# A Software-Defined Security Framework for Power IoT Cloud-Edge Environment

Rixuan Qiu[1], Yu Fu[1], Jian Le[2], Fuyong Zheng[1], Gan Qi[2], Chao Peng[1], and Yuancheng Li[3]
*(Corresponding author: Jian Le)*

State Grid Jiangxi Information & Telecommunication Company, Nanchang, China[1]
School of Electrical Engineering and Automation, Wuhan University, China[2]
Wuhan 430072, China
School of Control and Computer Engineering, North China Electric Power University, Beijing, China[3]
Email: ncepua@163.com

*(Received Apr. 20, 2022; Revised and Accepted Oct. 12, 2022; First Online Oct. 15, 2022)*

## Abstract

The application of cloud computing brings many security challenges to the power Internet of things, such as DoS attacks, location-based attacks, man-in-the-middle attacks, and sniffing. Aiming at the cloud-edge security of power Internet of things, we propose SD-PIoT, a framework based on the software-defined perimeter (SDP) and software-defined network (SDN). In the framework, SDP and SDN jointly enhance the security of the cloud edge in a complementary way. SDP protects the security of the cloud and the inner layer of SDN by rejecting all unauthorized edge traffic. SDN deploys SDP applications to the SDN application plane to realize SDP's flexible deployment and management. In addition, SDN improves the reliability of network communication by controlling the topology. The security, communication reliability, and performance are analyzed through simulation experiments. The results show that the scheme can effectively resist network attacks, improve communication reliability and performance, and improve the cloud-edge protection ability of power Internet of things.

*Keywords: Cloud-Edge Environment; Power IoT; Software-defined Network; Software-defined Perimeter*

## 1 Introduction

With the advent of the information age, tens of billions of power equipment are connected to the Internet, and a large amount of data on the network is transmitted between terminal devices [19]. In 2006, the cloud was proposed as a remote server for power companies, providing additional storage and data processing services. After the birth of cloud computing, paradigms such as edge computing and fog computing emerged, acting as the intermediary between edge devices and the cloud to improve the performance of the IoT. Gartner predicts that by 2025, more than 90% of enterprises will adopt cloud infrastructure and platform strategy [8], which greatly reduces the burden on power companies. Although cloud computing and other paradigms derived from it provide many advantages to the power IoT, some security challenges will inevitably arise in the cloud-edge environment. Reference [12]concludes by analyzing different types of cloud that although cloud computing itself has advantages, it also faces the danger of information leakage brought by insecure visitors. Reference [9]summarizes the security issues of cloud computing into four types of threats, including data level, privacy level, user level and provider level. It indicates that these threats come from attacks. Reference [10] analyzes the insecurity of traditional network boundaries to cloud computing servers and uses SDN technology to manage and control the network to alleviate cloud security problems. Reference [11] found that, due to many edge devices in the power IoT, the probability of attackers discovering terminal devices with defective network protocols increases, which will lead to these defective devices being used by hackers to launch DoS attacks on the cloud.

On the one hand, internal enterprise data is stored in the enterprise cloud, and the vast majority of enterprise employees need to have access and processing rights to these data. This expansion of access scope increases the probability of malicious "internal employees". On the other hand, edge/fog computing leverages several different techniques to build networks, introducing the possibility of multiple attacks, such as man-in-the-middle attacks, wireless jamming, and DoS attacks [18]. For the former, malicious "internal employees" and man-in-the-middle attacks threaten the confidentiality and integrity of information, and wireless jamming consumes bandwidth, spectrum, and computing resources at the edge. But in contrast, DoS attacks can occur in many ways, such as flooding, redirection, jamming, and spoofing, which are simple and effective. They can quickly lead to network confidentiality, integrity, and availability loss.

The power IoT cloud-edge environment requires a new framework to address the above security issues. Applying SDP technology in networks requiring remote interaction is a new and active area of research. References [14, 15, 19–21] all demonstrate the security advantages of SDP technology applied in environments such as the Internet of Things, which can resist various attacks, including DoS, and alleviate the security challenges caused by traditional boundary blurring. However, these studies did not address the flexible deployment and scalability of SDP applications, such as how to react when new devices are added to the network, nor did they consider the enterprise's network configuration. In addition, the literature [14] discussed that the challenges faced by SDP had not been solved, such as network interruption and configuration update. SDN separates the data plane and the control plane in a software-defined way, providing new ideas. SDN controls a global view of the network at the control plane in a software-defined way. The programmability of the SDN application layer brings more possibilities to improve the network in the power IoT.

Therefore, this paper proposes a software-defined power Internet of Things security protection framework SD-PIoT, which aims to help the power Internet of Things defend against cloud-side network attacks, simplify network management, and additionally improve the reliability of cloud-side communication under SDN's advantages. The main contributions of this paper are summarized as follows:

1) Aiming at the above security issues, we propose an SD-PIoT security framework, which uses SDP to protect the cloud-side security and introduces SDN to manage the network to alleviate the limitations of SDP.

2) The framework integrates the two by placing the SDP controller at the SDN application plane to provide the required security protection technology.

3) The solution has been tested for attack and network performance. The results show that the framework can protect the cloud-edge environment from attacks, improve communication reliability, and negligible network delay.

The rest of the paper is structured as follows. In the second section, some related works mentioned in the article are expounded. In Section 3, We described the SDP and SDN architectures. And a security framework SD-PIoT based on SDP and SDN is proposed and described. In Section 4, test implementation and result evaluation are presented. Finally, we summarize the proposed solution in Section 5.

