Enabling Secure and Versatile Packet Inspection With Probable Cause Privacy for Outsourced Middlebox

Hao Ren, *Student Member, IEEE*, Hongwei Li, *Senior Member, IEEE*, Dongxiao Liu, *Member, IEEE*, Guowen Xu, *Student Member, IEEE*, and Xuemin (Sherman) Shen, *Fellow, IEEE*

Abstract—Middlebox is an intermediary network equipment which can be outsourced to remote cloud servers for low-cost and customizable network services, such as load balancer and intrusion detection. A fundamental function of the middlebox is packet inspection, where both the packet header and payload are extracted and analyzed based on inspection rules. However, as the packet may contain sensitive individual or organizational information, it may raise severe privacy concerns without proper countermeasures. In this article, we propose a secure and versatile packet inspection scheme for outsourced middlebox. The proposed scheme builds upon two non-collusion cloud servers, where the first server conducts the inspection task over the ciphertext domain and the second reveal the inspection results. By doing so, the proposed scheme achieves versatile inspection functionalities: range-query-based header inspection and token-based payload inspection, while preserving the privacy of packet header, payload, and inspection rules. Moreover, we identify and address two challenging issues in the state-of-the-art literatures. First, we tailor the design of mis-operation resistant searchable homomorphic encryption (MR-SHE) and somewhat homomorphic encryption in the two-server model, to resist *offline dictionary attack on payload headers*. Second, we propose a key management mechanism with compelled access for the middlebox, to achieve *fine-grained probable cause privacy*. We also conduct extensive experiments and compare the results with existing schemes to demonstrate the feasibility of the proposed scheme.

Index Terms—Cloud computing, middlebox, privacy-preserving, homomorphic encryption

1 Introduction

Middlebox [2] is an intermediary network equipment. It is widely deployed in modern computer networks to provide a wide range of network services, such as firewalls and intrusion detections. As the network traffic is segmented into packets and sent from a source to a destination, packet inspection, that pries into both packet headers and payloads, serves as a fundamental function of the middlebox. First, the header inspection [3], [4], [5] aims to determine the legitimacy of the packet source and destination. For example, a firewall may ban IP addresses ranging from 192.168.1.1 to 192.168.1.100. Second, the payload inspection filters out malicious packets, e.g., a terrorist speech and Denial of Service (DoS) attacks [6], [7], [8], and intercepts sensitive information at the gateways of private networks.

With the ever-increasing of network traffic, the deployment and maintenance costs of the middlebox increase significantly for enterprise-level network managements. As a result, outsourcing the network middlebox to public cloud servers [2], [9] becomes a prevalent choice among enterprises, to enjoy low-cost, easy-to-implement and customized network services. However, the cloud-based service paradigm may raise severe privacy concerns on the packet inspection function, since a large amount of packets containing personal or commercial data are directed to the cloud servers [2], [10]. First, a packet header may reveal the location or affiliation of the sender [11] (e.g., a staff in an enterprise who sends packets to the external web sites). The whole enterprise network topology can even be recovered by analyzing large-scale IP addresses of the passing packets [4]. Second, the packet payload could contain sensitive personal or commercial information, such as photos and business contracts. As a result, it becomes an urgent requirement of the packet sender to preserve privacy of both the packet header and payload in the outsourced middlebox [10]. Third, the packet inspection rules should also be concealed since the customized inspection configurations directly indicate security policies of the enterprise networks. Once the inspection rules are grabbed by attackers, it will be much easier to evade the security functions of the outsourced middlebox.

A straightforward method to protect the packet privacy (header and payload) and inspection rule is to encrypt them

- *Hao Ren and Hongwei Li are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China, and also with the Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518000, China. E-mail: renhao.uestc@gmail.com, hongweili@uestc.edu.cn.*
- *Dongxiao Liu and Xuemin (Sherman) Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, Ontario N2L 3G1, Canada. E-mail: {dongxiao.liu, sshen}@uwaterloo.ca.*
- *Guowen Xu is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China. E-mail: guowen.xu@foxmail.com.*

Manuscript received 11 Mar. 2020; revised 26 Oct. 2020; accepted 8 Feb. 2021. Date of publication 12 Feb. 2021; date of current version 6 Dec. 2022.
(Corresponding author: Dongxiao Liu.)
Recommended for acceptance by Y. Yang.
Digital Object Identifier no. 10.1109/TCC.2021.3059026

using conventional crypto systems (e.g., AES, RSA, etc.). Unfortunately, significant technical challenges are introduced to support the same packet inspection functions over the encrypted packet contents. First, it becomes challenging for the outsourced middlebox to check the range of IP addresses in the ciphertext domain. Second, it is also a non-trivial task to support the richness of expressive inspection rules over the ciphertext, such as payload keyword filtering. In response to the challenges, several approaches are proposed recently. The first secure packet inspection scheme named as BlindBox [6] is proposed by Sherry *et al*. In BlindBox, two connections are built between the packet sender and packet receiver. One is used for secure inspection over the encrypted payload and the other is the traditional TLS/SSL connection. Meanwhile, the inspection rules are also obfuscated before being deployed on the middlebox. Shortly afterward, Yuan *et al*. [7] propose a secure and efficient packet inspection scheme for cloud-assisted middlebox. Only fast symmetric encryption and Cuckoo hashing [12] are used to encrypt the packet payload and the inspection rules. Canard *et al*. [13] and Fan *et al*. [14] explore public key based approaches for advanced functionalities including inspection result verification and malware detection. In Embark [3], the first secure header inspection scheme is proposed. A new symmetric key encryption scheme *PrefixMatch* is presented for secure and fast header inspection. In [4], [5], Guo *et al*. point out that *PrefixMatch* needs to use prefixes to represent the numerical ranges, which increases the number of rules. Thus, they utilize order-revealing encryption [15] (ORE) to support encrypted range detection [16].

While existing works achieve rich functionalities and performance gains for the secure packet inspection, some important and challenging issues still receive insufficient attention. 1) Dictionary attack on the packet header. The message space of header domains is usually small. For example, the range of TCP port number is $[0, 65535]$. As a result, it is easy for an attacker to conduct dictionary attack to the encrypted port number. Moreover, with some background information such as the usage frequency of each port, the attacker can easily infer the real port number from the encrypted network traffic. 2) Fine-grained probable cause privacy. It is time-consuming to conduct advanced payload inspection such as machine learning based malware detection over ciphertext domain [14], [17]. A new paradigm named as probable cause privacy [6] can mitigate such issue. Specifically, once strong evidences of malicious contents in the payload inspection over the encrypted network traffic are captured, the middlebox should be authorized to inspect the payload over plaintext domain. However, the contents of the regular packets should not be exposed to the middlebox [6]. Meanwhile, it is also risky to use the same secret key [6] for all packets. As a result, how to preserve the privacy of the regular packets for probable cause privacy remains a challenging problem.

In this paper, we propose a Secure and Versatile Packet Inspection (SVPI) scheme with probable cause privacy for outsourced middlebox. The middlebox is deployed over two non-collusion cloud servers [4], [18], where one server carries the most computational inspection task and the other reveals the final inspection results. SVPI supports versatile inspection functionalities: the range-query based header inspection and token-based payload inspection. Specifically, the contributions of this paper are as follows:

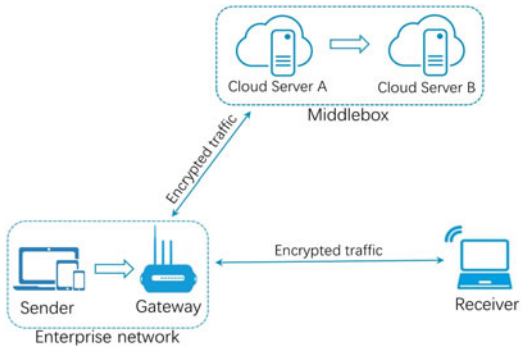


Fig. 1. System model of outsourced middlebox.

- We tailor the designs of mis-operation resistant searchable homomorphic encryption (MR-SHE) scheme [19] and somewhat homomorphic encryption [20] in the two-server model to resist offline dictionary attack on the packet header.
- The privacy of inspection rules, packet header and the packet payload are all well preserved without sacrificing inspection functionalities. Moreover, a new key management method is designed to support fine-grained probable cause privacy without breaching the confidentiality of regular packets.
- Extensive experiments are conducted with a real-world intrusion detection rule set. We compare the experimental results of SVPI with the existing schemes [4], [13] to demonstrate the efficiency of SVPI in terms of computation and communication overheads.

The remainder of this paper is organized as follows. In Section 2, the system, threat models and privacy requirements are described. In Section 3, we review some building blocks used in this paper. In Section 4, we show the details of SVPI. The security analysis and the performance evaluation are provided in Sections 5 and 6, respectively. Related works are reviewed in Section 7. Section 8 concludes the paper.

2 Models and Privacy Requirements

In this section, we first present the system model to show the role of each entity, which captures the most typical outsourced middlebox scenarios [4], [6]. We then give an example of the usage scenario and describe the work flow to show how a packet is sent from the sender to the receiver. Under such a system model, the threat model is defined and the privacy requirements of sensitive contents are given.

2.1 System Model

As shown in Fig. 1, the system model consists of four entities: Sender (S), Gateway (GW), Middlebox (MB) and Receiver (R).

- S: In an enterprise network, any authorized end user could be a packet sender (S). For instance, S could be a staff in a company who wants to send a commercial contract to the customer (i.e., packet receiver).

- GW: GW is the administrator of the enterprise network that offers system initialization and key management services. GW first prepares the encrypted rules for header and payload inspection and then outsources them to the cloud (i.e., MB). Then, it generates private and public keys and distributes them to S and MB. GW should be the only outlet of the entire enterprise network.
- MB: MB is deployed on two cloud servers denoted as A and B. A stores the encrypted rules of header and payload inspection, and carries the most computation tasks. B reveals the final detection result of each rule. If any rule is asserted to be matched, MB should trigger the pre-defined actions, such as sending an alert to GW or even cutting off the connection. Otherwise, MB will let the GW send the packet to R.
- R: Upon receiving the encrypted packets, it first decrypts them to obtain the packet contents. Then, R generates the verification objects and sends them to MB to verify whether S encrypts the packet payload correctly.

Discussion. In this paper, we use the classical two-server model [5] to instantiate MB with the following benefits:

- The first benefit is reducing the communication round between the cloud server and the packet sender. Since homomorphic encryption [19], [20] cannot directly support comparison over ciphertext, the cloud server must interact with the packet sender to reveal (decrypt) the final inspection result. When the number of senders reaches a certain threshold, the communication costs can be expensive. In a two-server setting, we can eliminate the communication between senders and the server by directly revealing inspection result at the server side.
- Since the decryption of the final inspection result is carried by the cloud server B, the computational overhead for the packet sender is significantly reduced, especially for the resource-limited mobile users.

