Sharekey®

Security Paper

SHAREKEY Swiss AG

April 2021

Sharekey®

All-in-One App

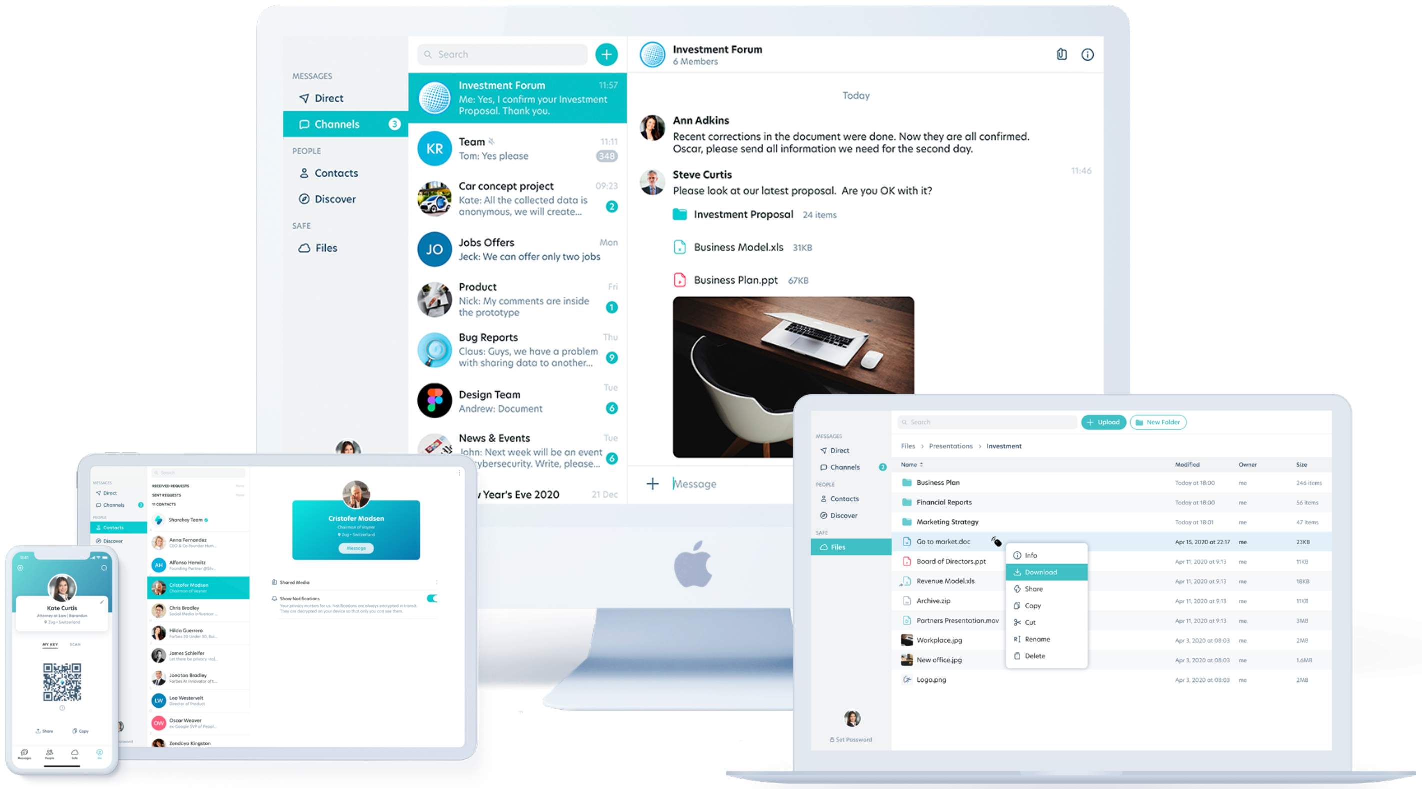
Easy to Use
Single Account
Multi Device
Sync History

App-to-App Encryption

All Data Encrypted
Auditable Source Code
256-Bit Crypto Engine
Sharekey Blind to All Data

Swiss Privacy

Swiss Cloud Storage
Strict Log Policy
EU GDPR Compliant
Swiss Privacy Laws



Investment Forum 11:57
Me: Yes, I confirm your Investment Proposal. Thank you.

Team 11:11
Tom: Yes please

Car concept project 09:23
Kate: All the collected data is anonymous, we will create...

Jobs Offers Mon
Jeck: We can offer only two jobs

Product Fri
Nick: My comments are inside the prototype

Bug Reports Thu
Claus: Guys, we have a problem with sharing data to another...

Design Team Tue
Andrew: Document

News & Events Tue
John: Next week will be an event cybersecurity. Write, please...

Year's Eve 2020 21 Dec

Investment Forum
6 Members

Today

Ann Adkins
Recent corrections in the document were done. Now they are all confirmed. Oscar, please send all information we need for the second day.

Steve Curtis 11:46
Please look at our latest proposal. Are you OK with it?
Investment Proposal 24 items
Business Model.xls 31KB
Business Plan.ppt 67KB

Files › Presentations › Investment

Name	Modified	Owner	Size
Business Plan	Today at 18:00	me	246 items
Financial Reports	Today at 18:00	me	56 items
Marketing Strategy	Today at 18:01	me	47 items
Go to market.doc	Apr 15, 2020 at 22:17	me	23KB
Board of Directors.ppt	Apr 11, 2020 at 9:13	me	11KB
Revenue Model.xls	Apr 11, 2020 at 9:13	me	18KB
Archive.zip	Apr 11, 2020 at 9:13	me	11KB
Partners Presentation.mov	Apr 11, 2020 at 9:13	me	2MB
Workplace.jpg	Apr 3, 2020 at 08:03	me	2MB
New office.jpg	Apr 3, 2020 at 08:03	me	1.6MB
Logo.png	Apr 3, 2020 at 08:03	me	2MB

macOS

Contents

Contents 3
Executive Summary 4
Key Differentiators 5
Overview 6
Security Design 9
- Identity 9
- Verification 9
- History 10
- Metadata 10

Security Culture 11
- Our People 11
- Our Planet 11

Key Management 12
- Principal Keys Overview 12
- Secret Phrase Generation (24 Secret Words) 12
- Key Generation 12
  - Master Secret Key 12
  - Master Signing Key 13
  - Random Key 13
- Key Derivation 13
- Session Authorisation Token 14

User Account & Device Management 16
- Account Creation (Sign Up) 16
  - User Verification 16
- Secret Phrase Generation 17
- Device Setup (Sign In) 18
- Authentication (Unlock App) 19

Connecting Users 20
- Discover 20
- Key Sharing 21
- Channel Members 21

Messaging 22
- Direct Messaging 22
- Channel Messaging 23

Storage 25
- Data Sharing 26
  - Folder Sharing 26
  - File Sharing 26
- Rights Management 27
- Local File Storage 28

Data Encryption 29
- Crypto Layer in App 29
  - Data Encryption & Integrity 30
    - XSalsa20 30
    - Poly1305 30
  - Digital Signatures 30
    - Ed25519 30
- Crypto Layer Output 31
  - View Encrypted Data 31
  - Signed Encrypted Data 31
- Transport Layer Security 32

Notifications 33
Infrastructure 34
- Data Hosting Infrastructure 34
- Kubernetes Cluster 35
  - Load Balancer 35
  - File Uploading Microservice 35
  - Monolith 35
  - Database 35
  - Backup Tool 35
  - Persistent Cluster 35
- User Data Processing 36

Testing & Certification 38
- Code Audits 38
- Penetration Tests 38
- Bounty Programs 38
- Certifications 38

Source Code 39
Website 40
Roadmap 41
- Expanded Functionality 41
- Metadata 41
- Increased Security 42
- Sharing Control 42
- Infrastructure & Data Storage 43

Conclusion 44
References 45
Acronyms 46
Glossary 47

Executive Summary

Sharekey is an All-in-One business collaboration app using App-to-App Encryption with decentralised keys for true privacy. It is expected to be particularly useful for Corporate Executives, Board Members, Legal and Financial staff to share confidential information, both within and across enterprise boundaries. Of course, the solution is also designed to be employed as a general purpose, secure corporate communication and collaboration tool.

Sharekey offers a privacy-focused alternative to traditional collaboration environments. Unlike other collaboration systems, where personal and enterprise data is visible to the service provider, all data stored on the Sharekey platform is encrypted using keys controlled by the end users and securely retained on their devices.

This level of privacy is essential at this time of frequent enterprise data breaches and growing regulatory compliance requirements. Sharekey offers an alternative for organisations concerned about the security of their enterprise data and the privacy of corporate communications. It can also be useful to those enterprises unable to use other solutions due to regulatory compliance concerns. For example, certain enterprises might be legally required to retain and decrypt historical messages, which might not be possible on other platforms but is a feature of Sharekey.

Currently, Sharekey offers support for individual and group communications, and file and folder storage and sharing. However, the Sharekey Roadmap includes adding support for voice and video conferencing and productivity-focused tools such as a calendar and project-based collaboration system. All of these systems will be built on Sharekey's end-to-end encrypted platform where users and enterprises own and have full control over their encryption keys.

The Sharekey platform includes the following key security features:

- All data, including privacy-sensitive metadata, is end-to-end encrypted;
- Users are the only ones with access to their encryption keys;
- Access controls on messages and shared data are cryptographically enforced;
- Users are identified via their corporate emails (allowing corporate policies to dictate access);
- User accounts are secure by default (with local files encrypted and password or passcode required for sign-in to the app);
- Backend servers in Switzerland used by Sharekey hold all relevant industry certifications (ISO27001, SOC2, PCI DSS, etc.).

The remainder of this document provides a description of the key features of Sharekey that differentiate it from similar (typically consumer) solutions, as well as a deep dive into the technical aspects of the security model that are at the heart of the solution.

Key Differentiators

Sharekey is a collaboration platform where all data is encrypted, and only Sharekey users have access to their encryption keys. This makes Sharekey immune to many of the cybersecurity risks that threaten other collaboration applications, such as:

- **Password Leaks:** Sharekey users' credentials are managed using Secret Keys that are stored only locally on their devices. If attackers compromise Sharekey's servers, there is no password hash database to be leaked.
- **Data Leaks:** All data stored on Sharekey's servers are encrypted with a key that never leaves the user device. An attacker with access to Sharekey's servers does not have the ability to access or exfiltrate any readable user data.
- **Social Engineering:** Sharekey employees have no access to user data or accounts because all data is encrypted with user-controlled keys. This makes social engineering attacks against Sharekey employees ineffective because they lack the ability to access Sharekey user data. The Sharekey Roadmap also includes the implementation of further protections against Man-in-the-Middle (MitM) attacks to eliminate the risk of attacks during the connection creation step on the server side.
- **Court-Ordered Disclosure:** Social media and collaboration platforms with access to a user's Secret Keys may be compelled by a court order to turn over user data. Sharekey can't decrypt user data, making it immune to such orders. Moreover, SHAREKEY Swiss AG being a Swiss company with its servers hosted in Switzerland, all user data is protected by the Swiss Federal Data Protection Act (DPA) and the Swiss Federal Data Protection Ordinance (DPO). These offer some of the strongest privacy protection in the world for both individuals and corporations. As Sharekey is outside of US and EU jurisdiction, only a court order from the Cantonal Court of Zug or the Swiss Federal Supreme Court can compel us to release the extremely limited user information we have.
- **Metadata:** Sharekey additionally minimises the collection of user metadata, as described in our Privacy Policy and Roadmap. Sharekey also does not employ any user behavior analysis tool or share any data with third parties.
- **Physical Device Access:** All data stored on a user's device is encrypted, with the exception of downloaded files opened in another application. This means that an attacker with physical access can only access data if they know the user's Password or Secret Phrase. As no reliable and superior authentication solutions exist to mitigate such attacks, this is not an attack scenario that Sharekey can effectively protect against, and attempting to do so could result in weaker security, as demonstrated by successful attacks against multiple versions of FaceID and TouchID.

Without encrypting all user data with user-controlled keys, it would be impossible for Sharekey to guarantee protection against all of these potential attack vectors. It is this decision to leave full control of their data in the user's hands that differentiates Sharekey from other collaboration and social media platforms.

Overview

Sharekey is a business-to-business collaboration platform with built-in App-to-App Encryption support for all data exchanges. App-to-App Encryption, as depicted pictorially in Figure 1, differs from the other types of encryption usually described as end-to-end. In many end-to-end scenarios, such as secure email or other consumer-oriented communication apps, some parts of the exchanged information, typically the metadata, is sent in the clear and read by intermediaries. But not so with Sharekey. All data is encrypted end-to-end, with only minimal metadata visible to Sharekey in order to offer its services — and even that limited metadata does not compromise user privacy in any way.

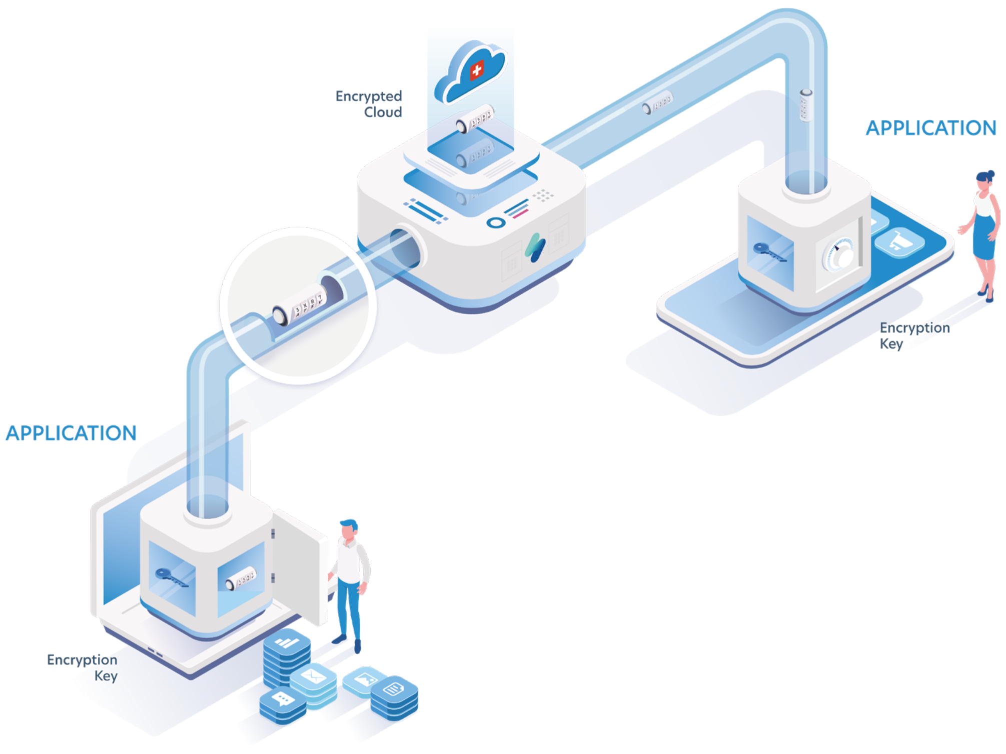


Figure 1 • App-to-App Encryption

Users on their Sharekey apps connect to one another by sharing their Public Keys while securely maintaining their private Master Secret Key. Sharekey gets its name from the fact that the connection is encrypted with a Shared Key derived from these Public Keys and private Master Secret Keys, as described later in the document. All data stored and shared via the Sharekey platform is encrypted end-to-end. Sharekey users have complete control over their private Master Secret Keys, meaning that none of their data is readable by Sharekey, but only by those they choose to share it with.

