Privacy-preserving Efficient Verifiable Deep Packet Inspection for Cloud-assisted Middlebox

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

Abstract—With the increasing traffic volume, enterprises choose to outsource their middlebox services, such as deep packet inspection, to the cloud to acquire rich computational and communication resources. However, since the traffic is redirected to the public cloud, information leakages, such as packet payload and inspection rules, arouse privacy concerns of both middlebox owner and packet senders. To address the concerns, we propose an efficient verifiable deep packet inspection (EV-DPI) scheme with strong privacy guarantees. Specifically, a two-layer architecture is designed and deployed over two non-collusion cloud servers. The first layer fast filters out most of legitimate packets and the second layer supports exact rule matching. During the inspection, the privacy of packet payload and the confidentiality of inspection rules are well preserved. To improve the efficiency, only fast symmetric crypto-systems, such as hash functions, are used. Moreover, the proposed scheme allows the network administrator to verify the execution results, which offers a strong control of outsourced services. To validate the performance of the proposed EV-DPI scheme, we conduct extensive experiments on the Amazon Cloud. Large-scale dataset (millions of packets) is tested to obtain the key performance metrics. The experimental results demonstrate that EV-DPI not only preserves the packet privacy, but also achieves high packet inspection efficiency.

Index Terms—Cloud computing, Middlebox, Network function outsourcing, Privacy-preserving.

1 Introduction

Middlebox [1] is a network equipment that supports a wide spectrum of network functions for enterprise networks. For instance, a middlebox can provide firewall, load balancer and deep packet inspection (DPI) services [2]. Nowadays, some of the modern middlebox services are delay sensitive. Moreover, it is also challenging to offer high efficiency facing with the explosion of traffic volume. For instance, DPI is a typical delay sensitive network function. One of its key performance metrics is the packet throughput within a certain period of time. Thus, to achieve high efficiency, the most appealing solution is outsourcing the DPI service to the *cloud platform* [3], [4]. Various *benefits* can be acquired with *the assistance of the cloud servers*. First, powerful computation and communication capabilities [5] are provided, which makes it feasible to support efficient DPI over large-scale traffic volume. Second, for the owner of middlebox, diverse DPI functions can be customized to meet the new requirements without purchasing additional hardware. Third, the heavy burden of the daily management of DPI system is released. In addition, the advanced DPI functions, such as machine learning [6], [7] based malware detection [8], can be efficiently supported by cloud computing. Consequently, significant attentions have been paid to the outsourcing of DPI for cloud-assisted middlebox [5].

Unfortunately, the DPI outsourcing also introduces several security and privacy concerns. In specific, the network traffic has to be redirected to the cloud for inspection. As a result, an important privacy concern is the exposure of packet payload. For example, the personal information of enterprise employees is inevitably disclosed to the cloud server if without any protection. The cloud service provider may even attempt to analyze the private contents for economic interest. Moreover, the passing packets may contain sensitive information that relates to commercial secrets of an enterprise. If these kinds of information are leaked to the cloud or any competitor, serious losses may be caused. Another crucial issue is the confidentiality of the DPI rules. Usually, the details of the DPI rules directly reflect the security and privacy policies. If an internal or external attacker has accessed the DPI rules, it will be easier to evade the inspection. With such strong background information, the attacker can even find some loopholes of the system. Thus, both the packet payload and the DPI rules should be protected from the public cloud. A simple way to achieve this goal is using standard crypto-systems (e.g., AES, RSA) to encrypt the packet payload and the DPI rules. Unfortunately, it is usually difficult to process DPI directly over ciphertext domain [5]. Therefore, it is challenging and urgent to design a privacy-preserving DPI scheme over

- *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, China (e-mail: renhao.uestc@gmail.com; e-mail: hongweili@uestc.edu.cn).*
- *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)*
- *Nan Cheng is with the School of Telecommunication Engineering, Xidian University, Xian 710071, Shanxi, China, (e-mail: dr.nan.cheng@ieee.org)*
- *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@uwaterloo.ca; sshen@uwaterloo.ca)*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2991167, IEEE Transactions on Cloud Computing

cloud platform.

Some approaches have been proposed to offer DPI service on the public cloud with privacy protection. The first milestone-like work BlindBox [2] formally defined the security and privacy requirements of middleboxes. It also provided an efficient solution using symmetric encryption. BlindBox [2] utilized garbled circuit [9] to obfuscate the DPI rules, which could be time-consuming for large-scale connections. Yuan *et al.* [10] adopted broadcast encryption [11]. It can support the sharing of encrypted rules between different connections. Later, their subsequent work [12] proposed an efficient method that is able to verify the inspection results. Recently, Guo *et al.* [13] designed a dynamic DPI scheme to support rule update. Several public key encryption based schemes [8], [14], [15] are also proposed to explore diverse functions such as malware detection and decryptable matching. Due to the using of public key crypto-system, computation overheads are inevitably increased. As a result, the time cost on packet sender side becomes higher. Meanwhile, the total packet throughput is significantly decreased.

Previously proposed works have provided diverse DPI services with different levels of privacy preservation. There are still some issues not fully addressed. On one hand, larger packet throughput without compromising the privacy protection is one of the crucial design goals. On the other hand, efficient and fine-grained inspection result verification is not well supported. These issues are challenging to solve due to the natural conflicts between functionality, efficiency and privacy. To tackle these challenges, we present *three observations* that are not considered by existing works. 1). First, in the reality, most contents of the packet payloads are not matched (more than 99%) by any DPI rules. Therefore, these packets should be fast filtered out. Intuitively, the content filtering and exact rule matching should be conducted separately. By doing so, the whole DPI process efficiency can be boosted significantly. 2). Second, result verification may introduce extra packet delay, if the results are verified before packet forwarding. As a practical method, the verification can be executed independently. 3). Third, since most of the packets will not be matched, only verifying the final execution result is insufficient. Thus, the execution details should be proved to offer a fine-grained verification.

In this paper, we propose an efficient verifiable deep packet inspection scheme (EV-DPI) with privacy protection over two non-collusion cloud servers. EV-DPI adapts fast symmetric encryption primitives [16] to support privacy-preserving DPI. EV-DPI also achieves inspection result verification using Cuckoo hashing [17], [18]. The verification and DPI can be processed independently. This design guarantees the high performance in terms of network latency. The main contributions of this paper are summarized as follows.

- We propose a two-layer inspection architecture. The first layer can fast filter out the most legitimate packets using encoded Bloom filter [19]. The second layer supports exact rule matching using carefully tailored conjunctive searchable encryption scheme [16]. By doing so, EV-DPI achieves lower packet processing cost on sender side and larger packet throughput on middlebox side. Moreover, the intermediate and final inspection results returned by both cloud servers can all be efficiently verified.
- EV-DPI can preserve the privacy of packet payload and the confidentiality of DPI rules against semi-honest cloud servers [20]. Moreover, to conceal the size pattern (i.e., the number of keywords) of each DPI rule, we propose a secure rule extension scheme. By doing so, the cloud server cannot distinguish two encrypted rules based on the size pattern. Thus, the confidentiality of DPI rules stored on the cloud servers is further enhanced.
- Extensive experiments are conducted over Amazon Cloud [21] to demonstrate the efficiency of EV-DPI. In specific, the network administrator (gateway) is simulated by a local server. The prototype of middlebox is implemented on the cloud based on the public DPI rule set [22]. Without compromising the privacy, EV-DPI is more efficient in terms of packet latency and packet throughput.

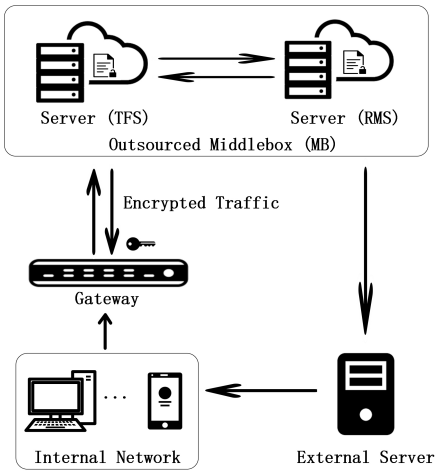


Fig. 1: System Model.

The remainder of this paper is organized as follows. In Section 2, the system and threat models are described. Based on the models, the design goals are presented. At last, the building blocks are reviewed. In Section 3, we show the design details of EV-DPI. The security analysis and the performance evaluation are provided in Section 4 and Section 5, respectively. In Section 6, closely related works are reviewed. Section 7 concludes the paper.

2 Models and Design Goals

In this section, we first review the system and threat model. Then, we present the design goals to capture the requirements of functionality, privacy and efficiency. At last, the cryptographic primitives used as the building blocks are briefly introduced.

2.1 System Model

As shown in Fig. 1, the proposed system consists of four entities to capture the typical scenarios of the cloud assisted middlebox. The gateway (GW) is the administrator of the

internal network. It is responsible for key management and DPI rule generation. At the system initialization phase, GW outsources the encrypted DPI rule set to the middlebox. Afterwards, all the packets sent from the internal network will be gathered by GW. The payload of each packet should be encoded for privacy protection. At last, the encoded packets are redirected to middlebox (MB). The MB is implemented by two non-collusion cloud servers. One is token filtering server (TFS) and the other is rule matching server (RMS). Note that, MB can be instantiated by two real or virtual cloud servers. The external server provides various services to the users of internal network that could be file storage, e-mail, web and so on. To clarify the details of the system, we describe the *work flow* as follows.

