

STRATH NAME

DOCTORAL THESIS

Thesis Title

Author:

John SMITH

Examiner:

Dr Man Page

Supervisor:

Dr. James SMITH

Phd Student:

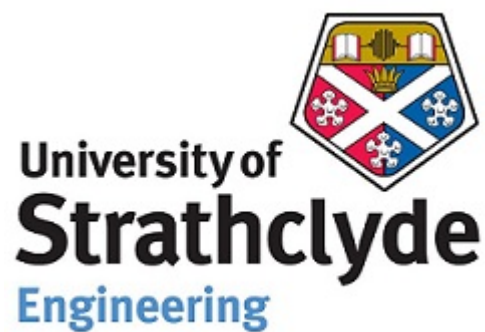
Greg

*A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy*

in the

Research Group Name
Department or School Name

March 11, 2016



Declaration of Authorship

I, John SMITH, declare that this thesis titled, “Thesis Title” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

STRATH NAME

Abstract

Faculty Name
Department or School Name

Doctor of Philosophy

Thesis Title

by John SMITH

The Thesis Abstract is written here (and usually kept to just this page).
The page is kept centered vertically so can expand into the blank space
above the title too...

Contents

Declaration of Authorship	ii
Abstract	iii
1 Introduction	1
2 Background	2
2.1 Cryptography	2
2.1.1 Asymmetric Key Encryption	2
2.1.2 Digital Signature	2
2.1.3 TweetNaCl	3
2.2 Types of attacks	4
2.2.1 Replay Attack	4
2.2.2 Man in the Middle Attack	4
2.2.3 Bit-Flipping Attack	4
2.2.4 Stream Cipher Attack	4
2.3 Technologies used	4
3 Implementation	5
3.1 IoT Platform	5
3.1.1 Arduino Due	5
3.2 Server Side	6
3.2.1 Java web app	6
3.2.2 PHP	6
3.3 NaCl	6
4 Strength Of Security	8
4.1 SubSection	8
4.2 Other SubSection	8
5 Results	9
5.1 Recording of temperature	9
5.2 Power Consumption!	9
5.3 Web App	9
5.4 Secure transmission of Public keys	9
6 Critical Evaluation	10
6.1 SubSection	10
6.2 Other SubSection	10
7 Conclusion	11
7.1 SubSection	11
7.2 Other SubSection	11
A Appendix Title Here	12

Chapter 1

Introduction

The Internet of Things or IoT is the concept of a huge network of physical objects connected and communicating to themselves and to the world wide web. Devices can include domestic appliances, buildings, cars. As it becomes a rapidly growing concept with over 50 million devices expected to be connected to the web by 2020, (need ref) the security of the transmissions of these devices is becoming a more and more pressing issue. IoT's main benefits are the remote control of devices and appliances, for the device to have the ability to send information about it's state, such as a vending machine reporting that it has run out of a certain item, and to allow the machines to be more automated and to work with other machines, like a home hub device that can turn on the lights and central heating when an occupant is arriving home, with the lights and heating not being connected to each other but to the central hub.

However IoT will be ultimately be useless if it is unsecure. IoT is an emerging field but there have already been some high profile security disasters. Ranging from relatively less serious problems such as some "hackers" been able to glean important wifi information from your internet connected lights and assuming control. There have been three examples of this with a Jeep Cherokee, Toyota Prius and Tesla. The hackers were able to control the accelerator, door locks and brakes, among other things. This highlights a very real problem that will only become more important. Too often security is an afterthought but it really needs to be built into products from the offset.

Within the last three years there have been three high profile security breaches on commercial cars, one on a Cherokee Jeep [1], a Toyota Prius [3] and a tesla [2]

With that in mind the subject of this report is the secure transmission of a users private home temperature data. If they have a system that monitors the temperature of all the rooms, that data can be used to figure out when they are likely to be home or not. So, using an Arduino Due as the base station that talks to the temperature sensors throughout the house, it takes the sensor data signs then encrypts it and sends it using an Ethernet Shield to a remote server.

