An Orchestration Framework for IoT Devices based on Public Key Infrastructure (PKI)

Justice Owusu Agyemang [1], Jerry John Kponyo [2]

Faculty of Electrical/Computer Engineering,
Kwame Nkrumah University of Science and Technology, Kumasi, Ghana.

[1] justiceowusuagyemang@gmail.com, [2] jjkponyo@ieee.org.

***Abstract* - Internet of Things(IoT), allows communication among heterogeneous devices including everyday existing ob- jects, embedded intelligent sensors, context-aware computing, traditional computing networks and smart objects that differ in their design, systems, protocols, intelligence, applications, vendors, and sizes. The direct connection of these hetero- geneous devices to the Internet poses security risks. Some of these devices are sometimes compromised and used in attacks such as DoS. This paper presents an orchestration framework for IoT devices based on public key infrastructure (PKI) aimed at providing logical isolation of IoT devices in situations where these devices have been compromised. We evaluate the performance of the proposed framework based on the computational time of the cryptographic algorithm, performance overhead when the framework was implemented on an IoT device and access revocation/restoration time.**

Keywords – IoT, DoS, RSA, OAEP, ECDSA, TLS

I. INTRODUCTION

The evolution of Internet of Things (IoT) has made it possible for objects to share information about themselves and their data with others. Factors that have led to the rapid development of IoT include: advanced networking capabilities, advancement in connectivity, improvement in cloud computing, availability of low-cost devices and low memory costs [1]. Currently, IoT is made up of a loose collection of disparate, purpose-built networks as shown in Figure 1.

The convergence of these devices and the Internet makes them susceptible to being compromised by attackers in situations where vulnerabilities are found on these devices and are not patched. Such threats include the Mirai botnet, also known as Dyn attack, which took the Internet by storm in the late 2016 by overwhelming several high-profile targets with massive Distributed Denial-of-Service (DDoS) attacks[18]. Furthermore, it was reported in 2017 that St. Jude's cardiac devices had vulnerabilities that could allow a hacker to access a device. Once in, they could deplete the battery or administer incorrect pacing or shocks. The devices, like pacemakers and defibrillators, are used to monitor and control patients' heart functions and prevent heart attacks[19].

Several mitigation techniques have been proposed to address security threats such as Man-In-The-Middle (MITM) attack, routing attack, Denial of Service (DoS) attack [3] and TCP/UDP flood attack [4]. These measures do not tackle instances where IoT devices are compromised and further used in attacking other systems.

This paper presents an orchestration framework for IoT device based on public key infrastructure to logically isolate IoT devices in instances where these devices have been compromised. It further ensures trust between endpoints. The rest of the paper is organized as follows: Section II presents related works. Section III is the motivation for the research. Section IV presents the methodology. Section V presents the results and discussion. Section VI is the conclusion.

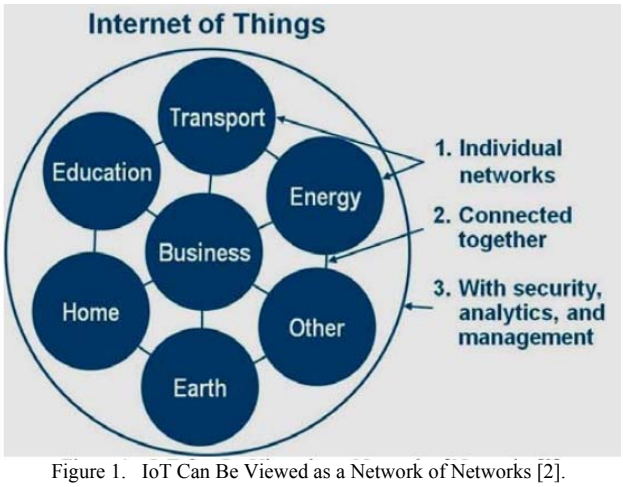

Figure 1. IoT Can Be Viewed as a Network of Networks [2].

II. RELATED WORKS

Currently, several IoT frameworks have been developed in order to make it easy to deploy and maintain IoT applications.

Amazon Web Services (AWS) IoT [5] is a cloud-based IoT platform developed by Amazon for easy connection of smart devices to the cloud and also facilitate interaction among these devices. It supports mutual authentication at all points using mechanisms such as X.509 certificates and incognito identities. AWS IoT cloud assigns a private home directory for each legitimate user. All private data are stored using symmetric key cryptography such as Advanced Encryption Standards (AES)[12].

