SlimBox: Lightweight Packet Inspection over Encrypted Traffic

Qin Liu, Member, IEEE, Yu Peng, Graduate Student Member, IEEE, Hongbo Jiang, Senior Member, IEEE, Jie Wu, Fellow, IEEE, Tian Wang, Member, IEEE, Tao Peng, Member, IEEE, and Guojun Wang, Member, IEEE

Abstract—Due to the explosive increase of enterprise network traffic, middleboxes that inspect packets through customized rules have been widely outsourced for cost-saving. Despite promising, redirecting enterprise traffic to remote middleboxes raises privacy concerns about the exposure of corporate secrets. To address this, existing solutions mainly apply searchable encryption (SE) to encrypt traffic and rules, enabling middlebox to perform pattern matching over ciphertexts without learning any sensitive information. However, SE is designed for searching pre-chosen keywords, and may cause extensive costs when applied directly to inspecting traffic in which the keywords cannot be determined in advance. The inefficiency of existing SE-based approaches motivates us to investigate a privacy-preserving and lightweight middlebox. To this end, this paper designs SlimBox, which rapidly screens out potentially malicious packets in constant time while incurring only moderate communication overhead. Our main idea is to fragment a traffic/rule string into sub-patterns to achieve conjunctive sub-pattern matching over ciphertexts, while incorporating the position information into the secure matching process to avoid false positives. Experiment results on real datasets show that SlimBox can achieve a good tradeoff between matching latency and communication cost compared to prior work.

Index Terms—Outsourced middlebox, privacy preserving, lightweight, pattern matching, searchable encryption.

1 INTRODUCTION

Middleboxes have been widely deployed as a vital component of modern networks, performing deep packet inspection (DPI) to monitor abnormal network traffic. For example, intrusion detection systems (e.g., Snort [1] or Bro [2]) are extensively used to detect if packets contain known attack patterns. Due to the increasing volume of network traffic, maintaining in-house middlebox infrastructure may incur expensive overheads. For cost effectiveness, it is a prevailing trend for enterprises to outsource middleboxes [3].

Despite promising, cloud-based middlebox services also face new security challenges [4], [5]. To illustrate, let us consider the following application scenario. Enterprise A is cooperating with enterprise B on a major project. To protect confidential corporate data from eavesdroppers, all enterprises enter,

Meanwhile, enterprise A subscribes outsourced middlebox services for exfiltration prevention. The middlebox needs to thoroughly inspect the packets out of the enterprise network to block accidental leakage of private data. In this scenario, the traffic redirected to the middlebox is encrypted, limiting the processing capabilities of exfiltration prevention. The simple approach that intercepts and decrypts the encrypted traffic would easily trigger potential man-in-the-middle attacks [6]. Furthermore, the packet inspection rules are the intellectual property of the security organizations (e.g., McAfee [7]) and should be protected against the external middlebox.

To address the above privacy concerns, existing work mainly applied searchable encryption (SE) [8]–[10] to encrypt traffic and rules, such that middleboxes can perform pattern matching over ciphertexts without decryption [11]–[19]. The main problem is that SE is designed for searching pre-chosen keywords, and may incur large communication and computational costs when supporting pattern matching over a traffic string that cannot be expressed as a sequence of pre-defined keywords. For example, BlindBox [11] fragmented the traffic string into different lengths for encryption, resulting in nearly $400 \times$ the cost than the original packet size. S4E [19] required heavy pairing operations to support shiftable encrypted patterns, consuming almost 1300s to inspect a packet over 3099 patterns. Recently, Lai et al. [20] exploited symmetric hidden vector encryption to design a bandwidth-efficient pattern matching scheme with the cost of disclosing the rule pattern length. Moreover, the deterministic cryptographic primitive used in the traffic encryption may make their scheme suffer frequency analysis attacks [21].

The insufficiencies or privacy problems of existing ap-
proaches motivate us to develop lightweight middleboxes for secure outsourcing. In this paper, we design SlimBox to efficiently offer privacy-preserving DPI services. Our main idea is to fragment a traffic/rule string into sub-patterns of fixed length and adapt the OXT scheme [22] to efficiently achieve conjunctive sub-pattern searching. The main trick is that the position information is subtly integrated into the secure matching process while ensuring traffic/rule privacy. Specifically, we first provide a basic construction, \textit{SlimBox}^0, which achieves constant-time filtering by using the cross searching technique of OXT, but incurs fair-sized bandwidth and extra leakage about the partial matching results. Based on carefully tailored techniques, our advanced design, \textit{SlimBox}^+ , achieves better privacy protection and reduced bandwidth, at the cost of almost the same computational complexity. Our contributions can be summarised as follows:

- We design \textit{SlimBox}, a privacy-preserving and lightweight middlebox, which supports pattern matching over encrypted traffic while incurring moderate overheads under well-defined leakage.
- We construct \textit{SlimBox} under different trade-off between security and efficiency. Compared with existing SE-based solutions, \textit{SlimBox} can rapidly screen out malicious packets.
- We design a randomization method in the pretreatment phase to further hide offsets between sub-patterns while avoiding leaking the rule pattern length.
- We formally analyze the security of \textit{SlimBox} and evaluate the performance on real-world datasets. Experimental results demonstrate that \textit{SlimBox} is extremely efficient for secure pattern matching.

**Paper Organization.** We introduce the related work in Section 2 and formulate the problem in Section 3. After overviewing this work in Section 4, we construct \textit{SlimBox}^0 and \textit{SlimBox}^+ in Section 5 and Section 6, respectively. We analyze the security in Section 7 before evaluating performance in Section 8. Finally, we conclude the paper in Section 9.

## 2 Related Work

With the increasing demand for outsourced DPI services, secure middleboxes have attracted widespread attention. Existing research in this field can mainly be classified into SE-based solutions [11]–[19] and hardware-based solutions [23]–[27]. The hardware-based solutions lean upon hardwared enclave (i.e., Intel SGX), guaranteeing that the middlebox functions are executed in a trusted environment by feeding the traffic into the enclave. The main problem with these kinds of solutions is that the cloud servers are required to be equipped with SGX, which is vulnerable to various side-channel attacks [28], [29].

The SE-based solutions rely on searchable encryption, which allows cloud servers to search specific keywords over encrypted data. Blindbox [11] is the first secure middlebox that tokenizes the payloads into fragments, each representing a keyword and is encrypted using SE. It allows the middlebox to perform pattern matching in a privacy-preserving way, but will incur a huge bandwidth for high matching accuracy. Since then, a few works have been conducted to improve the design of Blindbox in terms of performance and security. For instance, Embrak [12] enhances the token matching technique to realize prefix matching; BlindIDS [13] improves the performance of Blindbox in terms of connection setup time; Yuan et al. [14] build some encrypted rule indexes based on cuckoo hashing to broad support of inspection rules; SPABox [15] tokenizes the keywords and builds a Trie-like structure to accelerate the matching process; PrivDPI [16] reduces the setup delay by reusing intermediate results generated in previous sessions; Pine [17] further simplifies the preprocessing step of PrivDPI while protecting the rule privacy. These solutions are based on the tokenization technique proposed in [11], and thus have shortcomings of high communication overhead.

Recently, another line of work employed bilinear pairings to support pattern matching against the encrypted data. S4E [18] allows for pattern matching with keywords of arbitrary length by using shiftable encrypted patterns. S4E/AS3E [19] utilizes the fragmentation approach to improve the matching performance of S4E. Although the pairing-based solutions consume less bandwidth than the tokenisation-based solution, they require a lot of expensive pairing operations during matching process, and thus consume too much matching time to be deployed in practice. To further reduce the communication cost, Lai et al. [20] employed symmetric hidden vector encryption to realize encrypted traffic pattern matching. However, the inherent deterministic cryptographic primitive used in the traffic encryption make the scheme easily suffer frequency analysis attacks [21]. Our \textit{SlimBox} is built based on OXT, a sublinear SE scheme supporting conjunctive keyword search. Unlike the prior solutions that linearly scan the ruleset for each packet, our \textit{SlimBox} allows the middlebox to screen out malicious packets in constant time. In best cases, the middlebox is not required to perform any expensive operation at all. The comparison between our work and previous work is shown in Table 1.

