Data integrity on Internet of Things

Window Latch Project

ITM-498-15.16M Data Integrity in Internet of Things Devices

Bruno Augusto

Grasiele Brustolom

João Calmon

Matheus Pita

Illinois Institute of Technology

**Abstract**

This research project about data integrity on internet of things shows that, with a software solution it is possible to keep data secure and communicate over the internet, in this specific case, using a wired network. Using the four pillars of information security (confidentiality, integrity, availability, non-repudiation) as the main guideline, this project aims for keeping data secure checking if the receiver receives the correct information, if the data was not intercepted and changed during the communication and also encrypting everything, so the data cannot be intercepted and read during the transference. Using the AES/Rijndaela 16 bit string can be encrypted using a secret key that stays with the system owner, the system can encrypt several kinds of information, such as sensor readings, plain text and many more kinds of input. In this project, a sensor will be used to show how this software solution can be applied.

**Introduction**

The internet of things is becoming more common on our everyday lives each day. Due its vast usage, this technology has become present in companies, houses and public spaces in order to provide more accessibility and interactivity. Although this technology has many advantages, there is a critical problem that must be carefully handled: the data security.

Suppose that you have in your house a device that identifies a certain number on a card that allows you to enter in your house. Can you imagine what would happen if someone steals this information? Data stealing can be very harmful if there is not an efficient security system.

This project aims to implement a security system that allows two Arduinos communicate safely. Currently, there is some limitations that make difficult to implement a complex security system. Our objective is to implement this system respecting the device limitation and provide a communication that implements the main principles of data security.

About IoT …………………………………………………………………………………………………………………… 5

Scope …………………………………………………………………………………………………………………… 6

Restrictions …………………………………………………………………………………………………………………… 6

Project Brief …………………………………………………………………………………………………………………… 6

The AES Cryptography ………………………………………………………………………...………………………..…… 7

The Layers …………………………………………………………………………………………………………………… 7

CRC Checking method ….…………………………………………………………………………………………………… 7

Handshake prototype ……………………….……………………………………………………………………………… 8

Future Work …………………………………………………………………………………………………………………… 9

Real life application ………………………………………………………………………………………………………… 9

Parts Invoice …………………………………………………………………………………………………………………… 10

Shopping List ………………….………………………………………………………………………………..… 11

Conclusion ………………………………………………………………………………………………………………….. 12

References …………………………………………………………………………………………………………..……… 13

Appendices

Codes

Server ………………………………………………………………………………………………………. 14

Client ………………………………………………………………………………………………………. 19

Diagrams

Arduino Circuit diagram ………………………………………………………………………….. 25

Eletrical diagram …………………………………………………………………………….. 26

Data integrity on Internet of Things

**About IoT:**

The Internet of Things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with [unique identifiers](http://whatis.techtarget.com/definition/unique-identifier-UID) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

A [thing](http://whatis.techtarget.com/definition/thing-in-the-Internet-of-Things), in the Internet of Things, can be a person with a heart monitor implant, a farm animal with a [biochip transponder](http://searchsecurity.techtarget.com/definition/injectable-ID-chip), an automobile that has built-in [sensors](http://whatis.techtarget.com/definition/sensor) to alert the driver when tire pressure is low or any other natural or man-made object that can be assigned an [IP address](http://searchunifiedcommunications.techtarget.com/definition/Internet-Protocol) and provided with the ability to transfer data over a network.

This field of study has been growing exponentially, reaching an enormous range of device, from animals with transponders, as said before, to the smartphone that we carry with us every day.

Using an Arduino board, our project goal is to establish a safe, relying on the AES cryptography method to encrypt data and send it over the network, trying to keep it safe to reach the device that is set to receive, making sure that the data received is the correct data, and what was expected to be received.

As Arduino is open source, we were allowed to use open source libraries, available on github.com, designed to work within the small capacity that Arduino gives to the developers.

On this project, we are going to build an embedded system, which is nothing more than a system made to work on a specific device, created and made to work for that single purpose. And it can be either programmed or not.

**Scope:**

The final result will be two Arduino boards, with pre-loaded applications, designed to communicate with each other, using the cryptography library set to work with the applications, and a sensor as the input for the system.

**Restrictions:**

* Both of the Arduino boards (client/server) will need to be working with the Ethernet and a power source (if not working with a computer at the same time).
* The Arduino board memory capacity (the standard version memory size is 32k bytes, being used to store the libraries and codes to be used).
* The encryption key should be kept in secret, set by the main user or the one responsible for the setup and solution configuration.
* Using the Ethernet libraries because the Wi-Fi and Wi-Fi 101 libraries could use more memory than the current Arduino capacity.

