# Softwarized IoT Network Immunity Against Eavesdropping With Programmable Data Planes

Gang Liu, *Student Member, IEEE*, Wei Quan, *Member, IEEE*, Nan Cheng, *Member, IEEE*, Deyun Gao, *Senior Member, IEEE*, Ning Lu, *Member, IEEE*, Hongke Zhang, *Fellow, IEEE*, and Xuemin Shen, *Fellow, IEEE*

***Abstract*—State-of-the-art mechanisms against eavesdropping first encrypt all packet payloads in the application layer and then split the packets into multiple network paths. However, versatile eavesdroppers could simultaneously intercept several paths to intercept all the packets, classify the packets into streams using transport fields, and analyze the streams by brute-force. In this article, we propose a programming protocol-independent packet processors (P4)-based network immune scheme (P4NIS) against the intractable eavesdropping. Specifically, P4NIS is equipped with three lines of defenses to provide a softwarized network immunity. Packets are successively processed by the third, second, and first line of defenses. The third line basically encrypts all packet payloads in the application layer using cryptographic mechanisms. Additionally, the second line re-encrypts all packet headers in the transport layer to distribute the packets from one stream into different streams, and disturbs eavesdroppers to classify the packets correctly. Besides, the second line adopts a programmable design for dynamically changing encryption algorithms. Complementally, the first line uses programmable forwarding policies which could split all the double-encrypted packets into different network paths disorderly. Using a paradigm of programmable data planes—P4, we implement P4NIS and evaluate its performances. Experimental results show that P4NIS can increase difficulties of eavesdropping and transmission throughput effectively compared with state-of-the-art mechanisms. Moreover, if P4NIS and state-of-the-art mechanisms have the same level of defending eavesdropping, P4NIS can decrease the encryption cost by 69.85%–81.24%.**

***Index Terms*—Eavesdropping attacks, network immune scheme, programming protocol-independent packet processors (P4), three lines of defenses.**

Manuscript received July 1, 2020; revised September 29, 2020; accepted December 15, 2020. Date of publication January 1, 2021; date of current version April 7, 2021. This work was supported in part by the National Key Research and Development Program of China under Grants 2018YFE0206800 and 2017YFE0121300; and in part by the Project "FANet: PCL Future Greater-Bay Area Network Facilities for Large-Scale Experiments and Applications" under Grant LZC0019. This article was presented in part at the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada. *(Corresponding author: Wei Quan.)*

Gang Liu, Wei Quan, and Deyun Gao are with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: weiquan@bjtu.edu.cn).

Nan Cheng is with the School of Telecommunication Engineering, Xidian University, Xi'an 710071, China.

Ning Lu is with the Department of Electrical and Computer Engineering, Queen's University, Kingston, ON K7L3N6, Canada.

Hongke Zhang is with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China, and also with PCL Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen 518040, China.

Xuemin Shen is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L3G1, Canada.

Digital Object Identifier 10.1109/JIOT.2020.3048842

## I. Introduction

Internet of Things (IoT) makes many aspects of human life (e.g., online shopping, online banking, and social networks) be inseparable with the Internet [1], [2]. However, eavesdroppers could steal personal sensitive information from unreliable network protocols used by the online services. For example, the hypertext transfer protocol (HTTP), which is a vulnerable network protocol, is widely used by online services [3]. Particularly, the IoT devices send sensitive information (e.g., user name, password, and phone number) in the type of plaintext, which is easy for attackers to eavesdrop [4], [5].

Various mechanisms have been proposed to solve the aforementioned problem [6]–[10]. For example, the hypertext transfer protocol secure (HTTPS) can encrypt packets in the application layer without exposing their sensitive information. Besides, a long and complicated encryption key can be used to increase difficulties of eavesdropping for the IoT systems [11]–[15]. Several mechanisms in the industry encrypt packets in the application layer and transmit them via one single path. Therefore, attackers could easily intercept all the packets of a complete session, perform brute-force analysis, and launch man-in-the-middle attacks (MITM) [16]–[19]. Although the cryptographic authentication and public-key infrastructure (PKI) certificate perform better at protecting the secret keys, they are "weak" versus supercomputers with improving computational capacity. In other words, even the most powerful encryption can be bypassed by the advances in computation capabilities [20]. Therefore, we argue that cryptographic security in the application layer is powerful, but never total.

Complicated forwarding mechanisms (e.g., multipath routing) are designed recently to further increase difficulties of eavesdropping. A path-following algorithm is developed by Hoang *et al.* [21] to solve active eavesdropping problems under different power allocation to access points. The multipath mechanism is integrated to split data streams in different physical paths to mitigate the eavesdropping attacks [22]–[24]. Besides, a three-stage Stackelberg game is used by Fang *et al.* [25] to model the behaviors of eavesdroppers. In fact, these mechanisms could mitigate eavesdropping attacks by splitting data streams into different network paths.

However, what if versatile attackers simultaneously intercept multiple paths? In that case, attackers could easily aggregate all the packets and intercept the content. To solve this problem, one good mechanism could do two key operations: 1) encrypting packets from one stream to split them into different streams and 2) distributing data packets (rather than streams) into different network paths. However, there are two challenges for the operations against eavesdropping: 1) how to design the encryption algorithms? and 2) how many packets should be distributed in each path?

To deal with the aforementioned challenges, an earlier version of this article [26] designed a programming protocol-independent packet processors (P4)-based network immune scheme (P4NIS) against the eavesdropping attacks. To make attackers difficult to eavesdrop and aggregate all traffic packets, the first line has a promiscuous forwarding policy. The policy can encapsulate each packet using different IoT network protocol headers and forward the packet via different paths disorderly. Besides, P4NIS is equipped with the second line that uses a transport encryption algorithm. The encryption algorithm could distribute traffic packets from one stream into different streams, and disturb eavesdroppers to classify them correctly. However, the forwarding policy and transport encryption algorithm are fixed in [26], which is not applicable due to two considerations: 1) eavesdroppers could crack the fixed forwarding policy and encryption algorithm, further destroy the proposed network immune scheme and 2) IoT traffics without personal privacy could be useless for eavesdroppers, thus, the fixed algorithm is wasteful to encrypt the traffics.

In this article, we augment the P4NIS by providing a *softwarized* IoT network immunity against eavesdropping attacks. Particularly, the first line is a *programmable* design that supports different forwarding policies. For example, the first line could split security-sensitive traffics into multiple network paths by choosing a complicated forwarding mechanism, and split delay-sensitive traffics into one single path by selecting a fastforwarding mechanism. The second line is also programmable for designing different encryption/decryption algorithms flexibly. For example, the second line could design a complicated algorithm for security-sensitive traffics, and a simple algorithm for delay-sensitive traffics. The main contributions of this article can be summarized as follows.

1) This article strengthens our previously proposed P4NIS to provide a programmable IoT network immunity against eavesdropping attacks. P4NIS comprehensively has three flexible lines of defenses to meet security requirements for different types of traffics.
2) The first line of P4NIS novelly supports programmable forwarding policies. Based on that, operators could design various forwarding policies for different types of IoT traffics, and defense eavesdroppers by dynamically splitting traffic packets via heterogeneous network paths.
3) The second line of P4NIS originally supports programmable encryption algorithms. Operators could customize diverse algorithms for encrypting all packet headers in the transport layer. After that, eavesdroppers are difficult to classify the packets into streams correctly. Besides, the third line encrypts all packet payloads in the application layer using cryptographic mechanisms.
4) Using a paradigm of programmable data planes—P4, we implement the P4NIS and conduct series of experiments. Compared with state-of-the-art mechanisms, P4NIS can increase difficulties of eavesdropping significantly, and increase transmission throughput by 31.68%. Moreover, if P4NIS and state-of-the-art mechanisms have the same level of defending eavesdropping, P4NIS can decrease the encryption cost by 69.85%–81.24%.

The remainder of this article is organized as follows. Section II presents related works. Section III theoretically analyzes the advantages of P4NIS. The principle and deployment of P4NIS are introduced in Sections IV and V, separately. Several experiments are done in Section VI to evaluate the performance of P4NIS. Finally, we conclude this article in Section VII.

## II. Related Works

Traditionally, cryptographic-based mechanisms are proposed by network researchers to encrypt packet payload against eavesdropping attacks. For example, an efficient key management is proposed by Howarth *et al.* [27] to encrypt multicast traffic transmitted via satellite, where eavesdropping attacks can be easily performed. A new initial common key sharing protocol is presented by Yao *et al.* [28] to encrypt the secret data that are easily intercepted in the wireless links. Besides, a novel encryption scheme and decision fusion rules proposed by Jeon *et al.* [29] prevent passive eavesdropping in wireless sensor networks. The encoding scheme and advanced encryption standard (AES) algorithm are utilized by other researchers against eavesdropping attacks [30].

In addition to encrypt the packet payload, it is believed by network researchers that the packet header also needs to be encrypted. For example, service-oriented routers are used by Tennekoon *et al.* [31] for providing secure data transmission by encrypting data packets including the header and trailer information. An enhancing Internet Protocol Security (IPsec) mechanism is presented by Song *et al.* [32] against eavesdropping attacks. Besides, a IPsec tunnel-based scheme, named SDMNBlackhual, is proposed by Liyanage *et al.* [33] to secure the software-defined mobile network communication. A lightweight in-network anonymity solution, proposed by Wang *et al.* [34], encrypts a client's IPv4 address to protect user privacy. Surveillance protection in the network elements (SPINE) is presented by Datta *et al.* [35] to conceal IP addresses in packet headers from eavesdroppers.

