Journal of Emerging Computer Technologies
Research Article
Received: *2020-11-25* | ***Reviewing:*** *2020-11-26* & *2021-02-14* | ***Accepted:*** *2021-02-15* | ***Online:*** *2021-02-19* | ***Issue Date:*** *2021-06-30*

Edge Computing Security with an IoT device

Beyda Nur Kars
Dokuz Eylul University
Department of Computer Engineering
İzmir, TURKEY
beyda.kars@ceng.deu.edu.tr

***Abstract*— Information technologies are changing every aspect of human life day by day. In this context, Edge Computing, Internet of Things, Machine Learning and Big Data Analytics technologies are thought to be a part of this change. Edge computing aims to bring the computing power from the remote cloud environments to the endpoints/edges of networks. Thus, smart applications do not have to send all their data to the cloud and wait for the answers to come back over the same long route. Despite this advantage, there are security risks in the edge computing process. Encryption of information is of great importance especially for IoT devices to perform transactions safely. With a system established in this study, encrypted communication has been tried to be provided on IoT devices performing edge computing. In this way, it is aimed to make the communication secure. Arduino is used as an IoT device. In the encryption process, AES encryption is used with 128-bit and 256-bit key length.**

Keywords—Edge computing, security, AES, encryption, Arduino

I. INTRODUCTION

Edge Computing is an IT system designed to bring applications and computing capability as close as possible to the users or "objects" who need them. Edge Computing; mobile computing is driven by the reduced cost of computer components and the number of networked devices on the Internet of Things (IoT). Depending on the application, time-sensitive data in an Edge Computing can be processed by a smart device or sent to a medium server in a location close to the receiver. Edge computing is the optimization of the application's data with end-to-end encryption. That is, cloud computing turns it into a more sophisticated computing cloud architecture. Edge Computing is expected to play a serious role in the near future, especially in terms of information technologies. Edge Computing is referred to as end calculation in English. As it is known, the applications we use on smartphones have to deliver their existing data to the cloud, which is quite far away. Likewise, data is expected to return from this long journey. Edge Computing prevents this waste of time. Edge Computing, which can carry its computing power even to the extreme parts of the networks, brings serious relief in this sense. it cannot replace cloud computing or directly replace it. Edge Computing aims to process data from the ends to the cloud.

Despite all these positive aspects, Edge Computing can pose certain risks in terms of security. The fact that the architecture is distributed brings an increase in the number of vectors. Some vulnerabilities may also arise in the system. When more and more intelligence that an end customer has become more vulnerable, malware becomes infiltrations and vulnerabilities. On the other hand, licensing can force users to some extent, especially in terms of cost. Because each additional function is licensed separately, which naturally leads to higher costs.

Edge Computing brings advantages beyond classical architecture, as it optimizes resource usage significantly. First of all, the processors used in the related devices provide a relatively low power requirement and provide more efficient hardware security.

Mahadev Satyanarayanan, who is described as the father of Edge computing, explained how edge computing was born in his article [1]. According to Satyanarayanan, the limited resources of mobile devices have created a need for a platform that will perform the calculation process instead of these devices and have a more powerful resource. Cloud Computing, which promises much stronger resources (computing, memory and storage resources) compared to mobile devices, emerged to solve this problem. However, as latency-sensitive and bandwidth-hungry applications developed, gathering all computing power in the remote cloud environment started to cause problems. The main reason behind this problem is that the application packages have to pass through many routers controlled by many Internet Service Providers (ISPs) at different layers until they reach the cloud environment. Each router through which packets pass manifests itself as an increase in the Round Trip Time (RTT) of the packets of delay sensitive applications. In addition, the end-to-end delay values of the routes provided by routing protocols for packets can change dynamically due to ISPs and network problems and situations. In addition to all these, it is predicted that billions of IoT devices will connect to the internet and transfer data in the future. The fact that billions of devices send data to the cloud environment located at a single point in a remote location will inevitably lead to bandwidth bottlenecks. For all these reasons, it is clear that remote cloud environments cannot be a solution for latency sensitive and bandwidth hungry applications. Edge computing was born to solve this problem. Edge computing aims to prevent applications from being affected by long RTT times and bandwidth bottlenecks by moving cloud resources to the endpoints of networks.

II. RELATED WORKS

Confidentiality should be able to ensure that the data is only available to authorized users during the process and that it is not interfered by unauthorized people. Privacy is the most important security item in IoT because many devices can be integrated into IoT. It should be ensured that data received with a measuring device does not provide secure information to neighbouring devices. To ensure this privacy, advanced techniques and others, including key management mechanisms, should be developed and used [2].

