SoT- Security of Things

Akash Sharma - 201551097

Prashant Chaurasia - 201551088

Shivvrat Arya - 201551059

Vikas Rajput - 201551064

Under the supervision of:

Dr. Keyur Parmar

January 4, 2018

ABSTRACT

Internet of things is currently a buzzword in the industry of Computer Science and Technology. Most of you should have heard about it from some talk's articles or even magazines. The communication is between machines to machines, machines to environment and even machines to automobiles, the Internet of Things is present everywhere. In the recent times there are quite a few attacks on IOT devices that have led to very bad effects on the online community ranging from DDoS attacks to service attacks. Since the number of IOT devices are increasing, it also increases the use of these devices in hacking or any other type of attack. We can imagine IOT as large number of intelligent, inter-connected things that will have an effect on every aspect of our day-to-day life. The main reason of rise of internet of things. The computing power that hackers want are easily available in IOT attacks since these devices are present in every corner of the world and they are in a huge quantity.

INTRODUCTION

IOT security is the area that is concerned with the security of the devices that are connected to internet, i.e. IOT devices. Internet of Things involves the collection of all the devices that are connected to the vast network of internet. These devices can easily communicate with each via internet and also share the data. The main idea of this project report is to show how we can create a secure connection between these devices and thus help their owners to share the information in a secure manner. The main aim of this project is to show how we can create a secure connection on which the devices can share anything they want to, for example, we are sharing the temperature data via this connection. Even if the connection is insecure, the machines can easily communicate with each other since if a hacker tries to temper with the data, it would not happen since the data is encrypted and it is not easy to decrypt it without key.

PROBLEM

Statement

To design and build a specification for the encryption of electronic data that dynamically moves between different IOT devices. We want to code an efficient algorithm that can encrypt and de-crypt data in very less amount of time.

Description

This problem can be divided into following sub problems:

- Coding a cryptographic algorithm: Create an efficient algorithm that can be used to encrypt and decrypt data that is transferred between different type of IOT devices and networks.
- Make Temperature Sensing IOT device: Create a device that can detect temperature and send it over internet using Raspberry Pi.
- Make Android Application: Create an android application that notifies the user that about any significant change in temperature.
- Temperature Sensing by Raspberry Pi: DHT11 Temperature Sensor that transmits its data to the Raspberry Pi that then sends it to the Android Application does Temperature sensing.

ASSUMPTIONS

For this phase, following assumptions will be there.

- The error given by the Sensor is close to zero.
- The temperature in which the device is kept within the range of the temperature sensor readings.
- The Raspberry Pi should have enough computing power to use Advanced Encryption Algorithm within given time constraints.

THE PRESENT INVESTIGATION

Goal

To design a system which makes the IOT devices secure and is implementable on all the exiting IOT devices.

Devices Used

- Raspberry Pi
- DHT11 Temperature Sensor
- Electrical Components
- Android Device

Protocol Used

Consider Device A and Device B.

1. Assign Device A and Device B unique IDs.
2. Store the ID of A in B and ID of B in A (Both devices know their ID).
3. Collect the data you want to send from Device A to Device B.
4. Encrypt the Data (we used AES for encryption). Encrypt the Key used to encrypt the data using the Device ID.
5. Send the encrypted Data over the Internet from A to B.
6. At B, Use the ID of A to decrypt the Key.
7. Use the Key to decrypt the data.
8. Data securely received.

About AES

AES is an encryption algorithm that is the most used algorithm in the cryptography world. The ciphers are of 128, 192 or 256 bits. The data block is of 128 bits. Currently it is been said that this AES algorithm cannot be broken if we use the current computing power also. It would take many years before this AES is decrypted. The features of AES are as follows:

- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

Each round in AES comprise of four sub-processes.