- **System initialization:** GW generates an encoded filter, an encrypted rule set, and uploads them to TFS and RMS, respectively. All the secret keys will be generated and distributed to TFS and RMS. To support the verification of the inspection results, GW also constructs two encoded hash tables and uploads them to TFS and RMS.
- **Packet processing:** GW tokenizes the payloads of packets sent from the internal network. Each token is then encoded. Afterwards, encoded tokens along with the packets are redirected to MB for DPI process.
- **Token filtering:** This is the first layer of EV-DPI. Upon receiving the encoded tokens, TFS fast filters out all the matched tokens. If there is no token matched, the traffic should be transmitted to the external server. Otherwise, TFS and RMS will collaboratively conduct exact rule matching. Note that, TFS should generate the verification object for each token to prove the execution correctness.
- **Rule matching:** This is the second layer that returns the final inspection result. RMS needs to determine whether each rule is exactly matched or not. During the matching process, RMS may interact with TFS. If any rule is matched, RMS will trigger the pre-defined actions. RMS also needs to return the result to GW for further detection. Otherwise, the packet should be forwarded as usual. It is also needed for RMS to generate the verification objects.
- **Verification:** TFS and RMS each maintains a *verification space*. They are used to store the verification objects. The size of the verification space is decided by GW. And GW can verify the returned results at any time. The verification only involves GW. TFS and RMS are unaware of when and which packet is verified.

Life-cycle of a packet:

Here, we review the life-cycle of a packet to show the basic work flow. Suppose a packet sender needs to send a packet to the external server. The packet is first forwarded to GW. Then, the payload of the packet is segmented into tokens. GW encodes all the tokens and sends them to the cloud server. Each encoded token should be tested by TFS. It filters out the tokens that also appears in the outsourced middlebox rule. If no token is matched, TFS can transmit the packet to the external server as usual. Otherwise, TFS generates search tags for all the matched tokens. Then, the search tags are sent to RMS. In the last step, RMS conducts exact rule matching using the search tags. If any rule is asserted to be matched, RMS may choose to drop the packet or even cut off the connection. Meanwhile, RMS needs to return the result to GW. If no rule is matched, RMS forwards the packet as usual. This is how a packet is processed and inspected.

Benefits of using two non-collusion cloud servers:

Using two non-collusion cloud servers to implement secure and privacy-preserving system is a common method in public key based schemes [15], [23]. Normally, one server carries the computational burden and the other reveals the result. Recently, Guo *et al.* [13] also adopt this design to implement a symmetric key based scheme. By doing so, more functions are achieved with less information leakage. In this paper, we use this method to boost the efficiency and reduce the information leakage simultaneously. We summarize the benefits as follows.

1). *Reduced leakage:* TFS is unaware of the final matching result. Thus, it is difficult to launch attacks rely on the information of result pattern. RMS cannot access the original encoded tokens. Moreover, some side channel information such as the volume of matched tokens is protected from RMS.

2). *Improved efficiency:* First, due to the using of two cloud servers, the interaction between MB and GW is avoided. Thus, the network delay is also reduced. Second, TFS filters out most of the unmatched tokens and generates new search tokens for previously matched tokens. Then, GW is released from heavy token encryption burden.

2.2 Threat Model

In this paper, GW is considered to be fully trusted. It is permitted to have full access to all the secret keys and encrypted packets. The cloud-assisted middlebox including TFS and RMS are semi-honest [24], [25]. They will follow the predefined protocols, yet they may be interested in the contents of passing packets. TFS and RMS may attempt to recover the distribution or even the contents of encrypted DPI rule set and passing tokens. Thus, any information leakages of the protocols may be abused by TFS or RMS. For instance, the size of the DPI rule set, the matching result of DPI rules and so on. These two servers are not allowed to exchange any information that is not pre-permitted. Formally, this model is defined as two non-collusion servers [13], [23]. Moreover, the cloud servers are possible to be "Lazy" [12]. For economic benefit, the cloud service provider may only perform partial computational tasks (e.g., only 70% packets are inspected) to reduce the computational cost.

2.3 Design Goals

Under the system and threat model, EV-DPI should be carefully designed to meet the requirements of functionality, privacy protection and efficiency. In specific, the design goals of EV-DPI are shown as follows.

- *Functionality:* As the basic DPI function, rule matching must be supported. Any token that appears in rule set should be detected. Also, the corresponding actions should be triggered. If any rule is matched,

the middlebox is required to return the encrypted matched rules as well as the matched tokens to GW. Verifiability is a desirable function that should be supported. It allows GW to verify the correctness of inspection result.

- *Privacy protection:* To preserve the privacy of senders, the contents of the packet payload and tokens should be concealed from MB. The confidentiality of the outsourced DPI rules should also be preserved.
- *Efficiency:* On the GW side, the token encryption algorithm should be lightweight. On the MB side, the unmatched tokens should be filtered out efficiently. To boost the efficiency, we aim to support *paralleled computing* on both GW and MB sides.

2.4 Cryptographic Primitives

Basic Cryptographic Primitives: 1). A pseudo-random function (PRF) is a one-way function $G : X \times K \rightarrow Y$ if for all randomly chosen key $k \in K$, all probabilistic polynomial-time adversary $\mathcal{A}$ cannot distinguish $G(k,\cdot)$ from a real random function $f(k,\cdot)$ from X to Y. 2). A symmetric encryption consists three polynomial-time algorithms $\Sigma = (\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$. $\mathrm{KeyGen}(\cdot)$ can generate the secret key k. Message m is encrypted as $c = \mathrm{Enc}(k,m)$ and decrypted as $m = \mathrm{Dec}(k,c)$.

Bloom Filter: Bloom filter [19] is a storage-efficient data structure that supports membership test. Given a binary vector $BF[i] = 0$, where $i \in [1,l]$ and l is the length of BF. To add an element x into BF, k hash functions $\{h_1, ..., h_k\}$ are used as $BF[h_j(x) \ mod \ l] = 1$. Thus, given an element y, we can compute $h_j(y) \ mod \ l$ and check the value $BF[h_j(y) \ mod \ l]$. If for all $j \in [1,k]$, $BF[h_j(y) \ mod \ l] = 1$, then y is within BF. The standard Bloom filter supports the operation of adding new element. However, it fails to process deletion of inserted elements. Bloom filter may introduces false positive during the membership test. The formal analysis of the false positive rate is given in [19].

Cuckoo Hashing: Cuckoo hashing [17] is implemented by two tables T_1, T_2 with same size S. Given two hash functions h_1, h_2, an element e can be inserted into one of the two locations $T_1[h_1(e) \ mod \ S], T_2[h_2(e) \ mod \ S]$. If the locations $T_1[h_1(e) \ mod \ S]$ and $T_2[h_2(e) \ mod \ S]$ are all occupied. Then, it let e still take the computed location and the original element is deleted and inserted again using the same method. If no empty location is found during the insertion, we have to re-build two larger hash tables. To search an element, at most two times of hashing are required.

Tuple Set: Tuple set (T-set) is presented by Cash *et al.* [26]. It is an inverted index that supports searchable encryption (encrypted keyword search). T-set consists three algorithms $\Gamma = (\mathrm{TsetSetup}, \mathrm{TsetGenTag}, \mathrm{TsetSearch})$. An array T is indexed by keywords. Algorithm $\mathrm{TsetSetup}(T)$ returns the private key K and the index Tset. T-set runs $\mathrm{stag} \leftarrow \mathrm{TsetGenTag}(K,w)$ to obtain the search token stag. $\mathrm{TsetSearch}$ is the search algorithm that can return the identifiers of matched files associated with stag.

Symmetric Hidden Vector Encryption: Symmetric hidden vector encryption (SHVE) is recently proposed by Lai *et al.* [16]. SHVE is a prediction encryption that is able to determine the intersection of two sets over ciphertext domain. Let Θ be an attribute set with finite size, $*$ be the wildcard and $\Theta_* = \Theta \cup \{*\}$. Given a PRF $F : \{0,1\}^\lambda \times \{0,1\}^{\lambda+log\lambda} \rightarrow \{0,1\}^{\lambda+log\lambda}$, a symmetric encryption $\Sigma = (\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$ with the same plaintext and key space $\{0,1\}^{\lambda+log\lambda}$.

```
alert tcp $HTTP_SERVERS $HTTP_PORTS -> $EXTERNAL_NET any (msg:"ATTACK
RESPONSES http dir listing"; content: "Volume Serial Number";
flow:from_server,established; classtype:bad-unknown; sid:1292; rev:4;)
```

Fig. 2: An example of DPI rule.

- $\mathrm{SHVE.Setup}(\lambda)$: It takes the security parameter λ as the input and samples a mater key mk uniformly from $\{0,1\}^\lambda$. The message space is defined as $\mathcal{M}$ ={"True"}. It outputs $(mk, \mathcal{M})$.
- $\mathrm{SHVE.KeyGen}(mk, \mathbf{u} \in \Theta_*^s)$: $\mathbf{u} = (u_1, ..., u_s)$ is the predicate vector with s elements. We use $L = \{1 \le l_i \le s | u_{l_i} \neq *\}, 1 \le i \le |L|$ to denote the locations in $\mathbf{u}$ that are not wildcard characters. We samples a key K uniformly from $\{0,1\}^{\lambda+log\lambda}$ and compute $k_1 = (\bigoplus_{i \in [|L|]} (F(mk, u_{l_i} || l_i))) \bigoplus K$ and $k_2 = \mathrm{Enc}(K, 0^{\lambda+log\lambda})$. Then, it outputs the decryption key $K_D = (k_1, k_2, L)$.
- $\mathrm{SHVE.Enc}(mk, m =$"True", $\mathbf{v} \in \Theta^s)$: It takes the master key mk, the message m and the index vector $\mathbf{v} = \{v_1, ..., v_s\}$ as the inputs. For all $i \in [s]$, it calculates $c_i = F(mk, v_i || i)$ and outputs $C = \{c_j | j \in [s]\}$ as the ciphertext.
- $\mathrm{SHVE.Query}(K_D, C)$: It takes the ciphertext C, the decryption key K_D as the inputs and parses them as $C = \{c_j | j \in [s]\}, K_D = (k_1, k_2, L)$. It computes $K = (\bigoplus_{i \in [|L|]} c_{l_i}) \bigoplus k_1$ and $m' = \mathrm{Dec}(K, k_2)$. If $m' = 0^{\lambda+log\lambda}$, it outputs "True"; otherwise $\perp$.

