Journal of Positive School Psychology
2022, Vol. 6, No. 3, 8508–8516

IOT Security: Data Encryption for Arduino-based IOT Devices

Shapina Abdullah[1]; Noorhayati Mohamed Noor 2; Nor Azimah Khalid [3]; Zolidah Kasiran[4]; Affiq Juzaily Khairol Hisham[5]

[1.2.3.4.5] *Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia*

Email: [1] piena@fskm.uitm.edu.my, [2] noorhayati@fskm.uitm.edu.my, [3]azimah@fskm.uitm.edu.my, [4]zolidah@fskm.uitm.edu.my, [5]afiqjuzaily@gmail.com

Abstract

The Arduino Data Encryption Project is motivated by the lack of focus on data security in Internet of Things (IoT) powered projects. The focus of many IoT projects were on the functionality and the efficiency of the device. Security is just an afterthought after an IoT project is developed and often overlooked. Many of the IoT devices are vulnerable to data stealing by hackers that sniff an entire network targeting vulnerable IoT devices. IoT devices that are not secure communicate with the data in the form of plain text or string which makes the Arduino easily identified by the attackers. The valuable data being communicated in a network can also be stolen by sniffers that has breach the private network of the IoT device. The main aim of the project is to deploy cryptographic functions to encrypt the data being sent by the IoT system over a Wi-Fi network. There are three cryptographic algorithm s developed that have various forms of complexity and processing speed. The three cryptographic algorithms are Caesar Cipher, SHA 256 and AES 128 Bit. With the cryptographic functions coded into the Arduino Uno, the data of the Arduino can be encrypted and has been successfully proven by the Arduino IDE serial monitor. It has added an extra layer of security as the data no longer be understandable by the data thieves that sniff the network. The addition of cryptographic functions of the Arduino Uno also align with the objective of the project that is to encrypt the data being transferred by the Arduino Uno.

Key-words: Internet of Things (IoT), Data Encryption, Security.

I. INTRODUCTION

Recent advancement of wireless technology and Internet of Things (IoT) have brought a significant development such as Vehicular Ad hoc Networks (VANETs) (Hatim, Elias, Awang, & Darus, 2018). The Li-Fi technology is anticipated to provide a cheap and reliable for the IoT network to connect to the internet (February, Ahmad, Hwaitat, & Science, 2020). The extensive implementation of IoT is causing a paradigm shift in infrastructure, consumer industries and services to be fast and efficient in a smarter way (Mittal, 2019). Unfortunately, these IoT devices are exposed to imminent danger from hackers seeking precious data such as through collusion attacks, impersonation and eavesdropping (Tao et al., 2018). Therefore, the focus is now shifting more towards trust and security of these IoT devices. The resource constrained IoT based applications demand for lightweight, flexible and low cost cryptography (Sehrawat & Journal, 2019).

The threat for data leakage must be addressed to secure these devices (Azizi et al., 2019) Researchers in (Sui, 2018) and (Choi, 2018) focuses on the user authentication side of IoT with no regards to securing the transmitted data. The data mentioned here is the string data value inputs that is collected by the sensors given to the Arduino. The data is then transmitted over the internet to the user to be viewed. This poses a security risk as the data is transmitted in the

© 2022 JPPW. All rights reserved

form of string variable and can be easily read. The method in (Muhammad N Aman, Kee Chaing Chua, 2017) focuses on data provenance which is the trust on validity of data by encrypting data packets with his own formulated encryption algorithm, however it doesn't have a local decryption key stored on the IoT device, whereby the processing in taken online and not locally on the IoT device.

The purpose of this project is to develop a mechanism to encrypt the data stored in the Arduino while it is being sent over the network by using Caesar's Cipher, SHA256 and AES 128 Bit. It also comprises of a mechanism to encrypt the string data sent to the Arduino from a webpage in order to simulate real life uses of the Arduino whereby users developed a specific webpage to send the data to the Arduino micro-controller. However, to the best of our knowledge these methods of IoT security does not focus on securing the sensor data transmitted over a network. The purpose of this paper is to develop a mechanism that could make the data unreadable by unauthorized users using an encryption algorithm. It then would prevent precious data from being stolen by the attackers through network sniffing.