ARM embed [6] IoT platform through its ecosystem, provides all requirements needed to develop IoT applications for ARM microcontrollers. It uses Transport Layer Security (TLS)/Datagram Transport Layer Security (DTLS) for communication and authentication purposes.

Azure IoT Suite [7], released by Microsoft, provides a set of services that enables end-users to interact with their IoT devices. It uses TLS for secure communications and SHA-256 for authentication purposes.

Google released Brillo/Weave [8] platform for the development of IoT applications. Brillo is an android operation system for lower power embedded devices where Weave is the communication shell for interactions and message- parsing. Link-level security is provided by Weave through the use of SSL/TLS protocol.

Calvin [9], an open source IoT platform developed by Ericsson, enables the development and management of distributed applications for IoT devices. Authentication can be done locally or through an external machine and also using a RADIUS server. Secure communication is done using TLS/DTLS protocol.

HomeKit [10] is an IoT framework developed by Apple. It facilitates managing and controlling connected accessories and appliances. Securing communication is achieved using TLS/DTLS with AES-28-GCM and SHA-256.

Kura [11] is an Eclipse IoT project that provides a Javabased framework for IoT gateways that run M2M applications. It consists of security components such as a security service, a certificate service, a Secure Socket Layer (SSL) manager and a cryptography service. All communications are secured using SSL/TLS protocol.

Orchestration systems aimed at providing automated workflow to physical resources (deployment and scheduling) and workload execution management with Quality of Service (QoS) control [13], [1] have been proposed.

Other researchers have also proposed an architecture that allows the orchestration of objects that are part of the Internet of things based on Simple Business Modeling Notation (SBMN) [14] and also orchestration frameworks based on Software-Defined Networking (SDN) [15], [16], [17].

III. MOTIVATION

From the related works, a number of IoT frameworks have been developed. These frameworks enable end-users to interact with IoT devices. Some of these frameworks lack security and reliability [15], [16], [17]. Current IoT frameworks that have been developed or proposed do not provide a logical isolation of IoT devices when they have been compromised due to an exploited vulnerability. Hence, these devices are used as endpoints to perform several attacks.

IV. METHODOLOGY

The orchestration framework employs the use of Public Key Infrastructure in providing logical isolation of IoT devices in instances where these devices are compromised.

A. Architecture

The entire architecture for the orchestration framework is shown in Figure 2.

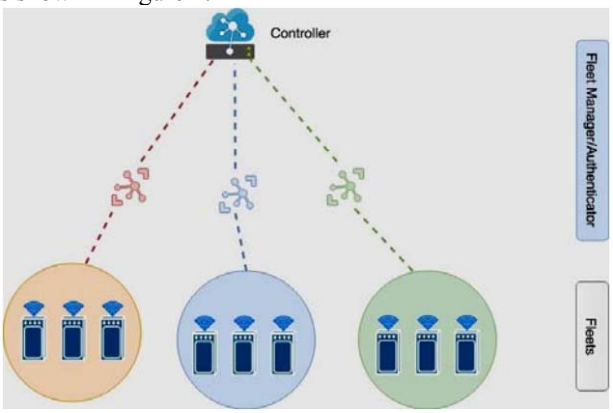

Figure 2. Architecture

The architecture consists of IoT devices which together form a fleet and are connected to a fleet manager. The various fleet managers are also connected to a main controller. The flow of communication downstream is from the controller to the fleet managers then to the fleet devices. The reverse takes place for upstream communication.

The fleet managers are responsible for authorizing IoT devices belonging to a particular fleet. The controller coordinates and authorizes the various fleet managers. The authorization and management functions of both the fleet managers and the controller can be chained together through Network Function Virtualization (NFV).

B. Flow of Communication

Communication from one endpoint to another is secured using public key cryptography. A pair of asymmetric keys (P riv, P ub) keys are generated at each endpoint. The (P ub) keys are used in encrypting data between endpoints where as (P riv) is used for decrypting received data and also verifying a received data as valid or not, as shown in Figure 3.