All Sharekey-supported data stored on a user's device is encrypted, again with the user's Master Secret Key, with the exception of downloaded files opened in another application. This means that an attacker with physical access to a device can only access Sharekey-enabled data if they know the user's Password or Secret Phrase. These are explained more fully in later sections of this document.
Figure 2 shows a high-level view of the Sharekey platform.

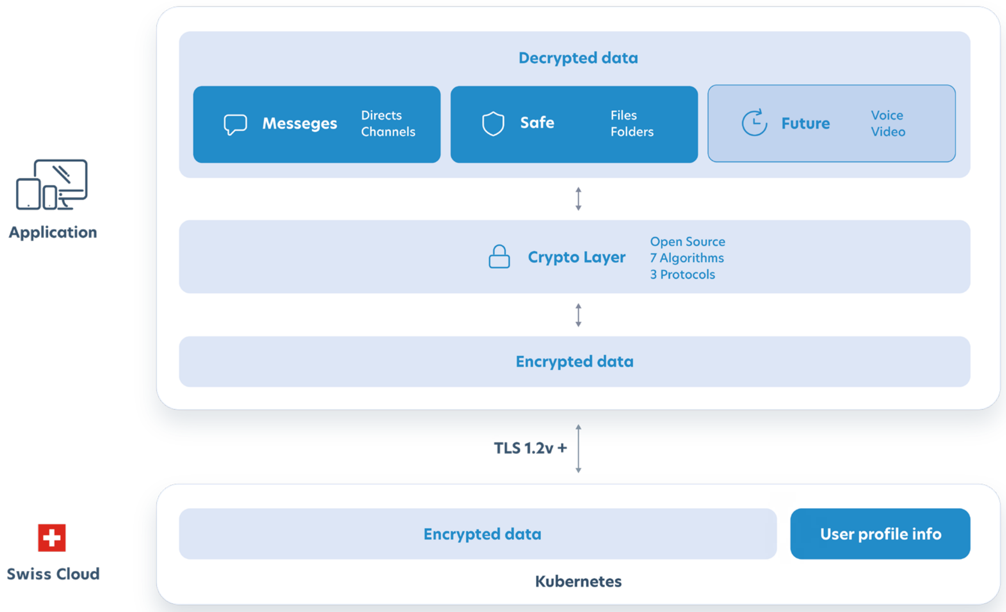


Figure 2 • Sharekey Platform Overview

Sharekey users exchange encrypted messages, files and folders using one-on-one Direct Messaging or one-to-many Channels.

Sharekey is built around an integrated Crypto Layer as shown at the center of Figure 2. This Crypto Layer implements standardised data encryption, authentication, and integrity verification for all Sharekey functions. This makes it possible for Sharekey to implement a secure and privacy-focused collaboration platform, including encrypted cloud-based file storage, in a way that is both secure and easy to use.

A complete history of a user's communications and stored files and folders is stored encrypted (with the user's Master Secret Key) on Sharekey's servers, hosted by a Swiss cloud provider. Sharekey only has access to encrypted data and holds this encrypted data solely for the purpose of restoring a user's data on a new device.

This remainder of this Security Paper walks through every aspect of Sharekey's security solution and deployment infrastructure and outlines key decisions made to provide a secure, private environment for business-to-business communications. The document includes everything from initial account setup and key creation to the various ways to communicate and use the platform leading to the nuts and bolts of the technical solution and architecture.

The technical architecture of our solution, as shown in Figure 3, will be described throughout the rest of the document.

Briefly, it describes our assembly of various open-source cryptographic algorithms and protocols to achieve our goal of app-to-app encryption. All messages and data stored and shared via Sharekey are encrypted. Data is encrypted locally and uploaded to Sharekey servers in their encrypted form.

The Crypto Layer — which is at the heart of our solution — contains our implementation of what we consider true App-to-App encryption. We believe that our solution is a true disruption in an area where this level of privacy support is usually lacking.

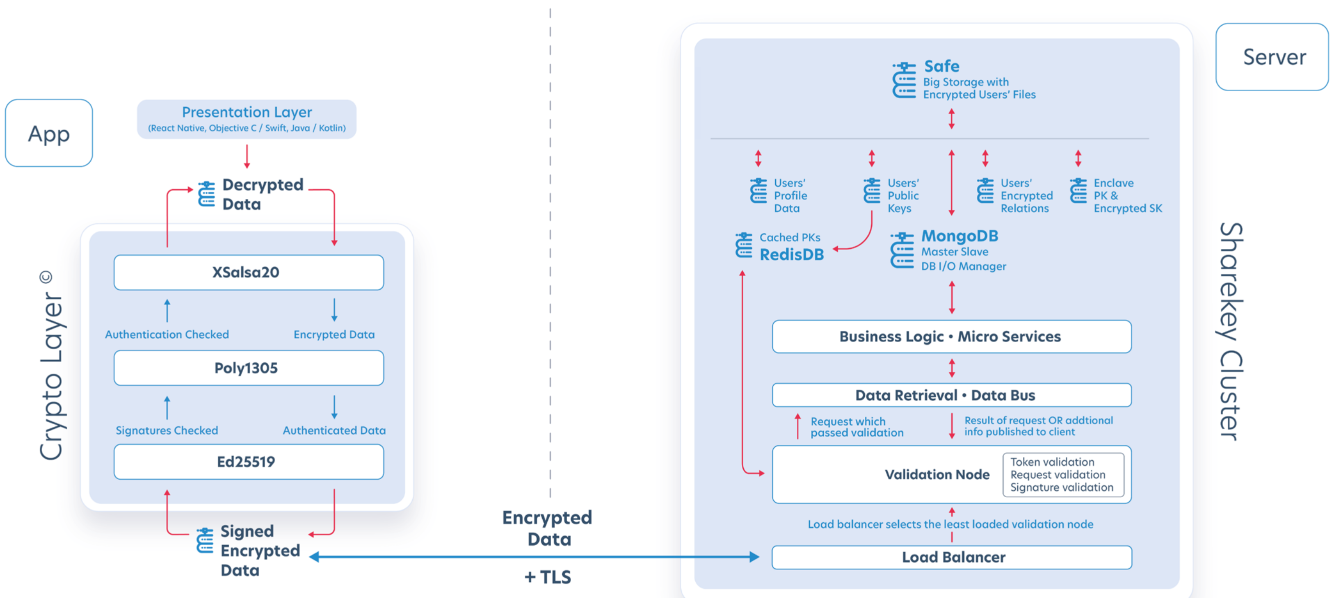


Figure 3 • Sharekey Technical Architecture

Security Design

Several design decisions have guided Sharekey's security controls and measures. Sharekey has chosen a security model that provides both end-to-end data protection as well as user-friendly functionality. These usability features are motivated by the feedback received from users and the identified needs of Sharekey's target users, as well as our analysis of the market.

The key decisions and our rationale are described in this section.

Identity

As a business collaboration app, Sharekey has chosen corporate email as the user identifier as there is no need for anonymous users in a corporate environment. Moreover, enterprise management has full control of these identities and once an employee leaves an organisation, they no longer have access to the email linked to their Sharekey account. Ex-employees cannot log into their Sharekey account, nor add new devices to the account. This greatly limits the potential for leakage or inappropriate use of company-owned data stored within the Sharekey app.

Future plans include additional security features such as the ability to have corporate IT management log out an ex-employee from all their current devices and wipe out Sharekey-encrypted files stored locally on these devices.

Using corporate identifiers also allows integration with enterprise single sign-on (SSO) solutions.

Verification

We have chosen account verification via email as a privacy feature. SMS-based verification is insecure for a number of different reasons, including the potential for interception of SMS messages via Signaling System 7 network vulnerabilities and the potential for social engineering and SIM-swapping attacks. In the latter scenarios, an attacker can gain access without physical access to the phone and potentially without the target's knowledge. For this reason, all Sharekey authentication and two-factor authentication requests are transmitted via email.

Also, email traffic is often quicker and more reliable than SMS, supporting a more rapid initial sign-up for a Sharekey account as well as subsequent sign-in to the app from various devices. While these emails are not encrypted end-to-end (but are TLS-encrypted between email servers), they carry lower risk than SMS, which are more exposed due to shortcomings in telecommunication protocols and over-the-air, unencrypted communications.

The Sharekey app is signed using an Extended Validation (EV) X.509 code signing certificate from SSL.com. This offers the highest level of authenticity and integrity assurance.

History

Many enterprises, especially those in regulated areas such as legal and financial services, have the requirement to preserve conversations. Many consumer-facing solutions that create self-destructing or end-to-end encrypted messages do not meet such regulatory requirements and are increasing the subject of scrutiny by the relevant authorities in several countries.

As a business app, Sharekey allows users to access their conversations (pairwise or group) on different devices, with the guarantee that all devices show the full conversation history. In particular, a new device should be capable of showing the previous messages in a given conversation. Furthermore, devices should smoothly recover conversations after being offline for an extended period of time and generally minimise the risk of data loss that could arise from a desynchronised state, a common issue with secure messengers.

The Shared Key used for encrypting Direct Messaging or Channel communication is not "ratcheted", which ensures that all devices can decrypt previous messages. This prevents the protocol from providing perfect forward secrecy or providing the guarantee that an attacker compromising the application's keys cannot decrypt past messages. However, if a machine or device is compromised, an attacker can, in most cases, directly access the local history of decrypted messages or their decryption keys.

Metadata

Sharekey requires only the absolute minimum metadata on its users – just enough to be able to offer the necessary features. It needs the following metadata to be able to operate:

- **Connections:** the initiator of the Direct Messaging or the creator of a Channel (also called the Owner), the name of the Channel and a pseudonymised list of Members.
- **Files and Folders:** the creator of a file and folder (also called the Owner), the pseudonymised list of Members it is shared with, and the role of each user (Owner, Administrator, Editor, Viewer).

On the whole, Sharekey receives and processes less metadata than similar communications apps. The Sharekey Roadmap includes efforts to reduce the metadata collected by Sharekey by eliminating most of these privacy-sensitive types of metadata. The plan is to enable Sharekey to reduce its metadata collection to the minimum required to operate the service.

Security Culture

Security is at the heart of what Sharekey does, and this is manifest in our workforce and corporate culture.

Our People

Our security culture follows all the best practices from ISO 27001, including the ones on the recruitment and ongoing security practices during employment.

Before they join our staff, Sharekey verifies an individual's education and previous employment, and performs internal and external reference checks. The extent of these background checks depends on the desired position.

All new employees are required to read and understand Sharekey's documented Security Policy. While employed, Sharekey employees undergo frequent security awareness training and are made aware of the latest cyber threats and how to prevent such attacks.

Also, we emphasise to our engineers the need for ethical development. These include the need to ensure that our solution maintains our users' data privacy and that we do not monetise their data or metadata.

Our Planet

Also, as a good corporate citizen, Sharekey emphasises that its employees be practitioners of Green IT, an informal but popular movement that encourages the IT industry to work towards solutions that reduce our carbon footprint.

To this end, Sharekey has taken several steps to support Green IT:

- **Server:** Sharekey runs its backend servers on 100% renewable energy.
- **Storage & Networking:** The key feature of our solution – sharing keys instead of files – is itself a contribution to the principles of Green IT, as it reduces both storage and communication bandwidth, thereby reducing the amount of infrastructure and hence the energy needed to run it.
- **Client:** Our optimised code on clients ensures that devices, in turn, do not inadvertently bear any additional burden as a result of server-side optimisations.

And all this, without compromising the security of our solution.

Key Management

Principal Keys Overview

When a user signs up for a Sharekey account, the initial key management process starts with the generation of a Secret Phrase. This is used for the first login to the Sharekey app on a device.

The Secret Phrase is used as input to a key generation process that creates a Private Key — the Master Secret Key — and its public counterpart. The Public Key is available as a part of the user's profile and is exchanged with other users to establish direct, encrypted communications links.

A second public/private keypair is also generated for purposes of creating and verifying digital signatures. A signature is generated using the private Master Signing Key.

The Sharekey app on a device is protected using a Password (on a desktop) or a Passcode (on mobile devices).

The remainder of this section provides the technical details of how these keys are derived, used and protected.

A later section will describe how the encryption key — the Shared Key — for a Direct Message or Channel is generated and used.

Secret Phrase Generation (24 Secret Words)

The key generation process draws randomness from a system-provided Cryptographically Secure Pseudorandom Number Generator (*CSPRNG*). For desktop and web apps, this is exposed via the *window.crypto* browser functionality, while *SecRandomBytes* and *SecureRandom* are used for iOS and Android respectively. This secure source of randomness is used to randomly select twenty-four words from a built-in dictionary of 1700 words. This produces a Secret Phrase with a total entropy of $1700^{24} > 2^{256}$ bits.

This Secret Phrase is provided to the user as part of the account creation process and is used to perform initial login to the Sharekey application. The user is strongly urged to securely store this Secret Phrase, as it is essential for signing in to the Sharekey app on a new device or restoring an account if the Password or Passcode used to unlock the Sharekey app is forgotten.

Key Generation

The Sharekey account creation process creates two keypairs and makes these available to the application during the sign-in process to a new device.

Master Secret Key

The 256-bit Private Key used for data encryption/decryption — hereafter called the Master Secret Key — is derived from a randomly selected, twenty-four-word Secret Phrase using the *scrypt* key derivation function as described below.

This private Master Secret Key is securely stored on the device. The public part of this key pair is derived using the standard elliptic-curve Diffie-Hellman key agreement protocol. This Public Key is sent to the Sharekey server and used later during the connection setup between users.

Master Signing Key

Another secret – hereafter called the Master Signing Key – will be used later for digital signature creation and verification. It is generated using the Cryptographically Secure Pseudorandom Number Generator (*CSPRNG).*

The Master Signing Key is encrypted with the Master Secret Key and stored on the Sharekey server. Its use is described in a later sub-section.

Random Key

Random keys are generated using *CSPRNG* when needed, as, for instance, when a Channel needs to be set up for group communication.

Key Derivation

scrypt is a password-based key derivation function [1]. It takes the Secret Phrase mentioned previously and, along with a few other parameters, uses it to generate the Master Private Key for various algorithms mentioned below. Sharekey uses the following implementation of *scrypt*:

- Desktop/Web: https://github.com/ricmoo/scrypt-js
- Mobile: https://github.com/crypho/react-native-scrypt

```
Inputs:
     Passphrase:                  Bytes     String of characters to be hashed
     Salt:                        Bytes     Random salt
     CostFactor (N):              Integer   CPU/memory cost parameter
     BlockSizeFactor (r):         Integer   Blocksize parameter
     ParallelizationFactor (p):   Integer   Parallelization parameter. (1..2^32-1 * hLen/MFlen)
     DesiredKeyLen:               Integer   Desired key length in bytes
Output:
     DerivedKey:                  Bytes     Array of bytes, DesiredKeyLen long
```

The image above shows the function definition for the use of *scrypt* as a key derivation function. The user's Secret Phrase will be provided as the *Passphrase* and the *SHA256* hash of the user's email is used as the *Salt*. The desired output length will depend on the algorithm in which the final key will be used. Sharekey uses 256 bits for both data encryption (*XSalsa20*) and digital signatures (*Ed25519)*.

A 256-bit key means that there are 2^{256} – ie 115792089237316195423570985008687907853269984665640564039457584007913129639936 – potential Secret Keys. While the time it would take to brute force a 256-bit key depends on the hardware used, no modern computer is capable of breaking AES-256 in a reasonable amount of time. The Tianhe-2 supercomputer – the fastest supercomputer in the world – would take millions of years to expose by brute force a single AES key [2].

scrypt is designed to be particularly memory intensive. These high memory requirements combined with *Salt* (unique for each user) make it difficult for an attacker to generate rainbow tables, which would expedite the password cracking process.