### Table 1: SE-based middleboxes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Matching speed</th>
<th>Communication overhead</th>
<th>Fast filtering</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrivDPI [16]</td>
<td>Fast</td>
<td>High</td>
<td>×</td>
</tr>
<tr>
<td>S4E [18]</td>
<td>Slow</td>
<td>High</td>
<td>×</td>
</tr>
<tr>
<td>S4E/AS3E [19]</td>
<td>Slow</td>
<td>High</td>
<td>×</td>
</tr>
<tr>
<td>SlimBox^+</td>
<td>Fast</td>
<td>High</td>
<td>✓</td>
</tr>
<tr>
<td>SlimBox^+</td>
<td>Fast</td>
<td>Moderate</td>
<td>✓</td>
</tr>
<tr>
<td>SlimBox (two-round)</td>
<td>Fast</td>
<td>Small</td>
<td>✓</td>
</tr>
</tbody>
</table>

## 3 Problem Formulation

### 3.1 System Architecture

As shown in Fig. 1, the system consists of four entities: the rule generator (RG) managed by the organization (e.g., McAfee), the gateway inside the enterprise network (IGW), the gateway in the external network (EGW), and the middlebox (MB) deployed in the cloud. The same architecture can be found in [4], [12], [20], [30].

The RG possesses a set of rules that formulate potential attacks and can be used to detect malicious traffic. For cost efficiency, the RG outsources the ruleset and delegates the cloud-based MB to perform traffic inspection. For anomaly
3.2 Threat Model

Our threat model assumes that the RG and IGW are trustworthy. This assumption is consistent with the threat model in existing work [11], [12], where at least one gateway is honest. The MB is assumed to be a honest-but-curious attacker. That is to say, the MB will faithfully execute the protocol given by the system, but may try to exploit sensitive information about the ruleset/traffic. Our design aims to preserve the following security properties:

- **Traffic/Rule confidentiality:** The MB cannot learn the sensitive contents from the encrypted traffic and rules.
- **Traffic indistinguishability:** Given the encrypted traffic matching all encrypted rules, the MB cannot decide the frequency of occurrence for any pattern appearing in it.

3.3 Notations and Cryptographic Preliminaries

**Notation.** For integer $n$, notation $[n]$ represents a set of integers $\{1, \ldots, n\}$. We use notation $\{0, 1\}^n$ (resp. $\{0, 1\}^*$) to denote the set of binary strings of length $n$ (resp. arbitrary length). For a bit string $B$, its length is denoted by $|B|$ and the $i$-th element by $B[i]$ for $i \in [|B|]$. A bit string of $x$-bit ones is denoted by $\langle 1 \cdots 1 \rangle_x$, and a bit string that starts/ends with 0 and contains successive $x$ ones in the middle is denoted by $0/\langle 0 \cdots 0 \rangle_0$. A character string $S$ is defined on an universe character set $\Sigma$ (e.g., ASCII character set), with $|S|$ denoting the number of characters contained in $S$. The concatenation of two strings $S_1$ and $S_2$ is denoted by $S_1 || S_2$. For a finite set $X$, its cardinality is denoted by $|X|$, and $(x_1, \ldots, x_n) \overset{\$}{\leftarrow} X$ denotes uniformly sampling $x_i$ from $X$, for $i \in [n]$. Notation $\lambda \in \mathbb{N}$ denotes the security parameter throughout this paper.

**DDH Assumption.** Let $G$ be a prime order cyclic group of order $p$ generated by $g$. The DDH assumption holds in $G$, if $\text{Adv}^{\text{ddh}}_A(\lambda) = \Pr[A(g, g^a, g^b, g^{ab}) = 1] - \Pr[A(g, g^a, g^b, g^c) = 1]$ is negligible, where vector $a \in (\mathbb{Z}_p)^n$, vector $b \in (\mathbb{Z}_p)^n$, $g^{ab} \in \mathbb{G}^n$, and $M$ is uniform over $\mathbb{G}^n$. Let $\lambda$ denote the advantage for an adversary $A$ to distinguish the ciphertexts of two equal-length plaintexts. SKE is IND-CPA secure if for any PPT adversary $A$, $\text{Adv}^{\text{ske}}_A(\lambda)$ is negligible.

**Symmetric Key Encryption (SKE).** It consists of two polynomial-time algorithms $\text{SKE} = (\text{Enc}, \text{Dec})$. The encryption algorithm Enc takes a secret key $k_e \in \{0, 1\}^\lambda$ and a plaintext $m \in \{0, 1\}^*$ as its inputs and returns a ciphertext $c$. The decryption algorithm Dec takes the secret key $k_e$ and a ciphertext $c$ as its inputs, and returns $m$. Let $\text{Adv}^{\text{ske}}_A(\lambda)$ denote the advantage for an adversary $A$ to distinguish the ciphertexts of two equal-length plaintexts. SKE is IND-CPA secure if for any PPT adversary $A$, $\text{Adv}^{\text{ske}}_A(\lambda)$ is negligible.

**Pseudo-Random Function (PRF).** Let $\text{Adv}^{\text{prf}}_A(\lambda)$ denote the advantage for an adversary $A$ to distinguish a PRF function from a true random function. A PRF is secure if $\text{Adv}^{\text{prf}}_A(\lambda)$ is negligible for any PPT adversary $A$.

**The OXT scheme.** The basic idea is to integrate the following cross searching process into SE. Given a conjunction of query keywords $(w_1, w_2, \ldots, w_k)$, the server first performs inverted searching to find out the files containing keyword $w_1$, denoted by $\text{DB}(w_1)$, and then for each file $f \in \text{DB}(w_1)$, the server performs forward checking to test whether $f$ contains all the remaining keywords $w_2, \ldots, w_k$. As for security, the outsourced database consists of an oblivious index built for keywords, and a set of oblivious cross tags built for keyword/file pairs. The main trick is that by using customized blinding factors, the server can perform secure inverted searching on the oblivious index and calculate oblivious cross tags to support secure forward searching, without learning either keywords or files.

The reason of choosing OXT as the basic primitive is that the cross searching process allows the MB to quickly filter out all the benign packets. According to the observation from [31], only a fringe of the traffic is malicious (less than 0.01%). After inverted searching, the MB only needs to perform forward checking on a small fraction of potentially malicious traffic, thus affording improved performance.

4 Technical Overview

4.1 Pretreatment Phase

The original payloads and rule contents are in the form of character strings. To support secure pattern matching, a naive solution is to express each payload/rule content as a set of $k$-grams (i.e., a string of characters with length $k$), and directly apply OXT to support conjunctive $k$-gram search. However, this simple solution without considering $k$-gram positions will cause a high false positive rate. For example, when $k = 2$, the payload string “ABBC” (expressed as \{“AB”, “BB”, “BC”\}) will be mistakenly considered to match rule string “ABC” (expressed as \{“AB”, “BC”\}).

To eliminate false positives, our main idea is incorporating the information about $k$-grams and their positions into the cross searching process. Given a large prime $p$, let $F_p : \{0, 1\}^\lambda \times \{0, 1\}^* \rightarrow \mathbb{Z}_p$ be a PRF with secret key $K$. Alg. 1 is run to preprocess payloads/rules. Specifically, the pretreatment phase mainly consists of the following steps:

Transformation. The sliding window algorithm is adopted to transform a payload string $P$ into a set of
Algorithm 1 Pretreatment Phase

Preprocessing Traffic (PT)

**Input:** Payload string $P$, length $k$, secret key $K$

**Output:** Payload pairs $\{(kg_s, x_p)\}_{s=1}^{\lfloor |P| - k + 1 \rfloor}$

1. Transform $P$ into $\{(kg_s, pos_s)\}_{s=1}^{\lfloor |P| - k + 1 \rfloor}$
2. $x_0 \leftarrow Z_n^*$, $n \leftarrow |P| - k + 1$
3. for $s \in [n]$ do
   4. $x_p \leftarrow (x_{p-1} + F_p(K, kg_s)) \mod p$

Preprocessing Rules (PR)

**Input:** Rule string $R_i$, length $k$, secret key $K$

**Output:** Rule patterns $R_i = \{(okg_j, lkg_j)\}_{j=1}^{h}$

1. Transform $R_i$ into $\{(kg_s, ofs_j)\}_{j=1}^{h}$
2. $R_i \leftarrow \{(kg_s, \{(okg_j, ofs_j)\}_{j=1}^{h})\}$
3. for $j \in [h]$ do
   4. for $x = 1$ to $|ofs_j|$ do
      5. if $ofs_j > 0$ then
         6. Obtain the $k$-gram $okg$ with offset $x$ from $lkg_i$
         7. $\Delta_j \leftarrow (\Delta_j + F_p(K, okg)) \mod p$
      else
         9. Obtain the $k$-gram $okg$ with offset $-x+1$ from $lkg_i$
         10. $\Delta_j \leftarrow (\Delta_j - F_p(K, okg)) \mod p$
   11. Replace $ofs_j \in R_i$, with $\Delta_j$

payload pairs $\{(kg_s, pos_s)\}_{s=1}^{\lfloor |P| - k + 1 \rfloor}$ (line 1 of algorithm PT), where the $s$-th pair $(kg_s, pos_s)$ denotes the $k$-gram $kg_s$ located at position $pos_s$ of $P$. A rule string $R_i$ is transformed into a set of offset $k$-grams with relative offsets regarding a baseline $k$-gram (line 1 of algorithm PR), $\{(kg_s, \{(okg_j, ofs_j)\}_{j=1}^{h})\}$, where $kg_s$ is the baseline $k$-gram, and $(okg_j, ofs_j)$ denotes the offset $k$-gram $okg_j$ and its relative offset $ofs_j$ from $lkg_i$. In the choice of baseline $k$-grams, we require that each of them is distinct and can be used as a label denoting the rule$^1$. To make the transformation step easily be understood, we give an example in Fig. 2-(a).

