Whitepaper

Security in ginlo @work

7 June 2018

Contents

Management summary

ginlo @work is a collaboration platform for businesses that put data confidentiality first. Fully encrypted and hosted exclusively in Germany, it allows organizations to securely harness modern ways of working across departments and borders: ginlo @work can be used on any device to securely communicate, work on projects, and share data with other employees, partners, and customers, while keeping the company in control of the data.

This Whitepaper describes the security controls implemented in ginlo @work to keep customers' data secure. The focus lies on ginlo @work's innovative encryption concept with the following main benefits for organizations:

Full encryption, beyond end-to-end
ginlo @work relies on full encryption of all content at any time. Unlike many other solutions, it not only uses end-to-end encryption when sending messages, but also encrypts all content stored locally on the device.
(For details, see chapter "Data Transmission", page 11, and "Local encryption", page 12.)

Data ownership is with the company
We believe that companies should be able to have full sovereignty over all the content that circulates in their organizations. That is why, unlike most other encrypted business collaboration platforms, ginlo @work allows administrators to gain access to users' message encryption keys, e.g. for auditing purposes or to restore content after the loss of a device.
(For details, see chapter "Encryption Concept Overview", page 4.)

Provider shielding
Brabbler AG as the provider of ginlo @work never gains access to the encryption keys used in your organization. Unencrypted keys are only present on locally installed clients.
(For details, see chapter "Key Storage", page 11.)

Encryption

Encryption concept overview

Key escrow
For every user, a messaging key pair is generated when they register their account. However, this key pair is not only stored on the user's device, but also re-encrypted with a dedicated team identity encryption key pair and stored in a centralized team service that holds the keys in escrow [1]. The administrator has access to the team identity encryption key pair and thus to the users' messaging key pairs if required.
(For details, see chapter "Decryption of Messaging Key Pairs", page 7.)

Separation of identity key pairs
In addition to the messaging key pair, an identity encryption key pair and an identity signing key pair are also generated for every user. These key pairs will be used in upcoming versions to protect the user's identity against tampering. A backup containing all 3 key pairs is created and encrypted with a passphrase chosen by the user (see chapter "Key Backup", page 11); this allows users to restore their own keys. The backup is stored in the so-called metadata service that is held separately from the team service. Thus, the user's identity key pairs cannot be decrypted by the administrator.

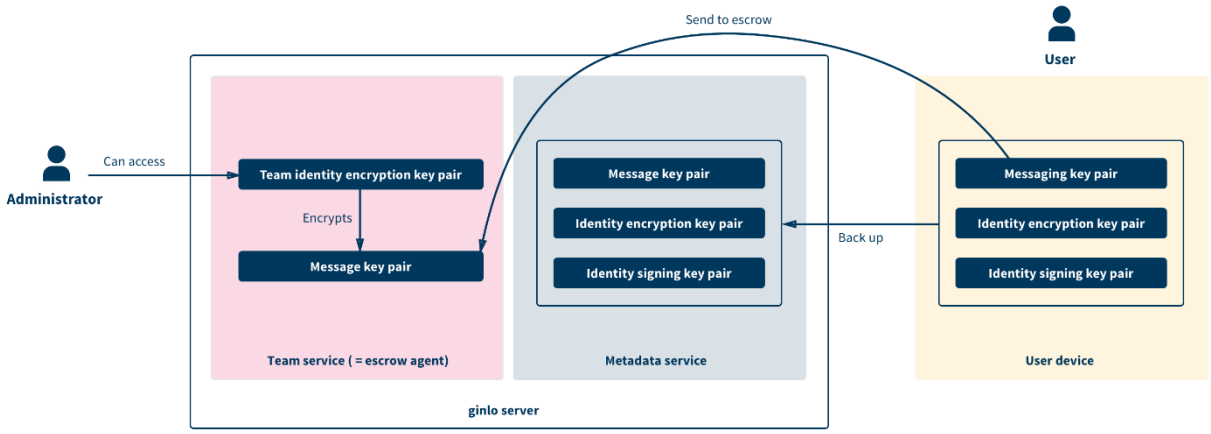

Figure 1: Handling of a user's key pairs in ginlo @work

Main processes in detail

Team registration

The registration of a team takes place in the desktop-based administration client “ginlo Team Manager” and includes the following steps:

Administrator keys

1. The following 3 key pairs are generated for the administrator:
   - **Messaging key pair**
   - **Identity encryption key pair**
   - **Identity signing key pair**
2. The 3 key pairs are backed up to the metadata service, protected with a key passphrase chosen by the administrator (see also chapter “Key Backup”, page 11). From this backup, the identity encryption key pair is retrieved whenever required to perform an administrative task. In addition, the administrator can use the backup to transfer the keys to the ginlo @work app on their smartphone or the desktop user client.