Implementation of *scrypt* has adjustable CPU and memory settings. Sharekey uses a CPU cost of 2^{15} and a memory cost of 4 (at least 16 MiB) to balance usability and security.

Session Authorisation Token

As a part of the sign-in process from a new device, a session authorisation token is generated that serves to ensure that only a legitimate Sharekey user can communicate with the backend.

The authorisation code is generated on the server using *CSPRNG* with a length of 43 characters and entropy of 6 bits per character, resulting in 256 bits of entropy. This authorisation code is valid for 90 days from generation. For many requests to the server, this token serves as "good enough" confirmation of user access. However, for security reasons, many requests, including any that involve sending messages or other high-risk actions, require a digital signature of the sent data using the user's Master Signing Key.

Figure 4 shows the interactions that verify that the user has access to the Secret Phrase and can therefore derive the Master Signing Key with which to create digital signatures.

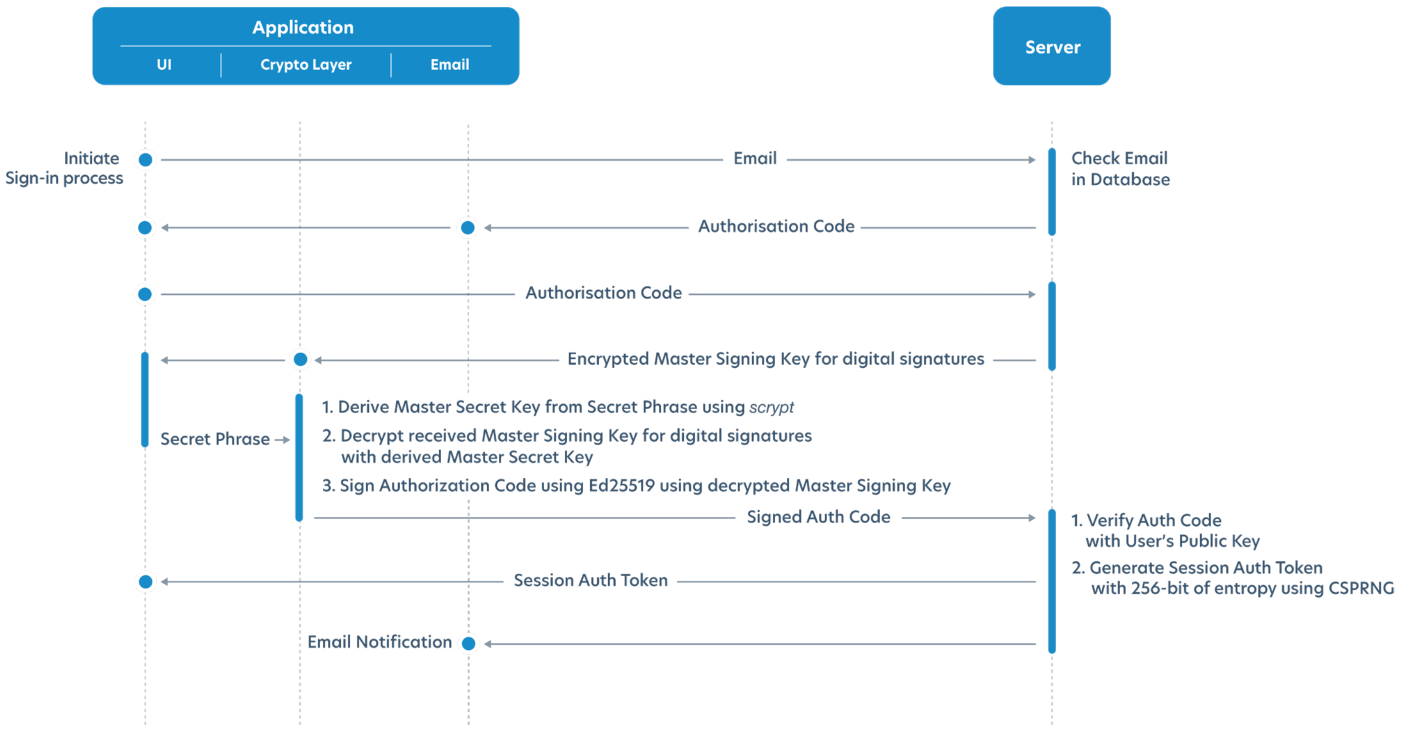

Application
UI
Crypto Layer
Email
Server
Initiate Sign-in process
Email
Check Email in Database
Authorisation Code
Authorisation Code
Encrypted Master Signing Key for digital signatures
Secret Phrase
1. Derive Master Secret Key from Secret Phrase using scrypt
2. Decrypt received Master Signing Key for digital signatures with derived Master Secret Key
3. Sign Authorization Code using Ed25519 using decrypted Master Signing Key
Signed Auth Code
1. Verify Auth Code with User's Public Key
2. Generate Session Auth Token with 256-bit of entropy using CSPRNG
Session Auth Token
Email Notification

Figure 4 • New Device Sign-in Interactions

The first three steps in Figure 4 show the sign-in process from a new device. Sharekey uses the user's email to identify the user, and validates that email by requiring the user to return, via the app and within five minutes, a 6-digit authorisation code sent to that email address. (The user experience for these steps is shown later in this document.) This step ensures that the user is able to access their email from that device.

At this point, the server returns the encrypted Master Signing Key — encrypted with the Master Secret Key — that was stored by Sharekey during the initial account setup (as described earlier). The purpose is to verify that the Owner of the device with that email address has proof of possession of the Secret Phrase which is essential to the process of creating a digital signature.

The first step on the new device is to derive the Master Secret Key from the Secret Phrase using *scrypt*. This then allows the decryption of the Master Signing Key. The Master Signing Key can now be used to sign the previously-received authorisation code using the *Ed25519* algorithm.

The returned signed authorisation code is then verified by the server using the public part of the Master Signing Key. A successful verification proves that the user and device have proof of possession of the Secret Phrase.

After the server verifies the signature of the received authorisation token, it generates a session authorisation token with 256 bits of entropy using *CSPRNG.* This will be used in subsequent communication between the app and the server.

Additionally, all communications occur within TLS tunnels.

User Account & Device Management

Account Creation (Sign Up)

The Sharekey account creation process takes place in three steps, as outlined below:

1. The user verifies their email address.
2. The Sharekey app generates the Secret Phrase that the user can use for future sign-ins to new devices.
3. Finally, Sharekey uses this Secret Phrase and the *scrypt* key derivation algorithm to generate the Master Secret Key. This key is used as the Private Key of the public/private keypair and is used when securely connecting with other users and to protect the other keys used for data protection and authentication throughout the Sharekey ecosystem.

Each stage is described in the subsections below.

User Verification

The Sharekey application is available from the Sharekey website (sharekey.com) or in mobile (iOS and Android) App Stores. To circumvent the most common type of phishing attack, where an attacker creates a similar webpage accessible via a closely related (but mis-spelt) domain name, Sharekey has registered and controls more than 430 variants of its domain name.

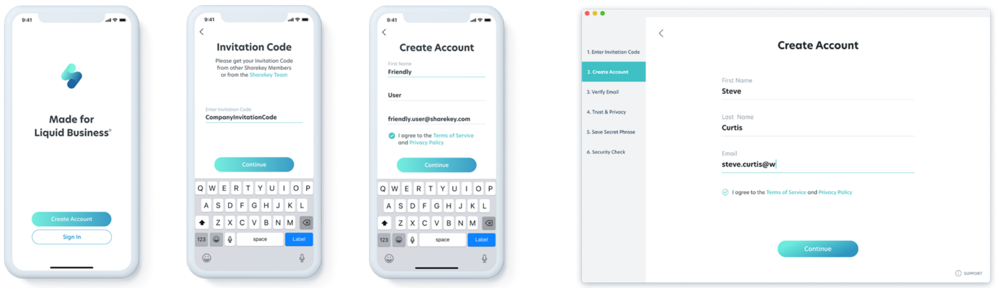


Figure 5 • User Account Creation (Sign Up)

After downloading and installing the application, users are presented with the leftmost screen shown in Figure 5.

After selecting Create Account, users are asked to enter an Invitation Code (which can be requested on the Sharekey website). Users then fill out their name and email address, which is all that is required to create a Sharekey user account. As an enterprise communications and collaboration application, Sharekey uses just this minimum amount of data to uniquely identify users, rather than seek additional information such as phone numbers and other data commonly collected by consumer-focused applications.

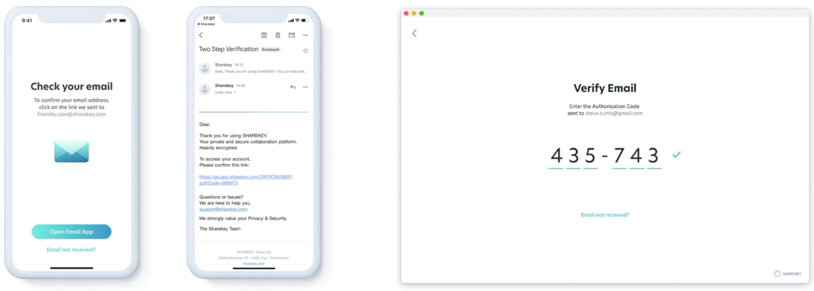


Figure 6 • User Account Creation Verification

To verify a user's email address, Sharekey offers two options, depending on whether the user is at a mobile device or a desktop.

- For a **mobile device**, Sharekey sends a confirmation email to the provided address, as shown on the left-hand side of Figure 6.
- For a **desktop**, the user enters the six digits received via email, as shown on the right-hand side of Figure 6.

Secret Phrase Generation

At this point in the initial account setup process, the Sharekey app generates the Secret Phrase which is used as input to the Secret Key generation process described earlier in the document.

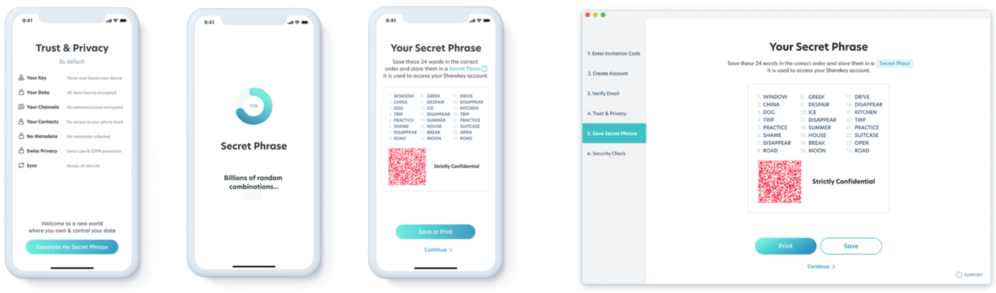


Figure 7 • Secret Phrase Generation

As Figure 7 shows, this Secret Phrase is displayed to the user and is used to perform an initial login to the Sharekey application.

The Secret Phrase is also used to restore an account if the application Password or Passcode on a device is lost or forgotten. If a user's Secret Phrase is lost or forgotten and they are not logged into any devices, there is no way to restore their Sharekey account.

Given this critical role played by the Secret Phrase, the user is made to save a copy or print it. As Figure 8 shows, Sharekey requires them to enter three words from the phrase to verify that they have done so.

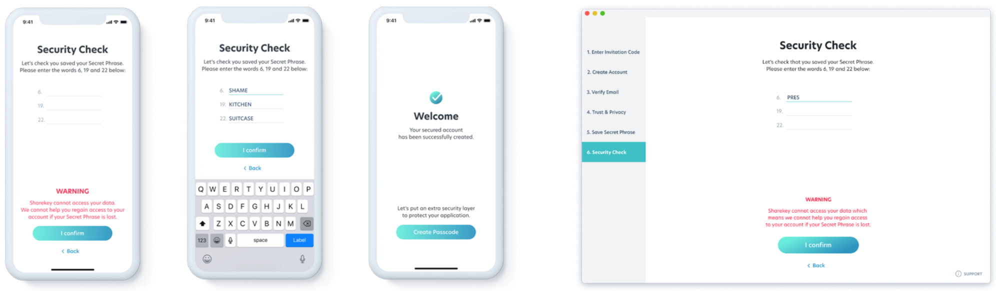

Security Check
Let's check you saved your Secret Phrase. Please enter the words 6, 19 and 22 below:
WARNING
Sharekey cannot access your data. We cannot help you regain access to your account if your Secret Phrase is lost.
I confirm
Back
SHAME
KITCHEN
SUITCASE
Welcome
Your secured account has been successfully created.
Let's put an extra security layer to protect your application.
Create Passcode
1. Enter Invitation Code
2. Create Account
3. Verify Email
4. Trust & Privacy
5. Save Secret Phrase
6. Security Check

Figure 8 • Secret Phrase Check

This completes the initial account setup process.

Device Setup (Sign In)

When setting up a Sharekey account on a new device, a user will be asked to go through the process illustrated by Figure 9.

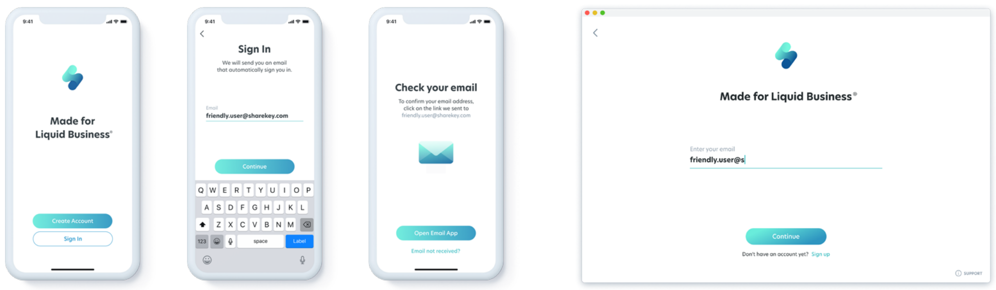

Made for Liquid Business®
Create Account
Sign In
We will send you an email that automatically sign you in.
friendly.user@sharekey.com
Continue
Check your email
To confirm your email address, click on the link we sent to friendly.user@sharekey.com
Open Email App
Email not received?
Enter your email
Don't have an account yet? Sign up

Figure 9 • New Device Setup (Sign In)

When signing into a Sharekey account on a new device, a user's email is validated by sending a confirmation link valid for five minutes to that address. This ensures that a Sharekey user is made aware that someone is attempting to add a new device with access to their account. It also acts as a second authentication factor.

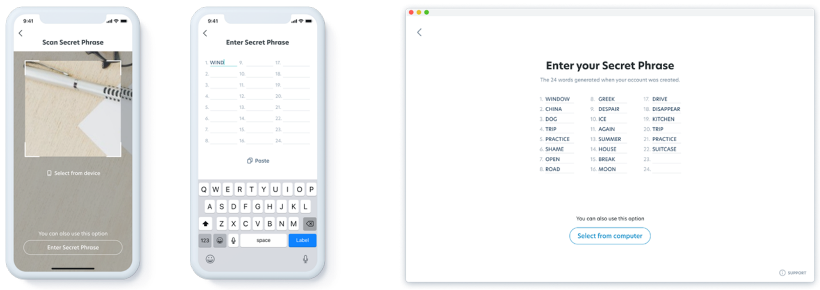


Figure 10 • Secret Phrase Entry

A user's Secret Phrase is used to derive the Master Secret Key, making it available to the new device. When setting up a Sharekey app on the new device, the Secret Phrase can either be entered using the QR code reader, or manually via the wordlist or by uploading the PDF file containing the Secret Phrase (choosing the "Select from Device" option), as shown in Figure 10. (Recall from the previous discussion that the user was required to save the Secret Phrase for precisely this purpose).

