# Password manager



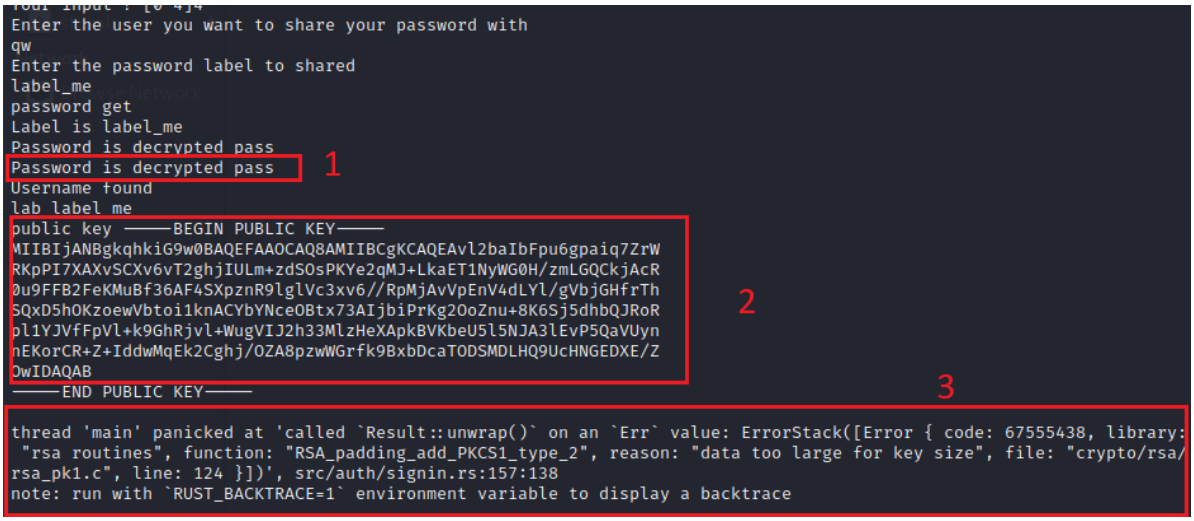
## Known bugs

The feature `Shared password` does not work.

| Step | Description | Work |
|---|---|---|
| 1 | Recover the password corresponding to the label with the user's private key | Yes |
| 2 | Retrieve target user's public key | Yes |
| 3 | Encrypt the password with the target user's public key | No |

Example

In this example, some information is displayed for debugging purposes. This is of course test data



# Description of program

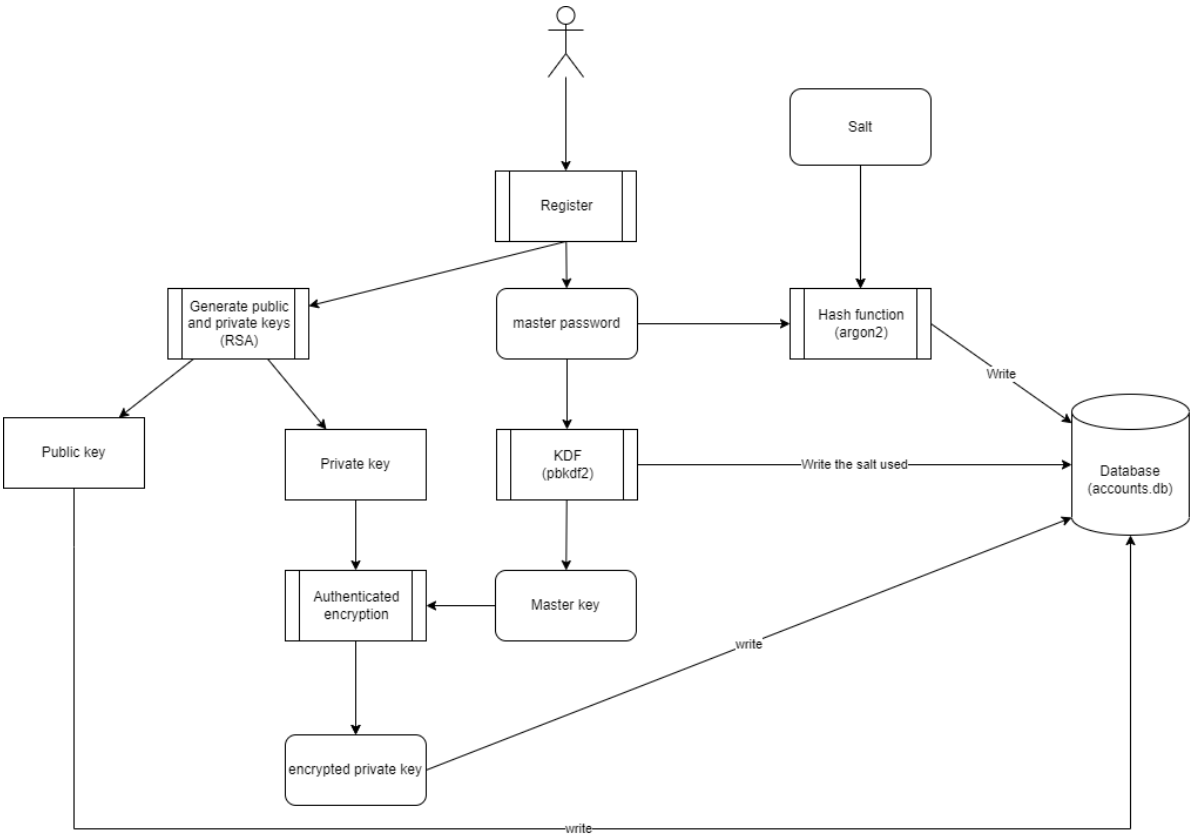
The user creates an account on the application. An RSA key pair is created. and the RSA private key is encrypted with the key derived from the master password.

