Wireless Door Lock Report

Joshua Satterfield

# Description

We successfully created a system which allowed you to wirelessly lock and unlock a deadbolt using an Android based application and an Arduino based controller.

# Specifications

Wireless: Arduino Wifi Shield

Development board: Arduino

Microcontroller: ATMEGA

Motor: Servo

Attachment: 3M Tape Strips

Device: sits around face of lock turn for easy installation

Wireless response time: less than a second

Power: able to be power from unmodified Arduino over USB. 500 mA, 5V (Apple)

Protocol: UDP based

# Challenges

Initially we tried using an Xbee Zigbee module with an Arduino Wireless shield. We discovered that it used a special radio protocol intended for networking between other devices of the same kind. Then we tried an Xbee WiFi chip, which uses the correct protocol. There was little documentation for using it on an Arduino. We tried using a third party library but I could not initialize the device. Then we got an Arduino WiFi shield. It was designed specifically for the Arduino and had more library information. The examples for that chip were easy to use to connect to a wireless network and to send and receive data. I then devised a simple prototype for a protocol which we could use to send signals to the Arduino. We tested the two initial commands for locking and unlocking using LED’s. Once we were able to control the Arduino wirelessly I began building a smart phone application.

I started developing a cross platform solution using the ionic mobile app framework. The advantages were that it allows programs to runs on Android or iPhone. It uses a hybrid model with Javascript controlling the web view and native plugins for libraries that use the hardware. Due to the networking libraries work with frameworks it was difficult to set up a networking object. I was not able to properly access the library. I also considered that native encryption would be preferable to a Javascript solution which may have been used in a hybrid app. I also tried Apple’s new Swift language. It has advantages over Objective-C – cleaner syntax, while retaining access to C library’s and other library’s that are set up in a simpler way than with a hybrid application. The new iPhone also has a fingerprint scanner and API. Due to budget and time constraints I decided to concentrate on the Android application. It uses the DatagramSocket library to send data over UDP. I used an asynchronous function for networking code to keep the UI from blocking.

After testing the wireless using a laptop we were able to control the LEDs using a smartphone with the Android application. Then we hooked the Arduno up to a servo attached to the enclosure. The servo has a connector that fits loosely to the thumb turn. This allow for easy installation using the adhesive to fasten to the surface of the door. It can quickly be installed on other standard sized deadbolt.

By using WiFi we achieve lower cost than GSM and leave it open to the possibility of control from the user’s home network. This allows a system to be constructed that sends commands to the users house from the internet using the existing receiver on the Arduino. In contrast to Bluetooth or GSM where an additional receiver would be needed to access WiFi. The disadvantage is that the Arduino needs authorization to use the home network although it can be set up to scan SSIDs. The advantage is that you can communicate as desired from other systems on your network directly or externally using port forwarding.

For encryption I began researching the DES protocol. It is now vulnerable to feasible attacks and 3DES is questionable. It still has vulnerabilities and can be slow for software implementations since DES was designed for hardware (Pornin). So I looked into AES. It has acceptable security for the foreseeable future. I was able to find an Arduino that implemented the basic features of AES and CBC mode.

To use AES you have to generate a initialization vector. In CBC mode it should be random. The library I am using does not have a method to generate one automatically so a static test vector is being used. Initially I planned to generate a random number and use it as a challenge, sending it to the phone. I was able to generate one. It appears that the iv serves the same purpose, to prevent replay attacks.

# Improvements

1. Encryption using AES
2. Use random IV
3. Use prototype setup button to send key to be stored on phone
4. Use separate accounts with separate keys
5. Secure the keys within the Arduino. Possible using hashing.
6. Create unified SMD circuit to reduce size of device.
7. Test battery usage with 9V
8. Consider desirability of proximity sensor for distance based triggering
9. Consider Message Authentication Code

# Code

Code and presentations are available at <https://github.com/jnsat/lock>

### References

Apple Support. “Apple Computers and Displays: Powering peripherals through USB” <https://support.apple.com/en-us/HT204377> Retrieved 2015-12-04

Thomas Pornin. Answer to “Why is AES more secure than DES?”. <http://stackoverflow.com/questions/3929325/why-is-aes-more-secure-than-des> Retrieved 2015-12-04.

Morris Dworkin. “Recommendation for Block Cipher Modes of Operation”. NIST. <http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf> Retrieved 2015-12.

Matthew Green. “How (not) to use symmetric encryption”. <http://blog.cryptographyengineering.com/2011/11/how-not-to-use-symmetric-encryption.html> Retrieved 2015-12.