A Pluggable Module for Enabling a Trusted Edge Device Management System Based on Microservice

Shih-Hsiung Lee and Jue-Zhi Liu

***Abstract*—In the development of the intelligent Internet of things, edge computing plays a key role. Not only that it has the characteristics of a quick response, but it can also effectively reduce the burden of cloud computing. In addition, it can also extend the application of network edge through the coexistence and collaboration with the cloud system. However, with a large number of edge devices being deployed, the previous remote device management system will face the challenges of resource constraints and software firmware compatibility. In addition, implementing remote fault management, configuration management, accounting management, performance management, and security management through edge device management is a key task. Therefore, this study has designed and implemented a trusted edge device management system based on a microservice and a plug-and-play hardware management device. The running state of the edge device, the functions of remote device control, and system restart are monitored remotely by in- and out-of-band management modes. In addition, in terms of data transmission security, the design concept of a trusted platform module is introduced to realize data encryption and authentication, and ensure security and reliability. In this paper, the operational feasibility of management services based on containerized microservice in edge devices is verified with system benchmarking tools without affecting system performance. According to the experimental results, the proposed architecture in this study can be effective in edge device management.**

***Index Terms*—Edge computing, Internet of things, microservice, remote device management.**

I. Introduction

With the intelligent era and flourishing development of information technology, mass data produced as millions of sensors and devices connect through the Internet of things are sent to cloud servers by embedded systems (e.g., processors, microcontrollers, and communication tools) and processed and stored by the cloud management platform. However, this process brings great pressure on the network transmission cost [1]. In order to solve these problems, edge computing has become widely used as a solution in various industries [2]. According to the edge computing architecture, moving computing resources to the network edge closer to end-users or terminal devices can reduce network bandwidth consumption and improve latency. After deploying Internet of things devices, regular software or firmware updates, bug fixes, potential fault crisis detection, and device maintenance are needed. Therefore, an efficient device management system is needed to effectively manage and maintain Internet of things devices [3], [4]. The Internet of things device management mainly comprises two aspects: Network management and device management [5]. The former mainly integrates heterogeneous networks and collects and analyzes mass data to provide effective decisions, while the latter provides information on device location and status. Ultimately, reliable communication protocols can help make the Internet of things services accessible over the network [6]. Therefore, remote fault management, configuration management, accounting management, performance management, and security management (FCAPS) are key tasks for complex and diverse integration scenarios [7]. Add to this, many large international cloud platform providers launch their edge computing products to expand computing to edge ports and form service models with cloud collaboration. For example, HUAWEI CLOUD supports product model definition, visual management of structured data storage, and device life cycle and assists users in unified remote monitoring, management, and edge port maintenance from cloud platforms [8].

Due to the scale expansion, heterogeneity, and interoperability of the Internet of things, building a highly extensible and maintainable Internet of things system is challenging. A microservice architecture, one of the best solutions for building Internet of things systems, helps realize independent service deployment and uses heterogeneous technologies to run on various devices [9], [10]. A microservice is a distributed structure that separates complex applications into lightweight services. Each component runs independently in programs and is managed in a distributed manner. Further, each service has its own code and can communicate with other services by defining API. Microservice can minimize the interdependency between components. Generally speaking, the failure of a single service has a lower affect on the operation of the entire system, making its operation highly flexible and secure, which is almost impossible in a centralized management architecture. Microservice architectures make systems more extensible and stable [11]. Therefore, this paper will deploy edge device and management platform services in a distributed manner based on microservice architectures.

Data exchange is a basic operation in the Internet of things ecosystems. However, such open mass communications are particularly attractive to users with illegal intentions [12]. Therefore, data security and privacy protection are challenges for edge device management. Insecure edge devices and net-

Manuscript received January 30, 2023 revised May 8, 2023; approved for publication by Junbeom Hur, Division 3 Editor, May 15, 2023.

This research is financially supported by National Science and Technology Council of Taiwan (under grant No. 111-2221-E-992 -070 -MY2).

S.-H. Lee and J.-Z. Liu were with the Department of Intelligent Commerce, National Kaohsiung University of Science and Technology, 58 Shenzhong Rd., Yanchao Dist., Kaohsiung City 82444, Taiwan, R.O.C. email: {shlee, f109156110}@nkust.edu.tw.

S.-H. Lee is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2023.000023

work communications may become entry points for hacking. When data are sent to cloud services, intruders can launch attacks to change data integrity, resulting in significant negative effects on the security of edge device management [13]. Therefore, the design concept of the trusted platform module (TPM) will be eventually introduced into the plug-and-play device proposed in this paper to realize the protection mechanism of data transmission and authentication. In addition, remote device management systems now mainly provide local data processing services through the Internet of things' gateways or uploading data to the cloud for centralized processing [14].

Due to the heterogeneity of edge devices, the past Internet of things remote management mode has been unable to meet the needs of the current edge devices, and there is no suitable solution for the compatibility of hardware architecture and software. Add to this, common remote device management systems are mostly used for server and data center management due to construction costs and complex and necessary management. The common management method is using the hardware chip of the baseboard management controller (BMC), independent of the host system, and an intelligent platform management interface (IPMI), which is a software communication protocol [15]. An IPMI, independent of the central processing unit and operating system of the host system, provides the specifications for computer system self-management and the protocol definition for management and monitoring. According to Xu *et al.* [16], existing Internet of things devices do not support IPMIs, and the cost restriction of Internet of things devices makes it difficult to implement IPMI. Therefore, this paper proposes a plug-and-play management device to enable an edge-computing management service system using in- and out-of-band management modes.