To sign into a new device, both access to the Secret Phrase (either as a wordlist or a QR code) and the user's email account are required (for multi-factor authentication purposes). The support of additional multi-factor authentication options currently planned are included in the Sharekey Roadmap.

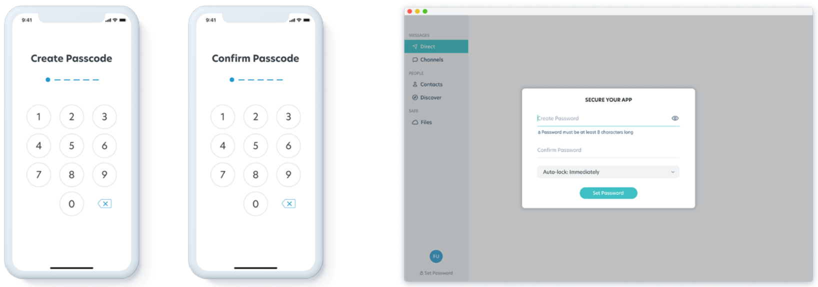


Figure 11 • Application Passcode or Password Creation

After the user has successfully logged into the Sharekey app, they are invited to create a Password (on desktop systems) or a Passcode (on mobile devices) as shown in Figure 11. The Passcode on a mobile is a six-digit numeric code (thus offering one million combinations), while Passwords on a desktop can support a minimum of eight alphanumeric characters and symbols allowed by the NIST recommendations described in SP 800-63B [3].

This Password or Passcode is used to unlock a logged-in user account on the desktop or mobile, respectively. By using a Password or Passcode instead of the user's Secret Phrase, Sharekey makes it easier to unlock the app.

- On a desktop, the Password protects the app from unauthorised access. A key derived from the Password encrypts the Secret Phrase in the computer or browser's RAM and is never stored locally in the device.
- On a mobile, the Secret Phrase is encrypted locally using the native protection system of the device. These steps prevent the disclosure of the user's Secret Phrase if the device is stolen.

Authentication (Unlock App)

For the user to have access to their data (messages, files, etc.), they need access to the user's Master Secret Key derived from their Secret Phrase. On the desktop, this Secret Phrase is stored encrypted with a key derived from the user's Password. On mobile devices, it is located in the device-specific secure storage (iOS Key Chain or Android Key Store). When the user authenticates to the app with their Password or Passcode, as shown in Figure 12, the app can retrieve this encryption key and use it to access the local database. Users are allowed ten failed attempts before they are locked out.

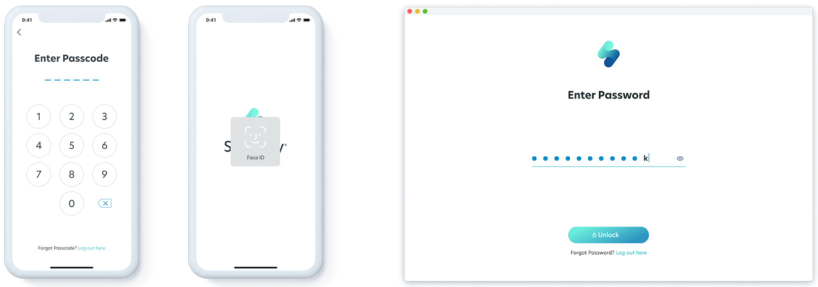


Figure 12 • Authentication (Unlock App)

Connecting Users

Direct encrypted connections between Sharekey users, using a key agreement protocol with the users' cryptographic identities, is a fundamental part of how the Sharekey system works.

Sharekey's privacy focus means that even group communications (called Channels) are set up via encrypted links between the Channel creator (also called the Owner) and its Members. The methods by which these keys are generated, shared, and used are detailed in the Messaging section below.

As a result, Sharekey users need a method for sharing their keys and establishing these communications links. Sharekey users can connect to one another in several ways:

- **Discover:** The Sharekey directory enables users to search for other users with public profiles. (User profiles are created private by default, but users can choose to make theirs public.)
- **Key Sharing:** Sharekey users can exchange Public Keys directly using QR codes or copy/paste these into out-of-band messages (e.g., emails, text, etc.).
- **Channel Members:** All Members of a Sharekey Channel can connect directly to one another.

The data that Sharekey collects and stores to create direct communications or Channels is outlined in our Privacy Policy [4].

Discover

Sharekey offers an integrated directory, called Discover, of Members who have set their profiles to be publicly visible. Users can also augment their public profile with their name, user-selected image, location and tagline. By using the Discover feature, Sharekey users can find and connect easily to other users of Sharekey, as shown in Figure 13.

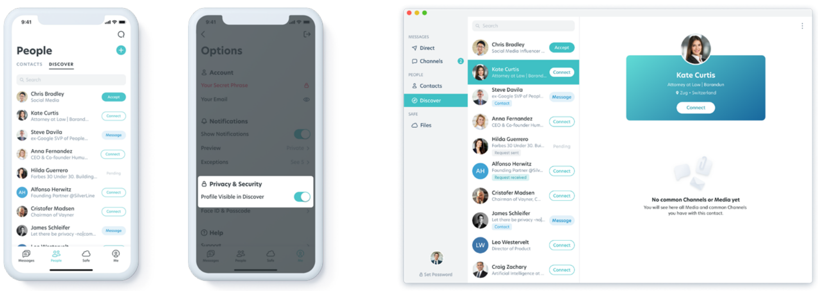


Figure 13 • Discovering Members

Only registered Sharekey users have access to the directory of Members. Thus, profiles can only be shared by and between registered Members.

By default, new users' profiles are hidden in Discover until they set their profile to visible under the Settings accessible from the Me page/pane. Using the Discover pane, Sharekey users can browse other users' profiles and establish a connection by sharing their Public Key.

Key Sharing

Members that wish to connect without being publicly visible in Discover have a few options for doing so. Two options exist to share a Public Key (also called "My Key" in the app) with another Member:

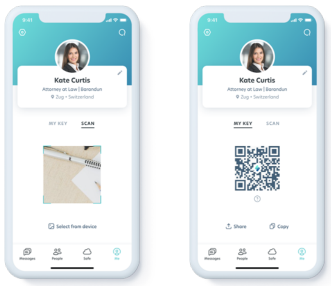


Figure 14 • Key Sharing

- Scanning the QR code from the Me screen on mobile as shown in Figure 14, or
- Using the Copy option on the My Key screen to access a Base64-encoded Public Key, which can be shared over another communications app. The recipient can then connect to the Member by pasting this Public Key in the Search bar of the Discover pane.

Channel Members

Sharekey supports group conversations within Channels. A Sharekey user can view the profile of any Members of a Sharekey Channel as shown in Figure 15.

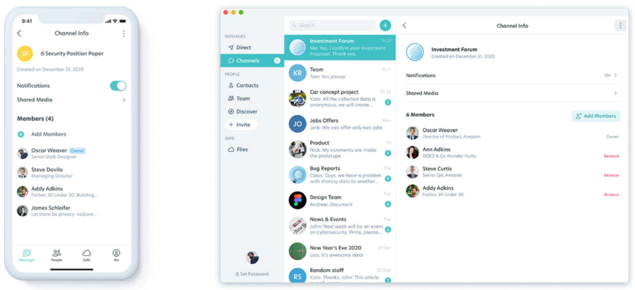


Figure 15 • Channel Information

This enables a Member to establish direct connections with other Channel Members. This is a third option for Members to connect without being visible in Discover, but it assumes that the Members are already part of the same Channel.

Messaging

All messaging on Sharekey is fully end-to-end encrypted. Upon initial connection, users exchange Public Keys. As these keys are currently exchanged via Sharekey servers, the Sharekey Roadmap includes the development of additional protections against Man-in-the-Middle (MitM) attacks. These keys are then used to establish Shared Keys that allow encrypted Direct communications or sharing of Channel encryption keys.

Direct Messaging

Direct Messaging (also called Direct Message or DM) involves only those two users, as shown in Figure 16. These users establish a Shared Key using the Diffie-Hellman key exchange protocol [5]. To aid readers unfamiliar with this standard key generation protocol, we offer a brief explanation of the steps by which two users derive a Shared Key which encrypts their subsequent communication.

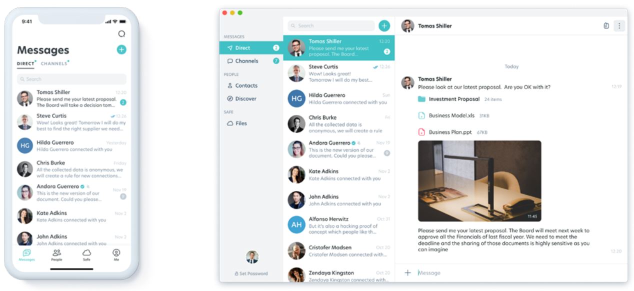


Figure 16 • Direct Messaging

As shown in the bottom of Figure 17, the Sharekey application assigns to Alice and Bob a long-term identity in the form of a cryptographic key pair (the Public Key and Master Secret Key), generated securely on their devices using their respective Secret Phrases. This process, which includes email-based identity verification, has been described in a previous section.

Alice and Bob exchange (or find via their Sharekey profiles) their Public Keys. If Alice does not have access to Bob's Public Key (perhaps because Bob did not make it available via his public profile), he can receive Alice's Public Key when she initiates the request for creating a Direct Message (called "Connection Request" in the Sharekey app). He returns his Public Key if he wishes to complete the connection (called "Accept" in the app).

Then, as shown in the middle of Figure 17, Alice takes Bob's Public Key and her Master Secret Key to create a Shared Key ("B2FD...A6F5" as an example in the figure). Similarly, Bob uses his Master Secret Key and Alice's Public Key to create the same Shared Key. That is the magic of this Diffie-Hellman based Shared Key generation algorithm!

All subsequent communication between the two users is now encrypted with this Shared Key. This means that, although messages pass through Sharekey's servers *en route* to their destination, Sharekey has no visibility into these messages.

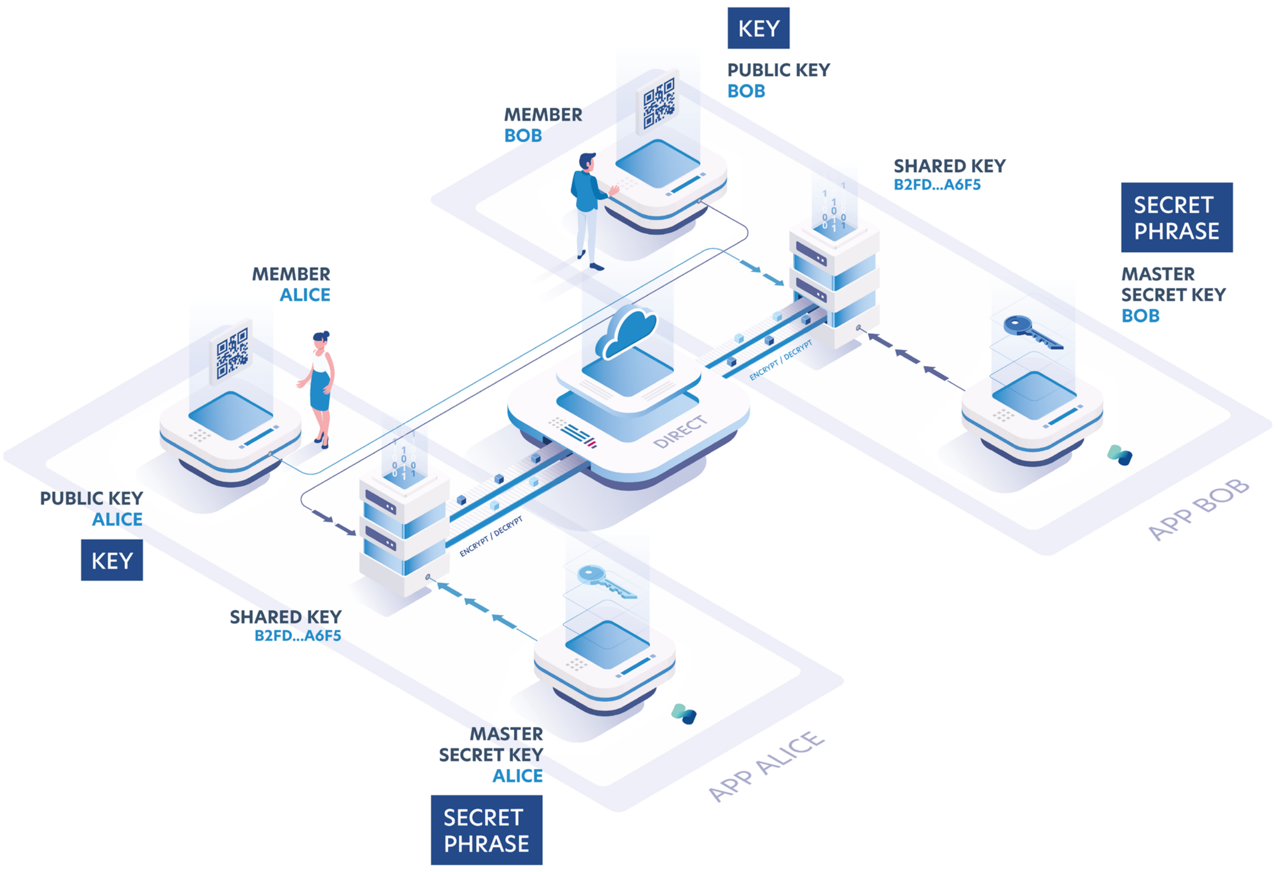
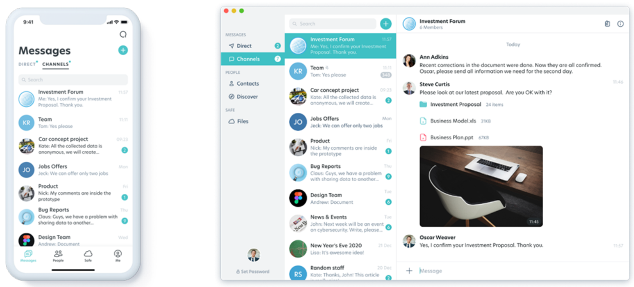


Figure 17 • Shared Key for Direct Messaging

Channel Messaging

Channel messaging (also called Channels) is designed for group communications, as shown in Figure 18, and is built up as a series of one-to-one Direct Messages.



Investment Forum 11:57
Me: Yes, I confirm your Investment Proposal. Thank you.

Team 11:11
Tom: Yes please

Car concept project 09:23
Kate: All the collected data is anonymous, we will create...

Jobs Offers Mon
Jeck: We can offer only two jobs

Product Fri
Nick: My comments are inside the prototype

Bug Reports Thu
Claus: Guys, we have a problem with sharing data to another...

Design Team
Andrew: Document

News & Events Tue
John: Next week will be an event on cybersecurity. Write, please...

New Year's Eve 2020 21 Dec
Lisa: It's awesome idea!

Random staff 20 Dec
Kate: Thanks, John! This article

Investment Forum
6 Members

Today

Ann Adkins
Recent corrections in the document were done. Now they are all confirmed. Oscar, please send all information we need for the second day.

Steve Curtis 11:46
Please look at our latest proposal. Are you OK with it?
Investment Proposal 24 items
Business Model.xls 31KB
Business Plan.ppt 67KB
11:45

Oscar Weaver 11:57
Yes, I confirm your Investment Proposal. Thank you.

Figure 18 • Channel Messaging

As illustrated in Figure 19, the Channel creator (also called the Owner), Charlie in our example, generates a 256-bit random key which will become the Shared Key ("C4E6...B6D4" as an example in the figure) for the Channel. This Shared Key is generated using the Cryptographically Secure Pseudorandom Number Generator (*CSPRNG*).

