Filen

Security Whitepaper

August 2021

Filen Cloud Dienste UG (haftungsbeschränkt)

Content

1 Introduction

1.1 What is Filen?

Filen is a fully zero knowledge end-to-end encrypted cloud storage and communication platform.
Zero knowledge end-to-end encryption means that there is no intermediary. Filen (or any of it's employees) never has access to the user's encryption keys and therefore data transmitted or stored on the platform truly remains in the user's hands.

1.2 End-to-end encryption and zero knowledge

Unlike most other cloud storage platforms, on Filen only the user can access data stored on the platform. From the ground up Filen was designed around user controlled end-to-end encryption. Data is encrypted on the user's machine before being transmitted to the platform. Only the user holds the encryption keys, not even Filen has access to them. If a user wishes to share data, they will encrypt the required encryption key with the recipients public key before transmitting them. This process guarantees 100% data ownership and privacy.

1.3 Transparency

All cryptographic operations needed take place directly on the user's machine. Filen provides transparency of the implementation by publishing the full up-to-date source code of it's applications. Links to the source code can be found on the main homepage.

1.4 Privacy

By using zero knowledge end-to-end encryption, Filen provides privacy by design. Unlike most other cloud storage and communication platforms, Filen's user controlled encryption makes sure only the user ever has access to data stored on the platform (except for when encryption keys are voluntarily shared). Even folder names, file metadata (such as mime type, size etc.) are encrypted. Filen does store various other transactional metadata, such as the user's email and IP address, to which privacy by policy applies. Filen's privacy policy can be found on the main homepage.

1.5 Vulnerability management

To ensure Filen remains secure at all times, Filen offers rewards to anyone who reports a previously unknown vulnerability or bug. If you find something you can contact our support and our development team will get back to you as soon as possible.

1.6 Redundancy

Filen directly operates it's infrastructure and does not rely on any third party cloud providers. All hardware is hosted in secure facilities located in Germany. No data is stored in the United States of America.

Filen has various categories of infrastructure, including:

1. Web cluster consisting of multiple nodes and load balancers to ensure high availability
2. API cluster
3. Database cluster of multiple nodes replicated to multiple datacenters
4. Storage cluster using 6+3 erasure coding split into multiple datacenters to ensure data consistency and availability. Failed hard drives can be hot swapped in minutes.
5. Realtime message broker service
6. Miscellaneous services like Filen's speedtest node and self hosted analytics service

1.7 Compliance

Filen is operated to achieve highest level of compliance with regulatory requirements. Filen's service is governed by German law. Even though it's impossible for Filen to view data stored on it's platform, it prompty removes copyrighted content when reported. Filen users can share data through the platform in the form of public links. These public links have the necessary decryption key appended to the URL hash when a user chooses to create them for a file/folder. When Filen receives abuse reports or notices it promptly removes or disables access to the offending file or files (folders includes), depending on the type of request, consistent with the Terms of Service agreed to by every registered user.

2 Registration and login

2.1 Registration process

When creating an account the user is required to enter his email address and password. Filen does not store passwords directly. For password processing, Filen uses the established PBKDF2 standard. While this is not the latest, best or state of the art, it is well known and has wide language support, especially in the WebCrypto API, where it importantly performs at native speed.

The password hash is computed as follows:

Salt	=	cryptographically secure generated random 256 character value
Hash	=	SHA-512
Iterations	=	200.000
Bit Length	=	512

Derived Key = PBKDF2(password, salt, iterations, hash, bitLength)

This operation will result into a 512 bit key which will be converted to hex characters.

The derived key will then be split evenly (from left to right). The first half will act as the user master key and the second half will be hashed again using SHA-512 and then act as the authentication key.

This data (user email address, salt and hashed authentication key) will then be sent to the API and an account will be created. After confirming the email sent to the user’s email address, logging in will be possible. Filen never stores unhashed authentication keys. Each hashed authentication key is hashed again server side using Argon2 to prevent pass the hash attacks in case of data leaks.