2.2 Usage Scenario

Suppose Alice is an employee in a company. Alice can use the company network to send personal messages, such as social media messages. In this case, the company would like to ensure that no sensitive information is sent out while Alice uses the company network. Alice can also sometimes try to deliberately send sensitive information from internal networks to an external entity Bob in a more convert way. To resist such risks, *all the packets* sent from the internal company network should be inspected by the middlebox. As the middlebox is implemented on the public cloud servers, Alice may feel uncomfortable for privacy concerns since her personal information is also inspected. For the first case, Alice can apply the traditional end-to-end encryption protocol (e.g., SSL/TLS) to hide the personal information. Meanwhile, the packet header and payload are also encrypted using homomorphic encryption to support header and payload inspection without disclosing the original contents of the packets. Thus, Alice has enjoyed the security function offered by the outsourced middlebox without compromising her privacy. For the second case, the company can install encryption modules on company devices to correctly encrypt packet header and payload for remote inspections. Similarly, Alice could also be a student in a university or a staff in the security agencies of government. In practice, Alice may attempt to send malware or fraud information to Bob [14]. Bob can verify the encrypted tokens sent from Alice for security concern [7].

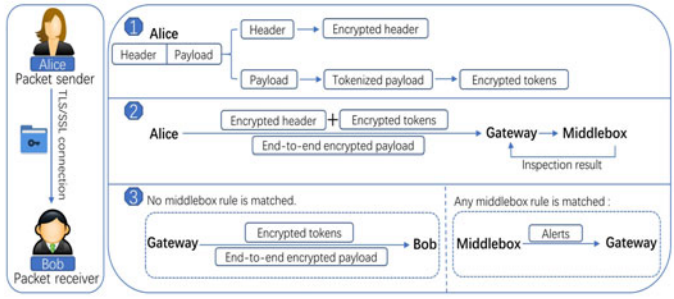


Fig. 2. Work flow of middlebox services.

2.3 Work Flow

As shown in Fig. 2, if Alice (S) wants to send a packet to Bob (R). She first builds a TLS/SSL connection with Bob and uses the conventional end-to-end encryption method to conceal the packet payload. To support header and payload inspection, each packet should be processed through the following three steps. In the first step, the packet header and payload are encrypted separately using homomorphic encryption. The payload should be segmented (tokenized) into tokens before encryption. In the second step, the encrypted header and tokens (used for inspection) along with the end-to-end encrypted payload (TLS/SSL connection) are all sent to the gateway (GW). GW holds the traffic, and all these encrypted contents are forwarded to the middlebox (MB) for inspection. After inspection, MB returns the result to GW. In the third step, if no rule is matched, GW should transfer the end-to-end encrypted payload along with the encrypted tokens to Bob. Otherwise, alerts will be triggered and sent to GW. Note that, since *probable cause privacy* is supported, some advanced inspections such as machine learning based malware detection [14] can be conducted by the outsourced middlebox. Thus, the advantages of cloud computing including large storage and high performance are fully explored.

2.4 Threat Model

Following the widely applied threat model of cloud server in [7], [21], we consider the cloud server A and B to be *semi-honest* (i.e., *honest-but-curious*) [22]. Concretely, the cloud servers will strictly execute the pre-defined protocols but may be interested in the content of the passing encrypted traffic for commercial or personal benefits. A and B do not collude with each other or exchange any information that is not permitted by the protocol. In practice, the two servers can be different cloud service providers with a middlebox service agreement, which legally regulates service providers's behavior. Moreover, due to the competitive relationship between service providers, it is of high risk for them to collude and break the agreement. Only A can get access to the encrypted packet header and

payload uploaded by S and B is allowed to possess the private keys. The security properties achieved by doing so will be discussed in Section 5. GW is the owner of encrypted rules and is fully trusted in the whole system. In the reality, S is possible to be malicious and try to evade the detection system. Thus, R should verify the correctness of received encrypted tokens.

Remark. We acknowledge that the two non-collusion model needs a more careful instantiation. In the reality, the cloud server is possible to be malicious and try to collude with the other server. Then, the threat model is changed from semi-honest to malicious. To resist malicious cloud server, GW has to verify the correctness and integrity of the returned result. Then, the cloud servers and GW have to bear additional computational overheads. In this paper, we adopt non-collusion (semi-honest) model to reap better performance. For real-world deployments of this model, users can choose a CSP with recognized reputation; especially the big ones, such as Amazon, Microsoft, Google, etc. Users can also deploy their middleboxes over two different CSP. Then, the risk of collusion will be significantly decreased. By doing so, the only extra cost is the increased communication delay between two cloud servers, which can be simply addressed by increasing the budget for network bandwidth.

2.5 Privacy Requirements

In this part, we describe the privacy requirements of the sensitive contents.

- *Header privacy:* Since the packet header may reveal sensitive information, the uploaded encrypted header should not disclose the original contents of the header.
- *Payload privacy:* The payload may contain personal data that should be concealed from MB. The end-to-end encrypted payload (TLS/SSL connection) as well as the encrypted tokens should not reveal the original contents of packet payload. Especially, the encrypted tokens are another ciphertext version of the packet payload. As a result, the information leakage introduced by these encrypted tokens may also breach the payload privacy. Therefore, both two parts should be carefully designed to preserve the privacy of payload.
- *Rule set privacy:* Rule set directly reflects all the security and privacy policies of the whole system. It will suffer from intensive internal and external attacks. Consequently, the rule set privacy should be well preserved.

3 Building Blocks

3.1 Basic Cryptographic Primitives

1). A function $F : \mathcal{X} \times \mathcal{K} \rightarrow \mathcal{Y}$ is a secure pseudo-random function (PRF) if for all randomly chosen keys $k \in \mathcal{K}$ (key space), all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$ and a real random function $f(k, \cdot)$ from $\mathcal{X}$ (message space) to $\mathcal{Y}$ (output space), $|Pr[\mathcal{A}^{f(k,\cdot)} = 1 | f : \mathcal{X} \rightarrow \mathcal{Y}] - Pr[\mathcal{A}^{F(k,\cdot)} = 1 | k \leftarrow \mathcal{K}]| \leq negl(k)$, where $negl(\cdot)$ is a negligible function.

2). A symmetric crypto-system consists of three polynomial time algorithms $\Pi = (KeyGen, Enc, Dec)$. $KeyGen(\cdot)$ is the key generation algorithm that takes the security parameter λ as the input and produces a secret key K. Given a message m, it is encrypted as $c = Enc(K, m)$. To decrypt the ciphertext c, we can compute $m = Dec(K, c)$.

3.2 Boneh-Goh-Nissim Crypto-System

Boneh-Goh-Nissim crypto-system (BGN) [20] is a somewhat homomorphic encryption consisted of three algorithms.

- BGN.KeyGen(λ): $\lambda \in \mathbb{Z}^+$ is the security parameter. The tuple $(p, q, \mathbb{G}, \mathbb{G}_1, e)$ is generated and $n = pq$ is the order of cyclic groups $\mathbb{G}, \mathbb{G}_1$, where p, q are two large primes. Let $g, u \in \mathbb{G}$ and $h = u^q$. It outputs public key $PK = (n, \mathbb{G}, \mathbb{G}_1, e, g, h)$ and private key $SK = p$.
- BGN.Enc(PK, m): Given message $m \in \{0, \mathbb{Z}_T\}, T \ll q$, it computes $C = g^m h^r \in \mathbb{G}$ and outputs ciphertext C, where $r \in \mathbb{Z}_n$ is a random number.
- BGN.Dec(SK, C): To decrypt C, it calculates $C^p = (g^m h^r)^p = (g^p)^m$ using private key p. Then, the message can be recovered by computing the discrete logarithm of C^p base g^p.

Let $\{C_{m_1}, C_{m_2}, C_{m_1+m_2}, C_{m_1 \cdot m_2}\}$ be the ciphertexts of messages $\{m_1, m_2, m_1 + m_2, m_1 \cdot m_2\}$ respectively. And e is the bilinear paring: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$. Given $g_1 = e(g, g)$, $h_1 = e(g, h)$ where $h = g^{\alpha q}, \alpha \in \mathbb{Z}$. Then, BGN supports following homomorphic properties. 1). $C_{m_1} \cdot C_{m_2} = C_{m_1+m_2} \in \mathbb{G}$. 2). $e(C_{m_1}, C_{m_2}) = C_{m_1 \cdot m_2} \in \mathbb{G}_1$. Note that, in group $\mathbb{G}_1$, the first property is still supported.

3.3 MR-SHE

MR-SHE is short for *Mis-operation Resistant Searchable Homomorphic Encryption*, which is proposed by Emura *et al.* [19]. To resist the mis-operations (homomorphic operation between two different data domains) launched by cloud servers, MR-SHE requires one-time keyword matching before conducting homomorphic operations. Thus, from the aspect of functionality, MR-SHE can support keyword search and homomorphic operation over ciphertext domain simultaneously. The definition of MR-SHE is shown as follows:

- MR-SHE.KeyGen(λ): A key generation algorithm that takes the security parameter λ as the input and produces the encryption public key PK along with a decryption secret key SK.
- MR-SHE.HKeyGen(PK, SK, w): Given a keyword $w \in \mathcal{W}$ and the key pair (PK, SK), it generates a homomorphic operation key HK_w for the keyword w, where $\mathcal{W}$ is the keyword space.
- MR-SHE.Trap(PK, SK, w): Given a keyword $w \in \mathcal{W}$ and the key pair (PK, SK), it generates a trapdoor T_w for the corresponding keyword w. Trapdoors refer to encrypted keyword for searching or matching over the encrypted data.
- MR-SHE.Enc(PK, w, m): Given a keyword $w \in \mathcal{W}$, a message $m \in \mathcal{M}$ and the public key PK, it encrypts m and returns the ciphertext C. $\mathcal{M}$ is the plaintext space.
- MR-SHE.Test(PK, T_w, C): To detect whether a ciphertext C matches a keyword w, the public key PK, trapdoor T_w of w and C are taken as the input, and it outputs the matching result: 1 (matched) or 0 (not matched).

TABLE 1
An Example of Firewall Rule Table

Rule number	Source IP	Destination IP	Source Port	Destination Port	Protocol	Action
1	192.168.1.*	172.16.1.*	*	*	TCP	Permit
2	172.16.3.*	192.168.1.10 ∼ 192.168.1.116	*	[1230,5150]	TCP	Permit
3	192.168.4.*	172.16.*.*	8090	*	TCP	Permit
4	192.168.3.*	192.168.*.*	*	[130,150]	UDP	Deny
...	...	...	...	...	...	...