With the increasing application requirements of edge computing, the stable operation and reliable management of edge devices have become significant. In terms of business value, many enterprises have many problems in management after edge device deployments, such as running statuses of edge devices, normal network connections, effective applications, and system breakdowns. In remote troubleshooting and restart, reducing the manpower cost of on-site maintenance and the extra cost arising from machine halting is a major problem. Therefore, this paper proposes a trusted edge device management system based on microservices and a plug-and-play device to solve the remote management of edge devices. In particular, these mainly include three parts, namely, plug-and-play hardware management device, trusted edge computing deployment management system, and cloud management platform. Plug-and-play hardware management devices are designed for effective monitoring, hardware encryption, and automatic return of the running states of edge devices and for restarting edge devices in cases of breakdowns. In addition to the design of the plug-and-play hardware management device, the trusted deployment management system and cloud management platform in the edge device are designed to divide the entire application into a set of independently running small services by combining microservice architectures and container virtualization technologies. It includes device real-time information, historical data query, remote control, and remote restart systems. This paper implements the containers-as-a-service (CaaS) to improve the extensibility and compatibility of the management system effectively. In order to ensure secure and reliable data transmission, different transmission channels and security authentication mechanisms are planned for different data types. The edge device management system and the plug-and-play hardware management device proposed in this paper contain not only in- and out-of-band management modes but also secure highly compatible deployment management service mechanisms. The contributions of this paper are as follows:

- A plug-and-play hardware management device, independent of the edge device system, is designed. The edge device system is connected to achieve information extraction, information transmission, data encryption, and control function.
- The design concept of the trusted platform module is introduced into the plug-and-play hardware management device to realize the data encryption protection mechanism and provide the authentication mechanism.
- The Docker and microservice architecture separate the complex management application into lightweight services. Each component performs tasks independently, which improves the compatibility and isolation of the framework.
- A cloud management platform is designed to assist administrators in remote monitoring and maintenance through secure two-way communications.
- In the management system proposed in this paper, the additional consumption of CPU, memory, disk, and power on the edge device is almost negligible. Without affecting the operation of the edge device, efficient remote edge device management is realized.
- The proposed system can achieve rapid deployment and troubleshooting, secure data transmission, reduction of labor costs, and commercial market demands in practical applications.

This paper is organized as follows. Section II discusses the related works and introduces background concepts and technologies. Section III describes the proposed system architecture and provides the details of plug-and-play hardware management device, microservices in edge node and management platform, and trusted data transmission mechanism. Section IV presents the results of the experiment and its verification. Finally, we conclude this paper.

II. Related Work

Before we introduce our proposed system, we present the related works and background concepts and technologies on device management system, microservice for Internet of things and trusted platform module.

A. Device Management Platform for Internet of Things

With the rapid growth in the number of Internet of things devices and their heterogeneity, manual device troubleshooting

is no longer suitable for the current environment, promoting the emergence of many Internet of things management platforms [4]. However, these management solutions need to consider not only the resource constraints and complex underlying communication mechanisms of low-power Internet of things devices but must also have the ability to cope with heterogeneous technologies. In order to ensure the connectivity, security, and interoperability of these connecting devices, secure remote access must be performed for network status monitoring, device hardware and software authentication, operating parameter configuration, fault detection, and maintenance [17].

For large-scale Internet of things deployment, a centralized Internet of things device management may lead to bottlenecks and massive delays. Mavromatis *et al.* [18] proposed a framework design of software-defined IoT management (SDIM) for multidomain wireless sensor networks edge and cloud computing. The proposed architecture uses software defined network (SDN) for device configuration, device control, and fault detection. In addition, multiaccess edge computing (MEC) is adopted to allocate resources between cloud and edge nodes, which can reduce data transmission delay and avoid bottlenecks in control or management. LwM2M [19] and NETCONF light IoT [21] device management protocols are compared by SDIM. The experimental results [18] show a significant efficiency improvement in device configuration time, control operation time, fault detection time, and energy consumption. Silva *et al.* [5] proposed a new Internet of things management platform with a user-friendly interface and extensibility called M4DN.IoT. M4DN.IoT based on Internet of things management issues, such as extensibility, interoperability, energy consumption efficiency, network topology configuration, QoS, fault tolerance, and security, creates a management platform and evaluates and validates its performance. The management issues proposed by M4DN.IoT are of a reference value. Internet of things management platforms and standard protocols are booming. LwM2M is the most widely accepted Internet of things management protocol in the world. As the Internet of things management technologies are becoming mature, there are still some challenges.

According to Silva *et al.* [21], there are many important issues on device management, including deployment, configuration, monitoring, communication, interoperability, and extensibility. In addition, device authentication, data transmission security, and maintenance are important challenges in the Internet of things management systems. These important issues on management are not only key success factors in the Internet of things but are also equally important in edge device management. Therefore, this paper proposes a trusted edge device management system based on microservices and a plug-and-play hardware management device. Without affecting the operation of the edge device, the device's running state, the functions of remote device control, and system restart are monitored remotely by in- and out-of-band management modes. In addition, in terms of data transmission security, the design concept of a trusted platform module is introduced to realize data encryption and authentication and ensure security and reliability. The Docker and microservice architecture are adopted to improve the flexibility and extensibility of device deployment and configuration.

B. Microservice for Device Management System

As virtualization is a basic function in the resource-constrained Internet of things environment, service providers face the challenge of deploying various services on Internet of things devices. In the device management system of a monolithic structure, all functional services are developed on a single memory pool. When system functions are updated or errors fixed, the entire system needs to be restarted, apart from ensuring the normal running of all other functional services. Moreover, the failure of some functional services may cause the entire system's breakdown. In the Internet of things environment that requires flexibility, extensibility, and maintainability, the management model of over-dependence between services no longer applies, and microservice architectures have become the key to meeting the demand [10], [22]. Microservice architectures use loosely coupled services that run independently in their processes and communicate through lightweight communication protocols. In simple terms, the application is built as a set of small services, with each service running in its process and deployed independently, and transfer information (e.g., user queries or data flows) from one service to another through HTTP resource API [11]. Villamizar *et al.* [23], taking Amazon, Netflix, and LinkedIn as examples, analyzed and tested the benefits and effectiveness of the microservice architecture model in development, testing, deployment, operation, and upgrade by using the Play Web framework. The experimental results [23] show that these companies can achieve effective extensibility, agility and reduce maintenance complexity. Microservices can be achieved by combining with virtualization technology. Ferreira *et al.* [24] proposed a middleware abstraction layer in a home energy management system (HEMS) that enables data access and system configuration between hardware and user applications using the concept of microservices. This paper [24] suggests that the microservices concept is a valuable option for applications that require edge computing methods. Kwon *et al.* [25] studied the design of microservices architecture based on the open-source EdgeX framework, which can manage configuration profile information of various IoT devices and support AI inference model operations, solving the problems of data processing speed delay and data security vulnerabilities caused by overloaded data processing centralized in the cloud. Thus, it can be inferred that microservices play a crucial role in the Internet of things. The proposed framework in this paper not only meets the security requirements of data transmission but also considers the in-and-out of band device management mechanism, achieving efficient remote edge device management.

