Computer Security Symposium 2020
26 - 29 October 2020

Empowering Resource-Constraint IoT Gateways with Port Knocking Security

Yuta Inoue[1,*] Seiya Kato[1] Aamir H. Bokhari[1] Katsunari Yoshioka[1†]
Tsutomu Matsumoto[1†]

Abstract: Digital boom brought empowerment to seamless connectivity by enabling manufacturers to harness the power of the Internet into their products, opening up the world of the Internet of Things (IoT). However, such connectivity has also brought the side effect of such power being abused by the bad actors, who scan open ports for services and then make their way in by exploiting vulnerabilities in the system. There is an increasing need to enable IoT devices to be fully patched and secured. This paper studies a unique but practical method of security called "Port Knocking." Our experimental results on a resource-constraint IoT gateway show that port knocking not only secures the device and provides a secure remote management option, but also makes power consumption lower. The results obtained make it an effective security layer for securing IoT devices.

Keywords: Authentication, IoT Gateway, Port Knocking, Remote Management, System Security

1. Introduction

In the 21st century, the Internet of Things (IoT) has opened up a new horizon for entrepreneurs and hackers. By 2020, Gartner expects 20 billion IoT devices connected to the Internet and estimates over 25% of known attacks will involve IoT, whereas the IT security budgets for IoT would be less than 10% [1]. Consumer's dependency on technology and plug-n-play concept makes it easy to compromise the IoT devices due to the absence of human roles as users usually do not engage in the management of their devices. The IoT device resources are also getting scarce due to small sizes designed for portable use. These will increase the opportunities for hacking [2]. Furthermore, botnets are using self-propagating malware, such as "Mirai" for attack purposes [3]. Thus, cyber-attacks exploiting IoT vulnerabilities in-network services and poor security are increasing.

Due to more focus on security-by-design, various security ideas are coming up mainly for the next-generation IoT devices. The existing IoT devices already in the market are being left unsecured. End users do not have adequate technical knowledge to fix security issues by themselves. The lack of security management and limited resources on such IoT devices is a big challenge. A possible approach is to update the security via remote patching, but that is also subject to common attacks on the well-known services, such as Telnet, FTP, SSH, and Web. Therefore, in this study, we focused on the challenge of securing remote management for resource-constraint IoT gateway devices already available in the market and power consumption by the suggested method to ensure available resources are not stressed out by the security.

Due to resource constraints and lack of user management, we cannot secure them with conventional security methods, such as firewalls. Thus, we examined port knocking, which is commonly used by system administrators of large systems for remote management. We tested the IoT gateway with two types of port knocking methods. One was based on python script utilizing pseudorandom number generator (PRNG) and chaotic random number generator (CRNG) algorithms. Whereas, the other was based on stream cipher utilizing authenticated encryption with associated data (AEAD) algorithm. In our lab test, we found the first method consuming almost 50% of CPU resources compare to 9-15% of CPU usage when updating the knocking sequence and generating key. Therefore, further testing was done based on the second method to measure the effectiveness of hiding port 22/TCP (SSH) when not in use, and device power consumption. The results showed that the device with port knocking enabled had zero SSH login attempt versus 1039. This shows that stream cipher based port knocking was able to reduce the attack surface significantly, adding to the security of the IoT device. The power consumption also was less than the one with no port knocking because of less number of packets received.

Based on the results of the experiments conducted to measure the effect of using port knocking security feature on resource-constraint IoT gateway, we can conclude that the experimental results imply the power consumption overhead by receiving incoming session requests (from scanners/malware on the Internet) would easily exceed the power consumption for running port knocking service. Thus, running port knocking service would be beneficial in terms of not only security enhancement but also power consumption.

---

1 Graduate School of Environment and Information Sciences, Yokohama National University.
† Institute of Advanced Sciences, Yokohama National University.
* inoue-yuuta-zr@ynu.jp

2. Related Work

Due to the popularity of IoT, a lot of research has been focused mainly on the IoT threats, vulnerabilities, attacks, limitations, and challenges. Many researchers have also examined the security of IoT and possible countermeasures. For example, Paper [2] highlights the issues with IoT devices and various types of IoT attacks along with possible countermeasures. It rules out the use of conventional cryptography in small IoT devices or limits it due to resource constraints. The focus then turned towards finding the kind of algorithms that can be used in the Internet of Things environment. Paper [4] proposed storing of data in the cloud directly from a smart device or via a gateway using lightweight encryption algorithms on IoT devices to address the issues of privacy and security. However, its focus was more on the privacy and security of the data than the IoT device itself.

