Data protection in Pycter

End-to-end encryption of organisation albums

November 2026

Executive summary

Pycter transfers photos automatically between devices.

Technical protection. With end-to-end encryption, content is encrypted on the sending device, and can be decrypted only by devices holding the keys. The keys never reach Pycter's servers unencrypted. Neither Pycter's servers nor its providers can read the content, only some metadata, listed below.

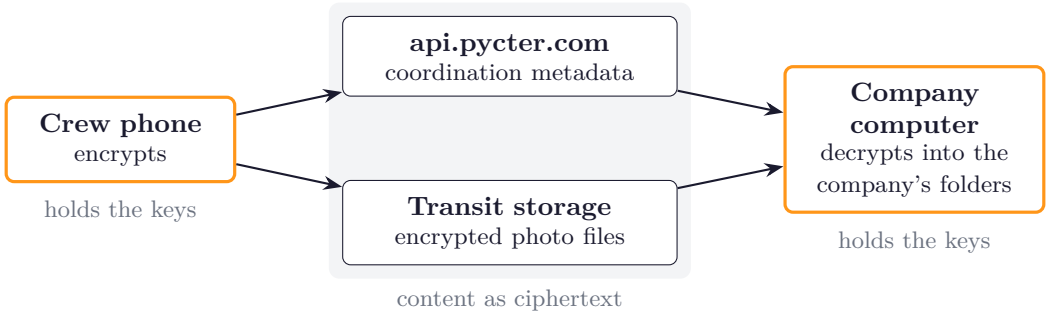

Figure 1: The keys exist only at both ends. Pycter's servers see the content only as ciphertext.

Pycter's servers cannot read	They can read
Photos and videos, at every size	Who the members are (account email and name)
File names and locations	When each photo was taken
Album title, descriptions, status, nicknames	How many files there are, and their sizes
The album key	When members are active, and from which IP address

Where. The API, the database and file storage are hosted in the EU: Google Cloud in Belgium, Cloudflare R2 in Western Europe.

On-premise storage. For more control, Pycter can host the file storage on the company's own network, in place of Cloudflare R2, so that the encrypted photo files stay with the company. `api.pycter.com` still carries accounts, membership and the encrypted album updates (Section 2.2.1).

Legal protection. Pycter uses content and metadata only to run the service. It never sells them, and never uses them for advertising, profiling or AI training. It reports any breach to the company without undue delay, and deletes the company's content and metadata within 90 days after the company stops using Pycter (Section 3).

Organisational protection. Roles limit who can change an album: only the company's admins can delete or edit photos, remove members or change an album's settings, while members can only add. If a key leaks, removing the person cuts them off, and a new album brings a new key (Section 4). At Pycter, only administrators have access to production.

Contents

1 Introduction **3**

2 Technical protection **3**

2.1 Threat model . . . 3

2.1.1 Trusted . . . 3

2.1.2 Not trusted . . . 3

2.1.3 Out of scope . . . 4

2.2 Architecture . . . 4

2.2.1 On-premise storage . . . 4

2.3 Layer 1: membership . . . 5

2.4 Layer 2: encryption . . . 5

2.4.1 Keys and cryptographic primitives . . . 5

2.4.2 Key distribution . . . 6

2.5 Data on Pycter's servers . . . 7

2.5.1 Identical files . . . 8

2.6 Infrastructure and operations . . . 8

2.7 Devices . . . 9

2.8 Known limitations . . . 9

3 Legal protection **9**

4 Organisational protection **10**

4.1 At the company . . . 10

4.1.1 Compromised key . . . 11

4.2 At Pycter . . . 11

5 Auditing Pycter **11**

A Cryptographic parameters **11**

B Network requirements **12**

1 Introduction

This document answers three questions: who can see an album's content (Section 2), what Pycter commits to doing with content and metadata (Section 3), and who can change an album and how incidents are handled (Section 4). Section 5 lists how the company can check these answers itself, and Appendix B lists the hosts the company's network must allow.