3 Proposed Scheme

In this section, we first give an example of real DPI rule to show the use case of EV-DPI. Then, a brief overview of the scheme is given. In specific, the inspection procedures of a packet is described. Afterward, the technical details of EV-DPI are presented. Specifically, the two-layer DPI processing and execution result verification are described step by step.

Example of real DPI rule:

We give an example of a DPI rule sampled from the public Snort rule set [22]. As shown in Fig. 2, the pattern"Volume Serial Number" that appears after "content:" are the keywords. And the number of this rule is 1292 (i.e., sid:1292). Thus, this rule can be tokenized as three keywords {"Volume","Serial","Number"}. If these three keywords appears in the same packet in succession, then the No.1292 rule can be asserted to be matched. For a matched rule, the pre-defined actions should be triggered by MB (RMS). The action could be sending an alert to GW (network administrator), discarding the packet, cutting off the connection, etc.

Scheme overview:

As shown in Fig. 3, a packet is used as an example to describe the more detailed work flow. It starts from the packet payload tokenization to the final step of EV-DPI. We

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2991167, IEEE Transactions on Cloud Computing

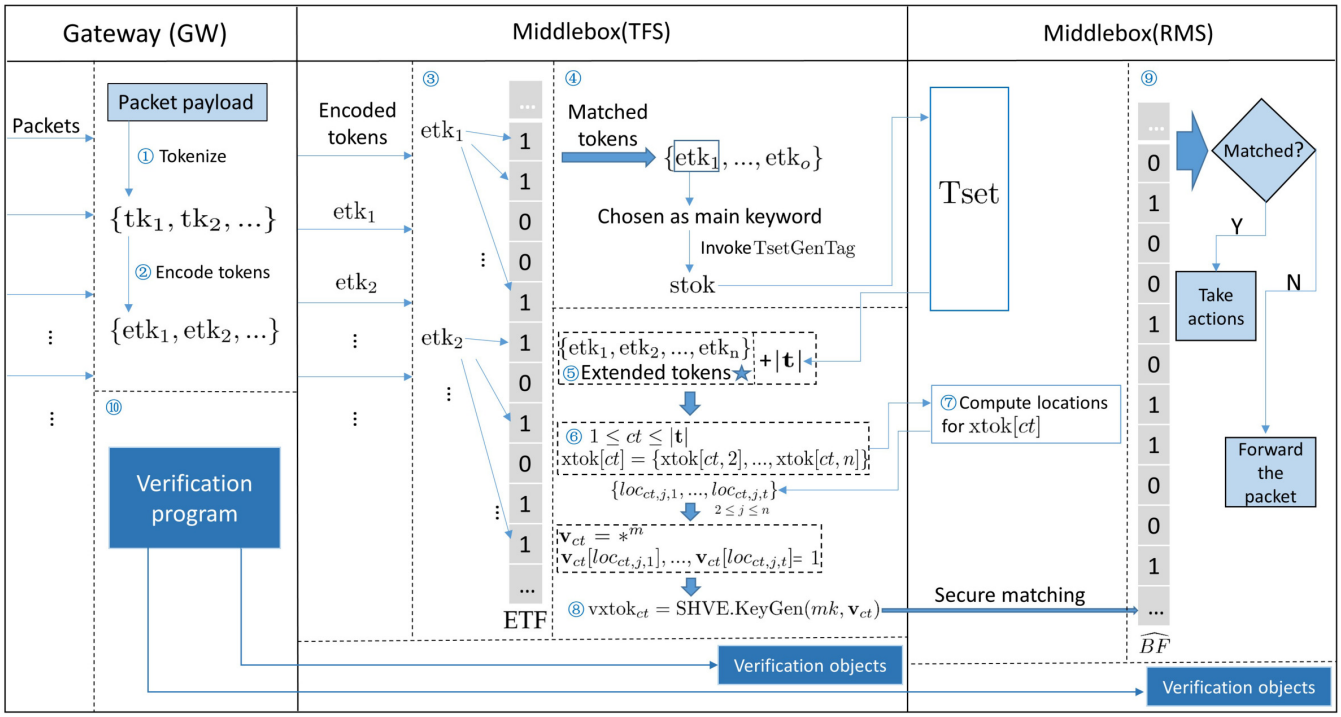


Fig. 3: Scheme Overview.

assume that the encoded token filter (ETF) and encrypted rule set (ERS) are all already generated by GW. ETF and ERS are then shared with TFS and RMS, respectively. ETF is a Bloom filter with all the encoded keywords of DPI rules as the elements. ERS can be parsed as ERS$=(\mathrm{Tset}, \widehat{BF})$. Tset is the inverted index of DPI rules. $\widehat{BF}$ is an encrypted Bloom filter that supports conjunctive keyword matching. Each step of EV-DPI is described as follow.

(1) *Packet tokenization:* Once the packet is forwarded to GW. The payload is tokenized as $\{\mathrm{tk}_1, \mathrm{tk}_2, ...\}$.
(2) *Token encoding:* Given the tokens, GW encodes them as $\{\mathrm{etk}_1, \mathrm{etk}_2, ...\}$ using PRF and sends them to TFS.
(3) *Fast token filtering:* For each encoded token, TFS uses ETF to fast filter out the unmatched tokens. This step is the standard Bloom filter membership test.
(4) *Main keyword search:* Let $\{\mathrm{etk}_1, \mathrm{etk}_2, ...\mathrm{etk}_o\}$ be the matched token. Then, etk_1 is chosen as the main keyword. Afterward, TFS invokes TsetGenTag to generate a T-set search token stok for etk_1. stok is sent to RMS. The size of the search result $|\mathbf{t}|$ (number of matched rules) are returned to TFS.
(5) *Token extending:* Given the matched tokens $\{\mathrm{etk}_1, \mathrm{etk}_2, ...\mathrm{etk}_o\}$, TFS extends them to n tokens as $\{\mathrm{etk}_1, \mathrm{etk}_2, ...\mathrm{etk}_n\}$, where n is also the size of extended DPI rules.
(6) xtok *generation:* To inspect the remain tokens $\{\mathrm{etk}_2, ...\mathrm{etk}_n\}$, TFS computes new search tokens as $\mathrm{xtok}[ct] = \{\mathrm{xtok}[ct, 2], ..., \mathrm{xtok}[ct, n]\}, (1 \leq ct \leq |\mathbf{t}|)$. Afterward, TFS sends $\mathrm{xtok}[ct]$ to RMS.
(7) *Compute the locations:* Upon receiving $\mathrm{xtok}[ct]$, RMS computes the locations using hash functions and return them to TFS. Here, locations indicate the "1" positions in a Bloom filter (not encrypted).
(8) *Vector encryption:* TFS generates a vector $\mathbf{v}_{ct}$ with value "1" in the computed locations. Then, TFS invokes SHVE.KeyGen to generate the search tokens vxtok_{ct} and sends them to RMS.
(9) *Matching result revealing:* After searching vxtok_{ct} over $\widehat{BF}$, RMS reveals the final matching result of each DPI rule. Once a rule is matched, pre-defined actions will be triggered by RMS. Otherwise, the packet should be forwarded to the external server.
(10) *Result verification:* GW can verify the results returned by TFS and RMS independently at any time.

3.1 Two-layer Deep Packet Inspection

• System initialization:

In this phase, GW generates ETF and ERS. Afterwards, ETF should be uploaded to TFS and ERS should be forwarded to RMS. ETF serves as the first layer of EV-DPI and ERS serves as the second layer.

The generation of ETF is clearly shown in Algorithm 1. First, each keyword in rule set $\mathcal{R}$ is encoded by PRF F with secret key K_T. Then, each encoded keyword is mapped into the Bloom filter BF using t hash functions. Afterward, BF is attached to ETF. Note that, in the reality, multiple rule sets can be represented by the same or different Bloom filters. The design details are similar to single rule set. Here, we only consider one rule set. At last, ETF along with the t hash functions are uploaded to TFS.

To implement the second layer of EV-DPI, we build an encrypted index (ERS) of the keyword set $\mathcal{R}$. As illustrated in Algorithm 2, ERS consists of three parts. The first part is an encrypted inverted index Tset [26]. The construction

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2991167, IEEE Transactions on Cloud Computing

Algorithm 1 Build ETF

Input: Select secret key K_T for PRF F, keyword set of the DPI rule $\mathcal{R}$, t hash functions $\{h_1, ..., h_t\}$, binary vector $BF[]$ whose length is m.
Output: ETF.
Set $BF[\cdot] = 0$.
for all keywords $k \in \mathcal{R}$ **do**
  for $i \in [t]$ **do**
    Compute $\widehat{k} = F(K_T, k)$.
    Compute $loc = h_i(\widehat{k}) \ mod \ m$, $BF[loc] = 1$.
  end for
end for
Set ETF = BF.
Output ETF.

of Tset is similar to OXT [26]. The ID of each DPI rule is denoted as rid. For the rules that contains the same keyword k ($\mathrm{rid} \in \mathcal{R}(k)$), the IDs of the rules will be attached to the list $\mathbf{t}$. Note that, PRF F_p can map a message to $\mathbb{Z}_p^*$, where p is a large prime number. $\mathrm{TsetSetup}$ also outputs a secret key K_R, that is used to generate search tokens for Tset. The implementation details of Tset can be found in [26]. The second part is an extension method of all the DPI rules. The main purpose of this operation is to hide the number of keywords in each rule. Therefore, we choose to extend it to the same size. Suppose the largest number of keywords in a single rule is n. Given any rule that contains o keywords, $n - o$ dummy keywords will be added. The dummy keywords are generated by embedding real keywords with the sequence numbers. The extended encoded keywords set is denoted as $\mathcal{R}^*$. In the third part, we map each encoded keyword $\widehat{k}$ into a Bloom filter and encrypt it as $\widehat{BF}$ using SHVE. In this step, the relationship between each encoded keyword $\widehat{k}$ and the rule identifier rid is build and encrypted. At last, the Algorithm. 2 returns ERS= $(\mathrm{Tset}, \widehat{BF})$. GW uploads RMS and the secret keys $\{K_R, K_D, K_X, mk\}$ to TFS.