Randomization. For the payload pair $(kg_s, pos_s)$, the position value $pos_s$ is replaced by $x_p = \sum_{s=1}^{\lfloor |P| - k + 1 \rfloor} F_p(K, kg_s) + x_0$, where $x_0$ is a random value from $Z_n^*$ (line 2-4 of algorithm PT). In practice, we can use the current time $t$ along with traffic id $id$ as the input of $F_p$ to generate $x_0$ (i.e., $x_0 = F_p(K, t||id)$). Let $okg'$ denote the offset $k$-gram with offset $x$ from baseline $k$-gram $lkg_i$. As shown in the line 3-11 of algorithm PR, for the pair $(okg_j, ofs_j) \in R_i$, the offset value $ofs_j$ is replaced by $\Delta_j = \sum_{x=0}^{\lfloor ofs_j \rfloor} F_p(K, okg') \mod p$ if $ofs_j > 0$, and by $\Delta_j = -\sum_{x=0}^{\lfloor ofs_j \rfloor} F_p(K, okg') \mod p$ if $ofs_j < 0$ (where $okg' = kg_s$). The corresponding example of the randomization step is shown in Fig. 2-(b). The randomization step aims to provide better privacy protection for position/offset values. The original domain of position/offset values is related small and can be easily doped out. After randomization, the real position/offset value is replaced by the sum of appropriate random values related to $k$-grams. It is worth noticing that this step will not change the matching results, since $(x_p = x_p + \Delta_j) \Leftrightarrow (pos_s = pos_s + ofs_j)$.

1. In our experiments, when $k \geq 4$, more than 99% of rules in datasets Snort and ETOpen can be uniquely denoted by a baseline $k$-gram.

Rule string: security

| uri, (sec, 3), (ity, 2) |

Traffic string: fog security

| (fig, 1), (og, 2), (g s, 3) |
| (sec, 4), (ecu, 5), (cur, 6) |
| (uri, 7), (rit, 8), (ity, 9) |

(a) Transformation step.

$F_p(K, fog)=55$ $F_p(K, og)=20$ $F_p(K, g s)=18$ $F_p(K, sec)=90$ $F_p(K, ecu)=04$

$F_p(K, cur)=81$ $F_p(K, rit)=36$ $F_p(K, ity)=42$ $F_p(K, sec)=60$ $xp_0=100$

Rule string: security

| uri, (sec, 211), (ity, 102) |

Traffic string: fog security

| (fig, 155), (og, 175), (g s, 193) |
| (sec, 283), (ecu, 377), (cur, 458) |

| uri, 494, (rit, 536), (ity, 596) |

(b) Randomization step.

Fig. 2: A sample preprocessed rule/traffic ($k = 3$).

With the pretreatment step, the cross searching process can be performed as follows: For each payload pair $(kg_s, x_p)$, the MB first checks whether there exists a rule $(lkg_i, \{(okg_j, \Delta_j)\}_{j=1}^{h})$, such that $lkg_i = kg_s$ (inverted searching). If so, this means that the baseline $k$-gram $lkg_i$ appears at position $xp_s$ of the payload, and the MB further tests whether the $j$-th offset $k$-gram $okg_j$ appears at position $xp_s + \Delta_j$, for $j \in [h]$ (forward checking). If all the $h$ pairs can be found, the rule is regarded fully matching the payload. As the example shown in Fig. 2, during matching, the MB first locates the 7-th pair (“uri”, 494) of the traffic, and then calculates the relative positions $283$ ($=494-211$) and $596$ ($=494+102$) for offset $k$-grams “sec” and “ity”, respectively. Since pairs (“sec”, 283) and (“ity”, 596) coexist in the traffic, the MB determines that the rule fully matches the traffic.

4.2 Syntax and Definitions

After pretreatment, the traffic is in the form of $T = (id, \{F_1, \ldots, F_n\})$, where $id \in \{0, 1\}^\lambda$ is the traffic identifier, $n = |P| - k + 1$ and the $s$-th payload pair $P_s = (kg_s, x_p)$ for $s \in [n]$. The ruleset consists of a set of pattern/action pairs $R S = \{(R_1, A_1), \ldots, (R_m, A_m)\}$, where $R_i = \{(lkg_i, \{(okg_j, \Delta_j)\}_{j=1}^{h})\}$ and $A_i$ is the corresponding action (e.g., alert and activate) for $i \in [m]$. The matching result between the traffic and the rule is defined as follow:

Definition 1 (Matching result). The rule $R_i$ fully matches the $s$-th pair of the traffic $T$ if $lkg_i = kg_s$ and $(okg_j, x_p + \Delta_j)$ exists in $T$ for $j \in [h]$. The rule $R_i$ partially matches the $s$-th pair of the traffic $T$ if $lkg_i = kg_s$ and there exists $j \in [h]$ s.t. $(okg_j, x_p + \Delta_j) \notin T$. The rule $R_i$ does not match the traffic $T$ at all if $lkg_i \neq kg_s$ for all $s \in [n]$.

SlimBox consists of the following algorithms: (1) (ERS) $\Leftrightarrow$ RuleEnc(RS, SK): The RG takes the secret keys $SK$ and the ruleset RS as input and outputs the encrypted

2. Note that the number of $k$-gram/offset pairs varies by rules. For ease of illustration, we use notation $h$ for the unified representation.
The above construction allows the MB to output correct matching results, but will cause two security problems. First, the label tag will expose the frequency of each k-gram in the traffic, due to the deterministic property of PRFs. To solve this problem, the IGW associates each k-gram with an incremental counter and generates the label tag for the k-gram/counter pair. Therefore, the label tags are distinct, achieving traffic indistinguishability. Second, the MB is able to calculate extra information from the matching process. To illustrate, we assume that two rules $R_i = (lkg_i, okg_i, \Delta_i)$ and $R_j = (lkg_j, okg_j, \Delta_j)$ match the s-th tag and the t-th pair of the traffic $T$, respectively. In this case, the MB can obtain four intermediate values $V_1 = g^{F_p}((okg_i) \times F_p(id) \times xid_s)$, $V_2 = g^{F_p}((okg_j) \times F_p(id) \times xid_s)$, $V_3 = g^{F_p}((okg_i) \times F_p(id) \times xid_s)$, and $V_4 = g^{F_p}((okg_j) \times F_p(id) \times xid_s)$. If okg$_i$ = okg$_j$, the MB can obtain the cross tags for pairs $(okg_i, xid_j + \Delta_j)$ and $(okg_j, xid_i + \Delta_i)$, by calculating $V_1 \times V_2$ and $V_3 \times V_4$, respectively. To avoid the extra leakage, the position/offset value is associated with a new blending factor, which can be cancelled only when the forward checking is correctly performed.

5.2 Basic Construction

Let $F : \{0, 1\}^* \rightarrow \{0, 1\}^*$ and $F_p : \{0, 1\}^* \rightarrow \{0, 1\}^*$ be PRFs, and let $\text{SKE} = (\text{Enc}, \text{Dec})$ be a SKE scheme. We assume that the IGW maintains a local map LM that records the number of appearances of each k-gram in all traffic flows. The maximum appearances of k-grams in all traffic flows is denoted by MAX. Given $SK = (K_0, K_1, K_2, K_3, K_4)$ randomly chosen from $\{0, 1\}^\ell$, the basic SlimBox construction is provided in Alg. 2, where the mod $p$ operation is omitted for brevity.

