layout: layouts/article.html title: "SimpleX File Transfer Protocol - a new protocol for sending large files efficiently, privately and securely." date: 2023-03-01 preview: CLI and relays implementing the new XFTP protocol are released - you can use them now! image: images/20230301-xftp.jpg imageWide: true

permalink: "/blog/20230301-simplex-file-transfer-protocol.html"

SimpleX File Transfer Protocol – a new protocol for sending large files efficiently, privately and securely.

Published: Mar 1, 2023

⚡ Quick start: send a file with XFTP CLI in 3 simple steps

Download XFTP binary for Linux from the release – you need the file `xftp-ubuntu-20_04-x86-64` - rename it as `xftp`.

Step 1: To send the file:

```
bash xftp send filename.ext
```

You can also send the file that can be received by multiple recipients using `-n` option:

```
bash xftp send filename.ext -n 10
```

Step 2: Pass file description(s) (files `rcvN.xftp`) to the recipient(s) securely, e.g. send it as a file via SimpleX Chat.

Step 3: To receive the file:

```
bash xftp recv rcvN.xftp
```

The sender also delete all file chunks from the relays before they expire in 48 hours with this command:

```
bash xftp del ./filename.ext/snd.xftp.private
```

What's the problem?

If you are using SimpleX Chat apps you know that support of sending files and images is not very good, and sending videos and large files is simply impossible. There are currently these problems:

- the sender has to be online for file transfer to complete, once it was confirmed by the recipient.
- when the file is sent to the group, the sender will have to transfer it separately to each member, creating a lot of traffic.
- the file transfer is slow, as it is sent in small chunks - approximately 16kb per message.

As a result, we limited the supported size of files in the app to 8mb. Even for supported files, it is quite inefficient for sending any files to large groups.

Why didn't we just use some existing solution?

We really hoped to find some existing open-source solution that we could integrate with SimpleX Chat.

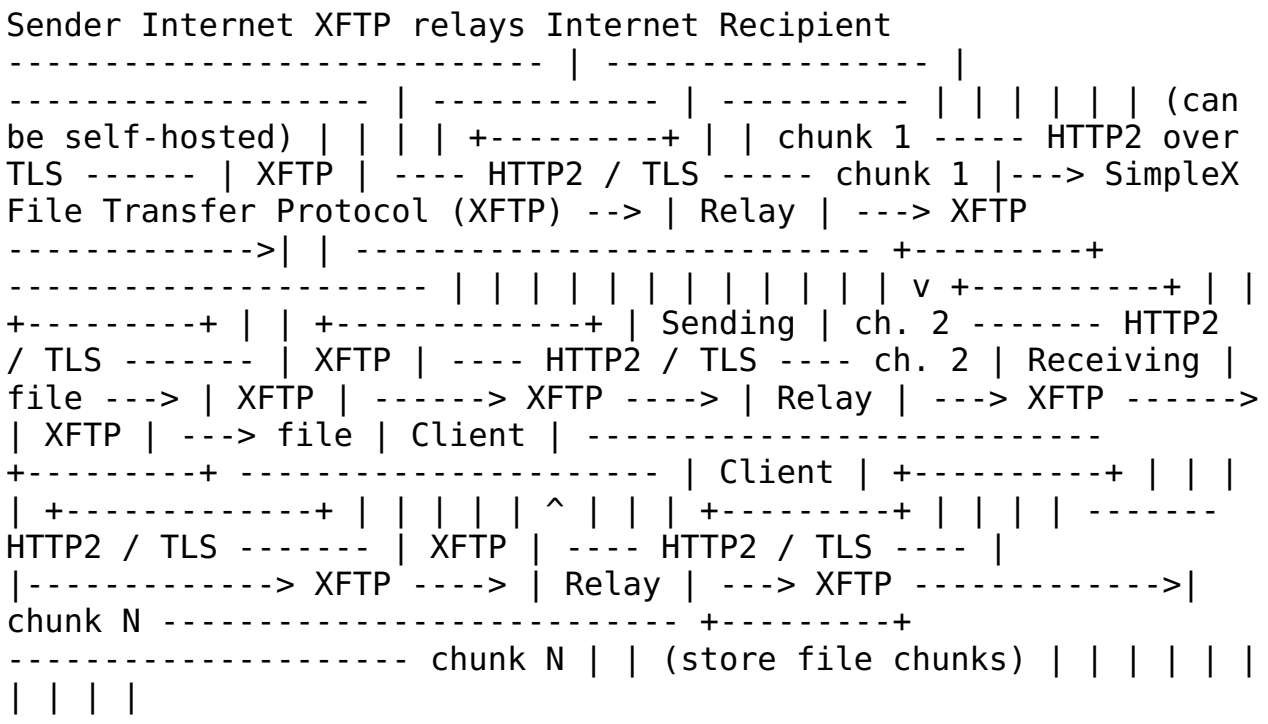
We decided not to use torrent-like or any other P2P solutions because of their lack of privacy, challenging legality in some jurisdictions and, in many cases, because they are inefficient in groups.

We reviewed several S3-compatible solutions (e.g., minio, garage, SeaweedFS), but they all require the development of a separate service layer, making them unusable as standalone services and harder to deploy for the users who want to self-host the file transfer service. As a side note, the solution that we developed can still be composed with S3-compatible storage for higher capacity servers with some privacy/efficiency trade-offs.

We also looked at a few independent implementations of file sharing, with some ad-hoc protocols (e.g., ceph and lufi), but neither seemed sufficiently mature, and also not as private as we would like.