1. Byte Substitution
2. Shift Rows
3. Mix Columns
4. Add round key

Demonstration

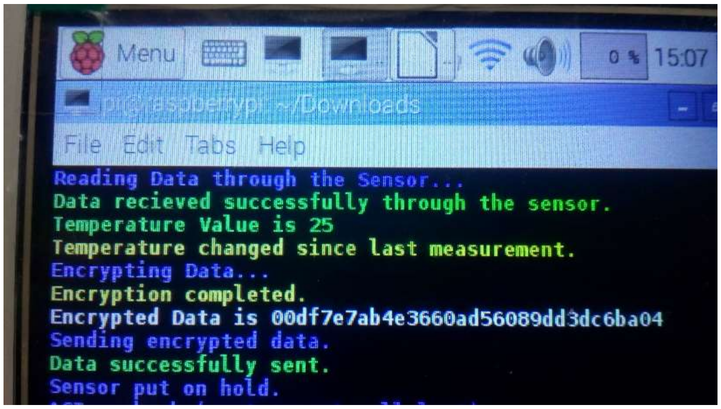
For demonstrative purpose of our solution, we created a **Temperature Keeper** This device reads the temperature of the room at regular intervals and sends an alert Notification to user's mobile device whenever there is a major temperature change.

Components of Temperature Keeper

1. **Hardware**: Raspberry Pi DHT11 setup to scans temperature at regular interval and send an encrypted alert to the server whenever there was major change in temperature.
2. **Server:** This acts as a medium for contact between Hardware and Mobile App. We used Firebase for our project.
3. **Mobile App (User):** Mobile App receives the data from server and displays notification alert accordingly.

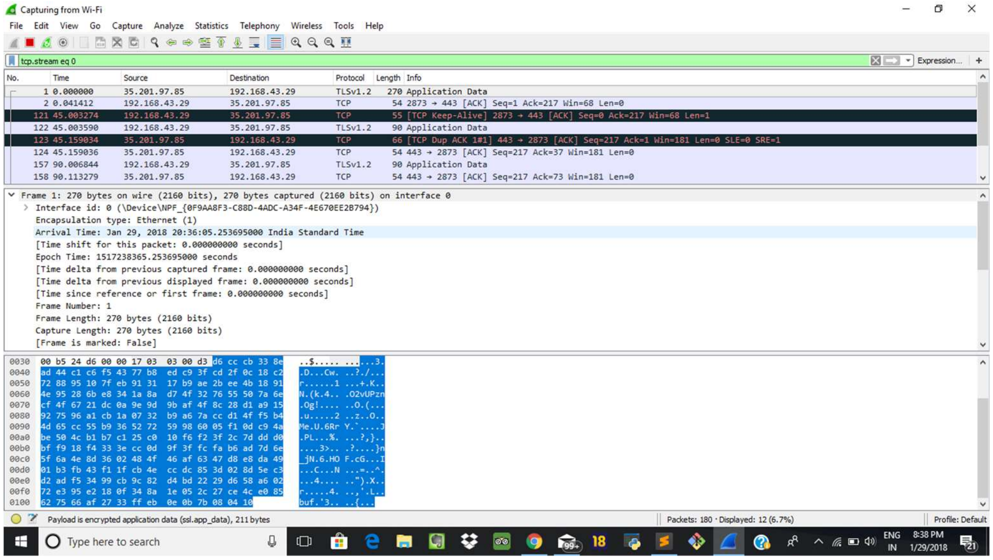
RESULTS AND DISCUSSIONS

Firstly, we were able to take temperature from the temperature sensor and send it to the Firebase server with the help of Raspberry Pi. Before sending this data, we encrypted it thus making the communications between the respected IOT devices (Raspberry Pi and Android Phone) safe and secure. Then we took this from Firebase server to the android phone and decrypted it there using the same key and thus we were able to show it on the android device.



```
Reading Data through the Sensor...
Data recieved successfully through the sensor.
Temperature Value is 25
Temperature changed since last measurement.
Encrypting Data...
Encryption completed.
Encrypted Data is 00df7e7ab4e3660ad56089dd3dc6ba04
Sending encrypted data.
Data successfully sent.
Sensor put on hold.
```

Figure 1: Showing the whole process happening on Device A.