The authors in paper [5] provided a port knocking approach utilizing symmetric key encryption, Message Authentication Code (MAC), and an encrypted keep-alive system to secure service port. However, it is limited to TCP ports with static IP configuration only and was tested using a hardware (CPU = Intel Core i7 3770 @ 3.40 GHz; RAM = 8GB DDR3) with plenty of resources. With respect to resources, only physical memory usage and network bandwidth was examined. This paper did not look at the device power consumption, which is more critical in today's resource-constraint IoT gateways. In paper [6] the author introduced an advanced method of port knocking that can reduce attacks by producing difficult to guess port knocking sequences based on PRNG and CRNG algorithms. As this solution utilizes both TCP and UDP ports, therefore, it can be a candidate for our purposes. Similarly, another paper [7] suggests a different algorithm based on RFC7539 [8] that pairs ChaCha20 stream cipher with Poly1305 authenticator to create an AEAD scheme for IoT applications to run on ARM Cortex-M4 processors. A security analysis report by KDDI Research Inc. concludes that they could not find the weakness in the AEAD algorithm [9]. Therefore, this method can also be a candidate for our purpose as it can be used for random port knocking sequences and key generation.

2.1 Port Knocking

Port knocking is not a new concept as it has been used by system administrators to manage the servers remotely. However, its application on IoT devices is relatively new. A common method of attack is to first look for open service ports using a port scanner. In the case of port knocking, the service port is closed by default and opens for service only for the requester that sends the right sequence of packets to the firewall for authentication, as shown in Fig.1. This figure was created by referring to [10].

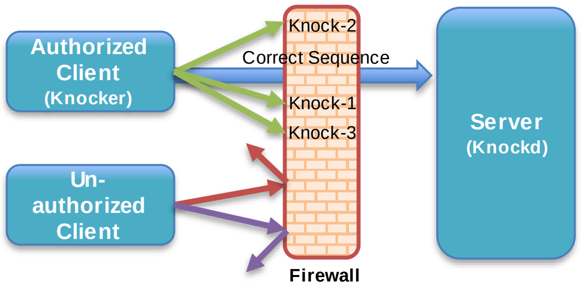

Fig. 1 Port knocking

In case of Linux based devices, a "knockd" daemon process (server) monitors the knock sequence and open/close ports via the "iptables". This knock sequence must be randomly generated and synchronized between the client and server in order to avoid replay attacks or man-in-the-middle (MITM) attacks. With the service port closed, the target port cannot be confirmed during scanning, and therefore, an attacker cannot target it [10].

Based on the research work in paper [6] and [7], we selected two approaches for our port knocking testing on IoT gateway device because they can be used in IoT environment and can generate completely random numbers using two sets of completely different algorithms as follows:

2.1.1 Python Script based Port Knocking Daemon

In this approach, the port knocking mechanism (python script based "knockd" server) is setup by a Python script that produces pseudo-random port numbers using system time as a seed with a PRNG algorithm, and then again combining the result with the CRNG algorithm for protocols (TCP/UDP), as shown in Fig.2. This figure was created by referring to [6].

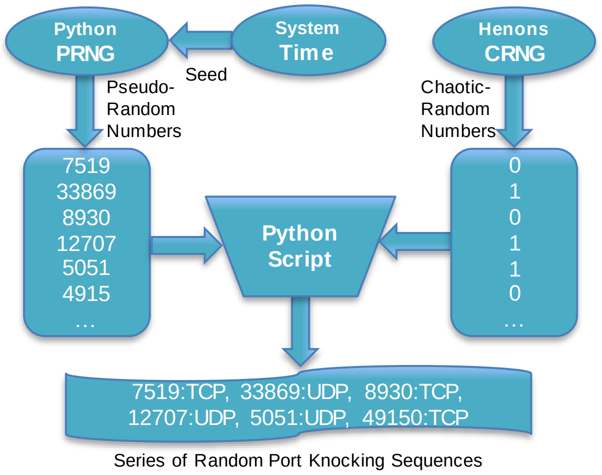

Fig. 2 Python-based PRNG-CRNG "knockd" server

This creates a system that always generates the same knock sequence based on the same algorithms running on the server and the client side [6]. Due to the pseudo-random nature of the algorithm, it mitigates playback and MITM attacks.