## 2 Related work

Even applications in leading cloud service providers are not guaranteed to be 100% free from attacks [7, 16] . Due to a large number of terminal devices at the edge of the power IoT, the security of the cloud faces greater threats, such as privacy theft, resource fraud, and DoS attacks [13, 26]. Traditional network tools are inefficient for the massive power data collection, storage, processing, and forwarding necessary in power IoT networks. Inherent uncertainties such as packet loss and communication outages hidden at the edge of the network greatly degrade the quality of service (QoS). Therefore, the power Internet of Things needs a more advantageous security framework to improve the security of the cloud edge environment.

SDP has been explored in IoT security protection as a relatively novel security architecture in recent years. Reference [15] applies SDP to IoT applications in Message Queuing Telemetry Transport (MQTT) to provide an additional layer of security by exploiting the network stealth properties of SDP. In Ref. [19], SDP addresses the security challenges of edge computing in IoT, insuring the cloud from edge traffic. Reference [21] applies the logical boundaries of SDP to narrow the scope of network access and connectivity of network virtual functions (VNFs) to trusted identities. Reference [20] discusses the security issues cloud infrastructure-as-a-service faces and proposes an SDP-based solution that allows only authorized clients to access protected services. The existing literature has used the SDP controller's authentication, dynamic authorization and other programs to achieve corresponding security protection. But they have not specifically discussed the deployment and management of these programs in the network and have not considered the program expansion that enterprises need.

A lot of work has been done to study the network management advantages of SDN. Reference [2] proposed a new IoT security architecture based on SDN, which helps to improve the security and communication reliability of IoT. Reference [22] developed a powerful control and network management platform using SDN in the smart grid scenario. Reference [5] used the centralized control logic of SDN to alleviate the edge uncertainty of the Internet of Vehicles and used the edge offload delay as an indicator to verify the scheme's effectiveness. Rohit *et al.* [4] introduced SDN into IoT to improve resource-constrained IoT low-power devices' network performance and evaluated the scheme according to latency and packet loss rate. SD-MIoT [23] is an SDN solution applied to the mobile IoT environment, which uses SDN to alleviate the communication problems caused by mobile nodes, and uses the message passing success rate to measure the reliability of the scheme.

However, there are still some security problems to be solved in SDN. For example, the open programmability of SDN brings a trust crisis. The global control of the SDN controller makes the network more vulnerable to denial of service (DoS) attacks. In addition, the internal SDN faces the security risk of the controller being invaded. The separation of control plane and data plane brings security challenges to the outer and inner layers of Sdn. The outer security challenge means that the server or switch is vul-

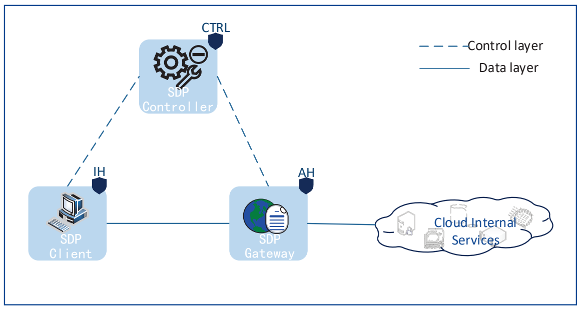


Figure 1: SDP client-gateway architecture

nerable to flood attacks, which may lead to the collapse of the entire security system. Inner security challenge refers to that intruders attack SDN controllers, generate false network data, and launch different attacks on the whole network. Reference [17] introduces the SDP framework to improve the outer layer security of SDN-based networks and discusses the interesting integration points between the two. It demonstrates the integrability of SDP and SDN, which lays the foundation for this paper, but the security issues within SDN are not addressed.

The difference between this paper and previous research is that SDP and SDN are cleverly combined to complement each other's shortcomings, bringing huge security advantages to the cloud-edge environment of the power Internet of things.

## 3 Method

### 3.1 Software-Defined Perimeter (SDP)

The Cloud Security Alliance (CSA) published the first SDP specification in 2014 [1], and SDP has attracted much attention in academia. SDP follows the idea of zero trust ”continuous verification, never trust”, only authenticated identities are allowed to access services, and unauthenticated identities remain in a state of ”network stealth” [3]. As shown in Figure 1, the SDP architecture consists of three main components, the SDP controller, the SDP initiating host (IH), and the SDP accepting host (AH). The SDP controller is located in the control plane and acts as the SDP brain, which determines any host's grant and access rights; the SDP IH is the host that initiates the service request; the SDP AH is the host that accepts the service request connection. After the IH is authenticated at the controller, it can connect and access the AH. Therefore, the SDP structure can effectively mitigate many network attacks that traditional boundary-based isolation protection methods cannot be solved, including server scanning, DoS and man-in-the-middle attacks [14].

Various deployment models for SDP include client-gateway, client-server, client-server-client, etc [1]. For the problem to be solved, this paper chooses the client-gateway model. In this model, the edge device acts as the client IH, the gateway acts as the AH, and the services and resources in the cloud are hidden behind the gateway to allow only connections from the gateway, which can make the cloud achieve ”network stealth” effect.