Chapter 2

Background

2.1 Cryptography

Cryptography is the practise and study of techniques for secure communication in the presence of attackers. To do so, one can use encryption where by messages are encoded in such a way that only authorised parties or at least parties in possession of the keys can view them. There are two main ways of encryption Symmetric Key encryption and Public Key encryption. In Symmetric Key encryption both parties have the same key the which can encrypt and decrypt messages that are sent between them. The problem is that if Bob wants to send an encrypted message to Alice, he must get the secret key to her. Currently the most secure way for the transmission of secret keys is to hand them over in person, in private. This this project Asymmetric Key encryption and Digital signatures.

2.1.1 Asymmetric Key Encryption

To get round the problem of securing sending secret keys, one can use Asymmetric Key encryption has a secret key and a public key, the public key is generated out of the secret key and are therefore mathematically linked but it is computationally infeasible to calculate the secret key from the public key. This public key can be given out freely and is not a secret. So if Bob sends a message to Alice he encrypts the message with her private key and she can decrypt it with her secret key. This type of key encryption gets past the sharing key problem but it only stops attackers from reading the message. It does not prove the message was sent by a certain person or that the message has not been altered in transit.

2.1.2 Digital Signature

This is a mathematical scheme for proving message authenticity, message integrity and message non-repudation. Similarly to asymmetric key encryption a random private key is created with a corresponding public key. Double check."!! The algorithm takes in a message and a private key and using SHA-512 and? it produces a signature. If Alice signs a message in this way, Bob can use another algorithm to verify the message with the public key and signature.

2.1.3 TweetNaCl

NaCl or “Salt” is a simple to use high-speed library for authenticated encryption. It provides both Asymmetric and Symmetric encryption, It provides authentication and message integrity with SHA-512. The authors are Daniel J. Bernstein, Tanja Lange and Peter Schwabe but at points it relies on third part implementations for parts. The API is simple, having only a handful of methods but uses high speed, high security primitives. [/refhttps://labs.opendns.com/2013/03/06/announcing-sodium-a-new-cryptographic-library/](https://labs.opendns.com/2013/03/06/announcing-sodium-a-new-cryptographic-library/)

Unfortunately the library wouldn’t work completely on a Arduino, one problem is that there is no `/dev/random` and therefore no `randombytes()` which means that it can’t create keypairs in the usual way. Also, as mentioned later Arduino can’t use the C library as is, it needs to be converted into C++. Another problem is that the library is relatively quite large, the Arduino Due has at it’s disposal 512KB flash memory and the full library is 3MB. Fortunately the same creators along with Bernard van Gastel, Wesley Janssen and Sjaak Smetsers made TweetNaCl. Which is a tiny implementation of NaCl, still providing speed and security but with a significantly smaller code size 40KB. It retains the same protections against (from tweetNack-201409..) timing attacks, cache-timing attacks, has to branches depending on secret data and no array indices depending on secret data. In addition it is thread-safe and has no dynamic memory allocation. It is portable and easy to integrate, the library is easily added as it consists of two files, there is no complicated configuration to be set up or any dependencies on external libraries. Because of this compactness it is easier to read and understand it’s operation. Although not as fast as NaCl it is still fast enough for most applications. “Most applications can tolerate the 4.2 million cycles that OpenSSL uses on an Ivy Bridge CPU for RSA-2048 decryption, for example, so they can certainly tolerate the 2.5 million cycles that TweetNaCl uses for higher-security decryption (Curve25519).” TweetNaCl is still small after compilation at 11KB thus avoiding instruction cache misses. It is a full library and not a set of isolated functions for a simple NaCl application, only six functions are needed. `crypto_box` for public-key authenticated encryption; `crypto_box_open` for verification and decryption; `crypto_box_keypair` to create a public key; and similarly for signatures `crypto_sign`, `crypto_sign_open`, and `crypto_sign_keypair`. It is open source and the developers encourage it to be used as much as possible.

NaCl will move to Ed25519 signature system, what is that?

TweetNaCl encrypts messages by xor’ing them with the output of Bernsteins Salsa20?? stream cipher

NaCl `crypto_stream` uses Bernsteins X

Again, TweetNaCl uses SHA-512 as it’s hash function with the Ed25519 signature scheme and the code is simplified compared to the NaCl implementation

For asymmetric cryptography TweetNaCl uses Bernsteins’ Ed25519 elliptic curve Diffie-Hellman key exchange??