Correctness. In algorithm RuleEnc$^0$, each rule/action pair $(R_i, A_i)$ is encrypted for MAX times. In the c-th encryption, a label trapdoor $ltrap_i, h$ cross trapdoor pairs $\{\{xtrap_j, xofs_j\}\}_{j=1}^h$, and an encrypted action $cp_h$ are generated, where $ltrap_i$ and the blinding factors hiding cross trapdoors are calculated based on $lkg_i$. In algorithm TrafEnc$^0$, the s-th pair $(kg_s, xps_s)$ is encrypted into a label tag $ltag_{s}$, a cross tag $xtag_s$, and two auxiliary tags $(\alpha_s, \beta_s)$. For traffic indistinguishability, $ltag_s$ and the blinding factors hiding auxiliary tags are calculated based on the k-gram/counter pair. Note that $xtag_s$ is calculated from $(id, kg_s, xps_s)$, and will not repeat itself in the traffic. Therefore, there is no need to associate cross tags with counter information.

In algorithm Match$^0$, the MB first checks whether $\text{ERS}[ltag_i] = \perp$ or not, for $(ltag_i, \alpha_s, \beta_s)$ in the encrypted traffic. If so, this means that there is no rule that matches the s-th pair $(kg_s, xps_s)$ of the traffic and the MB continues to check the next pair. Otherwise, this means that there exists a rule $R_i = (lkg_i, \{(okg_j, \Delta_j)\}_{j=1}^h)$ such that $lkg_i = kg_s$. Then, the MB further calculates the cross tags $\{xtrap_j\}_{j=1}^h \times xofs_j$ for k-gram/position pairs $\{(okg_j, xps_s + \Delta_j)\}_{j=1}^h$. Note that when $\text{ERS}[ltag_i] \neq \perp$, the blinding factors $z, v, l$ used in auxiliary tags and cross trapdoors are the same. Therefore, we have:

\[
\begin{align*}
\text{xtrap}^\beta_j \times \text{xofs}_j^\alpha & = g^{\text{xofs}_j \times \text{xid}_s \times (xps_s + l)} \times v^{-1} \times g^{\text{xofs}_j \times (\Delta_j - l) \times z^{-1} \times \text{xid}_s} \\
& = g^{\text{xofs}_j \times \text{xid}_s \times (xps_s + l)} \times g^{\text{xofs}_j \times \text{xid}_s \times (\Delta_j - l)} \\
& = g^{\text{xofs}_j \times \text{xid}_s \times (xps_s + \Delta_j)},
\end{align*}
\]

### Table 2: Summary of Notations

<table>
<thead>
<tr>
<th>$k$</th>
<th>The length of k-gram</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_k$</td>
<td>The baseline k-gram of $N_i$</td>
</tr>
<tr>
<td>$(lkg_i, xofs_j)$</td>
<td>The j-th offset k-gram $okg_j$ of a rule and its relative offset $ltrap_j$ from the baseline k-gram</td>
</tr>
<tr>
<td>$(okg_j, \Delta_j)$</td>
<td>The j-th offset k-gram $okg_j$ of a rule and its random offset $\Delta_j$</td>
</tr>
<tr>
<td>$ltrap_j$</td>
<td>The lab trapdoor of $N_k$</td>
</tr>
<tr>
<td>${xtrap_j, xofs_j}$</td>
<td>The j-th cross trapdoor pair of a rule</td>
</tr>
<tr>
<td>$P$</td>
<td>A payload string</td>
</tr>
<tr>
<td>$(kg_s, pos_\alpha)$</td>
<td>The k-gram $kg_s$ and its position $pos_\alpha$ located at $P$</td>
</tr>
<tr>
<td>$(kg_s, xp_\beta)$</td>
<td>The k-gram $kg_s$ and its random position $xp_\beta$</td>
</tr>
<tr>
<td>$(ltag_s, xtag_s)$</td>
<td>The label tag and cross tag of k-gram $kg_s$</td>
</tr>
<tr>
<td>$(\alpha_s, \beta_s)$</td>
<td>The auxiliary tags of k-gram $kg_s$</td>
</tr>
</tbody>
</table>

### Rule Set ERS.

$(2) \text{ET } \leftarrow \text{TrafEnc}(T, LM, SK)$: The IGW takes the traffic $T$, the map LM, and the keys $SK$ as input, and outputs the encrypted traffic $ET$. $(3) \text{Match}(ERS, ET)$: The MB takes the encrypted rule set $ERS$ and the encrypted traffic $ET$ as input and outputs the matching result $MR$. $(4) \text{AS } \leftarrow \text{Action}(MR, SK)$: The IGW takes the result $MR$ and the keys $SK$ as input and recovers the actions $AS$.

Specifically, given a set of rule patterns $\{(lkg_i, \{(okg_j, \Delta_j)\}_{j=1}^h)\}$ of the rule string $R_i$, the RuleEnc algorithm will encrypt the baseline k-gram $lkg_i$ into label trapdoor $ltrap_i$. The offset k-gram $okg_j$ and its random offset $\Delta_j$ will be encrypted into cross trapdoor pair $(xtrap_j, xofs_j)$. Given a payload pair $(kg_s, xps_s)$ of the payload string $P$, the TrafEnc algorithm will encrypt $kg_s$ into label tag $ltag_s$ along with generating the cross tag $xtag_s$ and the auxiliary tags $(\alpha_s, \beta_s)$. For quick reference, the most relevant notations are shown in Table 2.

5 The Design of SlimBox$^0$

5.1 Rationale

After pretreatment, the matching result between the traffic $T$ and the rule $R_i$ can be easily determined according the Definition 1. However, both $T$ and $R_i$ will be encrypted before going through the MB for security reasons. Therefore, our goal is to allow the MB to securely perform pattern matching based on ciphertexts. By using PRFs, it is easy for the MB to determine whether two k-grams are identical without learning their contents. The challenge is how to enable the MB to securely compute $\Delta_j + xp_j$ for $j \in [h]$.

Our main idea is to incorporate the position information into the oblivious cross tags proposed in OXT. Let $F$ and $F_p$ denote the keyed PRF with range $\{0, 1\}^\ell$ and $\{0, 1\}^*$, respectively. For ease of illustration, we omit the PRF keys and consider a basic extension: For each pattern $R_i = (lkg_i, \{(okg_j, \Delta_j)\}_{j=1}^h)$, the RG generates a label trapdoor $ltrap_i = F(lkg_i)$, and a set of cross trapdoors $\{(xtrap_j, xofs_j)\}_{j=1}^h$, where $xtrap_j = g^{F_p((okg_j) \times v)}$ and $xofs_j = g^{F_p((okg_j) \times \Delta_j \times z^{-1})}$, for each pair $(kg_s, xps_s) \in T$, the IGW generates a label tag $ltag_s = F(kg_s)$, a cross tag $xtag_s = g^{F_p(kg_s) \times F_p(id) \times xid_s}$, and two auxiliary tags $(\alpha_s, \beta_s)$. For $OXT$, $z$ and $v$ are two blinding factors which can be cancelled during matching only when $kg_s = lkg_i$. If there is a match, the MB further calculates $\text{xtrap}^\beta_j \times \text{xofs}_j^\alpha$ to obtain $g^{F_p((okg_j) \times F_p(id) \times (xps_s + l))}$, and tests if the computed result exists in the cross tag set.
Algorithm 2 Basic SlimBox Construction

RuleEnc0

Input: Rule set RS, secret keys SK
Output: Encrypted rule set ERS
1: Parse SK as \( (K_e, K_r, K_S, K_Z, K_V, K_L, K_X) \)
2: ERS \( \leftrightarrow \) empty map
3: for each \( (R_e, A_e) \) \( \in \) RS do
4: Parse \( R_e \) as \( (\text{lk}_S, (\text{okg}_S, \Delta_1), \ldots, (\text{okg}_h, \Delta_h)) \)
5: for \( c = 1 \) to MAX do
6: \( \text{ltrap}_z = F(K_S, \text{lk}_S)[c] \)
7: \( \text{cp}_j = \text{SKE.Enc}(K_e, A_j); z \leftarrow F_p(K_Z, \text{lk}_S)[c] \)
8: \( v \leftarrow F_p(K_V, \text{lk}_s)[c]; l \leftarrow F_p(K_L, \text{lk}_s)[c] \)
9: for \( j = 1 \) to \( h \) do
10: \( \text{xp}_j = g^{F_p(K_X, \text{okg}_j)} \times v \)
11: \( \text{okg}_j \leftarrow g^{F_p(K_X, \text{okg}_j) \times (\Delta_j - l) \times z^{-1}} \)
12: ERS[\text{ltrap}_z] \leftarrow \{ (\{\text{xp}_j, \text{xofs}_j\})_{j=1}^h, \text{cp}_j \}