Recently, various new network protocols have been proposed to transmit packets using multiple protocols on the Internet. For example, a novel network protocol, named SINET, is designed to solve triple bindings in the conventional Internet, including resource/location binding, user/network binding and control/data binding [36]. Besides, named data networking (NDN), which is a clean slate protocol proposed by Zhang *et al.* [37], can effectively forward packets using its name rather than an IP address. Locator/ID Separation Protocol (LISP) proposed by Farinacci *et al.* [38] decouples the binding between device identity and location. A novel variable-length identifier (VLI) datagram header is designed to effectively interconnect different scales of IoT networks [39]. A backwards-compatible and extensible Internet is proposed by McCauley *et al.* [40] to simultaneously merge various network protocols. Moreover, a software-defined adaptive transmission approach is designed by Quan *et al.* [41] to select various transmission control policies for the time-varying mobile network environment. A stateful forwarding protocol is presented to increase the efficiency and security of the Internet [42], [43]. Although these works enable transmitting packets using a new network protocol, the details about scheduling packets transmissions using multiple protocols are not introduced.

Currently, on one hand, a new control/data plane-separated network architecture is designed by network researchers to support a programmable control plane. For example, software-defined networking (SDN) proposed by McKeown *et al.* [44] separates the network's control logic from the underlying routers and switches to control the devices forwarding actions easily. Besides, the OpenFlow protocol is designed to implement the SDN concept by abstracting network communications as flows to be processed by network elements [45]. Such a protocol includes a communication interface between SDN controllers and switches, also defines the switches forwarding actions by a set of flow rules [46]. On the other hand, a novel network language is designed to support a programmable data plane. Notably, a high-level language is proposed by Bosshart *et al.* [47] for programming protocol-independent packet processors (named P4). Several target devices have been built to support the high-level programmable network language [48]. Besides, the aforementioned new network protocols (e.g., NDN) are first implemented by Signorello *et al.* [49] using the flexible network language.

## III. Multiple Lines of Defenses

This section theoretically analyzes the necessary of choosing multiple lines of defenses (e.g., P4NIS) rather than traditional cryptographic mechanisms with a longer key.

*Definition 1:* Let $E_t$ be a traditional cryptographic mechanism, and let $l_t$ denote the encryption key size. Besides, let $E_{p1}, E_{p2}$, and $E_{p3}$ be the first, second, and third lines of defenses, separately. Let $l_{p1}, l_{p2}$, and $l_{p3}$ denote the cryptographic key sizes of the first, second, and third lines, separately.

*Definition 2:* Let $\mathbb{M} = \{m_1, m_2, m_3\}$ be a packet in plaintext. Typically, an IP packet in plaintext could be denoted as

$$\left[\underbrace{\overbrace{..\,\mathrm{IP_{dst}}, \mathrm{IP_{src}}\,..}^{\mathrm{IP-header}}}_{m_1} \| \underbrace{\overbrace{..\,\mathrm{Port_{dst}}, \mathrm{Port_{src}}\,..}^{\mathrm{Transport-header}}}_{m_2} \| \underbrace{\mathrm{Payload}}_{m_3}\right] \quad (1)$$

whereas $m_1$ denotes $\{\mathrm{IP_{dst}}, \mathrm{IP_{src}}\}$ fields in the IP header. $m_2$ denotes $\{\mathrm{Port_{dst}}, \mathrm{Port_{src}}\}$ fields in the transport header. $m_3$ denotes the payload data carried for the application. Let $l_{m1}, l_{m2}$, and $l_{m3}$ be the lengths of the plaintext $m_1, m_2$, and $m_3$, separately. In terms of engineering practice, the relationship among $l_{m1}, l_{m2}$, and $l_{m3}$ satisfies: $l_{m1}, l_{m2} \ll l_{m3}$.

*Definition 3:* Let $\mathbb{C}_i(\mathbb{M})$ be a ciphertext, which is from encrypting a plaintext $(\mathbb{M})$ using a cryptographic mechanism $(i)$. Assuming that the traditional cryptographic mechanism has an encryption key $k_t$. Because the traditional mechanism encrypts the packet payload in the application layer, the corresponding ciphertext is

$$\mathbb{C}_t(\mathbb{M}) = E_{t,k_t}(\mathbb{M}) = E_{t,k_t}(m_3) \quad (2)$$

assuming that multiple lines of defenses have encryption keys $k_{p1}, k_{p2}$, and $k_{p3}$ for the first, second, and third lines of defenses, respectively. Then, the corresponding ciphertext is

$$\begin{aligned}\mathbb{C}_p(\mathbb{M}) &= E_{p,k_{p1},k_{p2},k_{p3}}(\mathbb{M}) \\ &= \left\{E_{p1,k_{p1}}(m_1), E_{p2,k_{p2}}(m_2), E_{p3,k_{p3}}(m_3)\right\}. \quad (3)\end{aligned}$$

*Theorem 1:* Multiple lines of defenses (e.g., P4NIS $E_{p1}, E_{p2}, E_{p3}$) are more difficult to perform brute-force attacks than one single line of defense (e.g., the traditional cryptographic mechanism $E_t$).

*Proof:* Assuming that an eavesdropper steals a pair of plaintext $\mathbb{M}'$ and ciphertext $\mathbb{C}'$. Then, the eavesdropper performs a brute-fore analysis for the correct key. If we use the traditional cryptographic mechanism $E_t$, then the plaintext $\mathbb{M}'$ has $2^{l_t}$ alterative ciphertexts (denoted as $i$) because the encryption key $k_t$ has $2^{l_t}$ alterative values. Whereas, $i = E_{t,k_t}(\mathbb{M}') = E_{t,k_t}(m_3')$, and there is only one correct key that makes $\mathbb{C}' = i$ come into existence. In that case, the eavesdropper could establish a table to store each pair of $k_t$ and $i$, then, match $\mathbb{C}'$ with each $i$ in that table to find the correct key $k_t$. The time complexity of brute-force searching the key is relevant to the table size, and could be denoted as

$$\mathrm{Crack}(E_t) = O\left(2^{l_t}\right). \quad (4)$$

If we use multiple lines of defenses (e.g., P4NIS), then the ciphertext could be denoted as $\{E_{p1,k_{p1}}(m_1'), E_{p2,k_{p2}}(m_2'), E_{p3,k_{p3}}(m_3')\}$. Whereas, $k_{p1}$ is the key for the first line of defense and has $2^{l_{p1}}$ alternative values; therefore, the plaintext $m_1'$ has $2^{l_{p1}}$ alterative ciphertexts (denoted as $j$). Whereas, $j = E_{t,k_{p1}}(m_1{}')$, and there is only one correct key that makes $\mathbb{C}(m_1') = j$ come into existence. In that case, the eavesdropper could establish a table to store each pair of $k_{p1}$ and $j$, then, match $\mathbb{C}(m_1')$ with each $j$ in that table to find the correct key for the first line of defense. The time complexity of brute-force searching the key for the first line of defense could be denoted as $O(2^{l_{p1}})$. Similarly, the time complexity of brute-force searching the keys for the second and third lines of defenses could be $O(2^{l_{p2}})$ and $O(2^{l_{p3}})$, separately. Therefore, the time complexity of brute-force searching keys for multiple lines of defenses (e.g., P4NIS) could be denoted as

$$\mathrm{Crack}(E_p) = O\left(2^{l_{p1}} * 2^{l_{p2}} * 2^{l_{p3}}\right) = O\left(2^{l_{p1}+l_{p2}+l_{p3}}\right). \quad (5)$$

Generally, the mechanism $E_{p1}$ applied in the first line of P4NIS splits packets from one stream into different network paths. In terms of engineering practice, there are not too much network paths for the first line to choose; therefore, the relationship among $l_{p1}, l_{p2}, l_{p3}$ could be $l_{p1} \ll l_{p2}$, and $l_{p3}$. Besides, the mechanism applied in the third line of P4NIS is the traditional cryptographic mechanism, i.e., $l_{p3} = l_t$. Then, we have that $O(2^{l_t}) = O(2^{l_{p3}}) < O(2^{l_{p1}+l_{p2}+l_{p3}})$; therefore, $\mathrm{Crack}(E_t) < \mathrm{Crack}(E_p)$. That is, multiple lines of defenses are more difficult for eavesdroppers to perform brute-force attacks than the traditional cryptographic mechanism. ■

*Theorem 2:* If the traditional cryptographic mechanism increases its key length ($l_t$) to be the same with that of multiple lines of defenses (e.g., P4NIS $l_{p1} + l_{p2} + l_{p3}$), i.e., the time complexities of cracking those two mechanisms are the same, then multiple lines of defenses have a lower encryption cost than the traditional cryptographic mechanism.

*Proof:* Assuming that the time complexity of cracking a cryptographic mechanism is directly proportional to its key size, and there are keys $k'_t, k'_{p1}, k'_{p2}$, and $k'_{p3}$ that make $O(2^{l'_t}) = O(2^{l'_{p1}+l'_{p2}+l'_{p3}})$ come into existence. That is, the time complexity of cracking the traditional cryptographic mechanism is the same with that of multiple lines of defenses (e.g., P4NIS). Therefore, the length of key $k'_t$ is the same with hybrid keys $k'_{p1}, k'_{p2}$, and $k'_{p3}$, i.e., we have that

$$l'_t = l'_{p1} + l'_{p2} + l'_{p3} \tag{6}$$

assuming that the encryption cost is directly proportional to the encryption time and let $T(\bullet)$ be the encryption time using a cryptographic mechanism. Then, the encryption cost of the traditional cryptographic mechanism could be denoted as

$$T(E_t) = T\Big(E_{t,k'_t}(m_3)\Big) \tag{7}$$

and the encryption cost of multiple lines of defenses (e.g., P4NIS) could be denoted as

$$\begin{aligned} T(E_p) &= T\Big(\Big\{E_{p1,k'_{p1}}(m_1), E_{p2,k'_{p2}}(m_2), E_{p3,k'_{p3}}(m_3)\Big\}\Big) \\ &= T\Big(E_{p1,k'_{p1}}(m_1)\Big) + T\Big(E_{p2,k'_{p2}}(m_2)\Big) + T\Big(E_{p3,k'_{p3}}(m_3)\Big). \end{aligned} \tag{8}$$

According to Definition 2, we have that $l_{m1}, l_{m2} \ll l_{m3}$. It is noted that a longer plaintext has a longer encryption time [50]; therefore, we have that $T(E_p) \approx T(E_{p3,k'_{p3}}(m_3))$. According to (6), we can get that the length of key $k'_t$ is smaller than $k'_{p3}$. Besides, Amalarethinam and Leena [51] evaluated that a longer key length has a larger encryption time. Therefore, we have that $T(E_{p3,k'_{p3}}(m_3)) < T(E_{t,k'_t}(m_3))$. In summary, if the traditional cryptographic mechanism and multiple lines of defenses (e.g., P4NIS) have similar key length, then multiple lines of defenses (e.g., P4NIS) have a lower encryption cost than the traditional cryptographic mechanism. ■

## IV. P4NIS Principle

This section introduces how each line of defense looks like. For example, the technological principles and theoretical algorithms of each line are introduced in detail.