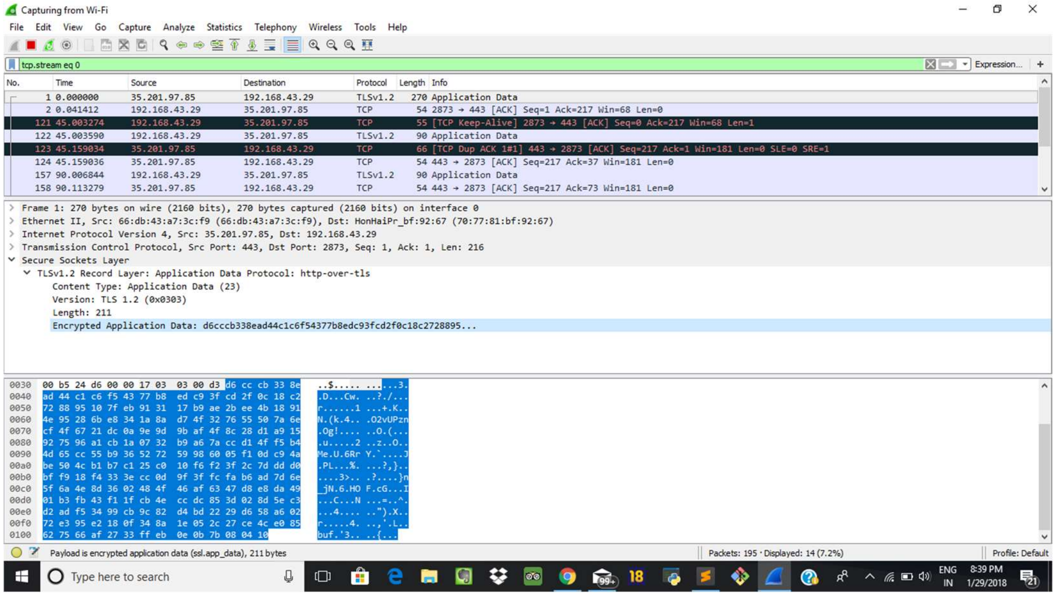
```
No.  Time        Source          Destination     Protocol Length Info
  1 0.000000     35.201.97.85    192.168.43.29   TLSv1.2  270 Application Data
  2 0.041412     192.168.43.29   35.201.97.85    TCP       54 2873 → 443 [ACK] Seq=1 Ack=217 Win=68 Len=0
121 45.003274    192.168.43.29   35.201.97.85    TCP       55 [TCP Keep-Alive] 2873 → 443 [ACK] Seq=0 Ack=217 Win=68 Len=1
122 45.003590    192.168.43.29   35.201.97.85    TLSv1.2   90 Application Data
123 45.159034    35.201.97.85    192.168.43.29   TCP       66 [TCP Dup ACK 1#1] 443 → 2873 [ACK] Seq=217 Ack=1 Win=181 Len=0 SLE=0 SRE=1
124 45.159036    35.201.97.85    192.168.43.29   TCP       54 443 → 2873 [ACK] Seq=217 Ack=37 Win=181 Len=0
157 90.006844    192.168.43.29   35.201.97.85    TLSv1.2   90 Application Data
158 90.113279    35.201.97.85    192.168.43.29   TCP       54 443 → 2873 [ACK] Seq=217 Ack=73 Win=181 Len=0

Frame 1: 270 bytes on wire (2160 bits), 270 bytes captured (2160 bits) on interface 0
  Interface id: 0 (\Device\NPF_{0F9AA8F3-C88D-4ADC-A34F-4E670EE2B794})
  Encapsulation type: Ethernet (1)
  Arrival Time: Jan 29, 2018 20:36:05.253695000 India Standard Time
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1517238365.253695000 seconds
  [Time delta from previous captured frame: 0.000000000 seconds]
  [Time delta from previous displayed frame: 0.000000000 seconds]