- MR-SHE.Dec(PK, SK, w, C): Given key pair (PK, SK), a keyword w and a ciphertext C, it decrypts C and returns the plaintext m if the keyword w is matched. Otherwise, it returns $\perp$.
- MR-SHE.Eval(PK, HK_w, C_1, C_2): To perform homomorphic operation over two ciphertexts C_1, C_2, it first checks the integrity of C_1, C_2 using PK and HK_w. If the integrity is not breached, a ciphertext C (multiplicative homomorphic operation result of C_1, C_2) is returned. Otherwise, it returns $\perp$.

3.4 Self-Revocable Encryption

Self-revocable encryption [23] (SRE) is a cryptographic technique that aims to address the problem of *compelled access* for the cloud storage system. In the reality, the authority may force an individual to surrender the access credentials (e.g., secret key, password, PIN, etc.) of sensitive data stored on cloud servers or even the personal mobile devices. When encountered with this compelled access, the authority will get full access of one's private data. To solve this privacy concern, SRE allows user to revoke an encrypted file and this operation is indistinguishable from deleting a file. Afterwards, if necessary, user can recover the revoked file using the private restoration token, which is securely stored on a private device. The authors propose a secure *erasable index* to achieve SRE, which is a tree with encrypted key on each node. The erasable index is adopted in this paper to support probable cause privacy.

4 Proposed Scheme

In this section, we present SVPI in detail. The proposed packet inspection scheme can support secure header and payload inspection simultaneously. In specific, the secure header inspection mainly focus on detecting the header information of each packet and payload inspection can filter out the illegitimate contents of the payload. To promote the practicality, a new mechanism is designed to support *probable cause privacy*, which allows the cloud server to decrypt the packet payload if some rules are matched or suspected contents are detected. Note that, header inspection should be executed before payload inspection, which is a standard practice [8].

4.1 Secure Header Inspection

Before we dive into the details, we show that as the typical use case of packet header inspection, firewall rules can be well-supported by using numerical range query [4], [5], [16]. As shown in Table 1, the first rule permits all the packets with source IP address 192.168.1.* and destination IP address 172.16.1.*. Here, "*" is the wildcard character. Thus, 192.168.1.* represents the IP range 192.168.1.1 ∼ 192.168.1.254 and 172.16.1.* represents the IP range 172.16.1.1 ∼ 172.16.1.254. The prefix matching is then converted to a range query. In the reality, most applications only open part of the TCP/UDP ports. Some of the rules only open specific source or destination ports, such as the third rule shown in Table 1. The other rules may open a range of ports or block dangerous ports which are shown in the second and forth rules. All these special cases can be resolved by conducting range query. Consequently, range query is frequently used, and header inspection is the foundation function of a firewall system.

Next, we illustrate the protocols of range query based secure header inspection on cloud server A and B over ciphertext domain. First, we define some symbols. Let $\mathbb{G}_a$ and $\mathbb{G}_b$ be groups with prime order q and $e : \mathbb{G}_a \times \mathbb{G}_a \rightarrow \mathbb{G}_b$ be a symmetric bilinear pairing [24]. $\mathcal{W} \subseteq \mathbb{Z}_q$ denotes the keyword space and $\mathcal{M} \subseteq \mathbb{G}_b$ represents the message space. Let $\{\mathcal{H} = \mathcal{H}_{HK} : \mathbb{G}^4 \rightarrow \{0, 1, \ldots, q-1\} | HK \in \mathcal{HK}\}$ be a TCR hash family [25], where $\mathcal{HK}$ is the homomorphic operation key space. $f : \mathbb{G}_b \rightarrow \mathcal{Y}$ is a smooth function. We use $x \leftarrow \mathcal{X}$ to denote an element x randomly chosen from set $\mathcal{X}$.

• *Overview*

It is difficult to directly conduct range query over the encrypted data. Thus, we propose a strategy to convert the problem of range query into the computation of vector inner product. This method can be well supported on two-server model. No additional communication round is required. Specifically, Cloud server A can compute the inner product and let server B reveal the result. We give a brief overview of the scheme as follows:

- *Initialization phase:* In this phase, GW generates the public, secret and homomorphic operation keys. GW also generates a trapdoor for each keyword. *Here the keyword means the types of packet header domains such as IP, TCP, UDP and so on*. The keys and trapdoors are then distributed to MB and S. Given the ranges of header information (defined in the rules), such as the permitted TCP ports, GW encrypts them and shares them with the outsourced MB.
- *Header encryption phase:* In this phase, S encrypts the keywords and corresponding header domains (e.g, IP address, TCP/UDP ports, etc.) using the same method as the encryption of the ranges of header information.
- *Header range detection phase:* In this phase, MB first checks whether the keyword is matched or not. Then, the header inspection is converted to range query over ciphertext domain. In specific, cloud server A computes the encrypted inner product of two encrypted vectors and sends the result to server

B. Then, B reveals the result and check that if the value of header information is located within the range using secret key.

• *Initialization*

- *Key generation:* Given the security parameter, GW invokes MR-SHE.KeyGen(λ) to obtain the public and secret keys (PK, SK) as:

$$\begin{cases} HK \leftarrow \mathcal{HK}, \{g, h_1, h_2, h_3, h_4\} \leftarrow \mathbb{G}_a, \\ \alpha \leftarrow \mathbb{Z}_q, g_1 = g^{\alpha}, \\ PK = \{g, g_1, h_1, h_2, h_3, h_4, f, HK\}, \\ SK = \alpha. \end{cases}$$

The homomorphic operation key HK_w and the trapdoor T_w of the keyword w are exactly the same. GW runs MR-SHE.Trap(PK, SK, w^*) to obtain HK_w and T_w as

$$\begin{cases} w^* = F(K_F, w) \ mod \ q, \\ r_{w,i} \leftarrow \mathbb{Z}_q, h_{w,i} = (h_i g^{-r_{w,i}})^{1/(\alpha - w^*)}, i = \{3, 4\}, \\ HK_w = T_w = \{g^{w^*}, r_{w,3}, r_{w,4}, h_{w,3}, h_{w,4}\}. \end{cases}$$

where F is an instance of PRF and K_F is the secret key of F.

- *Key distribution: PK* is shared to all the other entities in the system including S, R and MB. K_F should be delivered to S. SK is sent to cloud server B. $\{T_w, HK_w\}$ along with the encoded keywords w^* are sent to cloud server A.
- *Review of* MR-SHE.Enc*:* Given the encoded keyword w^* and the message m, randomly choose a number $r \leftarrow \mathbb{Z}_q$, MR-SHE.Enc$(PK, w^*, m)$ computes as

$$\begin{cases} c_{m,1} = g_1^r g^{-rw^*}, c_{m,2} = e(g, g)^r, \\ c_{m,3} = m \cdot e(g, h_1)^{-r}, c_{m,4} = e(g, h_2)^r, \\ \theta_m = \mathcal{H}_{HK}(c_{m,1}, c_{m,2}, c_{m,3}, c_{m,4}), \\ \eta_m = f(e(g, h_3)^r e(g, h_4)^{r\theta_m}), \\ C_m = (c_{m,1}, c_{m,2}, c_{m,3}, c_{m,4}, \eta_m). \end{cases}$$

Let C_m be the ciphertext of message m.

- *Encrypt ranges of header:* Given a range of packet header (e.g, permitted TCP ports) $[u, v]$, let $-u$ and $-v$ be the inverse of u and v in group $\mathbb{G}_b$. GW uses *Catalano-Fiore* transformation [26] to encrypt u as

$$\begin{cases} \mu_1 = (-u) - \nu_1 \ mod \ T, \nu_1 \leftarrow \mathbb{Z}_T, \\ G = e(g, g), C_{\nu_1} = \text{MR-SHE.Enc}(PK, w^*, G^{\nu_1}). \end{cases}$$

Then, the ciphertext of u is (μ_1, C_{ν_1}). The encryption of v is exactly the same as u and we use (μ_2, C_{ν_2}) to denote the ciphertext of v.

- *Distribution of encrypted ranges of header:* All the encrypted range boundaries such as (μ_1, C_{ν_1}) and (μ_2, C_{ν_2}) are forwarded to the server A.

Note that, the original MR-SHE scheme [19] disclosed g^w to the cloud server as part of the trapdoor. According to the discrete logarithm problem [27], adversary cannot deduce w with g, g^w. However, the background knowledge is easy to be accessed by the cloud server and the keyword space {TCP,UDP,IP, ...} is small, the adversary can raise dictionary attack to reveal w. Thus, we further encode w using PRF F with secret key K_F, adversary cannot infer w without K_F.

• *Header Encryption*

- *An example of header encryption:* Given the source TCP port number $port\_num$ as an example, S encrypts it as follows:

$$\begin{cases} \mu = port\_num - \nu \ mod \ T, \nu \leftarrow \mathbb{Z}_T, \\ C_\nu = \text{MR-SHE.Enc}(PK, w^*, G^\nu). \end{cases}$$

Thus, the ciphertext of $port\_num$ is (μ, C_ν).

- *Packet uploading:* S sends the encrypted headers to GW. Afterwards, for header inspection, the encrypted headers are uploaded to MB. Note that, the encrypted payload for payload inspection process should also be sent to MB.

• *Header Range Detection*

MB detects the packet header by conducting keyword matching and range query over encrypted rules. To resist the unexpected homomorphic operations cross different header domains, MB should first test whether the keyword of the packet domain matches the trapdoor. If so, MB performs range query. If the encrypted header domain falls into the pre-defined range, MB will return the result to GW and the pre-defined actions should be taken. If the keyword is not matched, MB continues to check the next rule. If none of the rule is matched, MB informs GW of the result and the packet should be sent to R by GW.

- *Convert range query:* We give an example to show how to convert the range query into one-time vector inner product and one time positive/negative test. Given the source port number $x = port\_num$ and the permitted domain range $[u, v], u < v$, the following inequations are equivalent to each other.

$$\begin{aligned} u \le x \le v &\Leftrightarrow (x - u) \cdot (x - v) \le 0 \\ &\Leftrightarrow x^2 + (-u)x + (-v)x + (-u)(-v) \le 0, \\ &\Leftrightarrow (x, x, x, -u) \cdot (x, -u, -v, -v) \le 0. \end{aligned}$$

Thus, the problem of range query is converted into checking an inequation $V_1 \cdot V_2 \le 0$, where $V_1 = (x, x, x, -u), V_2 = (x, -u, -v, -v)$. Note that, there are more than one instances of V_1 and V_2. For example, the third inequation still holds if $V_1 = (x, -x, -x, u), V_2 = (x, u, v, v)$. Furthermore, all the inequations are still equivalent to each other if differen instances of V_1 and V_2 are used. As mentioned above, x is encrypted by S and u, v should be encrypted by GW. Thus, if $V_1 = (x, x, x, -u), V_2 = (x, -u, -v, -v)$, S will not need to compute the inverse of x. By doing so, the computational burden of S is further reduced.

