Database encryption

Approach

Using SQLCipher - it is a drop in replacement for SQLite that works for non-encrypted databases without any changes (TODO test on iOS/Android).

`direct-sqlite` and `sqlite-simple` libraries are forked and renamed to `direct-sqlcipher` and `sqlcipher-simple`, with replaced cbits in `direct-sqlcipher` (TODO include SQLCipher as git submodule with a script to upgrade cbits).

While SQLCipher provides additional C functions to set and change database key, they do not necessarily need to be exported as they are available as PRAGMAs.

Moving from plaintext to encrypted database (and back) requires migration process using sqlcipher_export() function.

The approach would be similar to database migration for the notifications:

1. the current users will be offered to migrate to encrypted database once, with a notice that it can be done later via settings.
2. the new users will be asked to enter a pass-phrase to create a new database (it can be empty, in which case the database won't be encrypted).
3. during the migration the database backup will be created and the old database files will be preserved - in case of the app failing to open the new database right after the migration it should revert to using the previous database.

When opening the database the key must be passed via chat command / agent configuration, some test query must be performed to check that the key is correct: https://www.zetetic.net/sqlcipher/sqlcipher-api/#PRAGMA_key

Options to support in chat settings:

- encrypt database (with automatic rollback in case of failure)
- decrypt database (-"-)
- change key (using PRAGMA rekey)