Team keys

3. The following team key pair is generated:
   - **Team identity encryption key pair**
4. The administrator’s messaging key pair is encrypted using the team identity encryption key pair and stored in the team service. This step is repeated for every new user (see also chapter “User Registration”, page 6).
5. The team identity encryption key pair is encrypted with the administrator’s identity encryption key pair and stored in the team service. This step is repeated for every new administrator (see also chapter “Administrator Enrolment”, page 8).

The following diagram shows the key setup after team registration: Using their key passphrase, the administrator can access their key backup. Using their identity encryption key pair, which is included in this backup, the administrator can decrypt the team identity encryption key pair and thus the messaging key pair in the metadata service.

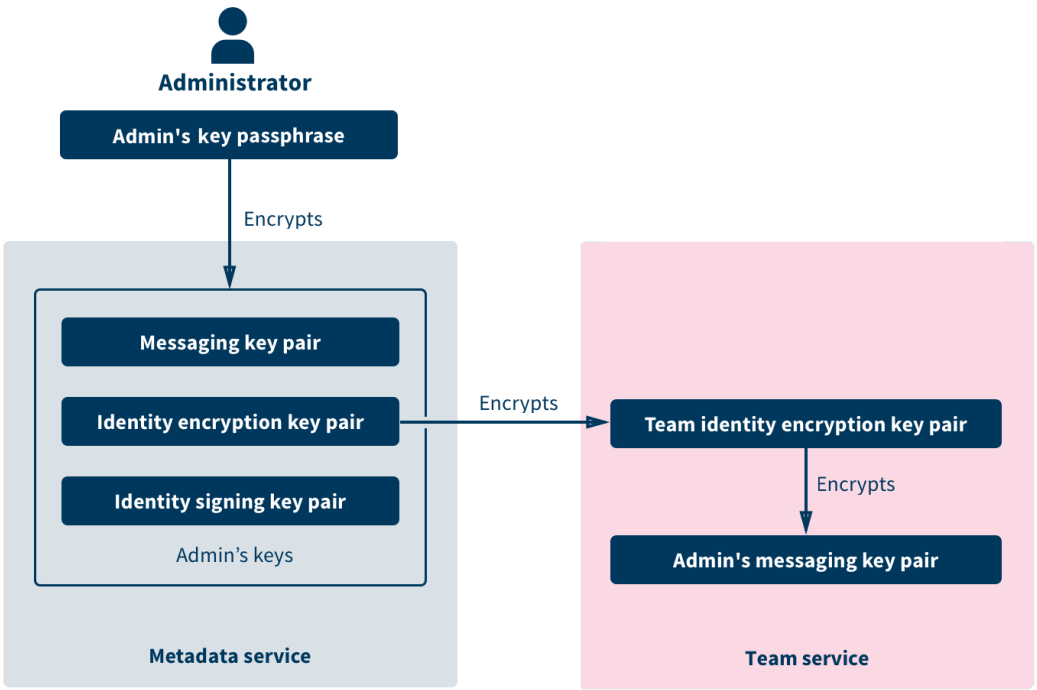

Figure 2: Key setup after team registration

User registration

The registration of a new user takes place in the ginlo @work app on their smartphone or the desktop user client and includes the following steps:

1. The following 3 key pairs are generated for the new user:
   - **Messaging key pair**
   - **Identity encryption key pair**
   - **Identity signing key pair**
2. The 3 key pairs can be backed up to the metadata service, protected with a key passphrase chosen by the user (see also chapter “Key Backup”, page 11). Via this backup, the user can transfer the keys to other devices.
3. The user’s messaging key pair is encrypted using the team identity encryption key pair and stored in the team service.

The following diagram shows the key setup after the registration of an additional user: It allows the administrator to access the user’s messaging key pair stored in the team service, but prevents them from accessing the user’s identity key pairs.