TrafEnc0

Input: Traffic \( T \), local map LM, secret keys SK
Output: Encrypted traffic ET
1: Parse SK as \( (K_e, K_r, K_S, K_Z, K_V, K_L, K_X) \)
2: Parse \( T \) as \( (\text{id}, \{ (\text{lk}_S, \text{xp}_S) \}_{s=1}^n) \)
3: \( (LT, XT, ET) \leftarrow \emptyset; \text{id}_s \leftarrow F_p(K_I, \text{id}) \)
4: for the \( s \)-th pair \( (\text{lk}_S, \text{xp}_S) \) \( \in \) \( T \) do
5: if \( LM[\text{lk}_S] = \emptyset \) then
6: \( c \leftarrow 1 \\
7: \text{else} \)
8: \( c \leftarrow LM[\text{lk}_S] + 1 \)
9: \( LM[\text{lk}_S] \leftarrow c + 1 \)
10: \( \text{xofs}_j \leftarrow g^{F_p(K_X, \text{okg}_j) \times \text{xid} \times \text{xid} \times (\text{xid} \times \text{xid} + l) \times v^{-1}} \)
11: \( \text{ltag}_s \leftarrow F(K_S, \text{lk}_S)[c] \)
12: \( \text{LT} \leftarrow \text{LT} \cup \{ (\text{ltag}_s, \alpha_s, \beta_s) \}; \text{XT} \leftarrow \text{XT} \cup \text{xtag}_s \)
13: ET \( \leftarrow \) (LT, XT)

Match0

Input: Encrypted rule set ERS, encrypted traffic ET
Output: Matching results \( MR \)
1: \( MR \leftarrow \emptyset; \) parse ET as \( (LT, XT) \)
2: for each \( (\text{ltag}, \alpha, \beta) \) \( \in \) LT do
3: if \( ERS[\text{ltag}] \neq \emptyset \) then
4: obtain \( \{ (\{\text{xp}_j, \text{xofs}_j\})_{j=1}^h, \text{cp}_j \} \) from ERS[\text{ltag}]
5: for \( j = 1 \) to \( h \) do
6: \( \text{xp}_j \leftarrow \text{xp}_j \times \text{xofs}_j \)
7: if \( x_1, \ldots, x_h \) \( \in \) XT then
8: \( MR \leftarrow MR \cup \{ (\text{ltag}, \text{cp}) \} \)

Action0

1: The IGW runs SKE.Dec to recover actions in \( MR \)

where \( \text{xofs}_j = F_p(K_X, \text{okg}_j) \).

From Eq. (1), the MB can check whether \( (\text{okg}_j, \text{xp}_s + \Delta_j) \) exists in the traffic for \( j \in [h] \) and output correct matching results.

Complexity. In terms of computation costs, we only consider the operations related to group \( \mathbb{G} \). For a rule \( R_i = (\text{lk}_S, \{ (\text{okg}_j, \Delta_j) \}_{j=1}^h) \), algorithm RuleEnc0 requires \( 2h \times \text{MAX} \) exponentiations in \( \mathbb{G} \) for generating cross trapdoors. For a traffic \( T = (\text{id}, \{ P_1, \ldots, P_n \}) \), algorithm TrafEnc0 requires \( n \) exponentiations in \( \mathbb{G} \) for generating cross tags. If \( R_i \) fully/partially matches the \( s \)-pair of traffic \( T \), algorithm Match0 requires \( 2h \) exponentiations and \( h \) multiplication in \( \mathbb{G} \). Otherwise, no group-related operation is required. As for communication costs, the encrypted rule set contains \( m \times \text{MAX} \) entries, where each entry contains two \( \lambda \)-bit strings and \( 2h \) elements in \( \mathbb{G} \). The encrypted traffic consists of two sets \( LT \) and \( XT \), where \( LT \) contains \( n \) \( \lambda \)-bit strings and \( 2n \) elements in \( \mathbb{G}^\lambda \), and \( XT \) contains \( n \) elements in \( \mathbb{G} \). The matching results \( MR \) contains two \( \lambda \)-bit strings.

6 The Design of SlimBox*

6.1 General Ideas

SlimBox0 allows the MB to securely inspect traffic, but still has the following insufficiency: (1) The size of the encrypted rule set is related to \( \text{MAX} \), the maximum appearances of \( k \)-grams in history, and thus is extensively large. (2) During forward checking, the partial matching result, i.e., the \( j \)-th pattern matches/matches the traffic for \( j \in [h] \), is leaked to the MB. (3) The encrypted traffic consumes considerable costs, due to the large size of group elements.

For the first problem, our solution is resetting the counters for each packet to reduce the value of \( \text{MAX} \). Therefore, there is no need for the MB to locally maintain the counter information. However, this will enable the \( k \)-gram appearing in different packets to have the same label tag. For traffic indistinguishability, our main idea is to introduce a random salt \( s \) in the calculation of label tags and label trapdoors. Intuitively, the IGW applies PRFs to uniquely encode each \( (\text{lk}_e, c) \) pair and use \( s \) to keep track of the occurrence of the pair appearing in the traffic. To support pattern matching, the MB keeps the initial random salt for each \( (\text{lk}_e, c) \) pair, and updates the random salt to generate a new label trapdoor if a match happens.

For the second problem, a simple solution is to let the IGW withhold the cross tag set \( XT \) and let the MB return back the action ciphertext as well as the calculated cross tags, enabling the IGW to perform forward checking on behalf of the MB. The main problem is that the MB needs to return \( h \) group elements for each matched rule, resulting in large communication costs. To solve this problem, our solution is letting the MB return the XOR result of the action ciphertext and the hash values of \( h \) cross tags, together with an encrypted bit string, both of which are of constant lengths. As shown in Alg. 4, the offset information is encoded into the bit string, from which the IGW can correctly locate cross tags in \( XT \) to recover the action ciphertext.

The solution to the second problem actually can help cut the traffic size in half. To further reduce the cost, a two-round interaction can be performed as follows: In the first round, the IGW only sends the label tags to the MB for inverted searching, and receives the matched label tags from the MB. In the second round, the IGW sends the
corresponding auxiliary tags to the MB, and receives the matching results from the MB. Since only a fraction of auxiliary tags needs to be transmitted, the traffic size is greatly decreased.

6.2 Advanced Construction

Let $F : \{0, 1\}^\lambda \times \{0, 1\}^\lambda \to \{0, 1\}^\lambda$ and $F_p : \{0, 1\}^\lambda \times \{0, 1\}^\lambda \to \Z^\lambda$ be PRFs, and let $\text{SKE} = \{\text{Enc}, \text{Dec}\}$ be a SKE scheme. Let $H : \{0, 1\}^\lambda \to \{0, 1\}^\lambda$ be a collision-free hash function, and let $\text{MAX}^*$ denote the maximum appearances of $k$-grams in a payload. We assume that the IGW maintains a local map LM to record the number of appearances of each $k$-gram in a payload. Given $SK = (K_c, K_I, K_S, K_Z, K_V, K_L, K_X)$ defined as the basic construction, the detailed construction of SlimBox$^*$ is shown in Alg. 3, where the $mod \ p$ operation is omitted, and the pattern matching process is implemented in a single roundtrip. The cost savings brought by the two-round interaction will be demonstrated by experiments.

**Correctness.** For each rule/action pair $(R_i, A_i) \in RS$, algorithm RuleEnc$^*$ first runs Alg. 4 to construct a bit string $B_i$, and then encrypts $(R_i, A_i, B_i)$ for $\text{MAX}^*$ times. In the $i$-th encryption, a label trapdoor $\text{ltrap}_i$, $h$ cross trapdoor pairs $\{\text{xtrap}_j, \text{xsos}_j\}_{j=1}^h$, an auxiliary trapdoor $\text{lt}_i$, a random salt $s_i$, an encrypted action $\text{cp}_i$, and an encrypted bit string $\text{cb}_i$ are generated. The main difference from algorithm RuleEnc$^0$ is that the label trapdoor is computed by $\text{ltrap}_i \leftarrow H(s_i || \text{lt}_i)$, where the random salt $s_i$ is initialized by $F(K_S, \text{lkgs} || c || 0)$ and will be incremented by 1 once a packet matches the rule, and $\text{lt}_i = F(K_S, \text{lkgs} || c || 1)$.