• Packet processing:

Packets sent from the internal network are all converged to GW. Firstly, GW tokenizes the packet payload using delimiter-based scheme [2]. Delimiters could be any symbols such as space, comma, connector, etc. For instance, given a string "login.net?user=Allen", the typical tokens could be "login.", "login.net", "user=Allen", etc. Formally, given a long string (i.e., the packet payload), it is segmented as follows. Let $\{S_1, S_2, ..., S_m\}$ be all the non-delimiter strings and $\{P_1, P_2, ..., P_n\}$ be all the delimiters. They both appear in the order of their indexes. Then, the algorithm scans the payload and finds the first delimiter P_1. If the S_2 appears after P_1, the generated tokens are $\{S_1, S_1P_1, S_1P_1S_2\}$. Then, the algorithm repeats the same method for P_2. If P_2 just appears after P_1, it continues to find S_3. Then, the generated tokens are $\{S_1, P_1P_2, , S_1P_1P_2\}$, and the link of all the strings from P_2 to S_3 (S_3 is not included). The algorithm can segment the whole payload recursively using this method. Secondly, each token tk is encoded as $\mathrm{etk} = F(K_T, \mathrm{tk})$. Finally, all the encoded tokens and the packets are sent to MB (TFS).

Algorithm 2 Build ERS

Input: Secret keys K_T, K_S, K_D for PRF F, K_I, K_Z, K_X for PRF F_p, K_E for symmetric encryption Σ, mk for SHVE. Keyword set of the DPI rule $\mathcal{R}$. t hash functions $\{h_1, ..., h_t\}$. Binary vector $BF[]$ whose length is m.
Output: ERS.

(a). Build an index for all the keywords using Tset:
for all keywords $k \in \mathcal{R}$ **do**
  Set list $\mathbf{t} = \{\}$.
  Compute $\widehat{k} = F(K_T, k)$, $K_E = F(K_S, \widehat{k})$.
  $\mathcal{R}(k)$: all the rules that include keyword k.
  for $\mathrm{rid} \in \mathcal{R}(k)$ **do**
    Set counter $ct = 1$.
    Compute $\mathrm{xrid} = F_p(K_I, \mathrm{rid})$.
    Compute $z_k = F_p(K_Z, k||ct)$, $y_{ct} = \mathrm{xrid} \cdot z_k^{-1}$.
    Compute $\mathrm{erid}_{ct} = \mathrm{Enc}(K_E, \mathrm{rid})$.
    Set $ct = ct + 1$ and add $(y_{ct}, \mathrm{erid}_{ct})$ to $\mathbf{t}$.
  end for
  Set $\mathbf{T}[\widehat{k}] = \mathbf{t}$.
end for
Run $(\mathrm{Tset}, K_R) \leftarrow \mathrm{TsetSetup}(\mathbf{T})$.

(b). Extend the DPI rules into same size:
for all rules $R \subset \mathcal{R}$ and $\mathcal{R}^* = \varnothing$ **do**
  Parse each rule R as $\{k_1, ..., k_o\}$.
  Compute $\{\widehat{k}_1, ..., \widehat{k}_o\} = \{F(K_T, k_1), ..., F(K_T, k_o)\}$.
  for $i = o + 1; i \leq n; i++$ **do**
    Compute $\widehat{k}_i = F(K_D, \widehat{k}_1||...||\widehat{k}_o||i)$.
  end for
  Set $R^* = \{\widehat{k}_1, ..., \widehat{k}_n\}$ and $\mathcal{R}^* = R^* \cup \mathcal{R}^*$.
end for

(c). Map the extended rules into an encrypted Bloom filter:
Set $BF[\cdot] = 0$.
for all encoded keywords $\widehat{k} \in \mathcal{R}^*$ **do**
  for all $\mathrm{rid} \in \mathcal{R}^*(\widehat{k})$ **do**
    Compute $\mathrm{xrid} = F_p(K_I, \mathrm{rid})$.
    for $j = 1; j \leq t; j++$ **do**
      Compute $loc = h_j(g^{F_p(K_X, \widehat{k}) \cdot \mathrm{xrid}}) \ mod \ m$.
      Set $BF[loc] = 1$.
    end for
  end for
end for
Compute $\widehat{BF} = \mathrm{SHVE.Enc}(mk, \text{"True"}, BF)$.

Return ERS= $(\mathrm{Tset}, \widehat{BF})$.

• Fast token filtering:

TFS uses ETF to filter out most unmatched tokens. Specifically, given an encoded token etk, TFS computes t hash functions as:

$$loc_1 = h_1(\mathrm{etk}) \ mod \ m, ..., loc_t = h_t(\mathrm{etk}) \ mod \ m.$$

Then, TFS checks whether $\mathrm{ETF}[loc_i] = 1$ holds, where $1 \leq i \leq t$. If so, token etk can be asserted to be matched. TFS continues to check the remain tokens until all the encoded tokens of a packet are inspected. If there is no token matched, the packet should be forwarded to external

server as usual. Otherwise, TFS continues to conduct exact rule match with the help of RMS.

• **Exact rule matching:**

In this phase, TFS should generate search trapdoors using matched tokens. Then TFS sends them to RMS to find out the final rule matching results. Suppose a series of matched tokens $\{\mathrm{etk}_1, ..., \mathrm{etk}_o\}$ appear one after the other. Then, $\{\mathrm{etk}_1, ..., \mathrm{etk}_o\}$ will be encrypted as a trapdoor that supports conjunctive keywords matching [16], [26] over ERS. The details are shown as follows:

Step 1: Given a series of matched tokens $\{\mathrm{etk}_1, ..., \mathrm{etk}_o\}$. TFS randomly chooses an encoded token as the main token. Without loss of generality, we let etk_1 be the main token. Then, TFS computes $\mathrm{stok} = \mathrm{TsetGenTag}(K_R, \mathrm{etk}_1)$ and sends stok to RMS.

Step 2: RMS first parses ERS as $(\mathrm{Tset}, \widehat{BF})$. Then, RMS invokes $\mathbf{t} = \mathrm{TsetSearch}(\mathrm{stok}, \mathrm{Tset})$ to obtain the search result $\mathbf{t}$ of the main token. If $\mathbf{t}$ is empty (no rule is matched), RMS let TFS transmit the packet to external server. And TFS should continue to process the next packet. Otherwise, the number of matched rules $|\mathbf{t}|$ will be returned to TFS.

Step 3: Upon receiving $|\mathbf{t}|$, TFS extends the tokens as $\mathrm{etk}_i = F(K_D, \mathrm{etk}_1||...||\mathrm{etk}_o||i)$, where i ranges from $o+1$ to n with step length 1. Then, for ct ranges from 1 to $|\mathbf{t}|$ and j ranges from 2 to n, both with step length 1; TFS computes

$$\mathrm{xtok}[ct, j] = g^{F_p(K_Z, \mathrm{etk}_1||ct) \cdot F_p(K_X, \mathrm{etk}_j)},$$

and sends $\mathrm{xtok}[ct] = \{\mathrm{xtok}[ct, 2], ..., \mathrm{xtok}[ct, n]\}$ to RMS.

Step 4: For ct ranges from 1 to $|\mathbf{t}|$ and j ranges from 2 to n, both with step length 1; RMS parses the result pairs $(y_{ct}, \mathrm{erid}_{ct})$ from $\mathbf{t}$ and computes:

$$\mathrm{xtag} = \mathrm{xtok}[ct, j]^{y_{ct}},$$

$$loc_{ct,j,1} = h_1(\mathrm{xtag}) \ mod \ m, ..., loc_{ct,j,t} = h_t(\mathrm{xtag}) \ mod \ m.$$

Then, RMS returns $\{loc_{ct,j,1}, ..., loc_{ct,j,t}\}$ to TFS.

Step 5: For ct ranges from 1 to $|\mathbf{t}|$ and j ranges from 2 to n, both with step length 1; TFS initializes $\mathbf{v}_{ct} = *^m$ and sets $\mathbf{v}_{ct}[loc_{ct,j,1}], ..., \mathbf{v}_{ct}[loc_{ct,j,t}] = 1$. Afterwards, TFS invokes

$$\mathrm{vxtok}_{ct} = \mathrm{SHVE.KeyGen}(mk, \mathbf{v}_{ct}).$$

Finally, TFS sends $\{\mathrm{vxtok}_1, ..., \mathrm{vxtok}_{|\mathbf{t}|}\}$ to RMS.

Step 6: Upon receiving $\{\mathrm{vxtok}_1, ..., \mathrm{vxtok}_{|\mathbf{t}|}\}$, for ct ranges from 1 to $|\mathbf{t}|$, RMS invokes

$$\eta_{ct} = \mathrm{SHVE.Query}(\mathrm{vxtok}_{ct}, \widehat{BF}).$$

Let $\Gamma = \{\}$ be the result set. If η_{ct} ="True", the corresponding DPI rule is exactly matched. RMS should trigger the action and set $\Gamma = \Gamma \cup \{\mathrm{erid}_{ct}\}$. The matched encrypted rule set Γ along with the uploaded trapdoors should be returned to GW for further analysis. If no η_{ct} equals to "True", packet will be transmitted to external server.