2.1.2 Stream Cipher based Port Knocking Daemon

The second method based on stream cipher knockd server [8] was tested with a 256-bit secret key shared between server and client, generating a 512-bit key stream from the secret key and time using Authenticated Encryption with Associated Data algorithm, as defined in IETF's RFC 8439 [11]. Key-stream was split in 10 knock sequences (1 sequence = 3 ports = 48bits). Frequency for updating knock sequence was set at every 60-sec intervals. The key-stream was regenerated every 10 min interval [12].

3. Proposed Method

For testing our proposed solution, we selected commonly available off-the-shelf IoT gateway devices [13], [14] having a raspberry-pi hardware configuration, as shown in Table 1.

Raspberry-pi is one of the popular off-the-shelf hardware being used commonly as an IoT gateway for connecting various kinds of sensors with applications.

3.1 Test Setup

3.1.1 Test-1: Port knocking effectiveness in terms of CPU usage

First, we need to select a port knocking method approach that does not put too much stress on the CPU of the IoT device. We will install and test both approaches (Python script based port knocking and stream cipher based port knocking) on the same IoT test device one-by-one, targeting port 22/TCP for SSH service with port knocking and measure the CPU usage via log analyzer, as shown in Fig.3.

3.1.2 Test-2: Port knocking effectiveness in terms of security

Once we have identified an efficient port knocking method, then we will examine how effectively it can reduce the unauthorized SSH login attempts on a default port 22/TCP, as shown in Fig.4.

Table 1 IoT test devices

		VX2	BX1
CPU	Model	Intel Atom E3805	Intel Atom ® Processor
	Speed	1.33 GHz	500 MHz
	Cache	1024 KB	1024 KB
Memory		2 GB	1 GB
Storage		32 GB	4 GB

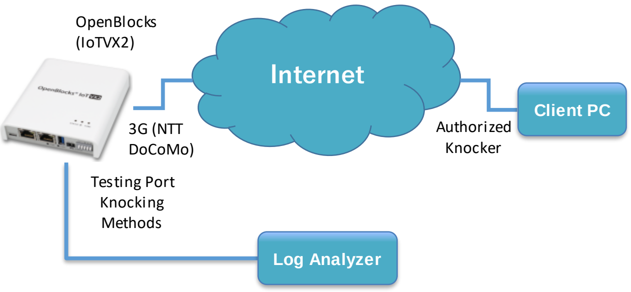

Fig. 3 Test-1 setup

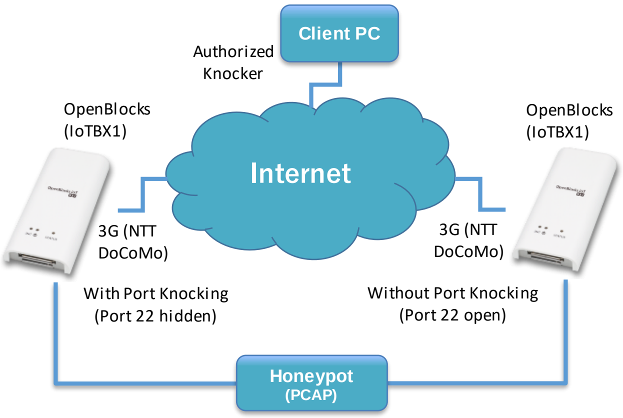

Fig. 4 Test-2 setup

3.1.3 Test-3: Port knocking effectiveness in terms of power consumption

In order to examine the power consumed with and without the port knocking feature, we used two identical IoTBX1 gateway devices that were running on the same software and hardware. USB testers were connected to each IoT device, and these testers were then connected to a self-powered USB hub. A note PC was also connected with the USB hub for observational purposes, as shown in Fig.5.

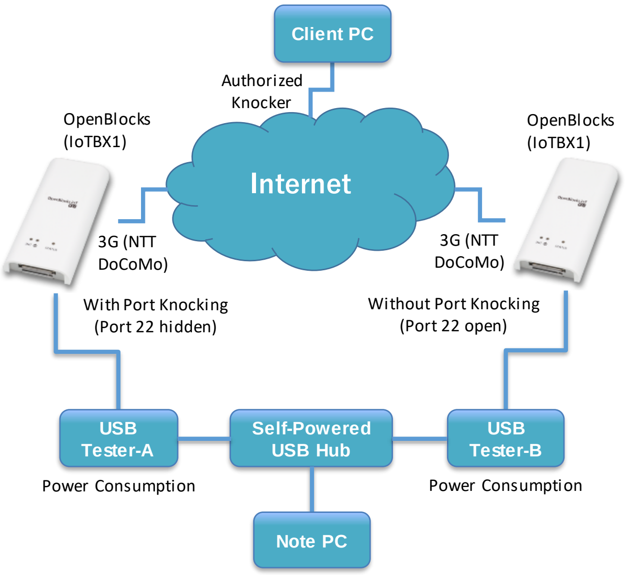

Fig. 5 Test-3 setup

3.2 Experiment Results

In the case of test-1 for finding out the effectiveness of port knocking in terms of CPU usage, the Python-script-based approach of port knocking method consumed more CPU resources than the Stream Cipher based port knocking method, as shown in Fig.6.

