Nextcloud

# Server-side Encryption

## Securing data at rest

nextcloud.com

Nextcloud

# Contents

# Nextcloud Server-side Encryption

Nextcloud provides an enterprise file synchronization and collaboration solution developed in alignment with industry standards such as Clause 14 of ISO/IEC27001-2013 and related standards, guidance and security principles. Built around combined assurance layers of rich security features, applied best practices governed by policy and a design itself validated by industry standard testing processes. One component part of this multi-layered security approach is enterprise-grade, server-side encryption employed to ensure data is protected at rest. This white paper describes the design of Nextcloud Server-side Encryption, its benefits and limitations.

## Place in the security stack

Nextcloud's Server-side Encryption (SSE) is designed to complement other security layers at the disposal of system administrators. These include TLS at the transport level, End-to-end Encryption at the client (Client-side Encryption, CSE) and Two-Factor Authentication to protect authentication. Passive security features like Brute Force Protection, Content Security Policy, code signing and others harden the server, providing additional protection. Nextcloud further employs a series of automated checks with associated warnings to system administrators in monitoring and securing their system. These features are built as part of a third-party verified design and development process and backed by a transparent vulnerability disclosure process and industry-leading Security Bug Bounty program.

## Server-side Encryption Threat Model

Server-side Encryption was designed to protect from a specified and defined Threat Model. This enables users to decide whether it is useful in their specific circumstances.

- Server-side Encryption is designed to protect the content of files on **external storage locations** and on the server itself only **while at rest**, that is, when a user is not actively accessing the data.
- The encryption keys do not leave the Nextcloud server and are encrypted with a server key **which is stored on the server** or, optionally, with a per-user key using user passwords **which are not stored on the server**. In both cases, it means a third party storage like an object storage in the cloud or a Network Attached Storage will only ever be used to store encrypted content and **gain no access to the data**.
- The names of files and folders are **not encrypted** (though, when using object storage, these are not leaked either).
- Server-side encryption only encrypts data in the files, files_versions and files_trashbin folder. **File previews or a full-text search index remain unencrypted**. These should be disabled on systems where this is a problem.
- It does not encrypt files on external storage added through direct access rather than via Nextcloud.
- Encryption can be enabled or disabled separately for each storage. That includes primary storage.

- When using the server key, the password is stored in config.php and thus can be found on the server and a stolen drive, which contains the config file, can be used to decrypt data. This can be prevented using per-user keys. Per-user keys come at a performance and feature costs. Encryption is slower and group shares don't adjust automatically, encryption key passwords must be changed when the password changes, if the user loses his/her password, all access to data is lost, the impersonate app does not work and user-backends which don't provide a user password (like most SSO backends) don't work.
- When using per-user keys, the cryptographic key used to encrypt the files is stored in encrypted form when the user is not logged in. During file access, the key is decrypted with the users' password and stored as part of the user session information. This session information is encrypted at all times with a token provided by the user's client device. A sufficiently advanced, malicious server administrator or an adversary who gained full, unlimited access to the server would, however, be able to intercept keys when they have the capability to watch the entire process. When looking at data at rest such as in case of a stolen hard drive or powered-down server, these would only contain encrypted content.
- Nextcloud Server-side Encryption is designed to work with an external key storage mechanism, including a Hardware Security Module. This enables IT departments to adjust the way data is stored and encrypted to comply with legal or practical requirements and guidelines.

## Benefits

- Nextcloud Server-side Encryption enables enterprises to use third party, cloud based file storage solutions while maintaining security and confidentiality of data, encrypting it before it leaves the control of IT.
- Server-side Encryption employs proven, widely adopted technologies and standards like OpenSSL and AES-256 as recommended by organizations such as NIST.
- Server-side Encryption is designed to perform well for large organizations and large files.
- Nextcloud's Server-side Encryption can be customized to match the requirements of enterprises like custom encryption algorithms and tools or special key management technology.

# Modular Design

The Server-side Encryption app is designed in a modular way, providing a framework for 'encryption modules' to execute the actual encryption and key handling. Nextcloud provides a 'default' encryption module, with the internal ID **"OC_DEFAULT_MODULE"**. It has been designed to encrypt data at rest and handle encryption keys in a simple but secure matter on the Nextcloud server itself.

A third-party encryption module can be written implementing the interface to:

`\OCP\Encryption\IEncryptionModule`

This enables custom encryption methods and algorithms as well as custom key management fitting the need of a specific use case. We will not further cover this option, however, Nextcloud GmbH offers consulting for enterprises who need to implement their own encryption module.

# The default encryption module

The default encryption module creates user keys, encrypts files and enables sharing. It encrypts the following data:

- Files created or uploaded via WebDAV. This includes the Nextcloud web interface and clients as all are designed to use WebDAV.
- File versions and files in the trash bin

We will describe the processes of generating keys, sharing, encrypting and decrypting files. The process below assumes per-user keys. With a server key, the same key generation and encryption/decryption process is followed but in a simpler way: only one private / public keypair is generated and used to encrypt and decrypt all files.