2 Technical protection

This section describes what the software and the cryptography enforce, whatever anyone decides: who can see an album's content, and what Pycter's servers can read.

2.1 Threat model

This subsection states what the protection trusts, what it does not trust, and which cases it does not cover.

2.1.1 Trusted

Pycter clients. The apps Pycter publishes, which hold the keys on members' devices.

Platforms. Each client relies on the platform it runs on, which the company already trusts with the device as a whole: Apple on iPhone, Google on Android, Microsoft on Windows.

2.1.2 Not trusted

Party	What they could obtain	Why it is useless to them
Pycter server	Encrypted content, and the metadata listed in Section 2.5	The album key is never on the servers unencrypted (Sections 2.4.1, 2.4.2)
Pycter's providers	At most the same, for their part	Same
Other Pycter users	Nothing	The server refuses non-members, and they lack the key (Section 2.3)
Former members	Nothing added after their removal	Removal takes away membership
The network	Nothing	TLS, and the content inside is already encrypted

A breach of Pycter's servers. Someone in control of the running servers could tamper with how encryption is turned on, since it is turned on from a server flag. The keys are never on the servers unencrypted, so stored content stays unreadable. Mitigations:

- *Encryption is one way.* It cannot be turned off from the server: once a device encrypts an album, nothing makes it stop.
- *Each device shows it.* The album settings show, on every device, that an album is end-to-end encrypted (Section 5).
- *Pycter notifies the company* of any breach (Section 3).

Outside Pycter. Invitation links do not go through Pycter's servers: the app that carries a link sees it, and so does Google Play on Android (Section 2.4.2).

2.1.3 Out of scope

- *Someone publishing an app in Pycter's name.* Only the holder of Pycter's signing keys can publish an app or an update that devices accept, and such an app could bypass everything in this document. Phone apps come only from the App Store and Google Play; the desktop app installs only updates signed by Pycter.

2.2 Architecture

This subsection gives the overall picture: the components, what the server keeps for each album, and the two layers that protect the content.

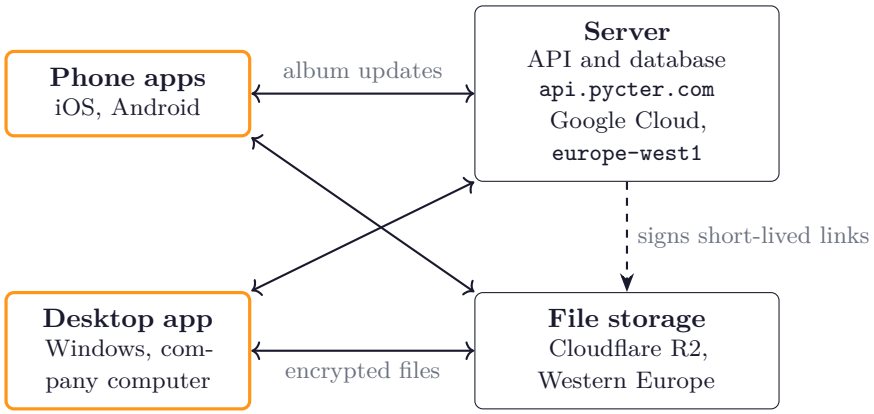

Figure 2: Components. Orange boxes hold keys; the others handle the content only as ciphertext.

The server is a relay. It keeps two things for each album:

- *The album updates*: every change to the album, such as a photo added or the title changed, stored encrypted in the database (Section 2.4.1).
- *The files*: each photo is encrypted on the device, at several sizes, and travels through short-lived signed links straight between devices and file storage; the API only issues the links.

Two layers. Two independent mechanisms stand between an outsider and the content:

1. *Membership* decides who may *fetch* the ciphertext. The server enforces it on every request (Section 2.3).
2. *The album key* decides who can *read* it. Only devices of members hold it (Sections 2.4.1, 2.4.2).