### A. First Line of Defense

The first line has multipath defense, multiprotocol defense, and programmable forwarding mechanisms. We can customize the forwarding mechanism to schedule the multipath/protocol defenses. For example, the forwarding mechanism encapsulates one packet with the IPv6 header and emits it via one path. Then, the mechanism encapsulates another packet with the IPv4 header and emits it via another path.

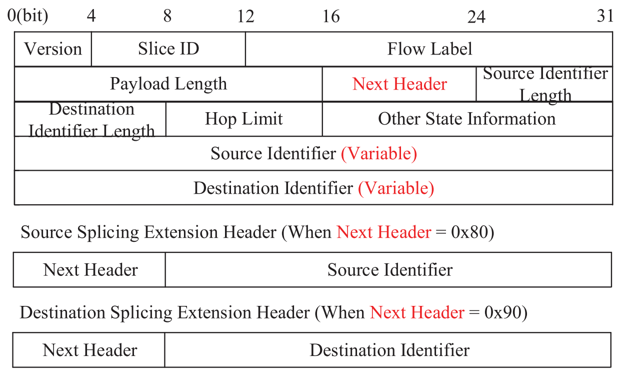
0(bit)
4
8
12
16
24
31
Version
Slice ID
Flow Label
Payload Length
Next Header
Source Identifier Length
Destination Identifier Length
Hop Limit
Other State Information
Source Identifier (Variable)
Destination Identifier (Variable)
Source Splicing Extension Header (When Next Header = 0x80)
Next Header
Source Identifier
Destination Splicing Extension Header (When Next Header = 0x90)
Next Header
Destination Identifier

Fig. 1. VLI packet header.

*1) Multipath Defense:* To augment the promiscuous and stateful forwarding policy by having multiple alternative options, the first line of P4NIS manages multiple heterogeneous paths in which the packets are difficult to be entirely eavesdropped. For example, P4NIS is equipped with three 4G LTE modules to forward packets through wireless links. The 4G LTE modules are runnable on the Linux, Windows, and MAC operating system by coding a software driven script,[1] To make the wireless links have heterogeneous parameters (e.g., delay, packets loss rate and signal strength), each 4G module is embedded with the China Unicom, China Mobile, and China Telecom SIM card, separately.

*2) Multiprotocol Defense:* For forwarding policy to have various alternative network protocols, the first line of defense has the traditional IPv4 and IPv6 network protocols. Besides, a novel VLI datagram header called VLI [39], which aims to effectively interconnect different scales of networks, is also embedded into P4NIS scheme.

1) IPv4 is one of the core protocols of the Internet and is widely used in today's Internet. Without the options filed, the IPv4 header has a size of 20 B. Otherwise, the maximum size is 60 B.
2) IPv6 packet header consists of eight fields, of which the next header field provides the opportunity to extend the protocol in the future. Therefore, the IPv6 packet header has a fixed size of 40 B.
3) VLI is a new network datagram that aims to merge different scales of networks. As shown in Fig. 1, the VLI packet header consists of 11 fields, of which the next header field provides the same capacity as the IPv6 packet header. However, different from the IPv6 packet header, the VLI packet header can provide arbitrary bit addresses (the maximum is 256 b) using the splicing extension header (as shown in the bottom of Fig. 1).

*3) Programmable Forwarding Defense:* To provide a programmable network immunity against eavesdropping attacks, the first line of defense has a scalable design that supports programmable forwarding policies. The design (as shown in Fig. 2) has three main parts: 1) ingress pipeline; 2) forwarding policy controller; and 3) egress pipeline. It is noted that the ingress and egress pipelines correspond to the P4 ingress

[1]The driven script for 4G LTE modules: https://github.com/KB00100100/P4NIS/tree/master/Multi_paths.

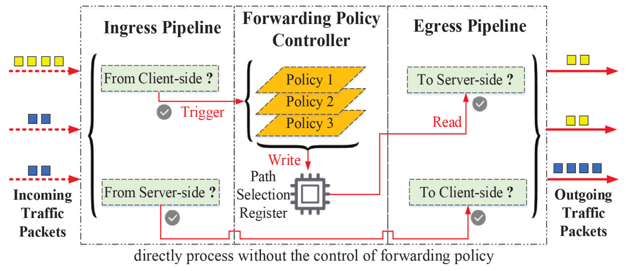

Fig. 2. Programmable forwarding defense.

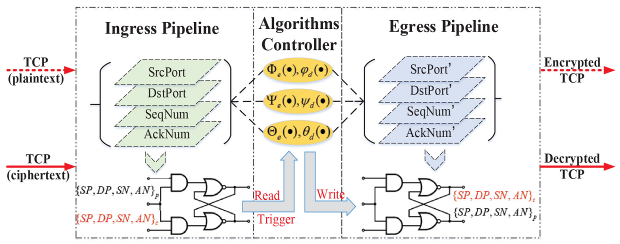

Fig. 3. Programmable cryptographic design.

and egress pipelines. Besides, the controller does not correspond to traditional SDN controller because the former one controls a switch's register while the SDN controller controls the switch's flow tables. The forwarding policy controller has a path-selection register, which could be written by forwarding polices and read in the ingress and egress pipelines. Since a policy only needs to write its selection of forwarding path to the register, network operators could program different forwarding policies using the scalable design. Besides, both the ingress and egress pipelines have processes to determine whether traffic packets are from client side or server side. If one packet are from client side, then the path-selection register chooses a path to forward the packet to the server side. Otherwise, the packet is directly forwarded to the client side.

Based on the coordination of these three parts, traffic packets could be forwarded disorderly. Particularly, when incoming a client-side packet, the ingress pipeline first triggers the forwarding policy controller. Then, the controller chooses one forwarding policy and writes the value of selected path to the path-selection register. Finally, the egress pipeline reads the path-selection register and forwards the packet to the selected path. Besides, when incoming a server-side packet, the ingress and egress pipelines directly process it without the control of forwarding policies. If the packet destination is the client, the packet will be forwarded to the path connected to the client; otherwise, it will be dropped in the egress pipeline. Using the scalable design, operators could easily program diverse forwarding policies and dynamically change them to satisfy different types of traffics. For example, splitting security-sensitive traffics into multiple different paths, and delay-sensitive traffics into one single path.

*4) Promiscuous and Stateful Forwarding Policy:* Based on the programmable forwarding defense, we propose a simple but powerful stateful forwarding policy to prevent eavesdropping. That is, although attackers can eavesdrop every single path/protocol easily, they can hardly intercept all traffic packets because the forwarding policy can forward the traffic packets through different paths/protocols. Particularly, the promiscuous policy is a stateful whose state can be denoted by

$$S_{ij}, \{i = 1, 2, 3; j = 1, 2, 3, 4\} \tag{9}$$

of which $i$ denotes the type of the protocol and $j$ denotes the number of the path. For example, $i = 1, 2, 3$ represents the IPv4, IPv6, and VLIs (e.g., VLI) protocol, respectively, $j = 1, 2, 3, 4$ severally represents the 1st, 2nd, 3rd, and 4th path. In that way, the state $S_{ij}$ denotes the traffic packets are forwarded through the $j_{th}$ path by using the protocol $i$. Besides, the packets forwarding state $S_{ij}$ can transfer to $S_{i'j'}$ according to the link characteristics and service requirements. The transition can be denoted by

$$S_{ij} \xrightarrow[\mathbb{R}(\bullet)]{\mathbb{P}(i')>\mathbb{P}(i) \text{ and } \mathbb{P}(j')>\mathbb{P}(j)} S_{i'j'} \tag{10}$$

whereas $\mathbb{P}(i)$ is a set of link characteristics when using the network protocol $i$. Such a set includes RTT, packet loss rate, and other link characteristics. $\mathbb{R}(\bullet)$ depicts a set of service requirements including QoS, security. and etc. Particularly, if one of the link characteristics is always better than another one, then the forwarding state is still $S_{ij}$. Such a forwarding state reaches the QoS requirement perfectly because the current selected link characteristic is the best one. However, it is not suited for the security requirement because the packets are always forwarded via one path using one protocol. Therefore, considering some service requirements, such as the security, the forwarding state should be frequently changed to make all the traffic packets too difficult to be intercepted. Algorithm 1 shows the pseudocode of the forwarding state transition process.

**Algorithm 1** The Forwarding State Transition

```
Input: Array $\mathbb{P}(i,j)$, $\mathbb{R}(\bullet)$ and Register $S_{ij}$
Output: Register $S_{i'j'}$
1.  procedure Process ($\mathbb{P}(i,j)$, $\mathbb{R}(\bullet)$, $S_{ij}$)
2.    //select the best link characteristics
3.    $i', j' \Leftarrow MAX(P(i,j))$;
4.    //get current forwarding state
5.    $S_{ij}$ = register.read();
6.    //consider the service requirements
7.    case $\mathbb{R}(\bullet)$:
8.      when QoS: return register.write($S_{i'j'}$);
9.      when Security: return register.write(not $S_{ij}$);
10.     default: return register.write($S_{i'j'}$);
11.   end case
12. end procedure
```

## B. Second Line of Defense

The second line of defense has a programmable cryptographic design to support various encryption algorithms. Using the design, operators could customize different cryptographic algorithms and encrypt packets from one stream to split them into different streams. In that case, eavesdroppers are difficult to classify the encrypted packets into streams correctly.