  [Time since reference or first frame: 0.000000000 seconds]
  Frame Number: 1
  Frame Length: 270 bytes (2160 bits)
  Capture Length: 270 bytes (2160 bits)
  [Frame is marked: False]

0030  00 b5 24 d6 00 00 17 03  03 00 d3 d6 cc cb 33 8e   ..$..... ......3.
0040  ad 44 c1 c6 f5 43 77 b8  ed c9 3f cd 2f 0c 18 c2   .D...Cw. ..?./...
0050  72 88 95 10 7f eb 91 31  17 b9 ae 2b ee 4b 18 91   r......1 ...+.K..
0060  4e 95 28 6b e8 34 1a 8a  d7 4f 32 76 55 50 7a 6e   N.(k.4.. .O2vUPzn
0070  cf 4f 67 21 dc 0a 9e 9d  9b af 4f 8c 28 d1 a9 15   .Og!.... ..O.(...
0080  92 75 96 a1 cb 1a 07 32  b9 a6 7a cc d1 4f f5 b4   .u.....2 ..z..O..
0090  4d 65 cc 55 b9 36 52 72  59 98 60 05 f1 0d c9 4a   Me.U.6Rr Y.`....J
00a0  be 50 4c b1 b7 c1 25 c0  10 f6 f2 3f 2c 7d dd d0   .PL...%. ...?,}..
00b0  bf f9 18 f4 33 3e cc 0d  9f 3f fc fa b6 ad 7d 6e   ....3>.. .?....}n
00c0  5f 6a 4e 8d 36 02 48 4f  46 af 63 47 d8 e8 da 49   _jN.6.HO F.cG...I
00d0  01 b3 fb 43 f1 1f cb 4e  cc dc 85 3d 02 8d 5e c3   ...C...N ...=..^.
00e0  d2 ad f5 34 99 cb 9c 82  d4 bd 22 29 d6 58 a6 02   ...4.... ..").X..
00f0  72 e3 95 e2 18 0f 34 8a  1e 05 2c 27 ce 4c e0 85   r.....4. ..,'.L..
0100  62 75 66 af 27 33 ff eb  0e 0b 7b 08 04 10         buf.'3.. ..{...

Payload is encrypted application data (ssl.app_data), 211 bytes    Packets: 180 · Displayed: 12 (6.7%)    Profile: Default
```