It is essential to use wireless data transmission and to encrypt the information transferred between the nodes in order to keep it confidential. The most appropriate encryption algorithms and adequate key management systems are required to secure this data [3].

Wireless sensor networks have a large number of trust-based intrusion detection systems (IDS) that are used to defend against attacks. However, the effectiveness of IDS decreases in IoT due to the large amount of data produced in a short time. Meng et al. [4] proposed a Bayesian-based trust management method that incorporates traffic sampling into IDS under a hierarchical structure [5].

James King of Lulea University created an IoT network in his work in 2015 within the local network. Arduino used it as a gateway and collected and encrypted data from different devices and sensors. The encryption algorithm used is the Advanced Encryption Standard (AES) with both 128-bit and 256-bit key length [6]. Mahmudur Rahman, Bogdan Carbunar and Umut Topkara from Florida International University collects instant values from devices such as stopwatch, heart rhythm tracker and moisture meter used in fitness studies and transmits the IoT device to a web server remotely. In this study, FirstBeat is used as an encryption algorithm and Arduino Uno device is used as an IoT device. It also belongs to the IoT device from the webserver by taking the coordinate data, we can track the point we are at via GPS [6].

There is a need for special security mechanisms produced for low-resource devices such as IoT and wireless sensor network devices. It is necessary to use random number generators [7-11], authentication protocols [12-14] and some encryption algorithms that will be built-in in some of these devices.

III. PROPOSED WORK

In this section, we will briefly mention the target in the study. We will talk about the hardware materials that will be uses later and the flow of my work.

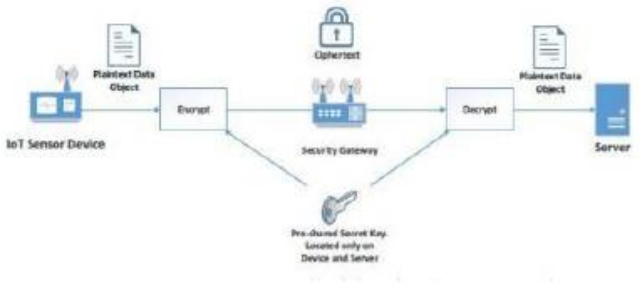

Figure 1. Encrypted data transfer

Our goal in this study is to encrypt data sent from IoT devices. This operation is shown in Figure 1. We used two Arduino devices in this study. The first one was used as a gateway device, the second one was used to get sensor data. DHT11 is used as the sensor to get data from the physical environment.

The data received from the sensor will be encrypted with AES 128 and sent to the network gateway. Thus, the data sent by IoT devices will be secured. Arduinos are also communicated among themselves. A wired connection has been made.

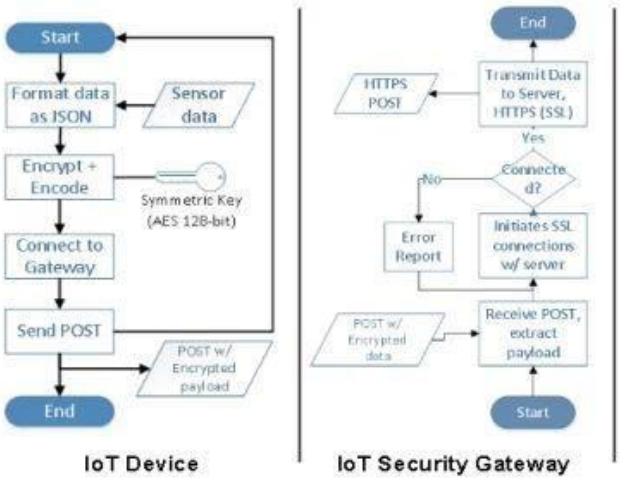

Figure 2. Workflow

In Figure 2, the workflow of operations of the two Arduino IoT device can be seen.