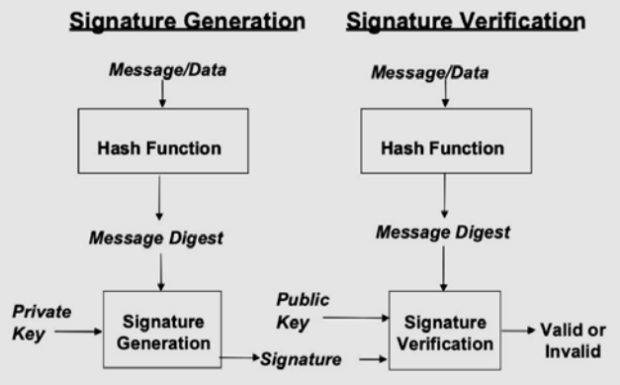

Figure 3. Signature Generation and Verification [20]

When a fleet manager wants to communicate to an IoT device belonging to its fleet, it encrypts the data using the (P ub) key of the IoT device. The reverse goes for communication between an IoT device and a fleet manager. This is illustrated in Figure 4.

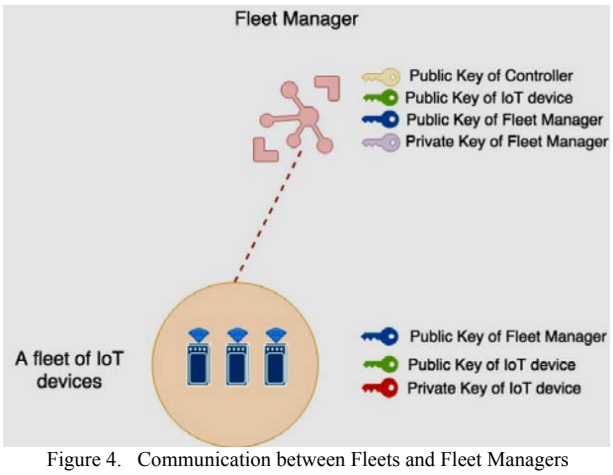

Figure 4. Communication between Fleets and Fleet Managers

When a fleet manager wants to communicate to the controller, it encrypts the data with the controllers (P ub) key. The received data is decrypted by the controller using its (P riv) key. The reverse goes for communication between the controller and fleet managers. This is shown in Figure 5.

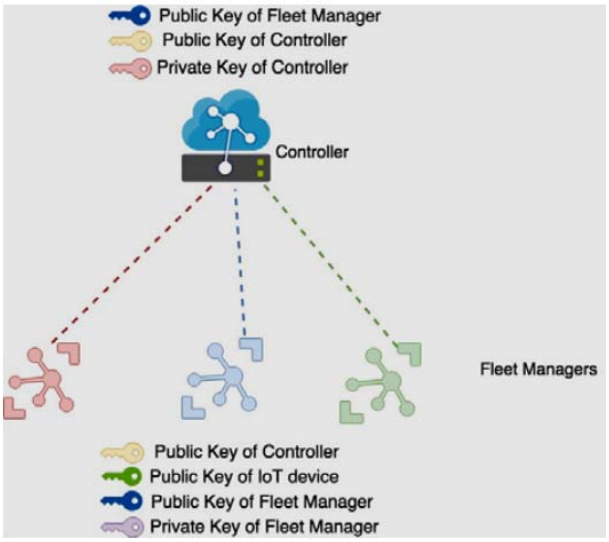

Figure 5. Communication between Fleet Managers and Controller

C. Encryption Algorithm

The orchestration framework implements Rivest-ShamirAdleman (RSA) cryptography system with Optimal Asymmetric Encryption Padding (OAEP). RSA cryptosystem generates (P riv, P ub) keys based on the practical difficulty of the factorization problem of the product of two large prime numbers; with a modular exponentiation for all integers m ($0 \leq m \leq n$) , $(me)d \equiv m \pmod n$, knowing e and n or even m it is extremely difficult to find d. We employ OAEP because it adds an element of random- ness into the RSA algorithm. Also, OAEP prevents partial decryption of cipher text by ensuring an adversary cannot recover any portion of the plaintext without being able to invert the one-way permutation function. The algorithm for (P riv, P rub) keys generation using RSA and RSA with OAEP is shown in Algorithm 1 and 2 respectively.

