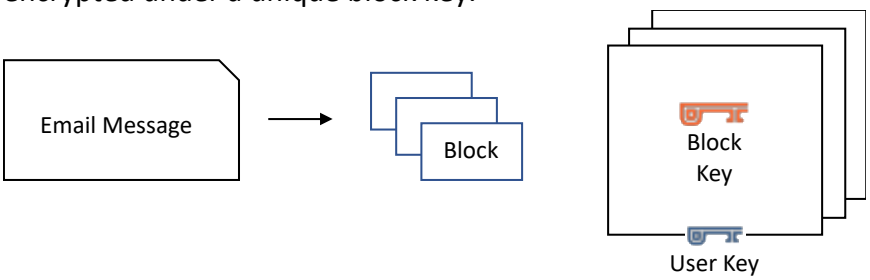
Un-share (and re-key) a Collection

To un-share or change any other sharing permission, the Collection is essentially re-shared with an updated set of keys. The client making the request generates a new set of collection keys (read, write, view ACL, etc.) and encrypts the appropriate ones under the User Key of each recipient. It then sends the request to the server to update all of the new keys with the new list of shared users and their permissions. As descripted above, the asymmetric key data structure includes a list of key versions with prior versions encrypted under the key of the latest version (so version 0 is wrapped under version 1, which is wrapped under version 2, etc.). This means that someone who no longer has access to a collection will be unable to decrypt the latest set of keys; nor will it be able to retrieve the list of prior versions and access them. It also means that files need not be re-encrypted when Collections are re-keyed, which would be extremely expensive from a computation and data transport perspective.

Email

Overview

Email uses the same storage mechanisms that files do. Email content (including attachments) is organized into blocks, encrypted with block keys, and stored in Amazon S-3 (or alternatives as previously noted). Each attachment is just like a file — organized into its own set of blocks each encrypted under a unique block key.



Email Messages partitioned into blocks then encrypted

A user's mailbox is organized as a special kind of Collection (it's called a Mail Collection), without all the usual collection read, write, and share keys. Instead, the User private key is used to decrypt items in this Collection, and the User public key encrypts the files that comprise email messages. This special collection's subdirectories contain the user's mailboxes — Inbox, Deleted Items, Sent Items, Drafts, plus user-defined mailboxes. Each mailbox subdirectory contains the files that comprise individual email messages and attachments. The Outbox is contained on a client only.

The Postgres database on the collection server contains metadata for the mailboxes, such as block IDs of the messages, encrypted block keys, and other message data.

Just as file collections have logs associated with them, the special User Collection also contains log entries that track email and other activities performed by the user. The mechanism works the same as with files: log entries are generated by the server, encrypted under the collection's Log Key, and hash-chained to prove authenticity.

Sending a message

To send a message, the client first packages the message and its attachments into blocks, creates block keys for each, and encrypts the blocks. It also prepares the message "snippet," which is contains the first few lines of the message which are to be displayed in the user's message list. The client retrieves public User Keys for each recipient of the message and then encrypts all of the block keys under each recipient's key. The client creates a server request to send the email, including in the request a list of recipients, block IDs referencing the blocks that contain message and attachment data, encrypted block keys for all the recipients, the encrypted message snippet, and other metadata. The request is then signed with the User Key

and sent to the server. The server copies the encrypted blocks to S-3 storage and updates recipients' User Collections with relevant message data including encrypted block keys.

Note that some recipients' User Collections may reside on different servers than the sender's. In that case, the sender's server will relay the message data to each of the recipients' servers for processing.

Receiving a message

Client devices regularly poll the server to see if any new messages are available. If a new message is waiting, server sends the client a list of block IDs, encrypted block keys, and message metadata. The client retrieves the message blocks from S-3 storage. It uses its User private key to decrypt the block keys and then uses the block keys to decrypt the message blocks and assemble them into an email message and file attachments. Before presenting the message to the user, the client checks the signature of the message (and attachments) to verify the sender's authenticity.