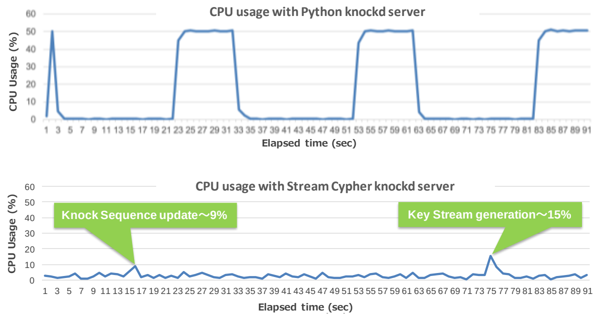

Fig. 6 Port knocking algorithm's CPU usage comparison

As the Python-script-based approach consumed almost 50% more CPU resources than the Stream Cipher based approach, therefore, further testing was focused on utilizing stream cipher for checking its effectiveness against detection of SSH port by putting it in the honeypot, next to another device without port knocking. For further testing, we used the IoTBX1 model as it is more resource constraint than IoTVX2. The test-2 results are summarized in Table 2.

Table 2 Port knocking security effectiveness.

	With Port knocking	**Without Port knocking**
SSH Login Attempts	0 Times	1,039 Times
Total Number of Packets Received	232	43,739
Source IP Addresses	133	202

The stream cipher based port knocking method proved to be quite effective in hiding port 22tcp as we detected only 232 packets coming from 133 different source IP addresses and observed not a single SSH login attempt (0 times) on the IoT test device with port knocking. This is due to the fact that port 22/TCP was hidden and did not respond to any SYN packets it received. Therefore, not many packets were observed on port 22/TCP with port knocking.

In comparison, the device with no port knocking had 1,039 SSH login attempts from 202 different IP sources due to detectable SSH port. As port 22/TCP was open and responded to SYN packets, therefore, the scanning host follows up with other packets to complete the TCP handshake, attempting to establish or exploit the SSH service. Hence, we see a large number of packets (43,739) in case of the device with no port knocking compare to a small number of packets (232) on the device with port knocking enabled. This shows that stream cipher based port knocking was able to reduce the attack surface significantly, adding to the security of the IoT device.

Next, we look at the test-3 results for assessing power consumption with and without the port knocking feature.

(a) Voltage Stability: In order to make sure we do not observe any other influence, we first connected both USB testers only to the USB hub and measured voltage for 24 hours. Both came out to be stable around 5.14 volts, as shown in Fig. 7.

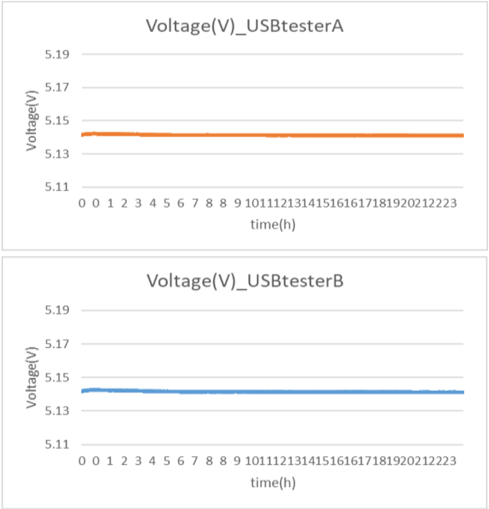

Fig. 7 Voltage stability.

(b) Confirming power consumption without port knocking: Next, we measured the power consumption of both IoT devices without port knocking. We observed almost similar readings on both IoT devices, as shown in Figures 8 and 9.