The hash of the master password is stored in the database (accounts.db).

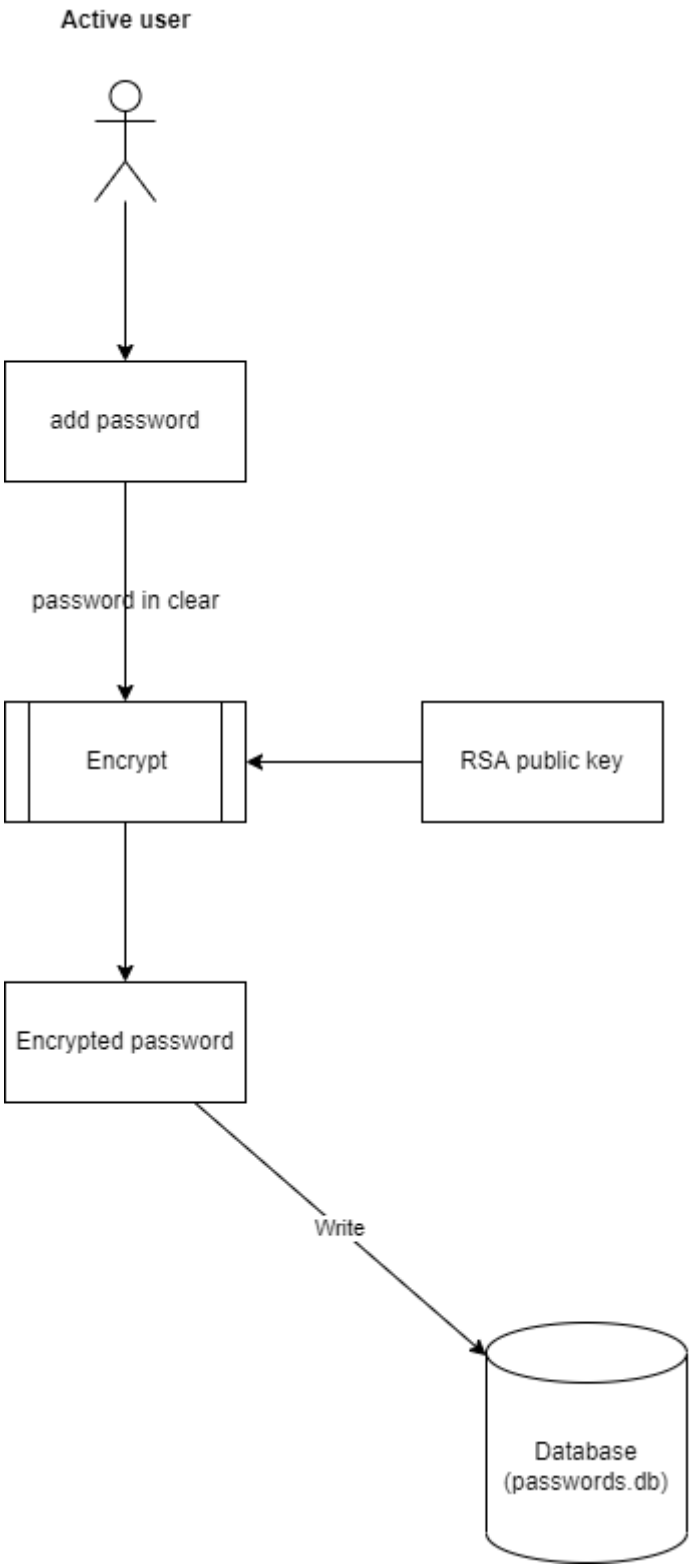
When the user adds a new password to the database (passwords.db), it is encrypted with the RSA public key.