## Generating user keys

When the system administrator enables the module, keys have to be generated for users.

1. Nextcloud generates a private/public keypair for every user upon their first login. The RSA key is 4096 bits and is generated using "`openssl_pkey_new`".
2. A private key password is generated by running the users' login password through a PBKDF2 key derivation with 100,000 iterations, a salt which is specific for the Nextcloud instance and 32 (AES-256) or 16 (AES-128) byte key size (this can be configured). This password is used to encrypt the private key of the user. When using an instance key (default), this password is stored in config.php and used to generate the key.
3. The keys are stored in **"data/$username/files_encryption/OC_DEFAULT_MODULE" as "$username.publicKey" and "$username.privateKey".**
4. Two more key pairs are generated for the system, one for public link shares and optionally one for recovery. These are stored in the encryption folder (data/files_encryption).

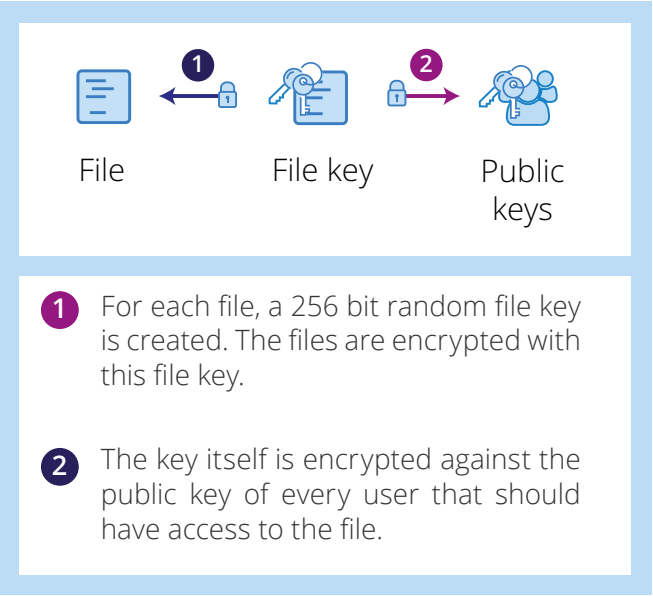
**Note:** System administrators can enable the recovery key feature which allows them to recover data in case a user forgot their password. However, users have a choice in enabling this feature for their account. This results in effectively having a server key, increasing performance and adding the features like group handling and SSO support that are associated with a server key.

## File key handling

Each file is encrypted with its own, unique *file key*.

1. For each file that is encrypted, a 256 bit random *file key* is created and stored at **"data/$username/files_encryption/keys/files/$filepath/$filename/OC_DEFAULT_MODULE/fileKey".**
2. The files are encrypted with this *file key* while the key itself is encrypted against the public key of each user that should have access to the file (or the server key if enabled). This sharing encryption step is done using a *share key*, a 128 bit long random secret used to control file access, unique for each user that has access. These are stored in **"data/$username/files_encryption/keys/files/$filepath/filename/OC_DEFAULT_MODULE/username.shareKey"**. The encryption of the file key to the public keys is done using openssl_seal in RC4 mode with the *share key*.

Graphic: File key handling



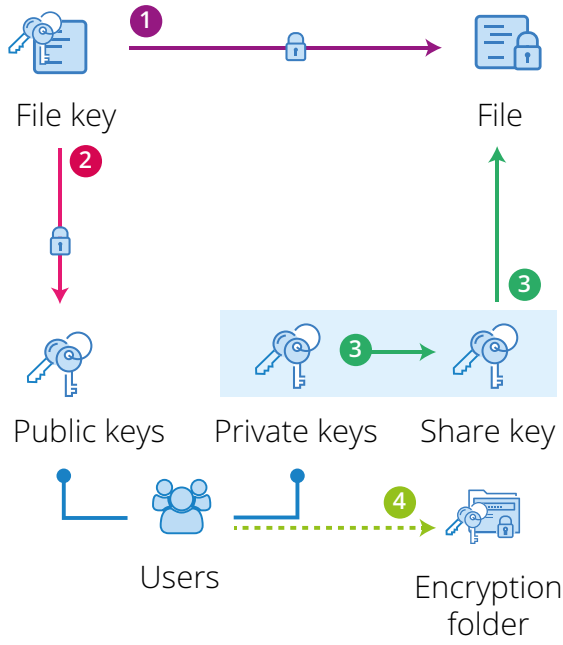

**Note:** The *share key* mechanism is used to avoid having to re-encrypt the files themselves when new users are given access or when access is revoked. This saves significantly on the server overhead of the default encryption module.

**Note:** To protect data against modifications, an HMAC is calculated for each chunk of encrypted files by hashing together the file key, file version and location of the chunk appended by "a". When reading files, the integrity is verified and users and system administrators are warned when problems are found.