II. RELATED WORK

The emerging of IoT technologies have opened many rooms for improvements in terms of security. Lately, IoT security has become the subject of scrutiny after a number of high-profile incidents where a common IoT device was used to infiltrate and attack the larger network. Implementing security measures is critical to ensuring the safety of networks with IoT devices connected to them.

The security aspect of IoT devices has often be an afterthought and not being viewed as the integral part of any IoT system that laid an IoT security taxonomy in (Sachin Babar, Parikshit Mahalle, Antonietta Stango, 2010). In addition, they proposed that security should now be the main focus in the development process of IoT devices so that it aligns to one of the fundamental IoT objectives that is security and privacy. The three security requirements would be to focus on user identification with the use of unique device identifiers in encryption algorithm, to secure data communication whereby in traffic sniffing will be prevented by use of data encryption, and lastly to ensure legitimate data is tamper resistant as a hacker cannot inject their own fake data into the communication as original data is encrypted.

The current relative work on the implementation of IoT security mainly focuses on authenticating the access of IoT devices. For example, a project (Muhammad N Aman, Kee Chaing Chua, 2017) focuses on the data provenance of IoT devices. Data provenance being the assurance of data created by the intended party at a specific location and time. The paper focuses more on user authentication and fingerprinting of IoT devices. Another work in (Venkatesan, 2018) proposed user access authentication by way of secure vaults. The secure vaults authenticate users by a session key generated by XOR operation. A lightweight cryptography implementation combines the software implementation and hardware implementations. In (Tao et al., 2018), KATAN algorithm was modified and optimized on the FPGA hardware platform is an example of such lightweight cipher designs. The implementation is meant for protecting patients' data privacy. Design metrics such as block sizes, the number of rounds and the key scheduling are considered in the design(Tao et al., 2018).

The use of encryption techniques in limited computing capabilities and storage space devices such as Arduino Uno can be seen in several papers. In (Julham, 2017), the author shows that encryption technique works well with the use of a substitution encryption technique for data communication security between Arduino Uno. In his work, the Caesar chipper method was used to exchange or replace each letter of the plain text with another letter with 3-letter interval of the plain text letter. The encryption keys generated were used in the encryption and decryption process. The results indicate that low memory storage involves for such execution.

© 2022 JPPW. All rights reserved

In (Roshidi Din , Rosmadi Bakar , Raihan Sabirah Sabri , Mohamad Yusof Darus, 2019), an analysis was conducted to review on the study of steganography in natural language based on 11 performance metric. Out of the 11 performance metrics, the study observed that the highest performance metrics used in previous studies are capacity, security and robust. The data demonstrates that the encryption is the best way to provide a secure communication by hiding information process safety and embedding the secret message. A third-party software is embedded as agents to ease the encryption process in (Norkhushaini Awang, Nurul Hidayah Ahmad Zukri, Nor Aimuni Md Rashid, 2017). Two agents were directly involved in the security of user data which are application agent and crypto agent increases the security of storing users' information by using a strong encryption algorithm. The paper however aims for password management application.

III. SYSTEM ARCHITECTURE

The network attack tools must first be determined to be used for the sniffing attack over Wi-Fi simulation. It must be able to identify the Arduino machine in the network by IP address and MAC Address and also intercept the unencrypted string data being send by users connected to the Arduino or from the Arduino to the user. Ettercap is chosen mainly because the application is easier to use and it meets the required aspects for the data sniffing to work. Ettercap can easily show the number of host connected to the Wi-Fi network provided that the attacker's computer is already in the vulnerable Wi-Fi network. During the choosing process of the network attack tool, Ettercap proves to be more reliable and can easily find the Arduino in the network in all the network attack test.

To enable the Arduino Uno to be connected to the Wi-Fi network, and ESP8266 Wi-Fi module is used. The ESP8266 Wi-Fi module has the capability to connect to the standard 2.4 GHz Wi-Fi frequency that is used a lot in organizations and workplaces. Since the Wi-Fi connectivity solely rely on the ESP8266 module, the Arduino.ino sketch file has to be uploaded on the Wi-Fi module itself and not the Arduino. ESP8266 has the same size of memory onboard with 32 KiB of memory, the same amount found on the Arduino Uno. The Arduino Uno mainly acts as the controller of the ESP8266 Wi-Fi module and provides power to it by connecting it to the pins of the host Arduino Uno.