When  a user wishes to share a password with another user, he also uses the public key.

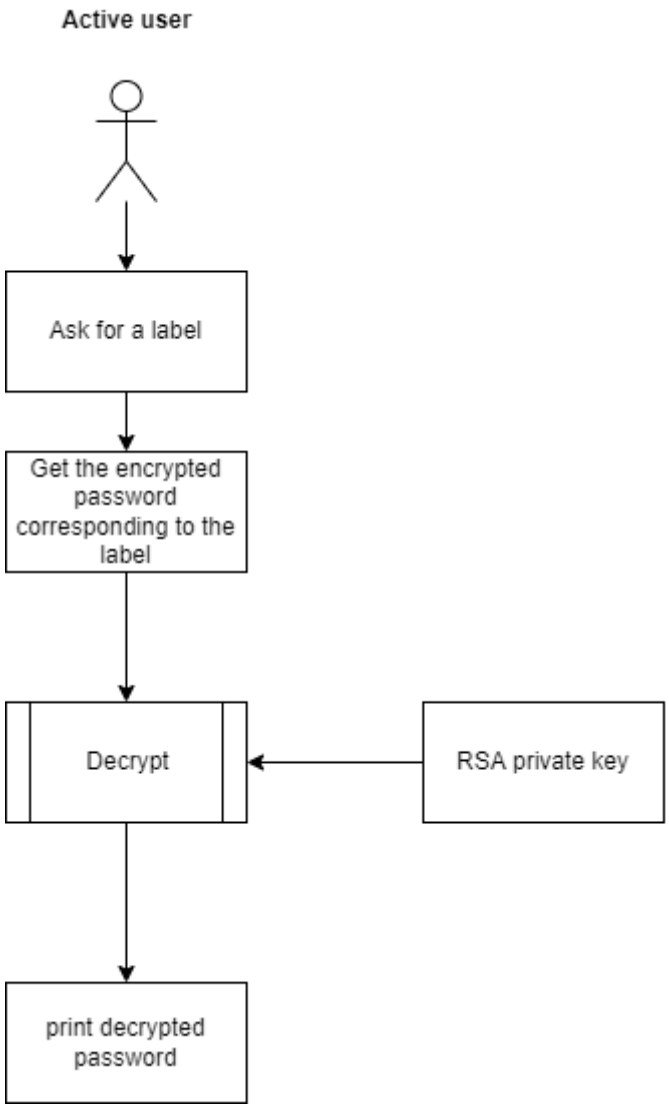
## Creating an account



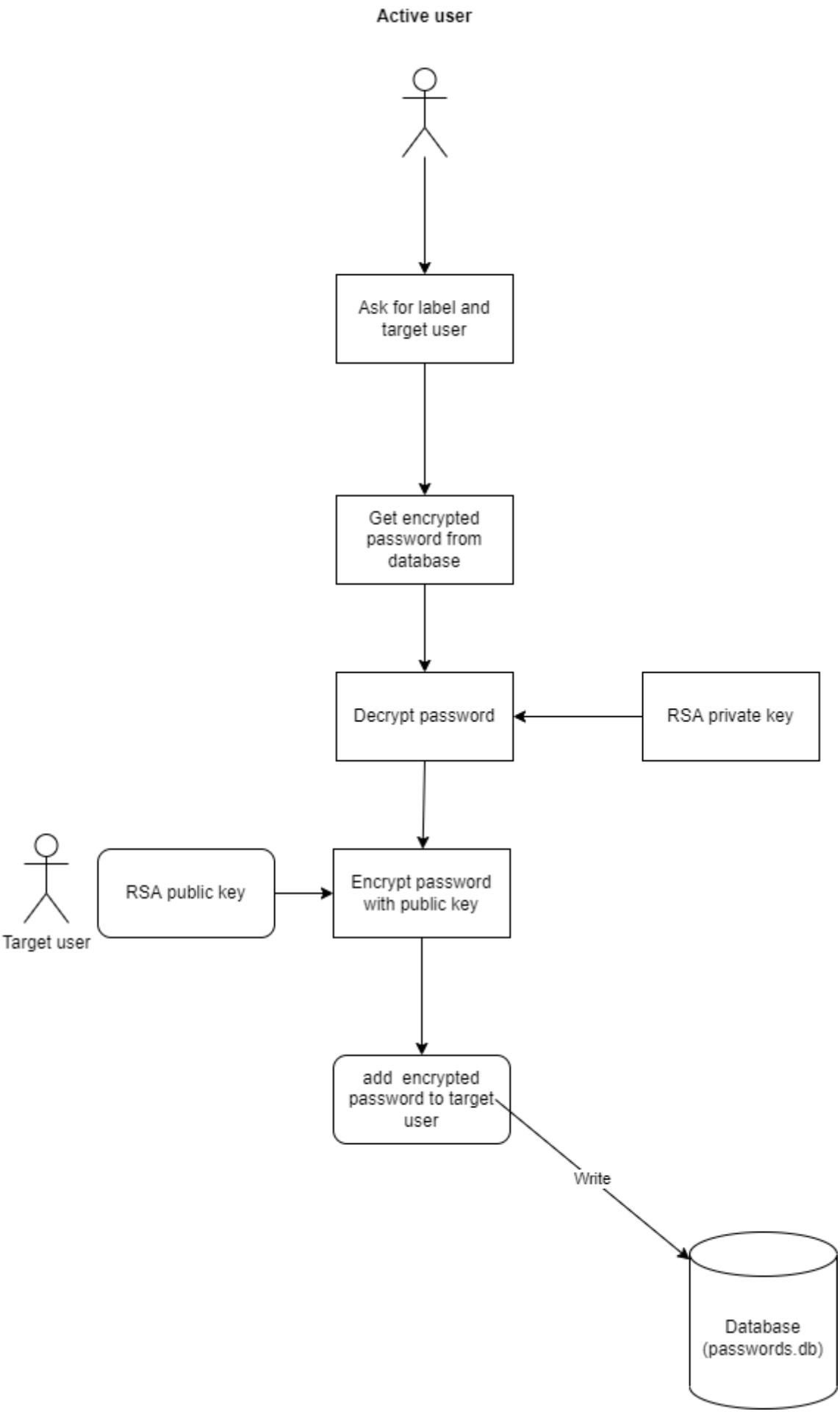

## Add a password




## Recover a password




## Sharing a password


Active user
Ask for label and target user
Get encrypted password from database
Decrypt password
RSA private key
Target user
RSA public key
Encrypt password with public key
add encrypted password to target user
Write
Database (passwords.db)

## Assets

| Name | Description | Protection |
| --- | --- | --- |
| RSA public key | The RSA public key is used to encrypt a password | no |
| RSA Private Key | The RSA private key is used to decrypt a password | Encrypted with the key derived from master password (chacha20_poly1305) |
| Master password | The master password unlocks the state of the program. | Only the hash of the password is stored in the database (argon2) |
| Derived Key master password | The key derived from master password is necessary to decrypt the RSA private Key | It's not registered anywhere |
| Stored password | Passwords stored in the database | Encrypted with the RSA public key |

# Algorithm

## Password master storage

The hash of the master password is created with Argon2.

