SY201 – Cyber Operations Fundamentals; Programming Assignment `03`

Password Stretching

Name: ______________________________ Alpha: __________________________

Begin this assignment by making sure your repo is up to date. Change to **`~/repo201`** and type:

```
git pull
```

All the files required for this assignment, including an electronic version of this handout, are available in:

`~/repo201/programming/pa03`

Read this entire handout at least once before you begin your research or start coding

1. **Background**:

In this assignment you will create a program that performs *password stretching;* repeating a hashing algorithm a number of times to obfuscate a password. Password stretching is a defensive mechanism that can significantly increase the computational time required for an adversary to "brute-force" a password. Modern-day stretching is more complex and secure than the basic stretching that you will implement in this lab, but the core concepts are the same.

What is a hashing algorithm?

"There are three types of cryptography algorithms: secret key, public key, and hash functions (algorithms). Unlike secret key and public key algorithms, hash functions, also called *message digests* or *one-way encryption*, have no key. Instead, a fixed-length hash value is computed based on the plaintext that makes it impossible for either the contents or length of the plaintext to be recovered."[1]

Good hashing algorithms have three important characteristics:

a. It's easy (*for a computer*) to compute the hash for a given string.
b. It's hard (*takes way too long*) to start with a hash value and reverse-engineer a string that hashes to it.
c. It's impossible (*or practically impossible*) to start with one string and find a

[1] https://www.sans.edu/cyber-research/security-laboratory/article/hash-functions

different string that hashes to the same value – an ideal hash function will produce unique keys (called: *avoiding collisions*).

Conduct some basic research by visiting these resources:

a. Read the introduction and sections: 1.5 (*Protecting Data*), 1.9 (*Standard uses of hashing in cryptography*), and 2 (*Properties*) for the Wikipedia article at this link:

   `https://en.wikipedia.org/wiki/Hash_function`

b. Read the *Salt (cryptography)* Wikipedia article, available at this link:

   `https://en.wikipedia.org/wiki/Salt_(cryptography)`

c. Read the *Key stretching* Wikipedia article, available at this link:

   `https://en.wikipedia.org/wiki/Key_stretching`

2. **Your Program Should Do This:**

a. Display a menu with the following options:

```
*** Password stretcher ***

1. MD5 1000 Rounds
2. SHA-1 4096 Rounds
3. SHA256 41000 Rounds
4. User Specified
5. Quit
```

b. For options 1, 2, and 3, you'll always use the number of rounds listed above. For the *User Specified* option (4), you'll further prompt the user for the algorithm to use (`MD5`, `SHA-1`, or `SHA256`), and then separately prompt the user again for the number of rounds to use (a valid integer `> 0`). After selecting option 4, an example of the *User Specified* prompts is shown below:

```
Choose an algorithm:

1. MD5
2. SHA-1
3. SHA256

Enter the number of rounds:
```

c. Prompt the user to enter a password, and then prompt the user to enter a salt.

d. Concatenate the salt and password together, with the salt first (*salt* + *password*).

c. Your program should repeatedly loop, hashing the combination of *salt* + *password* once, then hashing the result over and over until the specified number of times is reached. For example: if I were to manually stretch a string three times using `md5`, it would look like this in pseudocode:

```
myString = "Hello World"
hashValue = md5hash(myString)    # Stretch 1
hashValue = md5hash(hashValue)   # Stretch 2
hashValue = md5hash(hashValue)   # Stretch 3
```

d. Once you've stretched the *salt* + *password* combination, display the results to the screen, showing the hashing algorithm used and the number of rounds processed.

e. Using the hashing library in Python is pretty straightforward, though the syntax can be a bit confusing. As you know, you can extend Python's capability by importing additional libraries. To use the cryptographic hashing functions in Python, put the following line at the top of your code:

```
import hashlib
```

Below is an example of how you create the hexadecimal representation of the `md5` hash of a string:

```
import hashlib

myString = "Hello World"
hashValue = hashlib.md5(myString.encode()).hexdigest()
print("Your md5 hash is:",hashValue)
```

This stores the hexadecimal representation of the hash of "`Hello World`" in the variable `hashValue`.

f. For this assignment, you'll make use of the following `hashlib` library calls:

   i. `hashlib.md5( )`
   ii. `hashlib.sha1( )`
   iii. `hashlib.sha256( )`

3. **<u>Design Requirements:</u>**

Think carefully about how you want to approach this assignment before you start coding. It's rarely effective when you open up your editor and just start typing. Think first about the kinds of variables you will use, their names, and how the different sections of your code will interact. Take out a piece of paper and pencil and sketch out the major components of your design (collect input, validate input, perform calculations, present results, etc.). Draw simple flow diagrams or write code-like snippets (called *pseudocode*) that capture the major steps in your program. Your design sketch and / or pseudocode is a required deliverable for this assignment.

4. **<u>Error Handling</u>**

a. If an invalid menu option is entered in the main menu, then report an error and redisplay <u>the main menu</u>.

b. For option 4 (*User Specified*):

   i. If an invalid menu option is selected for the *Choose an algorithm* menu, then report an error and start over by redisplaying <u>the main menu</u>.

   ii. If an invalid number of rounds is entered (not a number, or not a positive integer), then report an error and <u>exit the program</u> without further processing.

5. **Program Flow**

Here's are two sample runs of the program:

```
*** Password stretcher ***

Please select an option:

1. MD5 1000 Rounds
2. SHA-1 4096 Rounds
3. SHA256 41000 Rounds
4. User Specified
5. Quit

Selection: 2
Enter password: Go Navy!
Enter salt: abc
sha1 hash for 4,096 rounds: 8021743c3a30cb9358ab3a6b7bb0750f2e133fd3

------------------------------------------------------------------------------------------

*** Password stretcher ***

Please select an option:

1. MD5 1000 Rounds
2. SHA-1 4096 Rounds
3. SHA256 41000 Rounds
4. User Specified
5. Quit

Selection: 4

Choose an algorithm:

1. MD5
2. SHA-1
3. SHA-256

Selection: 1
Enter the number of rounds to stretch: 7534
Enter password: BillTheGoat
Enter salt: yyz
md5 hash for 7,534 rounds: 91b393ed1662ee349267732771ab12fd
```

6. **Testing:**

Thoroughly testing your program can be a difficult task. For complex programs, there may be hundreds or thousands of possible cases to test. I recommend conducting some sample runs on the executable version of the program (see **Hints**, below) and comparing them against your results.

7. **Hints:**

   a. Give your variables meaningful, descriptive names. While it's true that you could represent the password using a variable named `p`, it will be much easier to read and debug your code if you use a variable name like `passWord` instead.

   b. You may find *Liang*, section 5.7 (p. 151) useful.

   c. The directory for this project also includes an executable version that you can run to get a sense for how your program should operate. The program is called: `stretcher`.

   Let's assume you're working in your `~/shares/sy201` directory. You can run `stretcher` by first copying it to your working directory and then adjusting its permissions with the following commands:

      i. `cp ~/repo201/programming/pa03/stretcher .`
      (*the trailing space and period in the command above are important*)
      ii. `chmod 755 stretcher`

   You only have to perform steps (i) and (ii) once. After that, you can run the program any time you want by typing: `./stretcher`

8. **Deliverables And Due Dates:**

   a. Using paper, and pen or pencil, complete your pseudocode / flow diagram *before* you start coding. Be sure to indicate your name and alpha on this page.

   b. Your completed source code (the one ending in `.py`).

   c. Submit parts (a) and (b) by 2359 on Tuesday, 18 September, in accordance with your instructor's directions.