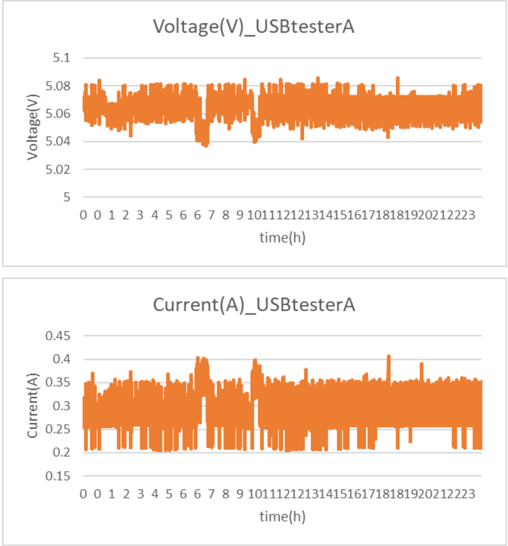

Fig. 8 IoTBX1 – tester-A

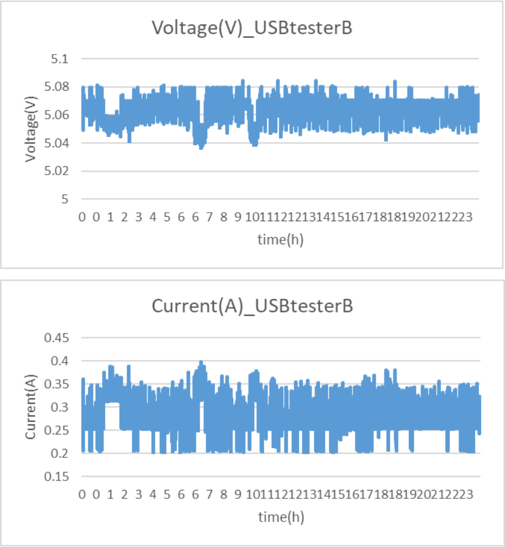

Fig. 9 IoTBX1 – tester-B

(c) Confirming power consumption with port knocking: After ensuring we have stable readings without port knocking, we then enabled stream cipher based port knocking on the IoTBX1 device connected to the USB Tester-A. We measured the power consumption of both IoT devices for one week. We also observed the number of packets received by running "netstat–statistics" command every hour. The results showed that the current slowly increases with the number of packets received. Since the IoTBX1 device connected to USB Tester-A was running the port knocking feature, therefore, the number of packets received on its port 22/TCP was quite less compared to the device without the port knocking feature, as shown by the graph in Fig.10.

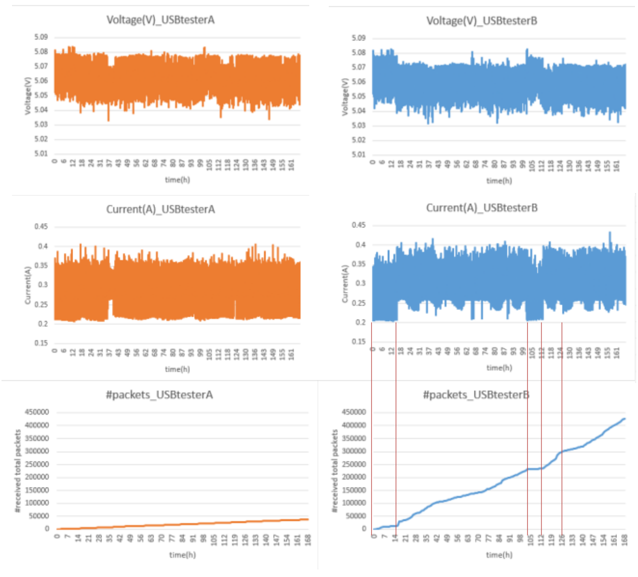

Fig. 10 Port knocking vs. without port knocking

4. Conclusion

In the case of port knocking, the stream cipher based algorithm is much practical to use on resource-constrained IoT gateway devices. It not only kept the CPU usage to a very reasonable level and hid the default service port effectively, but also helped in greatly reducing the unauthorized traffic coming to that service port. This could provide a secure remote management option for authorized users without causing much overhead on existing resources of the IoT device.

Hence, we can conclude that the experimental results imply the power consumption overhead by receiving incoming session requests (from scanners/malware on the Internet) would easily exceed the power consumption for running port knocking service. Thus, running port knocking service would be beneficial in terms of not only security enhancement but also power consumption.

5. Considerations

The current study was carried out by testing the port knocking feature only for the SSH service using key-authentication on the default port 22/TCP. We have tested it on the raspberry-pi hardware platform for determining the effectiveness of hiding port from unauthorized traffic (from scanners/malware on the Internet) and its effect on the IoT device's power consumption, with and without port knocking security feature. Though this proposed solution has shown encouraging results, however, we have not tested it for the other services and non-default ports. Also, what other security methods can be applied and the choice of security layers must take into account the available resources as we saw around 50% CPU utilization in the case of Python script based port knocking solution.

