What is TOTP?

Having two-factor authentication on all your accounts is a good way to keep your data more secure. With two-factor authentication logins, not only is a username and password needed, but also a one-time-use code. There are a few different ways to get that code, such as by email, phone, or SMS. But my favorite way is to do it via a "Google Authenticator" time-based OTP (one time password), also known as a TOTP.

Using an app on your phone like Authy or Authenticator, you set up a secret given to you by the service, then every 30 seconds a new code is generated for you. What’s extra nice is that the Google Authenticator protocol is supported by just about every service and phone/tablet.

In our hacker household, Ladyada does not own a phone. A cell-phone jammer, yes. A cell phone, no. Fill is essentially the phone as needed or a tablet can be used. Why purchase a phone just for two-factor authentication?

Luckily for us, the Google Authenticator protocol is really simple. You just need to be able to know the current time, and run a SHA1 hash with both a known secret (given to you by the online service) and the UNIX epoch time in seconds divided by 30 (so you have plenty of time between code-updates).

We built a simple device that does nothing but generate TOTPs, using CircuitPython - it’s Python for microcontrollers! It uses a Feather ESP8266, which has Wi-Fi so it can connect to NTP to get the current time on startup, and a Feather OLED to display text nice and clearly. You can use an ESP8266 with just about any OLED, with some hacking.

Every time a new code is needed, click the reset button and, within two seconds, it displays the three most common TOTPs on hand (yes, it is that fast!)
Flash the Latest Version of CircuitPython
(You'll Need v2.2 or Higher)

We're using the ESP8266 Feather, which means it has lots of memory and Internet capability. We use the Internet part to get the current time with NTP. Since the ESP8266 doesn't have native USB, we have to upload our code using Ampy, an open-source command line tool that "types out" the Python script and saves it to the ESP8266 Flash memory.

Once you've gotten Ampy working, you'll need a bunch of Python libraries to get the OLED working. Use Ampy to create a directory called "lib" and upload "adafruit_ssd1306.mpy", "adafruit_register", and "adafruit_bus_device" library folders (https://learn.adafruit.com/welcome-to-circuitpython/circuitpython-libraries).

Then check with Ampy's ls command to verify all your files are in place!

Now you can download the main script to your computer and save it as "main.py".

The code is on GitHub along with an extended how-to (https://github.com/adafruit/Adafruit_Learning-System_Guides/blob/master/CircuitPy_OTP/main.py)

Don't upload it via Ampy yet! The current file has fake tokens in it that need to be set. Before uploading, change these two lines to your network SSID and password:

`ssid = 'my_wifi_ssid'
password = 'my_wifi_password'

You'll also need to get two-factor "authentication tokens/secrets." Each site is a little different with regards to how they do this. For example, when you set up Gmail for two-factor authentication, it will show you a QR code, which is great for phones. For us, we need the base32-encoded token. Click the "Can't Scan It?" link or otherwise request the text token.

That string of letters and numbers may be upper case or lower case. It may also be 16 digits or 24 or 32 or some other quantity. It doesn't matter! Grab that string, and remove the spaces so it's one long string like ra4nd2ut-totesol564z3jjj5jo677. Note that the number 0 and number 1 never appear, so anything that looks like an O, 1, or an I is a letter. It doesn't matter if it's upper or lower case.

Now edit this section of the code. You can display up to three accounts on a Feather OLED. If you pad the name with spaces, the numbers will be right-justified, but it's not important - we are just picky. Here's our demo setup:

```
totp = [('Discord ',
        'JBSWY3DPEHPK3PXP'), ('Gmail ',
        'abcdefgijklmnopqrstuvwxyz
        234567'), ('Accounts',
        'asfdkwefoalwejfa323nfjk1')]
```

If you want to test the setup first, you can keep the Discord entry, which is the "PyOTP" example token. Then, scan this with your phone in Authy or Google Authenticator.

OK, once you've set everything up, let's test! Run the program directly on the Feather with OLED attached using ampy --port portname run main.py.

You'll see it connect to your local network, get the time via NTP, then calculate and display OTP codes both on the OLED and on the serial port (you'll need to wait until the program is done to see the serial output).

If you do have a phone, check against your phone to make sure the codes are correct. Once you're satisfied, tweak these two lines to change the behavior. Then finalize by uploading main.py with Ampy's put command.

```
ALWAYS_ON = False # Set to true if you never want to go to sleep!
ON_SECONDS = 60 # how long to stay on if not in always_on mode
```

Good night and good luck!