Like algorithm TrafEnc$^0$, the TrafEnc$^*$ algorithm also encrypts the $s$-th pair $(\text{kg}_s, \text{xp}_s)$ into a label tag $\text{ltag}_s$, a cross tag $\text{xtag}_s$, and two auxiliary tags $(\alpha_s, \beta_s)$. The main difference is that the label tag is computed by $\text{ltag}_s \leftarrow H(s || F(K_S, \text{kg}_s || c || 1))$, where $s$ is a random salt initialized by $F(K_S, \text{kg}_s || c || 0)$ and will be incremented by 1 once the $(\text{kg}, c)$ pair appears in the traffic. Furthermore, the cross tags are put into the map XT and will not be sent to the MB.
Specifically, $XT[s]$ keeps tags $(\text{ltag}_s, \text{xtag}_s)$ for the $s$-th pair. Thus, if a rule $R_i = ([\text{lk}_{g_i}],[\text{ok}_{g_i},\Delta_i])_{j=1}^{m}$ fully matches the $s$-th pair, then for each offset $k$-gram ok$_{g_j}$ with offset $o$ from the baseline $k$-gram lk$_{g_j}$, the calculated cross tag $\text{xtrap}_{j}^{\Theta} \times \text{xofs}_{s}^{\alpha}$ should be equal to that stored at $\text{XS}[s + o]$. The Match$_{1}^{h}$ algorithm performs inverted searching as before. If a match succeeds, the MB updates the random salt and calculates a new label trapdoor. According to the construction of algorithms RuleEnc$_{o}$ and TrafEnc$_{o}$, the inverted searching can be performed correctly. However, unlike algorithm Match$_{0}^{h}$ directly returning the action ciphertexts of fully matched rules, Match$_{1}^{h}$ returns the encrypted bit string cb and the processed ciphertext $\text{cp} = \text{cp} \oplus h_{H}(\text{xtrap}_{j}^{\Theta} \times \text{xofs}_{s}^{\alpha})$ for each rule passing inverted searching. In forward checking, the IGW first decrypts cb to obtain offset information and then recovers cp if all the calculated cross tags can be found in the map $\text{XT}$. Since the cross tags are computed in the same way as before, the matching results are correct as long as the IGW can correctly locate the cross tags in $\text{XT}$ based on the bit string.

Now, we will analyze the correctness of the encoding algorithm. Assume that $\lambda \geq h + \max |R| + 3$, and that the baseline $k$-gram is located at position $\theta$ of the rule string. Intuitively, the pattern $(0(1 \cdots 1)_{h+1}0)$ is used to denote the position of the baseline $k$-gram. Note that the number of offset $k$-grams is $h$, which means that there exists at most one pattern consisting of successive $(h + 1)$ ones. Therefore, the IGW can correctly find the pattern $0(1 \cdots 1)_{h+1}0$ and obtain their positions $\theta, \ldots, \theta + h + 2$ in $B$. For $i < \theta$, $B[i]$ denotes the offset from the label $k$-gram is $-\theta - i$, and for $i > \theta + h + 2$, $B[i]$ denotes the offset is $i - \theta - h - 2$. For example, for the rule string “security” and pattern (“uri$^{\Theta}$,”(“sec”$^{\Theta}$),(“-3”),(“ity”$^{\Theta}$))”, the first ten bits of $B$ is in the form of 1000111001. Therefore, the encoding algorithm and the matching process are correct.

**Complexity.** In terms of computation costs, we only consider operations related to group $G$. The advanced algorithms consume the same complexity as the basic algorithms. As for communication costs, the encrypted ruleset contains $m \times \max$ entries, where each entry contains four $\lambda$-bit strings and $2h$ elements in $G$. The encrypted traffic only consists of a set $ET$, which contains $n$ $\lambda$-bit strings and $2n$ elements in $Z_{p}^{\Theta}$, where each label tag can be further compressed as prior work [11], and the number of transmitted auxiliary tags can be reduced through a two-round interaction. The matching results $MR$ contains three $\lambda$-bit strings. The latency of traffic processing is sensitive to bandwidth. Since the size of encrypted packets is significantly declined, SlimBox$^{h}$ offers better user experience than SlimBox$^{o}$.

### 7 Security Analysis

We will analyze the security of SlimBox from the aspects of traffic and rule confidentiality. Since SlimBox$^{h}$ with enhanced privacy is constructed based on SlimBox$^{o}$, we will focus on the security of SlimBox$^{o}$. To simplify presentation, we consider a simple scenario, where each rule $R_i$ consists of two $k$-grams only, and is expressed as $(\text{lk}_{g_i}, \text{ok}_{g_i},\Delta_i)$. Furthermore, the adoption of counters is to realize traffic indistinguishability. We assume each $k$-gram appears in the traffic only once to avoid the usage of counters. Thus, the local map $LM$ is omitted and the rule set size will not be expanded by $M$ times.

#### 7.1 Traffic Confidentiality

We follow the simulation-based security and define a leakage function to capture what can be learned by an adversary. Let $A_1$ be an adversary attempting to learn the sensitive data in the traffic, and let $S_1$ be a simulator parameterized by a leakage function $L_1 = (n, MR, IR)$ defined as below:

- $n$ is the number of payload pairs in the traffic $T$.
- $MR$ is the matching result of rules. (1) $MR[i] = 0$ if $R_i$ does not match any pair in $T$ at all. (2) $MR[i] = (1, s)$ if $R_i$ fully matches the $s$-th pair of $T$. (3) $MR[i] = (2, s)$ if $R_i$ partially matches the $s$-th pair of $T$.
- $IR$ is the intersection result of rules. $IR[i,j] = 1$ if $\exists P_s = (\text{kg}_s, \text{xp}_s) \text{ and } P_t = (\text{kt}_t, \text{xp}_t) \text{ s.t. } \text{lk}_{g_s} = \text{kg}_s \land \text{lk}_{g_j} = \text{kt}_t \land \text{ok}_{g_j} = \text{ok}_{g_j} \land \text{xp}_s + \Delta_i = \text{xp}_t + \Delta_j$. Otherwise, $IR[i,j] = 0$.

SlimBox$^{o}$ achieves traffic confidentiality if $A_1$ cannot distinguish the following experiments:

- **Real$_{A_1}(\lambda)$**: On input the traffic $T$, the experiment runs algorithm TrafEnc and gives $ET$ to $A_1$. For a ruleset RS adaptively chosen by $A_1$, the experiment runs algorithm RuleEnc and gives $ERS$ to $A_1$. Finally, $A_1$ outputs a bit $b \in \{0, 1\}$.

- **Ideal$_{A_1}^{o}(\lambda)$**: On input the traffic $T$, $S_1$ generates $ET$ with the given leakage $L_1$ for $A_1$. For a ruleset RS adaptively chosen by $A_1$, $S_1$ generates $ERS$ with leakage $L_1$. Finally, $A_1$ outputs a bit $b \in \{0, 1\}$.

**Theorem 1.** If $F, F_p$ are secure PRFs, SKE is CPA secure, and the DDH assumption holds in group $G$, then SlimBox$^{o}$ is $L_1$-secure against adversary $A_1$.

**Proof sketch.** Given the leakage $L_1$, the simulator $S_1$ first calculates the partition index for each rule. Let $i$ denote the partition index of rule $R_i$. First, it computes a relation $\equiv$ on rules by defining $i \equiv j$ iff $IR[i,j] \neq 0$, and makes $\equiv$ an equivalence relation by using transitive closure. Then, it assigns each partition of an equivalence relation with a distinct index. Note that for all rules falling in one partition, their partition indexes are identical.

On input a packet $T$ consisting of $n$ pairs provided by $A_1$, $S_1$ computes tags by setting $\text{ltag}_s \leftarrow \lambda \{0, 1\}^\lambda$, $\text{xtag}_s \leftarrow G$, and $(\alpha_s, \beta_s) \leftarrow Z_p$ for $s \in [n]$. To simulate ERS for an adaptively chosen ruleset RS, $S_1$ needs to maintain a map $X$ to record the calculated cross tag for each partition. For rule $R_i \in RS$, $S_1$ generates trapdoors according to the following cases: (1) $MR[i] = 0$, $S_1$ generates trapdoors by setting $\text{lttrap}_i \leftarrow \lambda \{0, 1\}^\lambda$, and $(\text{xtrap}_i, \text{xofs}_i) \leftarrow G$. (2) $MR[i] = (1, s) \lor MR[i] = (2, s)$, $S_1$ first checks whether $X[i] = \bot$ or not. If so, for the case of $MR[i] = (1, s)$, $S_1$ sets $X[i] \leftarrow \text{xtag}_s$, and for the case of $MR[i] = (2, s)$, $S_1$ sets $X[i] \leftarrow G$. Then, $S_1$ chooses two random elements $r_1, r_2 \in G$, s.t. $r_1 \times r_2 = X[i]$, and calculates the trapdoors by setting $\text{lttrap}_i \leftarrow \text{ltag}_s$, $\text{xtrap}_i \leftarrow r_1^\alpha \beta$ and $\text{xofs}_i \leftarrow r_2^\alpha r_1$. In both cases, the action ciphertext is set to $\text{cp}_i \leftarrow \lambda \{0, 1\}^\lambda$.

