QABEL ARCHITECTURE

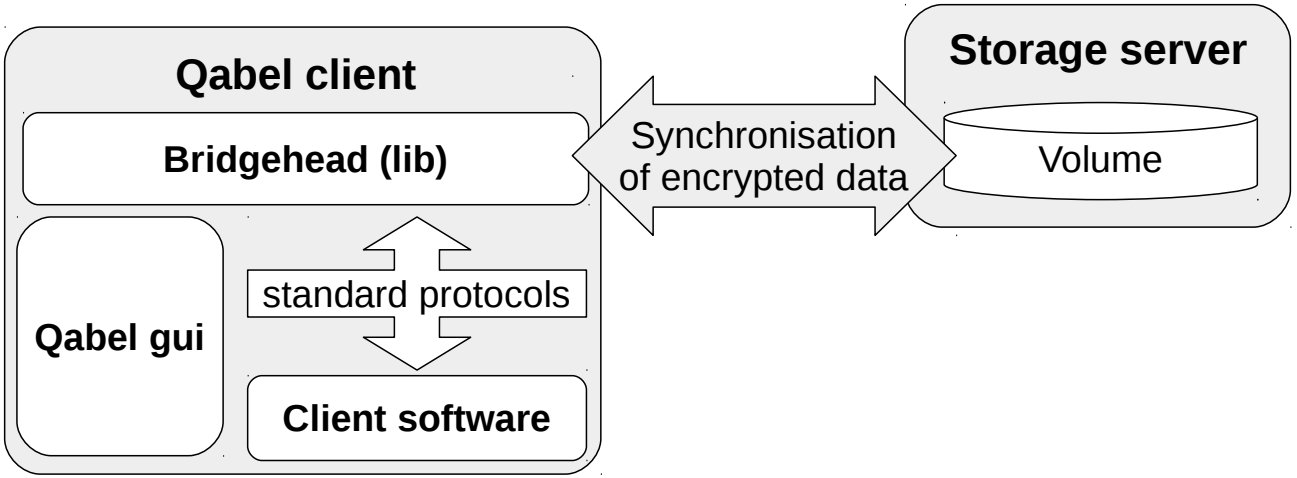
doc v0.1 – 2014-06-11

BASIC CONCEPT

- Qabel is a free, open source, expandable software platform that makes communication and data exchange on the internet as technically safe as possible and thus easy to use.
- Any data is encrypted end-to-end
- Cryptography completely on the client side
  - Key stays with the user
  - Neither the hoster nor anyone else is given the technical opportunity to read or change the user's data or behaviour.
- Metadata is either prevented or largely obscured
- Qabel can depict complex services such as:
  - shared calendars
  - collaborative office
  - social networks,...

FAT CLIENT

- User data is encrypted on the client side
  - That's why all of the data processing has to be done on the client.
  - Therefore, the „bridgehead“ implements standard protocols (e.g. IMAP) to be able to use widely known end user client software (e.g. Thunderbird).
  - Only encrypted data will leave the client and is saved on a “storage server”.
  - The encryption key stays in the hands of the user.



DROP SERVER

- Messaging is done via a „drop server“
  - Everybody can write and read messages on the drop server.
  - Messages are encrypted asymmetrically with the public key of the receiving account and posted on the drop server.
  - A client listens to one or many drop servers for messages, but can only decrypt the ones determined for him.

USE CASE: ALICE SHARES A FILE WITH BOB

- Alice klicks: „file“ → „share with“ → „Bob“.
  - Alice splits the file in many small blobs,
  - generates a random key and encrypts each blob with this key,
  - uploads the encrypted data to a storage server,
  - generates a message to Bob, consisting of the encryption key and a url to download the blobs from the storage, and encrypts this message with the public key of Bob,
  - posts this message to one of the drop servers she knows Bob is listening to.
- Bob gets notification from his Qabel Client, klicks „accept file“.
  - Bob polls on one or many drop servers for messages,
  - tries to decrypt incoming messages with his private key.
  - If successful, does things. In this case: processes the encryption key and file url,
  - downloads data, decrypts data with shared key, packs data back into a file.

METADATA ON STORAGE SERVER

- No authentication for read-access needed
  - Everyone can read encrypted data from everyone else, but does not necessarily have a decryption key.
  - So, the storage server does not know who can read data from whom (if clients hide behind TOR for read access).
  - Relevant meta data: none.
- Authentication for write-access
  - Needed for quota-based accounting
  - Every user has a full amount of storage space provided
  - Qabel client will write random data to obfuscate real data
  - Relevant meta data: timestamps of write access (probably by an identifyable user)

METADATA ON DROP SERVER

- No authentication needed at all
- Everyone can and potentially will read everything
- Users have many drop ids to be adressed by
- Users can hide behind TOR
- Relevant meta data: none.