Algorithm 1 RSA $(Priv, Pub)$ Key Generation

```
randomSeed = Random.generate()
rsa = RSA.generate(keyLength, randomSeed)
pubKey = rsa.publicKey()
privKey = rsa
```

Algorithm 2 RSA with OAEP $(Priv, Pub)$ Key Generation

```
randomSeed = Random.generate()
rsa = RSA.generate(keyLength, randomSeed)
pubKey = PKCS1_OAEP.new(rsa.publicKey())
privKey = PKCS1_OAEP.new(rsa)
```

D. Communication Protocol

The framework uses HyperText Transport Protocol (HTTP) with Transport Layer Security (TLS). To overcome the limitation of HTTP being synchronous, we employ the use of websocket for asynchronous requests and responses. This enables a two-way communication process with faster request and response times and also minimal performance overhead.

E. Message Format

The message format consists of a message to- ken/identifier, a signature verification field and the payload (shown in Figure 6).

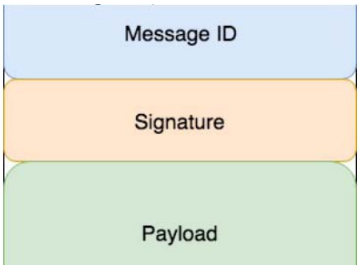

Figure 6. Message Format

The message token/identifier is generated using the operating system's random number generation algorithm. For operating systems that do not implement the (os.urandom) random number generation algorithm, WichmannHill's random number generation algorithm can be implemented. The computational time between the two is quite insignificant (shown in Figure 7). WichmannHill's algorithm averages 0.0000751 seconds whereas the System Random algorithm averages 0.0000597 seconds. The message token/identifier guards against message replay attacks between endpoints. The signature field is used to validate the endpoints. The payload contains the data to be received by the endpoint; which can be encrypted using the (P ub) key of the receiving node.

F. Evaluation of Proposed Orchestration Framework

The entire orchestration framework was implemented using a RaspberryPi [22] as an IoT device belonging to a particular fleet. The fleet manager and the controller were implemented in the cloud using NFV. The following key performance indicators were used in measuring the efficiency of the orchestration framework:

- The computational time of the public key cryptographic algorithm.
- The performance overhead of the algorithm on end-points (IoT devices).
- The time it takes to revoke and restore an IoT device's access.

V. RESULTS AND DISCUSSION

A benchmark performance on the two algorithms was conducted on a RaspberryPi [22] to determine which of the two is less computationally intensive and what length of bits will be appropriate for use. This is shown in Figure 8. Key lengths from 1024 to 4096 for both RSA and RSA with OAEP produce an average computational time of 0.147 seconds and 0.408 seconds as compared to fast Elliptic Curve Digital Signature Algorithm (ECDSA) with an average computational time of 9.085 seconds [23]. A key length of 2048 was used in the orchestration in the RSA-OAEP algorithm. This offers both low computational overhead together with secure communication process.

The average CPU utilization of the framework implemented on an IoT device averages 0.8% (shown in Figure 10. The revocation and restoration time averaged 4.225 seconds and 4.272 seconds, as shown in Figure 7 below.

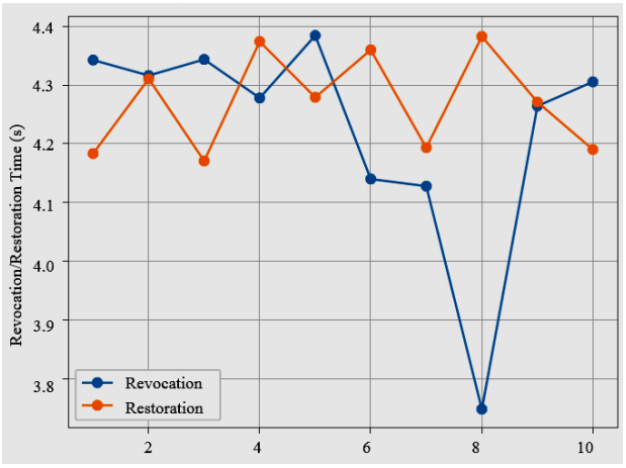

Figure 7. Revocation and Restoration Time

VI. CONCLUSION

In summary, the paper presents an orchestration frame- work based on public key infrastructure to provide logical isolation of Internet of Things devices in instances where these devices are compromised. This can help mitigate security attacks such as DoS since the access of these devices can be revoked. The proposed orchestration framework can be used by IoT device manufacturers to deliver firmware updates to IoT devices as a means of patching identified vulnerabilities. Future works will consider providing a cross proxy between the framework and other existing protocols such as Constrained Application Protocol (CoAP); which is used by resource constrained IoT devices.