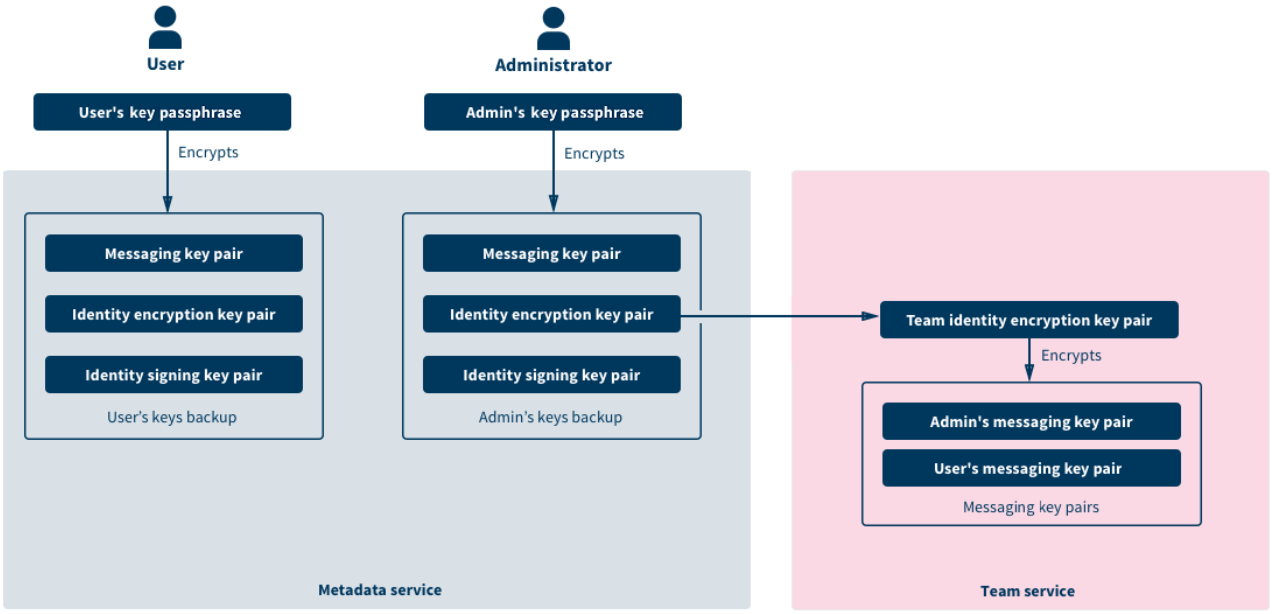

Figure 3: Key setup after team and user registration

Decryption of messaging key pairs

If required, the administrator can decrypt a user's messaging key pair stored in the team service, e.g. in the following situations:

- An audit of a user's content needs to be conducted.
- A user has left the company, and their content must be retrieved.
- A user has lost their device and forgot the passphrase protecting their key backup.

The process of decrypting a user's messaging key pair is as follows:

1. The administrator retrieves their identity encryption key pair from the metadata service and decrypts it in their local ginlo Team Manager installation using their key passphrase.
2. The administrator retrieves the encrypted team identity encryption key pair from the team service and decrypts it using their own identity encryption key pair.
3. The administrator retrieves the user's encrypted messaging key pair from the team service and decrypts it using the team identity encryption key pair.

The administrator can then download the encrypted archive of the user's messages and decrypt it locally.

The administrator can also transfer the message key pair to the user's new device if the user requests this, e.g. because they forgot the passphrase for their key backup. In this case, the following steps are added:

4. The user triggers the process on their new device by sending the admin a request to restore their messaging key pair.
5. This causes new identity key pairs to be generated on the user's new device. The new public keys are included in the request.
6. The administrator receives the request. If the administrator approves, the user's messaging key pair is re-encrypted with the user's new identity encryption public key.
7. The encrypted messaging key pair is transferred to the user's device.
8. The user can decrypt their messaging key pair and thus restore their content.
9. The new identity public keys are transferred to a dedicated users service where all users' identity public keys are stored. The new public keys replace the user's previous ones. From there, they are synced to the devices of all other users in the team.

As soon as the process is completed, all unencrypted keys are deleted from the ginlo Team Manager.