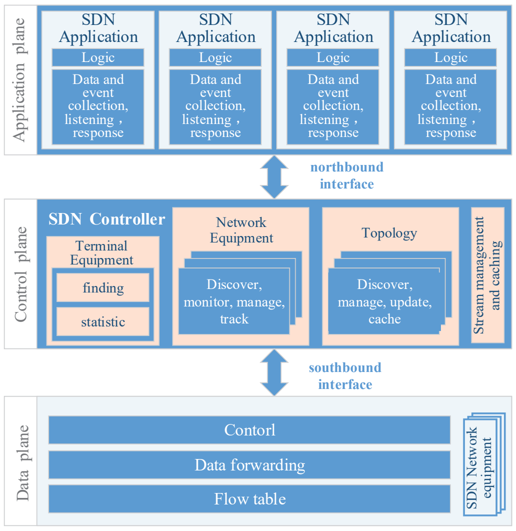


Figure 2: SDN architecture

### 3.2 Software-Defined Network (SDN)

With the development of network virtualization, SDN is proposed as an implementation of network virtualization [25]. As shown in Figure 2, the hierarchical structure of SDN consists of three planes, namely the application plane, control plane and data plane. The main components of these three planes are SDN application, SDN controller, and SDN network device, respectively.

In the control plane, the SDN controller has logical centralization and programmability. And it maintains the overall information of the global network, which is convenient for operators and researchers to implement different policy decisions through upper-layer programming. On the data plane, an SDN network device with a data forwarding function stores a flow table in which different forwarding rules are stored for different types of data packets. After receiving a data packet, the SDN device will search the entire flow table in descending order of priority and process the data packet according to the matching flow table entry rules [25]. The application plane contains various SDN applications, which run on the SDN controller. Events from the lower layers will trigger their different responses to the SDN controller, allowing operators to program and deploy new applications without caring about the details of the lower layers. Therefore, SDN can significantly simplify network control and enable flexible and efficient management of SDP-introduced power IoT networks.

## 3.3 Proposed Software-Defined Framework

This section will introduce the flexible combination and complementation of SDP and SDN, and the protection and control capabilities they provide for the cloud-edge environment of the power IoT. Figure 3 depicts the overall design of our proposed framework SD-PIoT, which extends the SDN architecture. For the cloud-side security of the power IoT, we add an edge device plane and a cloud center plane to the original SDN architecture.

### 3.3.1 Framework description

The framework contains five main components: SDP client (IH), SDP controller (CTRL), SDP gateway (AH), SDN controller and SDN switch.

1) SDP client (IH): The edge device in the network that needs to access the cloud is used as the IH in the SDP, and the SDP client software is installed on it. After the edge device goes online, it can communicate with other components.

2) SDP controller (CTRL): In this solution, the SDP controller runs on the SDN controller as an SDN application. It monitors the SDN controller's status, responds to the events, and determines whether every IH can access the AH. For clarity, the SDP controller is denoted by CTRL after this.

3) SDP gateway (AH): The gateway, as an AH, obeys the arrangement of the SDP controller, initially configures the firewall rule as "deny all", provides access connections to cloud services for authorized identities, and monitors and records the entire connection process.

4) SDN controller: The SDN controller holds a global view of the entire network, controls the underlying network consisting of switches, and provides an abstraction of the underlying network resources to SDN applications such as CTRL to wake them up and make them responsive.

5) SDN switch: A SDN device at the network data plane. The SDN switch forwards the data (in packet units) of other components. A flow table is stored in the switch, and each item of the flow table consists of matching fields and actions. The switch decides the removal and retention of incoming packets according to the flow table and the instructions of the SDN controller.

Application Plane: There are SDN applications (such as Graphical User Interface, routing procedures) required by enterprises on the application plane. Different SDN applications implement corresponding network functions. SDN applications can execute the assigned functions by manipulating the SDN controller when an event from the control layer or an external input drive is detected.

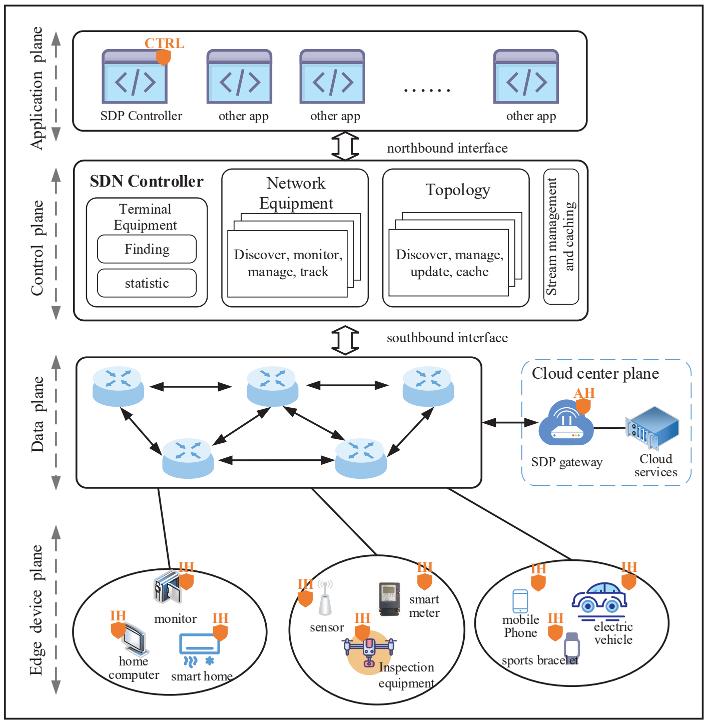


Figure 3: SD-PIoT model

Users/developers can manage and control the network without touching the underlying network by programming different SDN applications. In addition to basic SDN applications, power companies can design different SDN applications according to their needs to realize network programmability. It is worth mentioning that CTRL can be a program [6], so this paper places CTRL as an SDN application in the application layer. After receiving the access request from the IH, the CTRL verifies whether the device completes the SPA authorization. If it passes the verification, it instructs the SDN controller to forward the request to the AH. Otherwise, instructs the SDN controller to discard the packet. The authentication of CTRL is shown in Algorithm 1.