The HTML Web Server will be hosted by the ESP8266 connected to the Arduino Uno. It will be accessible to any devices whether it is a PC or a mobile device that is connected to the same Wi-Fi network as the Arduino Uno. This is implemented as users of the Arduino Uno usually has access to their Arduino remotely by implementing a web server on it. The HTML web server allows the users to input the string that they want to be encrypted and choose from 3 different available encryption with different levels of complexity. The encryption algorithms available are Caesar's Cipher, SHA256 and AES 128 Bit. After both of the text fields are inputted, the Arduino will process the chosen encryption and print it on the serial monitor.

3.1. Encryption Algorithm

The emerging In the .ino sketch of the ESP8266 is implemented with three commonly used encryption algorithms with varying degrees of complexity from the simple Caesar's Cipher , the more intermediate SHA256 and the industry standard AES 128 Bit encryption.

(a) Caesar's Chipper

The Caesar's Cipher implemented in the Arduino sketch is the commonly used 13 shifts encryption or commonly known as ROT 13 as shown in Figure 1. The "+ 13 "code is essentially telling the program to shift the *inputString* variable that is inputted by the user to shift it 13 places. This will result in all the characters of the original input string to be obscured by 13 places. Example, the letter "A" will be shifted as the letter "N". It works with alphanumerical strings as well.

Figure 1 – Caesar's Chipper Algorithm

© 2022 JPPW. All rights reserved

```
if (encryption == "CaesarCipher")
{

ms = micros ();
for(i = 0; (i < 100 && inputString[i] != '\0'); i++)
inputString[i] = inputString[i] + 13; //the key for encryption is 3 that is added to ASCII value

int n = inputString.length();
char char_array[n + 1];
strcpy(char_array, inputString.c_str());

Serial.print ("Encrypted text :");
Serial.println(char_array);

Serial.print (" Time taken : ");
Serial.println ((micros() - ms));
}
```

(b) SHA 256

SHA-256 or "secure hash algorithm" is a cryptographic hash function with a digest value of 256 bits. It is also a keyless cryptographic function that utilizes Boolean operations such as AND, XOR and OR (denoted by ∧, ⊕ and ∨) combined with an integer addition module of base 2. SHA 256 also has padding implemented complete its encryption.

Figure 2 – SHA 256 Algorithm

```
void sha256_init(SHA256_CTX *ctx);
void sha256_update(SHA256_CTX *ctx, const uint8_t data[], size_t len);
void sha256_final(SHA256_CTX *ctx, uint8_t hash[]);

#define ROTLEFT(a,b) (((a) << (b)) | ((a) >> (32-(b))))
#define ROTRIGHT(a,b) (((a) >> (b)) | ((a) << (32-(b))))

#define CH(x,y,z) (((x) & (y)) ^ (~(x) & (z)))
#define MAJ(x,y,z) (((x) & (y)) ^ ((x) & (z)) ^ ((y) & (z)))
#define EP0(x) (ROTRIGHT(x,2) ^ ROTRIGHT(x,13) ^ ROTRIGHT(x,22))
#define EP1(x) (ROTRIGHT(x,6) ^ ROTRIGHT(x,11) ^ ROTRIGHT(x,25))
#define SIG0(x) (ROTRIGHT(x,7) ^ ROTRIGHT(x,18) ^ ((x) >> 3))
#define SIG1(x) (ROTRIGHT(x,17) ^ ROTRIGHT(x,19) ^ ((x) >> 10))

static const uint32_t k[64] = {
  0x428a2f98,0x71374491,0xb5c0fbcf,0xe9b5dba5,0x3956c25b,0x59f111f1,0x923f82a4,0xab1c5ed5,
  0xd807aa98,0x12835b01,0x243185be,0x550c7dc3,0x72be5d74,0x80deb1fe,0x9bdc06a7,0xc19bf174,
  0xe49b69c1,0xefbe4786,0x0fc19dc6,0x240ca1cc,0x2de92c6f,0x4a7484aa,0x5cb0a9dc,0x76f988da,
  0x983e5152,0xa831c66d,0xb00327c8,0xbf597fc7,0xc6e00bf3,0xd5a79147,0x06ca6351,0x14292967,
  0x27b70a85,0x2e1b2138,0x4d2c6dfc,0x53380d13,0x650a7354,0x766a0abb,0x81c2c92e,0x92722c85,
  0xa2bfe8a1,0xa81a664b,0xc24b8b70,0xc76c51a3,0xd192e819,0xd6990624,0xf40e3585,0x106aa070,
  0x19a4c116,0x1e376c08,0x2748774c,0x34b0bcb5,0x391c0cb3,0x4ed8aa4a,0x5b9cca4f,0x682e6ff3,
  0x748f82ee,0x78a5636f,0x84c87814,0x8cc70208,0x90befffa,0xa4506ceb,0xbef9a3f7,0xc67178f2
};
```

