Yubiduino: TOTP on an Arduino

Jessica Fleck, Brandon Mills, Paul Tela
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210
{fleck.48, mills.511, tela.3}@osu.edu

***Abstract*—Two-factor authorization (2FA) is a more secure method for authenticating users during login. Many 2FA systems, including the one used by the popular Gmail service, follow the Time-based One Time Password (TOTP) algorithm standardized in RFC 6238. Per the standard, the client and server agree on a shared secret key and starting timestamp at setup. Then, each time the user attempts to log in, the algorithm generates an ephemeral numeric passcode by running a cryptographic function with the shared secret key and the number of elapsed time intervals since the starting timestamp as inputs. However, this system requires a user to obtain and manually enter another unique password every time they log in. Inspired by the commercial YubiKey multi-factor authorization key, we designed and built a device, called a Yubiduino, that calculates and automatically enters the one-time password on the user's behalf. The device consists of an Arduino Due microcontroller, a DS3231 real time clock with a battery backup, a microSD card for persistent storage, and a button that the user presses to activate the algorithm. After setup, the user has only to plug the Yubiduino into a USB port, select the field in the login form for the one-time password, and press the button, and it will calculate and type the password automatically.**

I. Introduction

Internet services permeate every facet of modern life. Communication, finances, and even our very identities all live online. Through the power of data, Google, Facebook, and dating comany OkCupid know more about their users than their users know about themselves. This trove of personal and financial information is an enticing treasure to malfeasants, who contrive ever more sophisticated attacks through backdoors and social engineering. Multi-factor authentication stands in the way of an attempted account hijacking by verifying identity through a combination of things only the user *knows*, *has*, or *is*. In the case of the Yubiduino, the user's standard login password is something only the user *knows*, and the one-time password is generated from the secret key that nobody but the user *has*. Without both, no attacker can gain illegitimate access to an account. The Yubiduino lets users add this second layer of security to any service that supports Two Factor Authentication via the Time-based One Time Password standard, including Gmail, GitHub, and Facebook.

II. System Design

Hardware assembly was necessarily the first step in building the Yubiduino. The part list was fairly short:

- Arduino Due microcontroller
- Breadboard
- Arduino-compatible Ethernet shield
- MicroSD card
- DS3231 real-time clock module with backup battery
- Momentary push-button switch
- 10 kΩ resistor
- Various wires

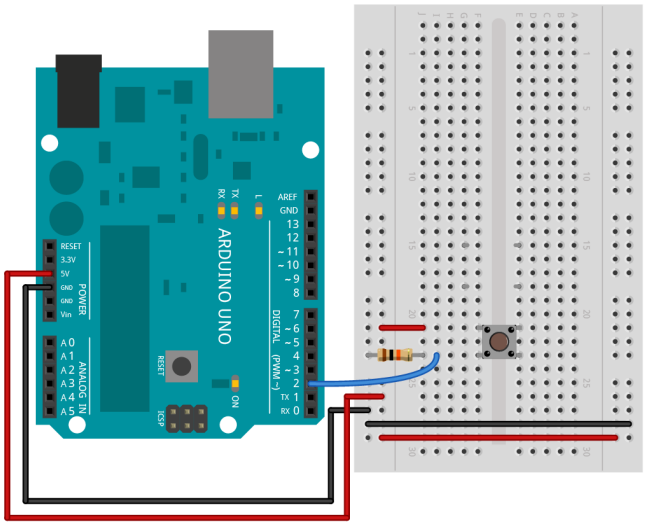

Fig. 1. Button Circuit

Assembly consisted of connecting the button as shown in Figure 1 and the real-time clock (RTC) module as shown in Figure 2, then stacking the Ethernet shield directly on top of the Arduino.

As part of two-factor authorization setup, Gmail and Facebook will make available a base32-encoded secret key. To set up the Yubiduino, this key should be copied to a file named "key.txt" on the Yubiduino's FAT-formatted microSD card. This must only be done once. When the user logs in online, plugs in the Yubiduino, and activates it by pressing the button, it emulates a USB keyboard and "types" the digits of the one-time password into the currently-focused form field. The code itself was determined by running the TOTP algorithm using the shared secret key from the microSD card and the timestamp from the RTC.

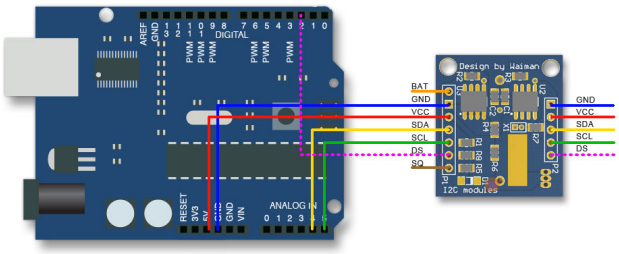

Fig. 2. RTC Circuit

III. Implementation and Evaluation

A. Implementation

* Handling button press * Programming the RTC * Writing via the keyboard * Accessing SD card storage and reading key from file. * Base 32 decoding * Getting the time from the RTC * Fixing Sha library * Generating C to pass to SHA1-HMAC * Calling SHA1-HMAC * Truncating the result according to TOTP spec

Yubiduino's software was implemented in C++ using the AVR-GCC compiler to target the Arduino Due board. This board uses a 32 bit ARM core micro controller running at 84Mhz. This micro controller allows for 4 byte wide data operations to occur in one clock cycle, which provides significant performance benefits for the cryptographic hashing code.

Several libraries were used in order to build the Yubiduino software package. Standard libraries used were `Serial`, `Keyboard`, `SD`, and `SPI`. In addition to these standard libraries, two third party libraries were used. These were `DS3231` and `Sha`.

The `Serial` library provides a way for the Arduino to communicate via a serial interface. This is primarily used for interacting with the Arduino using the Arduino IDE's built in Serial Monitor. The `Keyboard` library is used to allow the Arduino to send keyboard input to a connected computer. The `SD` library is used to read and write files on an SD card and supports both the FAT16 and FAT32 file systems. An ethernet shield was used in order to provide an SD card input. The `SPI` library allows for access to the Serial Peripheral Interface. This interface allows for short distance communication between micro controllers. This library was used to communicate with the Real Time Clock (RTC) over I^2C.

Third party libraries were used when standard library was not available. The `DS3231` library was used to communicate with the RTC. It provided convince methods for reading the current time and converting the time between different formats. The `Sha` library provides SHA1 and SHA1-HMAC cryptographic hashing capabilities.

B. Evaluation

IV. Conclusion

Security is very important when it comes to a users information. The Yubiduino helps make sure that everyone's information is secure and safe from intruders while being very easy to use and set up. Two Factor Authentication makes it harder to get into accounts because the second factor has to be physically there. The Arduino, ethernet shield, SD card, and real time clock made it possible to create this second factor to help shield intruders from getting into important accounts.

Additional references
http://arduino.cc/en/Main/ArduinoBoardDue
http://arduino.cc/en/reference/serial
http://arduino.cc/en/Reference/MouseKeyboard
http://arduino.cc/en/Reference/SD
http://arduino.cc/en/Reference/SPI
https://www.yubico.com/about/intro/yubikey/
http://tools.ietf.org/html/rfc6238
http://en.wikipedia.org/wiki/Time-based_One-time_Password_Algorithm
http://en.wikipedia.org/wiki/Multi-factor_authentication
http://blog.okcupid.com/
https://www.authy.com/add-2-factor-authentication-facebook
https://help.github.com/articles/about-two-factor-authentication/

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.