6. Summary and Future Work

This study has provided us some promising results for using the port knocking security concept with which we can provide a secured channel to remotely manage IoT devices via SSH service without exposing it to unwanted traffic. This method also helps in lowering the total number of packets received by the IoT device on the hidden ports that helps with power consumption.

For future work, the port knocking security feature can be coupled with other lightweight security options to provide a layered defense (defense-in-depth) approach for the resource-constraint IoT gateway devices. A longer time period testing is recommended to explore its effectiveness with services running on the non-default TCP /UDP ports as well as with multiple services at the same time, using both default and non-default ports.

Acknowledgement A part of this work was obtained from an EU-Japan collaboration project "Multi layered Security technologies to ensure hyper-connected smart cities with Blockchain, Big Data, Cloud and IoT (MSEC)", jointly funded by the European Union's Horizon 2020 research and innovation program (contract No 814917) and by the Commissioned Research of National Institute of Information and Communications Technology (NICT), JAPAN (contract No. 19501).

References

[1] Hung, M.: Leading the IoT, Gartner Inc. (2017), available from <https://www.gartner.com/imagesrv/books/iot/iotEbook_digital.pdf > (accessed 2020-07-14).

[2] Ramakrishna, C., Kumar, G.K., Reddy, A.M. and Ravi, P.: Survey on various IoT attacks and its countermeasures, *International Journal of Engineering Research in Computer Science and Engineering* (*IJERCSE*), Vol.5, No.4, pp.143-150 (2018).

[3] Jaramillo, L.E.S.: Malware detection and mitigation techniques: Lessons learned from Mirai DDOS attack. *Journal of Information Systems Engineering & Management*, Vol.3, Issue 3, No.19 (2018), available from <https://doi.org/10.20897/jisem/2655> (accessed 2020-07-14).

[4] Shafagh, H., Hithnawi, A., Droescher, A., Duquennoy, S. and Hu, W.: Poster: Towards encrypted query processing for the Internet of Things, *Proc. 21st Annual International Conference on Mobile Computing and Networking* (*MobiCom'15*), pp. 251–253 (2015).

[5] Sathyadevan, S. ., Vejesh V., Doss, R. and Pan, L.: Portguard - an authentication tool for securing ports in an IoT gateway. *IEEE International Conference on Pervasive Computing and Communications Workshops* (*PerCom Workshops*), pp. 624-629 (2017).

[6] Andreatos, A.S.: Hiding the SSH port via smart Port Knocking, *International Journal of Computers*, Vol. 11, pp. 28-31 (2017).

[7] Santis, F.D., Schauer, A. and Sigl, G.: ChaCha20-Poly1305 authenticated encryption for high-speed embedded IoT applications. *Design, Automation & Test in Europe Conference & Exhibition* (*DATE'17*), pp. 692-697, (2017).

[8] ChaCha20 and Poly1305 for IETF Protocols (RFC 7539), available from <https://tools.ietf.org/html/rfc7539> (accessed 2020-06-15).

[9] KDDI Research Inc.: Security analysis of ChaCha20-Poly1305 AEAD. Cryptography Research and Evaluation Committees, Japan, pp.32-33 (2016), available from <https://www.cryptrec.go.jp/exreport/cryptrec-ex-2601-2016.pdf> (accessed 2020-07-14).

[10] Kereki, F.: Implement port-knocking security with Knockd, *Linux Journal* (2010), available from <https://www.linuxjournal.com/magazine/implement-port-knocking-security-knockd> (accessed 2020-06-24).

[11] ChaCha20 and Poly1305 for IETF Protocols (RFC 8439), available from <https://tools.ietf.org/html/rfc8439> (accessed 2020-06-22).

[12] Tex2e/ChaCha20-Poly1305 – GitHub, available from <https://github.com/tex2e/chacha20-poly1305> (accessed 2020-06-25).

[13] OpenBlocks IoT VX2. Plat'Home, (2019), available from <https://www.plathome.co.jp/product/openblocks-iot/vx2/> (accessed 2020-06-30).

[14] OpenBlocks IoT BX1. Plat'Home, (2019), available from <https://www.plathome.co.jp/product/openblocks-iot/bx1/> (accessed 2020-06-30).