As a lightweight virtualization technology [26], Docker can quickly create predictable environments that are isolated from other applications. Applications can remain consistent regardless of the deployment environments, making systems flexible and extensible [27]. Recent studies [28]–[33] show the growing trend of microservice and Docker service in

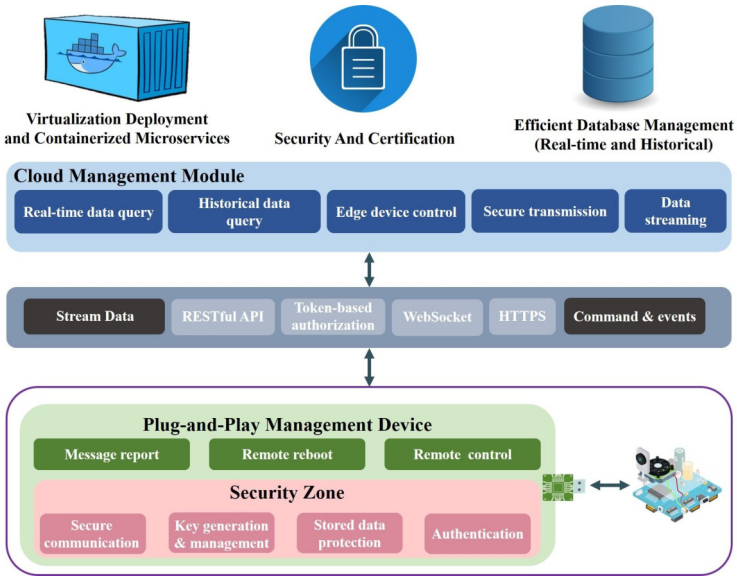

Fig. 1. Architecture of proposed trusted edge device management system.

IoT environments and explore the tradeoff between flexibility and performance overhead. According to the experimental results in the literature [30], [33], the overhead arising from container-based virtualization technology is negligible, such as resource consumption, service activation time, and energy efficiency. Furthermore, there are outstanding advantages in manageability and extensibility. According to the research literature [33], Docker Swarm [34] can simplify the deployment and management of multiple containers. Therefore, Docker and microservice architectures are used in this paper to realize device deployment and service management mechanisms. In the experiment section, the overhead costs of running the management services based on containerized microservice on edge devices are estimated with system benchmarking tools, which proves that the effectiveness of edge devices is not affected.

C. Trusted Platform Module

Edge computing still has some security concerns, such as data flow security and access authentication. Therefore, Ma *et al.* [35] pointed out that trusted computing and blockchain technology shall be introduced to improve the credibility and adaptability of edge computing security mechanisms. The concept of trusted computing is proposed by the trusted computing platform alliance (TCPA). The alliance is committed to developing secure and reliable hardware computing platforms. In addition, the alliance has expanded and become the trusted computing group (TCG). Subsequently, they proposed the hardware security architecture standards for the TPM [36]. Prakash *et al.* [37] explained that TPM not only provides additional security for authentication, encryption, and signature but also protects BIOS and operating systems of start-up sequences from tampering. There are software and hardware implementations in TPM. Aaraj *et al.* [38] explained that in embedded systems with limited resources, the cost of hardware TPM is unacceptable, and computing power and energy consumption need to be considered in the software implementation. Therefore, this paper introduces the design concept of a trusted platform module. In this paper, electrically-erasable programmable read-only memory (EEPROM) is embedded into plug-and-play management devices. It combines with advanced encryption standard (AES) protocol and token to realize data encryption mechanisms and authentication to ensure security and reliability.

III. Proposed System

The edge device management architecture proposed in this paper mainly includes plug-and-play hardware management devices, containerized microservice, data security transmission and authentication mechanism, edge device service import, database design, and cloud management platform and configuration, as shown in Fig. 1. Core technologies comprise the plug-and-play hardware management device supporting edge devices, data security transmission and authentication mechanism, virtualization deployment technology, database design based on memory and disk IO, and management platform design. Management services include real-time data and historical data processing, edge device control, security transmission, and data flow modules. Authentication and encryption mechanisms are provided in terms of secure transmission to ensure data security and accuracy.

The first goal of this paper is to establish a plug-and-play management device that supports edge devices of Linux ARM-based systems without affecting the deployed edge devices. The second goal is to import management services quickly based on microservice architectures into edge devices without affecting the programs already running on edge devices. The third goal is to introduce trusted mechanisms into the device management framework to prevent hackers from reading and tampering with data. The fourth goal is to

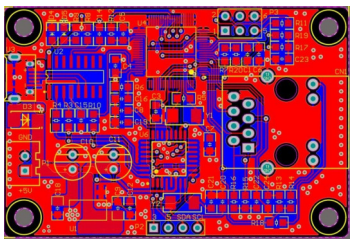

Fig. 2. PCB layout placement of pluggable hardware device.

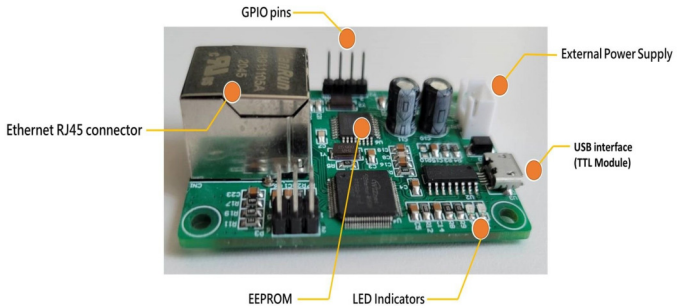

Fig. 3. Engineering sample of pluggable hardware device.

apply microservice-based architectures to cloud management platforms to help managers quickly view edge devices and remotely control devices.

In plug-and-play hardware, in-and-out of band data transmission can perform data encryption and authentication through a security zone in the hardware. This security zone is directly embedded in the hardware device and is similar to the TPM design mechanism. This is a low-cost TPM conceptual design method. In addition, since the hardware designed in this paper provides an in-and-out of band transmission mode, maintainers can quickly perform remote troubleshooting and monitoring. In terms of deployment and management, the mechanism of microservices are used in the cloud management platform and edge devices. The microservice architecture is considered to be one of the best solutions for building IoT systems. Through the independent deployment of processes and the decentralized structure of heterogeneous technologies, the scalability and stability of the management platform can be improved. The most important thing is that the microservice mechanism can be rapidly deployed and updated on edge devices. This feature has commercial value in practical applications. Therefore, a trusted edge device management system is realized by the concept of trust, microservice, and plug-and-play hardware proposed in this paper.