Reading the content takes both.

2.2.1 On-premise storage

This subsection describes the option of keeping the encrypted files on the company's premises, in place of Cloudflare R2.

The storage. Pycter deploys an S3-compatible object storage, as a Docker container, on the company's network. It holds every encrypted file, at every size: no file reaches Cloudflare. Accounts, membership and the album updates stay on `api.pycter.com`.

Reach. By default, the storage is reachable only from the company's network: phones transfer files on the company's Wi-Fi or VPN, and wait until then. The company can also expose it through its

firewall, so that phones transfer from building sites.

Links. The server signs links to the storage as it does for Cloudflare R2, with an access key to the storage. Signing needs no connection: Pycter's servers never connect to the storage. Unless the company exposes it, the key is useless outside the company's network.

Certificate. Phones accept only HTTPS links. The storage's certificate is issued by the company's internal certificate authority, which the company installs on its devices.

Retention. The company sets how long the storage keeps files; Pycter's servers cannot delete them.

2.3 Layer 1: membership

The first layer decides who may fetch an album's ciphertext. This subsection describes how the server authenticates devices and checks membership.

Every request is authenticated. Each device generates a 256-bit random token and keeps it on the device: in the Keychain on iPhone, in the app's private storage on Android, and in the app's database on the computer. The server keeps only its SHA-256 hash. Accounts sign in with a one-time code sent by email. Firebase App Check attests that certain account requests come from the genuine phone apps.

Every access is checked. Before it serves or accepts an album update, or issues a link to a file, the server checks that the device's account is a member of the album. Links to files are valid 15 minutes for upload and 1 hour for download, and name the stored object by a number that says nothing about its content.

Encryption is locked on. An album can join an encrypted organisation only if it is encrypted. Inside the organisation, nobody, admins included, may send the update that changes encryption, and no update turns encryption off.

Removal. When an admin removes someone from an album, the server deletes their membership and every pending invitation to that album. From then on, it refuses them all updates and all files, apart from download links issued before the removal, which stay valid at most 1 hour. Removal takes away membership; the album key itself does not change, so what the person received before their removal stays with them, like any photo they could have saved while they were a member.

2.4 Layer 2: encryption

The second layer decides who can read the content. This subsection describes the keys and primitives (Section 2.4.1) and how the album key reaches members' devices (Section 2.4.2).

2.4.1 Keys and cryptographic primitives

Every platform uses *libsodium*, a widely reviewed cryptographic library: through `sodiumoxide` on desktop, `LazySodium` on Android, and directly on iOS. Pycter implements no cryptographic primitive of its own. Exact parameters are in Appendix A.

Album key. 256 random bits, generated by the device that creates the album. It never changes, and it is never sent to the server in clear. Two sub-keys are derived from it with libsodium's key derivation (BLAKE2b): the *update key* and the *wrapping key*, described below.

Update key. It encrypts each update. Devices rebuild the album by applying its updates in order.

File key. Each file is encrypted with a key computed from its own content: the BLAKE2b-256 hash of the file. Each file is stored in pieces; each piece is encrypted with XChaCha20-Poly1305