The library used is argon2 : https://docs.rs/argon2/0.4.0/argon2/ with the default algorithm : **Argon2id**, a hybrid version combining both Argon2i and Argon2d.

The generated salt has a size of 128 bits.

Another library, rust-argon2, has only a salt of 64 bits. That's why it wasn't chosen.

Link : https://docs.rs/rust-argon2/latest/argon2/

## Master key derivation

From the password, a key is derived with the algorithm pbkdf2. The library used is pkdf2 : https://docs.rs/pbkdf2/latest/pbkdf2/

The algorithm used is different from that for the hash of the password, because an attacker could then obtain the key from the database.

## Asymmetric encryption

For the asymmetric encryption, it is the RSA algorithm. With a key of 2048 bits, it offers a satisfactory protection. It is easier to implement in Rust than an algorithm with elliptic curve.

The used library is openssl : https://docs.rs/openssl

An another library exists - RSA : https://github.com/RustCrypto/RSA. It has been audited, but the vulnerabilities found have not all been corrected by the developers...

For the padding, the openssl library offers three different paddings : PKCS1, PKCS1_OAEP, PKCS1_PSS

- PKCS1 is not IND-CPA secure
- PKCS1_PSS_PADDING is designed for signature, not encryption
- PKCS1_OAEP is IND-CCA2, but it is vulnerable to Manger's attack.