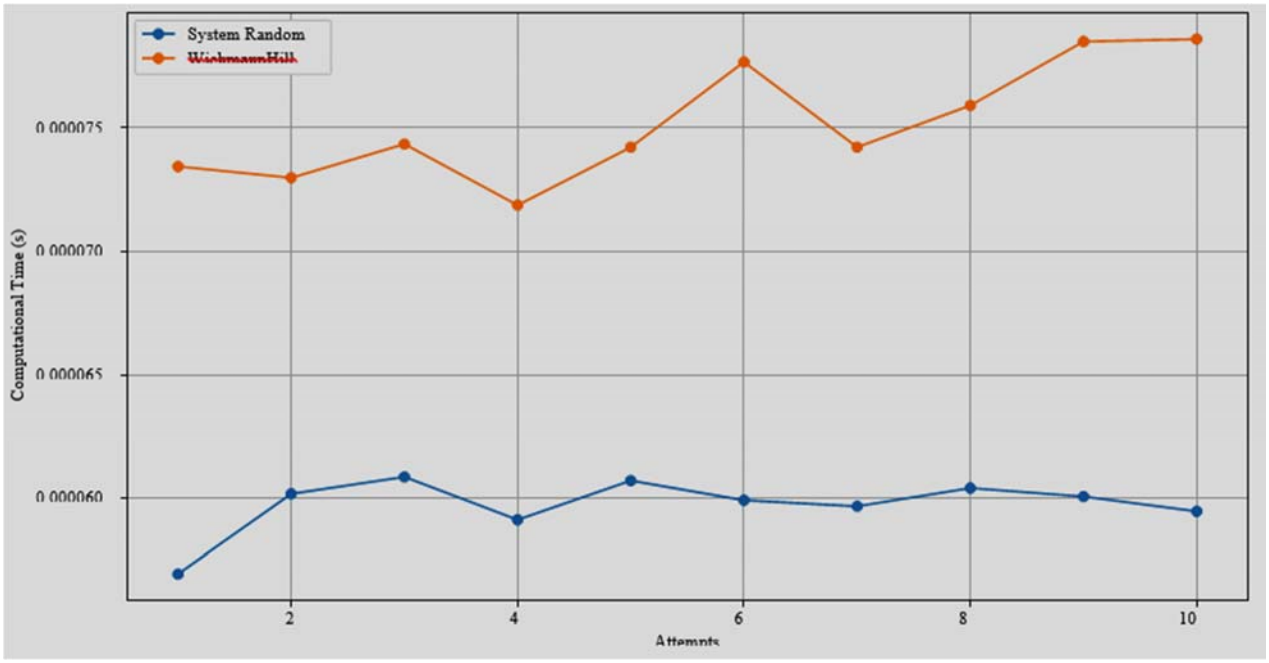

Figure 8. Comparison of the Computational Time of WichmanHill and the Operating System's Random Number Generation Algorithm for 10 attempts each of 1000 iterations.

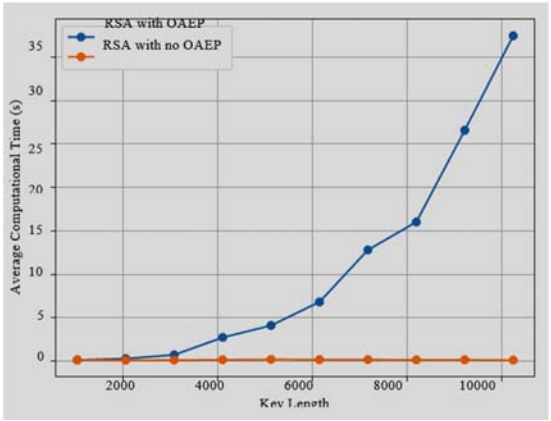

Figure 9. Benchmark Performance of RSA and RSA with OAEP

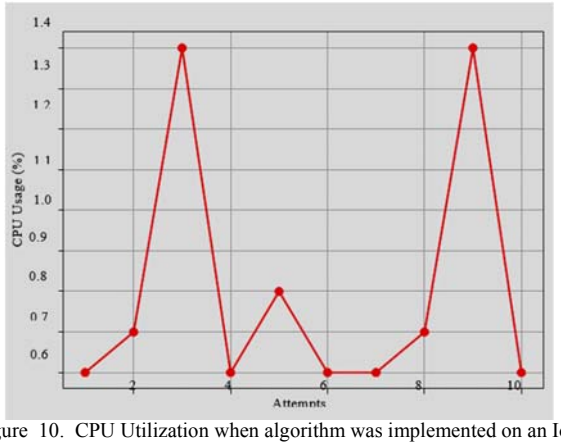

Figure 10. CPU Utilization when algorithm was implemented on an IoT

REFERENCES

[1] I. Lee and K. Lee, The internet of things (iot): Applications, investments, and challenges for enterprises, Business Horizons, vol. 58, no.4, pp. 431–440, 2015.