Control plane: The control plane has a logically centralized SDN controller that maintains a global view of the entire network, implements various policy decisions, and facilitates efficient and reliable communication services. In addition to the SDN controller, the control plane also includes northbound interfaces and southbound interfaces, which are respectively open to the application plane and the data plane. As the brain of SDN, the SDN controller realizes centralized logic control, provides the application plane with an abstract model of the underlying network, including states and events through the northbound interface, and wakes up SDN applications to perform corresponding functions. In addition to the above, the control plane embodies the application plane's request to configure, manage, and control the data plane according to its logic.

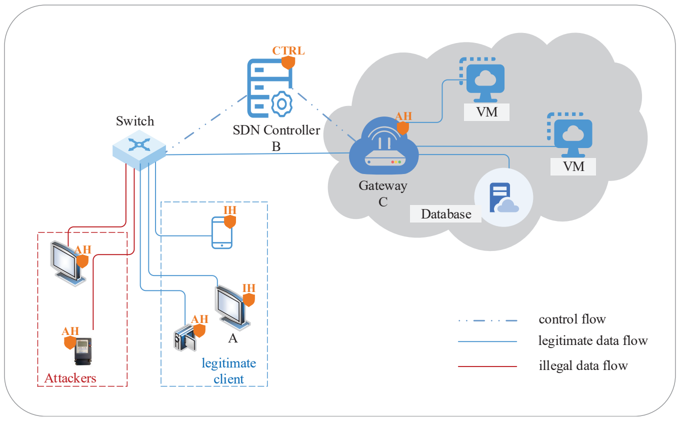


Figure 4: Case of the scheme

Data plane: As shown in Figure 3, this plane consists of SDN network devices (switches are used in this paper) with data forwarding and processing functions. All components in the framework are connected through switches. All switches in the network work under the control of the SDN controller, and the data layer communicates with the control layer through the southbound interface of the control layer. A forwarding table resides in each switch. When the switch receives a packet, it will search for the forwarding table in priority order. If it finds a flow entry that matches the packet, it will execute the configured action. If no match is found, the packet is discarded or forwarded to the controller (the latter is used in this paper).

Edge device plane: The network edge devices of the Power IoT all reside in this layer, such as sensors, detectors, smart meters, mobile devices, etc. Edge devices in this layer are mostly heterogeneous, come from different networks and use different communication technologies such as cellular, WiFi, and RFID. These devices are configured with IH and have the function of applying for cloud access. When edge devices want to access cloud services, they need to send SPA packets to CTRL for authentication. An unauthenticated IH will not be granted access. Algorithm 1 is an edge device authentication.

Cloud center plane: Many services and resources need to be protected in the cloud. These service resources are hidden behind the gateway. For IHs without CTRL authentication, the services in the cloud are inaccessible. Therefore, the cloud has remained "Network stealth" to potential attackers.

### 3.3.2 Scheme Process

Figure 4 shows the specific use case of the scheme.

1) Initialization: A default flow f0 with priority 0 is set on all switches. The task of flow f0 is to forward all traffic from the edge to the SDN controller. Traffic is forwarded between switches in two paths, optimal and redundant, as determined by the SDN controller.

**Algorithm 1** Edge Device Authentication

```
    The network has been deployed;
    The basic format of the flow entry is ($priority, source, action, duration$);
    Add default flow $f_0(0, all-edge-devices, forwarded-to-SDN-controller, all-time)$ to SDN switch;
    AH is online and authorized by CTRL;
    IH (device id is $u_0$) goes online and sends an authentication request $Req$ to CTRL;
1:  if $Req$ is a valid SPA package then
2:      CTRL verifies $u_0$ certificate and key;
3:      if certificate and key are verified then
4:          CTRL determines the accessible cloud service list $l_0$ of $u_0$;
5:          CTRL instructs the SDN switch to add flow $f_1(10, u_0, forwarded-to-AH, k)$
              and the k value is adjusted by the power company according to the situation;
6:          CTRL makes $l_0$ as a message and sends it to the IH, informing it of the list of
              cloud services it can access;
7:          CTRL sends $l_0$ to AH, informing it to update firewall rules and allow IH to
              access services in the list;
8:          AH updates firewall rules after receiving $l_0$, and allows IH to communicate with
              services in $l_0$ within time $k$;
9:      else
10:         CTRL commands the SDN controller to drop $Req$;
11:         CTRL instructs the SDN switch to add a flow $f_2(11, u_0, drop, k/2)$ to punish $u_0$
              for not being able to authenticate within $k/2$ time;
12:     end if
13: else
14:     CTRL commands the SDN controller to drop $Req$;
15:     CTRL instructs the SDN switch to add a flow $f_2(11, u_0, drop, k/2)$ to punish $u_0$ for
        not being able to authenticate within $k/2$ time;
16: end if
```

The firewall configuration of the SDP gateway is initialized to "deny all".