A. Pluggable Hardware Design for Edge Device Management

Baseboard management controllers (BMCs) are usually used to design the hardware of remote device management systems, and their hardware is mostly used in data centers or large server systems. Edge devices are expensive to build and have limited computing power. In addition, BMCs are independent of host machines and can be compatible or integrated with the hardware, firmware, and software of

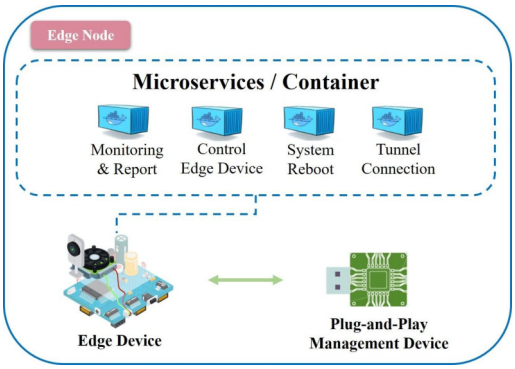

Fig. 4. Microservice in edge device.

BIOS or UBOOT. BMCs are not perfect now for supporting Linux ARM-based systems. Therefore, this paper designs a plug-and-play hardware management device that supports Linux ARM-based systems. In this paper, Atmega328P [39], characterized by its low power consumption, is used as the main microcontroller core. The USB bus is the hardware interface connecting with the edge device, used for data conversion and control through serial communications to become pluggable without affecting the edge device. An independent network interface is designed on the network communication hardware architecture to return the state through this network interface even if the edge device system network fails. In order to strengthen the security of the whole edge device, the design concept of TPM is introduced. EEPROM is embedded into plug-and-play management devices. It combines with the AES to directly provide encryption and authentication functions for hardware to increase credibility and reliability. External GPIO control pins and I2C interfaces are reserved to obtain peripheral devices, such as remote reset systems, restart systems, or fan control. This function can effectively solve the problem that the maintenance personnel only need to restart devices on-site because 70% of device failures can be resolved by restarting systems. In addition, to prevent outage of the plug-and-play management device caused by the failure of the edge device's power management circuits, the external power supply function in the design of the power supply is provided to ensure the stable operation of the plug-and-play management device. The hardware is 60 mm long, 40 mm wide, and 15 mm high. The layout placement of the printed circuit board is shown in Fig. 2. The actual hardware is shown in Fig. 3, which shows that the out-of-band management mechanism is achieved.

B. Microservice in Edge Node

In terms of edge device nodes, this paper takes the embedded Linux-ARM-based system, which can run the Docker virtualization technology as the main development environment. Management services are deployed on edge devices. Meanwhile, Docker and microservice architectures are used to divide service applications into lightweight and independently running services, including hardware monitoring and automatic return container service, device control container

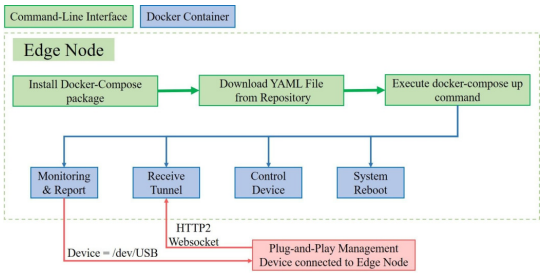

Fig. 5. The flow of starting edge device management service.

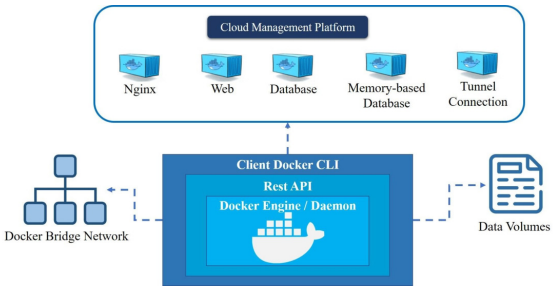

Fig. 6. Microservice in management platform.

service, secure reception tunnel container service, and restart container service shown in Fig. 4. In addition, the bridge network of each container, the privilege to connect interface devices, and container opening and restart mechanisms are defined by the Docker-compose multi-container deployment tool [40] in the YAML file format [41]. This paper introduces a simple edge device management service process, as shown in Fig. 5. First, the Docker-compose package is installed. Then, the multi-container deployment file of the edge device management service is downloaded. Finally, the command is executed to deploy the containerized management services rapidly. Container services of edge devices securely transmit monitored status data to plug-and-play management devices and deliver data to management platforms.

C. Container Service of Monitoring and Automatic Reprot

The process of monitoring and automatic report in container service is that the container extracts system operating information, including CPU temperature and use rate, GPU temperature and use rate, disk usage, memory usage, fan speed, system load, device temperature and network connection (including physical network connection), and peripheral control. It also stores data in the Javascript object notation (JSON) format. AES encryption is conducted after data are sent to the plug-and-play management device. Since the data length must be a multiple of 16, data are filled to be multiples of 16 in the container in advance.

D. Container Services of Secure Reception Tunnel, Device Control, and Restart

The process of secure reception tunnel, device control, and restart in container services is that when sending device control requests, the cloud management platform will carry out WebSocket handshaking protocol with the plug-and-play management device and transmit control instructions, including device control and system operation. After receiving messages, the plug-and-play device communicates with the secure reception tunnel container through serial ports and transmits instructions. The secure reception tunnel container determines event types and sends operating instructions to appropriate containers. There are mainly device control and system restart. In terms of device control, pulse-width modulation (PWM) is used in this paper to adjust the fan speed. In cases of overtemperature devices, the administrator can adjust the fan speed through the cloud management platform. The compulsory system restart of the plug-and-play management device is different. Moreover, the application scenario of the restart container service is used when the device runs normally. However, when the device is abnormal and unresponsive, the edge device is forcibly restarted by external GPIO pins from the plug-and-play management device.