*1) Programmable Cryptographic Design:* To support various encryption algorithms for P4NIS, we propose a backward-compatible design in the second line of defense. Particularly,

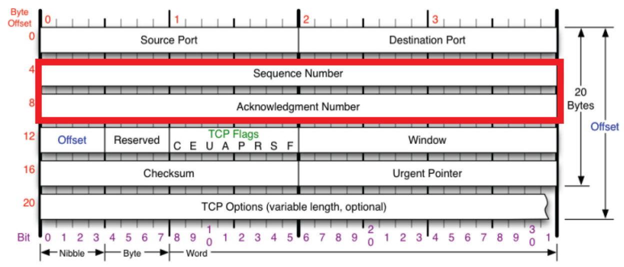

Fig. 4. TCP packet header.

the design (as shown in Fig. 3) has three main parts: 1) ingress pipeline; 2)algorithms controller; and 3) egress pipeline. It is noted that the ingress and egress pipelines correspond to the P4 ingress and egress pipelines. Besides, the controller does not correspond to the traditional SDN controller because the former one controls a switch's register while the SDN controller controls the switch's flow tables. In the ingress pipeline, a TCP packet, whether it is in the type of plaintext or ciphertext, is used to extract the fields of source port, destination port, sequence number, and acknowledge number (as shown in Fig. 4). The four fields can be combined together (denoted as $\{SP, DP, SN, AN\}$) and stored in the local registers. Note that $\{SP, DP, SN, AN\}_p$ represents that the four fields are in the type of plaintext, $\{SP, DP, SN, AN\}_c$ represents that the four fields are in the type of ciphertext. After finishing the storage, the algorithm controller is triggered to encrypt/decrypt the fields using selected algorithms. As Fig. 3 depicts, $\Phi_e(\bullet)$, $\Psi_e(\bullet)$, and $\Theta_e(\bullet)$ are the encryption algorithms, $\varphi_d(\bullet)$, $\psi_d(\bullet)$, and $\theta_d(\bullet)$ are the decryption algorithms. Aside from selecting the existing encrypted/decrypted algorithms, the controller can also provide a backward-compatible selection for other algorithms. It is worth mentioning that all the algorithms can be dynamically replaced even if the controller is at runtime. The en-/decryption process can be expressed by

$$\begin{Bmatrix} \text{SP} & \text{DP} \\ \text{SN} & \text{AN} \end{Bmatrix}_p \xrightarrow[\text{read}]{\text{reg}} \begin{Bmatrix} \Phi_e \\ \Psi_e \\ \Theta_e \end{Bmatrix} \xrightarrow[\text{write}]{\text{reg}} \begin{Bmatrix} \text{SP} & \text{DP} \\ \text{SN} & \text{AN} \end{Bmatrix}_c$$

$$\begin{Bmatrix} \text{SP} & \text{DP} \\ \text{SN} & \text{AN} \end{Bmatrix}_c \xrightarrow[\text{read}]{\text{reg}} \begin{Bmatrix} \varphi_d \\ \psi_d \\ \theta_d \end{Bmatrix} \xrightarrow[\text{write}]{\text{reg}} \begin{Bmatrix} \text{SP} & \text{DP} \\ \text{SN} & \text{AN} \end{Bmatrix}_p.$$

In the egress pipeline, the four fields are replaced with the values stored in local registers. In that case, the process of encrypting/decrypting TCP packets is completely finished.

*2) Encrypted/Decrypted Transport Algorithms:* Based on the programmable cryptographic design, we propose an encrypted/decrypted transport algorithm to prevent the eavesdropper from stealing all the packets from one stream and recovering the payloads. Particularly, an S-BOX matrix is used to encrypt TCP packets, and an inverse S-BOX matrix is used to decrypt the encrypted TCP packets. Both the matrices are based on the AES-128 encryption algorithm which has $2^{127}$ keys, they can be generated by using multiple nonlinear mapping rules. Typically, the generation can be done in three steps. First, the S-BOX is initialized with $16 \times 16$ B elements and sorted in ascending order. For example, the first line elements

**Algorithm 2** The encrypted/decrypted transport

```
Input: Class RawPacket or EncryptedPacket
Output: Class EncryptedPacket or RawPacket
1.  constant S-BOX = {}, S-BOX_I = {};
2.  //generate the (inverse) S-BOX
3.  S-BOX = GenerateAlgorithm();
4.  S-BOX_I = GenerateInverseAlgorithm();
5.  procedure Process (Packet)
6.    if Packet.raw() == True then
7.        Packet.seq ⇐ encrypt(Packet.seq, S-BOX);
8.        Packet.ack ⇐ encrypt(Packet.ack, S-BOX);
9.    else if Packet.encrypted() == True then
10.       Packet.seq ⇐ decrypt(Packet.seq, S-BOX_I);
11.       Packet.ack ⇐ decrypt(Packet.ack, S-BOX_I);
12.   end if
13. end procedure
```

of the S-BOX are {00}, {01} , . . . , {0F}, the second line elements of the S-BOX are {10}, {11}, . . . , {1F}, the $x$th line and $y$th column element can be denoted as $\{(x-1)(y-1)\}$. Second, each byte element in the S-BOX is mapped as its inverse value in a finite field. Such an inverse can be denoted by

$$\{\text{(x-1)(y-1)}\} \xrightarrow{GF(2^8)} \left\{\text{(x-1)}'\text{(y-1)}'\right\}, \{00\} \rightarrow \{00\} \tag{11}$$

whereas $GF(\bullet)$ is a finite field, $2^8$ represents there are 256 elements in the finite field. Third, convert each bit of all the bytes elements using the following equations:

$$\begin{aligned} b_i' = b_i &\oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \\ &\oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i \end{aligned} \tag{12}$$

$$c = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{13}$$

whereas $b_i$ is the $i$th bit of a byte element, $c_i$ is the $i$th bit of the constant byte $c$. Using the aforementioned three steps, the generated inverse S-BOX matrix is shown as

$$\begin{bmatrix} 63 & 7c & \cdots & ab & 76 \\ ca & 82 & \cdots & 72 & c0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ e1 & f8 & \cdots & 28 & df \\ 8c & a1 & \cdots & bb & 16 \end{bmatrix} \tag{14}$$

and the generated inverse S-BOX matrix is shown as

$$\begin{bmatrix} 52 & 09 & \cdots & d7 & fb \\ 7c & e3 & \cdots & e9 & cb \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a0 & e0 & \cdots & 99 & 61 \\ 17 & 2b & \cdots & 0c & 7d \end{bmatrix}. \tag{15}$$

By using the generated (inverse) S-BOX matrix, the second line of defense can encrypt/decrypt the TCP sequence and acknowledge fields to avoid eavesdroppers from using the two fields to decode the packet payload. Algorithm 2 shows the pseudocode about how to encrypt/decrypt the key fields using the (inverse) S-BOX matrix.

## C. Third Line of Defense

The third line of defense is the traditional cryptographic mechanism, such as RSA2048 to encrypt packet payload. It is

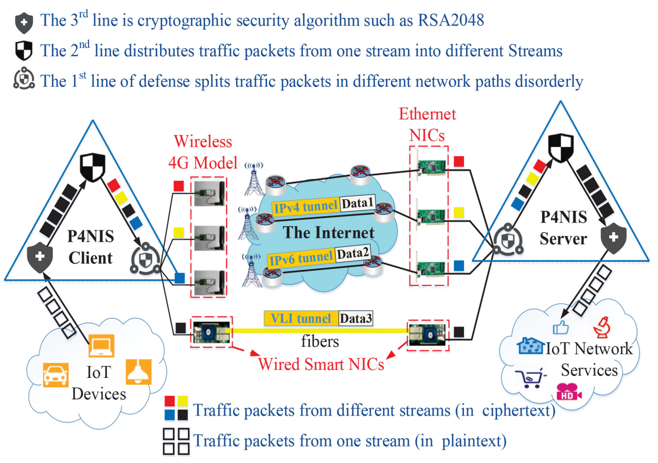
The 3rd line is cryptographic security algorithm such as RSA2048
The 2nd line distributes traffic packets from one stream into different Streams
The 1st line of defense splits traffic packets in different network paths disorderly
Wireless 4G Model
Ethernet NICs
IPv4 tunnel Data1
The Internet
IPv6 tunnel Data2
VLI tunnel Data3
fibers
Wired Smart NICs
P4NIS Client
P4NIS Server
IoT Devices
IoT Network Services
Traffic packets from different streams (in ciphertext)
Traffic packets from one stream (in plaintext)

Fig. 5. P4NIS system.

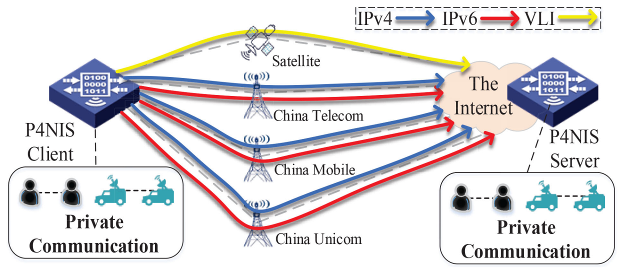

Fig. 6. Typical use case of P4NIS.

noting that the encryption is executed by IoT devices rather than P4NIS devices. Besides, since the first and second lines of P4NIS could increase difficulties of eavesdropping, the IoT devices could use simple and lightweight cryptographic mechanisms due to the constricted resources. For simplicity, we do not introduce RSA2048 in detail because it is a well-known cryptographic mechanism.

## V. P4NIS Workflows and Deployment

Based on the P4NIS principle, this section introduces how each line of defense works together. Besides, a typical application scenario of P4NIS is also introduced in detail.