Deleting a message

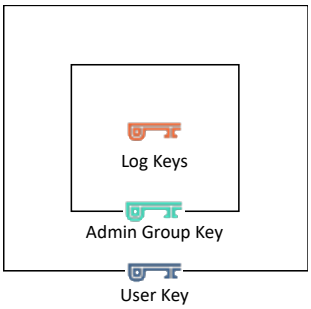
Deleting a message simply involves moving the message to the user's Deleted Items mailbox. Messages deleted from this mailbox are flagged as "expunged" and deleted during a garbage collection phase per a predetermined mail retention policy.

Enterprise Features

Overview & Organizations

The primary distinguishing feature providing for enterprise versus individual use of the PreVeil system is the notion of an **Organization.** Organizations have administrators to manage user accounts and, with certain restrictions, access user data. Enterprise users belong to an Organization, which can contain members from any email domain. Thus, Organizations can contain employees, subcontractors, partners, etc. who may not share a common email address domain.

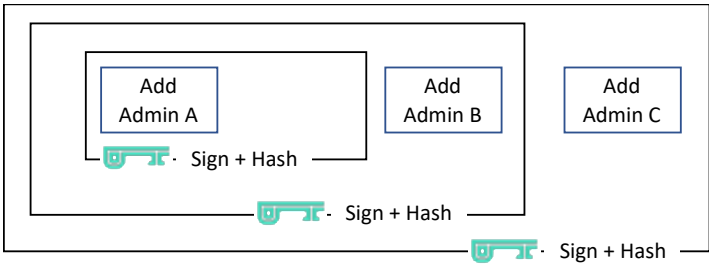
An Organization is identified by a unique ID. Its data is stored in the Postgres database on a specific server, just as users and Collections are managed by specific servers.



Admin Key Wrapping

Administrators

Organizations have Administrators, which are users with certain privileges. All the Administrators for an Organization together form an Admin Group. Members of an Admin Group have access to an asymmetric Admin Group Key, which is encrypted under each member's User key.



Hash-chained Administrator List

The server keeps a list of the members of the Admin Group that's comprised of the original Administrator and a sequence of changes (i.e. add/delete) to the membership of this group. Since only administrators can add or remove users from the Admin Group, each change is

signed by the admin that generated it, with changes hash-chained together. Since administrators have special privileges, the signed and hash-chained entries guarantee that that the Admin Group list is authentic.

Any change to the Admin Group requires the Admin Group Key to be regenerated. The change is processed on the client device of the administrator making the change, with the new key re-encrypted with the User Key of each member of the new group.

Organization-wide Event Stream

One reason for the special care taken to authenticate the Admin Group has to do with a powerful mechanism called the Event Stream. In PreVeil, many cryptographic operations need to be performed on the client as the server is trusted as little as possible. For example, when a user's Approval Group changes, the shards of the key to be recovered by the Approval Group are created on the client, encrypted, and pushed to the server. In an enterprise, it's often an administrator who would manage changes to Approval Groups. It would be impractical to expect each user to manage the changes themselves. When an administrator changes an Approval Group, the server sends a request to each affected client that directs each client to generate new keys and shards and re-encrypt them for the new approvers. The mechanism whereby the server sends requests to clients is called the Event Stream. Before a client executes an event stream request from the server, the client validates the request by checking the signatures on the hash-chained Admin Group list. Only then can the client prove that the request was initiated by a valid administrator.

Enterprise Approval Groups

Approval Groups and their application in recovering private keys have been discussed here:

Approval Groups. However, in enterprise settings, Approval Groups can also be used to authorize privileged actions.

When Approval Groups **authorize a server operation**, the server sends a request to each approver, and the approvers reply with a response signed by their User Key. Once the required number of approvals is obtained, the server proceeds with its operation.

Logs

Since administrators have access to logs of all activity on the system, all log keys are encrypted under the Admin Group Key. Note that log entries are created by the server, but have metadata encrypted by a client (such as file names). A user's client traversing a log of their own data has access to the Collection Read Keys necessary to decrypt the log metadata. An administrator to whom the Collection has not been shared will only be able to see the nature of the log entry (e.g. a file was changed), but not the metadata (e.g. the file name).