- *Keyword matching:* Let the ciphertext of $port\_num$ be (μ, C_v), T_w be the trapdoor of keyword w. Cloud server A invokes MR-SHE.Test(PK, T_w, C_v) to conduct keyword matching as follows:

1). A parses the trapdoor T_w and C_v as

$$\begin{cases} T_w = (g^{w^*}, r_{w,3}, r_{w,4}, h_{w,3}, h_{w,4}), \\ C_v = (c_{v,1}, c_{v,2}, c_{v,3}, c_{v,4}, \eta_v). \end{cases}$$

2). A calculates $\theta_v = \mathcal{H}_{HK}(c_{v,1}, c_{v,2}, c_{v,3}, c_{v,4})$.

3). A checks if $\eta_v = f(e(c_{v,1}, h_{w,3}h_{w,4}^{\theta_v})c_{v,2}^{r_{w,3}+\theta_v r_{w,4}})$ holds. If so, w is matched. Otherwise, A continues to check the next rule.

- *Review of* MR-SHE.Eval *:* Let the ciphertext of u and v be (μ_1, C_{ν_1}) and (μ_2, C_{ν_2}). As an example, $PK, HK_w, C_{\nu_1}, C_{\nu_2}$ are taken as the inputs.

1). A parses C_{ν_1} and C_{ν_2} as

$$\begin{cases} C_{\nu_1} = (c_{\nu_1,1}, c_{\nu_1,2}, c_{\nu_1,3}, c_{\nu_1,4}, \eta_{\nu_1}), \\ C_{\nu_2} = (c_{\nu_2,1}, c_{\nu_2,2}, c_{\nu_2,3}, c_{\nu_2,4}, \eta_{\nu_2}). \end{cases}$$

2). To verify the integrity of C_{ν_1}, C_{ν_2}, A computes

$$\begin{cases} \theta_{\nu_1} = \mathcal{H}_{HK}(c_{\nu_1,1}, c_{\nu_1,2}, c_{\nu_1,3}, c_{\nu_1,4}), \\ \theta_{\nu_2} = \mathcal{H}_{HK}(c_{\nu_2,1}, c_{\nu_2,2}, c_{\nu_2,3}, c_{\nu_2,4}), \\ t_{\nu_1} = f(e(c_{\nu_1,1}, h_{w,3}h_{w,4}^{\theta_{\nu_1}})c_{\nu_1,2}^{r_{w,3}+\theta_{\nu_1} r_{w,4}}), \\ t_{\nu_2} = f(e(c_{\nu_2,1}, h_{w,3}h_{w,4}^{\theta_{\nu_2}})c_{\nu_2,2}^{r_{w,3}+\theta_{\nu_2} r_{w,4}}). \end{cases}$$

If $\eta_{\nu_1} = t_{\nu_1}$ and $\eta_{\nu_2} = t_{\nu_2}$, the ciphertext is integrated and A returns 1. Otherwise, it returns $\perp$.

3). If A returns 1, A chooses a random number $r \leftarrow \mathbb{Z}_q$ and computes

$$\begin{cases} c_1 = c_{\nu_1,1}c_{\nu_2,1} \cdot g_1^r g^{-rw}, \\ c_2 = c_{\nu_1,2}c_{\nu_2,2} \cdot e(g,g)^r, \\ c_3 = c_{\nu_1,3}c_{\nu_2,3} \cdot e(g,h_1)^{-r}, \\ c_4 = c_{\nu_1,4}c_{\nu_2,4} \cdot e(g,h_2)^r, \\ \theta = \mathcal{H}_{HK}(c_1, c_2, c_3, c_4), \\ \eta = f(e(c_1, h_{w,3}h_{w,4}^{\theta})c_2^{r_{w,3}+\theta r_{w,4}}), \\ C = (c_1, c_2, c_3, c_4, \eta). \end{cases}$$

Finally, A returns C as the homomorphic operation result of C_{ν_1}, C_{ν_2}.

- *Encryption of* $V_1 \cdot V_2$ *:* Let (μ_1, C_{ν_1}) and (μ_2, C_{ν_2}) be the ciphretext of u and v. Let "+" be one time execution of MR-SHE.Eval$(\cdot)$ and $\mu_1 C_{\nu_2}$ be μ_1 times execution of MR-SHE.Eval$(\cdot)$. A encrypts $V_1 \cdot V_2$ as:

1). A computes

$$C_{uv} = \text{MR-SHE.Enc}(PK, w^*, \mu_1\mu_2) + \mu_1 C_{\nu_2} + \mu_2 C_{\nu_1}.$$

2). Let $(C_{uv}, C_{\nu_1}, C_{\nu_2})$, (C_{xx}, C_v, C_v), (C_{xu}, C_v, C_{ν_1}) and (C_{xv}, C_v, C_{ν_2}) be the ciphertexts of $u \cdot v, x \cdot x, x(-u)$ and $x(-v)$, respectively.

3). A computes $\Sigma = C_{xx} + C_{xu} + C_{xv} + C_{uv}$.

4). A returns $C_{V_1 \cdot V_2}$ that equals to

$$(\Sigma, (C_v, C_v), (C_v, C_{\nu_1}), (C_v, C_{\nu_2}), (C_{\nu_1}, C_{\nu_2})),$$

along with the encoded keyword w^* to the cloud server B.

- *Result revealing:*

1). B decrypts each component of $C_{V_1 \cdot V_2}$ using $SK = \alpha$ with encoded keyword w^*. By invoking the algorithm MR-SHE.Dec$(\cdot)$ and computing these discrete logarithms, the plaintext of $V_1 \cdot V_2$ will be revealed.

```
SID:494: alert tcp $HTTP_SERVERS $HTTP_PORTS -> $EXTERNAL_NET any
(msg:"ATTACK RESPONSES command completed"; content:"Command
completed"; nocase; flow:from_server,established; classtype:bad-unknown)
```

Fig. 3. No. 494 snort rule [28].

2). If $V_1 \cdot V_2 \leq 0$, the rule can be asserted to be matched. Then, MB triggers the pre-defined action. Otherwise, B continues to process the next rule. Note that, the action could be allowing the packet passing or any other actions.

4.2 Secure Payload Inspection

MB aims to filter out the keywords in the passing *packet payloads* that are considered as malicious contents defined in the rules. S should first pre-process (tokenization) the packet payload and encrypt the tokens. The encrypted tokens are then uploaded to MB by GW. Note that, the repeated symbols in this part are irrelevant to the header inspection scheme.

4.2.1 Traffic Pre-Processing

Example of Payload Inspection Rule. We first give an example of a real payload inspection rule. As shown in Fig. 3, the No. 494 rule has two keywords, that are "Command" and "completed" (the words after "content:"). If a passing packet contains both keywords (i.e., the pattern "Command completed"), this rule is matched. The actions could be sending an alert to R or cut off the connection, which are pre-defined by GW.

Then, we give a brief introduction on the tokenization method. As we know, there are two methods to tokenize the packet payload. That are delimiter and window based schemes [6], [14]. Window based scheme segments the string into tokens with same length. Delimiter based method segments the string according to the position of the delimiters. Thus, the lengths of the tokens are different. In this paper, we choose delimiter-based method, because the size expansion of the traffic is smaller than window based scheme. Both the traffic and rule keywords are tokenized before being sent to MB. For an example, given a string "login.php?user=David", it can be tokenized as "login.", "login.php", ..., etc.

4.2.2 Keyword Matching

• *Initialization*

GW first invokes BGN.KeyGen(λ) to generate public key $PK = (n, \mathbb{G}, \mathbb{G}_1, e, g, h)$ and private key $SK = p$. PK is then shared to other entities of the system including MB, S and R. SK is only sent to cloud server A. Given an encrypted keyword k that is tokenized, it is encrypted as $\widehat{k} = g^{n-k}h^{r_1} \in \mathbb{G}$, where r_1 is randomly chosen from $\mathbb{Z}_n$. All the tokenized keywords should be encrypted in the same way. Finally, the encrypted keywords are sent to server A. It is worthy to point out that the decryption of $\widehat{k}$ is not $n - k$. As mentioned in Section 3.2, n is the order of $\mathbb{G}$ and $n - k = (p-1)q + q - k$. Thus, $q - k$ is the decryption of $\widehat{k}$.

• *Traffic Sending*

S and R build a TLS/SSL connection and encrypt the traffic using the generated session key. Then, S converts the packet payload into tokens using delimiter based method.

For token t, it is encrypted as $\widehat{t} = g^t h^{r_2} \in \mathbb{G}$, $r_2 \in \mathbb{Z}_n$. Finally, the encrypted traffic (only contains encrypted packet payloads, i.e., TLS/SSL protocol) along with the encrypted tokens are redirected to MB (cloud server A) by GW.

• *Inspection*

Here, we use the encrypted token $\widehat{t}$ as the example to show the inspection processing. Upon receiving $\widehat{t}$, server A computes the matching discriminator as $d = \widehat{t} \cdot \widehat{k} = g^{n+t-k} h^{r_1+r_2} \in \mathbb{G}$. Then, d will be sent to B to reveal the matching result. Directly decrypting d is very time-consuming due to the computation of discrete logarithms. Thus, we let B only conduct one-time modular exponentiation. Specifically, if $t - k = 0$, $\bar{d} = (g^n)^p$ should be the identity element of the group $\mathbb{G}$. For simplicity, we let $\bar{d} = (g^n)^p = 1$. Thus, if $\bar{d} = 1$ holds, the token is then matched. B should hold the traffic and continues to inspect the remain discriminators. If any rule is matched, the result should be returned to GW, and the corresponding actions should be executed by GW. Otherwise, the encrypted tokens along with the encrypted traffic are forwarded to R by GW. Note that, if $t \neq k$, $\bar{d} = d^p = (g^p)^{q-k+t} \in \mathbb{G}$. Thus, thes difference value $\Delta = t - k$ may be disclosed to server B.

• *Verification*

According to the threat model described in Section 2.4. S is possible to be compromised and attempts to evade the inspection rules. Thus, R should verify the encrypted tokens sent from S. Since the decryption of the BGN encrypted tokens leads to low efficiency, we let B decrypt the traffic and tokenize the payload using the same method of S. Afterwards, w randomly chosen tokens $\{n - t_1, n - t_2, \ldots, n - t_w\}$ are encrypted using BGN as the verification object that should be sent to MB. Then, MB can simply conduct equality checking between verification object and encrypted tokens uploaded by S. If any token is found to be not the same (not matched), warnings will be sent to R. The size of verification object is determined by R based on the historical performance of S. The verification operation is also optional for R which depends on the trust level of S and the state of the connection.

