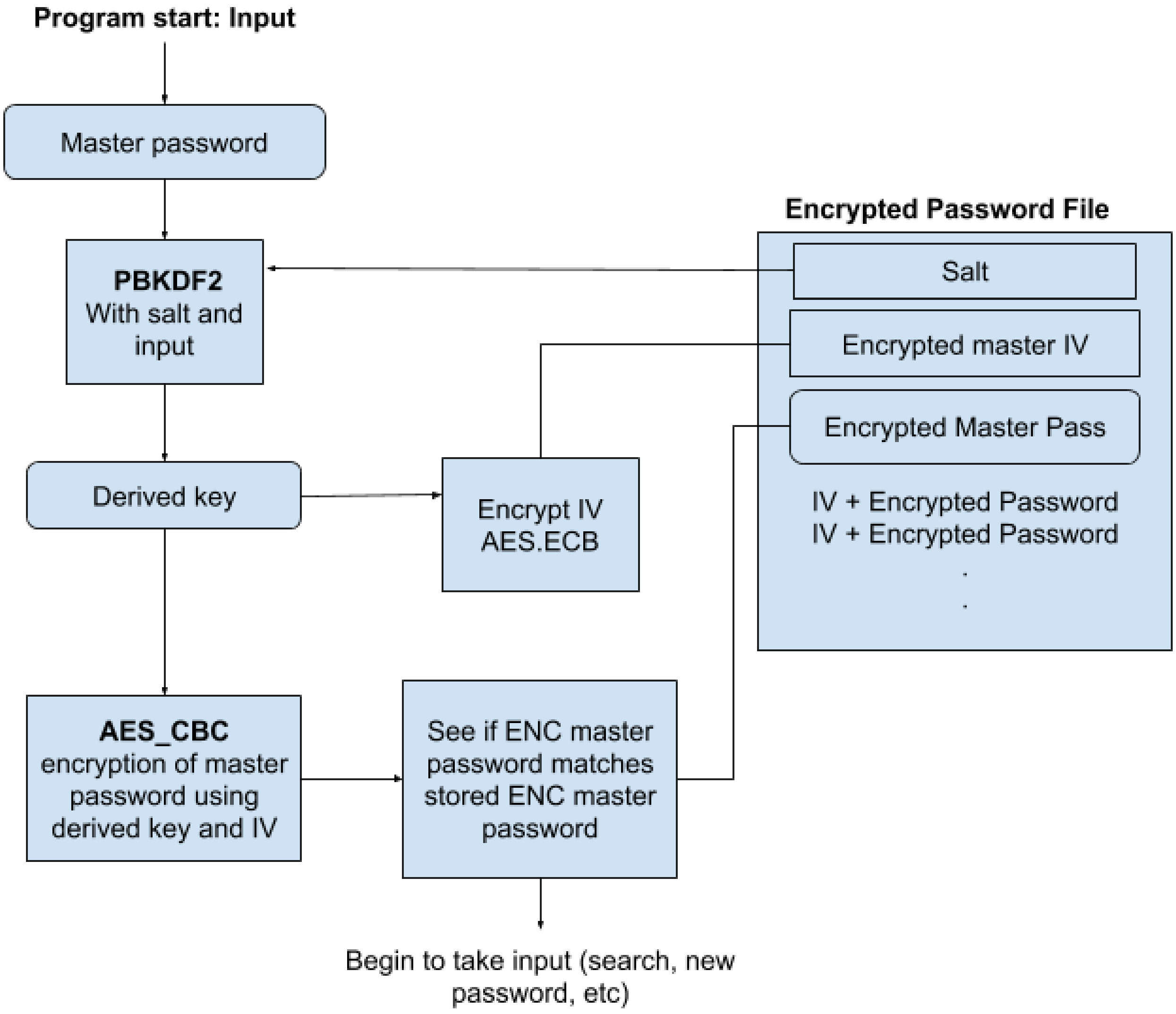
Group: Sammy Stolzenbach, Peter Mehler, Henry Jacobs
Option B - Password Management System