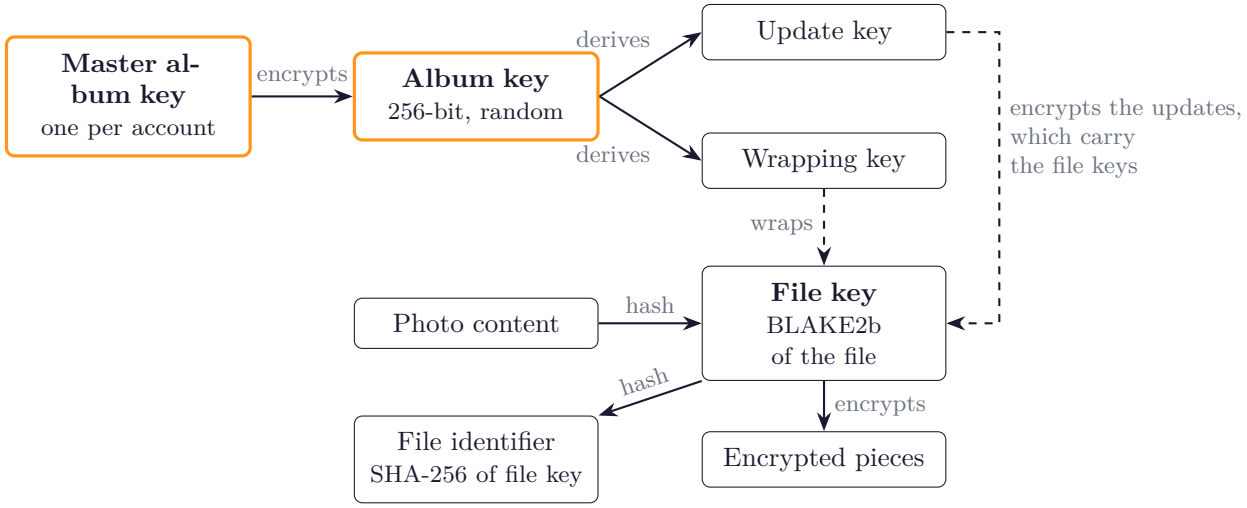

Figure 3: Key hierarchy. Orange keys never leave the members' devices unencrypted.

(libsodium's *secretstream*) under that key, with a fresh random header each time, and every 1 MiB block carries its own authentication tag: any change to a stored piece is detected on decryption. Devices learn a file's key from the album updates: it travels inside the encrypted update that adds the file.

File identifier. The server refers to a file by the first 128 bits of the SHA-256 hash of its file key. It reveals nothing about the content, but identical files share it (Section 2.5.1).

Wrapped file key. For each file in each album, the server also stores the file key XOR-ed with a keystream (XSalsa20) generated from the album's wrapping key, with the file identifier as nonce. It lets a device that holds the album key, but has not applied the album's updates, obtain a file's key from the server. Without the album key, this value is indistinguishable from random bytes: it is a one-time pad over the file key.

Master album key. Each account has a private *master album* whose updates list the albums the account joined, each with its album key encrypted under the master album key. This is how an account's devices learn the keys of its albums (Section 2.4.2). It is 256 random bits too.

2.4.2 Key distribution

Invitation links. An invitation link looks like `https://pycter.com/i#token`. The part after `#`, the *anchor*, is a 128-bit random token. Browsers never send the anchor to any server; if the link opens in a browser, the page's script reads it locally to hand it to the app. One exception: on an Android phone without Pycter, the page sends the visitor to Google Play with the token in the store link, so that the app can join the album as soon as it is installed. Google then sees the token.

From the token, the inviting device derives two values (libsodium key derivation over SHA-512 of the token):

- an *identifier*, under which the server stores the invitation;
- a *key*, which encrypts the invitation's content (XSalsa20-Poly1305): the album's name, its key and its cover's file key.

The server stores the identifier and the ciphertext only. A joining device derives the same two values, fetches the invitation by its identifier, and decrypts it locally.

The link is a password. Whoever holds it can join the album, for up to 15 days. Invitations are deleted within 15 days, and all of an album's invitations are deleted as soon as a member is

removed from it (Section 2.3). The anchor is hidden from Pycter; how far the link travels depends on the channel.

Via a QR code. The inviting phone shows the link as a QR code, which the new member scans in person. The link never passes through a messaging service or an email provider, so this is the safest channel. Only the Android exception above remains.

Via a link. The link can be sent through any app. That app sees the whole link: a messaging service or an email provider could use it to join the album within 15 days. On Android, Google Play also sees the token.

By email address: not available. Inviting someone by email address, or by picking a colleague from another album, lets the server add the account directly. The server cannot hand over the album key, so it refuses these invitations for encrypted albums.