• *Privacy Enhancing Operation*

As discussed in *Inspection* phase, the difference value Δ of the token and keyword may be disclosed to server B. Using the private key p, B can easily reveal the plaintext of $\Delta = t - k$ when $t > k$. Base on this leakage, when large-scale difference values are obtained, it is possible to reveal the data distribution. Thus, the encrypted tokens are then put into risk. To solve this problem, a privacy enhancing operation is proposed.

The main idea is randomizing the difference values before being sent to B. A first samples a random number r_3 from $\mathbb{Z}_n$ and computes the hash value as $H(r_3)$. $H(\cdot)$ is a cryptographic hash function. Afterwards, $H(r_3)$ is encrypted as $\widehat{H}(r_3) = g^{H(r_3)} h^{r_3} \in \mathbb{G}$. Given a matching discriminator $d = g^{n+\Delta} h^{r_1+r_2} \in \mathbb{G}$, we randomize d by mapping it to group $\mathbb{G}_1$ which is shown as follow.

$$\begin{aligned} \widehat{d} &= e(d, \widehat{H}(r_3)) \\ &= g_1^{H(r_3)(n+\Delta)} h_1^{(n+\Delta)r_3 + r_3(n+\Delta) + \beta q (r_1+r_2) r_3} \in \mathbb{G}_1, \end{aligned} \tag{1}$$

where $\beta \in \mathbb{Z}$. If $\Delta = 0$, $(\widehat{d})^p = (g^n)^p = 1$. Therefore, this operation still supports equality checking. If $\Delta \neq 0$, $(\widehat{d})^p$ is a random number. Consequently, Δ is protected from B. In order to boost the efficiency, all the hashing and encryption of random numbers can be pre-processed. Also, the bilinear paring can be computed in parallel.

4.2.3 Probable Cause Privacy

In this part, we propose a method to achieve the property of *probable cause privacy* [6]. We first review the basic concept of probable cause privacy and the scheme proposed in BlindBox [6]. Then, we point out the critical privacy risks of BlindBox [6] and discuss the drawbacks of the heuristic scheme. Afterwards, a practical method is proposed by carefully tailoring the key technical component (*erasable index*) of SRE [23].

Probable cause privacy [29] is a privacy policy that requires the middlebox is authorized to decrypt the encrypted traffic *only if* the suspicious keywords or rules with high security risks are matched. In specific, GW defines the PCP policy such as malicious keywords. A packet containing the malicious keyword should be decrypted and inspected over plaintext by the middlebox. For example, if the keyword "violence" is found in a packet, decryption of the packet is then permitted. The outsourced middlebox can further conduct inspection over the plaintext of the packet. In reality, some complex payload inspection functions, such as regular expressions and deep learning-based techniques for malware detection, must access the plaintext. BlindBox [6] addresses this problem by embedding the secret TLS/SSL session key SK_{SSL} into the encrypted tokens. Thus, the MB can easily obtain the secret keys. This method brings two privacy risks. First, the secret key is easy to be revealed by middlebox or any passive adversary, which directly puts high risk on the confidentiality of the packets encrypted with SK_{SSL}. Second, if MB is authorized to reveal the secret key, all the follow up packets can also be decrypted by the same key.

To address these issues, a heuristic method is using a new secret key when middlebox requests plaintext inspection. This method is simple and easy to be deployed. Nevertheless, the history packets that have already been inspected are exposed to the cloud server if MB is outsourced. Another troublesome problem is the network administrator has to ask S to update the secret key. In another word, S will know that the previously sent packet is required to be decrypted. There must be some keywords matched. Thus, a malicious S may analyze this packet and try to avoid using sensitive words in the following packets. Thus, S should not get involved into secret key update when plaintext inspection happens, which makes it difficult to support probable cause privacy.

To this end, we find that a privacy-preserving and practical *key management* scheme is a practical method. Inspired by the concept of *compelled access* and SRE [23], we consider S is the owner of the packets who has the right to delete/restore any packet. Meanwhile, MB is authorized to decrypt the payload using secret key when strong evidence is provided. We propose a key management scheme that meets these two conflicting requirements. In the initialization phase, as shown in Fig. 4, S constructs a key tree using symmetric encryption $\Pi = (KeyGen, Enc, Dec)$. The root node

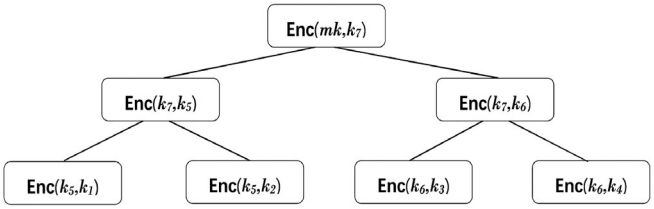


Fig. 4. An example of key tree.

is a secret key k_7 encrypted by the master key mk. The children of the root are also secret keys k_5, k_6 encrypted by k_7. In the same way, all the leaf nodes are secret keys $\{k_1, \ldots, k_4\}$ encrypted by k_5, k_6. S should share $\{k_1, k_2, k_3, k_4\}$ with R as the TLS/SSL session keys. Intuitionally, the best security could be achieved by using different keys for each packet. However, the number of keys will explode and both S and R have to manage huge amount of session keys, which is impractical. Thus, we define an integral constant l to represent the number of generated session keys. l is set by S. The session keys are denoted as $\{k_1, k_2, \ldots, k_l\}$. The Nth packet payload is then encrypted by k_i, where $i = N \bmod l$. If S doubts that key k_i may be disclosed, S can delete the corresponding leaf node. In doing so, no entity except R can decrypt the packets encrypted by k_i. Thus, for MB, the packets encrypted by k_i are then *deleted*. Moreover, if the root of the key tree is removed by S, all the previously sent encrypted packets are then deleted in the view of MB.

To support the *compelled access*, all the session keys are encoded and uploaded to GW. Given key $k_i, 1 \leq i \leq l$; S computes $\widehat{k_i} = k_i \oplus tk_{res} \oplus i$, where tk_{res} is a random number generated by S. Thus, the encoded keys stored on GW can be denoted as $\{\widehat{k_1}, \widehat{k_2}, \ldots, \widehat{k_l}\}$. S also shares tk_{res} with MB as the token for session key request. Once a keyword or an inspection rule is matched and plaintext inspection is highly recommended by the policy, MB sends the current packet number N and all the matched encrypted tokens to GW to request the session key. Then, GW detects the tokens to figure out whether the key request is necessary or not. If so, GW computes $i = N \bmod l$, and $\bar{k_i} = \widehat{k_i} \oplus i$. Afterwards, GW sends $\bar{k_i}$ to MB. The session key k_i can be revealed by conducting one-time XOR as $k_i = \bar{k_i} \oplus tk_{res}$. Finally, MB can decrypt the packet payload using k_i to conduct further analysis. Note that, in this scheme, only cloud server A is involved to support probable cause privacy.

Discussion. As mentioned above, all the packets are encrypted cyclically using a group of keys (represented by a encrypted key tree). Suppose packet A is encrypted using key k_1, and the next packet B is encrypted using k_2, the decryption of P_1 will not breach the privacy of P_2.

We argue that PCP may not improve the comprehensive performance of secure header inspection significantly. Compared to payload inspection, the function of the header inspection is simple. Advanced operations such as the regular expression based inspection is not needed. Moreover, the encrypted header cannot be used for packet routing directly. Thus, it needs at least one round of interaction between the cloud server and the gateway. If any dangerous content is detected, the gateway can receive the warning information immediately. Since most of the header fields are numerical values, cryptographical tools are sufficient to support the inspection functions. Overall, the benefit of introducing PCP to header inspection is not significant.

5 Security Analysis

In this section, we give a thorough analysis on the security and privacy issues of SVPI according to the privacy requirements listed in Section 2.5, these are header privacy, payload privacy and rule set privacy. Versatile security properties proposed in the privacy requirements are achieved.

5.1 Header Privacy

In SVPI, the header domains are encrypted by MR-SHE [19], which is formally proved to be secure under chosen ciphertext attack (IND-CCA) secure. In the view of cloud server A, only the ciphertexts of the header domains are accessible. No PPT adversary is able to deduce additional information over the ciphertexts. The decryption algorithm of MR-SHE is able to detect the illegal homomorphic operations over different domains. For instance, A cannot invoke MR-SHE.Eval$(\cdot)$ over encrypted TCP and UDP data (the keywords are different). Thus, without the secret key K_F that is used to encode the keywords, no PPT adversary is able to generate the ciphertext that can be the input of MR-SHE.Eval$(\cdot)$. Due to the security property provided by PRF, the encoded keywords cannot be distinguished from a real random number by a PPT adversary. Consequently, the problem of small message space is solved. In the view of cloud server B, the inner product of V_1 and V_2 is disclosed. Therefore, the rough range of the encrypted header domain is revealed to B. According to the threat model, server A and B are not allowed to exchange data unless it is required by the pre-defined protocol. Thus, B still cannot figure out the original value of the encrypted header domains. In a word, the header privacy is well preserved.

5.2 Payload Privacy

Since the payload is encrypted using two different methods, we discuss the privacy issues separately.

5.2.1 End-to-End Encrypted Payload

A widely applied and simple way to protect the privacy of packet payload is using the end-to-end encryption provided by HTTPS protocol. Previously proposed private key based schemes [3], [6], [7] all choose this method. The packet payload is encrypted with the session key of HTTPS. Therefore, MB cannot extract any sensitive private information from the traffic. The public key based protocols in [13], [14] also keep the session key as a secret from MB. Thus, traffic privacy is well preserved.

Another security property achieved by SVPI is that the probable cause privacy scheme is transparent to S. Using the shared token tk_{res} and the encoded secret key received from GW, MB can recover it by conducting XOR operation. Meanwhile, S is not involved. This property is important because if S knows which packet is required to be decrypted, it may attempt to analyze the content of the packet payload or even cut off the connection to evade the further inspection. According to the threat model, GW is fully trusted. But we still let S encode the secret keys before

uploading them to the MB. This operation strictly follows the rule that one's privacy should be protected unless there is evidence for suspicion [6].