Figure 2: The packet that is sent is captured through Wireshark.


Capturing from Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...

tcp.stream eq 0

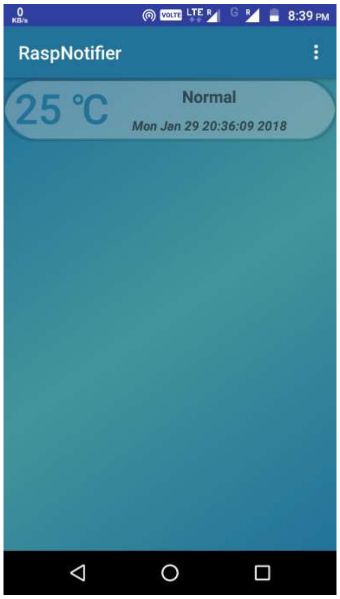
```
No.   Time       Source          Destination     Protocol Length Info
    1 0.000000   35.201.97.85    192.168.43.29   TLSv1.2   270 Application Data
    2 0.041412   192.168.43.29   35.201.97.85    TCP        54 2873 → 443 [ACK] Seq=1 Ack=217 Win=68 Len=0
  121 45.003274  192.168.43.29   35.201.97.85    TCP        55 [TCP Keep-Alive] 2873 → 443 [ACK] Seq=0 Ack=217 Win=68 Len=1
  122 45.003590  192.168.43.29   35.201.97.85    TLSv1.2    90 Application Data
  123 45.159034  35.201.97.85    192.168.43.29   TCP        66 [TCP Dup ACK 1#1] 443 → 2873 [ACK] Seq=217 Ack=1 Win=181 Len=0 SLE=0 SRE=1
  124 45.159036  35.201.97.85    192.168.43.29   TCP        54 443 → 2873 [ACK] Seq=217 Ack=37 Win=181 Len=0
  157 90.006844  192.168.43.29   35.201.97.85    TLSv1.2    90 Application Data
  158 90.113279  35.201.97.85    192.168.43.29   TCP        54 443 → 2873 [ACK] Seq=217 Ack=73 Win=181 Len=0
```

```
> Frame 1: 270 bytes on wire (2160 bits), 270 bytes captured (2160 bits) on interface 0
> Ethernet II, Src: 66:db:43:a7:3c:f9 (66:db:43:a7:3c:f9), Dst: HonHaiPr_bf:92:67 (70:77:81:bf:92:67)
> Internet Protocol Version 4, Src: 35.201.97.85, Dst: 192.168.43.29
> Transmission Control Protocol, Src Port: 443, Dst Port: 2873, Seq: 1, Ack: 1, Len: 216
v Secure Sockets Layer
  v TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
      Content Type: Application Data (23)
      Version: TLS 1.2 (0x0303)
      Length: 211
      Encrypted Application Data: d6cccb338ead44c1c6f54377b8edc93fcd2f0c18c2728895...
```

```
0030  00 b5 24 d6 00 00 17 03  03 00 d3 d6 cc cb 33 8e   ..$..... ......3.
0040  ad 44 c1 c6 f5 43 77 b8  ed c9 3f cd 2f 0c 18 c2   .D...Cw. ..?./...
0050  72 88 95 10 7f eb 91 31  17 b9 ae 2b ee 4b 18 91   r......1 ...+.K..
0060  4e 95 28 6b e8 34 1a 8a  d7 4f 32 76 55 50 7a 6e   N.(k.4.. .O2vUPzn
0070  cf 4f 67 21 dc 0a 9e 9d  9b af 4f 8c 28 d1 a9 15   .Og!.... ..O.(...
0080  92 75 96 a1 cb 1a 07 32  b9 a6 7a cc d1 4f f5 b4   .u.....2 ..z..O..
0090  4d 65 cc 55 b9 36 52 72  59 98 60 05 f1 0d c9 4a   Me.U.6Rr Y.`....J
00a0  be 50 4c b1 b7 c1 25 c0  10 f6 f2 3f 2c 7d dd d0   .PL...%. ...?,}..
00b0  bf f9 18 f4 33 3e cc 0d  9f 3f fc fa b6 ad 7d 6e   ....3>.. .?....}n
00c0  5f 6a 4e 8d 36 02 48 4f  46 af 63 47 d8 e8 da 49   _jN.6.HO F.cG...I
00d0  01 b3 fb 43 f1 1f cb 4e  cc dc 85 3d 02 8d 5e c3   ...C...N ...=..^.
00e0  d2 ad f5 34 99 cb 9c 82  d4 bd 22 29 d6 58 a6 02   ...4.... ..").X..
00f0  72 e3 95 e2 18 0f 34 8a  1e 05 2c 27 ce 4c e0 85   r.....4. ..,'.L..
0100  62 75 66 af 27 33 ff eb  0e 0b 7b 08 04 10         buf.'3.. ..{...
```

Payload is encrypted application data (ssl.app_data), 211 bytes | Packets: 195 · Displayed: 14 (7.2%) | Profile: Default

Figure 3: Packet that is intercepted has encrypted data.



RaspNotifier

25 °C Normal

Mon Jan 29 20:36:09 2018

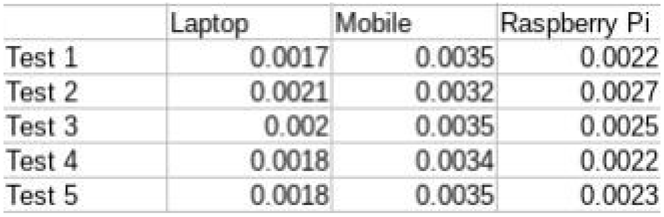
Figure 4: Data received successfully on Device B.