E. Microservice in Management Platform

As shown in Fig. 6, the cloud management platform architecture is mainly composed of 5 Docker containers: Web server container service, web container service, secure transmission tunnel connection container service, and database management container service. The database management container service consists of MongoDB [42] and Redis databases [43]. In the face of real-time demand, the real-time response is achieved through the Redis database, which reads and writes fast. The data stored in memory for a certain period will be written back and updated to the MongoDB database, making the data management more flexible. In addition, the interdependent containers connected to the same bridge network allow containers to communicate with each other. On the other hand, containers not connected to the bridge network will be isolated from other containers. In the edge device and plug-and-play management device, data are transmitted to the management platform through encrypted SSH tunnels. The administrator can manage and maintain the edge device remotely with the browser. Its main functions are real-time device status view, historical data query, a remote device controlling and system restart, and in- and out-of-band management mechanism.

F. Real Time Data Monitoring and Historical Data Query

The real-time data monitoring is that the plug-and-play hardware management device automatically transmits the data to the cloud server through the HTTP2 communication protocol. In particular, the verification process after data entering the cloud management platform consists of the following steps. The cloud management platform obtains the identity tokens of all users from the Redis database and verifies whether they are valid. AES decryption keys are extracted according to the identity tokens. After decryption, the data will be temporarily saved to Redis and displayed on web pages.

The historical data of the edge device status are mainly stored in MongoDB. If the network of the edge device fails (in-band), its previous information can still be queried through the

cloud management platform. The Redis database is used for quick response in querying recent device data to make the data management more flexible, while MongoDB is used for previous data.

G. Remote Device Controlling

The remote device controlling is mainly achieved by the plug-and-play management device, which can be performed even if the edge device network fails. There are two service functions in this process: Remote fan control and system restart. The edge device normally running sends instructions to its container services through the plug-and-play management device. When the operating command from the outside world cannot be received because the edge device runs abnormally or the system crashes, the plug-and-play management device can be used to restart the system forcibly and implement the out-of-band management mechanism.

H. Trusted Data Transmission Mechanism

In this paper, a trusted mechanism is imported into the edge device management framework to ensure secure data transmission and authentication to prevent tampering or hacker infringement. The two types of transmitted and controlled objects include web browsers and edge devices. Meanwhile, HTTP2 [44] and WebSocket [45] are used as transport protocols, as shown in Fig. 7. In terms of the web browser, HTTPS is used for encryption to ensure secure communication between the browser and the cloud management platform. The browser is encrypted by SSL/TLS [46] and connected to the Nginx container by HTTP2. The Nginx container proxy sends requests to the web container, while the web container examines whether the accounts and passwords stored in the database are correct and grants access to users. Furthermore, the encrypted method from hardware side is based on the concept of TPM to strengthen the security of data transmission of the edge device. Concerning the plug-and-play hardware management device, encryption keys and tokens are generated in advance and stored in EEPROM. The encryption standard uses AES to provide encryption and authentication directly on the plug-and-play hardware management device. The data are authenticated while being sent to the server, which greatly reduces the vulnerabilities during data access. In addition, two encryption mechanisms are provided in this paper to encrypt the data using plug-and-play hardware management device, as shown in Fig. 8. In the first method, monitoring data are transmitted, through the USB interface, from the edge device to the plug-and-play management device. The plug-and-play management device gets the key from its EEPROM, encrypts the data, and obtains the tokens to authenticate the transmitted data. In the second method, the edge device sends encryption authentication requests, and the plug-and-play management device sends back the key and the tokens to the edge device. Meanwhile, the edge device is used for encryption and transmission. The two methods can achieve the in- and out-of-band management mechanisms, and appropriate data transmission methods can be implemented for platforms with different computing capabilities.

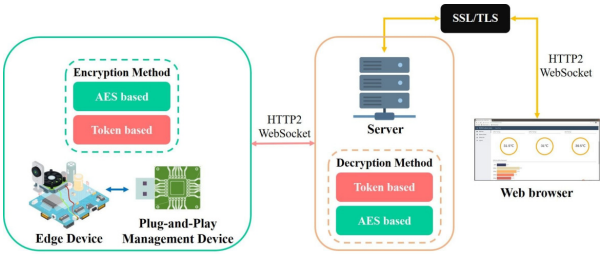

Fig. 7. The mechanism of trusted communication.

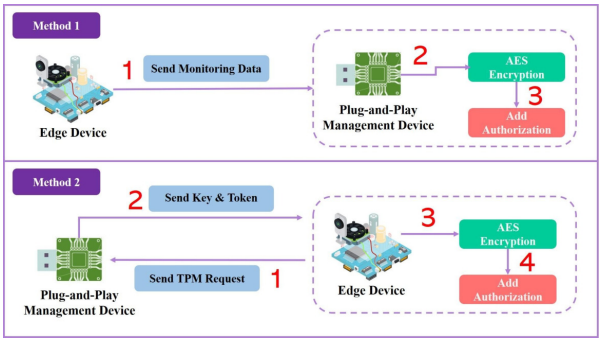

Fig. 8. Two mechanisms for different applications.

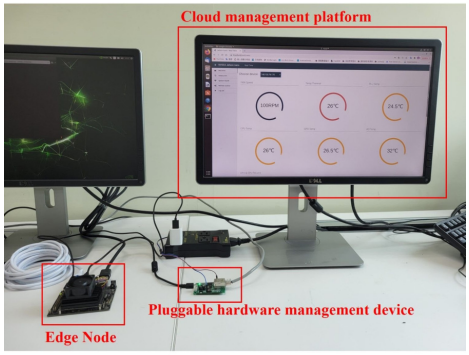

Fig. 9. Real system environment scenario.

IV. Experiment

A. System Environment Construction

In the experimental environment construction in this paper, NVIDIA Jetson Nano [47], an embedded Linux ARM-based system device, is adopted as the edge device. The plug-and-play hardware management device is developed in this study, as shown in Fig. 3. Table I shows the detailed specifications of the edge device used in this paper. Fig. 9 shows the entity construction.