5.2.2 Encrypted Tokens

In SVPI, we use BGN to encrypt each token. BGN [20] is a probabilistic crypto-system that is proved to be IND-CPA secure. Thus, in the view of A, all the received tokens cannot be distinguished from the random numbers. Without the private key, no additional information can be deduced from the encrypted tokens. The information leaked to server B is the matching discriminators. B can recover the final result using discriminator and the secret key to obtain a random number or 1 (identity element). Since the encrypted tokens and keywords are stored on A and are out the reach of B. Therefore, B cannot deduce the original values. Even if the privacy-enhancing operation is not conducted for the sake of efficiency, it is still difficult for B to infer the raw values of the token and keyword. Given a simple example, the difference value of keyword k_1 and token t_1 is denoted as $\Delta = t_1 - k_1$ ($t_1 > k_1$). Then, B can reveal the plaintext of Δ using private key. However, there should exist other pairs of tokens and keywords with exactly the same difference value (Δ). Formally, $\exists(t_i, k_j), 0 < i \neq 1, j \neq 1 < T$ let the equation $\Delta = t_i - k_j$ holds. Consequently, the token privacy is well protected.

5.3 Rule Set Privacy

The rule set for header inspection is encrypted by MR-SHE [19] which is proved to be IND-CCA secure. Due to the property of mis-operation resistance of MR-SHE, server A cannot guess the range of an individual rule by conducting homomorphic operations. Specifically, A could encrypt a number within the message space and conduct homomorphic operation on the ciphertext of this number and the encrypted range. Then, A can figure out if the number locates within the range with the help of server B. Since the message space is small (e.g., the TCP port number range is [0, 65535]), it is possible to recover the encrypted range only using normal homomorphic operations. MR-SHE solves such problem by embedding the domain name (e.g., TCP,UDP, . . . ,etc) into the ciphertext. However, since the space of the domain name is extremely small, it is easy to enumerate all the domains. We conquer this problem by encoding the domain name using PRF and the secret key is only shared with GW and S. Without the encoded domain name, A cannot generate ciphertext that is eligible to support homomorphic operation. Furthermore, A may force to raise homomorphic operation which will be detected by B. Therefore, we claim that SVPI provides strong rule set privacy protection.

The rule set for payload inspection is encrypted by BGN. Since the message spaces of the tokens as well the keywords are large, it is difficult to recover the plaintext only using homomorphic operation. Thus, the privacy of the rules is well preserved.

6 Performance Evaluation

In this section, we give a comprehensive discussion on the performance of SVPI in terms of functionality, communication and computational overheads. We mainly focus on the factors that have key impacts on the performance. The proposed Secure Header Inspection (SHI) scheme and Secure Payload Inspection (SPI) scheme are evaluated separately. We also compare the experimental results with GWYJ (presented by Guo *et al*. [4]) and BlindIDS [13] to demonstrate the feasibility of SHI and SPI.

TABLE 2
Comparison of Functionality

	[6]	[14]	[13]	[4]	SVPI
Header inspection	×	×	×	✓	✓
Payload inspection	✓	✓	✓	×	✓
Probable cause privacy	✓	×	×	×	✓

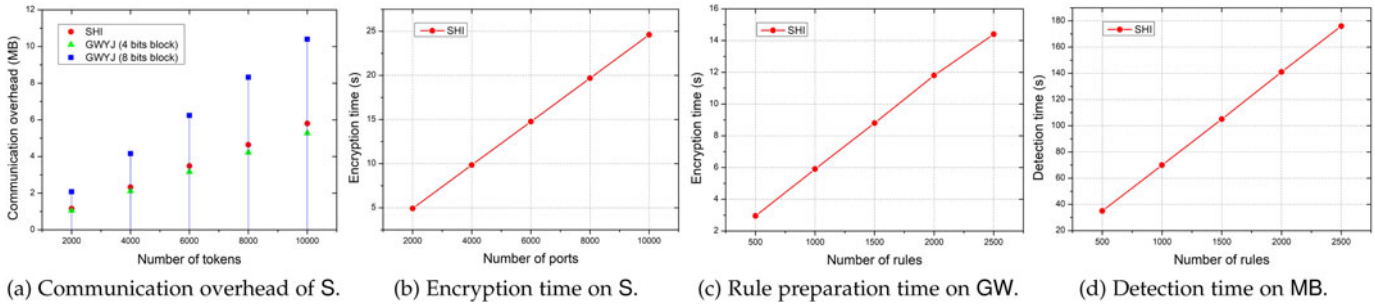
The experiments are conducted on a 2.10 GHz processor (Intel(R) Xeon(R)), 16 GB memory server with Ubuntu 18.04 operation system. The program language is Java 8. The public lightweight intrusion detection rule set Snort [28] is used as the payload inspection rules and a simulated firewall rule set is adopted. We take the real website data (e.g., ABC, CNN, BBC, etc.) as the data source.

6.1 Functionality

In this paper, the secure header and payload inspection are supported simultaneously. Moreover, in order to deal with advanced and complex inspection, probable cause privacy is also supported. Compared to existing schemes [3], [4], [6], [13], [14], SVPI achieves versatile functionality. BlindBox [6] has focused on payload inspection and probable cause privacy. SPABox [14] and BlindIDS [13] support payload inspection. GWYJ [4] pays attention to header inspection. The listed schemes in Table 2 are representative schemes.

6.2 Evaluating Secure Header Inspection

Communication Overhead of S. The communication between S and MB includes TLS/SSL traffic, encrypted tokens and some other control information. We only consider overhead brought by encrypted tokens that are the main different factor between different schemes. The tokens encrypted by MR-SHE consists of two parts. Using (μ, C_v) as an example, μ is sampled from $\mathbb{Z}_T$ where T should be no larger than the result of inner product. Since IP address is small than 2^{32}, T is set as $4 \cdot 2^{64}$. Then 66 bits will be enough for T. C_v consists of five parts, that are one element of group $\mathbb{G}_a$, three elements of group $\mathbb{G}_b$ and a hash value. We use SHA-512 as the hash function and $|\mathbb{G}_a| = |\mathbb{G}_b| = 128$ Bytes. Thus, the total size of one encrypted token should be 0.58 KB. GWYJ [4] uses ORE [15] to encrypt the tokens. As claimed in [4], when the block size is 4 bits, the ciphertext will be 528 Bytes. If the block size is 8 bits, the ciphertext will be 1040 Bytes. Larger block size provides a higher security level. GWYJ uses AES-128 and HMAC-256 to encrypt the tokens. As shown in Fig. 5a, when the number of tokens reaches 10^4, the total communication overheads will be 10.5 MB for GWYJ (8 bits block), 5.3MB for GWYJ (4 bits block) and 5.8MB for SHI. In modern access networks, the time cost for sending 5.3MB message should be acceptable. In north America, the average network speed has reached 17MB per second in 2020. Thus, sending 10^4



(a) Communication overhead of S. (b) Encryption time on S. (c) Rule preparation time on GW. (d) Detection time on MB.

Fig. 5. Performance of SHI (SVPI) and GWYJ [4] (communication overhead).

encrypted tokens only cost roughly 0.34 seconds. The communication overheads of both schemes increase linearly with the number of tokens.

Encryption Time on S. To encrypt one port, SHI has to conduct five modular exponentiations, five bilinear parings and three hashing operations. It should be time-consuming if we process these operations one by one. Since g, h_1, h_2, h_3, h_4 are elements of the public key, all the paring operations $\{e(g,g), e(g,h_1), e(g,h_2), e(g,h_3), e(g,h_4)\}$ could be operated in advance. To boost the efficiency, the modular exponentiation as well as the encryption of different ports could be executed in parallel. On average, it will take roughly 2.6ms to encrypt one port. When the number of ports is 10^4, the encryption time reaches 24.5s. Thus, on average, to encrypt one port, the time cost is 2.45 ms. For the packet sender, such time cost is little and no significant latency is introduced. As shown in Fig. 5b, the executing time increases linearly with the number of ports.

Rule Preparation Time on GW. Here, a rule refers to a port range. GW prepares the encrypted rules by generating the ciphertext version of each single range. Thus, after the keyword is encoded using PRF, the upper bound and the lower bound of the range are encrypted with exactly the same way as the token encryption on sender side. On average, it takes 5.9ms to prepare one rule. When the number of rules is 2500, the time cost will be 14.4s. Since the encrypted rules are reusable for different connections, 14.4s time cost is little. As shown in Fig. 5c, the encryption time of SHI increases linearly with the number of rules.

Detection Time on MB. MB should detect whether an encrypted port is within the encrypted ranges. First, the cloud server A processes keyword matching using hash functions. If the keyword (header domain name) is matched, A computes the encrypted inner product of vector V_1 and V_2. Then, A sends the encrypted inner product to server B for decryption. Otherwise, A continues to check the next rule. Hashing is very fast while the encryption and decryption of vector inner product are time-consuming. On average, if the keyword is matched, the time cost is 67ms. We assume that every keyword is matched. Thus, as shown in Fig. 5d, when the number reaches 2500, the time cost is 167s. The detection time increases linearly with the number of rules. Note that, for real firewall rule, one port can only match with a part of the rules ($< 1/3$). Moreover, we let the server conduct detection one port by one port, one rule by one rule. To boost the efficiency, they can be computed in several virtual machines in parallel. By doing so, the time cost will be significantly reduced. Here, we only show the lower bound of the detection efficiency and leave the optimization process as the future work.

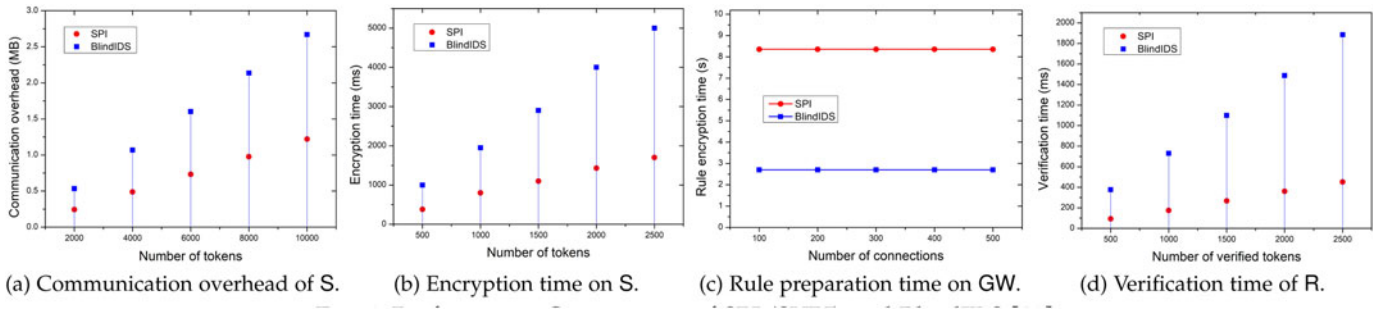
Discussion. The encryption, decryption and detection efficiency of symmetric encryption based schemes [3], [4] are indeed higher than SHI. We acknowledge that this should be the limitation of public key based schemes. Nevertheless, SHI has enhanced the packet header privacy and reduced the communication overhead of S. The interested readers can refer to literature [4] to acquire the performance details of GWYJ.