The account's own devices. When an account joins an album, its device adds an update to the account's master album: the album's identifier, and the album key encrypted under the master album key. The account's other devices apply that update and learn the key. The server stores the update but cannot open it.

Pairing the company computer. The desktop app joins an account by pairing with a phone:

1. The desktop generates a key pair (X25519) and shows a QR code holding a channel number and its public key.
2. The phone scans the code and seals the master album key and the account's recovery key to that public key (libsodium *sealed box*).
3. The server relays the sealed box; the desktop opens it locally.

The public key travels from screen to camera, never through the server, so the server cannot substitute its own. Channels are single-use and expire after 3 minutes.

A new phone. The master album key and the recovery key are stored in the phone platform's keychain: iCloud Keychain on iPhone, Google Block Store on Android. Both synchronise to the user's other phones end-to-end encrypted (Block Store does so when the phone has a screen lock). On a new phone, the recovery key gets the account back, and the master album key comes from the keychain. The server stores only the SHA-256 hash of the recovery key.

An account password. In the app's settings, a user can turn on end-to-end encryption for their own account, which then also covers their personal albums. This asks for a password. The device derives a key from it with Argon2id (Appendix A) and encrypts the master album key under that key; the server stores the result with the account, so that the user's new devices get the master album key back from the password. Pycter's servers, and anyone who signs in to the account, can try passwords against it offline. A password found gives the master album key, and with it the key of every album the account joined, organisation albums included. Choosing a strong password is the user's responsibility. Without an account password, the servers hold no copy of the master album key.

2.5 Data on Pycter's servers

This subsection is the single list of what Pycter's servers store, encrypted or in clear. The summary table on the first page is drawn from it.

Stored	**Form**
Content and keys, encrypted: unreadable without the album key	
Photos and videos, at every size	Pieces, XChaCha20-Poly1305 under the file key

Stored	Form
Each file's record: file key, original file name, capture time, location	Inside the update that adds the file, under the update key
Album title, photo descriptions, album status, member nicknames	Updates, under the update key
Album key	Never stored in clear: encrypted under the master album key, or inside an invitation (Section 2.4.2)
Master album key, if the user set an account password	Encrypted under a key derived from the password (Section 2.4.2)
File keys, a second time	Wrapped under the wrapping key (Section 2.4.1)
Metadata, in clear	
Accounts	Email address, name, recovery key hash, device token hashes
Devices	Label, last seen, push token, which files each device holds, which account last asked it to upload a file
Membership	Who belongs to which album; organisation, its name and roles
Updates	For each update: its kind (e.g. "photo added", "video added"), author, time, and the identifiers of the files it concerns
Files	Identifier, number and size of pieces per size, capture time (to order the album)
Album settings	Eco mode, archived flag, maximum radius
Request logs	IP address, path and status code, 30 days (Section 2.6)

2.5.1 Identical files

Since a file key is computed from the file's content, the same file gets the same key and the same identifier in every album. Two consequences:

- The server can tell that two albums hold the same file, without seeing the file. It never tells other users.
- A file is only as protected as its least protected copy. If the very same file is also placed in an album that is not end-to-end encrypted, its key can be read from there (Section 2.8).

The trade-off is deliberate: it lets Pycter store and transfer a file once, however many albums hold it.

2.6 Infrastructure and operations

This subsection describes where the servers run, how long data is kept, and what is logged and backed up.

Regions. The API runs on Google Cloud Run and the database on Google Cloud SQL (PostgreSQL), both in `europe-west1` (Belgium). File storage is Cloudflare R2, in its Western Europe location.

Transport. All traffic to the API and to file storage uses HTTPS (TLS).

Retention.

- Screen, full and original sizes: deleted 2 days after upload.
- Thumbnails: 180 days.

- Invitations: at most 15 days, or until a member is removed.
- Album updates: kept while the album has members. An album without members is deleted, with all its records, by a daily job.