2.2 Types of attacks

2.2.1 Replay Attack

When this attack occurs the attacker replays a valid message. If Bob wants Alice to prove who she is and she duly provides some encrypted signature to prove so. Eve can capture that signature, She does not know what the signature is but she knows that it is a signature. She can then connect to Bob and use this message to pretend she is Alice. To prevent this attack, use an identifier that is only valid for one use, this can be session tokens or one-time passwords.

2.2.2 Man in the Middle Attack

In this attack there is an attack between two parties, Bob and Alice, who wish to communicate. The man in the middle, Eve, changes messages as they are in transit and manages to pretend that she is the person that the other thinks they are talking to. An example is if Alice asks for Bob's public key, Eve can capture that public key, replace it with her own and send that and because Alice has no way to prove that it is Bob's key or not she accepts it. So when Alice sends a message that has been encrypted with what she thinks is Bob's key, Eve can take it, decrypt it with her key, change the message then encrypt it with Bob's real public key. Which Bob receives and believes the message is from Alice.

2.2.3 Bit-Flipping Attack

This is where the attacker can change the cipher text in some way that cause a predictable change in the plain text. The attacker does not know exactly what the plain text is. If Alice was to send a message to Bob saying that she owes him £100. If Eve knows the format of the message, she can change the number at the end into £1000.

2.2.4 Stream Cipher Attack

stream cipher attacks, if the same key is used chosen plain text attack
Side channel attack, does NaCl protect a bit against this

2.3 Technologies used

In this project the Arduino was programmed using C++. The C++ was developed in the Arduino IDE. An Arduino Due, Arduino UNO, DS1820S temperature sensor with resistor and two Ethernet Shield R2 boards. On the server side there is an SQL server, PHP scripts that accept that Arduino data and put it in the SQL server. And a Java web application that uses JDBC.. to access the SQL server and outputs dynamic HTML when accessed. The Java code was developed using Eclipse Jee Mars.

Chapter 3

Implementation

3.1 IoT Platform

The basic concept of this platform is an Arduino Due that takes the current temperature of the room from a DS1820 temperature sensor. Then that data is signed and encrypted with TweetNaCL before being transmitted, using an Ethernet Shield, across to an SQL server. A web application takes the SQL data decrypts, checks the signature is valid then displays on a website.

graphic here pls

Why was the Due chosen, 32 bit?

DS1820 is a lost cost temperature sensor that is very accurate, 12 bits of precision? and is also low power. It can scavenge power from the data with the arduino and thus does not need it's own power source.

For the prototype, an Ethernet Shield was used as it is much cheaper than a WiFi shield but ultimately completes the same job. The shield is a simple way to connect arduinos to the internet. The shield used was the second revision, R2 and has a w500 ethernet controller.

What are some of the options for the base station, Due/MSP430? And for the internet connection Ethernet?WiFi shield?

3.1.1 Arduino Due

The Due is used in two separate ways in this project; first is the client to the java web app and second is a demonstration of secure public key transmission with another Arduino Uno and Ethernet shield. The first takes the raw hex values from the DS18S20 temperature sensor, example code for this is fairly common as the Arduino can use the OneWire library which is a proprietary protocol developed by Dallas Semiconductor. In essence the command 0x44 is sent to the device using `ds.write(0x44)`, where `ds` is an instance of the OneWire class, the sensor reads the internal ADC and copies the data to the scratchpad registers which can be read by sending the command 0xBE then using the command `ds.read()` which returns the value. Those values are put into an array which is then added to the end of an array where the first 32 bytes entries are 0x00. The message is signed and encrypted. It is much simpler to send a string across the internet than an array of bytes so the resulted signed cipher is converted into a string. Connection to the server with the IP and port number and a POST request is built with the data packaged inside.

3.2 Server Side

For the prototype, an Apache server, SQL server and Tomcat server was set up using XAMPP on a desktop. A Java web app was created as that is the language the writer has the most experience in and there are Java implementations of the TweetNaCl library, among other variations, available. The SQL table is a simple table that holds a key, timestamp and the signed and encrypted temperature sensor. (example?). The web app upon being accessed decrypts and checks the signature of each entry, using the keys that it has stored, in the table before converting the raw hex temperature data into more readable integers and displaying in a simple HTML table that can be accessed by the user. When the Arduino has data to send it will make a POST request to a PHP file on the Apache server which takes the data given to it and places it in the SQL server. (security flaw!)