6.3 Evaluating Secure Payload Inspection

Communication Overhead of S. An encrypted token of SPI only has one element of the group $\mathbb{G}$. We set $|\mathbb{G}| = 128$ Bytes. The ciphertext of one token generated by BlindIDS [13] consists of four parts. That are two elements of a cyclic group with prime order, one random number in $\mathbb{Z}_q$ and a half hash value, where q is a large prime. The used hash function is SHA-256 and the size of the group is not given. To be fair, we set the same size as $\mathbb{G}$, and use 127 bits Mersenne prime to instantiate q. Thus, the total size of an encrypted token will be roughly 280 Bytes. As shown in Fig. 6a, the communication overheads of SPI and BlindIDS are 1.2 MB and 2.7 MB respectively when the number of tokens is 10^4. Thus, SPI has reduced more than half of the communication cost. As mentioned in section 6.2, in modern network environment, such communication load can be fastly transmitted.

Encryption Time on S. To encrypt one token, only one-time BGN [20] encryption is required for SPI. It requires one time of bilinear pairing and four times of modular exponentiation for BlindIDS to encrypt a token. As shown in Fig. 6b, the time costs of token encryption of SPI and BlindIDS [13] increase linearly with the number of tokens. To encrypt 2500 tokens, it takes SPI and BlindIDS 1.8s and 5s respectively. Therefore, SPI has reduced the encryption time on sender side.

Rule Preparation Time on GW. Since the public key based payload inspection schemes allow the users to share the same encrypted rules, we evaluate the rule preparation efficiency over the same size (3K) rule set. Thus, as shown in Fig. 6c, the encryption time of SPI and BlindIDS [13] are two constants. This should be an advantage of public key based schemes. The number of connections is irrelevant to the rule preparation time. It is a constant. Specifically, to encrypt a rule, BlindIDS needs one modular exponentiation and one hashing operation. On average, for whole rule set, the time cost is 2.7s.



(a) Communication overhead of S. (b) Encryption time on S. (c) Rule preparation time on GW. (d) Verification time of R.

Fig. 6. Performance comparison of SPI (SVPI) and BlindIDS [13].

For SPI, the rules are also encrypted by BGN (i.e., one keyword one BGN encryption). The encryption time of SPI is totally 8.358s. As a result, SPI is able to deal with large-scale and frequently established/revoked connections. However, symmetric key based scheme BlindBox [6] has to generate new rules for newly built connections, which leads to high connection initialization overhead.

Detection Time on MB. The total rule detection time of BlindIDS, our basic and advanced scheme (privacy enhancing operation is conducted) are clearly shown in Table 3. One-time bilinear pairing and one-time hashing are required for BlindIDS [13] to determine the matching result of a token and a keyword. For our basic scheme, only one modular multiplication and one exponentiation operation are required for single keyword matching. In our advanced scheme, we enhance privacy with one-time bilinear paring and one-time hashing. On average, to conduct 3K rule detection for one packet, the time cost of BlinIDS and our basic scheme are 475.17s, 287.53s respectively. Nearly half the time cost is reduced. In BlindIDS, the experimental result is 74s using a server with better configurations. Indeed, public key based schemes are time-consuming and can hardly fit the real network traffic speed. As mentioned above, we argue that, since all the computation can be processed in parallel over several virtual machines, it is possible to significantly boost the efficiency.

Verification Time of R. According to the threat model, the sender could be compromised and generate the tokens without following the pre-defined protocol. Thus, the receiver has to verify the received encrypted tokens. The decryptable searchable encryption [30] used in BlindIDS [13] supports the recovery of encrypted tokens. Most existing schemes attempt to check every token one by one. We argue that it is unnecessary to verify every encrypted token. In SPI, w tokens are randomly chosen for verification. If the total number of tokens is n, then $w = 0.1 \cdot n$. As shown in Fig. 6d, both BlindIDS and SPI lead to linear time costs, but SPI performs better. When the number of tokens is 2500, the time cost of SPI is 451ms while BlindIDS needs 1885ms.

TABLE 3
Comparison of Detection Time

Description	BlindIDS	Basic scheme	Advanced scheme
1 Token, 1 Rule	5.11ms	2.84ms	7.37ms
1 Packet, 1 Rule	286.26ms	173.35ms	463.17ms
1 Token, 3K Rules	4.53s	2.72s	7.79s
1 Packet, 3K Rules	475.17s	287.53s	727.18s

7 Related Work

In this section, we give a brief introduction to the recent state-of-art works that closely relate to our scheme.

The first work that aims to outsource the traditional middlebox services to the cloud server is proposed by Sherry *et al.* [2]. The authors have designed and implemented a platform called APLOMB that significantly improves the scalability and functionality of middleboxes with reasonable overheads. However, they did not consider the security and privacy issues brought by the outsourcing. This concern could be the barrier of the deployment of cloud-assisted middleboxes. Fruitful achievements are presented to address such issues. They can be divided into two categories, that are secure header and payload inspection. We elaborate on each of them as follows:

7.1 Secure Header Inspection

Lan *et al.* [3] designed and implemented a system called Embark to deploy middlebox services on the cloud server with privacy preserving. The key technique contribution of Embark is the efficient and secure prefix matching algorithm named as PrefixMatch. This algorithm only supports IPv6 and is failed to process range query based rules. Theoretically, it is possible to convert a port range to several prefixes. The storage and communication overheads will be increased due to the expansion of rules. Embark also supports payload inspection by directly using the existing searchable encryption schemes [31] without any modification. To extend the functionality, Guo *et al.* [4] presented a range query based scheme using the order-revealing encryption [15]. ORE provides IND-CPA secure with the cost of significant ciphertext expansion. If the message space is large, the storage overhead will be high. Besides, the rule update is complex and time-consuming. Guo *et al.* [5] also presented a secure stateful firewall. The authors added counters into rules and using ORE to conduct range query. Note that, the above schemes are all designed base on symmetric encryption. Thus, if any compromised user discloses the secret key, the whole system will be under risk.

7.2 Secure Payload Inspection

The first work towards secure payload inspection named as BlindBox is proposed by Sherry *et al.* [6]. The authors use Gabled circuit [32] to obfuscate the rules. The tokens are encrypted by advanced encryption standard (AES). Due to the using of heavy gabled circuit and each connection requires a newly obfuscated rule set, the connection initialization of BlindBox becomes time-consuming. To solve this problem, Yuan *et al.* [7] use Cuckoo hash table [12] as the token filter. In doing so, the preparation of the rule filter and the encoding of the tokens are significantly boosted in terms of time complexity. Recently, Ning *et al.* [33] design a secure traffic inspection scheme named as PrivDPI. It offers reusable obfuscated rules using public key encryption. PrivDPI has significantly boosted the efficiency of building connections and achieved the same security and privacy guarantee. Moreover, the encrypted tokens are also reusable for packet senders. Yuan *et al.* also presented a verifiable secure payload inspection scheme [34] for cloud computing that allows the users to verify the correctness of the protocol execution results. In [35], the authors have proposed an lightweight and fine-grained verification scheme for symmetric key based packet inspection. Another important issue is the update of the encrypted rules, especially for symmetric encryption based schemes. Thus, Guo *et al.* [5] also use two-server architecture to support secure and dynamic payload inspection without disclosing the access pattern. Similar to header inspection schemes, the symmetric encryption based payload inspection schemes are also plagued by the problem of secret key leakage.

In [14], Fan *et al.* designed a new and simple additive homomorphic crypto-system based on discrete logarithm problem to support secure payload inspection. The authors also explore malware detection using privacy-aware machine learning method (SVM) [36]. The proposed SPABox [14] gains higher efficiency with the cost of leaking the plaintext of rule set to middlebox. Thus, it may be difficult to deploy SPABox on cloud server. Recently, BlindIDS [13] was proposed by Canard *et al.* that adopts decryptable searchable encryption [30] to process token matching. BlindIDS allows the users to share the same encrypted rule set. However, the heavy sender side encryption and cloud side detection are the barriers of BlindIDS for real deployment. Indeed, public key based schemes [13], [14] including SVPI all face the problem of heavy encryption operations. Thus, we suggest that the middlebox services can be implemented on the local fog nodes [16] to reduce the high traffic latency.

Discussion. Secure and privacy-preserving middleboxes developed on Trusted Execution Environment (TEE) [37] (e.g., SGX [38]), also have been intensively explored. The private and sensitive data can be processed over the enclave without using cryptograph techniques. Compared to cryptographic schemes [3], [6], [13], [14], the performances of TEE based schemes can be increased for specific applications. As reported in LightBox [37], which should be the latest result, the packet I/O can achieve 10Gbps. In ShieldBox [39], the authors designed a secure cloud-assisted framework for rich network functions atop shielded execution [40]. The ENDBOX [41], for the first time, implemented the middleboxes on the client machine to improve the scalability of middleboxes. We argue that it seams out of reach to design a highly efficient scheme without using TEE. However, TEE needs additional equipment costs and is difficult to be outsourced or virtualized. Moreover, TEE requires trusted remote attestations and specialized code designs for different applications, and may suffer from side-channel attacks [42]. Thus, users may decide which system to implement depending on the demands.

8 Conclusion

In this paper, we have proposed a packet inspection scheme to support secure header and payload inspection on two non-collusion cloud servers. We have carefully tailored the designs of latest cryptographic techniques to offer versatile functions of range query based header inspection and token-based payload inspection. The security analysis has proved that the proposed SVPI can well preserve the privacy of packet header, payload and the outsourced inspection rules. The performance evaluation has demonstrated that SVPI offers competitive performance. The observations and countermeasures of privacy leakages in the real-world packet inspection functions may shed light on the design and analysis of future middlebox services. Therefore, in the future, we will explore secure and highly reusable inspection rules for cloud-assisted middleboxes. We will also investigate privacy-preserving network function virtualization for the next generation network.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grants 2017YFB0802300 and 2017YFB0802000, in part by the National Natural Science Foundation of China under Grants 62020106013, 61972454, 61802051, 61772121, and 61728102, in part by Sichuan Science and Technology Program under Grants 2020JDTD0007 and 2020YFG0298, in part by the Peng Cheng Laboratory Project of Guangdong Province PCL2018KP004. Special thanks should be given to the professors and research colleagues of BBCR Lab at University of Waterloo for the valuable discussion. The preliminary result of this article was presented at the 10th International Conference on Wireless Communications and Signal Processing (WCSP 2018) [1].