Logs. Request logs record the IP address, path and status code of each request, and are kept 30 days in Google Cloud Logging. They contain no request bodies, no content and no keys.

Backups. Photo files are not backed up.

2.7 Devices

This subsection describes what Pycter keeps on devices and where photos go once decrypted.

Keys live on devices. Album keys are kept in the app's database, on the phone or computer. Device tokens are kept on the device (Section 2.3). On the company computer, the app's database sits in the user's profile, protected by the platform like the user's other files (Section 2.1).

The company's folders. The desktop app decrypts photos into the folder layout the company chose. From there, the company's own access controls apply.

The in-app camera. Photos taken with Pycter's camera go to the app's private storage, not to the phone's gallery, so the gallery's cloud sync (iCloud Photos, Google Photos) never sees them. Photos picked from the gallery were already in the gallery before Pycter, and stay there.

Phone backups. Like every app, Pycter keeps its data in the phone's app-private storage. The phone's own backup (iCloud Backup, Android Backup) copies it along with the rest of the phone, under the same protection as the user's photos and messages. Pycter adds no party beyond the phone's platform provider, which the company already trusts with the phone as a whole. A company that manages its phones can turn these backups off for Pycter.

On-device analysis. Face grouping and text recognition run on the phone. Their results never leave it.

2.8 Known limitations

This subsection lists what the technical protection does not cover today.

- **Only organisation albums.** An employee's personal albums are not end-to-end encrypted (unless they set up account-wide encryption). A work photo placed in a personal album is not protected there, and, since identical files share their key (Section 2.5.1), neither is its copy in the organisation album.
- **Metadata in clear.** The metadata listed in Section 2.5 is readable on Pycter's servers.
- **Removal.** Removal takes away membership, not what the person already received (Section 2.3).
- **Invitation links** work like a password for up to 15 days. Whoever carries a link sees it, and so does Google Play when an Android phone installs the app from one (Section 2.4.2).
- **Server breach.** Someone in control of Pycter's servers could keep encryption from being turned on, but not turn it off (Section 2.1).

3 Legal protection

This section sets out what Pycter commits to doing, and not doing, with the content and the metadata of Section 2.5.

Roles. The company is the data controller. Pycter processes data on its behalf, as processor under Article 28 of the GDPR.

- **Only to provide the service.** Pycter uses content and metadata only to run the service: delivering albums, signing users in, and keeping the service working.
- **No sale.** Pycter never sells or rents content or metadata.
- **No advertising or profiling.** Pycter shows no ads and builds no profile of users.
- **No AI training.** Neither content nor metadata trains any model, Pycter's or a third party's. Face grouping and text recognition run on the phone (Section 2.7).
- **Subprocessors.** Pycter shares metadata only with the subprocessors listed below, for their role there, and informs the company of any change.
- **Breach notification.** Pycter informs the company without undue delay of any breach affecting its data, so the company can meet its own 72-hour obligation.
- **Deletion.** When the company stops using Pycter, Pycter deletes the company's content and metadata within 90 days.

Subprocessors.

Provider	Role	Data
Google Cloud	API, database, logs	Everything in Section 2.5; content as ciphertext
Cloudflare	File storage (R2)	Encrypted files
Google Firebase, Apple Push Notification service	Push wake-ups, app attestation	Device push tokens. A push carries only "refresh", or the id of the account asking for a file
Google Firebase Crashlytics	Crash reports from the phone apps	Stack trace, device model, OS and app version, keyed to a random install identifier, not to the account
Brevo	Login-code emails	Email address, one-time code

4 Organisational protection

This section covers the protection that depends on people: who may change an album, what to do when a key leaks, and who may access Pycter's production systems.

4.1 At the company

This subsection describes the organisation's rules and roles, and the procedure when a key leaks.

The organisation album. The organisation is marked as encrypted on the server. An album belongs to it only if it is encrypted, and inside the organisation encryption cannot be changed (Section 2.3).

