SecAuth: Stronger authentication with keystroke features

Report by: Sanya Ralhan (903248565)

Ravi Prakash Giri

To build:

Command: sudo python main.py <input\_filename>

Initialization:

* As per this step we select a random 160 bit long prime q, h\_pwd<q and h = 5 (we store features for the first five attempts and start checking from the 6th login)
* We select a random polynomial with order m-1 where m is size of password (we have assumed a max size of password as 20)

Creation of Instruction Table:

We have list which puts the calculated α and β values as per paper ie (SHA hash function is used for hashing)

For slow feature:

α =alpha\_cal(pwd, i+1, poly)

β = beta\_cal(pwd+str(random.randrange(0, 1000)), i+1, polynomial\_gen(max\_features-5, random.randrange(0, q\_val-1)))

For fast feature :

α = alpha\_cal(pwd+str(random.randrange(0, 1000)), i+1, polynomial\_gen(max\_features-5, random.randrange(0, q\_val-1))) β = beta\_cal(pwd, i+1, poly)

For undistingushable feature

α=alpha\_cal(pwd, i+1, poly)

β=beta\_cal(pwd, i+1, poly)

Basically for each features, if it slow then β is calculated on a new random polynomial as well as pwd appended with big randomnumber and if it fast then α is calculated like that.

Login attempt:

* If feature is fast , we select α and β (slow or undistinguishable) otherwise.
* X,y values are derived using the value selected above.
* New h\_pwd is calculated using lagrange interpolation.
* Without error correction is file doesn’t decrypts correctly print 0 else 1.
* All values are reinitialized ie q, polynomial and instruction table id recalculated based on the average of feature values coming from history file +current features. Current features are written into history file.

Error Correction:

We have a function which tried to decrypt the history file with h\_pwd and if decryption fails ie. DecryptionException then we change value of each feature from slow to fast or vice versa and call check at step 1 onwards again, if this also fails we print 0(unsuccessful) else 1(successful)

History File:

Size of history file is fixed to 500 and padded with S character and “$$$$” is used as a redundant separator between the padding and file text. SHA of h\_pwd is used as key to encryption. This is used to retrieve data after decryption Encryption is done using simple-crypt-4.0.0 library functions.

Input Validation:

if length of password entered >20 or number of features is not m- 1 where m is password length, the program exits

Error Handling :