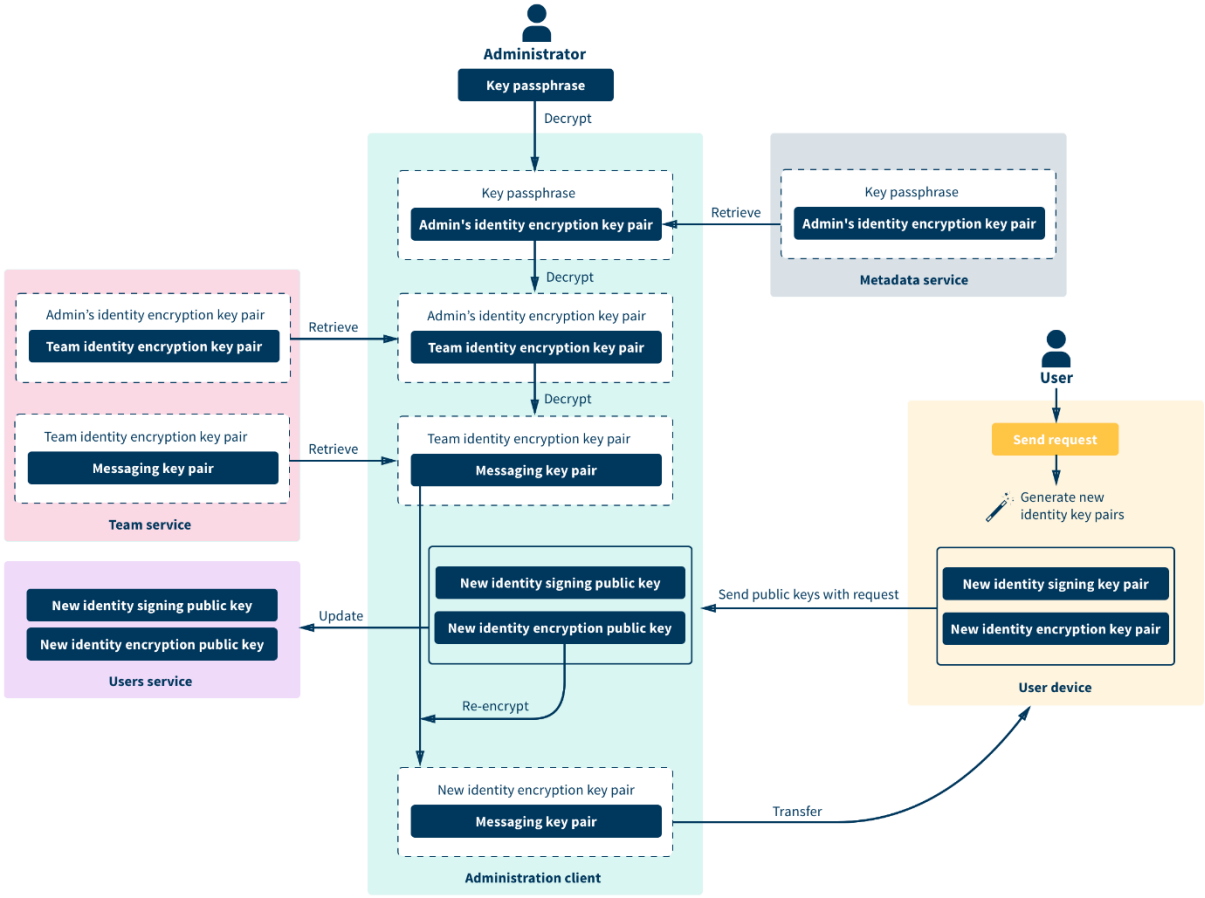

Figure 4: Decryption of a user's messaging key pair

To prevent misuse, the affected user is notified as soon as the administrator decrypts their messaging key pair.
For future versions, an audit trail is planned: Every action performed by an administrator will be logged and signed with that administrator's private key. This read-only log can be made available to other administrators for mutual control or (if there is only one administrator) to all users in the organization.

Administrator enrolment

In future versions, an existing administrator can give administrator rights to other users as follows:

1. The existing administrator retrieves their identity encryption key pair from the metadata service and decrypts it in their local ginlo Team Manager installation using their key passphrase.
2. The existing administrator retrieves the encrypted team identity encryption key pair from the team service and decrypts it using their own identity encryption key pair.
3. The existing administrator retrieves the new administrator's identity encryption public key stored in a dedicated users service.
4. The team identity encryption key pair is encrypted with all administrators' identity encryption key pairs using saltpack [2]. It can now be decrypted by any of the administrators.
5. This encrypted team identity encryption key pair is transferred to the team service. From there, the new administrator can retrieve it and decrypt it in their own ginlo Team Manager installation when this is required to perform an administrative task.

As soon as the process is completed, all unencrypted keys are deleted from the ginlo Team Manager.