Due to the security of PRFs and SKE, the simulated trapdoors and tags are indistinguishable from the real game.
TABLE 3: The rule encryption time(s) under different rule sets and parameters.

<table>
<thead>
<tr>
<th>MAX</th>
<th>Snort</th>
<th>ETOpen</th>
<th>Snort</th>
<th>ETOpen</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>510</td>
<td>402</td>
<td>6033</td>
<td>4661</td>
</tr>
<tr>
<td>7</td>
<td>607</td>
<td>474</td>
<td>6973</td>
<td>5146</td>
</tr>
<tr>
<td>8</td>
<td>679</td>
<td>535</td>
<td>7955</td>
<td>6276</td>
</tr>
<tr>
<td>9</td>
<td>774</td>
<td>608</td>
<td>8733</td>
<td>6855</td>
</tr>
<tr>
<td>10</td>
<td>855</td>
<td>672</td>
<td>9923</td>
<td>7606</td>
</tr>
</tbody>
</table>

TABLE 4: The size(MB) of ERS under different rule sets and parameters.

<table>
<thead>
<tr>
<th>MAX</th>
<th>Snort</th>
<th>ETOpen</th>
<th>Snort</th>
<th>ETOpen</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>9.94</td>
<td>7.93</td>
<td>11.42</td>
<td>9.15</td>
</tr>
<tr>
<td>7</td>
<td>11.60</td>
<td>9.25</td>
<td>12.23</td>
<td>9.34</td>
</tr>
<tr>
<td>8</td>
<td>13.26</td>
<td>10.57</td>
<td>12.40</td>
<td>9.16</td>
</tr>
<tr>
<td>9</td>
<td>14.91</td>
<td>11.90</td>
<td>12.41</td>
<td>9.34</td>
</tr>
<tr>
<td>10</td>
<td>16.57</td>
<td>12.32</td>
<td>12.93</td>
<td>11.25</td>
</tr>
</tbody>
</table>

when the DDH assumption holds in group $G$. Thus, we conclude that for every adversary $A_1$, it has a negligible probability to learn more information from the traffic than the defined leakage function $L_1$. The formal proof of Theorem 1 could be found in Appendix A.

7.2 Rule Confidentiality

Let $A_2$ be an adversary attempting to learn the sensitive data from the ruleset $RS$, and let $S_2$ be a simulator parameterized by a leakage function $L_2 = (m, n, MR, IR)$, where $(n, IR)$ are defined similarly as leakage function $L_1$, $m$ is the number of rules in the ruleset, and $MR$ is the matching result of the $s$-th payload pair $P_s$: (1) $MR[s] = 0$ denotes $P_s$ does not match any rule at all. (2) $MR[s] = (i, 1)$ denotes $P_s$ fully matches $R_i$. (3) $MR[s] = (2, i)$ denotes $P_s$ partially matches $R_i$.

SlimBox achieves rule confidentiality if $A_2$ cannot distinguish the following experiments:

- **Real**$_{A_2}(\lambda)$: On input the ruleset $RS$, the experiment runs algorithm RuleEnc and gives ERS to $A_2$. For the packet $T$ adaptively chosen by $A_2$, the experiment runs algorithm TrafEnc and gives ET to $A_2$. Finally, $A_2$ outputs a bit $b \in \{0, 1\}$.

- **Ideal**$_{A_2}(\lambda)$: On input the ruleset $RS$, $S_2$ generates ERS with the given leakage $L_2$ for $A_2$. For the packet $T$ adaptively chosen by $A_2$, $S_2$ generates ET with leakage $L_2$. Finally, $A_2$ outputs a bit $b \in \{0, 1\}$.

**Theorem 2.** If $F, P$ are secure PRFs, SKE is CPA secure, and the DDH assumption holds in group $G$, then SlimBox$^0$ is $L_2$-secure against adversary $A_2$.

**Proof sketch.** As the simulator $S_1, S_2$ first calculates the partition index $i$ for each rule $i$ based on the leakage IR. Let $p$ denote the set of partition indexes. $S_2$ maintains a map $X$ to record the exponent of calculated cross tag for each partition, and uses a map $F$ to record whether the entry of $X$ has been used before. For each $j \in p$, $S_2$ sets $X[j] \leftarrow Z_p^*$ and $F[j] \leftarrow 0$. On input a rule $R_i$ provided by $A_2$, $S_2$ generates the trapdoors by setting $ltrap_i \leftarrow \{0, 1\}^\lambda$ and $xtrap_i \leftarrow g^{x_i}$, and sets the action ciphertext as $cp_i \leftarrow \{0, 1\}^\lambda$, where $x_i, y_i$ are randomly chosen from $Z_p^*$. To simulate ET for the adaptively chosen traffic $T$, $S_2$ generates tags for the $s$-th payload pair $P_s$.

![Fig. 3: The results of SlimBox$^0$ and SlimBox$^*$ for encrypting a payload with 1500 bytes with varied k.](image)

8 Evaluation

We will analyze the performance of SlimBox from the aspects of computational and communication costs. To validate the effectiveness, we conduct experiments on two real datasets, and compare SlimBox to BlindBox [11] and S4E [19], which are based on the SE.

8.1 Experiment Settings and Datasets

We deploy the MB on a server with Intel(R) Xeon(R) Gold 5218 CPU and 128GB RAM, and regard the PC with Intel Core i5 3.2GHz CPU and 32GB RAM as the IGW. We implement the experiments in Java, and set the security parameter $\lambda$ to 256. For cryptographic algorithms, HMAC and SHA-256 are utilized to implement PRFs and collision-free
8.2 Performance Evaluation

- **TrafEnc.** To evaluate the influence of parameter \( k \), we run algorithm TrafEnc\(^0 \) / TrafEnc\(^* \) 100 times and obtain the average results. From Fig. 3-(a), we can see that a larger \( k \) will cause shorter execution time for both algorithms. The reason is that for fixed-length payloads, a larger \( k \) will result in fewer \( k \)-gram/position pairs. In addition, TrafEnc\(^* \) requires extra hash calculations compared with TrafEnc\(^0 \), and thus consumes more execution time. From Fig. 3-(b), we know that TrafEnc\(^* \) saves nearly half of the communication cost compared with TrafEnc\(^0 \). The reason is that TrafEnc\(^* \) does not need to forward cross tags to the MB.

- **RuleEnc.** Table 3 and Table 4 show the performance of algorithm RuleEnc\(^0 \) / RuleEnc\(^* \) under different rulesets and parameter settings. Compared with Snort, ETOpen contains a larger number of rules and consumes more time to generate a larger-scale encrypted ruleset. As \( k \) grows, the number of \( k \)-gram/offset pairs will decrease, rendering the execution time and the size of ERS to reduce for both algorithms. As \( MAX \) increases, the size of rulesets becomes larger, and thus the execution time and the size of ERS will increase for both algorithms. In the same settings, RuleEnc\(^* \) needs to encode and encrypt bit strings, thus requiring more execution time compared with RuleEnc\(^0 \). Similarly, the size of ERS in RuleEnc\(^* \) is a larger size than RuleEnc\(^0 \) because the former needs to store extra information.

- **Match and Action.** We evaluate the performance of Match and Action for SlimBox\(^0 \) and SlimBox\(^* \) under different parameters and rulesets. We first evaluate the influence of parameter \( MAX \) on the execution time of algorithm Match\(^0 \) / Match\(^* \). From Fig. 4-(a) and Fig. 5-(a), we can find that \( MAX \) has little influence on the matching time. The main reason is that SlimBox supports fast filtering, dispensing with linearly scanning the ruleset. Furthermore, Match\(^* \) requires more execution time compared with Match\(^0 \). This is because the former needs extra hash and XOR operations. Fig. 4-(b)(c) and Fig. 5-(b)(c) show the influence of parameter \( k \) on the performance of algorithm Match\(^0 \) / Match\(^* \). From these figures, we know that: (1) The inspection latency as well as the size of matching results in ETOpen is larger than Snort under the same settings. This is because the larger the ruleset size, the more the number of rules passing
inverted searching. (2) The inspection latency and the size of matching results decreases as \( k \) increases. The main reason is that the smaller \( k \) results in more number of rules passing inverted searching. In terms of the action recovery time illustrated in Fig. 4-(d) and Fig. 5-(d), \textit{SlimBox} requires the additional operations (e.g., bit string decryption and hash) and thus consumes more execution time than \textit{SlimBox\textsuperscript{0}}.