3.2.1 Java web app

The Java Web app details what the server is do when it gets various types of request, be it get or post...(more on requests?). In this type of application you can dynamically printout all the HTML that will be used to make up the page. The usual HTML, head, body tags are printed at the top and the titles in the table are printed as well. (How it gets the keys?!). The web app uses JDBC to create a driver(idk man) to get the connection to the SQL database. Then using Java language it builds up a SQL query to take out all the values from the database and executes that. This puts all the table entries into a result set and the app cycles through that results set getting the relevant information out. The signed and encrypted hex is encoded as string and some leading zeros are lost in the conversion from byte array to string in the Arduino so these are added now before the string is converted back into a byte array. There is a try catch around the `crypto_box_open` and `crypto_sign_open` method so the server doesn't crash if one result set has been broken. Following this is the conversion from hex into integer for the user to read (how does it do it?) and finally the values are written to the browser along with the ending html tags.

3.2.2 PHP

PHP is a server-side scripting language

There are two files in the server, `connect.php` and `add.php`. The Arduino makes a post request to the `add.php` which effectively just calls it and the first thing the `add` file does is call `connect` which has the server details and makes creates a connection. Following that there is a SQL query that inserts the values sent in the post request into the appropriate table entries then close the connection.

3.3 NaCl

The TweetNaCl library as it stands in it's original form is not compatible with Arduinos. The C library compiles without errors but the compiler warns that the TweetNaCl method names are undefined and as a result do not perform their tasks. The method simply returns random numbers, it is

possibly that it is trying to access some area of memory and simply returns whatever it finds.(ask greg/james). It is not understand why this is the case but it is a simple case of converting the library into C++ syntax. With a header file that has the main methods used in the project and the #defines and a cpp file with the TweetNaCl code. This is added in the same way to the Arduino IDE and in the code an instance of the class is created and methods are accessed with the dot operator.

The keypair, crypto_sign, crypto_box and equivalent opens were used. These are simple to use, abstracted methods that make this library easy to use. For the encryption the method needs the message to be encrypted which needs to have the first 32 bytes be zero, an empty array that needs to be at least the size of the message with the leading zeros, the length of the message, the nonce, arduino public key and the servers private key. This will reveal the temperature data with the signature. To remove the signature, the crypto_sign_open method needs the server secret signature key and the signed cipher array.

Chapter 4

Strength Of Security

4.1 SubSection

Sign then encrypt to get round little, subtle pit falls like the if the message is encrypt first then an attacker could peel off the signature then attach their own. But if it's encrypt last then they have no access to the message without the secret key.

can protect against the attacks mentioned earlier because...

4.2 Other SubSection

Chapter 5

Results

[link to objectives](#)

5.1 Recording of temperature

Using the thing it can record temp, sign, encrypt to ward off the attacks
mention in the background. The temperat

5.2 Power Consumption!

5.3 Web App

Users can access the temperature table anywhere

5.4 Secure transmission of Public keys

Two arduino Dues in the same network can send their public keys securely
and a person has checked that
show lots of pictures?

Chapter 6

Critical Evaluation

6.1 SubSection

6.2 Other SubSection

Chapter 7

Conclusion

7.1 SubSection

security measures will never be fully secure so all we can do is make it not worth the attackers while. At the moment there are so many devices that simply have no protection (web security “hack” example pls), so by having even the minimal amount of protection you can dissuade would be attackers because they have plenty of easier targets. Or make it so that the reward of breaking into your system simply is not worth the effort.

7.2 Other SubSection

Appendix A

Appendix Title Here

Write your Appendix content here.

Bibliography

- [1] Andy Greenburg. *Hackers Remotely Kill a Jeep on the Highway—With Me in It*. 2015. URL: <http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/>.
- [2] Jeremy Korzeniewski. *Tesla Model S successfully hacked by Zhejiang University team*. 2014. URL: <http://www.autoblog.com/2014/07/18/tesla-model-s-successfully-hacked-by-zhejiang-university-team/>.
- [3] Jill Scharr. *Hackers Hijack Prius with Mac Laptop*. 2013. URL: <http://www.tomsguide.com/us/hackers-hijack-prius-with-laptop,review-1797.html>.