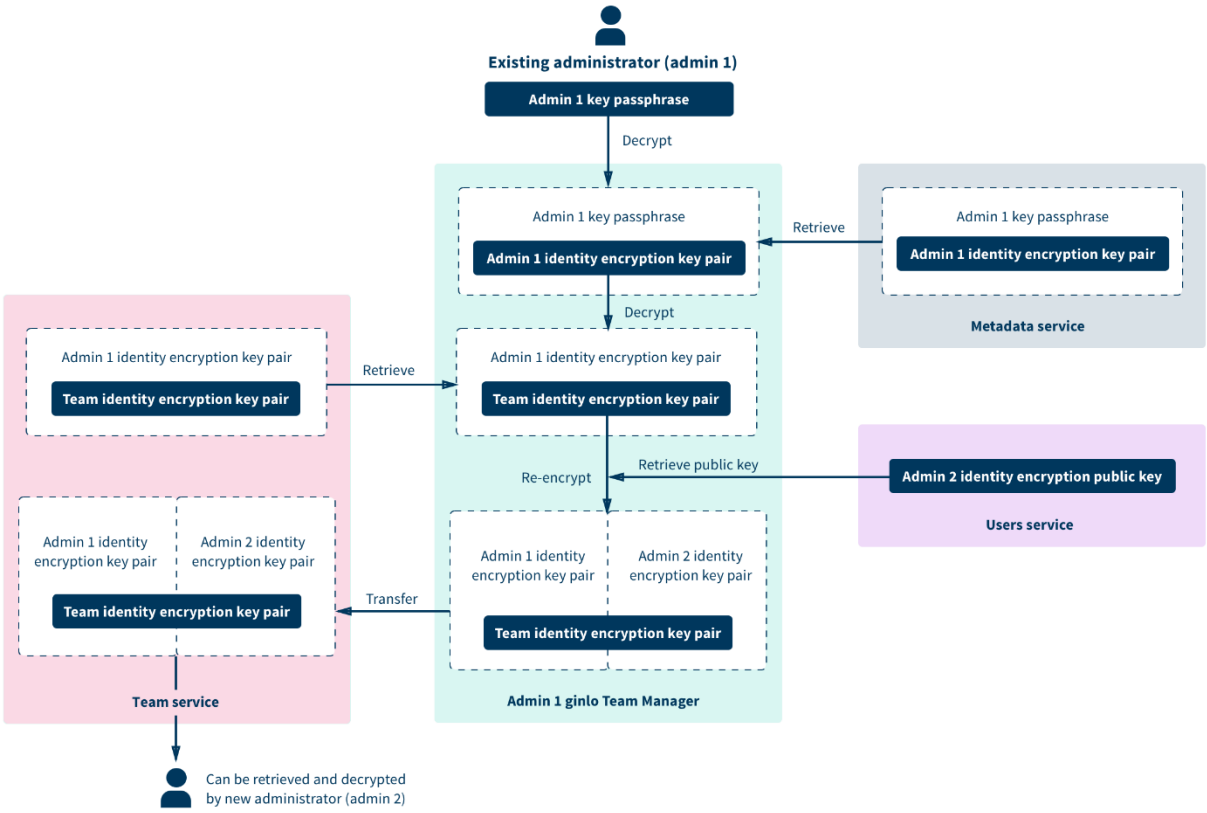

Figure 5: Enrolment of a new administrator

Administrator removal

A process for removing administrators is currently being developed.

Encryption keys overview

This section provides an overview of all encryption keys used for ginlo @work teams.

Key name	Purpose	Origin	Storage	Can be accessed by
Team identity encryption key pair	Re-encrypts the messaging key pairs of all users in the team	Team registration	▪ Team service	Administrator
User identity encryption key pair	Administrators use it to re-encrypt the team identity encryption key pair; used to provide a user with their restored messaging keys.	User registration	▪ User's device ▪ Metadata service (encrypted with user's key passphrase) ▪ Users service (public key only)	User
User identity signing key pair	Not used at the moment; will be used for signing messages in future versions.	User registration	▪ User's device ▪ Metadata service (encrypted with user's key passphrase)	User
User messaging key pair	Encrypts the user's messages. As more modules will be added (e.g. file sync and share), a dedicated key pair will be generated for each module.	User registration	▪ User's device ▪ Metadata service (encrypted with user's key passphrase) ▪ Team service (encrypted with team identity encryption key pair)	User and administrator

Table 1: Encryption keys overview

Key generation

All key pairs are asymmetric key pairs based on Elliptic Curve Cryptography (ECC) [3]. Every key pair consists of a public and a private key. Although the two keys of a pair are mathematically related, the private key cannot be calculated given only the public key. The private key is generated at random; the corresponding public key is calculated over the Elliptic Curve “Curve25519” [4]. This curve is generally recommended by security experts in contrast to NIST-recommended curves that have raised the community's concern regarding the generation and potential weaknesses [5].

Key storage

A user’s client stores all key pairs of a user. At rest, the keys are kept in an encrypted database (see also “Local Encryption”, page 12).
The administration client “ginlo Team Manager” does not store any keys. When keys are required for an administrative task, they are retrieved from the server, decrypted in the ginlo Team Manager, and deleted from the client as soon as the task is completed.
The team, metadata, and users services are stored on the ginlo server in an Apache Cassandra database management system [6]. It is encrypted at rest using Arx [7].