2.2 Login process

The login process works as follows:

1. The user must enter their email address and password into the client interface
2. The email address is sent to the API
3. If a record with the email address exists in the database, the API will respond with the user's salt
4. If the email address is not found in the database, the API will respond with a randomly generated salt, preventing email address brute force attacks
5. The client can now compute the user's master key and hashed authentication key as explained in the registration process
6. With the hashed authentication key computed, the client will send the email address and hashed authentication key to the API, which will then respond with the user's API key if authentication was successful
7. If it's the user's first login, the client will encrypt their master keys and send it to the API. A RSA-OAEP keypair (4096 bit modulus, SHA-512 hash) is also created, encrypted using the user's master key (AES-GCM 256 bit) and send to the API. Filen never stores unencrypted keys. The encrypted keys will be used to make sure other clients the user may use are able to decrypt data. In case of a password change for example, Filen appends the new derived master key from the new password to the old master key. Filen calls this master key chaining, which makes password changes easy. This process is explained in more detail in the account recovery section.

2.3 Account recovery

Since the user's password is the root of all client-side encryption, forgetting it results into the inability to decrypt the user's data, which will make it impossible for either Filen or the user to recover the data stored on the user's account.

If the user is still logged in on a different device, he can change his password. Changing the password will generate a new master key as explained in the registration process. This master key is then appended to the old master key, encrypted and sent to the API. Filen never stores unencrypted keys. Filen calls this master key chaining. This process makes decrypting data encrypted with the old master key possible, while new data is encrypted using the new encryption key.

2.4 Two Factor Authentication

Filen has implemented Two Factor Authentication (2FA) using Time Based One Time Passwords (TOTP). The shared secret generation uses 32 random bytes, this is converted to Base 32 and displayed to the user as a QR code and the plain Base 32 string. On activation, the user will be displayed a recover key, which they can use to recover their account by contacting our support in case of 2FA device loss.

3 Cloud Drive encryption and decryption

3.1 Data upload encryption

Each file has its own encryption key. File data, name, metadata and folder names are encrypted.

For encrypting files, a cryptographically secure random 256 bit key will be generated. Folder names are encrypted using the user's master key.

Each file gets split into 1 MB chunks, encrypted using AES-GCM 256 bit (authenticated to ensure data integrity) and is then uploaded to the API.

File metadata, including the file encryption key is then encrypted using the user's master key and sent to the API aswell.

File	= AES-GCM(chunkData, fileKey)
File metadata	= AES-GCM(fileMetadata, userMasterKey)
Folder name	= AES-GCM(folderName, userMasterKey)

3.2 Thumbnail generation

The client creates image thumbnails by downloading the encrypted data, decrypting it, resizing the resulting image, compressing it and then storing it in the client's local storage, where it can be re-used. Video previews are created almost the same way, except for not downloading the whole video but just the first few chunks to save bandwidth and local storage space.

3.3 Downloads

The client downloads a file by downloading the encrypted chunks, decrypting them and then streams the decrypted chunks in correct order into the user's local file system or Browser download directory.

Folder downloads work the same way, except for everything being zipped client sided. This makes sure the user does not download a lot of individual files. Since browsers have a memory limitation, folder downloads are limited in size. A user may download Filen's desktop client for unlimited size folder downloads.

4 Collaboration

4.1 Sharing data with other Filen users

When a user chooses to share a file or a folder with another Filen user, they will encrypt all metadata using the recipients RSA-OAEP public key.

File metadata = RSA-OAEP(fileMedata+fileKey, recipientPublicKey)

Folder metadata = RSA-OAEP(folderName, recipientPublicKey)

All data will then be sent to the API where it will be made available to the recipient. The recipient can then decrypt everything using their RSA-OAEP private key and access the shared data normally.

4.2 Public links

When a user chooses to share a file or a folder using a public link to non Filen users, the client will generate a new cryptographically secure 256 bit key. This key is then used to encrypt all metadata. The key will be appended to the URL hash of the public link. The key is also being encrypted using the user's master key and saved for later use.

File metadata = AES-GCM(fileMetadata+fileKey, newEncryptionKey)

Folder metadata = AES-GCM(folderName, newEncryptionKey)

Public links can be configured to expire or be password protected. The password hash for the public link will be derived from the original password using PBKDF2(password, salt, iterations, hash, bitLength), where salt is a random 256 bit string, iterations set to 200.000, hash set to SHA-512 and bitLength set to 512.