The recent DDoS attacks have shown that if the IOT devices do not required security then they can be very much vulnerable to future attacks too. Therefore, we took an example here and did the AES implementation for a temperature sensor. if this kind of device is used in a Server Room and if someone hacks into it and displays wrong information to the user then it can lead to very bad consequences. We can say this thing for other IOT devices too. In addition, there can be DDoS attacks too in which the computational power of these can be used to do hacking which can lead to very evident problems. In addition, if the IOT device is sending some confidential information then the hackers can read or even temper with this information and thus it can lead to disasters.

The main advantages of AES are:

- AES is more secure (it is less susceptible to cryptanalysis than TRIPLE-DES).
- AES supports larger key sizes than 3DES’s 112 or 168 bits.
- AES is faster in both hardware and software.
- AES’s 128-bit block size makes it less open to attacks.

Thus, we used AES since because of the fastness and security of the algorithm. We can encrypt the data in very less time and it does not require very less resources. Due to the way it has been designed, it is less open to the attacks. In addition, we can easily implement this algorithm in any of the major programming, which also makes it platform independent. It is currently widely used in various applications due to all of its features.



	Laptop	Mobile	Raspberry Pi
Test 1	0.0017	0.0035	0.0022
Test 2	0.0021	0.0032	0.0027
Test 3	0.002	0.0035	0.0025
Test 4	0.0018	0.0034	0.0022
Test 5	0.0018	0.0035	0.0023

Figure 5.1: AES Algorithm execution time in different devices

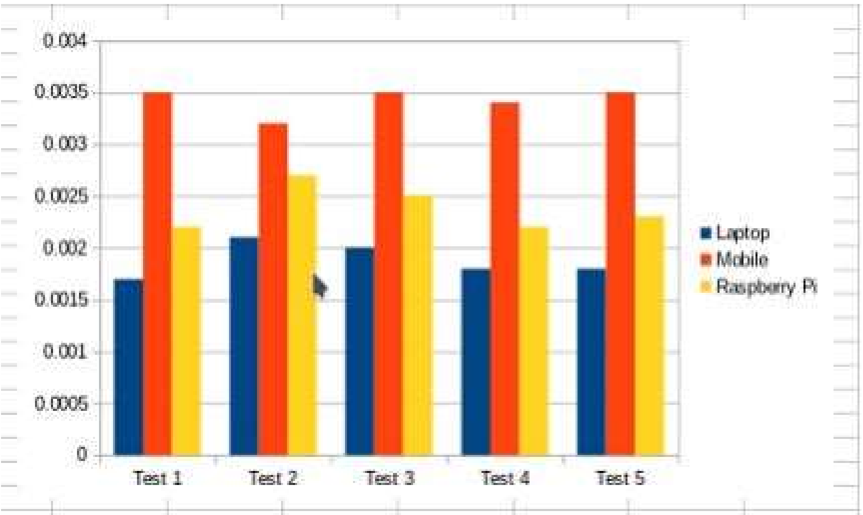


Figure 5.2: The graph depicting the table

CONCLUSIONS AND FUTURE WORK

Therefore, we can say that by using AES we can make the use IOT devices safe. No hacker can decrypt AES encrypted text at the current moment and thus it is secure to transmit data on public networks too.

Future Work

We would like to:

- Incorporation of RSA (Rivest Shamir Adleman) Cryptosystem for secure transmission of Keys.
- Having Dynamic Device IDs rather than Static Device IDs in our protocol.
- Making the communication between devices two way.

ACKNOWLEDGEMENT

We are highly indebted to Indian Institute of Information Technology, Vadodara for their guidance and constant supervision as well as for providing necessary information regarding the project also for their support in completing the project. Special thanks to Dr. Keyur Parmar for guiding and mentoring us during our project. Without his help completion of this project would not have been possible.