Programming Assignment 3: TOTP
CS 370

In this assignment you will be writing a program that can work with `Google Authenticator` (GA) [Android] [iOS] [Source]. Google authenticator supports different kinds of One Time Password (OTP) algorithms. In this assignment, you are going to implement 30-Second Time-Based One Time Password (TOTP). If you haven't used GA before, I encourage you to try Google Authenticator for your Google accounts to make sure you understand how it works from a user's perspective.

The specification for TOTP algorithm is defined in RFC 6238. RFCs are documents from Internet Engineering Task Force (ITEF) which is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. One of the learning objectives of this assignment is for you to get familiar with an RFC and learn how to read and translate an RFC into an implementation.

As a part of this assignment, you have to build a program that can generate QR codes which are readable by Google Authenticator Android App. Functionality wise, I assume, Android and iOS apps are same and are supposed to provide similar codes. In case you cannot test your program against GA Android app but only against an iOS app, you can specify that along with your submission. We recommend using Python3 to complete this assignment, but please get in touch with the TA in case you prefer implementing in a different language.

You must submit one zip / tar compressed file with a `Makefile` that compiles the program if it needs compiling. Also, please add explicit instructions on how to run your program in the `README` file. Also, explain your implementation briefly in `README` file. You are allowed to make any reasonable assumptions about your program's functionality as long as you mention them in your `README`. Your program should work like this (showing for Python):

- `[50pts]./totp.py --generate-qr:` This command should generate the secret key and encode it in a QR code. QR code should be output as a .jpg picture or .svg picture and should encode the URI GA expects. URI contains **secret keys** along with the **user id** required for the TOTP algorithm. Refer to this page here which provides details about the format of URI and how to add extra information in URI. Save the randomly generated secret key to a known location (e.g., `secret.txt`) You'll need it for the next step.

- `[100 pts]./totp.py --get-otp :` This command should generate an OTP corresponding to the secret in the `secret.txt` file which would match the OTP generated by the Google Authenticator and print the number of seconds (X) it is valid for. Then waits for the X seconds and prints the next OTP and continue to go in this manner printing a new OTP every 30 seconds in sync with Google Authenticator. **NOTE: Using python or other OTP libraries will NOT get you any points.** You will have to implement your own OTP generation function following the specification from the RFC. However, you can use a library to generate the QR code. Python QR code libraires: https://segno.readthedocs.io/en/stable/comparison-qrcode-libs.html

- `[BONUS: 25pts]:` If you protect your secret key stored in a known location (e.g. `secret.txt)` using a password and standard key storage format. README should include information on how to use this feature and describe how you protect the key using the password. You may extend the command line option accordingly and describe the usage in the README file. Full bonus score will only be given if the key is protected properly using the password. Partial credit otherwise.