The above code in Figure 2 shows the standard SHA 256 formula in use and that the message is processed by blocks of 512 (16 × 32 bits) whereby each block requiring 64 rounds.

(c) AES 128 Bits

AES 128 Bit is an encryption algorithm that has a block size of 128-bits. It separates the data into a 4 x 4 column amounting to 16 bytes. Then the block is derived with Rjindael's key schedule with a predetermined key as stated in hexadecimal as illustrated in Figure 3.

Figure 3 – AES 128 Bits Algorithm

3.2. Design Phase

In this section, the design phase of the development process dictates the overall presentation and flow of the project.

(a) Designing an Arduino to ,PC Connection over Wi-Fi

The circuit shown in Figure 4 is designed in a way that is a common Arduino circuit design that utilizes the ESP8266 Wi-Fi module for users to send data over Wi-Fi to the user's PC whereby any data that the Arduino is authorized to send will be sent to the user in the form of string by default. The simulation is set up in a way to focus on the security flaw of the string data in transmission over a Wi-Fi network. Theoretically, any sensors can be used to get data such as temperature readings, gas readings and many others that is send to the user's PC for processing.

The simulation should be complied with the following characteristics:

a) Arduino should have Wi-Fi communication with the ESP8266 Wi-Fi module.

b) Send string data from the Arduino to the host PC.

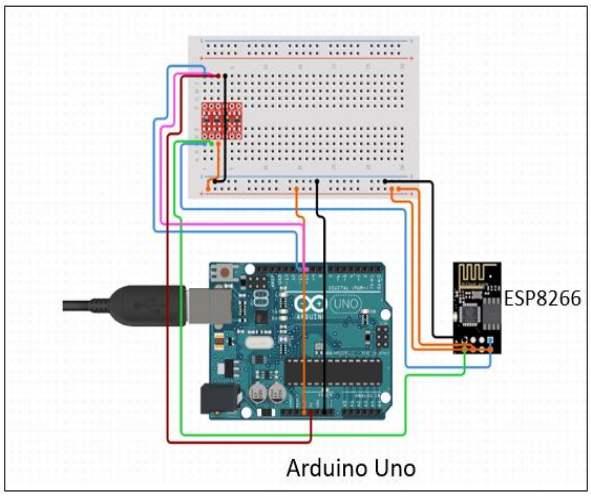
c) The host PC should be able to display the data transmitted by the Arduino.

d) The attacker's computer should be able to identify the Arduino in the network.

e) The attacker's computer should be able to sniff the string data in transmission and view the transmitted data.

Figure 4 – Physical Design of the Arduino Circuit

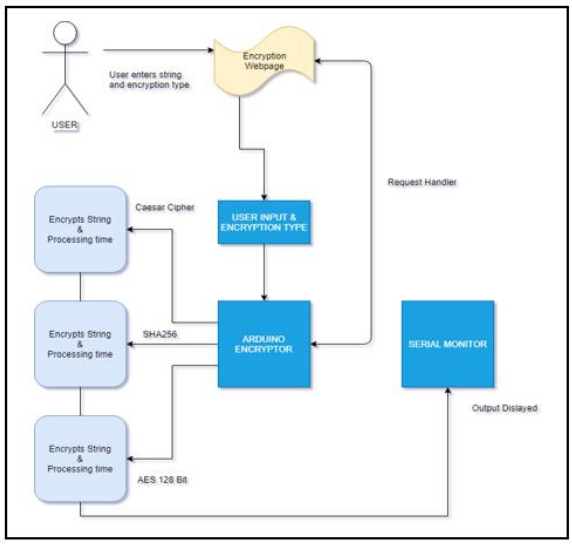
© 2022 JPPW. All rights reserved



(b) Data Flow Diagram

The data diagram as illustrate in Figure 5 elaborates the inputs of the user to the Arduino micro-controller and the flow of the results of the encryption process being shown at the serial monitor.