Discussion: The second layer of EV-DPI can support exactly rule matching using the technique of conjunctive searchable encryption [16]. As we know, compared to the previously proposed OXT [26], the used method in this paper introduces an additional communication round between the two cloud servers. Here, we argue that it is worthy to do so. First, OXT discloses the result pattern, which indicates the intermediate results (the search result of the partial of the keywords). This information leakage can be used to recover the search keywords [27], [28]. Thus, the cloud server is required to return the final result directly. Second, through the experiment, we find that few of the tokens (near 0.1%) are matched. And only matched tokens introduces one more round of communication. So, the extra delay is worthy for the benefit of less information leakage.

3.2 Execution Result Verification

As discussed in Section 2.2, the cloud server is possible to be "Lazy". Thus, the correctness of the returned result should be verified. In specific, the motivation of offering verification service to the GW (middlebox owner) are mainly two-folds.

First, in the reality, most of the packets will not be matched to any rule. Thus, the cloud server can just forward parts of the packets without inspection. It is also difficult for GW to detect such irresponsible behavior. Concretely, if 30% of the packets are not inspected, then the cloud service provider can save 30% of the computing resources. Thus, the verification is necessary to resist such "Lazy" cloud servers.

Second, any outsourced middlebox is the potential attack target of both internal and external attackers [5], [29]. It should be the cloud service provider's responsibility to resist the attacks. Thus, the cloud server needs to use the verification objects to prove that it strictly follows the protocols. Moreover, the verification offers a full control of the outsourced middlebox. GW can also measure the quality of the cloud computing service through verifying the inspection results.

In this paper, the fast filtering processing accomplished on TFS, and the exact rule matching conducted on RMS should all be verified by GW.

• **Verification on** TFS:

We use Bloom filter [19] to support the membership checking of the encoded tokens. And only fast hashing operations are conducted. Thus, the verification of the returned result is simple. If a token is asserted to be matched, the encoded token itself is the *verification objective* (VB). For an unmatched token, VB should be the encoded token and the hash functions that maps the token into the positions with value 0. Afterwards, VBs are stored on *verification space*. To verify the matched tokens, GW redo the hashing operations which is exactly the same as the filtering process. If all the mapped positions of the Bloom filter are set as 1, GW will *accept* the result. Otherwise, the result is incorrect and should be *rejected*. To verify the unmatched tokens, only one-time hashing is required. If the mapped position has the value 0, the result can be accepted. Otherwise, GW rejects the result and sends alert to TFS.

• **Verification on** RMS:

The verification of exact rule matching on RMS are accomplished with two steps. The first step is verifying the correctness of the search results on Tset and the second step is verifying the query results on $\widehat{BF}$. In the view of searchable encryption (SE) [30], [31], the first step is to verify the result of single keyword search on Tset. Here, we adopt

a latest non-dictionary verifiable SE scheme [18] to support such function. The details are shown as follow.

GW generates hash tables that consists of all the possible verification objectives and outsources them to RMS. For all the keywords $k \in \mathcal{R}$, we select a secret key K_V for PRF F. Then, GW computes $\widehat{k} = F(K_T, k)$ and $\widetilde{k} = F(K_V, 0||\widehat{k})$. Let $|\mathcal{R}|$ be the number of keywords; (h_1, h_2) be the hash functions used for building hash tables. Then, GW maps all the keywords $\mathcal{R}$ into two cuckoo hash tables (T_1^*, T_2^*) with size $|\mathcal{R}| + 1$. For each $\widetilde{k}$, we have $T_1^*[h_1(\widetilde{k})] = \widetilde{k}$ or $T_2^*[h_2(\widetilde{k})] = \widetilde{k}$. For $x = 1, 2$, i ranges from 1 to $|\mathcal{R}| + 1$ with step 1, if $T_x^*[i] = \widetilde{k}$, GW calculates

$$T_x[i] = \{\widetilde{k}, F(K_V, x||i||\widetilde{k}), F(K_V, 3||\widetilde{k}||\mathcal{R}(k))\};$$

otherwise, GW computes

$$T_x[i] = \{null, F(K_V, x||i||null), null\}.$$

Therefore, for each $\widetilde{k}$, one of the following two equations should hold.

$$\begin{cases} T_1[h_1(\widetilde{k})] & = \{\widetilde{k}, F(K_V, 1||h_1(\widetilde{k})||\widetilde{k}, F(K_V, 3||\widetilde{k}||\mathcal{R}(k)\}, \\ T_2[h_2(\widetilde{k})] & = \{\widetilde{k}, F(K_V, 2||h_2(\widetilde{k})||\widetilde{k}, F(K_V, 3||\widetilde{k}||\mathcal{R}(k)\}. \end{cases}$$

At last, GW shares the secret key K_V with TFS and outsources $\{T_1[\cdot], T_2[\cdot]\}$ along with $\{h_1, h_2\}$ to RMS.

To support verification on RMS, each main token etk (i.e., token used to search on Tset) should be encoded again as $\widetilde{\mathrm{etk}} = F(K_V, \mathrm{etk})$. Afterwards, $\widetilde{\mathrm{etk}}$ is sent to RMS. Let $\mathbf{t}$ be the search result on Tset. RMS searches the two hash tables as:

$$\{a_1, b_1, c_1\} = T_1[h_1(\widetilde{\mathrm{etk}})]; \;\; \{a_2, b_2, c_2\} = T_2[h_2(\widetilde{\mathrm{etk}})].$$

If $\widetilde{\mathrm{etk}} = a_1$, set $\mathrm{pf} = c_1$ and if $\widetilde{\mathrm{etk}} = a_2$, set $\mathrm{pf} = c_2$. Otherwise, $\mathrm{pf} = \{a_1, b_1, a_2, b_2\}$. Then, RMS outputs the verification objective $\mathrm{VB} = \{\widetilde{\mathrm{etk}}, \mathbf{t}, \mathrm{pf}\}$ of the searched encoded token etk and stores it on the verification space.

To verify the search result of etk, GW first checks whether $\mathrm{pf} = c_1$ or $\mathrm{pf} = c_2$. If so, it continues to check if $\mathrm{pf} = F(K_V, 3||\widetilde{\mathrm{etk}}||\mathbf{t})$ holds. If this equation holds, GW accepts the result. Otherwise, the result should be rejected. If $\mathrm{pf} = \{a_1, b_1, a_2, b_2\}$ and $\widetilde{\mathrm{etk}} \in \{a_1, a_2\}$ or $\mathbf{t} \neq \varnothing$ or $b_1 \neq F(K_V, 1||h_1(\widetilde{\mathrm{etk}})||a_1)$ or $b_2 \neq F(K_V, 2||h_2(\widetilde{\mathrm{etk}})||a_2)$, the result should be rejected; otherwise, accepted.

Note that, the search results on $\widehat{BF}$ are verified by GW. To boost the efficiency, TFS first links the remain $n - 1$ tokens as one token. Then, it is encoded using PRF F with secret key K_V. Thus, only one-time verification is required. Moreover, the generation of the hash tables and VBs are exactly the same as the main keyword. Here, we omit the details.

4 Security Analysis

In this section, we discuss the security and privacy properties of EV-DPI according to the threat model and design goals presented in Section 2. In specific, two issues are considered. One is the privacy of packet payload and the other is the confidentiality of DPI rules.

•Privacy of packet payload:

The content of packet payload is tokenized and encoded using PRF F with secret key K_T. In the view of TFS, the encoded tokens are randomized. Thus, it is hard to be recovered by a polynomial-time attacker according to the security property of PRF. For the matched tokens, TFS generates stags and xtoks using four secret keys $\{K_R, K_D, K_Z, K_X\}$. These four secret keys are kept private from RMS. Therefore, the generation of stags and xtoks has further promoted the security level of encoded tokens. In the view of RMS, no additional information can be deduced from received stags and xtoks. Note that, RMS cannot access the original encoded tokens. Moreover, the packet payload itself can be encrypted by using any secure and efficient end-to-end encryption method such as AES [2]. Thus, both TFS and RMS cannot infer the plaintext of passing encrypted packet payload. In a word, the privacy of packet payload is well-preserved by using PRFs.

•Confidentiality of DPI rules:

The confidentiality of outsourced DPI rules directly relies on the design details of two encrypted filters ETF, ERS and the Cuckoo hash tables generated for result verification. We discuss the confidentiality of ETF, ERS and the Cuckoo hash tables successively.

1). ETF is constructed using standard Bloom filter [19]. If the plaintext of each keyword is directly mapped into the Bloom filter, an attacker can infer the data distribution through large-scale membership test. In EV-DPI, each keyword is first encoded using PRF F with the secret key K_T. By doing so, the distribution of data is concealed. Therefore, it becomes difficult for RMS to recover the original data. Moreover, similar construction method used in [32] has been theoretically and practically proved to be semantic secure.

2). ERS can be parsed as ERS$= (\mathrm{Tset}, \widehat{BF})$. Thus, the confidentiality of DPI relies on the instantiation approach of Tset and the encryption method SHVE used for $\widehat{BF}$. In EV-DPI, we follow the instantiation method of Tset presented in [26], which is proofed to be secure. SHVE is constructed using PRFs and secure symmetric encryption. Formally, in [16], the authors proved that SHVE is selectively simulation-secure. Therefore, $\widehat{BF}$ can preserve the confidentiality of the dataset (i.e., xtags). Note that, TFS cannot access $\mathrm{Tset}, \widehat{BF}$. Moreover, the final matching result is also kept secret from TFS. Thus, it is hard for TFS to recover the encrypted DPI rules through searching huge number of tokens and analyzing the search result.