**Project Brief:**

This project, created on the Internet of Things base field, is meant to connect two embed systems, using wireless communication (Wi-Fi) or a cable based network(Ethernet), send and receive data over the network, which needs to be encrypted to keep data secure and avoid unwanted access. In this scope, a small light sensor will be used, but it will still be possible to apply this to a bigger scope, with more complex data to be sent, with just a few modifications needed.

**The AES cryptography**

Also known as Rijndael, AES is a secret key block cipher created to replace the DES method. The AES method allows key sizes of 128, 192 and 256 bits. The algorithm consists of a certain number of encryption rounds (10 for 128 bits, 12 for 192 bits and 14 for 256 bits). Each round uses a round key, derived from the original key, going through the layers in the process.

**The layers:**

1. The **ByteSub** Transformation (BS): This non-linear layer is for resistance to differential and linear cryptanalysis attacks.
2. The **ShiftRow** Transformation (SR): This linear mixing step causes diffusion of the bits over multiple rounds.
3. The **MixColumn** Transformation (MC): This layer has a purpose similar to ShiftRow.
4. **AddRoundKey** (ARK): The round key is XORed with the result of the above layer.

And as it is said, the process is repeated based on the key size.

This library was chosen based on the Arduino capacity and the algorithm complexity, in order to keep the process simple and safe.

**CRC checking method:**

Since the Arduino model used on this project contains a significant limited memory, the usage of an efficient and simple implementation of an error detecting code became required.

An error detecting code basically detects corruptions on the data received, giving assurance of the data integrity. The integrity is one of the main pillars of the data security concept, which makes the error detecting code highly required in order to establish a safe communication.

According to researches, was concluded that the CRC (cyclical redundancy check), a widely used error detecting code, would be the best option to be implemented on this project. The CRC is a binary sequence generated through a fixed polynomial. This binary sequence is appended to the data and sent to the receiver. The receiver use the same polynomial to calculate a new binary sequence according to the bits of the message received. The result of this new CRC tells the receiver whether the data received was corrupted or not.

**Handshake prototype:**

The handshake system should ensure that the connection is set, working with the correct device. The automatically system sets the communication parameters, rules to be followed when connecting to a foreign device. The handshaking system establishes an acceptable communication way between both ends, keeping is trustful and safe. Some of the parameters involved on the handshaking are communication channel, transfer rate, coding alphabet, parity and interruption procedure. The simplest handshake system would go from the receiver sending “I received your message and I am ready for the next one” to more complex protocols.

On this project, because of the hardware limitation, a simple prototype of the handshake system was implemented. It happens when some information is sent and with it a “tag” is included on the package. That “tag” should be exclusive to the referred board. When the server receives the package, it checks the “tag” and then, if it is correct, the package is decrypted, if not, the sender receives an error message.

**Future Work:**

The goal for the future is to add more security layers, as the software evolves on a really fast pace, just the AES encryption and the CRC checking should not be enough. But keeping in mind that the hardware should also receive future upgrades, the software and the hardware should be updated to the latest update, in order to maintain the usability of the system and integrity of the data. The hand-shake security layer can also be added or in this particular project case, just improved, as the current Arduino capacity does not allows much more libraries to be added.

**Real life application:**

As the Internet of Things field of study grows on a steady pace, more things get involved on the process, going from a smart TV to software and sensors used by big companies, where a single mistake could be extremely dangerous. With this scenario in mind, this application could be vital to a company, to make sure that the data being sent is trustful and reliable. And using the three pillars of data security (confidentiality, integrity, and availability) as our main quality standard, the project needs to ensure that the correct data will be sent to the correct destination and with no interference or interception.

**Parts Invoice:**

**Assembly List**

| **Label** | **Part Type** | **Properties** |
| --- | --- | --- |
| LED2 | Red (633nm) LED | package 1206 [SMD]; color Red (633nm) |
| LED3 | Green (565nm) LED | package 1206 [SMD]; color Green (565nm) |
| LED4 | Blue (505nm) LED | package 1206 [SMD]; color Blue (505nm) |
| LED5 | Yellow (585nm) LED | package 1206 [SMD]; color Yellow (585nm) |
| Part1 | Arduino Uno (Rev3) | type Arduino UNO (Rev3) |
| Part2 | Arduino Uno (Rev3) | type Arduino UNO (Rev3) |
| Part3 | Arduino Ethernet Shield (Rev3) | type Ethernet Shield (Rev3) |
| Part4 | Arduino Ethernet Shield (Rev3) | type Ethernet Shield (Rev3) |
| R1 | 1kΩ Resistor | package 0805 [SMD]; tolerance ±5%; resistance 1kΩ |
| R2 | 1kΩ Resistor | package 0805 [SMD]; tolerance ±5%; resistance 1kΩ |
| R3 | 1kΩ Resistor | package 0805 [SMD]; tolerance ±5%; resistance 1kΩ |
| R4 | 1kΩ Resistor | package 0805 [SMD]; tolerance ±5%; resistance 1kΩ |
| R5 | PHOTOCELL | package photocell-kit; variant pth-kit |
| R6 | 1kΩ Resistor | package 0805 [SMD]; tolerance ±5%; resistance 1kΩ |