When a new Member, Bob, is invited to the Channel, the person sending the invite (Charlie in our example) uses his Direct Messaging connection with Bob to send him the Shared Key for the Channel. Charlie will also use his Direct connection to send to Alice the same Shared Key for the Channel when he invites her to join (called "Add Members" in the Sharekey app).

Note:

- In the case of a Direct Message, the Shared Key is derived in the app from a combination of one user's Public Key and the other user's Master Secret Key, using the magic of the Diffie-Hellman key exchange protocol.
- In the case of a Channel, the Shared Key is generated randomly and then distributed individually to invite Members to join the Channel.

A new Channel Member's Public Key is made available to other Members of the Channel.

The new Channel Member can then download and decrypt past and future messages within the Channel. Optionally, the Channel Member who invites the new Member to the Channel can prevent the new Member from accessing previous messages in the Channel by generating a new Shared Key and only sharing this latest encryption key with the new Member as well as the other Members.

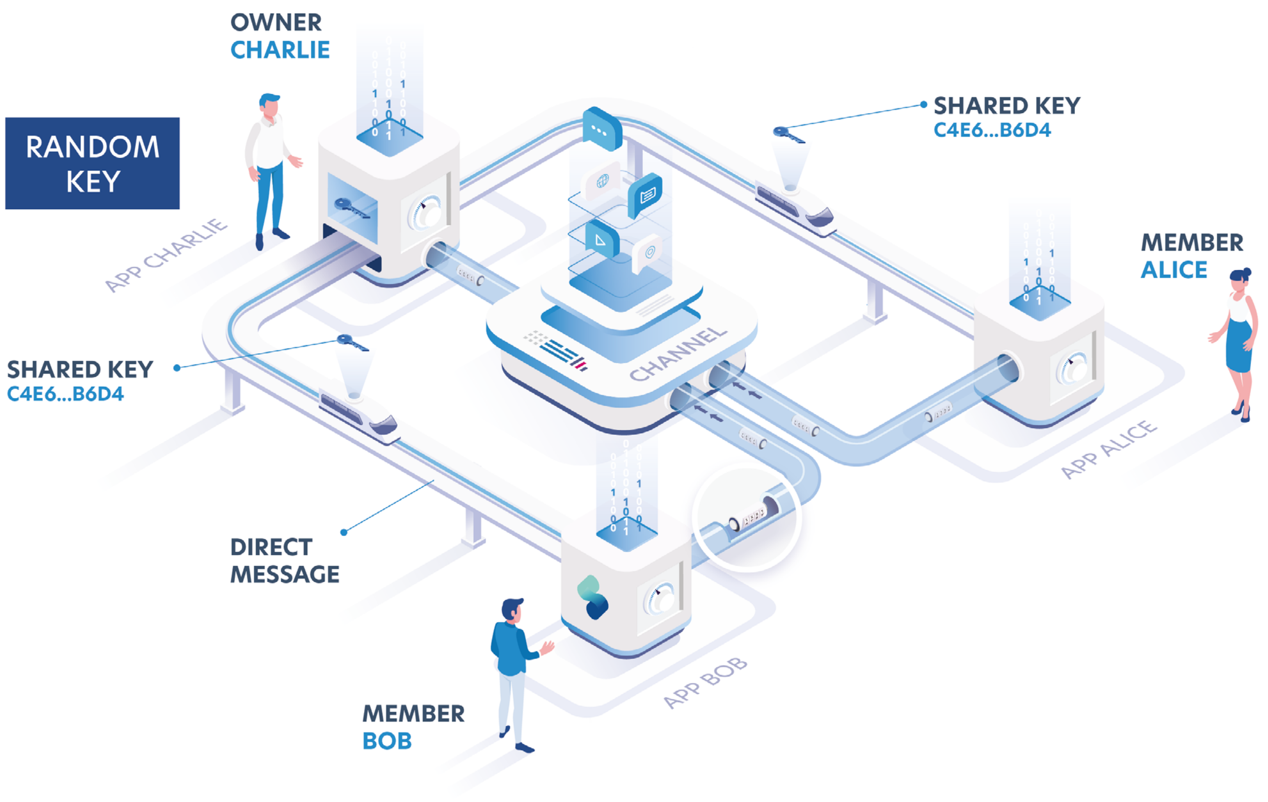


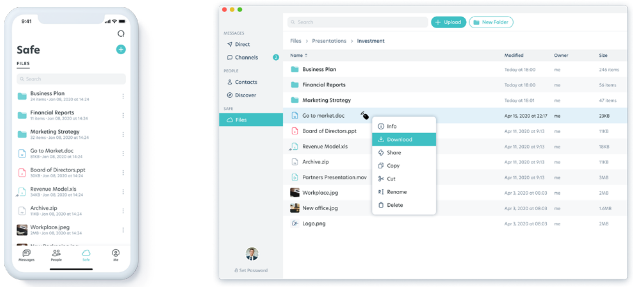
Figure 19 • Shared Key for Channel Messaging

If a Member is removed from a Channel, the Channel Member who removes the Member generates a new Shared Key and distribute it to all others Members via the same mechanism as for the Channel creation (i.e., via Direct Messaging). All future messages are encrypted only with this new Shared Key, making it impossible for the removed Member to view them.

Additionally, the revoked Shared Key is retained by the Channel and used by the remaining Members to access earlier messages that were encrypted with the now-deprecated Shared Key for the Channel.

Storage

Sharekey mediates data sharing between users by providing storage on its servers in the cloud. A user's storage on the server is called a SAFE. A SAFE supports storage of both files and folders.



Business Plan
24 items · Jan 08, 2020 at 14:24

Financial Reports
11 items · Jan 08, 2020 at 14:24

Marketing Strategy
32 items · Jan 08, 2020 at 14:24

Go to Market.doc
81KB · Jan 08, 2020 at 14:24

Board of Directors.ppt
30KB · Jan 08, 2020 at 14:24

Revenue Model.xls
34KB · Jan 08, 2020 at 14:24

Archive.zip
11KB · Jan 08, 2020 at 14:24

Workplace.jpeg
2MB · Jan 08, 2020 at 14:24

Files > Presentations > Investment

Name	Modified	Owner	Size
Business Plan	Today at 18:00	me	246 items
Financial Reports	Today at 18:00	me	56 items
Marketing Strategy	Today at 18:01	me	47 items
Go to market.doc	Apr 15, 2020 at 22:17	me	23KB
Board of Directors.ppt	Apr 11, 2020 at 9:13	me	11KB
Revenue Model.xls	Apr 11, 2020 at 9:13	me	18KB
Archive.zip	Apr 11, 2020 at 9:13	me	11KB
Partners Presentation.mov	Apr 11, 2020 at 9:13	me	3MB
Workplace.jpg	Apr 3, 2020 at 08:03	me	2MB
New office.jpg	Apr 3, 2020 at 08:03	me	1.6MB
Logo.png	Apr 3, 2020 at 08:03	me	2MB

Figure 20 • SAFE Storage

Sharekey allows users to store their data fully encrypted on its servers in the cloud. A user's storage is composed of:

- a SAFE, for files and folders, as shown on Figure 20;
- all messages and attachments shared via Direct Messages and Channels and visible only there.

Sharekey users share the data in their SAFE with other Sharekey users via messaging.

Each file stored in a user's SAFE is encrypted with a unique encryption key generated by the Owner of the file. This encryption key is then encrypted with the user's Master Secret Key.

Both the encrypted file and the encrypted key are uploaded to Sharekey's cloud storage. This makes it possible for users to access their data on any device. Moreover, because the user's Master Secret Key is required for decryption, Sharekey has no visibility into a user's saved files.

Figure 21 shows the types of information stored and shared for each type of content. All content types include an encrypted name and modification date. Folders provide additional information about their contents (including number of items inside and each of their encryption keys). File metadata contains information about the size of the file and the type of content- e.g., document (pdf, doc, etc.), or image (e.g., png, mpeg, etc.).

The Sharekey Roadmap includes adding support for an encrypted local database where all downloaded and opened files/folders can be stored.

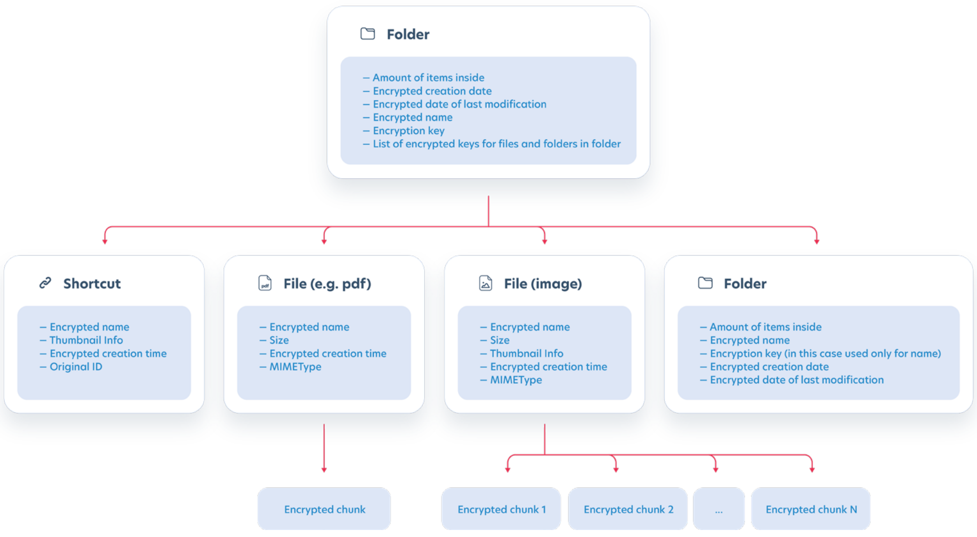


Figure 21 • SAFE Structure

Data Sharing

Sharekey allows users to send the data saved in their SAFE via messaging. Both file and folder sharing are possible.

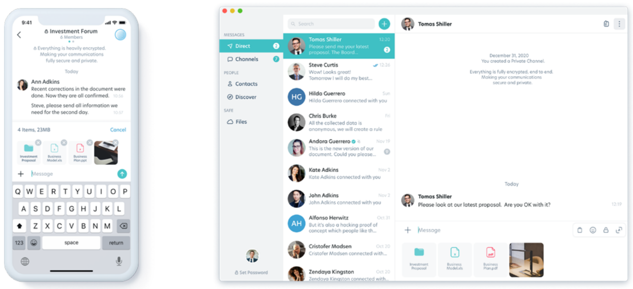


Figure 22 • Data Sharing

File Sharing

Files stored within a user's SAFE can be shared with other users, as shown in Figure 22. Sharing can occur either via Direct Messages or Channels.

If a user wishes to share a file via Direct Messaging or a Channel, the decryption key is shared with that message. This allows the recipients to download and decrypt the file. Moreover, because Direct Messages and Channels are encrypted, no one else can access and decrypt the file.

Folder Sharing

Sharekey users are also able to share folders via Direct Messages or Channels. Each folder has its own unique encryption key, similar to files. Each folder includes the list of the files and folders that it contains along with their encryption keys. This enables decryption of the contents of the files and folders within this folder. As every embedded folder contain the decryption keys of its contents, the entire directory tree can be recursively decrypted.

Rights Management

The Owner of files and folders has the ability to manage other users' rights to these assets. At a high level, file and folder Owners can grant or revoke access to them. These rights are stored as a collection of database records, and the Owner of a record (or a user who is granted access to this record) can revoke the rights for a specific user by deleting the specific record.

Sharekey also supports more granular rights management for files and folders. The Owner of a file or folder can assign to other users the role of a Viewer, Editor, or Administrator, providing read-only, read and write, or full access and sharing control respectively.

	ADMINISTRATOR (OWNER)	EDITOR	VIEWER	CUSTOM
FILE				
Open	✓	✓	✓	☐
Download and Copy	✓	✓	✗	☐
Add to Safe (shortcut)	✓	✓	✗	☐
Rename	✓	✓	✗	☐
View Members who have access to this file	✓	✓	✗	☐
Add Members to this file (reshare)	✓	✓	✗	☐
Remove Members (revoke access)	✓	✓	✗	☐
View list of Channels where this file is shared	✓	✓	✗	☐
Move	✓	✓	✗	☐
Delete	✓	✗	✗	✗
FOLDER				
Open	✓	✓	✓	☐
Download or Copy	✓	✓	✗	☐
Add to Safe (shortcut)	✓	✓	✗	☐
Rename	✓	✓	✗	☐
View Members who have access to this folder	✓	✓	✗	☐
Add Members to this folder (reshare)	✓	✓	✗	☐
Remove Members (revoke access)	✓	✓	✗	☐
View list of Channels where this folder is shared	✓	✓	✗	☐
Move this folder	✓	✓	✗	☐
Edit folder content	✓	✓	✗	☐
- upload file/folder	✓	✓	✗	—
- create folder -> create note v2 -> create file v3	✓	✓	✗	—
- delete items inside	✓	✓	✗	—
- move file/folder INTO this shared folder	✓	✓	✗	—
- move file/folder OUT of this shared folder	✓	✗	✗	—
Delete shared folder	✓	✗	✗	—

Figure 23 • Rights per Role for Files & Folders

Figure 23 shows the various permissions that each role has regarding a file or folder. By default, a user will be granted Editor permissions when a file or folder is shared with them.

However, the person sharing the file or folder can also define a custom set of permissions, allowing more granular permissions control. For example, it may be desirable to allow a user read/write access to a file but not allow them to reshare the files with others and to download them.

The rights that are granted for a particular file or folder are based upon a number of factors, including:

- **Item:** The shared file or folder;
- **Sharer:** The user who gives rights for the shared file/folder to the "Given to" (see below) party;
- **Owner:** The initial Owner of the file or folder;
- **Given to:** The user to whom rights to this file or folder is given.

These identities and permissions are used in the rights management process to ensure that a new user's access to a file or folder are appropriately limited. For example, the user sharing the file or folder cannot grant rights that they do not have themselves.

Local File Storage

When a file is shared via Sharekey and opened by the user, it is stored locally on the receiving device.

Files shared via Sharekey are decrypted and stored on a user's device, making it possible to open them using the appropriate application.

Sharekey supports opening images natively within the app (though support of additional file formats are in the Roadmap). All other file types must be stored after being decrypted at a location where an appropriate application (Adobe Reader, Microsoft Office, etc.) can access them for display to the user. For this reason, the use of a full-disk encryption solution like Windows BitLocker or macOS FileVault is highly recommended.

On both desktop and mobile devices, cached files are automatically deleted a week after last access. On a mobile device, cached files are automatically stored in the local Realm database of the app and are deleted when the app is deleted.

Data Encryption

The Crypto Layer — shown in Figure 3 earlier in the document and reproduced partially below in Figure 24 — is at the heart of the Sharekey system and contains our assembly of various **open-source cryptographic algorithms and protocols** to achieve our goal of app-to-app encryption.

All messages and data stored and shared via Sharekey are encrypted. Data is encrypted locally and uploaded to Sharekey servers in their encrypted form. Sharekey currently uses a combination of symmetric, asymmetric, and transport-layer encryption for security. Moreover, additional cryptographic support is currently on the Roadmap.

Crypto Layer in App

Sharekey uses public-key-based authentication encryption, namely the *crypto_box* implementation of NaCl [6]. While tweetnacl-js is currently used in JavaScript environments [7], the intent is to move to libsodium for all platforms. Both implementations use the following algorithms:

- **Data Encryption:** *XSalsa20*
- **Authentication:** *Poly1305*
- **Digital Signatures:** *Ed25519*

The flow of data through the system is shown in Figure 24.