Administrative Operations

Creating an Organization

An Organization is created when a user, who has already installed PreVeil as an individual member, visits a special URL (provided by PreVeil) in a browser. The client then has the user fill out a form with organization data and establishes that user as an Administrator. An Admin Group Key pair is created on the client, signed with the User Key, and the public Admin Group Key is pushed to the server. The private Admin Group Key is encrypted with the User Key and sent to the server. The admin can then proceed to create accounts for other users to the organization, set up Approval Groups, and promote certain users to Administrators.

Adding Users and Creating User Accounts

For an administrator to create an account for a user, they will first ensure that the user has PreVeil on their target device – either by asking the user to install PreVeil or by automatically installing PreVeil through support tools such as Active Directory. The administrator then enters the user's name and email address, upon which the server sends an email to the user's (non-PreVeil) email account with a link containing a URL to the PreVeil app and a unique code. The user clicks (or taps) on the link in the message which launches the PreVeil web app (on a computer) or the PreVeil mobile. The client app then generates the User Key pair for that user and sends the public User key to the server along with the unique code from the email message that identifies that user. It also creates a Log key for that user, if the client is part of an organization, encrypts the Log Key under the Admin Group Key. The server proceeds to set up the appropriate data structures for that user — both on the Postgres database as well as in a special User collection in S-3 storage.

Configuring an Approval Group

When an admin sets an Approval Group for one or more users, the admin's client creates a server request that includes the IDs of each approver in the (new) group and a list of users for whom the Approval Group will apply. It signs the request with its User Key. If an Approval Group has already been set for one or more of the affected users, the change must first be authorized by the existing Approval Group (see below for the mechanics of this authorization). Once the change has been authorized, the server uses the Event Stream mechanism described above to request each user's client to make the Approval Group Change. Each client receives a request via the Event Stream. The client first authenticates the request by ensuring that the request is signed by a currently valid administrator (by checking the list of signed and hash-chained changes to the Administrator Group). Each client then generates a new User Key, creates shards of the private part of the User Key for each approver, and encrypts each shard with the public User Key of each approver. The public part of the User Key as well as all the new encrypted shards are pushed to the server.

Changing approval groups

An Approval Group may be modified only if the current approvers authorize the change. Approval Group changes are processed asynchronously and can be requested by an admin for a group of users or by an individual user. An admin requesting a change for a group of users is processed as a batch of individual change requests. The client initiating the change creates and signs a request that includes the new set of approvers and sends this request to the server. The server validates the request and then sends requests to the clients of each approver. Each approver that concurs with the request sends a signed response back to the server.

When the server receives the required number of approvals, it uses the Event Stream to direct each affected user's client to re-key its User and other associated keys (e.g. Collection keys, etc.), generate new shards for the new approvers, encrypt the keys and shards, and post the new encrypted keys to the server.

Approval Group Functions

There are two ways in which approvers authorize a transaction through an Approval Group: asynchronously and synchronously. Synchronous authorization requires the requester to interact with an approver in real time. Asynchronous authorization allows approvers to respond to requests at their convenience, and multiple requests can be posted for a single approver who can process them in a batch. Importantly, *synchronous* approvals are processed on the *client* while *asynchronous* approvals are processed by the *server*.

Synchronous Authorization

This type of processing is required whenever Approval Groups are used to recover private keys, as the server is never allowed access to these keys. Synchronous Approval Group processing is used when recovering a User Key and for Data Export (see below).

Asynchronous Authorization

This type of processing is used to authorize other server-based transactions. These include promoting users to administrators, deleting users, and approving changes to Approval Groups. The server manages the process of authenticating asynchronous approvals. It sends a request to the client of each approver, which must sign its response. Once the minimum number of required approvals have been received and authenticated, the server proceeds with its transaction.

Recovering a user key.

If a user has lost all their keys (i.e. no longer has devices containing their User key), the key can be recovered using the synchronous approval process. The user can run this process directly, or an admin can do this. In either case, the process must be run on the user's device as the recovered key will be generated and stored there. If the key is to be recovered on a device

without PreVeil installed, PreVeil must be installed first, and at the end of the installation process, the user can choose to recover an existing account using their Approval Group.