[2] Dave Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything" Cisco Internet Business Solutions Group (IBSG) , pp. 4, April 2011.

[3] Garcia-Morchon O., Kumar S., Struik R., Keoh S., Hummen R.,Security considerations in the IP-based Internet of Things, IETF Internet-Draft, 2013.

[4] Prabhakaran Kasinathan, Claudio Pastrone, Maurizio A. Spirito and Mark Vinkovits "Denial-of-Service detection in 6LoWPAN based Internet of Things", 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob).

[5] Amazon, "Aws iot framework", https://aws.amazon.com/iot, [Ac- cessed Online], January 2019.

[6] ARM, "Arm mbed iot device platform", http://www.arm.com/products/iot-solutions/mbed-iot-device-platform, [Accessed Online], January 2019.

[7] Microsoft, "Tap into the internet of your things with azure iot suite", https://www.microsoft.com/en-us/cloud-platform/internet- of-things-azure-iot-suite, [Accessed Online], January 2019.

[8] MSV J. , "Google brillo vs. apple homekit: The battleground shifts to iot", https://www.forbes.com/sites/janakirammsv/2015/10/29/google-brillo-vs-apple-homekit-the-battleground-shifts-to-iot/, [Accessed On- line], January 2019.

[9] Ericsson, "Open source release of iot app environment calvin", https://www.ericsson.com/en/blog/2015/6/open-source-release-of-iot- app-environment-calvin, [Accessed Online], January 2019.

[10] Apple, "The smart home just got smarter", http://www.apple.com/ios/home/, [Accessed Online], January 2019.

[11] Organization E., "Kura framework", http://www.eclipse.org/kura/, [Accessed Online], January 2019.

[12] Mahmound Ammar, Giovanni Russello, Bruno Crispo, "Internet of Things, A Survey on the Security of IoT Frameworks", Journal of Information Security and Applications, pp. 8-27, 2018.
[13] Zhenyu Wen, Renyu Yang, Peter Garraghan, Tao Lin, Jie Xu, and Michael Rovatsos, Fog Orchestration for IoT Services: Issues, Chal- lenges and Directions, pp. 1, IEEE Internet Computing - March 2017.
[14] Alejandro G., Manuel A. A., Jordan P. E., Oscar S. M., Juan M. C. L., Cristina P. G-B., Introduction to Devices Orchestration in Internet of Things Using SBPMN , International Journal of Interactive Multimedia and Artificial Intelligence , December 2011.
[15] Galluccio, L., Milardo, S., Morabito, G. and Palazzo, S., SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks, IEEE Conference on Computer Com- munications (INFOCOM) (2015), 513-521.
[16] Gante, D., Aslan, M. and Matrawy, A. , Smart wireless sensor network management based on software-defined networking, 27th Biennial Symposium on Communications (QBSC) (2014), 71-75.
[17] Miyazaki, T., Yamaguch, S., Kobayashi, K., Kitamichi, J., Guo, S., Tsukahara, T. and Hayashi, T. , A software defined wireless sensor network, International Conference on Computing, Networking and Communications (ICNC) (2014), 847-852.
[18] Manos Antonakakis, Tim April et al, "Understanding the Mirai Botnet", 26th USENIX Security Symposium, pp. 1, August 2017.
[19] http://money.cnn.com/2017/01/09/technology/fda-st-jude-cardiac-hack/, [Accessed Online], 8th July, 2018.
[20] Digital Signature Standard, Information Technology Laboratory, National Institute of Standards and Technology, pp. 9, July 2013.
[21] Suchitra.C and Vandana C.P, "International Journal of Computer Science and Mobile Computing", Vol. 5, Issue. 1, January 2016, pg.133 – 139.
[22] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. [Accessed Online], 6th January 2019.
[23] https://github.com/AntonKueltz/fastecdsa [Accessed Online], 12th January 2019.