2) Authentication: The newly online IH(A) sends an authentication request to CTRL. Since the switch sets the default forwarding flow f0, the request packet finally reaches the SDN controller (B). CTRL on B detects the authentication request packet, checks A's credentials and verifies its identity and determines its access level:

a. If the verification is passed, CTRL instructs B to send a message to the SDP gateway AH (C), and instructs C to add a new firewall rule h1 to allow communication between A and C. At the same time, a new flow entry f1 with a priority greater than 0 is added to the switch. Within the specified time T, the service request packet from A is forwarded to C by default. After T time, the flow entry f1 and the new firewall rule h1 are automatically deleted.

b. If verification fails, CTRL instructs the SDN controller to discard the packet.

3) Request for service: Authorized A requests C to access services in the cloud. The switch forwards the request to C according to the priority without going through B, and C checks whether the request is a SPA packet:

   a. If the data packet is correct, C allows A to access some cloud services according to its own firewall rule h1.

   b. If the data packet is incorrect, C reports the relevant information to B and deletes the firewall rule h1 related to A. After the CTRL on B detects the message, it immediately instructs the switch to delete the related flow entry f1, and A will need to re-authenticate the CTRL.

During the A and C connection process, C monitors and records A's behavior through the tracking mechanism.

#### 3.3.3 Network Management and Control

In the cloud-edge interaction process of the power IoT network, the SDN controller is located in the control plane to monitor the global network and regularly collect information to update the network. The SDN controller can discover the newly added edge terminal equipment and switch network equipment for the first time. After discovering a new edge device, CTRL issues a certificate and user key for the device, and the SDN controller records the device's communication protocol. After discovering network devices such as switches, the SDN controller updates network topology, sets default flows, and maintains and tracks network status information. When the traffic from the new edge device enters the network, the controller determines the transmission technology and routing path according to the statistical information and adds a flow table entry suitable for the device in the switch to facilitate subsequent traffic forwarding. The control plane monitors the network in real-time and updates it regularly, which greatly improves the scalability of the network.

In order to prevent communication interruption, packet loss, and ensure real-time communication between the cloud and the edge, this solution takes advantage of the SDN controller to monitor the network topology in real-time, and forwards legitimate traffic through the optimal path and another redundant path to improve communication reliability. At the same time, SD-PIoT opens up the application plane's programmability to power enterprises. Enterprises can create their new applications according to their internal needs without caring about the differences between the underlying devices. Applications can modify the underlying forwarding rules through the northbound interface to achieve rapid network configuration and deployment, simplifying network management. For example, in zero trust, security protection includes several basic processes: user identity management, dynamic authorization control, and access control [24]. Suppose the device fails to authenticate many times during the service request. In that case, the enterprise can program an application to evaluate the trust degree of such identities and then authorize its access level according to the trust degree. The upper-layer pluggable application of the SDN controller can be programmed according to the internal requirements of the enterprise and the records of CTRL and AH in the authentication and service request phases, which realizes the application's scalability.

In addition, all edge traffic reaching the SDN controller in the network will first be verified by CTRL to decide whether to retain the traffic. If the verification fails, the corresponding packet will be directly discarded. This effectively prevents intruders from damaging the SDN controller and ensures the inner layer security of the SDN.

The proposed framework, SD-IoT, protects cloud-side security and achieves the flexibility, scalability, and programmability of the network. It ensures reliable communication between heterogeneous devices and the inner layer security of SDN, and simplifies network management.

## 4 Experiment and Analysis

We evaluate the proposed framework from security, communication, reliability, and performance aspects. The main goal of this framework is to improve cloud-based IoT security and communication reliability without compromising network performance.

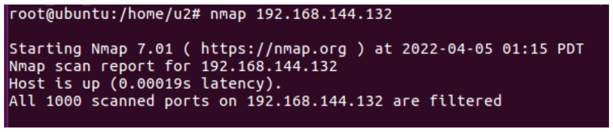
### 4.1 Testbed Environment

The testbed consists of SDP and SDN. The SDP part uses Waverley Labs' Open SDP project, and the SDN part uses public OpenFlow components. The simulation experiment uses six Linux Ubuntu16.04 virtual machines representing two edge devices, an SDN controller running the CTRL application, an SDP gateway, a cloud server, and an SDN switch. Table 1 presents the details of the components of the testbed.

First, the components of the SDN network are deployed and run. The SDP gateway and the CTRL module running on the SDN controller begin to work, and finally, the cloud server and edge devices are connected to the network.