B. Management Platform

1) Real time information: The administrator can choose edge devices based on the deployed IP address. In order to provide the real-time running status of the device, the advanced flask-socketIO package library is used for real-time push, which can realize two-way communications through

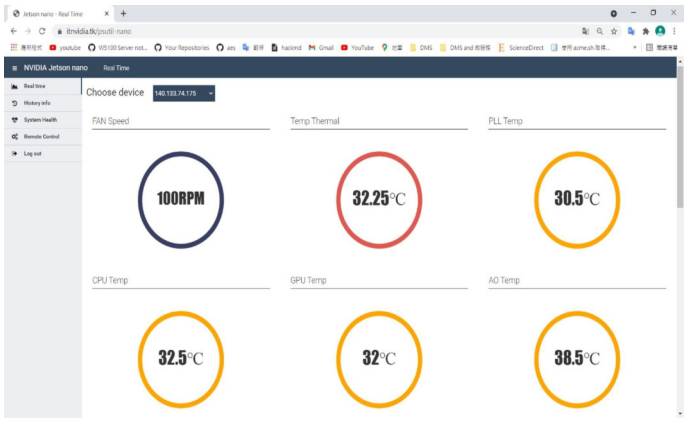

Fig. 10. Web user interface of real time data.

TABLE I
EDGE DEVICE ENVIRONMENT.

	NVIDIA Jetson Nano
CPU	Quad-core Arm® Cortex®-A57
GPU	128i NVIDIA Maxwell™ GPU
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
Storage	MicroSD 64 GB
OS	Ubuntu 18.04 LTS
Linux kernel	4.9.201-tegra
Power	5 W
Software package	• Docker 19.03.6 • Docker compose 1.29.2

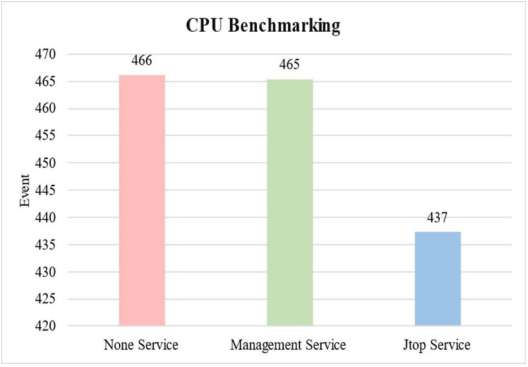

Fig. 11. CPU benchmark test in edge device.

WebSocket and Polling. The main information includes fan speed, CPU and GPU use rate, CPU and GPU temperature, phase-locked loops (PLL) and arctic oscillation (AO) temperatures, system load, memory and disk use rate, and network state. The actual management interface operation is shown in Fig. 10.

2) Historical data information: The administrator can view the historical data of the device through the management platform. The Redis database is used for quick response in querying recent device data, while the MongoDB database for previous data. Due to the batch writeback database mechanism adopted in this paper, data are first stored in the memory-based database and then written back to the disk-based database after a certain storage period to make the database management more flexible.

3) Remote device controlling: The remote control function can communicate with the plug-and-play management device via WebSocket and control the fan and restart the system. The administrator can control the fan speed by inputting PWM values (0–255) if the device temperature is too high. There are two methods to restart the system. The in-band method is used when the system runs normally. On the other hand, the out-of-band method is used, which can force the system to restart remotely, regardless of whether the edge device system crashes or stops unexpectedly.

C. Performance Evaluation

In recent years, whether containers cause additional performance overheads has been studied and explored in a large amount of literature. According to the literature, containers have little effect on CPU, memory I/O, and disk I/O. Due to a large number of complex calculations and tasks performed by the edge device, the additional performance overheads of management services must be reduced. Therefore, this paper examines whether there is any additional performance overhead arising from deploying the containerized management service on edge devices. In addition, there is Jtop, a monitor program in the NVIDIA Jetson Nano development software package. Hence, this paper compares the containerized management service with Jtop and tests its performance cost.

1) CPU benchmark test: This paper uses the Sysbench stress test tool [48] to test CPU performance. The experimental results are shown in Fig. 11. "Event" refers to the number of events that the CPU can process per second. Regardless of whether the containerized management service is enabled on the edge device, the CPU performance results are almost the same. However, the CPU performance is significantly reduced after the Jetson system monitor program is executed.

2) Memory bandwidth benchmark test: The mbw command [49], which determines the available memory bandwidth by replicating mass data, is used in this paper to test and

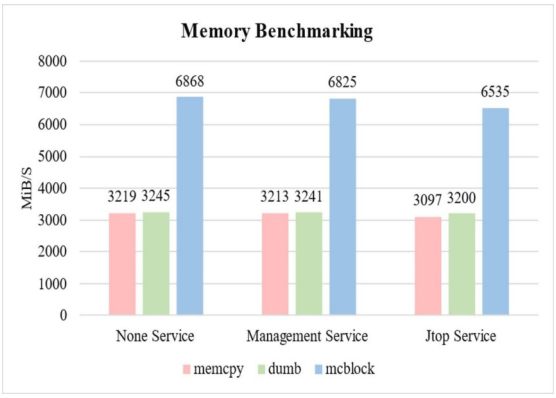

Fig. 12. Memory benchmark test in edge device.

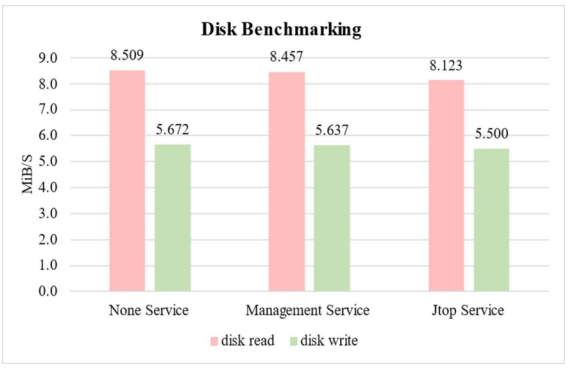

Fig. 13. Disk I/O benchmark test in edge device.

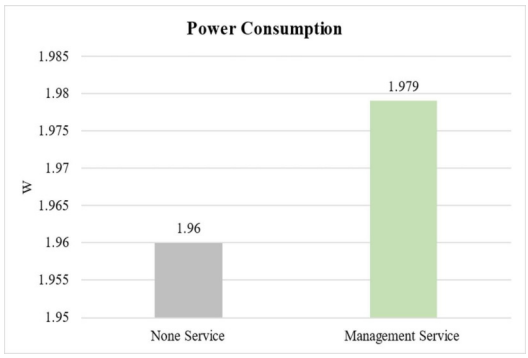

Fig. 14. Power consumption evaluation in edge device.