Therefore, padding OAEP  is used because it is the better solution between the three possibilities.

Link for documentation : https://docs.rs/openssl/latest/src/openssl/rsa.rs.html#50

### RSA private key encryption

The algorithm used is chacha20_poly1300 to perform authentication encryption. Also, the aes-gcm128 algorithm did not work with the openssl library for some unknown reason.

## Rust

### Memory

With the rust, the memory of an object will be erased as soon as the function ends. Its implementation is also more restrictive.

Unfortunately, it is a difficult language and I can not use its full potential.

It is also possible to delete objects immediately from memory with `drop`, without waiting for the end of the function : https://doc.rust-lang.org/std/mem/fn.drop.html. Nevertheless, I didn't have time to use it.

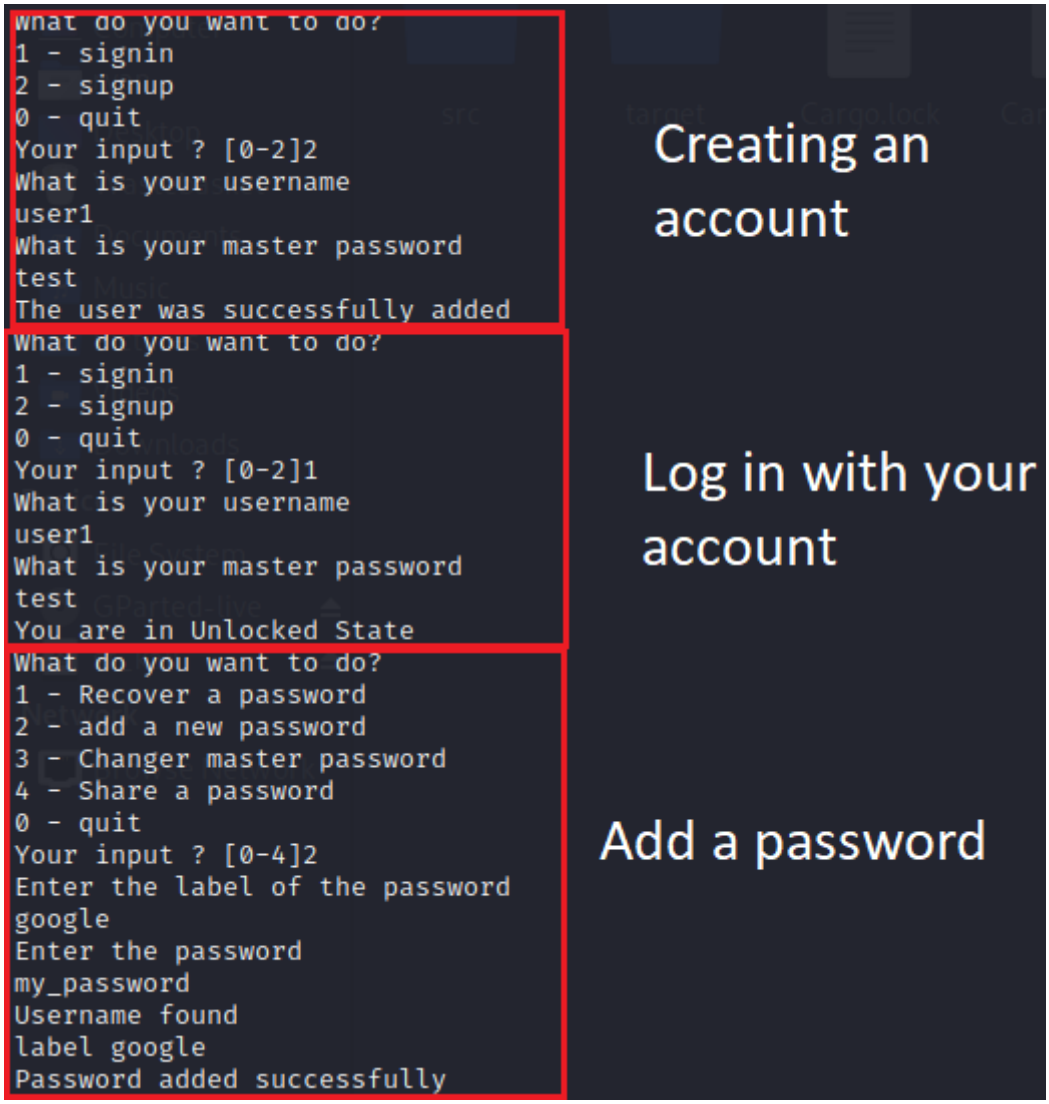
### Library

Personally, the crypto libraries were a big disappointment. Very few are audited and corrected. Moreover, many bugs are present.