Key backup

A backup of an administrator’s keys is a mandatory step during registration: From this backup, the identity encryption key pair is retrieved whenever it is required to perform administrative tasks. In addition, the administrator can use the backup to transfer the keys to the ginlo @work app on their smartphone or the desktop user client.
For users, a key backup is optional, but highly recommended as it is needed to e.g. set up a new device after loss of the original one.

A backup is created as follows:

1. The user defines a key passphrase from which an AES-256 key [8] and an HMAC [9] key are derived. The method used for the key derivation is PBKDF2 [10] with HMACSHA512 using an iteration count of 50,000. Encryption of the backup is then carried out in CBC mode [11] with a random initial vector. The concatenated initial vector and the encrypted message are then MAC'ed with the HMAC key using HMAC SHA-256.
2. The encrypted backup is uploaded to the ginlo server.

Data transmission

Sent messages

The encryption of messages sent between ginlo users is based on the libsodium library [12]: The messages are encrypted individually using a random per-message key generated on sender side (using libsodium's crypto_secretbox_* API). This session key is encrypted with the recipient’s messaging public key (using libsodium's crypto_box_* API). For messages sent as part of a group conversation, the session key is re-encrypted using the pubic keys of all recipients.

Sent files

All files (i.e. images, videos, documents, and voice messages) sent as part of a ginlo conversation are encrypted on the sender's client using a random AES-256 key in CBC mode combined with a random initial vector. The concatenated initial vector and the encrypted message are then MAC'ed with a random HMAC key using HMAC SHA-256. A dedicated key is generated for every file and encrypted individually for each recipient with their messaging public key. Both the encrypted file and the file key are transferred to the server from where the recipient's client can retrieve it. Using the recipient's messaging private key, the client decrypts the file key and can then decrypt the file.

Transport encryption

ginlo clients and servers communicate over HTTPS connections supporting TLS. The server indicates the order preference of cipher suites and communicates HTTP Strict Transport Security [13] to the clients.

Local encryption

All encryption keys, messages, and contacts stored locally in a user's client are kept in a database encrypted using SQL Cipher AES-256 encryption [14] in CBC mode. The encryption key for this database is a random byte sequence generated during client setup on that device. The encryption key is encrypted with a key that is derived from a random code that is used as input to PBKDF2 with an iteration count of 10,000. The first 32 bytes of the output serve verification purposes while the following 32 bytes are used as the key for the SQLCipher.

All files (i.e. images, videos, documents, and voice messages) sent as part of a ginlo conversation are AES-256-encrypted in CBC mode with a random per-file key combined with a random initial vector. The encrypted file is stored in the device's internal storage.

Account registration

E-mail address verification

To register a ginlo @work team, the administrator must provide their e-mail address. The administrator must also provide the e-mail addresses of all users they want to invite to their team. In both cases, the specified e-mail address must be verified within 7 days to complete registration.

For that purpose, an e-mail with a random verification code $c \in_R [10^5, 10^6 - 1]$ is sent to the given e-mail address. The client has a total of 3 attempts to send the correct verification code to the server. After 3 failed attempts, the registration process must be started again.