Table 1: Testbed configuration

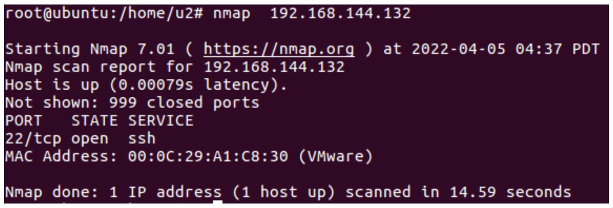
| machine | softwaree | detailed description |
|---|---|---|
| Edge device A +IH | -Linux Ubuntu16.04<br>-Fwknop module (IH)offered by Waverley Labs | Simulate the edge device in the wired network environment and run the SDP client IH model; directly connect to the SDN switch. |
| Edge device B +IH | -Linux Ubuntu16.04<br>-Fwknop module (IH) offered by Waverley Labs | Simulate the edge device in the wired network environment and run the SDP client IH model; directly connect to the SDN switch. |
| SDN controller +CTRL | -Linux Ubuntu16.04<br>-SDP controller module offered by Waverley Labs<br>-OpenDaylight Boron | Simulate an SDN controller running the CTRL application to control the global view of the network; all edge devices must authenticate with CTRL before they can access cloud services. |
| SDP gateway +AH | -Linux Ubuntu16.04<br>-Fwknop module (AH) offered by Waverley Labs | Simulate the SDP gateway, run the SDP client AH model; configure the firewall rule to "deny all". |
| Cloud server | -Linux Ubuntu16.04 | Simulate a cloud service hidden behind a gateway, and can perform basic SSH connection services. |
| SDN switch | -Linux Ubuntu16.04<br>-Open vSwitch 2.6 | Simulate multiple SDN switches with data forwarding and processing capabilities, and establish topological relationships. |



```
root@ubuntu:/home/u2# nmap 192.168.144.132

Starting Nmap 7.01 ( https://nmap.org ) at 2022-04-05 01:15 PDT
Nmap scan report for 192.168.144.132
Host is up (0.00019s latency).
All 1000 scanned ports on 192.168.144.132 are filtered
```

(a) With SDP



```
root@ubuntu:/home/u2# nmap  192.168.144.132

Starting Nmap 7.01 ( https://nmap.org ) at 2022-04-05 04:37 PDT
Nmap scan report for 192.168.144.132
Host is up (0.00079s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 00:0C:29:A1:C8:30 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 14.59 seconds
```

(b) Without SDP

Figure 5: Port scanning attack

## 4.2 Security Test

In the security test, we launched two types of attacks: 1) using the free nmap utility to launch a port scanning attack on the cloud server; 2) using the hping3 tool to launch a DoS attack on the cloud server, and using Wireshark to capture the traffic.

CTRL in SDP distributes the keys and certificates of edge devices. To simplify the operation during simulation, we randomly distribute keys and certificates for trusted edge devices manually. So for the port scanning part of the experiment, device A as an attacker, is not distributed keys and certificates. Device A performs port scanning attacks on cloud servers with and without SDP protection, respectively, and the results are shown in Figure 5(a)(b). The results show that when attacker A launches the attack without SDP protection, nmap scan shows that the TCP connection of the server is open. But under SDP protection, the device will not be granted access because the attacker is not CTRL authenticated. The result is as expected, and all ports are filtered.

In addition, from the 20s of capture, an SYN flood attack was launched against the network-protected service for 40 seconds. Wireshark was used to trace the captured traffic for the 80s, and the results are shown in Figure 6. The red curve is recorded as the traffic of attack packets, and the green curve represents the protected service traffic capture. CTRL instructs the SDN controller and the SDN switch to drop attack packets during the attack, and the server is not affected.

The results confirm that our proposed solution has "network stealth" properties, which can protect the server from unknown users.

## 4.3 Communication Reliability Test

For the cloud-edge environment of the power Internet of Things, ensuring real-time data transmission between the cloud and edge devices is the key for enterprises to efficiently provide services to users, such as remote control of smart homes and drone inspections of power equipment. Therefore, this paper uses the message delivery success rate (MDR) indicator to evaluate the communication reliability,

$$MDR = \frac{\sum R}{\sum S}. \quad (1)$$

where R is the number of messages received, and S is the number of messages sent. MDR is designed to calculate

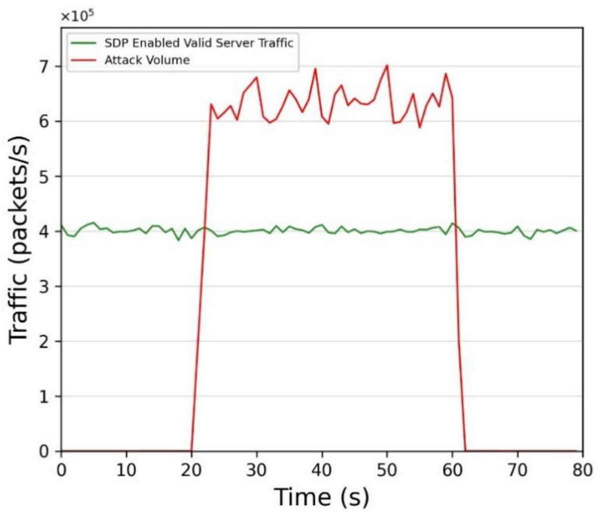


Figure 6: DoS attack capture using Wireshark.

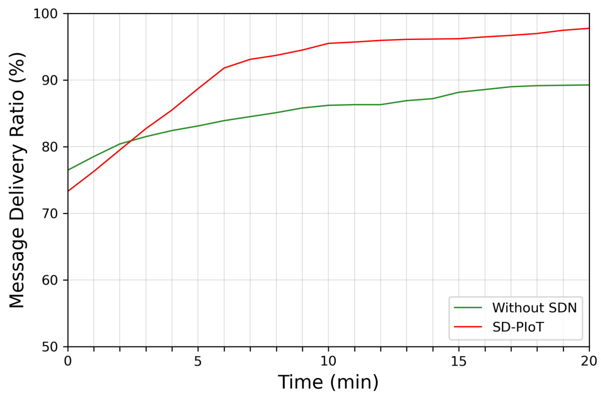


Figure 7: Comparison of MDRs

the ratio of the number of messages sent to the received in the network over some time. The closer the MDR value is to 1, the more reliable the communication.