The user's client generates a request for the server containing the user ID associated with the keys to be recovered and sends it to the server. The server sends an email to the user's regular (non-PreVeil) account for confirmation, along with a link with a secret code. When the user clicks on the confirmation email, the secret code is sent to the PreVeil Web app (on a computer) or PreVeil mobile app, which passes the code to the server. The server then looks up the recovery Approval Group for the user and sends the list to the client. The client presents the list of approvers to the user who then chooses approvers one at a time for approval. For each approval, the approver's client requests the name of the user whose key is to be recovered and uses this information to set up a connection with the client making the request.

The next step is to set up a secure channel is set up between the user and the approver using the Diffie Hellman Key Exchange protocol. To do this, the user's client generates an 8-digit code and presents it to the user, who reads it to the approver for entry into the approver's client. This 8-digit code forms the basis of a shared secret between approver and client necessary to establish the secure tunnel. The approver's client then retrieves the shard of the user's key that's been stored on the server, encrypted under the approver's private User Key. The approver's client decrypts the shard using the approver's private User Key and sends the shard via the secure tunnel to the user's client. The user client repeats the process for all necessary approvers, at which time the private User Key can be recovered using the Shamir Secret Sharing algorithm. The user's client can now access all of the user's data and keys (e.g. Collection Keys, Log Keys, etc.).

Having been recovered, the User Key must now be re-keyed to prevent the decrypted shards from being used again. Approvals are required for *each* key recovery. A new User Key is generated with the public portion pushed to the server. All of the user's other keys (e.g. Collection, etc.) are re-encrypted with the new User public key and pushed to the server.

Data Export & E-Discovery

Data export is a powerful feature whereby some or all an organization's data is decrypted and exported. A special Approval Group is assigned to this operation as organizations may want more stringent approval for this operation. When the Data Export Approval Group was created, each user's client was notified via the Event Stream to create shards of its User Key for this special Approval Group. This means that the Shamir Secret Sharing algorithm was used to create shards of the User Key, and then each of these shards was encrypted with each approver's public User Key and saved on the server.

Only administrators may request export of data. The process begins with an admin selecting the set of data to export, which can be all data, or a subset based on a date range or a list of users. The admin's client creates the data export request, signs it, and sends it to the server. The server initiates the synchronous Approval Group authorization process described above for

the required members of the Data Export Approval Group. The process is similar to recovering a single User Key, except the approver clients recover shards for *the requested* User Keys in the organization and send them to the admin's client. In other words, each Data Export approver is going to retrieve, decrypt, and send to the admin a shard for *each requested* User Key in the organization. Think of the process as an automated way for an approval group to authorize the recovery of each user's key. The admin's client, again using the Shamir Secret Sharing algorithm, reassembles the shards to re-create all User Keys.

The admin client then proceeds to traverse all Collections associated with the organization, using User Keys to decrypt the data, and placing the decrypted emails and files in a designated directory accessible to the admin client.

Once the process is completed, all User Keys are re-keyed and associated wrapped keys updated on the server.

Other administrative operations

Other admin operations, such as deleting a user and promoting/demoting admins may be authorized by an Approval Group. If no approval group has been specified, the server allows an admin to perform the operations directly. If an Approval Group exists, the server uses the asynchronous authorization process (detailed above) before allowing the operation.

Promoting a user to an administrator

This action involves sharing the Admin Group Key and updating the Admin Group. The admin client initiating the request encrypts private Admin Group Key under the new admin's User Key. It also updates the Admin Group. Recall that the Admin Group data structure contains a list of changes to the Admin Group. The client updates this list with a notation that the new admin has been added, signs the new list, and pushes it to the server. The Admin Group Key is now re-keyed, with the new private Admin Group Key encrypted with the public User Keys of each member of the new Admin Group. The new public Admin Group Key and encrypted private Admin Group Keys are then pushed to the server.