3). Three elements in Cuckoo hash tables $\{T_1[\cdot], T_2[\cdot]\}$ are all encoded by PRF F with secret key K_V. And only GW is allowed to access K_V. Thus, RMS cannot distinguish the elements in $\{T_1[\cdot], T_2[\cdot]\}$ without K_V. The verification objects stored on TFS leaks no additional information. Therefore, the proposed verification scheme can well conceal the original DPI rules.

In summary, the confidentiality of outsourced DPI rules is well preserved.

Discussion: Here, two critical security issues related to EV-DPI are discussed.

1). As shown in Algorithm 2, some dummy keywords are added into each rule. Thus, in the *exactly rule matching* phase, each query and rule are extended to the same size. RMS

then cannot distinguish two queries or two rules through the number of tokens or keywords. Also, the size of each encrypted rule is exactly the same. Therefore, the cloud server is unable to distinguish an individual rule through this side channel information.

2). Another important issue is the randomness of encoded tokens and keywords. Since PRFs provides no randomness over the same inputs, the equality between different encoded tokens and keywords are revealed. To address this issue, there are two existing methods [2], [10] are proposed. a). The first one is presented by Sherry *et al.* [2]. They construct two synchronization tables over the middlebox and the packet sender. In the table, each entry is a counter for a token. And the counters in two tables are updated simultaneously when a token appears repeatedly. Then, each token is encoded together with the counter. Therefore, both packet sender and middlebox have to construct and maintain two tables. Moreover, the message space of tokens could be large, which will definitely introduce large tables. b). To mitigate this problem, Yuan *et al.* [10] choose to trade storage for the efficiency. The main idea is to attach an integer number behind the token or keyword before encoding. Thus, each token has multiple encoded versions. All the versions will be used repeatedly. The only cost is that the size of DPI rule has to be extended. According to the analysis over real dataset, the authors find that 3 versions are adequate. However, this method still cannot provide real or even pseudo randomness. We argue that this could be an open problem of symmetric key based schemes.

5 Performance Evaluation

In this section, a comprehensive evaluation on the performance of EV-DPI is given. In specific, the functionality and efficiency are discussed. We also compare the experiment results with the latest proposed scheme GWJ [13] which is also designed based on symmetric key encryption.

•Functionality:

First, EV-DPI supports basic DPI rule matching over ciphertext domain. It is also supported in previously proposed schemes [2], [10], [14], [33]. Second, EV-DPI can return the matching details to GW for further analysis. Such as the encrypted rule identification, intermediate result, etc. Third, token-level verification is supported that allows GW to verify the matching result of each passing token. Thus, EV-DPI has achieved the comprehensive design goals in terms of functionality.

EV-DPI adopts a two-layer inspection architecture. The first layer fast filters out suspicious packets. The second layer could be enabled to decrypt the suspicious packets and inspect the packets in the plaintext domain. For instance, if some tokens or basic rules are matched, then such packet should be considered as *dangers* packet. The confidentiality of this packet will not be protected. Thus, the middlebox can decrypt it and perform complex inspection rules in the plaintext domain. This method is demoted as *probable cause privacy* [2]. By dong so, the basic DPI functions is supported and the privacy is preserved. Moreover, the advanced functions, including complex regular expression and machine learning [34] based detection can all be well supported.

•Implementation details:

The gateway is simulated by a local server. It offers Intel Xeon E5-2620 CPU (2.1GHz) and 32GB memory. The middlebox is simulated on the Amazon Cloud with instance "c4.4xlarge". This instance provides Intel Xeon E5-2666 CPU (2.9GHz) and 30GB memory. Through the test, the maximum network throughput of this instance is nearly 4.9 Gbit/s. Since EV-DPI supports parallel processing, the middlebox also can be simulated by multiple cloud servers. For instance, if the function of TFS is implemented over multiple servers, the inspection efficiency can be significantly improved. Especially when larger-scale of packets and connections need to be processed. The communication between gateway and middlebox is not simulated in the real world. The additional communication cost is mainly brought by the encoded tokens generated by gateway. These tokens will be sent from gateway to middlebox. Thus, we can directly compute the size of the total tokens based on the implementation details of PRF. The operation system is Linux and the programming language is Java. To boost the efficiency, the multi-threading technique is applied for every phase that supports parallel processing. The lightweight public instruction detection rule set Snort [22] is adopted. The public intrusion detection evaluation dataset DARPA [35] is used. We use OpenSSL to implement cryptographic tools. The HMAC-SHA2 is used to instantiate PRFs and the output is truncated as 128 bits. AES-128 is adopted as the symmetric encryption.

•Efficiency evaluation on GW:

Initialization overhead:

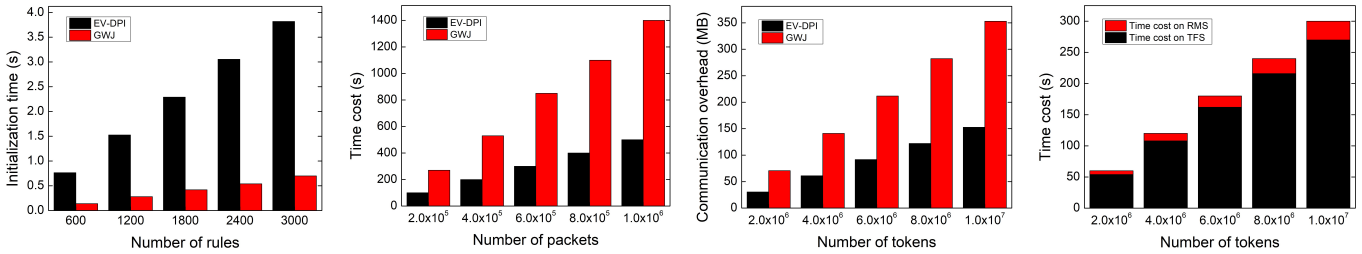
In EV-DPI, the initialization phase can be divided into three parts. That are the construction of encrypted filter ETF, the index ERS and the Cuckoo hash tables $\{T_1[\cdot], T_2[\cdot]\}$. As shown in Fig. 4a, the time costs of EV-DPI and GWJ [13] increase linearly with the number of rules. When the number of rules reaches 3000, the total cost of EV-DPI is roughly 3.82s. The construction of ERS takes nearly 3.1s while ETF and $\{T_1[\cdot], T_2[\cdot]\}$ only cost 0.23s and 0.49s, respectively. As shown in Alg. 2, modular exponential operations are conducted to build $\widehat{BF}$. Thus, the computation of ERS costs much more time than ETF and $\{T_1[\cdot], T_2[\cdot]\}$. Since only PRFs and symmetric encryption are used, GWJ requires totally 0.72s to build the index. Both schemes let GW tokenize and encode the packet payload. Thus, only one-time initialization is required (i.e., constant). Thus, we argue that 3 seconds additional cost is acceptable.

Packet processing time:

Packets gathered by GW are all tokenized and encoded before being uploaded to MB. For each token, EV-DPI only requires one-time PRF calculation while GWJ needs to conduct two-time of PRF calculation and one-time symmetric encryption. Therefore, the total packet processing time is significantly reduced. As shown in Fig. 4b, with the increasing of the number of packets, the time costs of EV-DPI and GWJ increase in linear. When the number of packets reaches 10^6, the time cost of EV-DPI and GWJ are roughly 500s and 1400s, respectively.

Communication overhead:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2991167, IEEE Transactions on Cloud Computing



(a) Initialization time. (b) Packet processing time. (c) Communication overhead on GW. (d) Verification time.

Fig. 4: Performance Comparison on GW side.

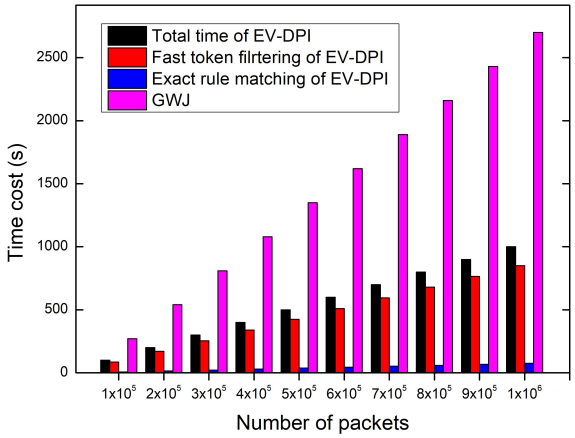


Fig. 5: Packet throughput of MB.

The communication overhead between GW and MB depends on the size of encoded tokens. It is determined by the encoding method. For EV-DPI, each token is encoded by a PRF with 128 bits (truncated) output. For GWJ, the encoded token consists of three parts. In specific, two parts are computed using PRF and one part is the ciphertext of the token. GWJ uses the same HMAC and truncates the output as 128 bits. As shown in Fig. 4c, the costs of EV-DPI and GWJ are 152.6MB and 352.9MB, respectively, when the number of tokens is 10^7. In practice, the communication latency is a crucial factor that significantly affects the whole performance of the system. Since EV-DPI produces smaller encoded tokens, it can achieve lower communication latency.

Verification time:

Since EV-DPI is able to support inspect result verification on both two cloud servers, we evaluate the efficiency respectively. For each server, the whole verification process consists of two parts. One is the generation of verification objects (VBs) and the other is verifying the result based on VBs. As shown in Fig. 4d, GW costs much more time to verify the result returned by TFS than RMS. Because the matching result of every token on TFS should be verified but only about 0.1% packets are matched on RMS. Roughly, GW takes 10% time to verify the result returned by RMS. When the number of tokens reaches 10^7, the total time cost is 300s. Note that, the total cost includes the generation of VBs, the interaction latency between GW and MB, and the verification result revealing.