Overview of system architecture and operation

In order to begin the setup of the password manager, the program first checks if the encrypted password file is empty. If this is so, the account creation begins. If not, move on to main program. The master password is required from the user with a minimum length of 16 characters with at least one capital, special character, and number. A random 64 bit salt is generated using PRNG. PBKDF2 is then used with inputs salt, master password, and 1000 iterations to generate the Derived Key. It will then encrypt the master password with AES.CBC and the 64 bit random IV with AES.ECB. The enc_password file is opened and the salt, encrypted IV and encrypted master password are written into the password file. The derived key will be stored for later session use.


Program start: Input
Master password
PBKDF2 With salt and input
Derived key
Encrypt IV AES.ECB
AES_CBC encryption of master password using derived key and IV
See if ENC master password matches stored ENC master password
Begin to take input (search, new password, etc)
Encrypted Password File
Salt
Encrypted master IV
Encrypted Master Pass
IV + Encrypted Password
IV + Encrypted Password
.
.

Main Program:
After the program start master password validation is completed, the user is prompted with two options: Add New Account and Search for Existing Password. If the user selects add new account, he or she is prompted to manually input the URL and Username, and has the option to manually input Password or to use an auto-generated password of character length 20. The Username and URL are stored in plain text in an Accounts file. The password is encrypted using AES.CBC, where the IV is decrypted using the derived key, and stored in the Encrypted Passwords file. If the auto-generated password is selected, a PRNG creates a password of 16 characters long.

For password retrieval, the program searches the Accounts file for matching URL or Username. Because the index of the information in the Accounts file will match the index of the information in the Password file, we can use only URL and or Username to also retrieve each password. Once password is found, copy to clipboard.

Attacker Models:
We assume that our attacker has access to the program itself, the encrypted passwords file, the user's screen while using the program, and the memory of the program while it is running.

Security Requirements:
In order to protect against Offline dictionary attacks, several holes must be patched. We assume that the attacker can brute force the master password offline by running the program many times. Because the salt is stored as plaintext in our file, which we assume the attacker has access to, we must make the brute force method as difficult as possible by increasing the amount of attempts and time required to correctly guess the master password. In order to do this, we will require specific unique character types as well as a minimum length for each password. Because of the nature of the way in which the master password was implemented, if the attacker is able to access the individual passwords in the password manager, this has no effect on the attacker's ability to guess the master password. In order to reduce the chances of the attacker having precomputed passwords, the salt would deter such advances.

The issue with CBC encryption is that we would be using the same IV for each encrypted password, which would produce the same ciphertext with the same plaintext. Although we considered using CTR mode, this would reveal the length of the passwords. Finally, we decided upon CBC encryption for each individual password, using a unique iv for each. This takes up more space, but it ensures an (almost) standard length for each password in the encrypted file. Unfortunately, because our accepted password range is 12-32, any password below 16 characters is going to appear as one block length because of padding, while any password above a block length will be padded to a size of two block sizes. Thus, an attacker would be able to tell which passwords are length 12-16, however we believe that the strength of the passwords is still enough to deter brute force given our accepted character range and restrictions against simple passwords.

In order to protect against Online Dictionary attacks, it is necessary to protect against brute force computation of the master password. Because of this, the program quits out and sends an email to the user who created the account after 3 incorrect tries of the master and locks out the user until the email is addressed (this may not be feasible for us practically speaking as we are unsure how to code it, but we will try). In order to protect against a potential memory attack in which the attacker has access to the system memory of the user's computer, we initialize the master password to a blank string as soon as it is encrypted. Finally, given that an attacker has the capability to view a user's screen while using the program, asterisks are used to hide the input of the passwords in the GUI.

Note: The idea to use MACs in the original design plan was removed due to the fact that the MACs did not add any security. The encryption of the master password with a MAC does not add anything to the security of the master password, because there is no incentive for an attacker to alter the master password, or any of the passwords, unless they wanted to simply break the program.