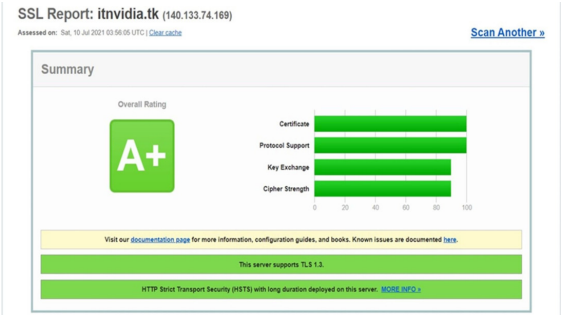

Fig. 15. SSL report by SSL labs.

evaluate the memory I/O performance by performing three different test projects (i.e., memcpy, dumb, and mcblock). The experimental results are shown in Fig. 12. The overhead arising from deploying the containerized management service of this paper is negligible.

3) Disk I/O benchmark test: This paper uses the Sysbench stress test tool to test the disk I/O performance. Before the disk I/O performance evaluation, the data file to be tested shall be first generated, and the generated data file must be at least larger than the memory. If the data in the archive can be fully stored in memory, the system will cache most data so that the results cannot effectively evaluate the intensive workload of disk I/O. The results are shown in Fig. 13. The performance overhead can be negligible regardless of whether the containerized management service is started.

4) Power consumption: In addition to performing its computing tasks, the edge device needs to complete management services. Therefore, this paper evaluates the additional power consumption generated by the management service on the edge device. The results are shown in Fig. 14. The power consumed to run the management service in this paper increases from 1.96 W to 1.979 W. However, only the 0.019 W increase means that the extra power consumption generated by this management service is almost negligible.

5) Secure sockets layer server test: The cloud management platform may be hacked or make important information stolen. Therefore, the cloud management platform SSL certificate and browser security are significant key factors. In this paper, Qualys SSL LABS [50] is used to test the security level of the cloud management platform SSL certificate. The cloud management platform implemented in this paper is scored A+, the highest security level, as shown in Fig. 15.

6) Security analysis: The edge computing device management framework proposed in this paper introduces trusted mechanisms for data encryption and identity authentication to ensure secure data transmission and authentication. Token-based identity authentication is adopted to prevent tampering and intrusion by hackers. HTTPS is used to encrypt communication between the browser and the cloud management platform to ensure secure communication. Therefore, the architecture proposed in this paper has the following claims in terms of security analysis:

1) The proposed mechanism can resist man-in-the-middle attacks: When transmitting messages, even if hackers intercept the packets using address resolution protocol (ARP) spoofing, the encrypted packets cannot be tampered with and can only be modified arbitrarily because the data between the transmission channels is encrypted using AES. Due to the identity authentication mechanism of the token, the packets that have been arbitrarily modified will be judged as illegal messages.
2) The proposed mechanism can resist impersonation attacks: The encryption key and token are pre-burned in the hardware device (EEPROM) and are a mechanism of the TPM. Hackers must know the relevant information of a legitimate device to impersonate the device, and in this

situation, hackers cannot easily impersonate a device.

V. Conclusion

This paper proposes a trusted edge device management system based on a microservice, mainly composed of a cloud management platform, plug-and-play hardware management device, and virtualization deployment technology to realize in- and out-of-band management modes. The design concept of the trusted platform module is introduced to the plug-and-play hardware management device to realize the data encryption mechanism and authentication. The system can effectively provide important information of the edge device and interoperability, including real-time device information, historical data query, remote control, and remote restart system. This paper designs a management service deployment mechanism, which integrates Docker and microservice architecture and provides rapidity, extensibility, and compatibility for deploying cloud management platforms and edge device management services. In addition, different transmission tunnel and security authentication mechanisms are provided among different edge devices, cloud management platforms, and browsers, including AES encryption and decryption, identity token authentication, encrypted SSH tunnel, and HTTPS. In terms of experimental data, the additional consumption of CPU, memory, disk, and power on the edge device is almost negligible in the management system proposed in this paper. In future work, we will consider the proposed platform's lightweight secure data transmission and authentication mechanisms in edge computing environments and verify the secure linkage of each module within the entire system scope.

References

[1] W. Yu *et al.*, "A survey on the edge computing for the Internet of things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[2] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the Internet of things: A case study," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 1275–1284, Apr. 2018.

[3] S. Sinche *et al.*, "Towards effective IoT management," in *Proc. IEEE SENSORS*, 2018.

[4] M. Asemani, F. Abdollahei, and F. Jabbari, "Understanding IoT platforms : Towards a comprehensive definition and main characteristic description," in *Proc. IEEE ICWR*, 2019.

[5] J. D. C. Silva, J. J. P. C. Rodrigues, K. Saleem, S. A. Kozlov, and R. A. L. Rabêlo, "M4DN.IoT-A networks and devices management platform for Internet of things," *IEEE Access*, vol. 7, pp. 53305–53313, 2019.

[6] J. de C. Silva, J. J. P. C. Rodrigues, J. Al-Muhtadi, R. A. L. Rabêlo, and V. Furtado, "Management platforms and protocols for Internet of things: A survey", *Sensors*, vol. 19, no. 3, p. 676, 2019.

[7] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts*, vol. 17, no. 4, pp. 2347–2376, 2015.

[8] H. Zhang *et al.*, "Research and application of industrial equipment management service system based on cloud-edge collaboration," in *Proc. CAC*, 2019.

[9] N. Dragoni *et al.,* "Microservices: Yesterday today and tomorrow" in *Present and Ulterior Software Engineering*, Cham, Switzerland:Springer, pp. 195–216, 2017.

[10] C. Santana, B. Alencar, and C. Prazeres, "Microservices: A mapping study for Internet of things solutions," in *Proc. IEEE NCA*, 2018.

[11] D. Lu, D. Huang, A. Walenstein, and D. Medhi, "A secure microservice framework for IoT," in *Proc. IEEE SOSE*, 2017.

[12] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, and E. K. Markakis, "A survey on the Internet of things (IoT) forensics: Challenges, approaches, and open issues," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1191–1221, 2020.

[13] A. H. Shehab and S. T. Faraj Al-Janabi, "Microsoft Azure IoT-based edge computing for smart homes," in *Proc. IEEE DASA*, 2020.