## File and sharing operations

To access or share files, a series of encryption operations is required. This description is assuming the per-user key. With the server key, the encryption only happens once against the server key and not against user keys.

- **When a new file is uploaded or created by a user**, Nextcloud generates the associated *file key* and encrypts the file with it. It then encrypts the *file key* to each public key of each user that should have access to the file - creating secure, 128 bit *share keys* which users can decrypt the file with using their private key. The encrypted *file key* is stored in the encryption folder for the user.

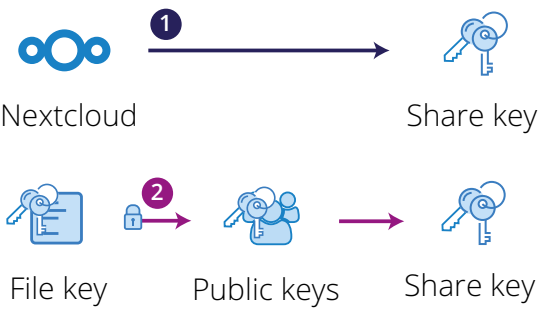



1. Nextcloud generates file key and encrypts the file with it.
2. File key is encrypted to each public key of each user that has access to the file.
3. Creates a secure 128bit share key which users can decrypt the file with using their private key.
4. The encrypted file key is stored in the encryption folder for the user.

- **When a new user is given access to a file**, Nextcloud generates a new 128 bit *share key* by encrypting the *file key* against all authorized users' public keys.

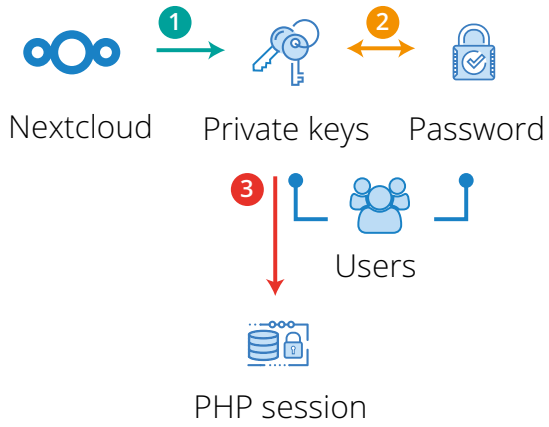

Nextcloud generates a new 128 bit share key by encrypting the file key against all authorized users' public keys.

- **When authorization to access a file is revoked**, Nextcloud removes the *share key* and re-encrypts the fileKey with the public keys of all users who have access to create a new one. Note that this requires the user who 'owns' the file to be logged in. Adding and removing users from groups can thus be done, but the new users only get access after the owner of the file logs in and Nextcloud gets a chance to re-generate the *share key*. This limitation is lifted when using the server key.




1. Nextcloud removes the share key.
2. Nextcloud re-encrypts the file key with the public keys of all users who have access to create a new share key.

- **When an authorized user accesses a file**, Nextcloud requires the users' *private key*. This key has been decrypted when the user logged into the server using the users' login password. The decrypted *private key* has been stored in the users' PHP session.




1. Nextcloud requires the users' private key.
2. The key has been decrypted when the user logged into the server using the users' login password.
3. The decrypted private key has been stored in the users' PHP session.

**Note:** The PHP session is encrypted by Nextcloud with AES-128 and a cookie sent by the user for each request to ensure the *private key* (or any other user data) is not stored unencrypted on disk. The *private key* is used to decrypt the *share key* to give the *file key* which is in turn employed to decrypt the file.

## Summary

The Nextcloud Server-side Encryption offers a secure, well performing, storage-independent solution to add an additional layer of protection for data at rest and on external storage. The implementation is uniquely flexible, enabling enterprises to use custom key management solutions or encryption algorithms.

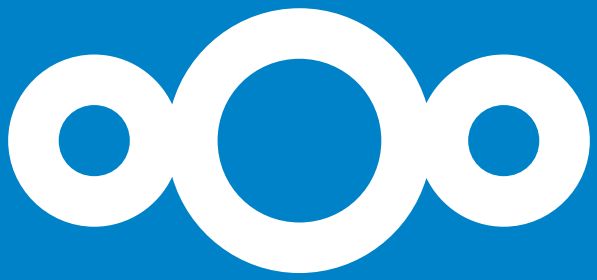
As part of the expansive set of Nextcloud capabilities developed under a strategic focus on security, Server-side Encryption can play an important role for enterprises looking to provide the utmost security for their data.

Contact our sales team for more information:

**Andreas Rode**
Head of Sales

Email: sales@nextcloud.com
Phone: +49 711 252428-94

nextcoud.com



Nextcloud GmbH
Hauptmannsreute 44A
70192 Stuttgart
Germany

Email sales@nextcloud.com
Phone +49 711 252428-90
Fax +49 711 252428-20

nextcloud.com