•Efficiency evaluation on MB:

We evaluate the packet throughput of MB to prove the efficiency, which is the key performance metric. To inspect one packet, all the encoded tokens are first fast filtered by TFS. Then, to support exact rule matching, additional computation and interaction with RMS are required. Since the membership test time for one token over the Bloom filter is constant, the whole-time cost of inspection over MB is significantly boosted. The reason is that only very few tokens are matched on TFS. From the result of experiment, we find that roughly 7.5% time is consumed for exact rule matching. And the proportion of two parts are clearly shown in Fig. 5. To inspect a large number of packets that reach one million, the whole time cost of EV-DPI is 997s. In order to support rule update, GWJ [13] needs to check the rules one by one for each token. So, the time cost of one token matching increases linearly with the number of DPI rules. It takes 2717s to inspect one million packets. On average, the total latency for single packet inspection of EV-DPI is 2.1ms and GWJ is 3.7ms. Thus, EV-DPI has improved the packet throughput and reduced the latency.

Conclusion: Here, we give a conclusion of the *main differences* between the baseline scheme GWJ [13] and EV-DPI. The first difference is the *functionality*. GWJ has considered the issue of rule update. The authors utilize two non-collusion servers to support the update of DPI rules. We acknowledge that this function is useful and it deserves further study. In EV-DPI, we target on the inspection result verification, which offers a strong control of outsourced middlebox. The second difference is the *initialization cost*. GWJ has only built one encrypted filter for all the DPI rules. In EV-DPI, two encrypted filters are built. One is used for fast token filtering and the other is used for exact rule matching. Therefore, EV-DPI causes more initialization costs than GWJ. Note that, both schemes only require one-time initialization for all the connections. In anther word, the cost is irrelevant to the volume of network traffic. The third difference is the *token generation*. One encoded token in GWJ contains three parts. In EV-DPI, one encoded token only contains one part. Thus, EV-DPI has reduced the token generation and communication costs. The fourth difference it the *inspection efficiency*. EV-DPI can filter out most of the regular packets efficiently in the first place. Thus, EV-DPI has improved the inspection efficiency.

Discussion: The accuracy of the inspection result is an important issue that should be considered. In specific,

inspection accuracy is the ratio of packets that should be matched to the given DPI rule, yet they are wrongly regarded as regular packets. In the proposed EV-DPI, if a packet matches a DPI rule, the packet will be captured in the second layer inspection. It is guaranteed by the adopted searchable encryption technique [16]. However, EV-DPI cannot support all the DPI rules [22] in the plaintext domain. For EV-DPI, the fraction of addressable rules in the ciphertext domain is 67%, which is the same as BlindBox [2] and GWJ [13]. It is worthy to note that EV-DPI can increase the overall ratio of addressable rules if *probable cause privacy* is well supported. Thus, we acknowledge that how to efficiently support *probable cause privacy* is an important and challenging future work.

6 Related Work

In this section, we give a brief review of previously proposed schemes that are closely related to this paper. Considerable amount of works is presented towards diverse secure and privacy-preserving network functions for outsourced middleboxes [36]. Roughly, they can be divided into two categories. The privacy-preserving DPI schemes [2] can inspect the packet payload and the secure header matching schemes [37] can detect the packet header. And all the works [5] explore diver functions over ciphertext domain. We also discuss the topic of searchable encryption [38] used in the second layer of EV-DPI.

•Privacy-preserving DPI:

Symmetric key based schemes: The first privacy-preserving DPI scheme named as BlindBox is proposed in [2]. BlindBox provides a definition of the threat model and the system model to lay a foundation of this topic. It assumes that there are two connections between the packet sender and the receiver. One is the traditional TCP connection and the other is a virtual connection used for payload inspection. The payload is firstly tokenized and then encrypted using AES. And the DPI rules are obfuscated using garbled circuit [9]. Yuan *et al.* [10] utilize broadcast encryption [11] to control the membership of the network. Thus, all the authorized users can share the same encrypted DPI rules. They also leverage Cuckoo hash [17] to fast filter the passing tokens. Yuan *et al.* [12] also explore anther important issue, that is the verification of the DPI execution results. In [12], the verification technique ringer [5] is adopted. The authors extend the basic scheme to support multiple middleboxes, which has the potential to support service function chain [39]. Lan *et al.* directly use searchable encryption scheme without any modification to implement privacy-preserving DPI. Thus, it may suffer from higher packet latency. Recently, Guo *et al.* [13] achieve dynamic DPI over two non-collusion cloud servers, which allows the middlebox to update the encoded DPI rules. The system and threat models of this scheme are similar to EV-DPI. EV-DPI has not only improved the efficiency, but also provided fine-grained result verification.

Public key based schemes: The first public key based privacy-preserving network function outsourcing scheme is presented by Melis *et al.* [23]. The somehow homomorphic encryption (BGN) [40] is used to support additive and one-time multiplicative homomorphic operations over the ciphertext domain. Our previously proposed scheme [15] also utilize BGN to process DPI and machine learning based malware detection. Similarly, Fan *et al.* [8] also leverage homomorphic encryption to support DPI as well as malware detection. Recently, a scheme based on decryptable searchable encryption [41] named as BlindIDS is presented by Canard [14]. BlindIDS allows the packet receiver directly decrypt the received tokens to verify that whether the tokens are generated and encrypted correctly.

•Secure header matching:

The first secure header matching is proposed in [33] by Lan *et al.* The authors design a new algorithm named as PrefixMatch to support the prefix matching over ciphertext domain. PrefixMatch is implemented in [33] and is proved to be quite efficient. Guo *et al.* [37] convert the header matching problem into privacy preserving range query [42], [43] over ciphertext domain. The numerical ranges in the rules are encrypted using order-revealing encryption (ORE) [44]. The right ciphertext of ORE can achieve semantic secure under chosen plaintext attack. Guo *et al.* [13] also leverage ORE to support stateful firewall, which lays a foundation for the real deployment of header matching based network functions.

•Searchable encryption:

As pointed out by Lan *et al.* [33], searchable encryption (SE) [38] can be adopted to support privacy preserving DPI. Moreover, the recent advancement of secure range query [25] shows that SE can also be adapted to support semantic secure and efficient range query. Thus, we argue that the SE techniques have the potential for processing the diverse outsourced network functions. SE was originally designed for keyword search [30] over ciphertext domain. With the fast development of this area, many useful properties such as multi-keyword search [45], conjunctive keyword search [16], [26] and even privacy preserving machine learning [46] are supported. In this paper, we adapt the scheme presented in [16] to implement the second layer of EV-DPI. In [16], the information leakage of the intermediate search result (result pattern) is formalized and concealed.

7 Conclusions

In this paper, we have proposed an efficient verifiable deep packet inspection (EV-DPI) scheme with privacy preservation. EV-DPI can well support the verification over final and intermediate inspection results. Both inspection and verification protocols are able to preserve the privacy of packet payload and confidentiality of DPI rules. We have demonstrated the high performance of EV-DPI through extensive experiments and compared the results with the existing scheme. In the future, we will explore the blockchain techniques and learning-based approach to secure diverse outsourced middlebox services.

Acknowledgement

We thank the anonymous reviewers for their important comments to improve the quality of this paper. This work is supported by the National Key R&D Program of China

under Grants 2017YFB0802300 and 2017YFB0802000, the National Natural Science Foundation of China under Grants 61972454,61802051, 61772121, 61728102, and 61472065, the Peng Cheng Laboratory Project of Guangdong Province PCL2018KP004, the Guangxi Key Laboratory of Cryptography and Information Security under Grant GCIS201804, the NSERC, Canada, the China Scholarship Council (CSC) NO. 201706070048. Special thanks should be given to the professor and research colleagues of BBCR Lab at University of Waterloo for the valuable discussion.

REFERENCES

[1] Y. Kanizo, O. Rottenstreich, I. Segall, and J. Yallouz, "Designing optimal middlebox recovery schemes with performance guarantees," *IEEE JSAC*, vol. 36, no. 10, pp. 2373–2383, 2018.

[2] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "BlindBox: Deep packet inspection over encrypted traffic," in *Proc. of ACM SIGCOMM*, 2015, pp. 213–226.

[3] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. Shen, "Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing," *IEEE TCC*, 2019, doi:10.1109/TCC.2019.2903240.

[4] X. Liu, R. Deng, K. R. Choo, and Y. Yang, "Privacy-preserving outsourced support vector machine design for secure drug discovery," *IEEE TCC*, 2018, doi:10.1109/TCC.2018.2799219.

[5] C. Wang, X. Yuan, Y. Cui, and K. Ren, "Toward secure outsourced middlebox services: Practices, challenges, and beyond," *IEEE Network*, vol. 32, no. 1, pp. 166–171, 2018.

[6] N. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, and X. Shen, "Space/Aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE JSAC*, vol. 37, no. 5, pp. 1117–1129, 2019.

[7] M. Hao, H. Li, X. Luo, G. Xu, H. Yang, and S. Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE TII*, 2019, doi:10.1109/TII.2019.2945367.

[8] J. Fan, C. Guan, K. Ren, Y. Cui, and C. Qiao, "SPABox: Safeguarding privacy during deep packet inspection at a middlebox," *IEEE/ACM ToN*, vol. 25, no. 6, pp. 3753–3766, 2017.

[9] E. M. Songhori, S. U. Hussain, A. Sadeghi, T. Schneider, and F. Koushanfar, "TinyGarble: Highly compressed and scalable sequential garbled circuits," in *Proc. of IEEE S&P*, May 2015, pp. 411–428.

[10] X. Yuan, X. Wang, J. Lin, and C. Wang, "Privacy-preserving deep packet inspection in outsourced middleboxes," in *Proc. of IEEE INFOCOM*, 2016, pp. 1–9.

[11] T. V. X. Phuong, G. Yang, W. Susilo, and X. Chen, "Attribute based broadcast encryption with short ciphertext and decryption key," in *Proc. of ESORICS*, 2015, pp. 252–269.

[12] X. Yuan, H. Duan, and C. Wang, "Bringing execution assurances of pattern matching in outsourced middleboxes," in *Proc. of IEEE ICNP*, 2016, pp. 1–10.