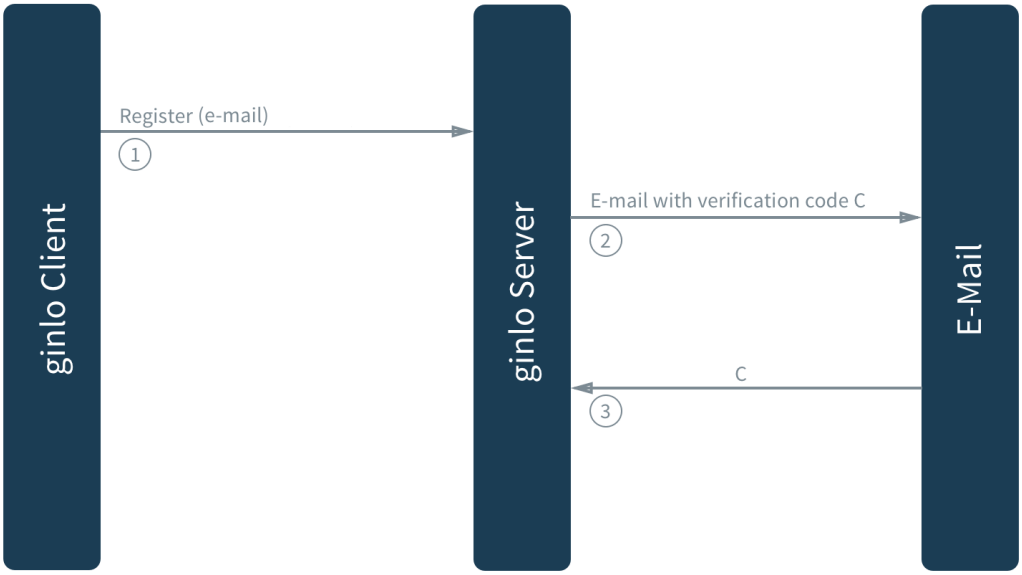

Figure 6: E-mail address verification

Account password

Users are required to define a password for their account. While this password is not stored on the client, the ginlo server stores the password's hash value, which is created as follows:
A SHA-512 hash [15] is generated and rehashed using the *bcrypt* password hashing function [16] with a strength of 11 and a per-user salt. This *bcrypt* hash is encrypted with a per-password AES-256 key in GCM mode [17]. That password-protecting encryption key is encrypted with a key that never leaves the encryption service.

Authentication

Tokens

Authentication is based on OAuth2 [18] with JWT access and refresh tokens [19]. The initial login generates a token of each type. The access token authenticates requests to protected API resources and is valid for 10 minutes. It is also bound to the device on which the user logged in. Besides, the encrypted IP address is stored together with the access token to allow for the detection of misuse (e.g. concurrent connections to the same account from different continents). When the access token expires, the client uses the refresh token to generate a new access token [20]. A refresh token expires after 30 days, i.e. if a user does not access the client for 30 days, they are required to log in again for the client to receive a new token pair. Every ginlo user account can have a maximum of 30 refresh tokens, i.e. a user can be logged in to ginlo on up to 30 devices at the same time; however, the solution is optimized for a maximum of 5 devices per user.

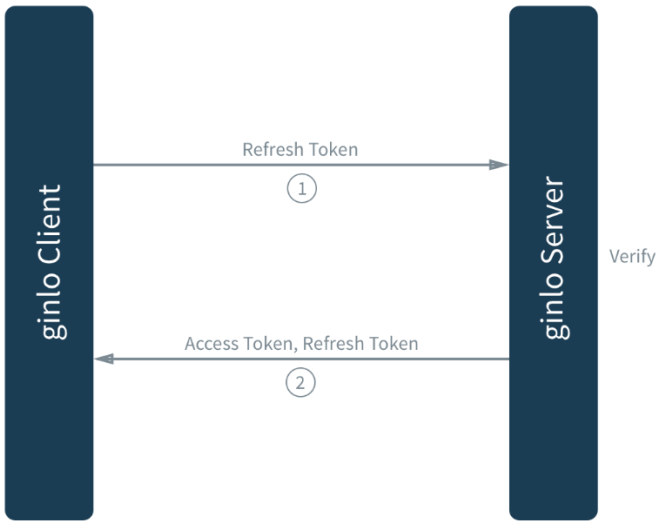

Figure 7: Token refresh

Account password reset

Forgotten account passwords can be reset as follows: The user specifies their registered e-mail address in a web form of the customer portal. The user then receives an e-mail with a link. Clicking on this link directs them to a form where they can set a new account password.

The verification procedure is identical to the verification during account creation (see chapter “Account Registration” on page 13) except for the following differences:

- There can be only 1 pending password reset per account at a time.
- The password reset process must be completed within 24 hours.