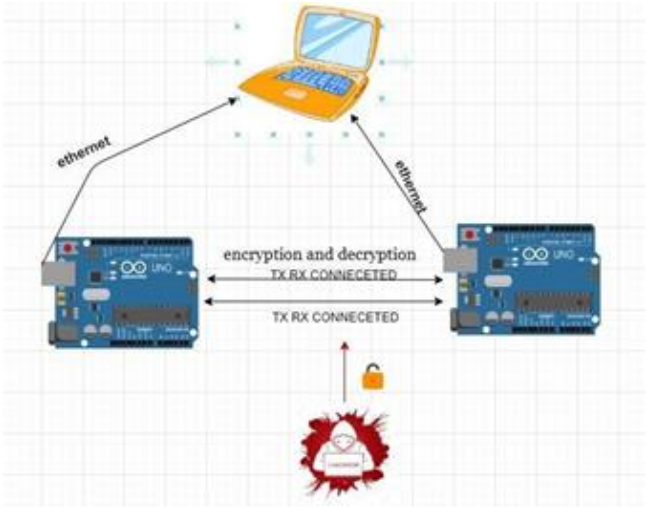

Figure 3. IOT Security Diagram

Figure 3 shows the relationship between all the necessary material in the workflow. Sensor data from Arduino 1 is sent to the laptop and encrypted there. And again from Arduino 2 to Arduino 1 (i.e. gateway is sent). Then Arduino 1 is decrypted.

The purpose of encrypting the data is to protect the data from malicious software while it is transmitted from one device to another.

IV. MATERIALS

As seen in the figures, we have two modules. The first one is the module with Arduino and sensor connection. The second one is Arduino, which will be used as gateway.

Figure 4 also has an Arduino module that allows us to receive data from the sensor.

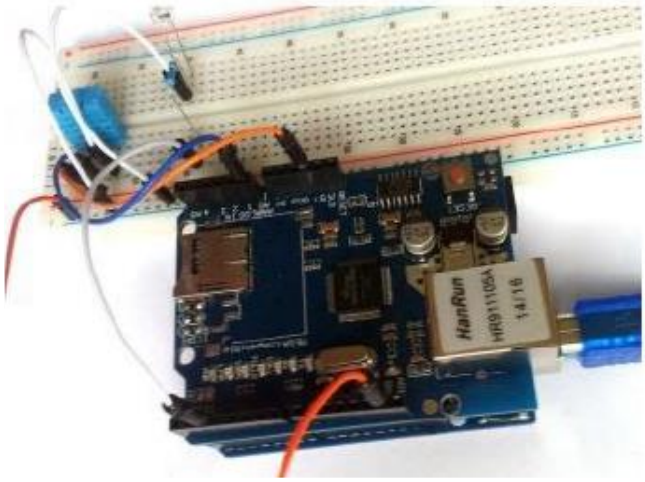

Figure 4. Arduino and Dht11 Connection

V. TEST RESULTS

Before talking about the test results in this part of the project, we will talk about the AES algorithm. We will give some information about the use of the AES algorithm in the working mechanism of this study.

The AES algorithm is a block cypher algorithm that encrypts 128-bit data blocks with 3 different key options, 128, 192 or 256-bit keys. The difference in key lengths makes the number of AES tour cycles differ. Since it is 128 bits, 10 rounds of processing are required. Since its length is 192 bits, 12 tours of processing is required. Because it is 256 bits, 14 rounds of operation are required [15].

Each round contains 4 different steps. These are as follows: byte replacement, row shift, column shuffling, and addition with a rotary switch. The data entered after 10 rounds is obtained as encrypted. The AES algorithm takes the key and passes it through certain processes, creating as many keys as the number of transactions. This number is 10 for 128-bit length. 10 different keys are created and the last key formed becomes the first key used to decrypt [16].

AES is a symmetric encryption algorithm, so the same key is used for both encryption and decryption. While deciphering the encrypted text, the operations performed while creating the encrypted text are applied in reverse.

The same key was used in encryption and decryption in the project as in Figure 5 and Figure 6.

```
uint8_t key[] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
aes128_enc_single(key, data);
```

Figure 5. Code for the usage of the encryption key

```
uint8_t key[] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
Serial.print("decrypted:");
aes128_dec_single(key, data);
```

Figure 6. Code for the usage of the decryption key

The Arduino's own library was used to encrypt the sensor data. The received data turns into encrypted text with the key we set. But since the sensor data that is received is a numerical value, the type conversion was made.

In the test part, we first load our code into our sensor module and continue to supply energy elsewhere by separating the module from the computer. Then we load the code into the module we will use as a gateway and provide the connection between the two Arduino devices. We ensure that encrypted data is sent to the gateway.

In Figure 7, circuits have all connections were shown.