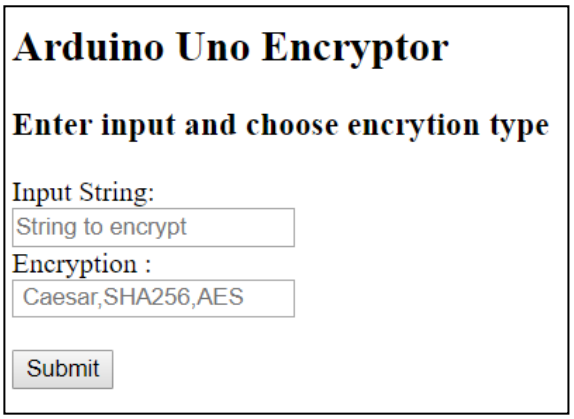
Figure 5 – Data diagram of the Arduino Data Encryption project



(c) Designing a HTML Web Server

The purpose of developing a HTML server for the Arduino is to clearly demonstrate the data in transmission for the user's to see and understand. The html webpage (see Figure 6) will be developed with simplicity in mind to outline the cryptographic functions implemented in the Arduino. The webpage will enable the user to enter the string to be encrypted as well as choose from the three available encryption that is Caesar's Cipher, SHA256 and AES 128 Bit. It is accessible only to users in the same network as the Arduino provided that the user enters the IP address of the Arduino in a search engine.

Figure 6 – The webpage developed called "Arduino Uno Encryptor"



3.3. Attack Simulation

The attack simulation will show how a hacker can take advantage of the vulnerability found in the Arduino setup. The vulnerability is that the data is not encrypted and is in the form of string being transmitted using HTTP, a famous but unsecured communication protocol. This is possible due to the Arduino lacking the support for HTTPS communication.

The attack begins with the installation of Wireshark in the hacker's machine. Next using Wireshark the hacker is able to see all the different types of communication in a network. Knowing that Arduino setups commonly use HTTP protocol, the hacker will apply a filter to view all the HTTP traffic. He later then sniffs the communication and captures all the data in the HTTP packet. If the data in not encrypted, the hacker is able to view the data and misuse it to their advantage.

3.4. Security Measurement Implemention

The main objective of the project is to show that the data sent by the Arduino can be encrypted and secured. The encryption algorithm chosen for deployment is Caesar's Cipher, SHA256 and AES 128 Bit. These encryption algorithm are of various encryption strengths and different processing time altogether so it would provide a realistic performance measure during the testing phase.

Firstly the encryption algorithms mentioned requires a custom library to be added for each algorithm. Note that many of the encryption-based custom library are developed for the PC

© 2022 JPPW. All rights reserved

and not for the Arduino. The ones that work on the Arduino require some editing in the library and some research to execute the code properly or the Arduino IDE would have an error compiling the .ino file for the Arduino.

To encrypt a string by using the Arduino the user needs to go to the html website served by the Arduino. (This approach is different compared to local-host html web serving as the website is entirely server by the Arduino) Then the user can encrypt any number of string sentences and choose between 3 encryption methods which are Caesar's Cipher, SHA256 and AES 128 Bit. The original data keyed in by the user is captured and sent to the serial monitor for viewing. After the encryption process have been completed, the encrypted message will be shown at the serial monitor together with the time taken (milliseconds) of the encryption.

While a hacker is trying to intercept the data sent by the Arduino to the user's PC, he or she will see that the data is no longer in the form of string as all the data sent is now encrypted using the mentioned algorithms. The 3 encryption algorithms also provide variety to the user as to choose which type of encryption to be used to accommodate more resource-heavy processes or highly secure data.

3.5. Testing and Verification

After the entire Arduino setup has been developed and all the programs has been loaded, the security measure will be tested and verified according to these factors;

a) Speed. Measured by the time taken to complete the data encryption process.

b) Reliability. Measured by the number of successful attempts at encrypting data.

c) Accuracy. Measured by the percentage of correctness of data when it is compared with the data in its original state.