**Shopping List**

| **Amount** | **Part Type** | **Properties** |
| --- | --- | --- |
| 1 | Red (633nm) LED | package 1206 [SMD]; color Red (633nm) |
| 1 | Green (565nm) LED | package 1206 [SMD]; color Green (565nm) |
| 1 | Blue (505nm) LED | package 1206 [SMD]; color Blue (505nm) |
| 1 | Yellow (585nm) LED | package 1206 [SMD]; color Yellow (585nm) |
| 2 | Arduino Uno (Rev3) | type Arduino UNO (Rev3) |
| 2 | Arduino Ethernet Shield (Rev3) | type Ethernet Shield (Rev3) |
| 5 | 1kΩ Resistor | package 0805 [SMD]; tolerance ±5%; resistance 1kΩ |
| 1 | PHOTOCELL | package photocell-kit; variante pth-kit |

**Conclusion:**

According to our results, was concluded that dealing with limited memory affects directly on the code quality. To implement a more complete security system, it is required a good processing capacity and a considerable memory space. It is important to perceive that dynamic allocation of memory might be a acceptable option in order to improve the memory usage.

The project was adapted to give priority to data integrity and encryption code. For this reason, the usage of a WIFI board was rejected and a wired connection was implemented. Although mobility becomes a problem on a wired connection, it provides a low latency and lower packet loss.

In conclusion, the project achieved its aims implementing a security system that applies all the basic principles of data security, considering the hardware limitations. The memory issues were handled as far as they were identified, although some new improvements can be possible under new researches.

**References:**

* Trappe, W. (n.d.). *Introduction to cryptogrphy with coding theory* (2nd ed.). Pearson.
* Arduino - Introduction. (n.d.). Retrieved July 10, 2016, from https://www.arduino.cc/en/Guide/Introduction
* What is embedded system? - Definition from WhatIs.com. (n.d.). Retrieved July 15, 2016, from http://internetofthingsagenda.techtarget.com/definition/embedded-system
* What is Internet of Things (IoT)? - Definition from WhatIs.com. (n.d.). Retrieved July 5, 2016, from http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT
* Landman, D. (n.d.). AES Library download. Retrieved June 30, 2016, from <https://github.com/DavyLandman/AESLib>
* CRC checking library. (n.d.). Retrieved July 10, 2016, from <https://github.com/vinmenn/Crc16/blob/master/Crc16.h>

**Codes:**

**-Server:**

#include <AESLib.h>

#include <SPI.h>

#include <Ethernet.h>

#include <Crc16.h>

#include <stdio.h>

// Enter a MAC address and IP address for your controller below.

// The IP address will be dependent on your local network.

byte mac[] = { 0x90, 0xA2, 0xDA, 0x0F, 0x4C, 0xD0 };

//the value of the Indetification ID of the other client

char valuetoComp= 'M';

// ip address of the server if the DHCP atrubuition is not working

IPAddress ip(192, 168, 1, 177);

Crc16 crc;

// telnet defaults to port 23

EthernetServer server(23);

boolean gotAMessage = false; // whether or not you got a message from the client yet

//key to ecript and decript using AES

uint8\_t key[] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};

void setup()

{

//pin of the data input (sensor)

pinMode(A0, INPUT);

// Open serial communications and wait for port to open:

Serial.begin(9600);

// start the Ethernet connection:

Serial.println("Trying to get an IP address using DHCP");

if (Ethernet.begin(mac) == 0)

{

Serial.println("Failed to configure Ethernet using DHCP");

}

// print your local IP address:

Serial.print("My IP address: ");

ip = Ethernet.localIP();

for (byte thisByte = 0; thisByte < 4; thisByte++)

{

// print the value of each byte of the IP address:

Serial.print(ip[thisByte], DEC);

Serial.print(".");

}

Serial.println();

// start listening for clients

server.begin();

}

void loop()

{

// wait for a new client:

EthernetClient client = server.available();