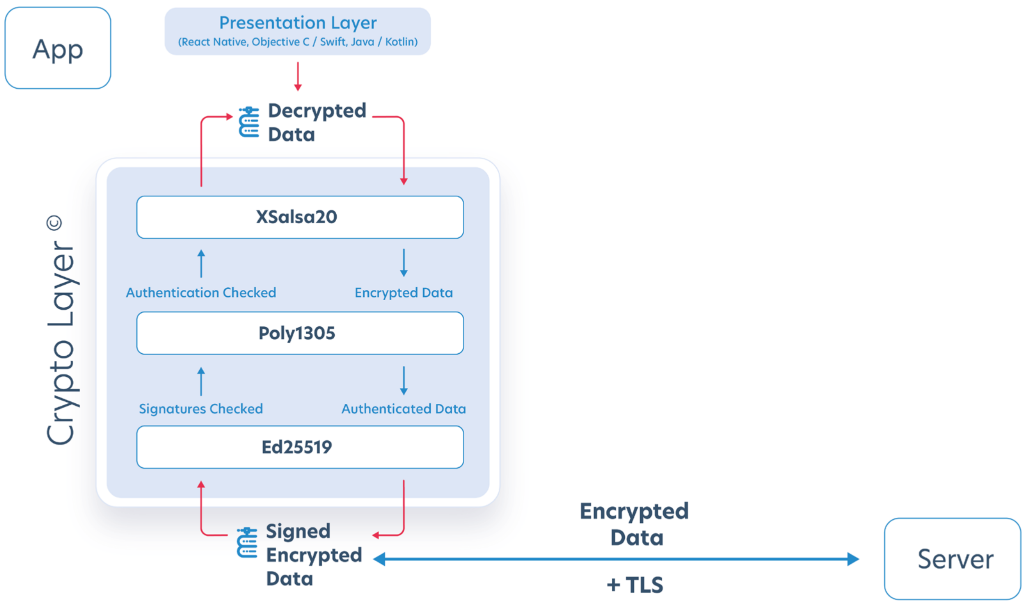


Figure 24 • Crypto Layer in App

As shown in Figure 24, signed encrypted data is first verified against the associated signature (*Ed25519*). Then, the data undergoes integrity checking using the authenticated encryption algorithm (*Poly1305*) and then decrypted with a 256-bit key (*XSalsa20*).

For data encryption, these steps are performed in reverse.

The selection of the *crypto_box* function and its associated algorithms is intended to optimise the usability and security of the Sharekey system.

Data Encryption & Integrity

Sharekey uses the *XSalsa20-Poly1305* Authenticated Encryption and Associated Data (AEAD) Algorithm. This simultaneously performs data encryption/decryption and generates/verifies a MAC for data integrity validation. Under the hood, *XSalsa20-Poly1305* is built from the *XSalsa20* and *Poly1305* open-source algorithms.

XSalsa20

The *XSalsa20* stream cipher is based upon the *Salsa20* algorithm but uses a longer nonce (192 bits vs. 64 bits). The use of a longer nonce makes it possible to use a weaker pseudorandom number generator (*PRNG*) with the same key without the risk of reusing the same key and nonce for multiple messages.

XSalsa20 uses a 256-bit Secret Key plus the first 128 bits of the nonce to generate a unique round key for each of *XSalsa20's* twenty rounds of encryption. This use of a 256-bit key makes *XSalsa20* **better** or at least equivalent in security to AES and other encryption algorithms used for protection of classified or sensitive information. Currently, no known attack is capable of breaking the full 20-round version of *XSalsa20*.

Poly1305

Poly1305 uses an encryption algorithm (*Xsalsa20* in NaCl), an additional key, and a random nonce to generate a MAC. Generating a MAC based upon a received message and comparing it to the one included with the message enables the recipient to detect if the message has been modified or corrupted in transit.

The security of *Poly1305* is largely dependent upon the underlying encryption algorithm used. Because *Xsalsa20* is a strong encryption algorithm, NaCl's implementation of *Poly1305* is strong as well.

Digital Signatures

For digital signatures, Sharekey uses NaCl's implementation of *Ed25519* (*edDSA* with *Curve25519*).

Ed25519

The use of *Ed25519* provides a number of different advantages:

- **Fast Computations:** *Ed25519* offers fast signature creation and verification.
- **Low Power Consumption:** Elliptic curve algorithms have lower power consumption compared to RSA signatures or DSA/Schnorr with multiplicative groups.
- **Low Data Usage:** *Ed25519* signatures are only 512 bits, and encryption keys are only 256 bits.
- **High Security:** *Ed25519* has a 2^{128} security target.
- **Side-Channel Resistance:** *Ed25519* allows for efficient implementations without data-dependent branching or other variable-time operation that would make it vulnerable to timing attacks.

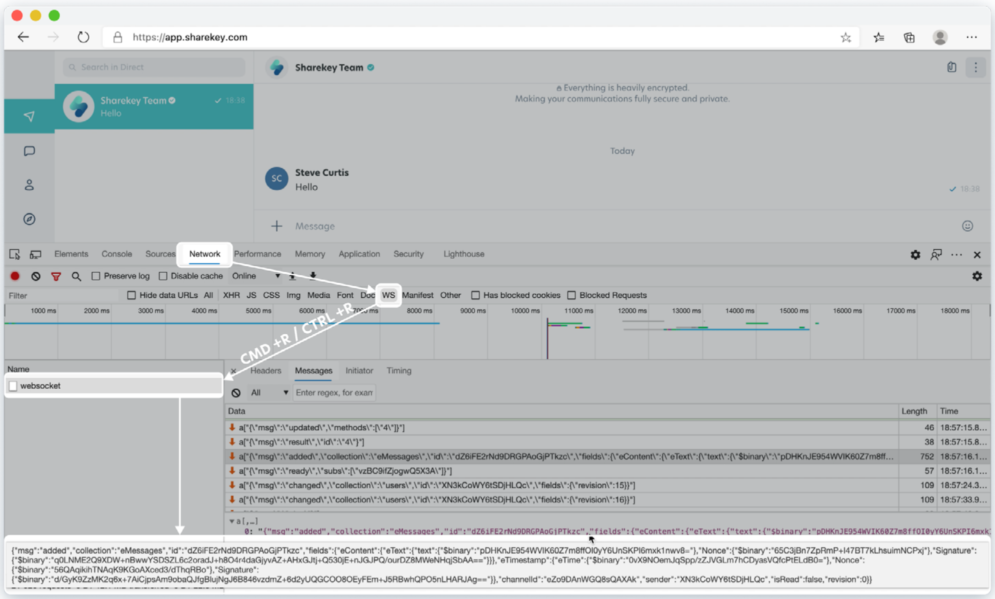
Crypto Layer Output

This section shows the output / outcome of the encryption done by the Crypto Layer.

View Encrypted Data

On a Web Application, a user can view this signed encrypted data by following these steps:

- Go to app.sharekey.com and sign-in
- Open Development Tool on your Browser by pressing F12
- Go to the Network tab and select WS
- Refresh the page by pressing cmd + R or CTRL + R
- Select "websocket" on the left under Name and select the "Messages" tab, as illustrated in the Figure 25.



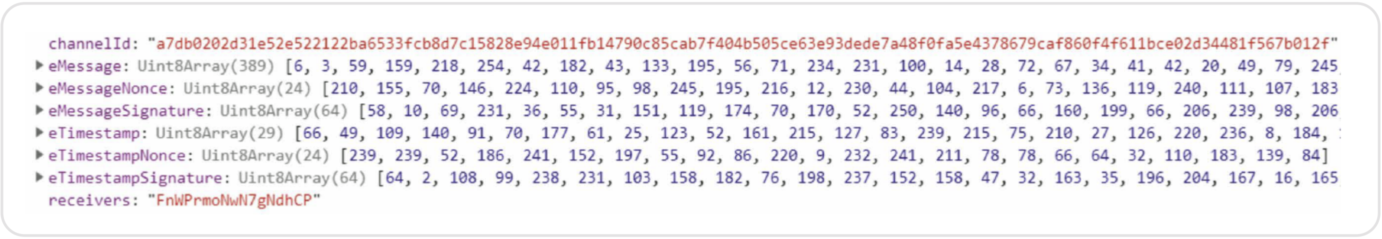
```
Data                                                                                                      Length  Time
a["{\"msg\":\"updated\",\"methods\":[\"4\"]}"]                                                                46  18:57:15.8...
a["{\"msg\":\"result\",\"id\":\"4\"}"]                                                                        38  18:57:15.8...
a["{\"msg\":\"added\",\"collection\":\"eMessages\",\"id\":\"dZ6iFE2rNd9DRGPAoGjPTkzc\",\"fields\":{\"eContent\":{\"eText\":{\"text\":{\"$binary\":\"pDHKnJE954WVIK60Z7m8ff...   752  18:57:16.1...
a["{\"msg\":\"ready\",\"subs\":[\"vzBC9ifZjogwQ5X3A\"]}"]                                                      57  18:57:16.1...
a["{\"msg\":\"changed\",\"collection\":\"users\",\"id\":\"XN3kCoWY6tSDjHLQc\",\"fields\":{\"revision\":15}}"]    109  18:57:24.3...
a["{\"msg\":\"changed\",\"collection\":\"users\",\"id\":\"XN3kCoWY6tSDjHLQc\",\"fields\":{\"revision\":16}}"]    109  18:57:33.9...
```

```
{"msg":"added","collection":"eMessages","id":"dZ6iFE2rNd9DRGPAoGjPTkzc","fields":{"eContent":{"eText":{"text":{"$binary":"pDHKnJE954WVIK60Z7m8ffOl0yY6UnSKPl6mxk1nwv8="},"Nonce":{"$binary":"65C3jBn7ZpRmP+I47BT7kLhsuimNCPxj"},"Signature":
{"$binary":"q0LNME2Q9XDW+nBwwYSDSZL6c2oradJ+h8O4r4daGjyvAZ+AHxGJtj+Q530jE+nJGJPQ/ourDZ8MWeNHqjSbAA=="}}},"eTimestamp":{"eTime":{"$binary":"0vX9NOemJqSpp/zZJVGLm7hCDyasVQfcPtELdB0="},"Nonce":
{"$binary":"56QAqikihTNAqK9KGoAXced3/dThqRBo"},"Signature":
{"$binary":"d/GyK9ZzMK2q6x+7AiCjpsAm9obaQJfgBlujNgJ6B846vzdmZ+6d2yUQGCOO8OEyFEm+J5RBwhQPO5nLHARJAg=="}},"channelId":"eZo9DAnWGQ8sQAXAk","sender":"XN3kCoWY6tSDjHLQc","isRead":false,"revision":0}}
```

Figure 25 • How to view Encrypted Data of the Crypto Layer

Signed Encrypted Data

Figure 26 shows an example of the signed and encrypted data in a Direct Message before it is further encrypted by TLS on its way to the server.

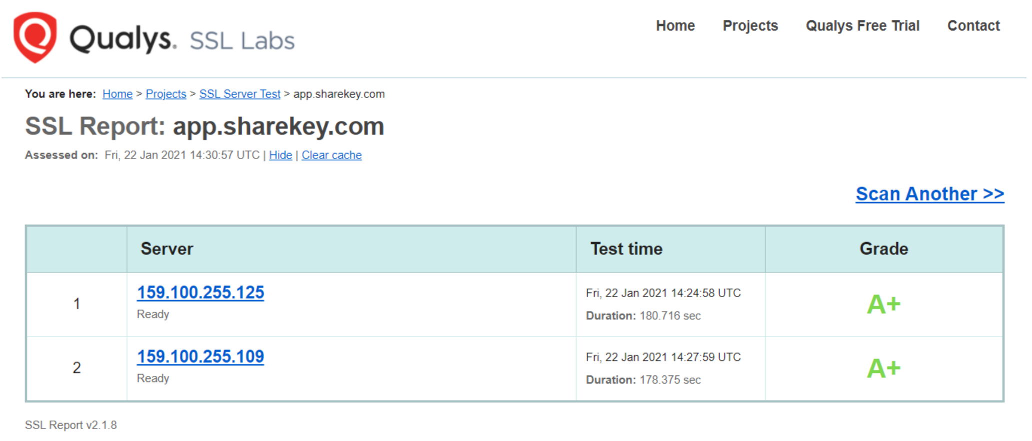


```
  channelId: "a7db0202d31e52e522122ba6533fcb8d7c15828e94e011fb14790c85cab7f404b505ce63e93dede7a48f0fa5e4378679caf860f4f611bce02d34481f567b012f"
▸ eMessage: Uint8Array(389) [6, 3, 59, 159, 218, 254, 42, 182, 43, 133, 195, 56, 71, 234, 231, 100, 14, 28, 72, 67, 34, 41, 42, 20, 49, 79, 245
▸ eMessageNonce: Uint8Array(24) [210, 155, 70, 146, 224, 110, 95, 98, 245, 195, 216, 12, 230, 44, 104, 217, 6, 73, 136, 119, 240, 111, 107, 183
▸ eMessageSignature: Uint8Array(64) [58, 10, 69, 231, 36, 55, 31, 151, 119, 174, 70, 170, 52, 250, 140, 96, 66, 160, 199, 66, 206, 239, 98, 206
▸ eTimestamp: Uint8Array(29) [66, 49, 109, 140, 91, 70, 177, 61, 25, 123, 52, 161, 215, 127, 83, 239, 215, 75, 210, 27, 126, 220, 236, 8, 184,
▸ eTimestampNonce: Uint8Array(24) [239, 239, 52, 186, 241, 152, 197, 55, 92, 86, 220, 9, 232, 241, 211, 78, 78, 66, 64, 32, 110, 183, 139, 84]
▸ eTimestampSignature: Uint8Array(64) [64, 2, 108, 99, 238, 231, 103, 158, 182, 76, 198, 237, 152, 158, 47, 32, 163, 35, 196, 204, 167, 16, 165
  receivers: "FnWPrmoNwN7gNdhCP"
```

Figure 26 • An example Output of the Crypto Layer

Transport Layer Security

In addition to encrypting each message with a Shared Key, Sharekey also uses Transport Layer Security (TLS) through the use of TLS v1.2 and 1.3.



You are here: Home > Projects > SSL Server Test > app.sharekey.com

SSL Report: app.sharekey.com

Assessed on: Fri, 22 Jan 2021 14:30:57 UTC | Hide | Clear cache

Scan Another >>

	Server	Test time	Grade
1	159.100.255.125 Ready	Fri, 22 Jan 2021 14:24:58 UTC Duration: 180.716 sec	A+
2	159.100.255.109 Ready	Fri, 22 Jan 2021 14:27:59 UTC Duration: 178.375 sec	A+

SSL Report v2.1.8

Figure 27 • Qualys SSL Labs Report

According to SSL Lab's Qualys' benchmarking, a screenshot of which is shown in Figure 27, Sharekey's web server configuration (for server to app communications) meets the requirements necessary to achieve an A+ grade.

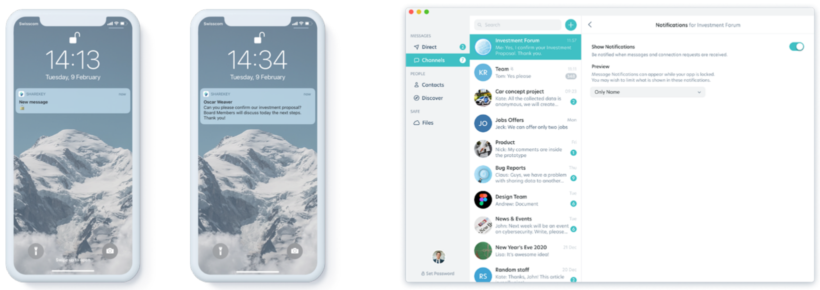
The use of TLS provides a number of security benefits, including:

- Preventing traffic analysis;
- Hiding Sharekey metadata;
- Protection against Man-in-the-Middle attacks;
- Preserving conversation integrity.