[13] Y. Guo, C. Wang, and X. Jia, "Enabling secure and dynamic deep packet inspection in outsourced middleboxes," in *Proc. of ACM SCC*, 2018, pp. 49–55.

[14] 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. of ACM AsiaCCS*, 2017, pp. 561–574.

[15] H. Li, H. Ren, D. Liu, and X. Shen, "Privacy-enhanced deep packet inspection at outsourced middlebox," in *Proc. of WCSP*, 2018, pp. 1–6.

[16] S. Lai, S. Patranabis, A. Sakzad, J. K. Liu, D. Mukhopadhyay, R. Steinfeld, S.-F. Sun, D. Liu, and C. Zuo, "Result pattern hiding searchable encryption for conjunctive queries," in *Proc. of ACM CCS*, 2018, pp. 745–762.

[17] G. Levy, S. Pontarelli, and P. Reviriego, "Flexible packet matching with single double cuckoo hash," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 212–217, 2017.

[18] W. Ogata and K. Kurosawa, "Efficient no-dictionary verifiable searchable symmetric encryption," in *Proc. of IFCA FC*, 2017, pp. 498–516.

[19] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[20] Y. Zhang, C. Xu, X. Lin, and X. Shen, "Blockchain-based public integrity verification for cloud storage against procrastinating auditors," *IEEE TCC*, 2019, doi:10.1109/TCC.2019.2908400.

[21] "Amazon cloud," https://aws.amazon.com/cn/, 2019, [Online, accessed 27-March-2019].

[22] "Snort rules," https://www.snort.org/, 2019, [Online, accessed 07-March-2019].

[23] L. Melis, H. J. Asghar, E. De Cristofaro, and M. A. Kaafar, "Private processing of outsourced network functions: Feasibility and constructions," in *Proc. of ACM SDN-NFV Security*, 2016, pp. 39–44.

[24] Y. Zhang, C. Xu, H. Li, K. Yang, J. Zhou, and X. Lin, "Healthdep: An efficient and secure deduplication scheme for cloud-assisted ehealth systems," *IEEE TII*, vol. 14, no. 9, pp. 4101–4112, 2018.

[25] H. Ren, H. Li, Y. Dai, K. Yang, and X. Lin, "Querying in Internet of Things with privacy preserving: Challenges, solutions and opportunities," *IEEE Network*, no. 99, pp. 1–8, 2018.

[26] D. Cash, S. Jarecki, C. Jutla, H. Krawczyk, M.-C. Roşu, and M. Steiner, "Highly-scalable searchable symmetric encryption with support for boolean queries," in *Proc. of CRYPTO*, 2013, pp. 353–373.

[27] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart, "Leakage-abuse attacks against searchable encryption," in *Proc. of ACM CCS*, 2015, pp. 668–679.

[28] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," in *Proc. USENIX Security*, 2016, pp. 707–720.

[29] X. Yuan, H. Duan, and C. Wang, "Assuring string pattern matching in outsourced middleboxes," *IEEE/ACM ToN*, 2018.

[30] H. Li, D. Liu, Y. Dai, T. Luan, and S. Yu, "Personalized search over encrypted data with efficient and secure updates in mobile clouds," *IEEE TETC*, vol. 6, pp. 97–109, 2015.

[31] I. Ghosh Ray, Y. Rahulamathava, and M. Rajarajan, "A new lightweight symmetric searchable encryption scheme for string identification," *IEEE TCC*, 2018, doi:10.1109/TCC.2018.2820014.

[32] S. Kamara and T. Moataz, "Boolean searchable symmetric encryption with worst-case sub-linear complexity," in *Proc. of EUROCRYPT*, 2017, pp. 94–124.

[33] C. Lan, J. Sherry, R. A. Popa, S. Ratnasamy, and Z. Liu, "Embark: Securely outsourcing middleboxes to the cloud," in *Proc. of USENIX NSDI*, 2016, pp. 255–273.

[34] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE TIFS*, vol. 15, no. 7, pp. 911–926, 2020, doi:10.1109/TIFS.2019.2929409.

[35] "DARPA traffic," https://www.ll.mit.edu/r-d/datasets, 2019, [Online, accessed 07-March-2019].

[36] 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. of ACM SIGCOMM*, 2012, pp. 13–24.

[37] Y. Guo, C. Wang, X. Yuan, and X. Jia, "Enabling privacy-preserving header matching for outsourced middleboxes," in *Proc. of IWQoS*, 2018.

[38] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. of IEEE S&P*, 2000, pp. 44–55.

[39] M. Huang, W. Liang, Y. Ma, and S. Guo, "Maximizing throughput of delay-sensitive nfv-enabled request admissions via virtualized network function placement," *IEEE TCC*, 2019, doi:10.1109/TCC.2019.2915835.

[40] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts," in *Proc. of IACR TCC*, J. Kilian, Ed., 2005, pp. 325–341.

[41] T. Fuhr and P. Paillier, "Decryptable searchable encryption," in *Proc. of ProvSec*, 2007.

[42] G. Xu, H. Li, Y. Dai, K. Yang, and X. Lin, "Enabling efficient and geometric range query with access control over encrypted spatial data," *IEEE TIFS*, vol. 14, no. 4, pp. 870–885, 2019.

[43] J. Liang, Z. Qin, S. Xiao, J. Zhang, H. Yin, and K. Li, "Privacy-preserving range query over multi-source electronic health records in public clouds," *Elsevier JPDC*, vol. 135, no. 7, pp. 127–139, 2020.

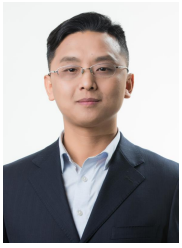
[44] K. Lewi and D. J. Wu, "Order-revealing encryption: New constructions, applications, and lower bounds," in *Proc. of ACM CCS*, 2016, pp. 1167–1178.

[45] Y. Yang, X. Liu, X. Zheng, C. Rong, and W. Guo, "Efficient traceable authorization search system for secure cloud storage," *IEEE TCC*, 2018, doi:10.1109/TCC.2018.2820714.

[46] J. Liang, Z. Qin, S. Xiao, L. Ou, and X. Lin, "Efficient and secure decision tree classification for cloud-assisted online diagnosis services," *IEEE TDSC*, 2019, doi:10.1109/TDSC.2019.2922958.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.2991167, IEEE Transactions on Cloud Computing

Hao Ren (S'14) is currently a Ph.D. candidate at the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), China. He is also a visiting Ph.D. student at ECE department, University of Waterloo, Canada. His research interests include Applied Cryptography, Security and Privacy for Cloud Computing and IoT.



Hongwei Li (M'11-SM'18) is currently the Head and a Professor at Department of Information Security, School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC). He received the Ph.D. degree from UESTC in June 2008. He worked as a Postdoctoral Fellow at the University of Waterloo from Oct. 2011 to Oct. 2012 under the supervision of Prof. Sherman Shen. His research interests include network security and applied cryptography. Dr. Li has published more than 100 technical papers. Dr. Li serves as the Associate Editor of IEEE IoT Journal, and PPNA, the Guest Editor of IEEE Network, IEEE IoT Journal and IEEE Transactions on Vehicular Technology. He also serves/served the technical symposium co-chair of ACM TUR-C 2019, IEEE ICCC 2016, IEEE GLOBECOM 2015 and IEEE BigDataService 2015, and technical program committees for international conferences, such as IEEE INFOCOM, IEEE ICC, IEEE GLOBECOM, IEEE WCNC. He won Best Paper Awards from IEEE MASS 2018 and IEEE HEALTHCOM 2015. He is the Senior Member of IEEE and the Distinguished Lecturer of IEEE Vehicular Technology Society.

Dongxiao Liu (S'13) received his B.S. and M.S. degree in School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), China in 2013 and 2016, respectively. Currently, he is pursuing the PhD degree at 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 (S'15) received his B.S. degree in information and computing science from Anhui University of Architecture in 2014. Currently, he is a Ph.D. student at 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.

Nan Cheng (S'12-M'16) received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo in 2016, and B.E. degree and the M.S. degree from the Department of Electronics and Information Engineering, Tongji University, Shanghai, China, in 2009 and 2012, respectively. He is currently a professor with School of Telecommunication Engineering, Xidian University, Shaanxi, China. He worked as a Post-doctoral fellow with the Department of Electrical and Computer Engineering, University of Toronto, from 2017 to 2018. His current research focuses on space-air-ground integrated system, big data in vehicular networks, and self-driving system. His research interests also include performance analysis, MAC, opportunistic communication, and application of AI for vehicular networks.

Xuemin (Sherman) Shen (M'97-SM'02-F'09) received Ph.D. degree from Rutgers University, New Jersey (USA) in electrical engineering, 1990. Dr. Shen is a University Professor, 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. Dr. Shen is a registered Professional Engineer of Ontario, Canada, an IEEE Fellow, 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.

Dr. Shen is the Editor-in-Chief for IEEE Internet of Thing Journal and the vice president on publications of IEEE Communications Society. He received the Joseph LoCicero Award in 2015, the Education Award in 2017, the Harold Sobol Award in 2018, and the James Evans Avant Garde Award in 2018 from the IEEE Communications Society. He has also received the Excellent Graduate Supervision Award in 2006, and the Outstanding Performance Award in 2004, 2007, 2010, 2014, and 2018 from the University of Waterloo, the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. Dr. Shen served as the Technical Program Committee Chair/Co-Chair for IEEE Globecom' 16, IEEE Infocom'14, IEEE VTC'10 Fall, the Symposia Chair for IEEE ICC'10, the Tutorial Chair for IEEE VTC'11 Spring, the Chair for IEEE Communications Society Technical Committee on Wireless Communications, and P2P Communications and Networking.