In the simulation, trusted device A and device B send packets of 128 bits to the cloud server at regular intervals using wired and wireless connections. The cloud server responded immediately after receiving the data and returned the packet. And this process lasted for 20 minutes. To test the flexibility of this solution, we set the message sending interval of device A to 0.025s and device B to 0.05s, resulting in a relatively congested link between device A and the cloud server. Comparing the scheme with the SDN module and the scheme without the SDN module in this paper, the message delivery success rate is shown in Figure 7. It can be seen from Figure 7 that at the end of the simulation (20 minutes), the MDR of the proposed SD-PIoT framework achieves a success rate of 97.76%, while the performance of the scheme removing the SDN module drops by 8.5%. This is because the network congestion will increase the probability of packet loss. In this solution, the SDN controller is introduced to control the network topology accurately. The packet is forwarded in the optimal path and another redundant path under the condition of trusted identity. This can effectively avoid packet loss and improve MDR. The experimental results strongly confirm the effectiveness of this scheme for the network communication reliability of the massive terminal network in the power IoT.

## 4.4 Performance Test

To more objectively evaluate the performance of SD-PIoT, we compare the theoretical latency with the actual latency in the experiments. Ideally, ignoring processing and queuing delays, the total delay can be expressed as:

$$TD = \frac{M}{V} + \frac{L}{R}. \qquad (2)$$

where M is the link length, V is the signal propagation speed, L is the packet length, R is the link bandwidth, and TD is designed to calculate the sum of transmission delay and propagation delay.

We monitored delays ten times using the Wireshark tool, analyzed packet exchanges between components, and averaged them to determine CTRL, switch, and gateway overhead. The statistical results are shown in Table 2. The latency of this scheme is higher than that of SDP-only and SDN-only. This is because edge devices must pass CTRL authentication and wait for the SDP gateway to update the firewall before accessing the server. In addition, during this period, the SDN controller will issue commands to instruct the SDN switch to add new flow entries. The SDN switch will search for matching flow entries when receiving packets from edge devices, inevitably increasing the delay. But from the results in the table, compared with the advantages of security, communication and network management brought by this framework, these time overheads are completely tolerable. In addition, the calculated theoretical value is not much different from the actual measured value, which verifies the correctness of our test.

## 4.5 Comparison of Security Features

To highlight our work's contribution, we summarize our project's features in this section. We have compared our scheme with other state-of-the-art schemes in terms of these characteristics. The results are shown in Table 3. This scheme improves the cloud-side security and communication reliability of the power IoT, ensures SDN security, and simplifies network management, which other schemes cannot achieve.

Table 2: Latency Analysis

| Delay | Theoretical (secs) | Actual Measured (Sec) |
|---|---|---|
| End-to-End latency with SD-IoT | 0.56184562 | 0.58145238 |
| End-to-End latency without SDP | 0.44853264 | 0.51365214 |
| End-to-End latency without SDN | 0.50124563 | 0.55856241 |
| CTRL overhead | - | 0.04512114 |
| Switch overhead | - | 0.02132412 |
| Gateway overhead | - | 0.04823564 |

Table 3: Features comparison

| **Features** | | **SD-PIoT** | **SDP-SDN [17]** | **Solution [20]** | **MEC-SDP [19]** |
|---|---|---|---|---|---|
| Service stealth | | Y | N | Y | Y |
| communication reliability | | Y | Y | Y | N |
| Network | Scalability | Y | Y | N | N |
| | Programmability | Y | Y | N | N |
| SDN inner layer security | | Y | N | - | - |

## 5 Conclusions

The in-depth integration of power IoT and cloud computing has led to various security challenges in the cloud-side environment. This paper proposes the SD-PIoT composition framework and illustrates its effectiveness in addressing these challenges. First, we discussed the security issues of the power IoT cloud-side environment and the difficulty of network management. We then describe how SDN and SDP can be cleverly combined to alleviate these challenges. Furthermore, we implement and test the proposed solution, demonstrating the superiority of the framework in security and network management. Tests and evaluations prove that the SD-PIoT framework can effectively resist DoS attacks and achieve ”network stealth” of services. And the framework leverages the centralized programmability and scalability provided by SDN technology to control the network, allowing power IoT devices from heterogeneous networks to communicate securely and reliably while maintaining performance. In comparison with other schemes, the contribution of this paper can be reflected intuitively. Next, we will explore the implementation of this scheme in a more realistic power IoT scenario.

## Acknowledgments

This work was supported in part by the State Grid Jiangxi Information & Telecommunication Company Project ”Research on de-boundary security protection technology based on zero trust framework ” under Grant 52183520007V.

## References

[1] B. Bilger, “Software defined perimeter working group sdp specification 1.0,” *Cloud Security Alliance, Tech. Rep*, vol. 4, 2014.

[2] M. Conti, P. Kaliyar, and C. Lal, “Censor: Cloud-enabled secure iot architecture over sdn paradigm,” *Concurrency and Computation: Practice and Experience*, vol. 31, no. 8, p. e4978, 2019.

[3] CSA. “Software defined perimeter,”. tech. rep., Cloud Security Alliance, http://downloads.cloudsecurityalliance.org/initiatives/sdp/SoftwareDefinedPerimeter.pdf, 2013.

[4] R. K. Das, N. Ahmed, F. H. Pohrmen, A. K. Maji, and G. Saha, “6le-sdn: an edge-based software-defined network for internet of things,” *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7725–7733, 2020.