References

[1] H. Li, H. Ren, D. Liu, and X. Shen, "Privacy-enhanced deep packet inspection at outsourced middlebox," in *Proc. 10th Int. Conf. Wireless Commun. Signal Process.*, 2018, pp. 1–6.

[2] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," in *Proc. ACM SIGCOMM Conf. Appl. Technol. Architectures Protocols Comput. Commun.*, 2012, pp. 13–24.

[3] C. Lan, J. Sherry, R. A. Popa, S. Ratnasamy, and Z. Liu, "Embark: Securely outsourcing middleboxes to the cloud," in *Proc. 13th Usenix Conf. Netw. Syst. Des. Implementation*, 2016, pp. 255–273.

[4] Y. Guo, C. Wang, X. Yuan, and X. Jia, "Enabling privacy-preserving header matching for outsourced middleboxes," in *Proc. IEEE/ACM 26th Int. Symp. Quality Service*, 2018, pp. 1–10.

[5] Y. Guo, C. Wang, and X. Jia, "Enabling secure and dynamic deep packet inspection in outsourced middleboxes," in *Proc. 6th Int. Workshop Secur. Cloud Comput.*, 2018, pp. 49–55.
[6] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "BlindBox: Deep packet inspection over encrypted traffic," in *Proc. ACM Conf. Special Interest Group Data Commun.*, 2015, pp. 213–226.
[7] X. Yuan, X. Wang, J. Lin, and C. Wang, "Privacy-preserving deep packet inspection in outsourced middleboxes," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
[8] C. Wang, X. Yuan, Y. Cui, and K. Ren, "Toward secure outsourced middlebox services: Practices, challenges, and beyond," *IEEE Netw.*, vol. 32, no. 1, pp. 166–171, Jan./Feb. 2018.
[9] J. Liang, Z. Qin, S. Xiao, L. Ou, and X. Lin, "Efficient and secure decision tree classification for cloud-assisted online diagnosis services," *IEEE Trans. Dependable Secure Comput.*, to be published, doi: 10.1109/TDSC.2019.2922958.
[10] X. Liu, R. Deng, K. R. Choo, and Y. Yang, "Privacy-preserving outsourced support vector machine design for secure drug discovery," *IEEE Trans. Cloud Comput.*, vol. 8, no. 2, pp. 610–622, Apr.–Jun. 2020.
[11] Y. Mao, Y. Zhang, X. Zhang, F. Xu, and S. Zhong, "Location privacy in public access points positioning: An optimization and geometry approach," *Comput. Secur.*, vol. 73, pp. 425–438, 2018.
[12] B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher, "Cuckoo filter: Practically better than bloom," in *Proc. 10th ACM Int. Conf. Emerg. Netw. Experiments Technol.*, 2014, pp. 75–88.
[13] S. Canard, A. Diop, N. Kheir, M. Paindavoine, and M. Sabt, "BlindiDS: Market-compliant and privacy-friendly intrusion detection system over encrypted traffic," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2017, pp. 561–574.
[14] J. Fan, C. Guan, K. Ren, Y. Cui, and C. Qiao, "SPABox: Safeguarding privacy during deep packet inspection at a middlebox," *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3753–3766, Dec. 2017.
[15] K. Lewi and D. J. Wu, "Order-revealing encryption: New constructions, applications, and lower bounds," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1167–1178.
[16] H. Ren, H. Li, Y. Dai, K. Yang, and X. Lin, "Querying in Internet of Things with privacy preserving: Challenges, solutions and opportunities," *IEEE Netw.*, vol. 32, no. 6, pp. 144–151, Nov./Dec. 2018.
[17] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "VerifyNet: Secure and verifiable federated learning," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 911–926, 2020.
[18] L. Melis, H. J. Asghar, E. De Cristofaro, and M. A. Kaafar, "Private processing of outsourced network functions: Feasibility and constructions," in *Proc. ACM Int. Workshop Secur. Softw. Defined Netw. Netw. Function Virtualization*, 2016, pp. 39–44.
[19] K. Emura, T. Hayashi, N. Kunihiro, and J. Sakuma, "Mis-operation resistant searchable homomorphic encryption," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2017, pp. 215–229.
[20] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Proc. Theory Cryptogr. Conf.*, 2005, pp. 325–341.
[21] W. Shen, J. Qin, J. Yu, R. Hao, J. Hu, and J. Ma, "Data integrity auditing without private key storage for secure cloud storage," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2019.2921553.
[22] A. Yang, J. Xu, J. Weng, J. Zhou, and D. S. Wong, "Lightweight and privacy-preserving delegatable proofs of storage with data dynamics in cloud storage," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2018.2851256.
[23] N. Tyagi, M. H. Mughees, T. Ristenpart, and I. Miers, "BurnBox: Self-revocable encryption in a world of compelled access," in *Proc. 27th USENIX Conf. Secur. Symp.*, 2018, pp. 445–461.
[24] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen, "Anonymous reputation system for IIoT-enabled retail marketing atop PoS blockchain," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3527–3537, Jun. 2019.
[25] E. Andreeva *et al.*, "New second-preimage attacks on hash functions," *J. Cryptol.*, vol. 29, no. 4, pp. 657–696, 2016.
[26] D. Catalano and D. Fiore, "Using linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1518–1529.
[27] N. P. Smart, "The discrete logarithm problem on elliptic curves of trace one," *J. Cryptol.*, vol. 12, no. 3, pp. 193–196, 1999.
[28] Snort rules, 2018, Accessed: May 31, 2018. [Online]. Available: http://https://www.snort.org/
[29] P. Ohm, "Probably probable cause: The diminishing importance of justification standards," *Minnesota Law Rev.*, vol. 94, 2009, Art. no. 1514.
[30] T. Fuhr and P. Paillier, "Decryptable searchable encryption," in *Proc. Int. Conf. Provable Secur.*, 2007, pp. 228–236.
[31] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.
[32] M. Bellare, V. T. Hoang, and P. Rogaway, "Foundations of garbled circuits," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 784–796.
[33] J. Ning, G. S. Poh, J.-C. Loh, J. Chia, and E.-C. Chang, "PrivDPI: Privacy-preserving encrypted traffic inspection with reusable obfuscated rules," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 1657–1670.
[34] X. Yuan, H. Duan, and C. Wang, "Bringing execution assurances of pattern matching in outsourced middleboxes," in *Proc. IEEE 24th Int. Conf. Netw. Protocols*, 2016, pp. 1–10.
[35] H. Ren, H. Li, D. Liu, G. Xu, N. Cheng, and X. Shen, "Privacy-preserving efficient verifiable deep packet inspection for cloud-assisted middlebox," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2020.299116.
[36] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, 2011, Art. no. 27.
[37] H. Duan, C. Wang, X. Yuan, Y. Zhou, Q. Wang, and K. Ren, "LightBox: Full-stack protected stateful middlebox at lightning speed," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 2351–2367.
[38] J. Van Bulck *et al.*, "Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution," in *Proc. 27th USENIX Conf. Secur. Symp.*, 2018, pp. 991–1008.
[39] B. Trach, A. Krohmer, F. Gregor, S. Arnautov, P. Bhatotia, and C. Fetzer, "ShieldBox: Secure middleboxes using shielded execution," in *Proc. Symp. SDN Res.*, 2018, pp. 1–14.
[40] S. Arnautov *et al.*, "SCONE: Secure linux containers with intel SGX," in *Proc. 12th USENIX Conf. Operating Syst. Des. Implementation*, 2016, pp. 689–703.
[41] D. Goltzsche *et al.*, "EndBox: Scalable middlebox functions using client-side trusted execution," in *Proc. 48th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2018, pp. 386–397.
[42] Y. Xu, W. Cui, and M. Peinado, "Controlled-channel attacks: Deterministic side channels for untrusted operating systems," in *Proc. IEEE Symp. Security Privacy*, 2015, pp. 640–656.

Hao Ren (Student Member, IEEE) is currently working toward the PhD degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), China. He is also a visiting PhD student at ECE Department, University of Waterloo, Canada from February 2018 to January 2020. His research interests include applied cryptography, security and privacy for cloud computing, and machine learning.

Hongwei Li (Senior Member, IEEE) received the PhD degree from the University of Electronic Science and Technology of China, China, in June 2008. He is currently the head and a professor with the Department of Information Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China, China. He worked as a postdoctoral fellow with the University of Waterloo, Canada from 2011 to 2012. His research interests include network security and applied cryptography. He is the distinguished lecturer of IEEE Vehicular Technology Society.

Dongxiao Liu (Member, IEEE) received the PhD degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada, in 2020. He is currently a postdoctoral research fellow with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research interests include applied cryptography and privacy enhancing technologies for blockchain.

Guowen Xu (Student Member, IEEE) received the BS degree in information and computing science from the Anhui University of Architecture, China, in 2014. Currently, he is working toward the PhD degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, China. His research interests include cryptography, searchable encryption, and the privacy-preserving deep learning.

Xuemin Shen (Fellow, IEEE) received the PhD degree from Rutgers University, Brunswick, New Jersey, in electrical engineering, in 1990. He is a university professor with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. His research focuses on resource management in interconnected wireless/wired networks, wireless network security, social networks, smart grid, and vehicular ad hoc and sensor networks. He is a registered professional engineer at Ontario, Canada, an Engineering Institute of Canada fellow, a Canadian Academy of Engineering fellow, a Royal Society of Canada fellow, and a distinguished lecturer of IEEE Vehicular Technology Society and Communications Society. He has received the R.A. Fessenden Award, in 2019 from IEEE, Canada, Award of Merit from the Federation of Chinese Canadian Professionals (Ontario) presents, in 2019, James Evans Avant Garde Award, in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award, in 2015, and Education Award, in 2017 from the IEEE Communications Society, and Technical Recognition Award from Wireless Communications Technical Committee (2019) and AHSN Technical Committee (2013). He has also received the Excellent Graduate Supervision Award, in 2006 from the University of Waterloo, Canada and the Premier's Research Excellence Award (PREA), in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee chair/co-chair for the IEEE Globecom'16, the IEEE Infocom'14, the IEEE VTC'10 Fall, the IEEE Globecom'07, the Symposia chair for the IEEE ICC'10, and the chair for the IEEE Communications Society Technical Committee on Wireless Communications. He is the elected IEEE Communications Society vice president for Technical and Educational Activities, vice president for publications, member-at-large on the Board of Governors, chair of the Distinguished Lecturer Selection Committee, and member of IEEE Fellow Selection Committee. He was/is the editor-in-chief of the *IEEE Internet of Things Journal*, *IEEE Network*, *IET Communications*, and *Peer-to-Peer Networking and Applications*.