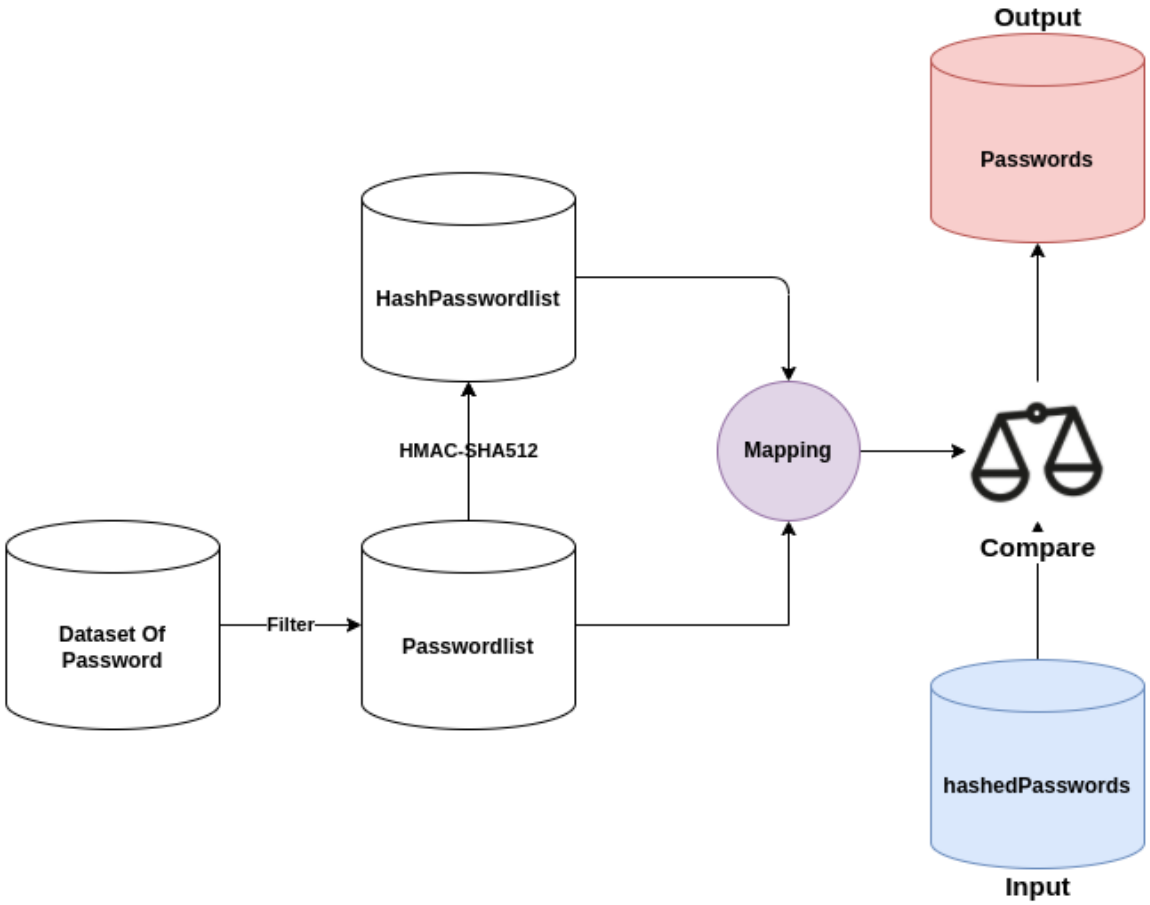
Given Data:

- **pass** >= 7 characters and **pass** in >=1 numeric
- **users** <= 100000
- **input** file format (num:user:hash)
- **output** file format (num:user:password)
- **PBKDF2** algorithm (iteration= 10,000 and len = 16 byte keys)
- **PRF** = HMAC-SHA512
- **salt** = "SKKU seclab"

The approach of solving the assignment:



Figuer1: Overview of the Password Cracker

I used during developing the system OpenSSL library, to use the HMAC-SHA512 to hash the passwords, as well as I used this dataset password of the list that contain at least 7 characters here to prove the speed of our mechanism in applying the password cracker to check the brute force attack.

As shown in figure1, after collecting the dataset it contains around 500000 passwords, that used by common users, and based on the criteria of the assignment, I reduced it to 100000 passwords, and this password should contain more than 7 characters and at least one numeric value, so after filtering it, right now the password list became 100000. Afterward, I used the OpenSSL library to use the HMACK-SHA512 to get the hash code of these passwords. Then I mapped the passwords with the hashed of it, to compare the hashed that will be received from the input file to get the right password as an output.

Limitation:

1. **Dataset Of Passwords:** As I mentioned I used the popular one dataset here, so I am not sure if users use the same as mine, So if the user did not have the same hash in my list, it will output as ***"Password Not Found".***
2. **File size of the output:** While I don't know the size of the input file, it will be hard to imagine how the volume of the output file, but I am sure based on my mechanism, the file will produce as output is less than 5MB.
3. **Restricted the length of the password:** so if the length of the password is more than **10**, it will not be included. I released that based on this article here most of the users use passwords length from 8 to 10.

Run the Code:

Requirements:

- Ansi C
- Openssl 1.1.1
- Crypto library

There are two files:

One to generate a hashed of the passwords and the second one to check the hashed of the input file with the hashed of the file that I generated from here to get the output file.

1) Generation_Process.c

   gcc -o test Generation_Process.c -L/usr/bin/openssl -lssl -l crypto

   To generate hash for list of passwords (470088), it will take around (2423.358448 sec)

2) Test_Data.c

   gcc -o test Test_Passwords.c

   To search in my list of hashed(470088), for one user it will take around (0.31 sec)