### A. Workflows of P4NIS

Fig. 5 shows the architecture of P4NIS mechanism. Particularly, a client is used to transmits users' traffic packets, and a server is used to transmit network services packets. Both the client and server have the first line of defense that has multipath resources including wireless 4G links and wired fiber links. Besides, the first line of defense also has multiprotocol resources, such as IPv4, IPv6, and variable-length identifiers (e.g., VLI). Using the multipath/protocol resources, a promiscuous and stateful forwarding policy is equipped to make the traffic packets or the services streams difficult to be eavesdropped. Even if a versatile attacker can destroy the first line, the P4NIS client and server have the second line of defense which is an encrypted/decrypted transport algorithm to prevent the eavesdropper decoding all packet payloads.

Generally, when the P4NIS client receives a traffic packet from the user groups, the second line of defense encrypts some key fields using the aforementioned algorithm. Then, the client's first line of defense selects a path and protocol to forward the packet using the promiscuous forwarding policy. After the selection, the client encapsulates the packet using IPv4, IPv6, and variable-length identifiers (e.g., VLI) tunnel, in which (not include the VLI tunnel) the packet can be forwarded to the P4NIS server across the Internet. After the server receiving the packet from a forwarding path of the first line, its second line of defense decrypts the key fields using the aforementioned algorithm and forwards the packet to a network service provider. Finally, the provider responds to the packet and forwards the acknowledgment packet using the reverse path between the P4NIS server and client. For conciseness, we do not introduce repeatedly the reverse process here because it is similar to the uplink process.

### B. Typical Application Scenario of P4NIS

Fig. 6 depicts a typical use case of P4NIS for secure private communication. Particularly, a business man is on the trip outside his company, and he wants to exchange secret information with his company. If the man uses traditional communication technologies, then the secret information (e.g., voice call) will be easily eavesdropped by the native communication operators. Furthermore, using the intercepted secret information, versatile attackers could blackmail the company and cause significant economic losses. Therefore, a portable P4NIS client is necessary for the man against eavesdropping.

The portable P4NIS client is easy to use without any configuration. Particularly, the business man first connects his mobile phone, tablet PC and laptop to the P4NIS client via wireless/wired links. Then, if the man downloads a private file from his company's Web or chats with his colleagues about backstair contracts, the P4NIS client can disorderly forward the secret information via multiple 4G links. In that case, eavesdroppers are difficult to intercept all the traffic packets, even if they can, the second defense line of P4NIS can avoid the attackers decrypting all packet payloads. After the disordered packets arrive at the P4NIS server, the server can quickly sort them and safely transmit them to a content producer or other users inside the enterprise.

## VI. Performance Evaluation

This section evaluates the performance of P4NIS in a practical network environment. Particularly, we implement the P4NIS using the P4 language, which includes the client part and the server part.[2] We conduct series of experiments to evaluate the performance in terms of success probability of eavesdropping, cryptographic complexity, and transmission throughput. For comparison, other state-of-the-art mechanisms, including single-path and multipath transmission, are also implemented.

### A. Experimental Setup

We choose P4 to implement P4NIS rather than other language (e.g., *C*) for three reasons.

[2]Codes for P4NIS are available via Github: https://github.com/KB00100100/P4NIS.

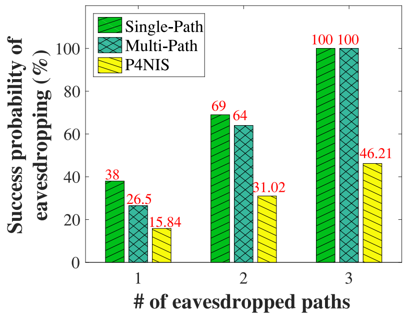

Fig. 7. Difficulties of eavesdropping.

1) The first line of P4NIS encapsulates the packets with different L3 header. P4 could integrate any L3 header flexibly with a few lines of codes while *C* could not. For example, current Linux is embed with TCP/IP stack, it is difficult to integrate NDN or other L3 protocol.
2) The first line of P4NIS splits the packets into different network paths. P4 could choose a network path for the packets using its meta-data while *C* could not.
3) The second line of P4NIS supports various encryption algorithms. P4 could provide external register interfaces for choosing encryption algorithms while *C* could not. Besides, a software switch, named BMv2, is installed to support running the P4 codes. The reason for choosing BMv2 is that we focus on evaluating the functions of P4NIS rather than its throughput. It is reasonable to choose BMv2 because that it has 553 Mb/s throughput tested with 16 cores 1.70-GHz CPU and a 64-GB RAM, which satisfies the requirement of 100 Mb/s. It is worth mentioning that the client has three wireless 4G links while the server does not have. In other words, the client is portable and runnable without using the wired smart NIC; while the server is only runnable at a fixed position (such as our laboratory). In our experiments, we deploy both the client and the server at our laboratory to evaluate the feasibility of the VLI protocol.

## B. Experimental Results of Eavesdropping Defense

*1) Eavesdropping Results:* To intuitively evaluate the performance of P4NIS three lines of defenses, we act as a versatile eavesdropper to intercept the traffic packets between the client and server. Notably, a legitimate PC is connected to the P4NIS client to download a text file whose content is repeated with “SINET is cool!”. Among the clients, we use a powerful Wireshark tool to sniff the packets forwarded in a single path, multiple paths and all available paths, separately. As a comparison, we use the Wireshark to sniff a legitimate user stream in the PC. It is worth mentioning that we decapsulate the origin IP packets from the eavesdropped packets to evaluate the performance of the encrypted transport algorithm.

All the evaluation results including the origin Wireshark pcap file can be found at the GitHub and Fig. 10 shows snapshots of the results. Particularly, eavesdropping on one single path can only intercept a part of traffic packets [encapsulated with IPv6 header in Fig. 10(a)]. While eavesdropping on multiple paths can intercept much more traffic packets than eavesdropping on one single path [encapsulated with two different IPv4 headers in Fig. 10(b)]. Besides, eavesdropping on all available paths can intercept all the traffic packets, but it cannot decode all packet payloads because the second line of defense is equipped with a powerful encrypted transport algorithm. As Fig. 10(c) shows, all packets are labeled with black background color by the Wireshark, which indicates that they are abnormal packets. Moreover, the Wireshark stream content shows some bytes missing in capture file which indicates that the payload included in the eavesdropped packets cannot be entirely recovered. Regarding the legitimate users, Fig. 10(d) depicts that all the traffic packets can be normally extracted by the Wireshark without the black label. The Wireshark stream content shows “SINET is cool!", i.e., the packets can be recovered with entire payload.

*2) Success Probability of Eavesdropping:* In our experiments, the difficulties of eavesdropping are quantified as the success probability of eavesdropping [52]. Particularly, comparison experiments between single-path, multipath, and P4NIS are done to evaluate that the first line of defense can effectively decrease the success probability of eavesdropping. Besides, we use a PC connected to the P4NIS client to download a text file and act as a versatile attacker to intercept the traffic packets from 1/2/3 paths randomly. It is worth mentioning that all packets are transmitted in the type of plaintext to evaluate the performance without the third line of defense. The single-path and multipath mechanisms are also implemented in the client without any encryption algorithm.

Fig. 7 shows the results. The success probability of eavesdropping is the ratio between the packets eavesdropped by an attacker and the packets transmitted by a legitimate user. For example, 100% means that the eavesdropper successfully intercepts all the traffic packets transmitted by the user. If we follow a TCP stream using all the eavesdropped packets in the Wireshark, we can entirely extract all packet payloads. As Fig. 7 depicts, whatever several eavesdropped paths are, P4NIS can prevent all the traffic packets from being intercepted. Concretely, success probabilities of eavesdropping are 15.84%, 31.02%, and 46.21%, separately, i.e., P4NIS approximately decreases the success probability of eavesdropping by 216.4% compared with the other two mechanisms. Besides, using the second line of defense, the success probability of eavesdropping can exponentially decrease by $2.164 \times 2^{127}$ times because the encryption algorithm has $2^{127}$ alternative keys.

*3) Cryptographic Complexity:* We also evaluate the effectiveness of P4NIS in the case of versatile attackers successfully eavesdropping all the traffic packets. In our experiments, the effectiveness is quantified as the cryptographic complexity, i.e., how many alternative keys for the eavesdroppers to brute-force search. For comparison, the state-of-the-art mechanism (multipath) is implemented, which encrypts all packet payloads in application layer and splits packets into multiple network paths. Besides, the application-layer encryption uses the Rivest–Shamir–Adleman (RSA) algorithm whose key length varying between 512 and 1152 b. Fig. 8 depicts the experimental results. Particularly, the lines marked with $+$, $*$,

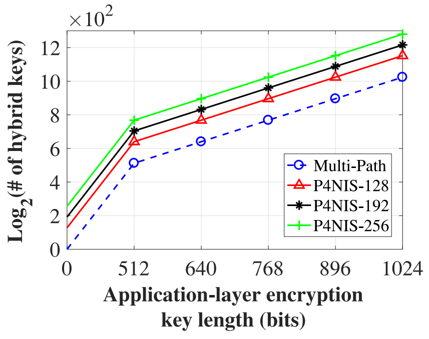

Fig. 8. Cryptographic complexity.

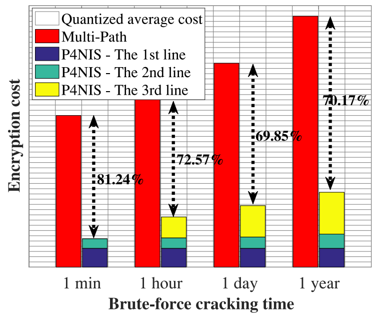

Fig. 9. Encryption cost.