//array data to recive the information from the client

char data[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

// when the client sends the first byte, say hello:

if (client)

{

if (!gotAMessage)

{

Serial.println("We have a new client");

client.println("Hello, client!");

gotAMessage = true;

}

// read the bytes incoming from the client:

char thisChar;

for(int i=0;i<19;i++)

{

thisChar = client.read();

Serial.print("Readed: ");

Serial.println(thisChar);

data[i]=thisChar;

if(i==18 && data[18]== valuetoComp)

{

aes128\_dec\_single(key, data);

Serial.print("desencript:");

Serial.println(data);

}

else if(i==18)

{

Serial.println("Device id does Not match!");

}

//use of the same function to generate the CRC; if the CRC is right the function should return 0

unsigned short verify = calcrc((char\*)data, 18);

//If CRC=0 indicates that the data integrity was guaranteed

if(verify==0)

{

Serial.println("\nTHE DATA WASNT CORRUPTED.");

}

else

{

Serial.println("\nERROR: THE DATA WAS CORRUPTED.");

}

// echo the bytes back to the client:

server.write(thisChar);

// echo the bytes to the server as well:

Ethernet.maintain();

}

}

}

int calcrc(char \*ptr, int count)

{

int crc;

char i;

crc = 0;

while (--count >= 0)

{

crc = crc ^ (int) \*ptr++ << 8;

i = 8;

do

{

if (crc & 0x8000)

crc = crc << 1 ^ 0x1021;

else

crc = crc << 1;

} while(--i);

}

return (crc);

}

**-Client:**

#include <SPI.h>

#include <Ethernet.h>

#include <AESLib.h>

#include <Crc16.h>

char data[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

byte datab[]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

//the value of the Indetification ID of the other client

char ValueOfId ='M';

// Enter a MAC address and IP address for your controller below.

// The IP address will be dependent on your local network:

byte mac[] = {

0x90, 0xA2, 0xDA, 0x0F, 0x48, 0x62

};

// ip address of the server if the DHCP atribuition is not working

IPAddress ip(192, 168, 1, 177);

// Enter the IP address of the server you're connecting to:

IPAddress server(192, 168, 1, 163); //por endereco do servidor

//atualiza com ip do servidor wifi

Crc16 crc;

// Initialize the Ethernet client library

// with the IP address and port of the server

// that you want to connect to (port 23 is default for telnet;

EthernetClient client;

//key to ecript and decript using AES

uint8\_t key[] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};

int i = 0;

void setup() {

//pin of the data input (sensor)

pinMode(A0, INPUT);

// start the Ethernet connection:

Ethernet.begin(mac); // Wifi beging

// Open serial communications and wait for port to open:

Serial.begin(9600);

// give the Ethernet shield a second to initialize:

delay(1000);

Serial.println("connecting...");

// if you get a connection, report back via serial:

if (client.connect(server, 23)) {

Serial.println("connected");

} else {

// if you didn't get a connection to the server:

Serial.println("connection failed");

}

}

void loop() {

// if there are incoming bytes available

// from the server, read them and print them:

if(readsens()==1){

// as long as there are bytes in the serial queue,

// read them and send them out the socket if it's open:

while (Serial.available() > 0) {

}

}

// if the server's disconnected, stop the client:

if (!client.connected()) {

Serial.println();

Serial.println("disconnecting.");

client.stop();

// do nothing:

while (true);

}

}

int calcrc(char \*ptr, int count)

{

int crc;

char i;

crc = 0;

while (--count >= 0)

{

crc = crc ^ (int) \*ptr++ << 8;

i = 8;

do

{

if (crc & 0x8000)

crc = crc << 1 ^ 0x1021;

else

crc = crc << 1;

} while(--i);

}

return (crc);

}

//this method gets the value from the sensor, maps from 0 to 9 and converts to a character, entering it into an array

int readsens(){

Serial.println("Enter '1' command to start the readings");

char readed = Serial.read();

while(readed!='1'){

readed = Serial.read();

}

if(readed == '1')

{

i = analogRead(A0);

i = map(i, 0, 1024, 0, 9);

switch(i)

{

case 0: data[0] = '0';

break;

case 1: data[0] = '1';

break;

case 2: data[0] = '2';

break;

case 3: data[0] = '3';

break;

case 4: data[0] = '4';

break;

case 5: data[0] = '5';

break;

case 6: data[0] = '6';

break;

case 7: data[0] = '7';

break;

case 8: data[0] = '8';

break;

case 9: data[0] = '9';

break;

default: Serial.println ("ERROR! Sensor not found");

break;

Serial.print("Sensor: ");