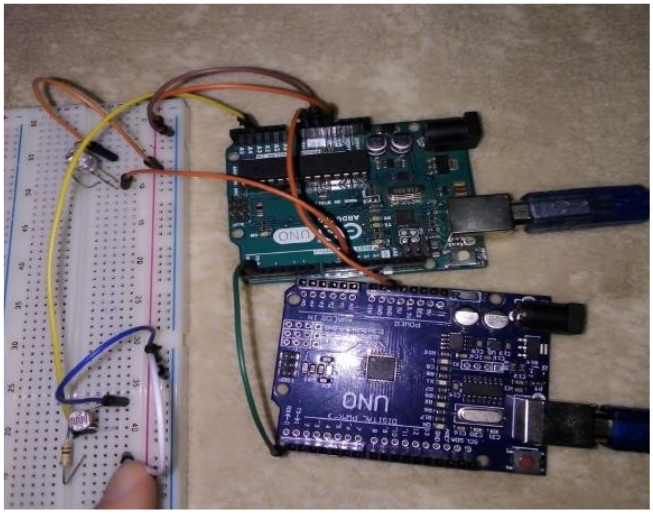

Figure 7. Hardware

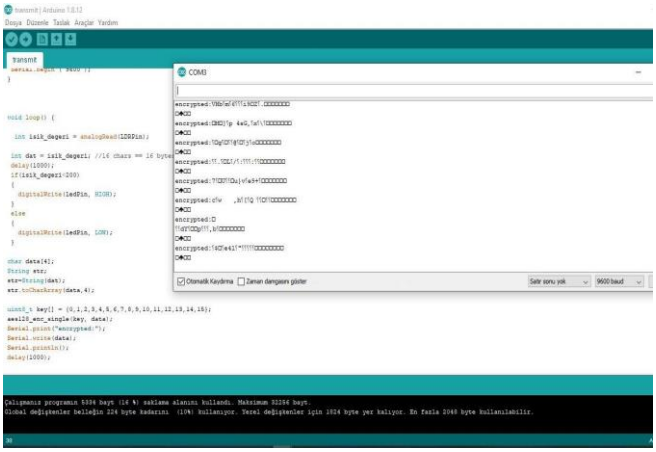

Figure 8. Transmit and Encrypted

Since there are two Arduino devices working as a receiver and a transmitter in the project, it was provided communication between the Arduino. This communication was made as a wired connection. Then the data received was encrypted and sent to the other party. The code part and serial port screen that is used for the receiver and the encryption process shown in Figure 8.

Figure 9 shows the encrypted version of the data received from the sensor. The code fragment in Figure 8 can be seen more clearly in Figure 10.

In Figure 11, there are pieces of code for both transmit and receive, that is decryption. Using these codes and operations, data communication is carried out by encrypting with AES. In this way, communication security will be provided for the edge computing process.

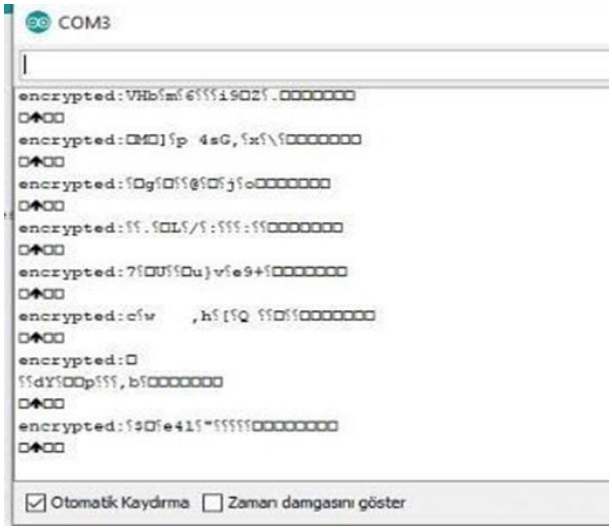

Figure 9. Receive and Decrypted

```
#include <AESLib.h>
int ledPin = 10;
int LDRPin = A3;
void setup() {
 pinMode(ledPin, OUTPUT);
 Serial.begin ( 9600 );
}
void loop() {
 int isik_degeri = analogRead(LDRPin);
 int dat = isik_degeri; //16 chars == 16 bytes
 delay(1000);
char data[4];
String str;
str=String(dat);
str.toCharArray(data,4);
uint8_t key[] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
aes128_enc_single(key, data);
Serial.print("encrypted:");
Serial.print(data);
Serial.println();
delay(1000);
}
```