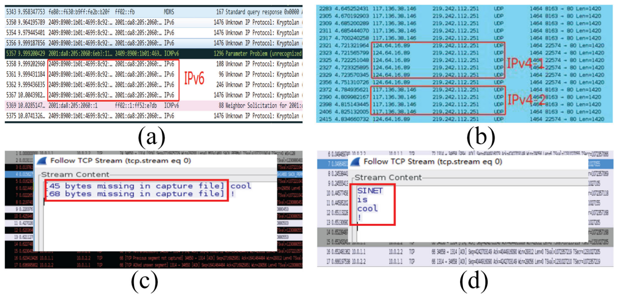
IPv6
IPv4-1
IPv4-2
Follow TCP Stream (tcp.stream eq 0)
Stream Content
[45 bytes missing in capture file] cool
[68 bytes missing in capture file] !
SINET
is
cool
!
(a)
(b)
(c)
(d)

Fig. 10. Snapshots of eavesdropping results. (a) Eavesdropping on one path. (b) Eavesdropping on multiple paths. (c) Eavesdropping on all available paths. (d) Eavesdropping with decryption keys.

and △ are P4NIS mechanisms whose second line of defense has 128, 192, and 256-b encryption keys, respectively. It is obvious that P4NIS has a larger key size than Multipath. For example, when the application-layer encryption key length is 1024 b, the Multipath and P4NIS have $2^{1024}$, $2^{1152}$, $2^{1216}$, and $2^{1280}$ alternative encryption keys, separately. This is because the second line of P4NIS has 128, 192, and 256-b encryption keys to encrypt packets in the transport layer.

*4) Encryption Cost:* To further compare P4NIS with the multipath mechanism, we theoretically evaluate the encryption cost for each mechanism. The result is shown in Fig. 9. Particularly, the cell colored with gray line depicts the quantized encryption cost, i.e., the average time when encrypting a 10-B plaintext using a 10-b key (denoted as $\tau$ s/grid). We assume that encrypting other length of plaintext using other length of key is linear to the quantized average cost. Besides, we use hardware acceleration devices such as FPGA to perform the cracking computation. The FPGA, which has 200–300 MHz operating frequency, costs 30–40 clock cycles for decryption [53]. That is, the device could perform $\text{avg}(2 * 10^8, 3 * 10^8)/\text{avg}(30, 40) \approx 2^{22.768}$ times of cracking computation in 1 s. To make eavesdroppers take 1 h to brute-force search the correct key, the multipath mechanism should encrypt all packet payloads in application layer with a 35-b key. P4NIS should have 6, 25, and 4-b keys for each line of defense, separately. In terms of the encryption cost, it is obvious that P4NIS costs less than multipath mechanism. This is because multipath mechanism has a longer application-layer key to encrypt a packet payload, and the packet payload size is much larger than the size of its transport header. Moreover, if P4NIS and multipath mechanisms have the same level of defending eavesdropping, i.e., the same brute-force cracking time taken by eavesdroppers, P4NIS decreases the encryption cost by 69.85%–81.24% (as shown in Fig. 9) compared with multipath mechanism in terms of the cracking time from 1 h to 1 year.

## C. Experimental Results of Throughput Performance

*1) Transmission Throughput:* We also evaluate the transmission throughput using Linux iperf tool. Particularly, we connect a laptop to the P4NIS client and run an iperf client on that laptop to emit traffic streams. Besides, we rent a Huawei cloud compute engine to run an iperf server whose maximum bandwidth is 100 Mb/s. Fig. 14 depicts the experimental results. Obviously, the single-path mechanism has the lowest TCP throughput whose average value is 2.57 Mb/s. The multipath mechanism has a 3.22-Mb/s average throughput, which is lower than P4NIS mechanism whose average throughput is 4.24 Mb/s. Besides, the single-path mechanism has a fluctuant throughput range from 1 to 8 Mb/s. The multipath and P4NIS have a stable throughput whose value is ~3 Mb/s and ~4 Mb/s, separately. In other words, P4NIS increases TCP throughput by 31.68% compared with multipath mechanism. However, P4NIS does not effectively aggregate the bandwidths of multiple network paths because its throughput is ideally four times larger than the Single-Path mechanism. P4NIS does not reach the expected throughput due to the out-of-order problem which is introduced in the next experiment. It is worth mentioning that the first line of P4NIS costs computing resources to schedule traffic packets on different network paths. Besides, the second line also costs computing resources to encrypt transmission ports. However, the computing resources are lightweight and affordable for the P4NIS client and server.

*2) Video Streaming Watching:* To further analyze the system performance in detail, we examine the P4NIS performance for video streaming watching. Similar to the previous experiment, a laptop is connected to the P4NIS client and plays an online video from YouTube. During the experiment, the Wireshark tool is used to sniff all the traffic packets transmitted through the laptop network interface and count the number of TCP retransmission and out-of-order packets. The experimental results are shown in Figs. 11–13. In terms of all TCP packets, it is obvious that P4NIS has the largest capacity

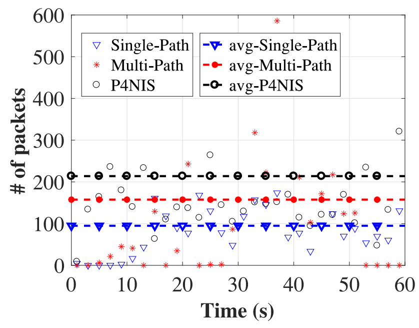

Fig. 11. Video streaming watching (all packets).

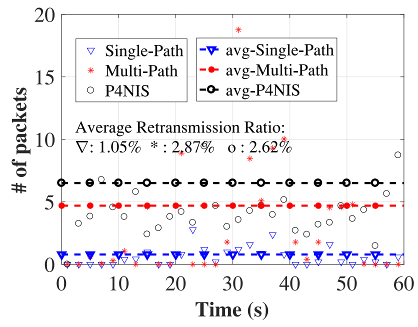

Fig. 12. Video streaming watching (retransmission packets).

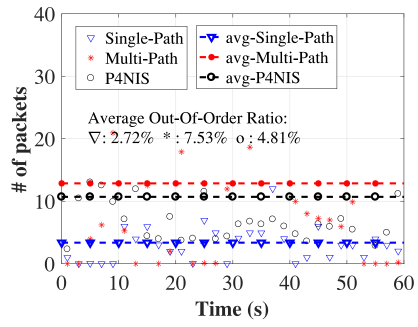

Fig. 13. Video streaming watching (out-of-order packets).

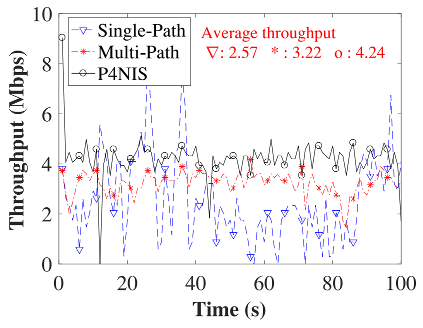

Fig. 14. Transmission throughput.

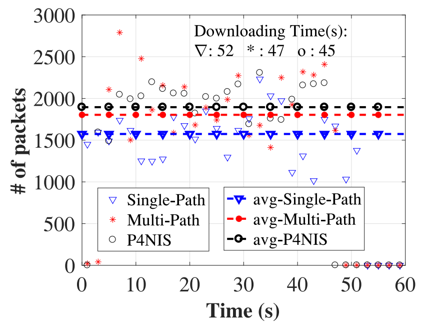

Fig. 15. Video file downloading (all packets).

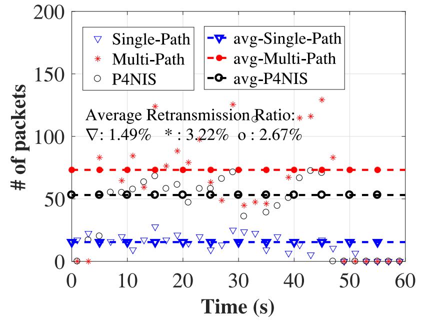

Fig. 16. Video file downloading (retransmission packets).

(on average ∼200 pps as shown in Fig. 11) to forward all the traffic packets, while the multipath mechanism has a larger capacity (on average ∼160 pps as shown in Fig. 11) than the single-path mechanism (on average ∼100 pps).

In terms of TCP retransmission packets, the multipath mechanism has the worst performance than the other two mechanisms. Concretely, P4NIS mechanism has around seven retransmission packets on average, which is much bigger than the other two mechanisms (∼5 and ∼1, respectively). This is because P4NIS and multipath mechanisms have various IPv4 and IPv6 forwarding paths whose link characteristics are greatly different from each other. In other words, the multipath mechanism has a larger retransmission ratio of 2.87% as shown in Fig. 12 due to the greatly difference between IPv4 and IPv6 paths. Besides, the P4NIS mechanism has a retransmission ratio of 2.62% due to the difference of wireless 4G links. The single-path mechanism has a retransmission ratio of 1.05% because it only forwards the traffic packets through one path.

In terms of the out-of-order packets, the multipath mechanism has the largest number of out-of-order packets whose average value is about 12, while P4NIS and single-path mechanisms have ∼10 and ∼3 out-of-order packets, respectively. That is, P4NIS has a out-of-order ratio of 4.81% which is larger than the Single-Path mechanism (2.72% as shown in Fig. 13). Besides, the multipath mechanism has the largest out-of-order ratio (7.53% as shown in Fig. 13) compared with other two mechanisms, because it has two wireless 4G links with greatly difference of link characteristics. Compared with multipath mechanism, P4NIS decreases the TCP out-of-order packets by 36.1% due to its sorting methods. Therefore, P4NIS is powerful in terms of bandwidth capacity and security, but weak in terms of retransmission and out-of-order ratio.