Notifications

Sharekey's mobile and desktop apps have the ability to notify a user about new content. Users have the ability to configure the app to show notifications for all messages or only those for specific Direct Messages or Channels. Users can choose how much information to reveal with the notification, such as the name or simply that some form of shared content (message, file, folder, etc.) is waiting for them. This is shown in Figure 28, for a mobile screen on the left and a desktop at the right.

Notifications do not reveal any sensitive information about the message, including contents, Channel names, etc. In fact, Sharekey doesn't provide any information about why the notification is appearing because the data needed to do so is encrypted until the user opens the app. After the application is opened and the user authenticates, the actual message is presented to the user.



New message

Oscar Weaver
Can you please confirm our investment proposal? Board Members will discuss today the next steps. Thank you!

Investment Forum 11:57 — Me: Yes, I confirm your Investment Proposal. Thank you.
Team 11:11 — Tom: Yes please
Car concept project 09:23 — Kate: All the collected data is anonymous, we will create...
Jobs Offers Mon — Jeck: We can offer only two jobs
Product Fri — Nick: My comments are inside the prototype
Bug Reports Thu — Claus: Guys, we have a problem with sharing data to another...
Design Team Tue — Andrew: Document
News & Events Tue — John: Next week will be an event on cybersecurity. Write, please...
New Year's Eve 2020 21 Dec — Lisa: It's awesome idea!
Random staff 20 Dec — Kate: Thanks, John! This article

Notifications for Investment Forum

Show Notifications
Be notified when messages and connection requests are received.

Preview
Message Notifications can appear while your app is locked.
You may wish to limit what is shown in these notifications.
Only Name

Figure 28 • Encrypted Notifications

Infrastructure

Sharekey supports caching recent messages and stored data on the device. Moreover, a complete history of a user's communications and stored files and folders is also stored encrypted on Sharekey's servers. This section provides an overview of Sharekey's server-side infrastructure.

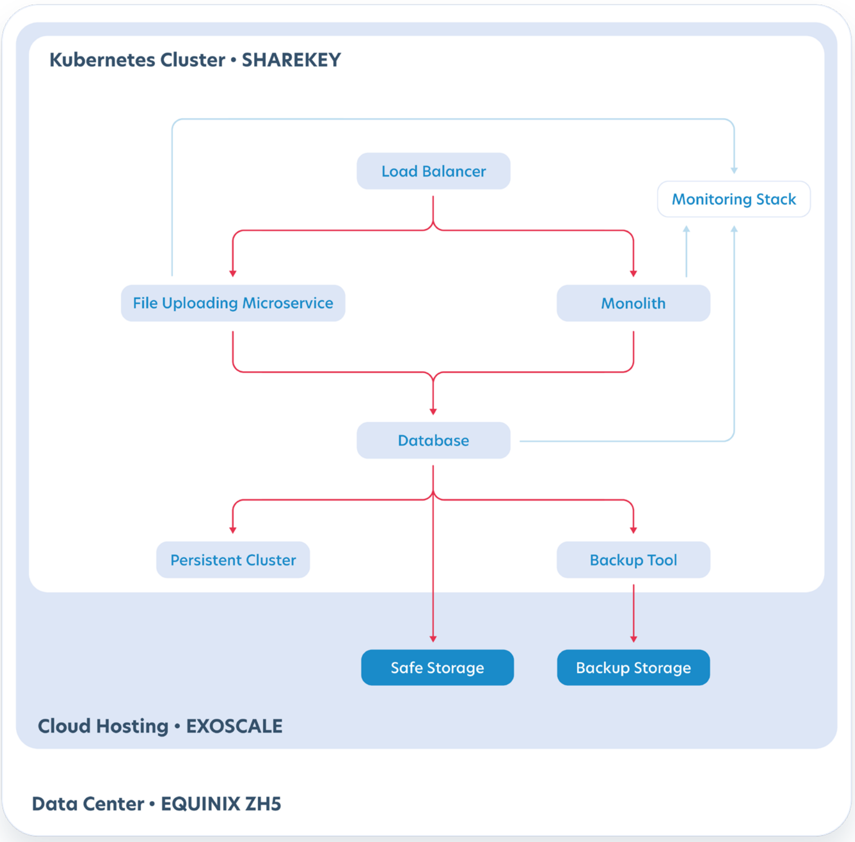


Figure 29 • Sharekey Infrastructure

Figure 29 provides a high-level view of Sharekey's server-side infrastructure. Some key elements of this infrastructure are described below.

Data Hosting Infrastructure

Sharekey has partnered with two Swiss service providers for its data hosting infrastructure:

- **Equinix:** Sharekey's primary datacenter is provided by Equinix [8]. Equinix provides high-performance data storage and uses 100% renewable energy. Equinix holds a number of different certifications, including ISO27001, PCI DSS, SOC-1 Type II, and others, enabling it to meet all of its customers' regulatory compliance requirements.

- **Exoscale:** Exoscale is Sharekey's hosting provider [9]. It operates multiple Swiss data centers and holds a number of certifications, such as ISO 27001, SOC-1 and SOC-2 Type II, PCI-DSS, and more. Its operations run completely on renewable energy and it provides full redundancy of hosted content to ensure business continuity.

Kubernetes Cluster

To enable high availability and easy transitions between data centers, Sharekey's backend infrastructure is implemented as a Kubernetes Cluster spread over multiple different physical instances. Additionally, the Sharekey infrastructure is described as code using Terraform and uses Helm to deploy software on the Kubernetes Cluster. This makes it infrastructure-agnostic and allows it to be rapidly deployed in a new environment if required.

The key elements of a Sharekey Kubernetes Cluster are described below.

Load Balancer

Sharekey operates multiple services instances. The Load Balancer service routes ingress requests from a client using a round-robin algorithm to any available target service instance that is ready to accept requests.

File Uploading Microservice

The file-uploader is a microservice which receive uploading requests, checks the integrity of the data uploaded by verifying its signature, before uploading onto Exoscale Simple Object Storage (SOS), a S3-compatible object storage provided by Exoscale. Sharekey securely stores all encrypted files and users' profile photo (also called avatars) on Exoscale SOS. Sharekey checks the integrity of the data received from clients to avoid MitM attacks.

Monolith

This contains backend code for the "full stack" app, written using the Meteor.js framework. The monolith is packed into a container image and runs on a cluster to obtain scalability and redundancy.

Database

A MongoDB is used for user data storage. It provides enhanced performance for our use case and, combined with ability to quickly evolve application logic, this is one of most important parts of our app. Thanks to ability to achieve quick horizontal scaling and ACID transactions, we able to meet both data safety and high performance even during the busiest hours of app usage.

Backup Tool

This is a critical service, as Sharekey cares deeply about maintaining its customers' data. To this end, Sharekey has created some self-maintained code for its backup solution, which runs as Kubernetes jobs, several times per day.

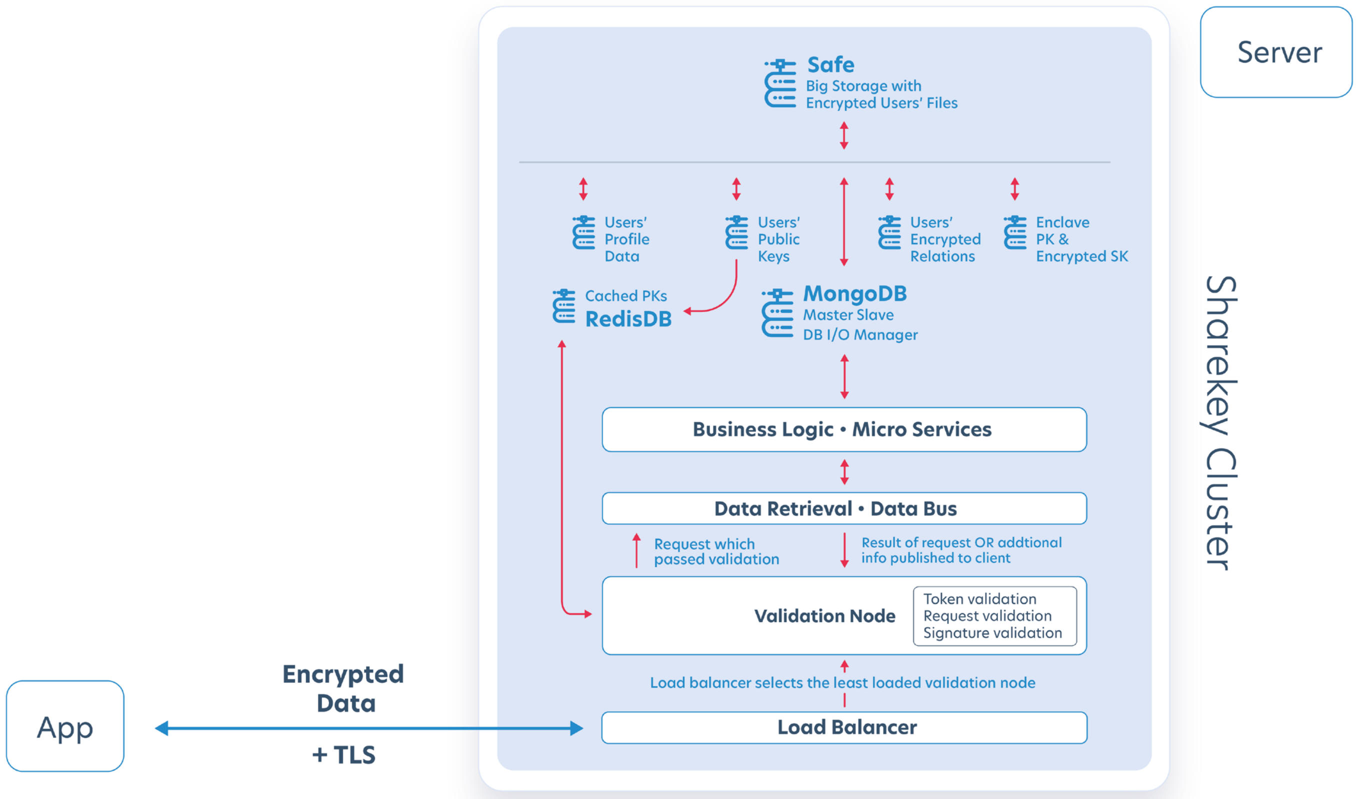
Sharekey retains database backups for two weeks, after which the database backups are stored externally on Exoscale SOS and therefore delegates backup storage maintenance to Exoscale.

Persistent Cluster

To provide fault-tolerant data storage, Sharekey operates an open-source network storage platform – Ceph. Ceph replicates data and makes it fault-tolerant, using commodity hardware and requiring no specific hardware support. As a result of its design, the system is both self-healing and self-managing, minimising administration and other operational expenses.

User Data Processing

Sharekey's back-end infrastructure is primarily implemented using Meteor.js. Data storage is provided by MongoDB with a RedisDB instance used to cache Public Keys requested within the last half hour.



Server
Safe
Big Storage with Encrypted Users' Files
Users' Profile Data
Users' Public Keys
Users' Encrypted Relations
Enclave PK & Encrypted SK
Cached PKs
RedisDB
MongoDB
Master Slave DB I/O Manager
Sharekey Cluster
Business Logic • Micro Services
Data Retrieval • Data Bus
Request which passed validation
Result of request OR addtional info published to client
Validation Node
Token validation
Request validation
Signature validation
Load balancer selects the least loaded validation node
Load Balancer
App
Encrypted Data
+ TLS

Figure 30 • Data Processing Architecture

Sharekey uses a WebSocket protocol for communications, enabling a single session to be maintained between the server and the app as long as the user remains online.

Figure 30 illustrates the standard flow of a request from a Sharekey client application to the server:

1. **Load Balancer:** Sharekey operates multiple different validation nodes within its data center. The Load Balancer identifies and forwards a request to the validation node with the highest available capacity.
2. **Validation Node:** The validation node is responsible for validating three components of a request:
    a. **Token:** Tokens generated using *CSPRNG* on the server are validated when a Sharekey app makes a connection to the server. This token is required for an app to resume a connection to the server. Tokens have a length of 43 characters and entropy of 6 bits per character, resulting in 256 bits of entropy which translates to a high level of session security for our users.

    b. **Request:** At this stage, the server verifies that the message is well-formed and matches the protocol definition.
    c. **Signature:** The validation node requests the sender's Public Key from the RedisDB cache, which forwards the request to MongoDB if necessary. This key is then used to validate the digital signature associated with the request.
3. **Data Retrieval:** A valid request is transmitted via the data bus to a computation cluster node. This node performs a request from MongoDB for any of the following types of data:
    a. **User Private Data and Files:** A user's data is stored encrypted within Sharekey's servers. It can be downloaded upon request and decrypted by the client application when used.
    b. **User Public Keys:** Sharekey's back-end infrastructure is primarily implemented using Meteor.js. Data storage is provided by MongoDB with a RedisDB instance used to cache Public Keys requested within the last half hour.
    c. **People:** Sharekey does not monitor the traffic on connections between users but stores the information on connections between users in its encrypted database. This enables a client app to download and decrypt a user's list of private contacts after sign-in.
    d. **Enclave:** Enclave stores a copy of a user's Public Key and encrypted Master Secret Key. This can be used to set up or restore a Sharekey application on a device.
4. **Response:** After a request has been made, the result of the request or any additional information generated flows back through the stack and to the client.

It is important to note that the only information stored unencrypted on Sharekey's servers are those provided by users as a part of their public profile and are the following:

1. A list of users' full names, profile photos (also called avatars), and Public Keys, the last named being essential when creating new connections between users.
2. Users' email addresses to send 2FA codes and notify them about attempted logins to their account.

All other data is stored encrypted and can only be decrypted by a client application once a user has authenticated to it.

At no point in time does Sharekey have access to a user's data or Secret Keys.

Testing & Certification

Code Audits

Sharekey's code has been audited several times by a noted French security expert. Additional audits are planned as new features are added to the code base.

Audit reports can be provided upon request.

Penetration Tests

Sharekey was subject to several penetration (pen) tests based on the OWASP Framework. Additional frameworks will be added in the future.

The reports of these pen tests can be provided upon request.

Bounty Programs

Sharekey plans to introduce a bounty program that will encourage white hat hackers and other security experts to conduct penetration and other tests to verify the security of our solution.

Certifications

In 2021, Sharekey will seek the first level of security certification (called CSPN) from the French National Cybersecurity Agency, ANSSI [13].

In the near future, it will also pursue ISO 27001 certification.

Source Code

Sharekey is committed to transparency and providing its customers with control over their data. For this reason, Sharekey will license its code as open-source in future.

Currently, some of Sharekey's code is available on its public GitHub repository at https://github.com/sharekey. Over time, an increasing amount of code will be available for review at that location as features currently under development approach their final state.

Sharekey's algorithms and protocols for various cryptographic functions are all open source, and its own code is limited to the assembling and reuse of such publicly available material.

Sharekey welcomes an audit of the available code by security experts to verify that it indeed fulfils what is asserted – especially its key feature that the Master Secret Phrase used for all encryption never leaves a device and remains fully under a user's control.

Website

Sharekey's website (sharekey.com) is designed to provide information about the Sharekey application and download links for the desktop and mobile versions. It is not connected to Sharekey's backend servers and is based upon a serverless architecture inspired by JAMstack. This serverless architecture means that there is no backend infrastructure to maintain or attack via this site.