\textbf{Comparison with prior work.} We first provide the performance comparisons between \textit{S4E} [19] and our \textit{SlimBox}. From Fig. 6-(a), we know that \textit{SlimBox} consumes less time/bandwidth in the process of traffic encryption compared with \textit{S4E}. The reason is that \textit{S4E} utilizes the fragmentation approach to speed up matching computation, rendering each byte in the traffic to be encrypted multiple times. From Fig. 6-(b) and Fig. 6-(c), we can see that our \textit{SlimBox} consumes more rule encryption time than \textit{S4E}. This is because \textit{SlimBox} needs to encrypt each rule \( MAX \) times. However, \textit{S4E} generates the bigger size of encrypted rulesets since each byte of data is encrypted to a group element. The rule encryption operations are usually performed during initialization process and can be done once for all. Therefore, the increased time overhead is acceptable in real applications. From Fig. 6-(d), we can see that \textit{SlimBox} performs much better than \textit{S4E} in terms of matching latency. The huge difference between \textit{S4E} and \textit{SlimBox} is because \textit{S4E} needs to perform expensive bilinear pairing calculations for each rule, but our \textit{SlimBox} only needs to perform group operations on a fraction of rules passing inverted searching.

Then, we give comparisons between \textit{BlindBox}, \textit{S4E}, and our \textit{SlimBox} in terms of traffic encryption time, communication cost, and matching latency. The comparison results are reported in Table 5. We encrypt a 1500-byte payload as the encrypted traffic and test the inspection latency on Snort ruleset, in which the size of distinct rule patterns is equal to 91. For \textit{SlimBox}, we set \( k = 4 \) and \( MAX = 10 \) which is the worst condition of our scheme from the above description. From Table 5, we can see \textit{BlindBox} and \textit{S4E} presented the least and the most inspection time. The main reason is that \textit{BlindBox} leveraged the lightweight cryptographic primitive and thus had the optimal inspection efficiency. Our schemes are based on OXT and hence perform worse than \textit{BlindBox}. As for the traffic encryption time, we can get the same comparison results as the inspection time. The \textit{BlindBox} performs best due to the fast SKE, and our \textit{SlimBox} also has acceptable performance. In terms of the communication cost, we can see that \textit{BlindBox} and \textit{SlimBox\textsuperscript{*}} will incur the most and the least traffic communication cost, respectively. The reason is that \textit{BlindBox} will consider all distinct rule pattern size in the traffic encryption process.

While \textit{SlimBox\textsuperscript{*}} does not need to send cross tag set to MB compared to \textit{SlimBox\textsuperscript{0}} hence incurs the least bandwidth. Especially, the two-round interaction \textit{SlimBox\textsuperscript{*}} only sends the label tags and the auxiliary tags of the matched label tags to MB in the first and second round, respectively. Thus it can further save almost 90% bandwidth compared to \textit{SlimBox\textsuperscript{*}}.

\section{Conclusion}

In this paper, we design \textit{SlimBox} to achieve lightweight and privacy-preserving middlebox services in cloud computing. The proposed \textit{SlimBox} supports fast filtering and rapidly pattern matching by subtly incorporating the position information into the conjunctive SE scheme. Experiment results demonstrate that \textit{SlimBox} is extremely efficient. As part of our future work, we will try to improve the inspection functionality of \textit{SlimBox} to support enriched patterns, e.g., multi-keyword matching and regular expression evaluation, over encrypted traffic.

\section*{Acknowledgment}

This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFE0201400, and 2020YFB1005804; NSFC grants 62272150, 61872133, U20A20181, and 61802076; the Hunan Provincial Natural Science Foundation of China (Grant No. 2020JJ3015); and the Postgraduate Scientific Research Innovation Project of Hunan Province (No. QL20210095).

\section*{References}


<table>
<thead>
<tr>
<th>Scheme</th>
<th>Traffic time (s)</th>
<th>Communication cost (bytes)</th>
<th>Inspection time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SlimBox</td>
<td>2.65</td>
<td>103293</td>
<td>180ms</td>
</tr>
<tr>
<td>S4E</td>
<td>2.58</td>
<td>199101</td>
<td>158ms</td>
</tr>
<tr>
<td>SlimBox\textsuperscript{0}</td>
<td>2.65</td>
<td>103293</td>
<td>180ms</td>
</tr>
<tr>
<td>SlimBox\textsuperscript{*}</td>
<td>2.58</td>
<td>199101</td>
<td>158ms</td>
</tr>
<tr>
<td>\textit{SlimBox} (two-round)</td>
<td>2.65</td>
<td>103293</td>
<td>180ms</td>
</tr>
</tbody>
</table>

\begin{table}[h]
\centering
\caption{Performance comparison between \textit{BlindBox}, \textit{S4E}, and \textit{SlimBox}.}
\begin{tabular}{|c|c|c|}
\hline
Scheme          & Traffic time (s) & Communication cost (bytes) & Inspection time \\
\hline
BlindBox        & 1.08             & 609630                     & 24ms            \\
S4E             & 31.94            & 376320                     & 1300s           \\
SlimBox\textsuperscript{0} | 2.58             | 199101                    | 158ms           |
SlimBox\textsuperscript{*} | 2.65             | 103293                    | 180ms           |
\textit{SlimBox} (two-round) | 2.65             | 103293                    | 180ms           |
\hline
\end{tabular}
\end{table}
[23] J. Han, S. Kim, J. Ha, and D. Han, “SGX-Box: enabling visibility on encrypted traffic using a secure middlebox module,” in Proc. of APNet, 2017.

Qin Liu received her B.Sc. in Computer Science in 2004 from Hunan Normal University, China, received her M.Sc. in Computer Science in 2007, and received her Ph.D. in Computer Science in 2012 from Central South University, China. She has been a Visiting Student at Temple University, USA. Her research interests include security and privacy issues in cloud computing. Now, she is an Associate Professor in the College of Computer Science and Electronic Engineering at Hunan University, China.

Yu Peng is currently pursuing the Ph.D. degree in the College of Computer Science and Electronic Engineering at Hunan University, China. His research interests include the security and privacy issues in cloud computing and networked applications.

Hongbo Jiang received the PhD degree from Case Western Reserve University, in 2008. After that, he joined the faculty of the Huazhong University of Science and Technology as a full professor and the deputy director of the Department of Communication Engineering. Now, he is a full professor with the College of Computer Science and Electronic Engineering, Hunan University. His research concerns computer networking, especially algorithms and protocols for wireless and mobile networks. He is serving as an editor for the IEEE/ACM Transactions on Mobile Computing, and associate technical editor for the IEEE Communications Magazine.

Jie Wu is the Chair and a Laura H. Carnell Professor in the Department of Computer and Information Sciences at Temple University, Philadelphia, PA, USA. Prior to joining Temple University, he was a Program Director at the National Science Foundation and a Distinguished Professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu has regularly published in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE TRANSACTIONS ON SERVICE COMPUTING and Journal of Parallel and Distributed Computing. Dr. Wu was general co-chair/chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, and ACM MobiHoc 2014, as well as program co-chair for IEEE INFOCOM 2011 and CCFCNCC 2013. He was an IEEE Computer Society Distinguished Visiting Actor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.

Tian Wang received her BSc and MSc degrees in Computer Science from the Central South University in 2004 and 2007, respectively. He received his PhD degree in City University of Hong Kong in 2011. Currently, he is a professor at the Institute of Artificial Intelligence and Future Networks, Beijing Normal University & UIC, China. His research interests include internet of things and edge computing.

Tao Peng received the B.Sc. in Computer Science from Xiangtan University, China, in 2004, the M.Sc. in Circuits and Systems from Hunan Normal University, China, in 2007, and the Ph.D. in Computer Science from Central South University, China, in 2017. Now, she is an Associate Professor of School of Computer Science and Cyber Engineering, Guangzhou University, China. Her research interests include network and information security issues.

Guojun Wang received B.Sc. degree in Geophysics, M.Sc. degree in Computer Science, and Ph.D. degree in Computer Science, at Central South University, China, in 1992, 1996, 2002, respectively. He is a Pearl River Scholarship Distinguished Professor of Higher Education in Guangdong Province, a Doctoral Supervisor of School of Computer Science and Cyber Engineering, Guangzhou University, China, and the Director of Institute of Computer Networks at Guangzhou University. He has been listed in Chinese Most Cited Researchers (Computer Science) by Elsevier in the past eight consecutive years (2014-2021). His research interests include artificial intelligence, big data, cloud computing, Internet of Things (IoT), blockchain, trustworthy/dependable computing, network security, privacy preserving, recommendation systems, and smart cities. He is a Distinguished Member of CCF, a Member of IEEE, ACM and IEICE.