*3) Video File Downloading:* To further examine the influence of the aforementioned retransmission and out-of-order weakness, we use a laptop connected to the P4NIS client to download a YouTube video file whose size is approximately 42.1 MB, and Figs. 15–17 depict the experimental results. Particularly, P4NIS has the largest downloading bandwidth

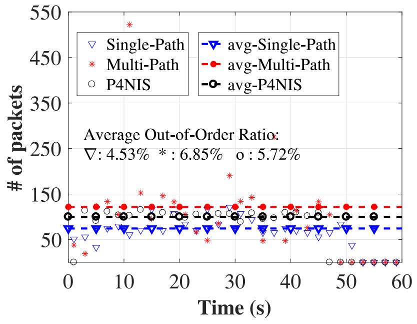

Fig. 17. Video file downloading (out-of-order packets).

capacity which is around 1900 packets per second. While the multipath and single-path mechanisms have ∼1800 and ∼1600 (pps) downloading bandwidth capacity, respectively. Besides, the experimental results show that P4NIS has the least downloading time which is 45 s, while the multipath and single-path mechanism take 47 and 52 s to download the video file, separately. It is worth mentioning that the throughput of P4NIS and multipath mechanisms is not multiple larger than single-path mechanism, this is because both P4NIS and multipath mechanisms suffer from a higher retransmission ratio and out-of-order ratio.

Regarding the retransmission packets, multipath mechanism has around 72 retransmission packets, which is the largest one compared with other two mechanisms (∼50 and ∼13 as shown in Fig. 16). Moreover, the retransmission ratio of P4NIS is 2.67%, around two times bigger than single-path mechanism (1.49% as shown in Fig. 16), and a little smaller than the multipath mechanism (3.22% as shown in Fig. 16) due to the sorting method of P4NIS. Regarding the out-of-order packets, as shown in Fig. 17, the multipath mechanism has approximately 128 out-of-order packets, which is the largest compared with other two mechanisms. Besides, the out-of-order ratio of P4NIS is 5.72%, which is a little smaller than multipath mechanism. The out-of-order ratio of P4NIS is bigger than single-path mechanism whose value is 4.53%.

## VII. Conclusion

In this article, we have proposed a programmable network immune scheme (named P4NIS), which is equipped with three lines of defenses to prevent the eavesdropping attacks. In case eavesdroppers crack the fixed forwarding policy, operators could dynamically update different forwarding policies in the first line. Besides, in case eavesdroppers crack the fixed encryption algorithm, operators could also dynamically change transport encryption algorithms in the second line. We implemented the proposed P4NIS and evaluate its performance. Experimental results showed that P4NIS can increase difficulties of eavesdropping significantly, increase transmission throughput by 31.68% compared with state-of-the-art mechanisms. Moreover, if P4NIS and state-of-the-art mechanisms have the same level of defending eavesdropping, P4NIS can decrease the encryption cost by 69.85%–81.24%.

In future work, we will further investigate both the sending and receiving scheduler mechanism to decrease the number of retransmission and out-of-order packets. Besides, since different traffics have different security requirements (e.g., delay-sensitive traffics may not need a high network immunity), we will design the forwarding policies and transport encryption algorithms with diverse security levels to meet various traffic requirements. Moreover, due to the throughput limitation of P4 software switch, we will implement P4NIS using high performance hardware devices.

## Acknowledgment

The authors would like to thank the reviewers for their valuable comments.

## References

[1] W. Iqbal, H. Abbas, M. Daneshmand, B. Rauf, and Y. A. Bangash, "An in-depth analysis of IoT security requirements, challenges, and their countermeasures via software-defined security," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10250–10276, Oct. 2020.

[2] R. Li, T. Song, N. Capurso, J. Yu, J. Couture, and X. Cheng, "IoT applications on secure smart shopping system," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1945–1954, Dec. 2017.

[3] N. Moustafa, B. Turnbull, and K.-K. R. Choo, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4815–4830, Jun. 2019.

[4] J. Gu, J. Wang, Z. Yu, and K. Shen, "Traffic-based side-channel attack in video streaming," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 972–985, Jun. 2019.

[5] A. Karati, C.-I. Fan, and R.-H. Hsu, "Provably secure and generalized signcryption with public verifiability for secure data transmission between resource-constrained IoT devices," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10431–10440, Dec. 2019.

[6] C. Lai, R. Lu, D. Zheng, and X. Shen, "Security and privacy challenges in 5G-enabled vehicular networks," *IEEE Netw.*, vol. 34, no. 2, pp. 37–45, Mar./Apr. 2020.

[7] A. Yang, J. Weng, N. Cheng, J. Ni, X. Lin, and X. Shen, "DeQoS attack: Degrading quality of service in VANETs and its mitigation," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4834–4845, May 2019.

[8] J. Xia *et al.*, "Secure cache-aided multi-relay networks in the presence of multiple eavesdroppers," *IEEE Trans. Commun.*, vol. 67, no. 11, pp. 7672–7685, Nov. 2019.

[9] X. Yuan *et al.*, "Enabling secure and efficient video delivery through encrypted in-network caching," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 8, pp. 2077–2090, Aug. 2016.

[10] G. Ouvrier, M. Laterman, M. Arlitt, and N. Carlsson, "Characterizing the HTTPS trust landscape: A passive view from the edge," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 36–42, Jul. 2017.

[11] Q. Jiang, J. Ni, J. Ma, L. Yang, and X. Shen, "Integrated authentication and key agreement framework for vehicular cloud computing," *IEEE Netw.*, vol. 32, no. 3, pp. 28–35, May/Jun. 2018.

[12] S.-Y. Tan, K.-W. Yeow, and S. O. Hwang, "Enhancement of a lightweight attribute-based encryption scheme for the Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6384–6395, Aug. 2019.

[13] T. V. X. Phuong, R. Ning, C. Xin, and H. Wu, "Puncturable attribute-based encryption for secure data delivery in Internet of Things," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Honolulu, HI, USA, Apr. 2018, pp. 1511–1519.

[14] M. Ma, D. He, N. Kumar, K.-K. R. Choo, and J. Chen, "Certificateless searchable public key encryption scheme for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 2, pp. 759–767, Feb. 2018.

[15] K. Muhammad, R. Hamza, J. Ahmad, J. Lloret, H. Wang, and S. W. Baik, "Secure surveillance framework for IoT systems using probabilistic image encryption," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3679–3689, Aug. 2018.

[16] Y. Mirsky, N. Kalbo, Y. Elovici, and A. Shabtai, "Vesper: Using echo analysis to detect man-in-the-middle attacks in LANs," *IEEE Trans. Inf. Forensics Security*, vol. 14, pp. 1638–1653, 2019.

[17] G. Oliva, S. Cioabă, and C. N. Hadjicostis, "Distributed calculation of edge-disjoint spanning trees for robustifying distributed algorithms against man-in-the-middle attacks," *IEEE Trans. Control Netw. Syst.*, vol. 5, no. 4, pp. 1646–1656, Dec. 2018.

[18] A. Esfahani *et al.*, "An efficient Web authentication mechanism preventing man-in-the-middle attacks in industry 4.0 supply chain," *IEEE Access*, vol. 7, pp. 58981–58989, 2019.

[19] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2027–2051, 3rd Quart., 2016.

[20] K. Sakai, M.-T. Sun, W.-S. Ku, J. Wu, and T. H. Lai, "Secure data communications in wireless networks using multi-path avoidance routing," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4753–4767, Oct. 2019.

[21] T. M. Hoang, H. Q. Ngo, T. Q. Duong, H. D. Tuan, and A. Marshall, "Cell-free massive MIMO networks: Optimal power control against active eavesdropping," *IEEE Trans. Commun.*, vol. 66, no. 10, pp. 4724–4737, Oct. 2018.

[22] D. Raposo, M. L. Pardal, L. Rodrigues, and M. Correia, "MACHETE: Multi-path communication for security," in *Proc. IEEE 15th Int. Symp. Netw. Comput. Appl. (NCA)*, Cambridge, MA, USA, Oct. 2016, pp. 60–67.

[23] M. Jadin, G. Tihon, O. Pereira, and O. Bonaventure, "Securing multipath TCP: Design & implementation," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Atlanta, GA, USA, May 2017, pp. 1–9.

[24] C. Pearce and S. Zeadally, "Ancillary impacts of multipath TCP on current and future network security," *IEEE Internet Comput.*, vol. 19, no. 5, pp. 58–65, Sep./Oct. 2015.

[25] H. Fang, L. Xu, Y. Zou, X. Wang, and K.-K. R. Choo, "Three-stage Stackelberg game for defending against full-duplex active eavesdropping attacks in cooperative communication," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10788–10799, Nov. 2018.

[26] G. Liu, W. Quan, N. Cheng, N. Lu, H. Zhang, and X. Shen, "P4NIS: Improving network immunity against eavesdropping with programmable data planes," in *Proc. IEEE INFOCOM Conf. Comput. Commun.Workshops (INFOCOM WKSHPS)*, Toronto, ON, Canada, 2020, pp. 91–96, doi: 10.1109/INFOCOMWKSHPS50562.2020.9162975.

[27] M. P. Howarth, S. Iyengar, Z. Sun, and H. Cruickshank, "Dynamics of key management in secure satellite multicast," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 2, pp. 308–319, Feb. 2004.

[28] T. Yao, K. Fukui, J. Nakashima, and T. Nakai, "Initial common secret key sharing using random plaintexts for short-range wireless communications," *IEEE Trans. Consum. Electron.*, vol. 55, no. 4, pp. 2025–2033, Nov. 2009.

[29] H. Jeon, J. Choi, S. W. McLaughlin, and J. Ha, "Channel aware encryption and decision fusion for wireless sensor networks," *IEEE Trans. Inf. Forensics Security*, vol. 8, pp. 619–625, 2013.

[30] J. Xu and B. Chen, "Secure coding over networks against noncooperative eavesdropping," *IEEE Trans. Inf. Theory*, vol. 59, no. 7, pp. 4498–4509, Jul. 2013.

[31] R. Tennekoon, J. Wijekoon, and H. Nishi, "On the effectiveness of IP-routable entire-packet encryption service over public networks (november 2018)," *IEEE Access*, vol. 6, pp. 73170–73179, 2018.

[32] J. Song, X. Yi, and X. Zhang, "Enhancing IPsec performance and security against eavesdropping," U.S. Patent 9 021 577, Apr. 2015.