Roles. In an organisation album, the server checks each new update against its author's role. Roles protect the album against mistakes: a new colleague cannot delete or edit its photos.

- *Admins* manage the album: its title, cover and status, archiving, deleting or editing photos, and removing members.
- *Members* add photos, videos and descriptions, and set their nickname. They cannot delete or edit photos.

- *Externals*, people outside the organisation who joined through an invitation, set their nickname only. They still see the whole album: inviting someone gives them read access.

4.1.1 Compromised key

An album key can leak through a former member, who keeps it after removal, or through a compromised device. Album keys are never rotated: an album keeps the key it was created with.

Remove the person from the album. The server then refuses them all updates and files (Section 2.3). The first layer stops them: the key alone gives them nothing new.

Start a new album for a new key. Create a new album and invite the whole team again. The new album gets a fresh key, which the former member or the compromised device never receives. Content in the old album stays under the old key.

4.2 At Pycter

This subsection describes who at Pycter can reach production.

Access. Only Pycter administrators have access to production. Service credentials are kept in Google Secret Manager.

5 Auditing Pycter

This section lists the checks and reviews through which the company can verify how its content is protected.

- **Source access.** On request, Pycter can arrange a review of the source code of the apps and the server, by the company's IT team or an auditor of its choice.
- **Parameters.** Appendix A lists every primitive and parameter, enough to check the source against this document.
- **The anchor stays local.** Open an invitation link in a desktop browser with its developer tools on: the request to `pycter.com` carries no anchor.
- **Who has access.** The apps list an album's members: every account the server lets fetch its content.
- **The album's state.** The phone apps show, in the album settings, that the album is end-to-end encrypted and since when.

A Cryptographic parameters

This appendix lists every primitive and parameter in use.

Item	Construction (libsodium)
Album key, master album key	32 bytes from the system CSPRNG (`crypto_kdf_keygen` / `randombytes_buf`)
Update key	`crypto_kdf_derive_from_key`, subkey id 2, context `"journal\0"`
Wrapping key	`crypto_kdf_derive_from_key`, subkey id 1, context `"blobxor\0"`
Update	Version byte `0x31` followed by `crypto_secretstream_xchacha20poly1305`, 1 MiB blocks, last block tagged `FINAL`
File key	BLAKE2b, 32-byte digest, of the file's bytes
File piece	`crypto_secretstream_xchacha20poly1305` under the file key, 1 MiB blocks
File identifier	First 16 bytes of SHA-256(file key)
Wrapped file key	`crypto_stream_xor` (XSalsa20), wrapping key, nonce = file identifier zero-padded to 24 bytes
Invitation token	16 random bytes, URL-safe base64 in the link's anchor
Invitation identifier	`crypto_kdf_derive_from_key` over SHA-512(token)[0..32], context `"invStora"`, hex
Invitation content	`crypto_secretbox` (XSalsa20-Poly1305) under a key derived the same way, context `"invPaylo"`
Desktop pairing	`crypto_box_seal` (X25519, XSalsa20-Poly1305) to the desktop's ephemeral public key
Recovery key	32 random bytes; the server stores SHA-256
Device token	32 random bytes; the server stores SHA-256
Account password (optional)	Argon2id13, `OPSLIMIT_INTERACTIVE`, `MEMLIMIT_INTERACTIVE`, 16-byte random salt; `crypto_secretbox` of the master album key

B Network requirements

This appendix lists the hosts that devices on the company network need to reach.

Host	Purpose
`pycter.com`, `api.pycter.com`	The Pycter service
`fd025f3f9d41dbacb3eb3606a952a737.r2.cloudflarestorage.com`	Photo transfer (Cloudflare R2)
`pub-8ca876afb44e45789633034606b43f7c.r2.dev`	Desktop app updates

Phones also use their platform's push service (Apple Push Notification service, Firebase Cloud Messaging) and Firebase App Check. These are usually already allowed on company networks.