Figure 10. Arduino code for encryption

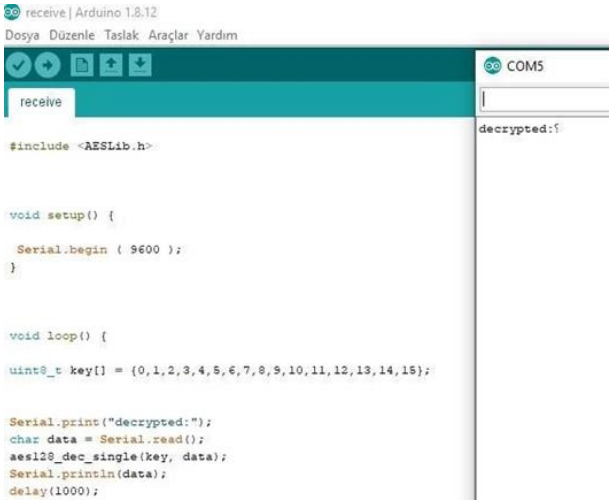

Figure 11. Receive and Decrypted

VI. Conclusion

In this study, an edge computing system was established with two Arduino devices. One of the devices was used as a gateway. The other is used to collect the data obtained from the sensor. The collected data is encrypted with AES encryption algorithm to ensure secure communication of data. Encryption methods with 128-bit and 256-bit key lengths have been tried for the encryption process. The AES library on Arduino was used for encryption. With this study, the data was sent encrypted and the data was reconstructed by decrypting the other side. Different encryption methods and different security mechanisms may be tried in the future.

References

[1] Satyanarayanan, M. (2017). Edge Computing. IEEE Computer, 50(10), 36-38.

[2] Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., & Zhao, W. (2017). A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. IEEE Internet of Things Journal, 4(5), 1125-1142.

[3] Bull, P., Austin, R., Popov, E., Sharma, M., & Watson, R. (2016, August). Flow based security for IoT devices using an SDN gateway. In 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud) (pp. 157-163). IEEE.

[4] Meng, W., Li, W., Su, C., Zhou, J., & Lu, R. (2017). Enhancing trust management for wireless intrusion detection via traffic sampling in the era of big data. Ieee Access, 6, 7234-7243.

[5] Wang, T., Zhang, G., Liu, A., Bhuiyan, M. Z. A., & Jin, Q. (2018). A secure IoT service architecture with an efficient balance dynamics based on cloud and edge computing. IEEE Internet of Things Journal, 6(3), 4831-4843.

[6] Akkuş, S. (2016). Gömülü sistem tabanlı, kriptolu TCP/IP veri haberleşmesi uygulaması.

[7] Cabuk, U. C., Aydın, Ö., & Dalkılıç, G. (2017). A random number generator for lightweight authentication protocols: xorshiftR+. Turkish Journal of Electrical Engineering & Computer Sciences, 25(6), 4818-4828.

[8] Kösemen, C., Dalkılıç, G., & Aydın, Ö. (2018). Genetic programming-based pseudorandom number generator for wireless identification and sensing platform. Turkish Journal of Electrical Engineering & Computer Sciences, 26(5), 2500-2511.

[9] O'Neill ME. PCG: A family of simple fast space-efficient statistically good algorithms for random number generation.Available online at http://www.pcg-random.org/pdf/toms-oneill-pcg-family-v1.02.pdf

[10] Aydın, Ö., & Dalkılıç, G. (2018, July). A hybrid random number generator for lightweight cryptosystems: xorshiftLplus. The 3rd International Conference on Engineering Technology and Applied Sciences (ICETAS).

[11] Aydın, Ö., & Kösemen, C. (2020). XorshiftUL+: A novel hybrid random number generator for internet of things and wireless sensor network applications. Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi, 26(5), 953-958.

[12] Armknecht F, Hamann M, Mikhalev V. Lightweight authentication protocols on ultra-constrained RFIDs-mythsand facts. In: International Workshop on Radio Frequency Identification: Security and Privacy Issues; 2015. pp.1-18. Springer, Cham.

[13] Lee JY, Lin WC, Huang YH. A lightweight authentication protocol for internet of things. In: 2014 InternationalSymposium on Next-Generation Electronics (ISNE); New York, USA; 2014. pp. 1-2.

[14] Aydın, Ö., Dalkılıç, G., & Kösemen, C. (2020). A novel grouping proof authentication protocol for lightweight devices: GPAPXR+. Turkish Journal of Electrical Engineering & Computer Sciences, 28(5), 3036-3051.

[15] Şeker, Ş.E. (2009). AES ve Rijndeal Şifreleme. *Bilgisayar Kavramları*. Retrieved from http://bilgisayarkavramlari.com/2009/06/03/aes-ve-rijndeal-sifreleme/ , Accessed date : 25.11.2020

[16] Surian, D. (2006). Algoritma Kriptografi AES Rijndael. TESLA Jurnal Teknik Elektro UNTAR, 8(2), 97-101..