[14] F. Samie *et al.*, "Fast operation mode selection for highly efficient IoT edge devices," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 39, no. 3, pp. 572–584, Mar. 2020.

[15] Intel Corporation, "Intelligent platform management interface specification v2.0 rev. 1.1," Intel Corporation, Hewlett-Packard, NEC, Dell, Tech. Rep., 2013.

[16] M. Xu *et al.,* "Dominance as a new trusted computing primitive for the Internet of things," in *Proc. IEEE SP*, 2019.

[17] S. Sinche *et al.*, "A survey of IoT management protocols and frameworks," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1168–1190, 2020.

[18] A. Mavromatis *et al.,* "A software-defined IoT device management framework for edge and cloud computing," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 1718–1735, Mar. 2020.

[19] "Lightweight machine to machine technical specification. Approved version 1.0.1," Open Mobile Alliance, San Diego, CA, USA, Rep., Jul. 2017.

[20] J. S. V. Perelman, M. Ersue, and K. Watsen, "Network configuration protocol light (NETCONF Light)," Internet Eng. Task Force, Fremont, CA, USA, 2014.

[21] J. de C. Silva, J. J. P. C. Rodrigues, J. Al-Muhtadi, R. A. L. Rabêlo, and V. Furtado, "Management platforms and protocols for Internet of things: A survey," *Sensors*, vol. 19, no.3, p. 676, 2019.

[22] P. Di Francesco, P. Lago and I. Malavolta, "Migrating towards microservice architectures: An industrial survey," in *Proc. IEEE ICSA*, 2018.

[23] M. Villamizar *et al.*, "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," in *Proc. IEEE 10CCC*, 2015.

[24] L. C. B. C. Ferreira *et al.,* "Edge computing and microservices middleware for home energy management systems," *IEEE Access*, vol. 10, pp. 109663–109676, 2022.

[25] T. -G. Kwon and K. Ro, "A Study on edge computing-based microservices architecture supporting IoT device management and artificial intelligence inference," in *ICEIC*, 2023.

[26] X. Sun, Y. Liang, and H. Huang, "Design and implementation of Internet of things platform based on microservice and lightweight container," in *Proc. IEEE ITAIC*, 2020.

[27] X. Wan, X. Guan, T. Wang, G. Bai, and B.-Y. Choi, "Application deployment using microservice and docker containers: Framework and optimization", *J. Netw. Comput. Appl.*, vol. 119, pp. 97–109, Oct. 2018.

[28] W. Jin, R. Xu, T. You, Y. -G. Hong, and D. Kim, "Secure edge computing management based on independent microservices providers for gateway-centric IoT networks," *IEEE Access*, vol. 8, pp. 187975–187990, 2020.

[29] N. D. Nguyen, L. -A. Phan, D. -H. Park, S. Kim, and T. Kim, "ElasticFog: Elastic resource provisioning in container-based fog computing," *IEEE Access*, vol. 8, pp. 183879–183890, 2020.

[30] R. Morabito, "Virtualization on Internet of things edge devices with container technologies: A performance evaluation," *IEEE Access*, vol. 5, pp. 8835–8850, 2017.

[31] K. Kaur, T. Dhand, N. Kumar, and S. Zeadally, "Container-as-a-service at the ddge: Trade-off between energy efficiency and service availability at fog nano data centers," *IEEE Wireless Commun.*, vol. 24, no. 3, pp. 48–56, Jun. 2017.

[32] L. Deshpande and K. Liu, "Edge computing embedded platform with container migration," in *Proc. IEEE SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI*, 2017.

[33] R. Morabito, I. Farris, A. Iera, and T. Taleb, "Evaluating performance of containerized IoT services for clustered devices at the network edge," *IEEE Internet things J.*, vol. 4, no. 4, pp. 1019–1030, Aug. 2017.

[34] Docker Swarm. [Online]. Available: https://docs.docker.com/engine/swarm/. (accessed on 23 August 2021).

[35] B. Ma *et al.*, "Security of edge computing based on trusted computing," in *Proc. ISSSR*, 2020.

[36] TPM main part 1 design principles: Specification version 1. 2, Trusted Computing Group, 2011.

[37] V. Prakash, A. Williams, L. Garg, C. Savaglio, and S. Bawa, "Cloud and edge computing-based computer forensics: Challenges and open problems," *Electronics*, vol. 10, no. 11, p. 1229, May 2021.

[38] N. Aaraj, A. Raghunathan, S. Ravi, and N. K. Jha, "Energy and execution time analysis of a software-based trusted platform module," in *Proc. IEEE DATE*, 2007.

[39] ATmega328P Data Sheet. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf.
[40] Docker Compose. [Online]. Available: https://docs.docker.com/compose/.
[41] YAML-a human friendly data serialization standard. [Online]. Available: https://yaml.org/.
[42] MongoDB. [Online]. Available: https://www.mongodb.com/.
[43] Redis. [Online]. Available: https://redis.io/.
[44] FC7540—HTTP/2. [Online]. Available:https://tools.ietf.org/html/rfc7540.
[45] RFC6455—WebSocket. [Online]. Available: https://tools.ietf.org/html/rfc6455.
[46] Rescorla, E., "The transport layer security (TLS) Protocol Version 1.3", RFC 8446, Aug. 2018.
[47] NVIDIA Jetson Nano. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/.
[48] Sysbench tool. [Online]. Available:https://github.com/akopytov/sysbench.
[49] mbw-memory bandwidth benchmark tool. [Online]. Available: http://manpages.ubuntu.com/manpages/bionic/man1/mbw.1.html.
[50] Qualys SSL LABS. [Online]. Available: https://www.ssllabs.com/ssltest/.

Shih-Hsiung Lee is currently an Associate Professor of the Department of Intelligent Commerce at National Kaohsiung University of Science and Technology. He received the B.Sc. degree in Department of Applied Mathematics from National Chung Hsing University in 2007, the M.Sc. degree in Department of Computer Science and Information Engineering from National Cheng Kung University in 2009 and the Ph.D. degree in Institute of Computer and Communication Engineering from National Cheng Kung University in 2018. His research interests include Internet of things, intelligent computing, signal processing, deep learning, machine learning, and computer vision.

Jue-Zhi Liu is currently pursing the M.S. degree with the Department of Intelligent Commerce at National Kaohsiung University of Science and Technology, Taiwan. His research interests include Internet of things, ubiquitous learning, and network security.