Wireless Door Lock

Joshua N. Satterfield, Marcus E. Alexander

Charleston Southern University

Table of Contents

[Abstract 3](#_Toc441940024)

[Wireless Door Lock 4](#_Toc441940025)

[Design and Methodology 4](#_Toc441940026)

[Wi-Fi 4](#_Toc441940027)

[Protocol 5](#_Toc441940028)

[Encryption 5](#_Toc441940029)

[Challenges 5](#_Toc441940030)

[Implementation 5](#_Toc441940031)

[Initialization Vector 6](#_Toc441940032)

[Experiment 6](#_Toc441940033)

[Conclusion 7](#_Toc441940034)

[Future work 7](#_Toc441940035)

Abstract

We successfully created a system which allows you to wirelessly lock and unlock your door using your smartphone. It uses an Android based application on the phone and an Arduino based controller to turn the deadbolt. It uses a simple protocol to communicate commands over Wi-Fi allowing the system to integrate with others easily.

Wireless Door Lock

This project is about a wireless door lock system. The system intends modular, hackable and to be secured with symmetric encryption. Securing your home is important. Unsecured access is a vulnerability for your family. We do not won’t unsecured items guarding your house and lending themselves to be exploit vectors. We want the “Internet of Things” to be helpful and not hurt us. We also want the convenience of locking/unlocking our doors while we are away without sacrificing the security.

We need the communication to be secured for the safety of your own home and family so we will encrypt the communication between devices. The protocol is designed without unnecessary complexity, solely completing the daily use case. We need a platform that is small, power efficient and inexpensive. It needs to be small to fit easily inside a case attached to the outside of your deadbolt. It attaches on the part of the deadbolt inside your house but is not embedded within the door, which makes it even to install. The platform needs to be easy to program.

We first present the reasons behind the project. Then we discuss the difficulties of the research and methodology of how to overcome each obstacle. We then present our experiments and our findings in the Implementation section. Finally, we present our conclusion.

# Design and Methodology

We choose the UDP protocol for its simplicity. We send short packets with commands over Wi-Fi. We had tried using the XBee ZigBee, which is not designed to communicate using Wi-Fi. Then we tried using the XBee Wi-Fi however we could not get the module initialized. Then we started using the Arduino Wi-Fi shield. The shield comes with a useful network library. Detailed documentation for it is available as well. Using simple chat server example code we added functionality to preform actions when certain commands were received.

In this study, we investigate a scenario of a home environment. We used Arduino and Wi-Fi shield to accomplish this research.

## Wi-Fi

By using Wi-Fi 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 user’s 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 Wi-Fi. 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.

## Protocol

The protocol uses ASCII encoding for its commands, which are a single character. In version one of the protocol the Arduino would enact commands received immediately.

|  |  |  |
| --- | --- | --- |
| l | u | a |
| Lock | Unlock | Authorize |

In version one of the protocol the Arduino would enact commands received immediately. In version two first the phone must send the authorization command, then wait for a response. The Arduino responds with the AES initialization vector. The phone then encrypts its command using the IV and sends the packet to the Arduino.

## Encryption

### Challenges

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.

### Implementation

The encrypted packet will naturally be the length of the AES key, which is 128 bits. In the case of a message being longer than a block (the key length) an algorithm must be used to encrypt each block without becoming insecure. We choose the CBC block chaining algorithm because it uses a random IV to prevent replay attacks and it is more secure than EBC. EBC can leave noticeable patterns in the data outputted. The initialization vector is distinct from the key; it is not secret. Thus the IV can be sent over the wire.

### Initialization Vector

As the initial state for the AES algorithm the IV effects the end result end a manner different than the key which is used. If you encrypt the same message with the same key twice you will get the same result. Thus if someone records you sending an encrypted command signifying “unlock” they could replay this command at any time to manipulate your device. This is a vulnerability that can exist in encrypted systems. To prevent this a random IV is generated each time the module sends an authorization command. The 16 byte IV comes From the Arduino to the Smartphone and dealt with using javax.crypto.Cipher. The known key (which may be stored on the phone) is combined with the random IV using the AES algorithm and the message is sent back. The next authorization command from a phone will send back a different IV, so the same message will no longer make sense to the Arduino when decrypted using an old IV.

# Implementation

The implementation uses Android API 23 with java.net.DatagramSocket for the UDP socket on the smartphone. We used an asynchronous function for networking code to keep the UI from blocking. We also used the Arduino Wi-Fi library to receive information from the phone. When it receiving a command if it regards the lock it will turn the servo. The servo is wired to this Arduino this way:

|  |  |  |
| --- | --- | --- |
| Red | Black | White |
| 5V | GND | Pin 9 |

The lock command turns it to 100 degrees and the unlock command turn it to zero degrees.

Using AES CBC with a random IV we prevent the replay attack exploit vector. Using the Arduino microcontroller prototyping platform and Android Studio we built a system which can run on inexpensive and easily obtainable hardware.

One use case our platform does not currently handle is automatically open when you approach the door. Safeties would have to be put in place you solely want to look outside the door. This could be controlled from the mobile app. Handling the approach could possible be done by sending out a pilot signal informing the phone that there is a lock nearby, then the phone can reply with a command. The phone would need a program running in the background to react to this situation.

# Conclusion

This has been an insightful project. We have learned a lot about practical application of computing concepts. We found that using Wi-Fi communication provides an adequate layer for our networking to be built upon and that the speed of transmission was adequate.

# Future work

We are interested in is creating a standalone SMD designed to have the same capabilities as the Arduino and the Wi-Fi shield in a smaller form factor.

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.