Serial.println(data);

Serial.println("");

for(int a=0;a<16;a++){

datab[a]=data[a];

}

unsigned short value = calcrc((char\*)datab, 16);

//Adds two CRC bytes on the end of the array

data[16] = highByte(value);

data[17] = lowByte(value);

//encrypts the data that was readed

aes128\_enc\_single(key, data);

data[18]=ValueOfId;

if (client.connected()) {

client.print(data);

}

for(int j=0;j<16;j++){

datab[j]=0;

}

for(int k=0;k<16;k++){

data[k]=0;

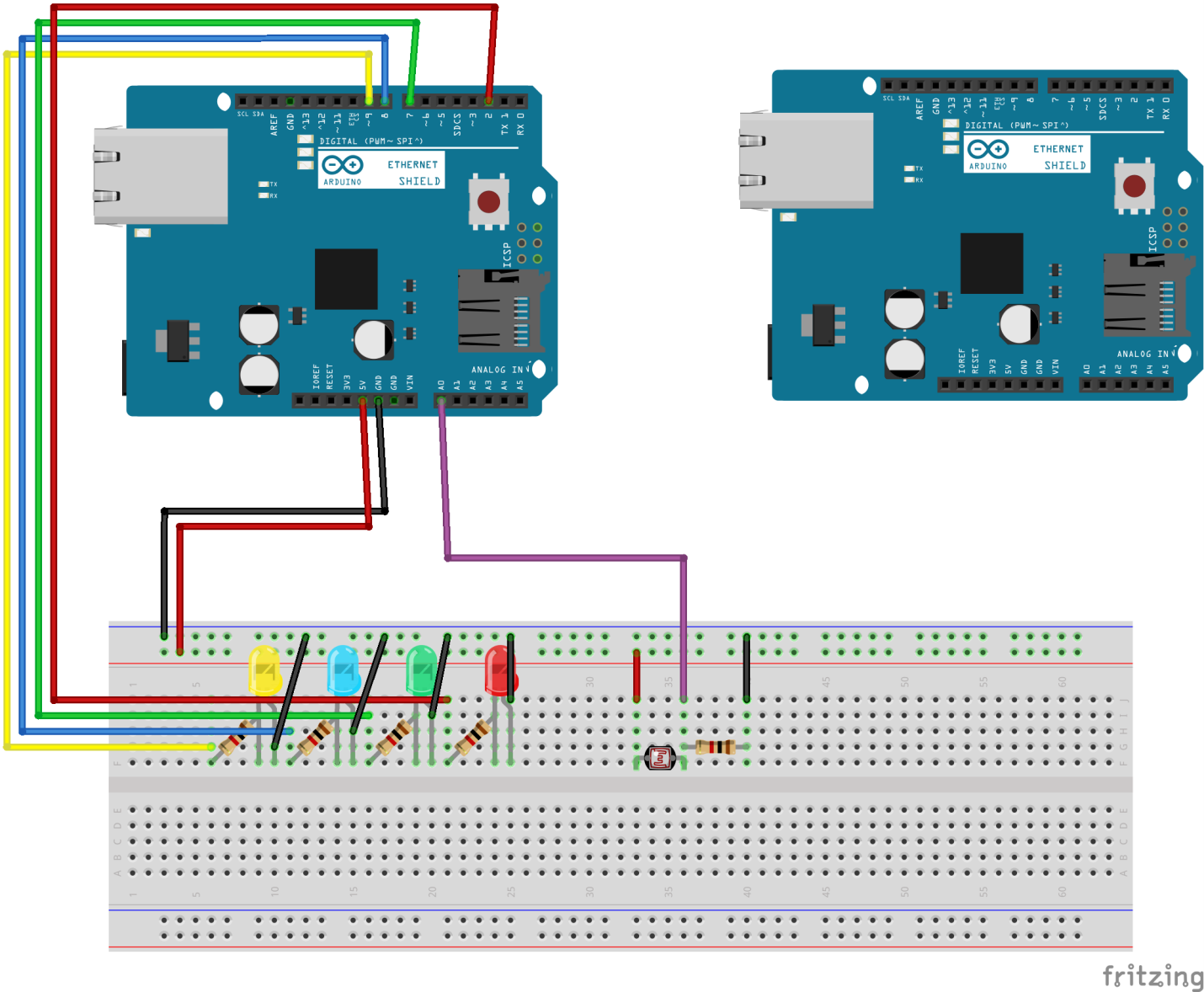
}

return 1;

}}

**Appendices:**

**Arduino circuit diagram:**

****

**Electrical circuit diagram:**