The advantage of using JAMstack over legacy three-tier web architectures is to avoid the processing overhead of running server-side code and creating dynamically-generated pages for every request. As Figure 31 shows, JAMstack relies on pre-built, static pages which are cached in edge CDNs for quicker responsiveness. Moreover, static pages are read-only and have no executable code to exploit. Also, the server-side infrastructure does not rely on APIs to external sources (e.g., a Content Management System), further reducing the attack surface.

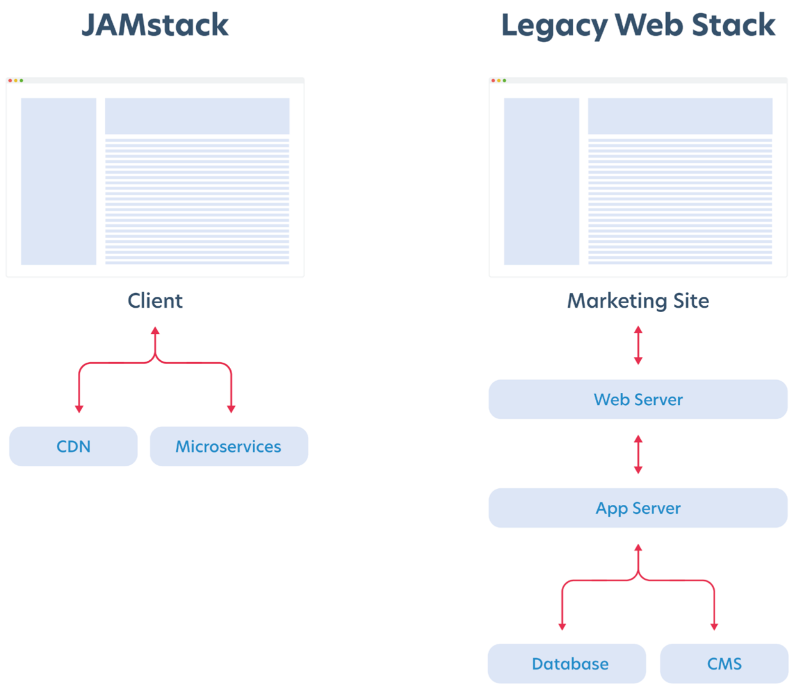


Figure 31 • JAMstack Architecture vs. Legacy Web Architecture

The Sharekey web application (app.sharekey.com) is accessible from the main landing page. To be clear, though, this web application is not buit using JAMStack. The Sharekey web app and desktop application have identical code (the desktop app is implemented using Electron) and both store encryption keys and user data in the same secure way.

Roadmap

Sharekey is a new and evolving technology. The Sharekey development roadmap includes expanded functionality, reduced metadata collection, security improvements, more granular control of shared data, and support for on-premises data storage.

Expanded Functionality

Currently, Sharekey offers direct and group messaging and data storage functionality. The Sharekey roadmap includes integrating a number of new features into its secure application:

- **Call:** The call feature will also provide support for end-to-end encrypted voice calls.
- **Video:** The video feature will add encrypted video conferencing functionality.
- **Calendar:** The calendar feature will provide a secure and integrated solution for schedule management.
- **Workspace:** The workspace feature will allow users to organise contacts, conversations, and shared data based upon projects and teams.

In addition to these new features, Sharekey also plans to add additional functionality to enhance usability and security for its users, such as:

- **Sync with Local Drive:** The development of a secure storage solution for desktops that enables files and folders on the device to be automatically synced with Sharekey and be searchable on the computer.
- **Desktop Offline Mode:** Currently, files and folders stored locally on desktop devices are not encrypted. Sharekey plans to add a local encrypted database to provide the same level of security on desktops as is available on mobile devices, especially for offline access.
- **Single Sign-On (SSO):** Sharekey plans to provide an Active Directory integration to support Single Sign-On (SSO), integrating it with an organisation's existing password management system.

Metadata

Currently, Sharekey has access to some user metadata needed to provide its services. This includes:

- **Connections:** Within a Channel, Sharekey currently receives metadata regarding the creator of the Channel (called the Owner), the name of the Channel and a pseudonymised list of Members.
- **Files and Folders:** Sharekey processes some metadata, including the creator of a file or folder (called the Owner), the pseudonymised list of Members, and the role of each Member.

On the whole, Sharekey receives and processes less metadata than similar communications apps. The roadmap includes efforts to reduce the metadata collected by Sharekey by eliminating most of these privacy-sensitive types of metadata. The plan is to enable Sharekey to reduce its metadata collection to the minimum required to operate the service.

Increased Security

Security and user data privacy are at the core of the Sharekey ecosystem. Sharekey plans to introduce a number of new security features:

- **Certificate Pinning:** Sharekey plans to implement certificate pinning for its applications [12]. This helps to protect against Man-in-the-Middle (MitM) attacks by forcing the client to verify that the certificate presented by the Sharekey server matches a copy stored locally on the client.
- **Out-of-Band Identity Verification:** Sharekey plans to implement a double-factor connection system where a user can exchange another verifier (such as a one-time passcode) out-of-band to verify that an exchanged Public Key actually belongs to the other user. This protects the user against Man-in-the-Middle (MitM) attacks if the Sharekey infrastructure is compromised.
- **2-Factor Authentication (2FA):** Attempted sign-in to a Sharekey account from a new device will require authorisation from one of the user's other devices.
- **Hardware-Based Key Storage:** Sharekey plans to integrate the use of hardware-based two-factor authentication (2FA) solutions, such as Yubikey. This will enable the Master Secret Key to be stored off the device, increasing the difficulty for an attacker attempting to gain access to a user's data.
- **Post-Quantum Cryptography:** Sharekey plans to introduce support for post-quantum cryptographic algorithms. These algorithms are designed to be resistant to quantum computing, providing long-term privacy and security protections for user data.
- **Formal Verification:** Sharekey plans to use a formal verification tool (such as Verifpal) to verify the correctness of its protocols. This will help provide security assurance in terms of confidentiality and integrity.

Sharing Control

Currently, as described earlier, Sharekey users can provide other users with full access to shared files or folders and remove this access if desired. In the future, users will have more granular control and be able to manage the exact rights and permissions a specific user should have on a specific shared file or folder.

Additionally, Sharekey users will have a number of different control settings that they can configure for messages and shared content, including:

Timers

- Self-Destruct: Set a message to delete itself a fixed amount of time after opening.
- Expiry: Set a message to delete itself on a certain date.
- Schedule: Set a message to only be sent after a certain date.

Rights Management

- No Reshare: Disable the recipient's ability to share the message or content with others.
- No Copy: Disable copy/paste for the message.
- No Download: Disable downloading of the shared content.
- No Print: Disable printing of the shared content.
- No Screenshot: Disallow screenshots of the message or shared content.

Access

- **Face/Touch ID:** Require the use of Face/Touch ID for authentication before viewing a message or shared content.
- **Password:** Require the entry of a password to access the content.

Infrastructure & Data Storage

Currently, a Sharekey user's messages and files are stored encrypted on Sharekey's back-end infrastructure. In the future, Sharekey will offer the option for enterprises to deploy their own on-premise infrastructure for employees to store corporate data.

Additionally, Sharekey plans to provide access to a redundant data center through a partnership with Mount 10 [10]. Mount 10 operates a datacenter located 1000 meters inside of a Swiss mountain. The planned redundant datacenter will use Snowberg's hosting services [11].

Conclusion

It is worth reiterating what makes Sharekey different from other communication platforms.

Sharekey was designed for business-to-business communications, specifically to guarantee confidential communication and data sharing both within and across organisations. With the digital transformation of corporate environments and the rise of remote working as the “New Normal”, the attack surface for cyber threats has increased dramatically. Thus, a business needs to be able to fully trust the collaboration tools and platforms that it uses.

And this is particularly critical for communication at the highest levels of enterprises, such as between CxOs and their staff, both within and across enterprise boundaries. Just imagine the serious consequences of a leak surrounding a planned merger, a yet-to-be-released financial report, plans for a key hire, or a contract under negotiation. That’s why we have created Sharekey, a Swiss communication and collaboration app with the highest level of security and privacy built for Business Executives. At the same time, the solution is also useful within an entire enterprise as a secure collaboration tool.

Sharekey is built on a foundation of strong encryption. Sharekey goes beyond end-to-end encryption of the communication channel — the full app containing all your data is heavily encrypted with your own key. This is called “App-to-App Encryption” — and it’s the future of data security. When you send a document using Sharekey, only your business counterparts can decrypt it within their Sharekey app. Nobody else can access your messages, files and folders nor track your interactions — not even Sharekey employees — because the encryption key never leaves your device and encrypts everything, including privacy-sensitive metadata.

Sharekey has created a platform where users can be confident of the privacy of their communication, and one that is immune to many of the cybersecurity and privacy threats that plague other collaboration platforms.

References

1. The scrypt Password-Based Key Derivation Function
https://tools.ietf.org/html/draft-josefsson-scrypt-kdf-05

2. How strong is 256-bit Encryption?
https://www.thesslstore.com/blog/what-is-256-bit-encryption/

3. NIST Special Publication 800-63B • Digital Identity Guidelines
https://pages.nist.gov/800-63-3/sp800-63b.html

4. Sharekey Privacy Policy
https://sharekey.com/legal/privacy-policy.pdf

5. Diffie-Hellman Key Agreement Method
https://tools.ietf.org/html/rfc2631

6. Public-key authenticated encryption: crypto_box
https://nacl.cr.yp.to/box.html

7. tweetnacl-js
https://github.com/dchest/tweetnacl-js

8. Equinix ZH5 (Zurich)
https://www.equinix.ch/locations/europe-colocation/switzerland-colocation/zurich-data-centers/zh5/

9. Exoscale • Akenes SA
https://www.exoscale.com/

10. Mount 10 AG
https://www.mount10.ch/en/home/

11. Snowberg Solutions AG
https://snowberg.ch/

12. Owasp • Certificate and Public Key Pinning
https://owasp.org/www-community/controls/Certificate_and_Public_Key_Pinning

13. ANSSI • Certification CSPN
https://www.ssi.gouv.fr/administration/produits-certifies/cspn/

Acronyms

DSS	Data Security Standard
EV	Extended Validation
MAC	Message Authentication Code
MitM	Man in the Middle
OWASP	Open Web Application Security Project
PCI	Payment Card Industry
SSL	Secure Sockets Layer
SOC	Security Operation Center
SSO	Single Sign-On
TLS	Transport Layer Security
2FA	Two Factor Authentication

Glossary

2FA	**Two-Factor Authentication (2FA)** is a technique by which a computer user is granted access to a resource only after successfully presenting two pieces of evidence (or factors) to the authentication mechanism: knowledge (something only the user knows, such as a password) and possession (something only the user has, such as a readout from a physical token).
Certificate Pinning	**Certificate Pinning** forces a client app to validate a server's X.509 certificate (containing, among other things, its Public Key) received during connection setup against a known copy stored on the client.
Channel	A one-to-many communication channel created by a Sharekey Member (called Owner) for encrypted group communication.
CSPRNG	A **Cryptographically Secure Pseudorandom Number Generator** (*CSPRNG*) is a pseudorandom number generator (*PRNG*) with properties that make it suitable for use in cryptography.
Diffie-Hellman key agreement protocol	A method for securely deriving a common cryptographic key to encrypt a communication channel.
Direct Messaging (DM)	An encrypted one-to-one communication channel created between two Sharekey Members.
Ed25519	*Ed25519* is a public-key signature system carefully engineered at several levels of design and implementation to achieve very high speeds without compromising security.
EV certificate	An **Extended Validation** (EV) PKI certificate provides the strongest level of identity assurance, and issued only after extensive vetting of an organisation's credentials.
libsodium	The **Sodium crypto library (libsodium)** is a modern, easy-to-use software library for encryption, decryption, signatures, password hashing and more.
MAC	A **Message Authentication Code (MAC)** is a small piece of data appended to a message to prove its authenticity, i.e., that the message comes from a trusted source, and has not been altered in transit.
Man-in-the-Middle	In cryptography and computer security, a **Man-in-the-Middle (MitM)** attack is a cyberattack where the attacker secretly eavesdrops and possibly alters the communications between two parties who believe that they are directly communicating with each other over a private connection.

Master Secret Key	The private key of a private/public key pair derived during the initial account sign-up process, and used when securely connecting with other users and to protect the other keys used for data protection and authentication throughout the Sharekey ecosystem.
Master Signing Key	The private part of another private/public key pair derived during the initial account set-up process on a new device, and which is used for creating digital signatures.
Meteor.js	**Meteor.js** is an open-source framework for building and deploying Web, Mobile, and Desktop applications in *Javascript*.
MongoDB	**MongoDB** is a source-available, cross-platform document-oriented database.
NaCl	**Networking and Cryptography Library (NaCl)** (pronounced "salt") is a public domain, high-speed software library for network communication, encryption, decryption, signatures, etc.
Passcode	A numerical code used on mobile devices to gain access to the Sharekey app.
Password	An alphanumeric piece of text meeting NIST guidelines used to gain access to the Sharekey app on desktops.
Private Key	A part of a key pair that is kept secret by the Owner and used to encrypt data or decrypt data encrypted by another party with the corresponding Public Key.
PRNG	A **pseudorandom number generator** (*PRNG* is an algorithm for generating a sequence of numbers whose properties resemble those of a sequence of random numbers.
Public Key	A part of a key pair that is shared with other parties and used by them to encrypt data intended for the Owner of the key pair.
PCI DSS	The **Payment Card Industry Data Security Standard (PCI DSS)** is an information security standard for use by organisations that handle branded credit cards from the major card schemes.
Poly1305	**Poly1305** is a cryptographic Message Authentication Code (MAC) used to verify the data integrity and the authenticity of a message.
RedisDB	Redis is an open source (BSD licensed), in-memory data structure store, used as a database, cache, and message broker.
scrypt	*scrypt* (pronounced "ess crypt") is a password-based key derivation function specifically designed to make it costly to perform large-scale custom hardware attacks by requiring large amounts of memory.
Secret Key	See Master Secret Key

Secret Phrase	A random set of 24 words generated using the *scrypt* key derivation function during initial account sign-up to the Sharekey app. It is used as input to the derivation of the Master Secret Key and as way to set up a Sharekey app on a new device.
Shared Key	A key derived where one party's Master Secret Key and the other party's Public Key are combined using the Diffie-Hellman key exchange protocol to encrypt the communication between them.
Single Sign-On	**Single Sign-On (SSO)** is an authentication scheme that allows a user to log in once with a single set of authentication credentials (e.g., ID and Password) to any one of several related, yet independent, software systems and not have to re-enter those credentials at the other systems.
SAFE	A Sharekey user's storage space for files and folders on a Sharekey server. All stored data is encrypted with the user's Master Secret Key.
SOC 2	**SOC 2 (System and Organisation Controls 2)** is a type of audit report that attests to the trustworthiness of services provided by a service organisation. It is commonly used to assess the risks associated with outsourced software solutions that store customer data online. **SOC 2** compliance is a minimal requirement when considering a SaaS provider.
TLS	**Transport Layer Security (TLS)** is a set of cryptographic protocols designed to provide privacy and data integrity over the communication channel between two computer applications.
WebSocket	**WebSocket** is a computer communications protocol, providing a full-duplex communication channel over a single TCP connection.
XSalsa20	**XSalsa20** is a symmetric encryption key, called a stream cipher, where the same key is used for both encryption and decryption.

Privacy is the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others.

Alan Westin, Privacy and Freedom, 1967

Sharekey®

SHAREKEY Swiss AG
Gotthardstrasse 26
6300 Zug
Switzerland
security@sharekey.com