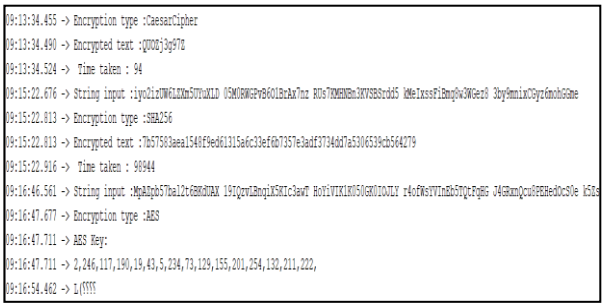
IV. . RESULT

The encryption algorithms are tested for four criteria that is processing reliability, processing speed, accuracy and strength.

(a) reliability

For reliability testing all of the encryption were tested with different phrases of varying lengths of characters and combinations. Starting with string length of 10, 100 to 1000 characters.

Figure 7 – All of three encryption (Caesar's Cipher, SHA256 and AES 128 Bits) with varying plain text length



The Caesar Cipher was able to encrypt up to more than 1000 characters whereas the SHA256 and AES 128 Bit capped of at 100 characters only. SHA 256 would display a processing time of 98944 for all characters above 100 whereas AES 128 Bit would freeze the Arduino. All encryption algorithms were determined to be usable with a 100 character limitation on SHA256 and AES 128 Bits.

(b) Speed

The processing speed of each encryption were tested with random strings generated from Random.org. The string length tested spans from 10 to 100 characters. The results are recorded below.

Figure 8 – Graph of Process Time Taken (ms) and Length of Plain Text for Caesar's Cipher Encryption.

Figure 9 – Graph of Process Time Taken (ms) and Length of Plan Text for SHA256 Encryption.

© 2022 JPPW. All rights reserved

Figure 10 – Graph of Process Time Taken (ms) and Length of Plan Text for AES 128 Bit Encryption.

Upon completing many rounds of testing it is concluded that AES 128 Bit is the fastest encryption algorithm followed by SHA256 and Caesar Cipher.

(c) Accuracy

The accuracy of the encryption algorithms were tested using tools developed online. These tools are proven to be accurate at deciphering and decrypt encrypted messages as it is widely used by users worldwide. The input for the test is "aaabbbcccdddeee" and was sent to the Arduino to be encrypted by Caesar's Cipher. SHA256 and AES 128 Bits.

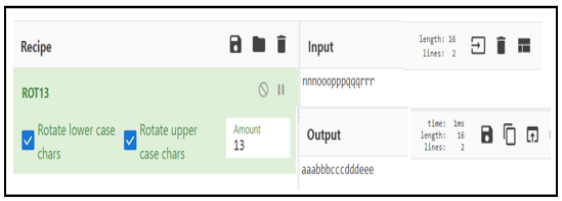


Figure 11 – Encoded Caesar Cipher text decoded using ROT 13.

The Caesar Cipher encryption was verified to be accurate using the ROT 13 decryption tool at https://gchq.github.io/CyberChef/ . The original string of "aaabbbcccdddeee" was able to be decoded.

Figure 12 – Encoded SHA256 decoded using SHA256() Encrypt & Decrypt.

The SHA256 encryption was verified to be accurate using the SHA256() Encrypt and Decrypt decryption tool at https://md5decrypt.net/en/Sha256/ . The original string of "aaabbbcccdddeee" was able to be decoded . AES 128 Bit has a randomly generated iv so it need to be bruteforced. In order to get the plaintext back below are some statistics shown to decrypt an AES encryption with possible values of keys.

V. CONCLUSION

The Arduino Data Encryption Project was a successful attempt at securing the data in transmission with cryptographic functions whereas before it was only sent in string. The 3 types of encryptions with various levels of strength, speed and resource usage should serve as proof that the small computing power of the Arduino micro-controller is in fact capable of securing the data just like full-fledged computers.

VI. ACKNOWLEDGEMENT

The authors like to extend the appreciation to Faculty of Computer and Mathematical Sciences of Universiti Teknologi MARA (UiTM) for the opportunity to engage in this research and publication.