So after a lot of searching we decided to design and implement a new protocol for file transfers, that both solved the problems above, and provided a higher level of metadata privacy than any other file transfer solution has.

What is XFTP and how does it work?



XFTP stands for SimpleX File Transfer Protocol. Its design is based on the same ideas and has some of the qualities of SimpleX Messaging Protocol, that is used in SimpleX Chat:

- recipient cannot see sender's IP address, as the file fragments (chunks) are temporarily stored on multiple XFTP relays.
- file can be sent asynchronously, without requiring the sender to be online for file to be received.
- there is no network of peers that can observe this transfer - sender chooses which XFTP relays to use, and can self-host their own.
- XFTP relays do not have any file metadata - they only see individual chunks, with access to each chunk authorized with anonymous credentials (using Edwards curve cryptographic signature) that are random per chunk.
- chunks have one of the sizes allowed by the servers - currently we allow 256kb, 1mb and 4mb chunks, so if you send, say 1gb file, to XFTP relays it will look indistinguishable from sending many small files, and they would only know that chunks are sent by the same user only via the transport information, but none of the relays will see all chunks. Also, once this feature is available in mobile apps you can use transport isolation per chunk, when each file fragment will be uploaded via a separate TCP connection (and Tor circuit, if you use Tor) – the CLI we released does not yet support per-chunk transport isolation.
- each chunk can be downloaded by multiple recipients, but each recipient uses their own key and chunk ID to authorize access, and the chunk is encrypted by a different key agreed via ephemeral DH keys (NaCl crypto_box (SalsaX20Poly1305 authenticated encryption scheme) with shared secret derived from Curve25519 key exchange) on the way from the server to each recipient. XFTP protocol as a result has the

same quality as SMP protocol - there are no identifiers and ciphertext in common between sent and received traffic inside TLS connection, so even if TLS is compromised, it complicates traffic correlation attacks.
- XFTP protocol also supports redundancy - each file chunk can be sent via multiple relays, and the recipient can choose the one that is available. The released CLI does not support redundancy though.
- the file as a whole is encrypted with a random symmetric key using NaCl secret_box.

So, how would any recipient know where to get all these file fragments from and how to put them back together into the original file? Normally, when you send a file via any file-sharing service it provides you a link that you can pass to the recipient. The link allows to download the original file, but it also provides the server a lot of file meta-data, that often includes file name and exact size, and in many cases the server also has access to a file content.

Instead of using a link, XFTP protocol includes a special format for a "file description" - it is a small text file containing the locations, access keys and digests for all file chunks, and also the encryption key and digest (SHA512) for the whole file. This file description does not contain the original file name or exact file size, so if it is used after the file fragments are expired or removed from XFTP relays, this information is not accessible.

CLI generates a separate file description for each intended recipient - you need to specify how many people you want to be able to receive this file. You can specify a larger number of recipients to avoid revealing the real number of recipients from XFTP relays. Mobile apps, when this protocol is integrated, will do it automatically, choosing some large random number of possible recipients, so while the relays will be able to observe how many people time the file was downloaded, they won't know how many intended recipients you had - sending to a group of 10 people and to 1 recipient can look the same to the relays.

File description is a security-sensitive file that contains private keys and chunk addresses necessary to receive the whole file, and also a symmetric key to decrypt the file. Therefore you must use a secure channel to send file description - e.g., it can be sent via SimpleX Chat. But once the recipient downloaded the file, CLI invalidates the file fragment addresses on relays and the same file description cannot be used again to download the file.

What is next?

We released and deployed several XFTP relays for you to experiment with (they are hardcoded in the XFTP CLI), and you can deploy your own relays either from downloadable binary or by compiling the source code. We also released XFTP CLI - it is available in the same release.

We are currently integrating support for sending large files using XFTP protocol into SimpleX Chat clients. SimpleX Chat v5.0 will have support for receiving files sent via XFTP protocol (you will be able to send a file description via a SimpleX Chat CLI app, so that mobile apps will be able to

receive them as normal files, only much faster), and v5.1 will fully support for sending large files (up to 1gb) in the mobile apps.

We will also publish a formal specification for XFTP protocol and overview of its security qualities and threat model. For now you can learn more about the protocol design and motivations from this internal XFTP protocol RFC.

Using and sending files with the available XFTP CLI will hugely help us stabilizing both the protocol and implementations. What we really like about this design is that it is completely independent from SimpleX Chat - you can use it on its own, sending files and passing file descriptions to your contacts via any other messenger - e.g. via Signal, – without this messenger being able to observe that you are in fact sending a large file.

We did not yet decide whether we will be making a separate security audit of XFTP implementation, or if we combine it with the next security audit of SimpleX Chat. The latter seems more likely, as XFTP uses the same cryptographic primitives that were reviewed during SimpleX Chat security assessment by Trail of Bits in November 2022.

SimpleX platform

Some links to answer the most common questions:

How can SimpleX deliver messages without user identifiers.

What are the risks to have identifiers assigned to the users.

Technical details and limitations.

How SimpleX is different from Session, Matrix, Signal, etc..

Please also see our website.

Help us with donations

Huge thank you to everybody who donated to SimpleX Chat!

We are prioritizing users privacy and security - it would be impossible without your support.

Our pledge to our users is that SimpleX protocols are and will remain open, and in public domain, - so anybody can build the future implementations of the clients and the servers. We are building SimpleX platform based on the same principles as email and web, but much more private and secure.

Your donations help us raise more funds – any amount, even the price of the cup of coffee, makes a big difference for us.

See this section for the ways to donate.

Thank you,

Evgeny

SimpleX Chat founder