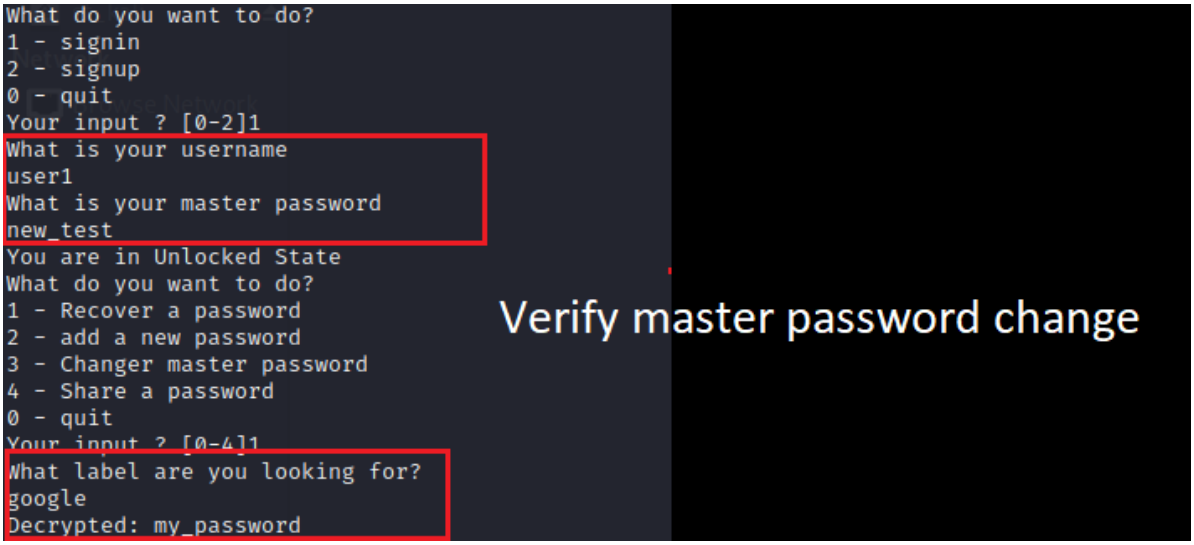
## Limitation

- The Derived Key from master password is vulnerable to side channel attack. When a user is connected, it is available in memory. The attacker can therefore decrypt the RSA private key to then decrypt all passwords from the target user.
- The IVs drawn for `chacha20_poly1305` are randomly drawn. It limits the number of private keys we can encrypted.
- The salt for generate the derived Key from master password is stored in clear in the database. For a better security, we could encrypt the salt with the user's public key.
- Encrypted passwords can be modified by an attacker in the database. No tag calculation is performed. However, it is likely that the decryption will cause an error.

### Implementation - Example


What do you want to do?
1 - signin
2 - signup
0 - quit
Your input ? [0-2]2
What is your username
user1
What is your master password
test
The user was successfully added
What do you want to do?
1 - signin
2 - signup
0 - quit
Your input ? [0-2]1
What is your username
user1
What is your master password
test
You are in Unlocked State
What do you want to do?
1 - Recover a password
2 - add a new password
3 - Changer master password
4 - Share a password
0 - quit
Your input ? [0-4]2
Enter the label of the password
google
Enter the password
my_password
Username found
label google
Password added successfully
Creating an account
Log in with your account
Add a password


What do you want to do?
1 - Recover a password
2 - add a new password
3 - Changer master password
4 - Share a password
0 - quit
Your input ? [0-4]1
What label are you looking for?
google
Decrypted: my_password
Recover a password
What do you want to do?
1 - Recover a password
2 - add a new password
3 - Changer master password
4 - Share a password
0 - quit
Your input ? [0-4]3
Enter your password
test
Enter your new password
new_test
Password changed successfully. You need to log again
thread 'main' panicked at 'No other solution found', src/auth/signin.rs:230:21
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
Change the master password


What do you want to do?
1 - signin
2 - signup
0 - quit
Your input ? [0-2]1
What is your username
user1
What is your master password
new_test
You are in Unlocked State
What do you want to do?
1 - Recover a password
2 - add a new password
3 - Changer master password
4 - Share a password
0 - quit
What label are you looking for?
google
Decrypted: my_password
Verify master password change