BIBLIOGRAPHY

1. AZIZI, M., ARIFFIN, M., VOLUME, I. J., AZIZI, M., ARIFFIN, M., RAHMAN, K. A., … SHAH, M. (2019). Data Leakage Detection in Cloud Computing Platform, (1).International Journal of Advanced Trends in Computer Science and Engineering Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse7081.32019.pdf
2. CHOI, P. M.; D. B.; B. J. (2018). Lightweight gait based authentication technique for IoT using subconscious level activities. In 2018 IEEE 4th World Forum on Internet of Things (WF-IoT) (pp. 564–567). Retrieved from 10.1109/WF-IoT.2018.8355210
3. FEBRUARY, J., AHMAD, K., HWAITAT, A., & SCIENCE, C. (2020). A Survey on Li Fi Technology and Internet of Things (IOT). International Journal of Advanced Trends in Computer Science and Engineering, 9(1). Science Available Online at

© 2022 JPPW. All rights reserved

http://www.warse.org/IJATCSE/static/pdf/file/ijatcse34912020.pdf

4. HATIM, S. M., ELIAS, S. J., AWANG, N., & DARUS, M. Y. (2018). VANETs and Internet of Things (IoT): A discussion VANETs and Internet of Things (IoT): A Discussion. Indonesian Journal of Electrical Engineering and Computer Science, 12(October), 218–224. https://doi.org/10.11591/ijeecs.v12.i1.pp218-224
5. JULHAM, F. F. AND H. A. A. (2017). Security of Data Communications Between Embedded Arduino Systems with Substitution Encryption. In Second International Conference on Informatics and Computing (ICIC) (pp. 1–5).
6. MITTAL, P. (2019). in Computer Science and A Survey on Internet of Things (IoT) Based Healthcare Monitoring System. International Journal of Advanced Trends in Computer Science and Engineering, 8(4), 1646–1653. Retrieved from http://www.warse.org/IJATCSE/static/pdf/file/ijatcse90842019.pdf
7. MUHAMMAD N AMAN, KEE CHAING CHUA, B. S. (2017). Secure Data Provenance for the Internet of Things. In IoTPTS '17: Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security (pp. 11–14). Retrieved from https://doi.org/10.1145/3055245.3055255
8. NORKHUSHAINI AWANG, NURUL HIDAYAH AHMAD ZUKRI, NOR AIMUNI MD RASHID, Z. A. Z. AND N. A. M. N. (2017). Multi-Agent Integrated Password Management (MIPM) Application Secured With Encryption. In The 2nd International Conference on Applied Science and Technology (ICAST'17) (pp. 1–7).
9. ROSHIDI DIN , ROSMADI BAKAR , RAIHAN SABIRAH SABRI , MOHAMAD YUSOF DARUS, S. J. E. (2019). Performance analysis on secured data method in natural language steganography. Bull. Electr. Eng. Informatics, 8(1), 298–304.
10. SACHIN BABAR, PARIKSHIT MAHALLE, ANTONIETTA STANGO, N. R. P. (2010). Proposed Security Model and Threat Taxonomy for the Internet of Things (IoT). In Recent trends in network security and applications. Third international conference, CNSA 2010, (pp. 420–429). Chennai, India.
11. SEHRAWAT, D., & JOURNAL, I. (2019). International Journal of Advanced Trends in Computer Science and Engineering. International Journal of Advanced Trends in Computer Science and Engineering.
12. SUI, J. C. ; Z. Z. ; H. L. ; R. (2018). An Improved User Authentication Protocol for IoT. In 2018 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC) (pp. 59–593). https://doi.org/10.1109/CyberC.2018.00022
13. TAO, H., BHUIYAN, Z. A., MEMBER, S., ABDALLA, A. N., HASSAN, M. M., ZAIN, J. M., & HAYAJNEH, T. (2018). Secured Data Collection with Hardware-based Ciphers for IoT-based Healthcare. IEEE Internet of Things Journal, PP(X), 410–420. https://doi.org/10.1109/JIOT.2018.2854714
14. VENKATESAN, T. S. AND S. (2018). Authentication of IoT Device and IoT Server Using Secure Vaults. In 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications12th IEEE

© 2022 JPPW. All rights reserved

International Conference On Big Data Science And Engineering (TrustCom/BigDataSE) (pp. 819–824). https://doi.org/10.1109/TrustCom/BigDataSE.2018.00117.

© 2022 JPPW. All rights reserved