[33] M. Liyanage, A. Braeken, A. D. Jurcut, M. Ylianttila, and A. Gurtov, "Secure communication channel architecture for software defined mobile networks," *Comput. Netw.*, vol. 114, pp. 32–50, Feb. 2017.

[34] L. Wang, H. Kim, P. Mittal, and J. Rexford, "Programmable in-network obfuscation of traffic," 2020. [Online]. Available: arXiv:2006.00097.

[35] T. Datta, N. Feamster, J. Rexford, and L. Wang, "SPINE: Surveillance protection in the network elements," in *Proc. 9th USENIX Workshop Free Open Commun. Internet (FOCI)* Aug. 2019, pp. 1–7.

[36] H. Zhang, W. Quan, H.-C. Chao, and C. Qiao, "Smart identifier network: A collaborative architecture for the future Internet," *IEEE Netw.*, vol. 30, no. 3, pp. 46–51, May/Jun. 2016.

[37] L. Zhang *et al.*, "Named data networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 66–73, 2014.

[38] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "The locator/ID separation protocol (LISP)," IETF, RFC 6830, 2013.

[39] G. Liu, W. Quan, N. Cheng, H. Zhang, and X. Shen, "VLI: Variable-length identifier for interconnecting heterogeneous IoT networks," *IEEE Wireless Commun. Lett.*, vol. 9, no. 8, pp. 1146–1149, Aug. 2020.

[40] J. McCauley, Y. Harchol, A. Panda, B. Raghavan, and S. Shenker, "Enabling a permanent revolution in Internet architecture," in *Proc. ACM Special Interest Group Data Commun. (SIGCOMM)*, 2019, pp. 1–14.

[41] W. Quan, N. Cheng, M. Qin, H. Zhang, H. A. Chan, and X. Shen, "Adaptive transmission control for software defined vehicular networks," *IEEE Wireless Commun. Lett.*, vol. 8, no. 3, pp. 653–656, Jun. 2019.

[42] S. Zhu, J. Bi, C. Sun, C. Wu, and H. Hu, "SDPA: Enhancing stateful forwarding for software-defined networking," in *Proc. IEEE 23rd Int. Conf. Netw. Protocols (ICNP)*, San Francisco, CA, USA, Nov. 2015, pp. 323–333.

[43] G. Liu, W. Quan, N. Cheng, H. Zhang, and S. Yu, "Efficient DDoS attacks mitigation for stateful forwarding in Internet of Things," *J. Netw. Comput. Appl.*, vol. 130, pp. 1–13, Mar. 2019.

[44] N. McKeown, "Software-defined networking," *INFOCOM Keynote Talk*, vol. 17, no. 2, pp. 30–32, 2009.

[45] X.-N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, "Rules placement problem in OpenFlow networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1273–1286, 2nd Quart., 2016.

[46] W. Li, W. Meng, and L. F. Kwok, "A survey on OpenFlow-based software defined networks: Security challenges and countermeasures," *J. Netw. Comput. Appl.*, vol. 68, pp. 126–139, Jun. 2016.

[47] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014.

[48] M. Shahbaz *et al.*, "Pisces: A programmable, protocol-independent software switch," in *Proc. ACM SIGCOMM Conf.*, 2016, pp. 525–538.

[49] S. Signorello, R. State, J. François, and O. Festor, "NDN.p4: Programming information-centric data-planes," in *Proc. IEEE NetSoft Conf. Workshops*, Seoul, South Korea, 2016, pp. 384–389.

[50] S. Y. Bonde and U. S. Bhadade, "Analysis of encryption algorithms (RSA, SRNN and 2 key pair) for information security," in *Proc. Int. Conf. Comput. Commun. Control Autom. (ICCUBEA)*, Pune, India, 2017, pp. 1–5.

[51] I. G. Amalarethinam and H. M. Leena, "Enhanced RSA algorithm with varying key sizes for data security in cloud," in *Proc. World Congr. Comput. Commun. Technol. (WCCCT)*, Tiruchirappalli, India, 2017, pp. 172–175.

[52] S. Balakrishnan, P. Wang, A. Bhuyan, and Z. Sun, "On success probability of eavesdropping attack in 802.11ad mmWave WLAN," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Kansas City, MO, USA, 2018, pp. 1–6.

[53] N. S. S. Srinivas and M. Akramuddin, "FPGA based hardware implementation of AES Rijndael algorithm for encryption and decryption," in *Proc. Int. Conf. Elect. Electron. Optim. Techn. (ICEEOT)*, Chennai, India, 2016, pp. 1769–1776.

**Gang Liu** (Student Member, IEEE) received the B.E. degree from Tiangong University, Tianjin, China, in 2016. He is currently pursuing the Ph.D. degree with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China.

His current research interests include information-centric networking, software-defined networking, network function virtualisation, programmable network language, and stateful forwarding.

**Wei Quan** (Member, IEEE) received the Ph.D. degree in communication and information system from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2014.

He is currently an Associate Professor with the School of Electronic and Information Engineering, BJTU. He has published more than 20 papers in prestigious international journals and conferences, including *IEEE Communications Magazine*, IEEE WIRELESS COMMUNICATIONS, IEEE NETWORK, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE COMMUNICATIONS LETTERS, IFIP Networking, IEEE ICC, and IEEE GLOBECOM. His research interests include key technologies for network analytics, future Internet, 5G networks, and vehicular networks.

Dr. Quan is a TPC Member of IEEE ICC in 2017 and 2018, IEEE INFOCOM (NewIP Workshop) in 2020, ACM MOBIMEDIA in 2015, 2016, and 2017, and IEEE CCIS in 2015 and 2016. He serves as an Associate Editor for *Journal of Internet Technology*, *Peer-to-Peer Networking and Applications*, and IEEE ACCESS, and as a technical reviewer for many important international journals. He is also a member of ACM and a Senior Member of the Chinese Association of Artificial Intelligence.

**Nan Cheng** (Member, IEEE) received the B.E. and M.S. degrees from the Department of Electronics and Information Engineering, Tongji University, Shanghai, China, in 2009 and 2012, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, in 2016.

He is currently a Professor with State Key Laboratory, ISN, and with the School of Telecommunication Engineering, Xidian University, Shaanxi, China. He worked as a Postdoctoral Fellow with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, from 2017 to 2018. His current research focuses on space-air-ground integrated system, network big data, and 6G networking technologies. His research interests also include performance analysis, MAC, opportunistic communication, and application of AI for vehicular networks.

**Deyun Gao** (Senior Member, IEEE) received the B.E. and M.E. degrees in electrical engineering and the Ph.D. degree in computer science from Tianjin University, Tianjin, China, in 1994, 1999, and 2002, respectively.

He spent one year as a Research Associate with the Department of Electrical and Electronic Engineering, Hong Kong University of Science and Technology, Hong Kong. Then, he spent three years as a Research Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore. In 2007, he joined the School of Electronics and Information Engineering, Faculty of Beijing Jiaotong University, Beijing, China, as an Associate Professor and was promoted to a Full Professor, in 2012. In 2014, he was a Visiting Scholar with the University of California at Berkeley, Berkeley, CA, USA. His research interests include Internet of Things, vehicular networks, and next-generation Internet.

**Ning Lu** (Member, IEEE) received the B.E. and M.E. degrees in electrical engineering from Tongji University, Shanghai, China, in 2007 and 2010, respectively, and the Ph.D. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 2015.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Queen's University, Belfast, U.K. Prior to joining Queen's University, he was an Assistant Professor with the Department of Computing Science, Thompson Rivers University, Kamloops, BC, Canada. From 2015 to 2016, he was a Postdoctoral Fellow with the Coordinated Science Laboratory, University of Illinois at Urbana–Champaign, Champaign, IL, USA. He also spent the summer of 2009 as an intern with the National Institute of Informatics, Tokyo, Japan. He has published more than 40 papers in top IEEE journals and conferences, including IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, ACM MobiHoc, and IEEE INFOCOM. His current research interests include real-time scheduling, distributed algorithms, and reinforcement learning for wireless communication networks.

**Hongke Zhang** (Fellow, IEEE) received the M.S. and Ph.D. degrees in electrical and communication systems from the University of Electronic Science and Technology of China, Chengdu, China, in 1988 and 1992, respectively.

From 1992 to 1994, he was a Postdoctoral with Beijing Jiaotong University (BJTU), Beijing, China. He is currently a Professor with the School of Electronic and Information Engineering, BJTU, where he is the Director of a National Engineering Lab on Next Generation Internet Technologies. He is also with the PCL Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen, China. He has authored of more than ten books and the holder of more than 70 patents. His research has resulted in many papers, books, patents, systems, and equipment in the areas of communications and computer networks.

Dr. Zhang is the Chief Scientist of a National Basic Research Program of China (973 Program) and has also served on the editorial board of several international journals.

**Xuemin (Sherman) Shen** (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990.

He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular *ad hoc* and sensor networks.

Dr. Shen received the R. A. Fessenden Award in 2019 from IEEE, Canada, the Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) in 2019, the James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, the Joseph LoCicero Award in 2015, the Education Award in 2017 from the IEEE Communications Society, the Technical Recognition Award from Wireless Communications Technical Committee in 2019, and the AHSN Technical Committee in 2013. He has also received the Excellent Graduate Supervision Award in 2006 from the University of Waterloo, and the Premiers Research Excellence Award in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee Chair/Co-Chair for IEEE Globecom'16, IEEE Infocom'14, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He was the Editor-in-Chief of IEEE INTERNET OF THINGS JOURNAL, IEEE NETWORK, and *IET Communications*. He was the elected IEEE Communications Society Vice President for Technical & Educational Activities, a Vice President for Publications, Member-at-Large on the Board of Governors, the Chair of the Distinguished Lecturer Selection Committee, a member of IEEE ComSoc Fellow Selection Committee. He is a registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Member, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society.