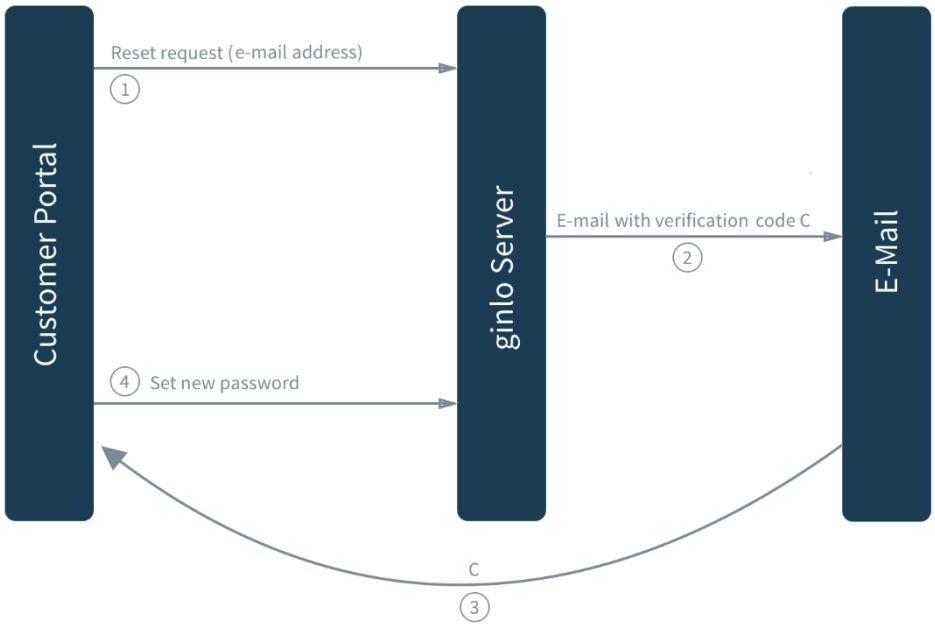

Figure 8: Account password reset

Passcode

To protect access to their app, users can optionally set a passcode, either as part of the registration or later in the settings. It can either be a simple, 4-digit passcode, or a complex one that must consist of 4 or more alphanumerical characters. The passcode serves to unlock ginlo when the app is started after it was closed by the user or the system's memory management. It can be changed or disabled at any time. If the passcode is set as part of the registration, it is additionally used to encrypt the key of the local database (see also chapter “Local Encryption” on page 12). Setting or changing the passcode after the registration does not have any influence on the encryption.

Notifications

ginlo currently provides push notifications over GCM [21] and APNs [22]. The content of the related ginlo message is not included in the notification. To receive a notification, a client needs to register push tokens during account registration or when it is added to an existing account:

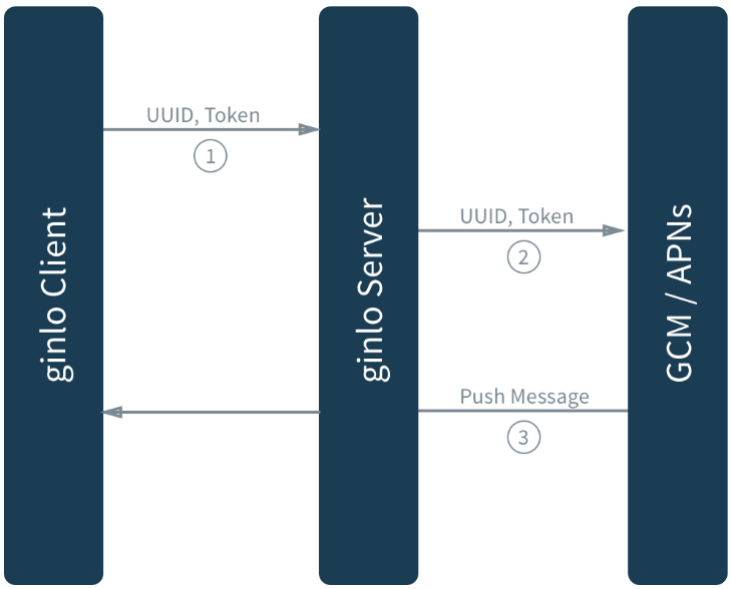

Figure 9: Push Token Registration

Contact

ginlo @work is currently in an early phase. Thus, its encryption concept as well as its security controls in general are constantly developed further. If you would like to contribute your thoughts and experience, or if you have any questions, please get in touch with us:

Brabbler Secure Message and Data Exchange AG
– IT Security Department –
Ria-Burkei-Straße 26
D-81249 München

E-mail: security@ginlo.net

References

[1] https://en.wikipedia.org/wiki/Key_escrow
[2] https://saltpack.org
[3] https://www.imperialviolet.org/2010/12/04/ecc.html
[4] https://cr.yp.to/ecdh.html
[5] https://tools.ietf.org/html/rfc7748
[6] https://cassandra.apache.org
[7] https://eprint.iacr.org/2016/591.pdf
[8] https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
[9] https://tools.ietf.org/html/rfc2104
[10] https://www.ietf.org/rfc/rfc2898.txt.pdf
[11] https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Cipher_Block_Chaining_.28CBC.29
[12] https://download.libsodium.org/doc/
[13] https://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security
[14] https://github.com/sqlcipher/sqlcipher
[15] https://en.wikipedia.org/wiki/SHA-2
[16] http://bcrypt.sourceforge.net
[17] https://en.wikipedia.org/wiki/Galois/Counter_Mode
[18] https://oauth.net/2/
[19] https://auth0.com/docs/jwt
[20] https://auth0.com/docs/tokens/refresh-token
[21] https://developers.google.com/cloud-messaging
[22] https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html