[5] X. Xu et al., “Secure service offloading for internet of vehicles in sdn-enabled mobile edge computing,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3720–3729, 2020.

[6] J. Garbis and J. Koilpillai, “Software defined perimeter architecture guide,” *Cloud Security Alliance*, 2019.

[7] S. S. Jajoo and K. N. Kumar, “A review on deep-learning based network intrusion detection systems,” *International Journal of Electronics and Information Engineering*, vol. 13, no. 4, pp. 170–179, 2021.

[8] Y. Jing. “Forecast of cloud computing market trend in 2022,”. tech. rep., Tencent, http://cloud.tencent.com/developer/article/1919520?from=15425, Dec. 2021.

[9] R. Kaur and J. Kaur, "Cloud computing security issues and its solution: A review," in *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 1198–1200. IEEE, 2015.

[10] H. Liang, H. Liu, F. Dang, L. Yan, and D. Li, "Information system security protection based on sdn technology in cloud computing environment," in *2021 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA)*, pp. 432–435. IEEE, 2021.

[11] J. Ma, "Research on ddos in cloud computing environment (in chinese)," *China Computer & Communication*, pp. 149–151, 2017.

[12] A. Markandey, P. Dhamdhere, and Y. Gajmal, "Data access security in cloud computing: A review," in *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, pp. 633–636. IEEE, 2018.

[13] S. Meena, E. Daniel, and N. A. Vasanthi, "Survey on various data integrity attacks in cloud environment and the solutions," in *2013 International Conference on Circuits, Power and Computing Technologies (ICCPCT)*, pp. 1076–1081. IEEE, 2013.

[14] A. Moubayed, A. Refaey, and A. Shami, "Software-defined perimeter (sdp): State of the art secure solution for modern networks," *IEEE network*, vol. 33, no. 5, pp. 226–233, 2019.

[15] A. Refaey, A. Sallam, and A. Shami, "On iot applications: a proposed sdp framework for mqtt," *Electronics Letters*, vol. 55, no. 22, pp. 1201–1203, 2019.

[16] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM conference on Computer and communications security*, pp. 199–212, 2009.

[17] A. Sallam, A. Refaey, and A. Shami, "On the security of sdn: A completed secure and scalable framework using the software-defined perimeter," *IEEE access*, vol. 7, pp. 146577–146587, 2019.

[18] S. N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison, "The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2586–2595, 2017.

[19] J. Singh, Y. Bello, A. Refaey, A. Erbad, and A. Mohamed, "Hierarchical security paradigm for iot multiaccess edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5794–5805, 2020.

[20] J. Singh, A. Refaey, and J. Koilpillai, "Adoption of the software-defined perimeter (sdp) architecture for infrastructure as a service," *Canadian Journal of Electrical and Computer Engineering*, vol. 43, no. 4, pp. 357–363, 2020.

[21] J. Singh, A. Refaey, and A. Shami, "Multilevel security framework for nfv based on software defined perimeter," *IEEE Network*, vol. 34, no. 5, pp. 114–119, 2020.

[22] A. Sydney, *The evaluation of software defined networking for communication and control of cyber physical systems*. Kansas State University, 2013.

[23] T. Theodorou and L. Mamatas, "Sd-miot: A software-defined networking solution for mobile internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4604–4617, 2020.

[24] Z. Xiaojian, C. Liandong, F. Jie, W. Xiangqun, and W. Qi, "Power iot security protection architecture based on zero trust framework," in *2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP)*, pp. 166–170. IEEE, 2021.

[25] C. H. Zhang, Y. Cui, H. Y. Tang, and J. P. Wu, "State-of-the-art survey on software-defined networking (sdn)," *Journal of software*, vol. 26, no. 1, pp. 62–81, 2015.

[26] B. Zhao, "Research on cloud computing security risk and security technology(in chinese)," *Computer Knowledge and Technology*, vol. 15, no. 2, pp. 27–28, 2019.

## Biography

**Rixuan Qiu** is in State Grid Jiangxi Information & Telecommunication Company? Nanchang, China. Email: qiurixuanwork@163.com.

**Yu Fu** is in State Grid Jiangxi Information & Telecommunication Company? Nanchang, China. Email: gzgsgdfy@126.com.

**Jian Le** was born in Huanggang, Hubei, China in 1975. He received his Ph.D. degree in electrical engineering from Tsinghua University (THU), Beijing, China in 2006. He is currently Associate Professor with the college of electrical engineering and automation at Wuhan University (WHU), where he has been working on smart grid operation and power quality control technology. Email: lej01@tsinghua.org.cn.

**Fuyong Zheng** is in State Grid Jiangxi Information & Telecommunication Company, Nanchang, China. Email: 414833044@qq.com.

**Gan Qi** was born in Hengyang, Hunan, China in 1999. He received his B.S. degree from the School of Electrical Engineering and automation at Wuhan University (WHU), Wuhan, China, in 2021. He is now working towards a Master degree in electrical engineering at Wuhan University. He has been working on distribution generation control technology and power electronics.

**Chao Peng** is in State Grid Jiangxi Information & Telecommunication Company, Nanchang, China. Email: pengchao1988421@163.com.

**Yuancheng Li** was a postdoctoral research fellow in the Digital Media Lab, Beihang University. He has been with the North China Electric Power University, where he is a professor and the Dean of the Institute of Smart Grid and Information Security. He was a postdoctoral research fellow in the Cyber Security Lab, college of information science and technology of Pennsylvania State University.